TMS570LS20x/10x Safety MCUs

Microcontroller

Technical Reference Manual (TRM)

TEXAS INSTRUMENTS

Literature Number: SPNU489C
February 2012

Table of Contents

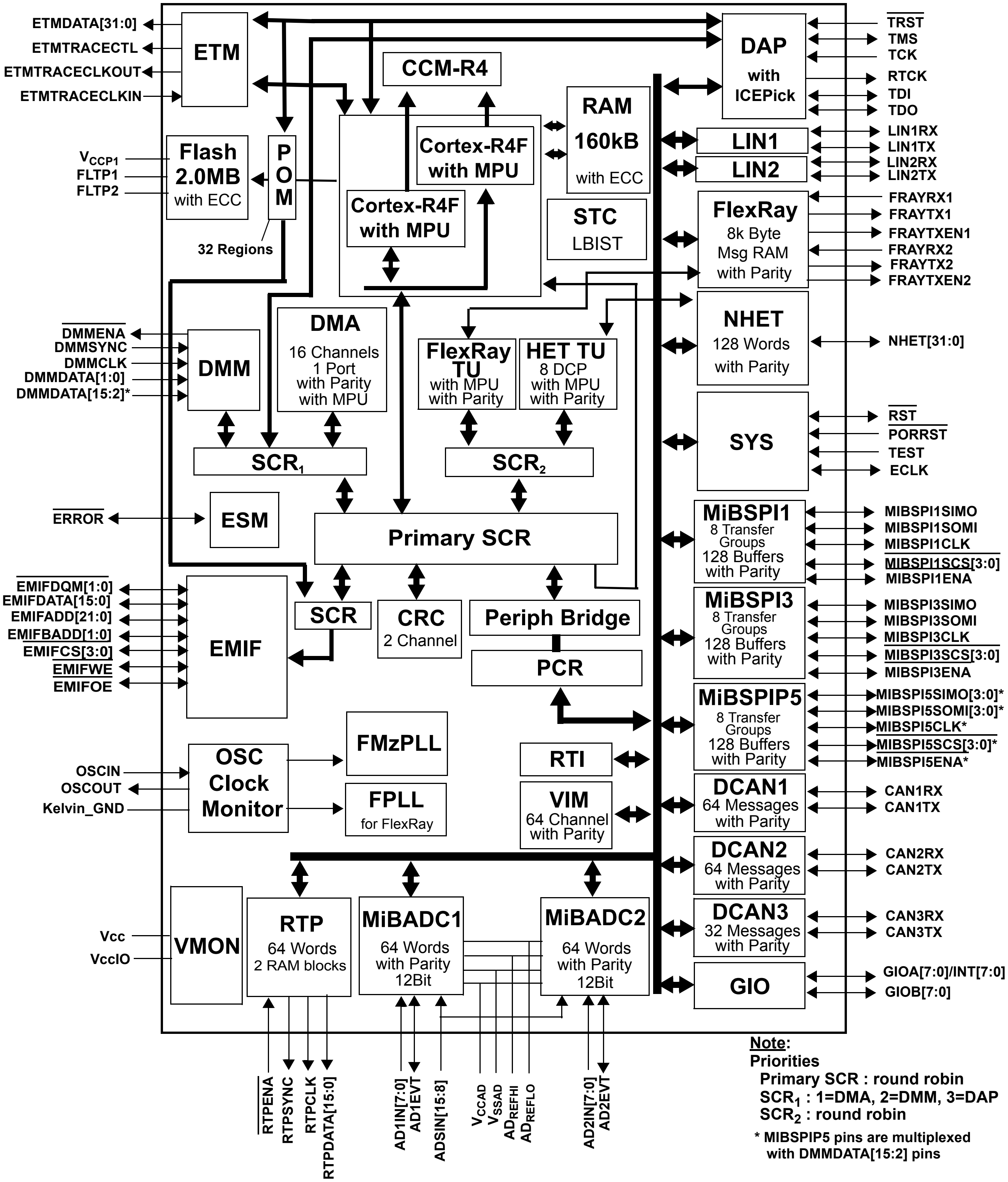
TMS570LS20x/10x Safety MCUs *Architecture Overview*

This TMS570LS20x/10x Safety MCUs are based on the TMS570 platform architecture. This chapter provides an overview of the TMS570 platform architecture and also describes the CPU memory map.

1.1 Functional Block Diagram

Figure 1-1 shows the functional block diagram of the TMS570LS20216 microcontroller.

Figure 1-1. Functional Block Diagram



Note:
Priorities
Primary SCR : round robin
SCR$_1$: 1=DMA, 2=DMM, 3=DAP
SCR$_2$: round robin

*** MIBSPIP5 pins are multiplexed with DMMDATA[15:2] pins**

1.2 Overview

The TMS570LS20x/10x Safety MCUs are based on the TMS570 platform architecture, which defines the interface between the CPUs and the modules. The platform is independent of a CPU and can be used with any ARM CPU. The system bus protocol used for the CPU interconnect varies per CPU as needed to maximize performance and minimize silicon area.

This TMS570LS20x/10x Safety MCUs use the ARM Cortex-R4F CPU, which connects to a Switch Central Resource (SCR) using the AMBA Advanced eXtensible Interface (AXI). More information about the ARM Cortex-R4F CPU and the ETM (Embedded Trace Macrocell) can be obtained from Advanced RISC Machines Limited (ARM).

The SCR (Primary SCR) is responsible for managing the arbitration priority between accesses from multiple masters to each of the slaves. The arbitration priority can be programmed to be either round-robin or fixed priority.

The bus masters on this TMS570LS20x/10x Safety MCUs are – Cortex-R4F CPU, Direct Memory Access (DMA) Controller, High-End Timer Transfer Unit (HET TU), FlexRay Transfer Unit (FTU), Data Modification Module (DMM), and the Debug Access Port (DAP).

The slave modules on this TMS570LS20x/10x Safety MCUs are – Cyclic Redundancy Checker module (CRC), CPU Program memory (flash) and data memory (RAM), the peripheral bridge, the Peripheral Central Resource (PCR) controller, and External Memory Interface module (EMIF).

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, etc. The system module registers also include status flags for indicating exception conditions – resets, aborts, errors, interrupts.

The Error Signalling Module (ESM) handles all the error signals generated by the various modules. It provides the ability to either generate a non-maskable interrupt to the CPU, or an abort based on the severity of the error. The ESM module also has the ability to indicate an error condition on a dedicated device pin.

Interrupt requests generated by different sources are handled by the Vectored Interrupt Manager (VIM) module. The vectored interrupt capability decreases the interrupt latency by providing to the CPU the address of the service routine for the pending interrupt with the highest priority.

The Real-Time Interrupt (RTI) module is specifically designed to support time-triggered operating systems and real-time operating systems.

The Cyclic Redundancy Checker module (CRC) provides two channels to perform background signature verification on any memory sub-system. It also supports maximum-length Parallel Signature Analysis (PSA) based on a 64-bit primitive polynomial.

The non-volatile CPU program memory (flash) is connected to the ATCM interface of the CPU. The CPU data memory (RAM) is connected in an interleaved fashion to the B0 and B1 TCM interfaces of the CPU. The Cortex-R4F provides a slave L2 interface so that the CPU program and data memories can be accessed by the other masters such as the DMA and the Transfer Units.

The device supports external memory. The external memory is driven by the External Memory Interface module (EMIF) allowing different memory configuration and speed.

The main peripherals of the device include several communication modules: Controller Area Network (DCAN) modules, Multibuffer Serial Peripheral Interface (MibSPI), Local Interconnect Network (LIN), and TI FlexRay module (FlexRAY) and the associated Transfer Unit, FlexRay Transfer Unit module (FlexRay TU). Other key modules as shown in the block diagram are the Enhanced High End Timer module (NHET), the HET Transfer Unit (HET TU), Multibuffer Analog to Digital converter (MibADC), General Purpose Input/ Output module (GPIO). Refer to the corresponding chapters for information on these modules.

Note:
Exact hardware configuration features and memory maps can be found in the device datasheet.

1.3 Memory Map

The TMS570LS20216 default memory map is shown in Figure 1-2.

Figure 1-2. Default Memory Map

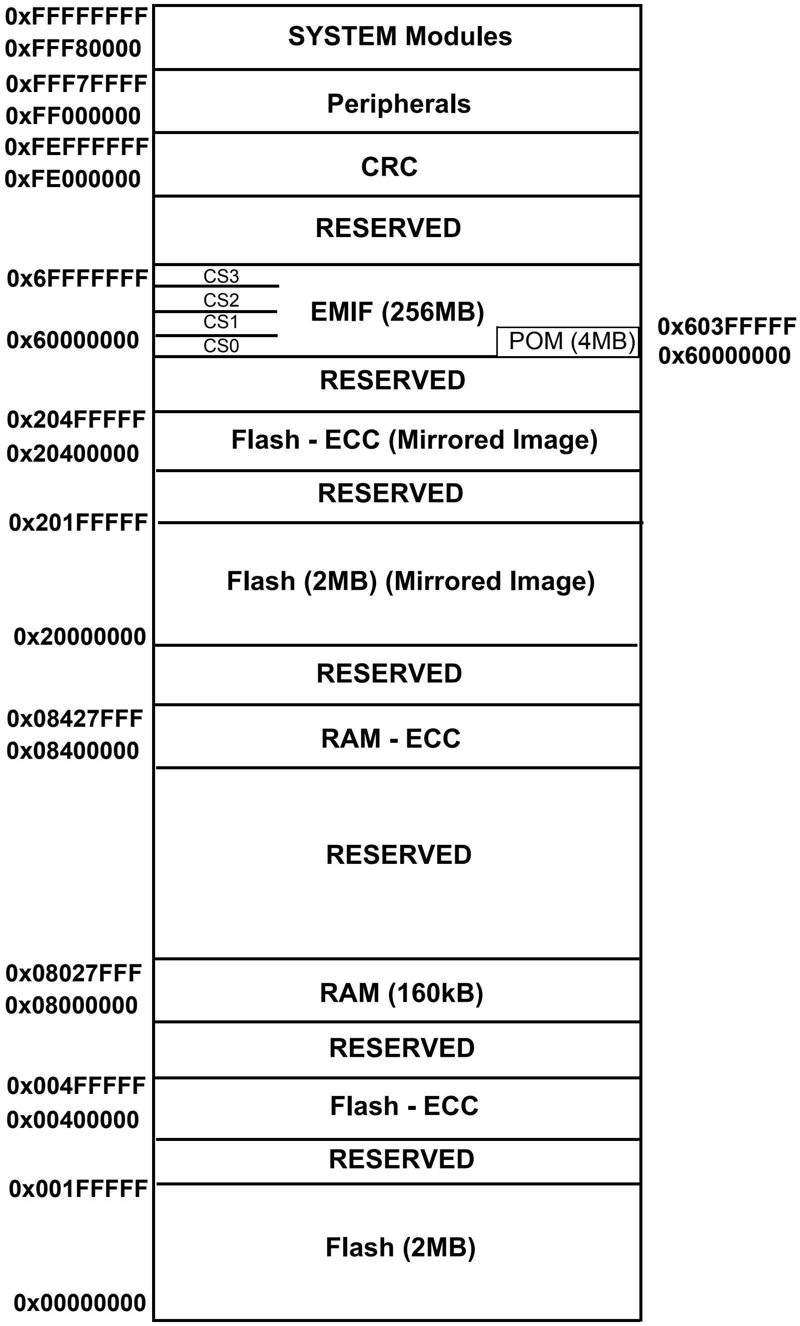


Table 1-1 shows the flash memory configuration in more detail. The 2MB of program memory is comprised of 4 physical flash banks, each bank being 512KB. The flash wrapper module provides a pipelined interface to the flash banks and is capable of providing four 32-bit words to the CPU when the pipelined mode is enabled. The flash wrapper module provides a pipelined interface to the flash banks and is capable of providing four 32-bit words to the CPU when the pipelined mode is enabled. The flash in pipeline mode is capable of accessing 128 bits at a time and provides two 64-bit pipelined words to the CPU. The minimum size for an erase operation is one sector. A single program operation can program either one 32-bit word or one 16-bit half word at a time.)

The flash banks support non-pipelined mode accesses at CPU clock frequencies up to 36 MHz, and pipelined mode accesses at CPU clock frequencies up to 160 MHz.

Table 1-1. Flash Memory Banks and Sectors

SECTOR NO.	SEGMENT	LOW ADDRESS	HIGH ADDRESS	MEMORY ARRAYS (OR BANKS)
Bank 0: 512K Bytes				
0	32K Bytes	0x0000_0000	0x0000_7FFF	BANK0 (512k Bytes)
1	32K Bytes	0x0000_8000	0x0000_FFFF	
2	32K Bytes	0x0001_0000	0x0001_7FFF	
3	8K Bytes	0x0001_8000	0x0001_9FFF	
4	8K Bytes	0x0001_A000	0x0001_BFFF	
5	16K Bytes	0x0001_C000	0x0001_FFFF	
6	64K Bytes	0x0002_0000	0x0002_FFFF	
7	64K Bytes	0x0003_0000	0x0003_FFFF	
8	128K Bytes	0x0004_0000	0x0005_FFFF	
9	128K Bytes	0x0006_0000	0x0007_FFFF	
Bank 1: 512K Bytes				
0	128K Bytes	0x0008_0000	0x0009_FFFF	BANK1 (512k Bytes)
1	128K Bytes	0x000A_0000	0x000B_FFFF	
2	128K Bytes	0x000C_0000	0x000D_FFFF	
3	128K Bytes	0x000E_0000	0x000F_FFFF	
Bank 2: 512K Bytes				
0	128K Bytes	0x0010_0000	0x0011_FFFF	BANK2 (512k Bytes)
1	128K Bytes	0x0012_0000	0x0013_FFFF	
2	128K Bytes	0x0014_0000	0x0015_FFFF	
3	128K Bytes	0x0016_0000	0x0017_FFFF	
Bank 3: 512K Bytes				
0	128K Bytes	0x0018_0000	0x0019_FFFF	BANK3 (512k Bytes)
1	128K Bytes	0x001A_0000	0x001B_FFFF	
2	128K Bytes	0x001C_0000	0x001D_FFFF	
3	128K Bytes	0x001E_0000	0x001F_FFFF	

Table 1-2 shows the memory map for the Cyclic Redundancy Check module (CRC), the Cortex-R4F, CoreSight debug modules and the System modules.

Table 1-2. System Modules Memory Map

FRAME NAME	ADDRESS RANGE	
	FRAME START ADDRESS	FRAME ENDING ADDRESS
CRC	0xFE00_0000	0xFEFF_FFFF
CoreSight Debug ROM Register	0xFFA0_0000	0xFFA0_0FFF
Cortex-R4F Debug Register	0xFFA0_1000	0xFFA0_1FFF
ETM-R4 Register	0xFFA0_2000	0xFFA0_2FFF
CoreSight TPIU Register	0xFFA0_3000	0xFFA0_3FFF
POM Register	0xFFA0_4000	0xFFA0_4FFF
DMA RAM	0xFFF8_0000	0xFFF8_0FFF
VIM RAM	0xFFF8_2000	0xFFF8_2FFF
RTP RAM	0xFFF8_3000	0xFFF8_3FFF
Flash Wrapper Register	0xFFF8_7000	0xFFF8_7FFF
PCR Register	0xFFFF_E000	0xFFFF_E0FF
Secondary System Registers FlexRay PLL/STC CLK Register	0xFFFF_E100	0xFFFF_E1FF
PBIST Register	0xFFFF_E400	0xFFFF_E5FF
STC Register	0xFFFF_E600	0xFFFF_E6FF
EMIF Register	0xFFFF_E800	0xFFFF_E8FF
DMA Register	0xFFFF_F000	0xFFFF_F3FF
ESM Register	0xFFFF_F500	0xFFFF_F5FF
CCMR4 Register	0xFFFF_F600	0xFFFF_F6FF
DMM Register	0xFFFF_F700	0xFFFF_F7FF
RAM ECC even Register	0xFFFF_F800	0xFFFF_F8FF
RAM ECC odd Register	0xFFFF_F900	0xFFFF_F9FF
RTP Register	0xFFFF_FA00	0xFFFF_FAFF
RTI Register	0xFFFF_FC00	0xFFFF_FCFF
VIM Parity Register	0xFFFF_FD00	0xFFFF_FDFF
VIM Register	0xFFFF_FE00	0xFFFF_FEFF
Primary System Register	0xFFFF_FF00	0xFFFF_FFFF

The peripheral frame contains the memory map for the peripheral registers as well as the peripheral memories. The architecture supports up to 32 peripheral registers' frames of 1KB each, and up to 64 peripheral memories' frame of up to 128KB each Table 1-3 shows the memory map for the peripheral memories while Table 1-4 shows the memory map for the peripheral module registers.

Table 1-3. Peripheral Memories Map

PERIPHERAL MODULE MEMORY	ADDRESS RANGE		PERIPHERAL SELECTS
	BASE ADDRESS	ENDING ADDRESS	
MIBSPIP5 RAM	0xFF0A_0000	0xFF0B_FFFF	PCS[5]
MIBSPI3 RAM	0xFF0C_0000	0xFF0D_FFFF	PCS[6]
MIBSPI1 RAM	0xFF0E_0000	0xFF0F_FFFF	PCS[7]
DCAN3 RAM	0xFF1A_0000	0xFF1B_FFFF	PCS[13]

DCAN2 RAM	0xFF1C_0000	0xFF1D_FFFF	PCS[14]
DCAN1 RAM	0xFF1E_0000	0xFF1F_FFFF	PCS[15]
MIBADC2 RAM	0xFF3A_0000	0xFF3B_FFFF	PCS[29]
MIBADC1 RAM	0xFF3E_0000	0xFF3F_FFFF	PCS[31]
NHET RAM	0xFF46_0000	0xFF47_FFFF	PCS[35]
HET TU RAM	0xFF4E_0000	0xFF4F_FFFF	PCS[39]
FlexRay TU RAM	0xFF50_0000	0xFF51_FFFF	PCS[40]

Table 1-4. Peripheral Registers Memory Map

PERIPHERAL MODULE	ADDRESS RANGE		PERIPHERAL SELECTS
	BASE ADDRESS	ENDING ADDRESS	
MIBSPIP5	0xFFF7_FC00	0xFFF7_FDFF	PS[0]
MIBSPI3	0xFFF7_F800	0xFFF7_F9FF	PS[1]
MIBSPI1	0xFFF7_F400	0xFFF7_F5FF	PS[2]
LIN2	0xFFF7_E500	0xFFF7_E5FF	PS[6]
LIN1	0xFFF7_E400	0xFFF7_E4FF	
DCAN3	0xFFF7_E000	0xFFF7_E1FF	PS[7]
DCAN2	0xFFF7_DE00	0xFFF7_DFFF	PS[8]
DCAN1	0xFFF7_DC00	0xFFF7_DDFF	
FlexRay	0xFFF7_C800	0xFFF7_CFFF	PS[12]+PS[13]
MIBADC2	0xFFF7_C200	0xFFF7_C3FF	PS[15]
MIBADC1	0xFFF7_C000	0xFFF7_C1FF	
GIO	0xFFF7_BC00	0xFFF7_BCFF	PS[16]
NHET	0xFFF7_B800	0xFFF7_B8FF	PS[17]
HET TU	0xFFF7_A400	0xFFF7_A4FF	PS[22]
FlexRay TU	0xFFF7_A000	0xFFF7_A1FF	PS[23]

The FlexRay transfer unit peripheral select 23 must be enabled before accessing a FlexRay register within the peripheral selects 12 or 13.

1.3.1 Boot Memory Selection

At boot up, the CPU starts fetching instructions starting at address 0x0000 0000. By default, this memory location is mapped to the internal flash program memory as shown in the default memory map in Figure 1-2. The device offers the capability to swap the internal flash and internal CPU data RAM memory maps. This is especially useful for debugging portions of the application code that can fit into the available CPU data RAM.

Table 1-5. Memory Map Swap Control Registers

0xC4 BMMCR1 Page 128	Reserved	
	Reserved	MEM SW
0xCC MMUGCR Page 131	Reserved	
	Reserved	CPU RESET

The Flash and RAM are swapped by first programming the MEM SW field of BMMCR1 register to 0x5 and then resetting the Cortex-R4F CPU by changing the setting of the CPU RESET bit in the MMUGCR register. A CPU reset is triggered when either the CPU RESET bit is changed from a 0 to a 1 or from a 1 to a 0. Figures 1 and 2 show the default (no swapping) and swapped memory mappings, respectively.

Figure 1. Memory Map before swapping RAM and Flash

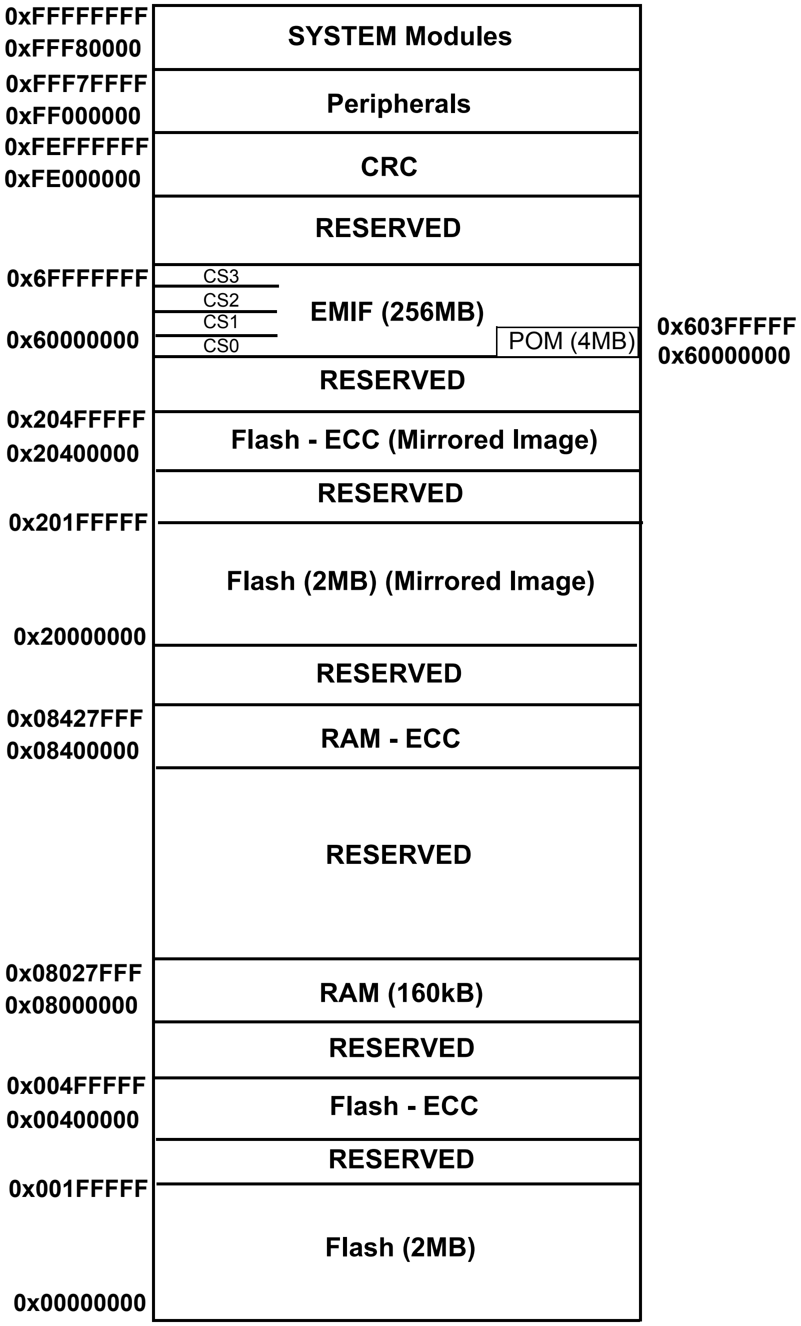
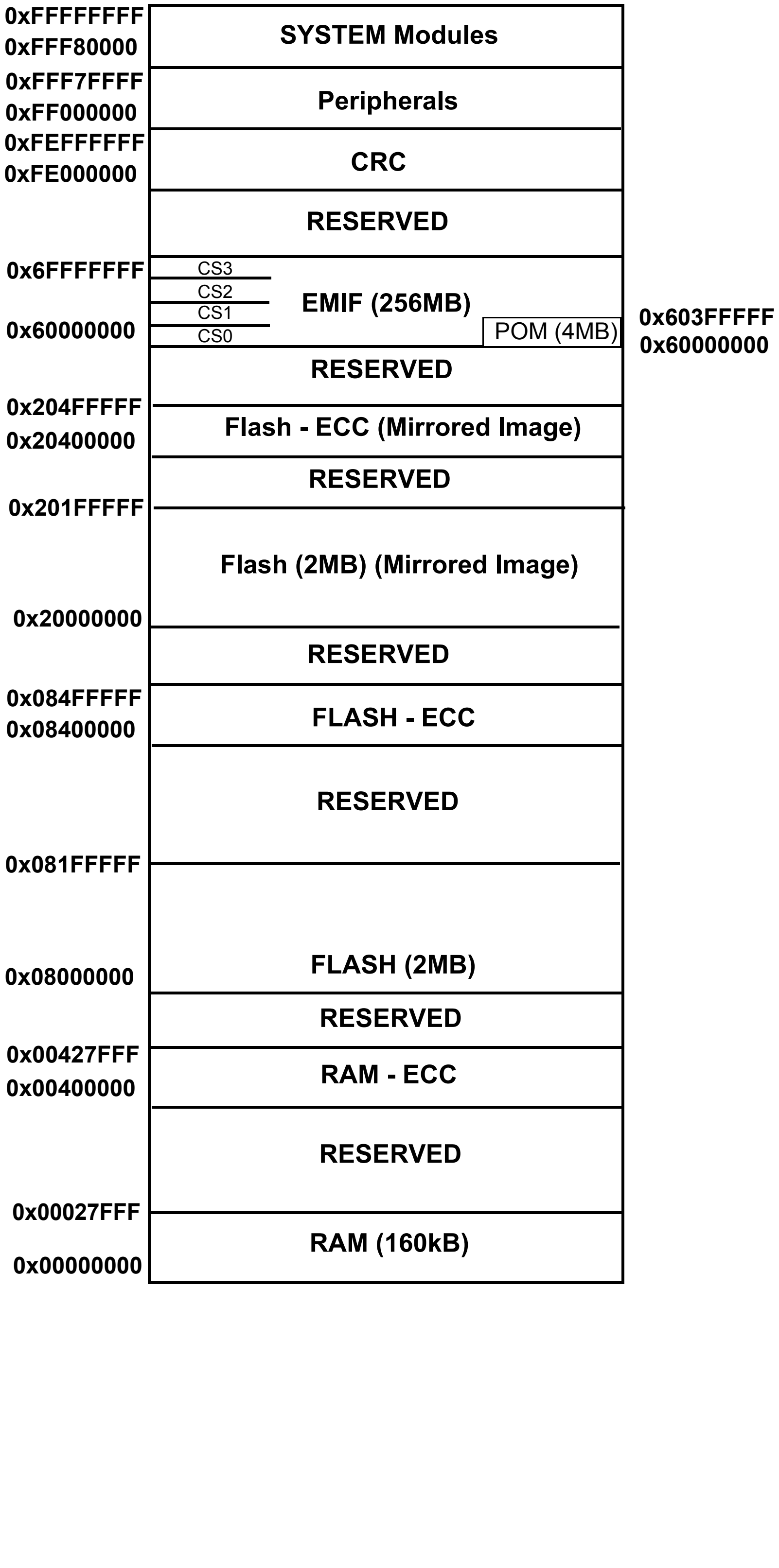


Figure 2. Memory Map after swapping RAM and Flash



1.4 Endianness

The TMS570LS20x/10x Safety MCUs use a word invariant big endian (BE-32) programmer's model. Table 1-6 shows which byte(s) of data are selected with varying offset address and transfer size.

Table 1-6. Active Byte Lanes for a 32-Bit Data Bus

Transfer size	Offset Address	DATA 31:24	DATA 23:16	DATA 15:8	DATA 7:0
Word	0	✓	✓	✓	✓
Half-Word	0	✓	✓		
Half-Word	2			✓	✓
byte	0	✓			
byte	1		✓		
byte	2			✓	
byte	3				✓

1.5 Memory Module Hardware Initialization

The TMS570LS20x/10x Safety MCUs system provide the capability to perform a hardware initialization on most memories on the system bus and on the peripheral bus. The memory used for the FlexRay message objects is not directly CPU addressable, hence there is no memory auto-initialization support for this memory.

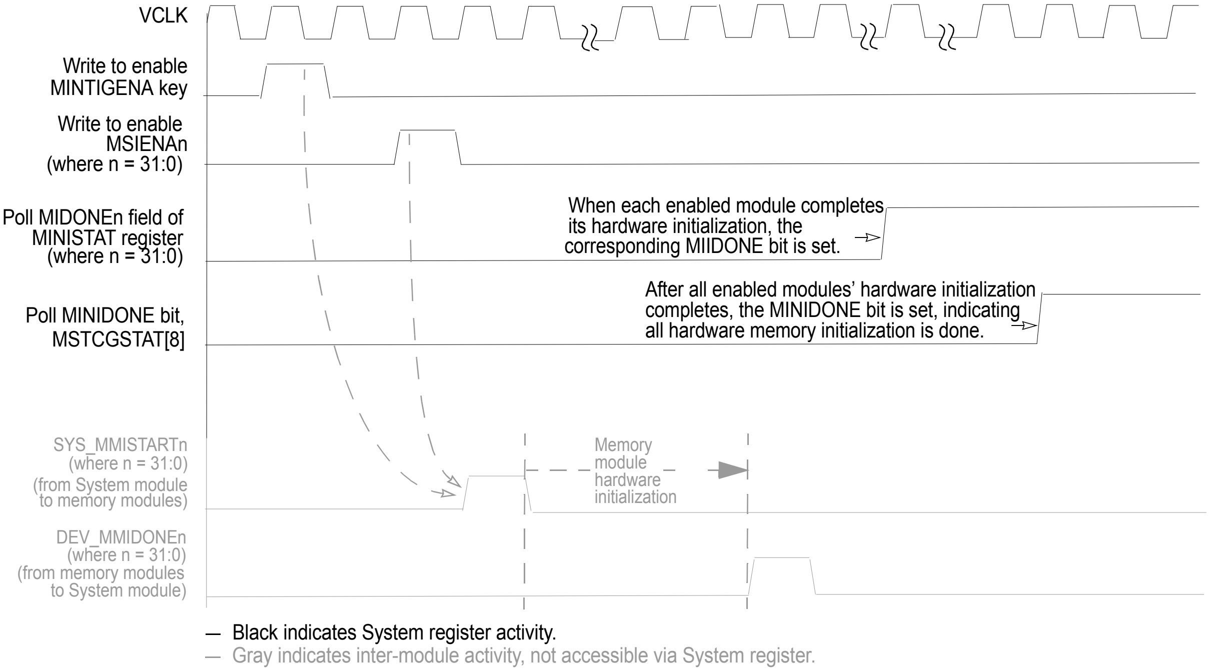
The intent of having the hardware initialization is to program the memory arrays with error detection capability to a known state based on their error detection scheme – odd/even parity or ECC. For example, the contents of the CPU data RAM after power-on reset is unknown. A hardware auto-initialization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

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

1.5.1 Memory Module Hardware Initialization Features

- Can run all the memory modules' initialization in parallel or individually.
- Captures memory module initialization completion results in status registers.

Figure 1-3. Hardware Memory Initialization Protocol


VCLK
Write to enable MINTIGENA key
Write to enable MSIENAn (where n = 31:0)
Poll MIDONEn field of MINISTAT register (where n = 31:0)
When each enabled module completes its hardware initialization, the corresponding MIIDONE bit is set.
Poll MINIDONE bit, MSTCGSTAT[8]
After all enabled modules' hardware initialization completes, the MINIDONE bit is set, indicating all hardware memory initialization is done.
SYS_MMISTARTn (where n = 31:0) (from System module to memory modules)
Memory module hardware initialization
DEV_MMIDONEn (where n = 31:0) (from memory modules to System module)
— Black indicates System register activity.
— Gray indicates inter-module activity, not accessible via System register.

The hardware memory initialization protocol is as follows:

1. Enable the global hardware memory initialization key by programming a value of 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 MSIENA(31–0) bits in the MSIENA register.
3. If the global hardware is enabled, on detecting a write of a value 1 to the MSIENAn bit of the MSIENA register, a pulse of 1 VCLK is generated on the corresponding MMISTART(31–0) signal, which triggers a

hardware initialization on the corresponding module. The mapping of the MSIENA bit to corresponding module is device specific and can be found in the device data sheet.

4. On detection of a pulse of 1 VCLK on its MMISTART (memory module initialization) input signal, the corresponding module will initialize its memories based on its memory error checking scheme (even parity or odd parity or ECC).
5. When the memory initialization is complete, the module will generate a pulse of 1 VCLK on its MMIDONE (memory module initialization done) output signal, which sets the corresponding bit in the system module MIDONE field of the MINISTAT register to indicate the completion of its memory initialization.
6. 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.

Clocks on TMS570LS20x/10x Safety MCUs

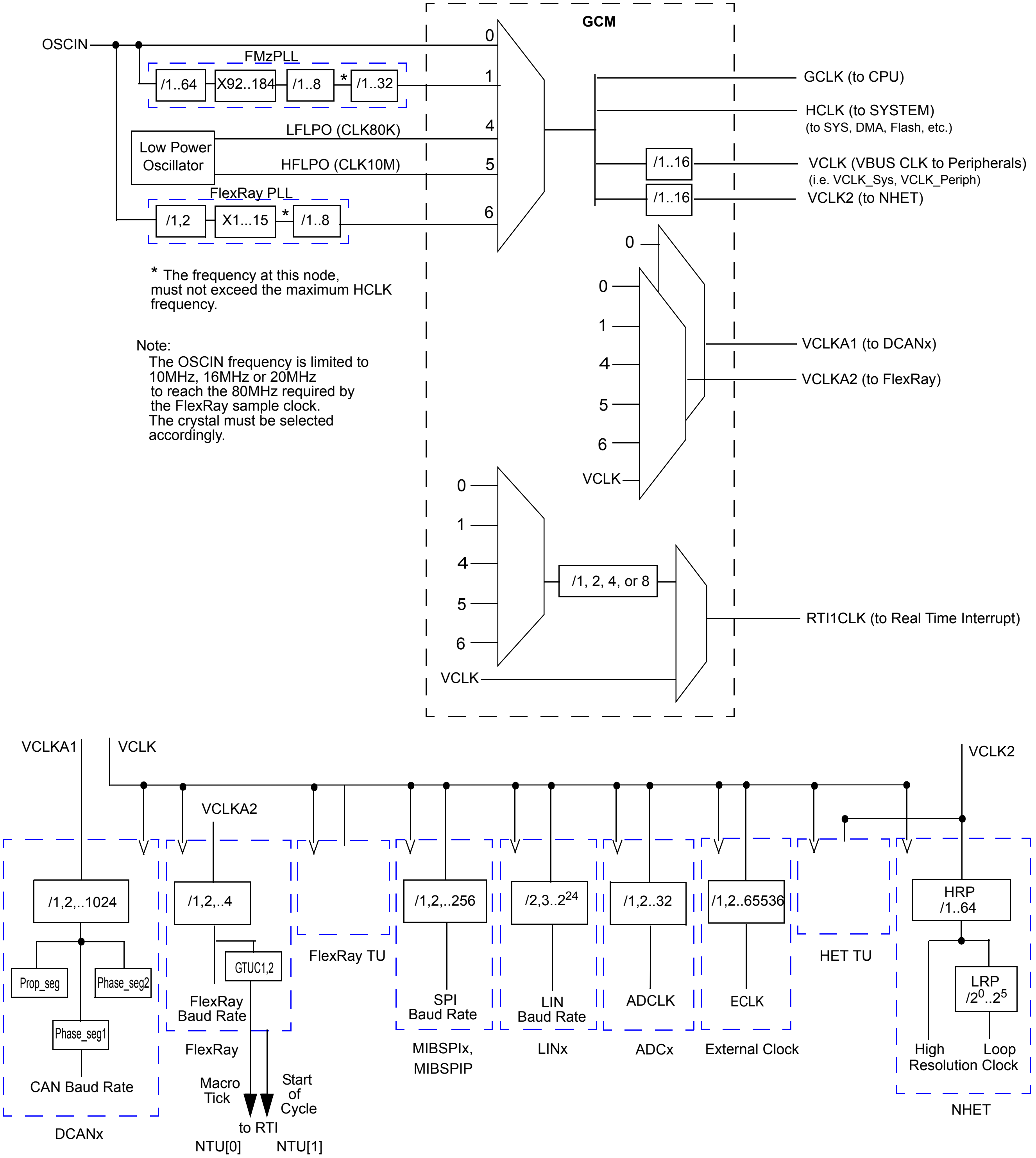
This chapter describes the clock sources and clock domains on the TMS570LS20x/10x Safety MCUs .

Topic Page

2.1 Clocks on TMS570LS20x/10x Safety MCUs

Figure 2-1 shows a block diagram of the clocks on the TMS570LS20x/10x Safety MCUs. As shown in the diagram, there are five clock sources and seven clock domains. The Global Clock control Module (GCM) manages these clock sources and clock domains.

Figure 2-1. TMS570LS20x/10x Safety MCUs Device Clocks



2.2 Clock Sources

Table 2-1 describes the clock sources. Refer to the device data sheet for electrical specifications and AC characteristics of these clock sources.

Table 2-1. TMS570LS20x/10x Safety MCUs Clock Sources Description

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	FMzPLL	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	Not implemented.	No clock source is available
3	Not implemented.	No clock source is available
4	LFLPO (Low Frequency LPO)	This is the low-frequency output of the internal reference oscillator. This is typically an 80 KHz signal which is used by the real-time interrupt module for generating periodic interrupts to wake up from low power mode.
5	HFLPO (High Frequency LPO)	This is the high-frequency output of the internal reference oscillator. This is typically a 10 MHz signal which is used by the clock monitor module as a reference clock to monitor the main oscillator frequency.
6	FPLL	This is the output of the PLL dedicated to generate a clock source for FlexRay. The generated clock source is non-modulated because the FlexRay protocol places severe jitter constraints on the clock used for generating the baud rate. A separate non-modulating PLL allows the generation of an asynchronous clock source that is independent of the CPU clock frequency.
7	Not implemented.	No clock source is available

Note: Unimplemented clock sources 2, 3, and 7 should not be enabled or used.

2.3 Clock Domains

Table 2-2 describes the various clock domains. Refer to the device data sheet for the maximum frequency constraints on each of these clock domains.

Table 2-2. TMS570LS20x/10x Safety MCUs Clock Domains Description

Domain	Clocked By	Generated By	Comments
CPU Clock Domain	GCLK	Any clock source. By default, GCLK is clocked by main oscillator, OSCIN.	GCLK controls all the CPU subsystems, including the Memory Protection Unit (MPU).
System bus clock domain	HCLK	Always the same clock source as GCLK, although HCLK could be active while GCLK is held static.	The ratio between GCLK and HCLK is always 1, i.e.. GCLK and HCLK frequencies are the same.
System Peripheral Clock Domain	VCLK_sys	Derived from HCLK	VCLK_sys is the primary clock for the VBUS clock domain. It is used for all the bus transactions as well as the system peripherals (system module, VIM, etc.), with the exception of the real-time interrupt (RTI) module. The ratio between HCLK and VCLK_sys is programmable from 1 to 16 via CLKCNTL[19:16], the VBUS Clock Ratio field (VCLKR) of the Clock Control register (CLKCNTL). The default HCLK:VCLK_sys ratio is 1:2, i.e.. the VCLK_sys frequency is HCLK divided by 2. HCLK and VCLK_sys share the same enable and disable control bits.
Peripheral Clock Domains	VCLK_periph and VCLK2	Derived from HCLK	VCLK_periph is the primary clock for the peripherals. VCLK_periph is synchronous with VCLK_sys (ratio 1:1), but it can be shut down separately. VCLK2 is a second VBUS clock domain that is used by the NHET and HET TU peripherals to allow it to run faster than VCLK_periph. The ratio between HCLK and VCLK2 is programmable from 1 to 16 via CLKCNTL[27:24], the VBUS clock2 ratio field (VCLKR2) of the Clock Control register (CLKCNTL). The default HCLK:VCLK2 ratio is 1:2, i.e.. the VCLK2 frequency is HCLK divided by 2. The frequency relationship between VCLK2, HCLK, and VCLK_sys is as follows: HCLK $\geq$ VCLK2 $\geq$ VCLK_sys The GCM only supports integer ratios between VCLK_sys, VCLK2, and HCLK.
Primary and Secondary Asynchronous Clock Domains	VCLKA1 and VCLKA2	Any clock source	The asynchronous clock domains are used specifically for communication modules with strict tolerance constraints that do not allow the use of frequency-modulated clocks, for example, DCAN and FlexRay.
Real-Time Interrupt Clock Domains	RTI1CLK	OSCIN or VCLK. Other low jitter sources could be used.	By default, RTI1CLK uses the same ratio as VCLK_sys, that is VCLK_sys frequency is HCLK divided by 2. The RTI1CLK switches to OSCIN under the following conditions: – The RTI timebase is set to OSCIN by the software – The device enters stand-by mode – The PLL is shut down

2.4 Mapping Clock Sources and Clock Domains

Table 2-3 summarizes the control registers that allow mapping of the clock domains to one of the available clock sources.

Table 2-3. Clock Source Selection for Clock Domains

0x48 GHVSRC Page 95	Reserved	GHVWAKE(3–0)	Reserved	HVLPM(3–0)
	Reserved			GHVSRC(3–0)
0x4C VCLKASRC Page 97	Reserved			
	Reserved	VCLKA2SRC(3–0)	Reserved	VCLKA1SRC(3–0)
0x50 RCLKSRC Page 99	Reserved			
	Reserved	RTI1DIV(1–0)	Reserved	RTI1SRC(3–0)

2.4.1 Mapping GCLK, HCLK and VCLK

The GCLK, HCLK and VCLK_sys / VCLK_periph / VCLK2 domains use the same clock source, as described in Table 2-2. This clock source is selected via the GHVSRC field of the GCLK/HCLK/VCLK Source (GHVSRC) register. The default clock source for these clock domains is the main oscillator, OSCIN. The GCM module also allows the option to select an alternate clock source under two conditions:

- When just the CPU clock is disabled via the Clock Domain Disable (CDDIS) register, the HCLK and the VCLK clock domains use the clock source selected via the HVLPM field of the GHVSRC register. By default, the main oscillator is also selected as the HVLPM source.
- When the device enters a low power mode in which the GCLK, HCLK and VCLK clock domains are disabled, the clock source for these domains upon wake up is selected via the GHVWAKE field of the GHVSRC register. By default, the main oscillator is also selected as the clock source for the GCLK, HCLK and VCLK domains after wake up.

2.4.2 Mapping VCLKA1 and VCLKA2

On the TMS570LS20x/10x Safety MCUs the VCLKA1 domain is used as the clock for generating the DCAN bit timings, while the VCLKA2 domain is used as the clock for generating the FlexRay timings. The DCAN and FlexRay protocols have very strict requirements in terms of the permissible jitter on the clock used for the timing generation.

The VCLKA1 and VCLKA2 domains are allowed to be mapped to any available clock source independent of the clock source used for the GCLK, HCLK and VCLK domains. The clock source for the VCLKA1 domain is selected via the VCLKA1SRC field of the VCLKASRC register, and the clock source for the VCLKA2 domain is selected via the VCLKA2SRC field of the VCLKASRC register.

By default, the VCLKA1 and VCLKA2 are mapped to VCLK as the source.

2.4.3 Mapping RTI1CLK

The RTI1CLK domain is used to generate the time base for the real-time interrupt (RTI) module. The source for the RTI1CLK can be selected via the RTI1SRC field of the RCLKSRC register. By default, the RTI1CLK domain is mapped to VCLK as the source.

Note: Changing the mapping of RTI1CLK

When the RTI1CLK domain is mapped to a clock source other than VCLK, the frequency of the clock input to the RTI module must be at least 3 times slower than the VCLK frequency. This can be managed by configuring the RTI1DIV field of the RCLKSRC register. The RTI1CLK divider only applies when the RTI1CLK is mapped to a clock source other than VCLK.

2.5 Enabling and Disabling Clock Sources and Clock Domains

Table 2-4 summarizes the control registers that allow each clock source and clock domain to be disabled and/or enabled as required by the application.

Table 2-4. Registers for Enabling / Disabling Clock Sources and Clock Domains

0x30 CSDIS Page 86	Reserved							
	Reserved	CLK SR6 OFF	CLK SR5 OFF	CLK SR4 OFF	Reserved		CLK SR1 OFF	CLK SR0 OFF
0x34 CSDISSET Page 87	Reserved							
	Reserved	CLK SR6 OFF SET	CLK SR5 OFF SET	CLK SR4 OFF SET	Reserved		CLK SR1 OFF SET	CLK SR0 OFF SET
0x38 CSDISCLR Page 88	Reserved							
	Reserved	CLK SR6 OFF CLR	CLK SR5 OFF CLR	CLK SR4 OFF CLR	Reserved		CLK SR1 OFF CLR	CLK SR0 OFF CLR
0x3C CDDIS Page 89	Reserved							
	Reserved	RTI1 CLK OFF	VCLKA2 OFF	VCLKA1 OFF	VCLK2 OFF	VCLKP OFF	HCLK OFF	GCLK OFF
0x40 CDDISSET Page 91	Reserved							
	Reserved	RTI1 CLK OFF SET	VCLKA2 OFF SET	VCLKA1 OFF SET	VCLK2 OFF SET	VCLKP OFF SET	HCLK OFF SET	GCLK OFF SET
0x44 CDDISCLR Page 93	Reserved							
	Reserved	RTI1 CLK OFF CLR	VCLKA2 OFF CLR	VCLKA1 OFF CLR	VCLK2 OFF CLR	VCLKP OFF CLR	HCLK OFF CLR	GCLK ENA CLR
0x54 CSVSTAT Page 101	Reserved							
	Reserved	CLK SR6V	CLK SR5V	CLK SR4V	Reserved		CLK SR1V	CLK SR0V

Upon reset, the clock sources 0, 4 and 5 are already enabled. These are the main oscillator, OSCIN, and the two outputs of the internal reference oscillator, LFLPO (CLK80K) and HFLPO (CLK10M) respectively. The other two available clock sources 1, and 6, the main PLL output and the FlexRay PLL output, are disabled upon reset. The status of the available clock sources is always reflected in the Clock Source Valid Status (CSVSTAT) register.

A clock source can be enabled by clearing the corresponding bit in the Clock Source Disable (CSDIS) register. Conversely, a clock source can be disabled by setting the corresponding bit. Separate registers with set and clear functionalities are provided so that the application can enable or disable the desired clock sources without having to perform a read-modify-write operation.

All the clock domains are enabled upon reset. A clock domain can be disabled by setting the corresponding bit in the Clock Domain Disable (CDDIS) register. Conversely, a clock domain can be enabled by clearing the corresponding bit. Separate registers with set and clear functionalities are provided so that the application can enable or disable the desired clock domains without having to perform a read-modify-write operation.

2.5.1 Clock Source and Domain Disabling Mechanism

The GCM module incorporates a hardware handshake mechanism in order to disable a clock source as well as a clock domain.

A clock domain disable request is generated when the application sets the corresponding bit in the Clock Domain Disable (CDDIS) register or the Clock Domain Disable Set (CDDISSET) register. This request to turn off the clock domain is sent to all targets on that clock domain. The GCM waits for all these targets to approve the clock domain disable request, and then disables the clock domain output from the GCM.

A clock source disable request is generated when the application sets the corresponding bit in the Clock Source Disable (CSDIS) register or the Clock Source Disable Set (CSDISSET) register. The GCM only disables the affected clock source once all clock domains using this clock source are disabled.

2.5.2 Clock Source and Domain Enabling Sequence

All disabled clock sources and clock domains are re-enabled upon a system reset condition. The TMS570LS20x/10x Safety MCUs also support specific “wake up” signals that can be used to re-enable any disabled clock sources and clock domains from a device low power mode. The wake up conditions supported are dependent on the actual low power mode entered, and are described in section 2.6, *Low Power Modes on TMS570LS20x/10x Safety MCUs*.

All clock domains are automatically enabled upon detecting a wake up condition. The GCM only enables clock sources that are chosen as clock source for at least one of the clock domains after wake up.

2.6 Low Power Modes on TMS570LS20x/10x Safety MCUs

The TMS570LS20x/10x Safety MCUs support the following low power modes:

2.6.1 Doze Mode

Doze mode is defined as follows:

- The primary oscillator is enabled.
- The RTI clock can be either enabled or disabled (depends on user setting).
- The low-frequency output of the internal reference oscillator can be either enabled or disabled by the user.

Since the primary oscillator is still enabled in doze mode, the device can wake up much faster and start executing instructions in the fewest amount of cycles of any low power mode. Doze mode will also have the highest current of any of the low power modes.

In order to enter doze mode, the following steps are required by the user software.

1. **Software writes to the flash control registers to put all flash pumps to sleep.** Flash pumps will not be disabled until all flash banks have been put to sleep. This is done by programming the fall back modes for the flash banks and pump.
2. **Software writes to the clock source disable register (CSDIS) to disable the clock sources that are not needed.** The main oscillator must still be enabled for the device to be in doze mode.

> **Note: Timing of actual disabling of clock sources**
> The selected clock sources will not actually be disabled until the clock domains that use them are disabled.

3. **Software writes to the clock domain disable register (CDDIS) to disable the GCLK (CPU clock), HCLK (system clock), VCLKP (peripheral VBUS clock), VCLK2 (peripheral VBUS clock2), VCLKA1 (asynchronous peripheral VBUS clock1), and VCLKA2 (asynchronous peripheral VBUS clock2).** All these domains must be disabled in order for the device to be in doze mode. A particular domain is not actually disabled until all modules using that domain assert their clockstop acknowledge signal. The RTI1CLK domain may or may not be disabled. Doze mode is typically used in conjunction with an RTI in order to wake up the device periodically from within.
4. **Software "idles" the ARM core, then stops the core clock.**

Sleep mode supports the following wakeup (return to normal operation) methods:

- Any asynchronous wakeup interrupt
  - RTI interrupt
  - An external GIO Interrupt
  - CAN message
  - SCI or LIN message
- Power-on reset or system reset

The clock source for each clock domain upon wake up is user programmable. For wake up from doze mode, typically the high-frequency oscillator (CLK10M) is used as the clock source after wake up for the GCLK, HCLK, and VCLK domains to minimize the time to start executing instructions.

If the PLL is used as the clock source for the GCLK, HCLK and VCLK after wake up, a longer delay will occur since the PLL needs to restart and re-lock. The PLL can be re-locked after waking up from doze mode in parallel with performing other operations. Once the PLL locks, the software can select the PLL as the source for the clock domains.

2.6.2 Snooze Mode

Snooze mode is defined as follows:

- The primary oscillator is disabled.
- The low-frequency output of the internal reference oscillator is enabled.
- The RTI clock can be either enabled or disabled (depending on application use case).

Since the primary oscillator is disabled in snooze mode, the current consumption will be less than in doze mode. It will, however, take longer to resume normal operation because the high-frequency oscillator must first be restarted. The device will start much faster by using the high-frequency output of the internal reference oscillator (HFLPO) as the clock source upon wake up.

The steps to enter snooze mode are the same as those for doze mode, except that in snooze mode the high-frequency oscillator is also disabled.

Sleep mode supports the following wakeup (return to normal operation) methods:

- Any asynchronous wakeup interrupt
  - RTI interrupt
  - An external GIO Interrupt
  - CAN message
  - SCI or LIN message
- Power-on reset or system reset

The clock source for each clock domain upon wake up is user programmable. For wake up from snooze mode, typically the high-frequency output of the internal reference oscillator is used as the clock source after wake up for the GCLK, HCLK, and VCLK domains to minimize the time to start executing instructions.

The application can then restart the main oscillator and subsequently the PLL to return to full speed operation.

2.6.3 Sleep Mode

In sleep mode all clock sources are disabled as shown below:

- The PLL is disabled.
- The primary oscillator is disabled.
- The LPO clock is disabled.

All clock domains are disabled, including the RTI1CLK. Since all clocks are disabled in sleep mode the current consumption of the device is limited to normal leakage currents.

The steps to enter sleep mode are the same as those for snooze mode, except that in sleep mode all clock sources must be disabled and all clock domains must be disabled. If any clock is still running the device is not in sleep mode.

Sleep mode supports the following wakeup (return to normal operation) methods:

- The following asynchronous wakeup interrupt
  - An external GIO Interrupt
  - CAN message
  - SCI or LIN message
- Power-on reset or system reset

The clock source to be used upon wake up is user programmable. Typically, the main oscillator or the high-frequency output of the internal reference oscillator is used as the wake up clock source for the GCLK, HCLK and VCLK. The selected clock source will start up before the clocks are released internally.

If the PLL is used after returning to normal operation, a longer delay may occur since the PLL needs to re-lock. The PLL can be re-locked after waking up from sleep mode in parallel with performing other operations. Once the PLL locks, the software can select the PLL as the source for the clock domains.

2.6.4 Cortex-R4F Idle Mechanism

The ARM Cortex-R4F has internal power management logic. The R4F has a dedicated instruction which may be used to enter low power modes: Wait for Interrupt (WFI).

When a WFI instruction is executed, the core flushes its pipeline, flushes all write buffers, cleans and invalidates all caches, and completes all pending bus transactions on both the L1 and L2 memory interfaces. At this time the core indicates to the system that it may stop the core clock (CLKIN) and free running core clock (FREECLKIN). This is signalled via the STANDBYWFI signal. The STANDBYWFI signal is used by the clock controller module as a clockstop acknowledge for purposes of externally stopping core clock generation.

Upon receipt of a wakeup request, the GCM will start both FREECLKIN and CLKIN to the R4F core. Code execution will not restart until receipt of an interrupt. Code execution will restart in the appropriate interrupt handler.

A Cortex R4F CPU may only be awakened by an interrupt or CPU reset.

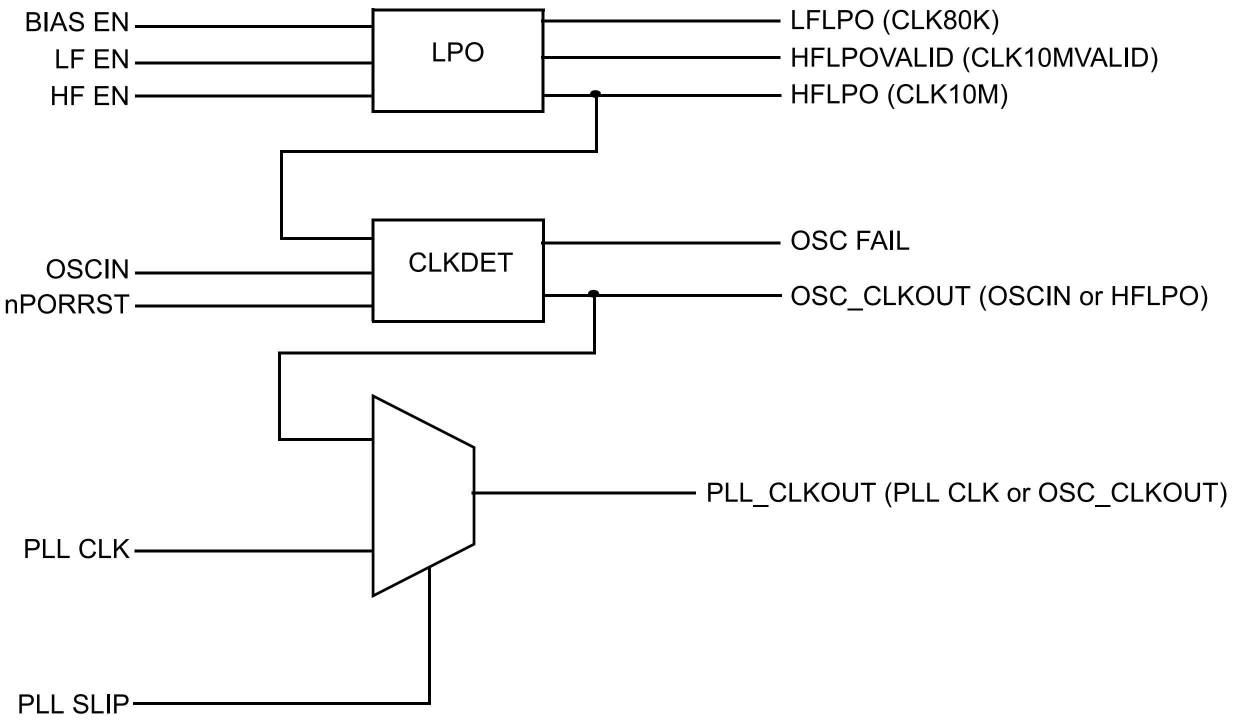
2.7 Clock Monitoring

The TMS570LS20x/10x Safety MCUs incorporate special functions to monitor the main oscillator clock as well as the primary PLL output clock. These monitoring mechanisms are described in the following sections.

2.7.1 Clock Monitor Block Diagram

Figure 2-2 shows a high-level block diagram for the Low Power Oscillator/Clock monitor module.

Figure 2-2. Low Power Oscillator/Clock monitor module block diagram



The LPO provides two different clock outputs: LFLPO and HFLPO. The typical frequencies of these outputs are 80KHz and 10MHz, respectively. Please refer to the TMS570LS20x/10x Safety MCUs data sheet for the minimum and maximum frequencies for these clock sources.

The CLKDET is the actual monitor for the oscillator input clock (OSCIN). It also provides a mechanism to respond to a PLL slip condition indicated by the FMzPLL module.

2.7.2 Oscillator Fail Detection and Response

The TMS570LS20x/10x Safety MCUs have a clock monitoring macro that uses an internal reference clock generator (LPO) to check whether the main oscillator frequency is within a certain range.

This oscillator fail detector is a gross failure detection mechanism intended to monitor the oscillator input pin. It determines whether the external reference input — either the crystal or the oscillator — is stuck low, stuck high, or running at a frequency too slow or too fast.

2.7.2.1 Oscillator Stuck Low/High Detection

The CLKDET module signals an oscillator fault (OSC FAIL) whenever there is no edge detected on the OSCIN signal within a fixed number of CLK10M cycles.

Note: No oscillator fault detection occurs if the device is powered up with the oscillator stuck at a high or low state

The main oscillator is used as the default clock source. A 1024 cycle counter is used to count OSCIN cycles before the CPU is released from reset. This system reset release is used to enable the CLK10M clock source in the LPO. Therefore, if the device is powered up with no oscillator input (stuck high or low), then the device will be in a persistent reset state.

2.7.2.2 Oscillator Frequency Range Detection

The CLKDET module also signals an oscillator fault (OSC FAIL) when the oscillator frequency fall outside a fixed range.

2.7.2.3 Oscillator Fault Response

The oscillator fail condition is flagged in the Oscillator Fail bit (OSC FAIL) in the system module Global Status register, GLBSTAT[0]. Upon detecting an oscillator fault the CLKDET module switches any clock domain using the main oscillator as its source to the high-frequency output of the LPO (CLK10M). The microcontroller is now said to be in “limp” mode. It allows the processor to continue to operate and take the appropriate actions to ensure system safety.

Note: Availability of main oscillator after OSC FAIL is detected

The main oscillator can be used as a source for any clock domain only after power-on reset is asserted.

The application can also choose to generate a system reset when an oscillator fault is detected. This is controlled by the Reset On Oscillator Fail bit (ROF) of the PLL Control 1 Register, PLLCTL1[23].

If ROF is set, a system reset is generated under an oscillator fail condition. This reset condition is flagged as the Oscillator Reset status bit (OSC RST) in the System Exception Status Register, SYSESR[14].

2.7.3 PLL Slip Detection and Response

The PLL macro includes slip detection logic that indicates whether the PLL has lost lock between the output frequency and the input reference frequency.

The system module Global Status register (GLBSTAT) contains a status flag that indicates an over cycle slip (FBSLIP bit) and a separate flag that indicates an under cycle slip (RFSLIP bit). The response to a PLL slip condition is controlled by configurable system module registers.

- If the Bypass on PLL Slip bit is not set, then the system only captures the PLL slip condition in the SYSESR register and does nothing else.
- If the Bypass on PLL Slip bit is set, and the Reset On PLL Slip (ROS) bit is not set, then the clock monitor will switch any clock domain using the main PLL’s output as its source to the primary oscillator output from the CLKDET module as shown in Figure 2-2.
- If both the Bypass on PLL Slip bit and the Reset On PLL Slip (ROS) bit are set, a system reset is generated whenever a PLL slip is detected. This reset condition is also flagged using the Oscillator Reset status bit (OSC RST) in the System Exception Status Register, SYSESR[14]. After the reset, the clock monitor will force any clock domain selecting the main PLL’s output as its source, to instead use the primary oscillator output from the CLKDET module.

2.8 External Clock (ECLK) Pin Functions

The TMS570LS20x/10x Safety MCUs have a dedicated device pin called ECLK, which has several functions associated with it. It can be used to output a continuous clock signal, to bring out clock sources for test and debug purposes, or as a general-purpose I/O pin. The following sections describe each of these functions of the ECLK pin.

2.8.1 ECLK as Output Clock Pin

The system module supports an external clock prescaler (ECP) that can be configured to drive out a continuous clock signal on the device ECLK pin.

Table 2-5. External Clock (ECLK) Control 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
0x00 SYSPC1 Page 77	Reserved															
	Reserved															ECP CLK FUN
0xD4 ECPCNTRL Page 134	Reserved							ECP SEL	ECP COS	Reserved						
	ECPDIV															

The continuous clock output on the ECLK pin is enabled by setting the ECP CLK FUN bit in the System Pin Control 1 (SYSPC1) register. This bit is not set by default. The default value for this bit configures ECLK as a general-purpose I/O pin.

The frequency of the signal output on the ECLK pin is divided down from the main oscillator or the VCLK_sys frequency by a user-programmable ratio defined by the ECPDIV field in the ECP Control (ECPCNTL) register. This is a 16-bit field and provides division ratios from 1 up to 65536. The clock source is selected using the ECP SEL bit. ECP SEL is 0 for VCLK_sys to be the source for the ECLK output, and is 1 for OSCIN to be the source for ECLK output.

2.8.1.1 Effect of Debug Mode

When the ECP Continue On Suspend (COS – bit ECPCNTL.23) is set, the ECP continues to output ECLK even when the CPU execution is halted.

2.8.1.2 Effect of Power Down

When the device begins powers down, the VCLK_sys clock domain is disabled. Therefore, the ECLK output is also disabled. After waking up from low-power mode, the ECP returns to its previous state, removing the need to reconfigure the ECPCNTL register.

2.8.2 ECLK in Clock Test Mode

In Clock Test mode, the device clock sources can be brought out to the ECLK pin. The Clock Test Mode is only supported for test and debug purposes and must not be used in an application.

Table 2-6. Clock Test Mode Control Register

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
0x8C CLKTEST Page 117	Reserved					ALT LIMP CLOCK ENABLE	RANGE DET CTRL	RANGE DET ENA SSEL	Reserved				CLK_TEST_EN(3–0)			
	Reserved				SEL_GIO_PIN(3–0)				Reserved				SEL_ECP_PIN(3–0)			

The Clock Test mode is enabled by writing 0x5 to the CLK_TEST_EN field of the CLKTEST register. Writing any other value to this field disables the clock test mode.

The SEL_ECP_PIN field is used to select the device clock source that is brought out on the ECLK pin. Table 2-7 lists the options available in this mode.

Table 2-7. Clock Test Mode: Sources for the ECLK pin

SEL_ECP_PIN[3:0]	CLOCK SOURCE
0	OSCIN
1	FMzPLL output
2	Reserved
3	Reserved
4	LFLPO
5	HFLPO
6	FPLL output
7	Reserved
8	GCLK
9	RTI1CLK
10	Reserved
11	VCLKA1
12	VCLKA2
13 to 15	OSCIN

2.8.3 ECLK As General-Purpose I/O Pin

As stated before, the ECP CLK FUN bit in the SYSPC1 register is 0 by default. This causes the ECLK pin to be a general-purpose I/O pin by default. The System module has the following registers to control the general-purpose I/O functionality of the ECLK pin.

Table 2-8. Control Registers for ECLK I/O Functionality

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
0x00 SYSPC1 Page 77	Reserved															
	Reserved															ECP CLK FUN
0x04 SYSPC2 Page 78	Reserved															
	Reserved															ECP CLK DIR
0x08 SYSPC3 Page 79	Reserved															
	Reserved															ECP CLK DIN
0x0C SYSPC4 Page 80	Reserved															
	Reserved															ECP CLK DOUT
0x10 SYSPC5 Page 81	Reserved															
	Reserved															ECP CLK SET
0x14 SYSPC6 Page 82	Reserved															
	Reserved															ECP CLK CLR
0x18 SYSPC7 Page 83	Reserved															
	Reserved															ECP CLK ODE
0x1C SYSPC8 Page 84	Reserved															
	Reserved															ECP CLK PDIS

Table 2-8. Control Registers for ECLK I/O Functionality

0x20 SYSPC9 Page 85	Reserved	
	Reserved	ECP CLK PSEL

The ECP CLK DIR bit in the SYSPC2 register is used to control the direction of the ECLK pin when used as a general-purpose I/O pin.

The ECP CLK DIN bit in the SYSPC3 register always reflects the logic state of the ECLK pin.

The ECP CLK DOUT bit in the SYSPC4 register is used to define the logic state output onto the ECLK pin. This bit is only effective when the ECLK pin is a general-purpose output pin, that is, the ECP CLK FUN bit is 0 and the ECP CLK DIR bit is 1.

The ECP CLK SET bit in the SYSPC5 register is used to drive a logic High state onto the ECLK pin. This bit is only effective when the ECLK pin is a general-purpose output pin, that is, the ECP CLK FUN bit is 0 and the ECP CLK DIR bit is 1.

The ECP CLK CLR bit in the SYSPC6 register is used to drive a logic Low state onto the ECLK pin. This bit is only effective when the ECLK pin is a general-purpose output pin, that is, the ECP CLK FUN bit is 0 and the ECP CLK DIR bit is 1.

The ECP CLK ODE bit in the SYSPC7 register is used to make the ECLK pin an open-drain output pin. This function requires the ECLK pin to be configured as a general-purpose output pin, that is, the ECP CLK FUN bit is 0 and the ECP CLK DIR bit is 1. When the ECP CLK ODE bit is set, then driving a 1 via the ECP CLK DOUT or ECP CLK SET bits will tristate the ECLK pin. Driving a 0 via the ECP CLK DOUT or ECP CLK CLR bits will output a logic low on the ECLK pin.

The ECP CLK PDIS bit in the SYSPC8 register is used to disable the pull on the ECLK pin. When this bit is set, there is no pull on the ECLK pin. When the bit is cleared, the pull on the ECLK pin is determined by the value of the ECP CLK PSEL bit in the SYSPC9 register.

When the ECP CLK PSEL bit is set the ECLK pin is pulled up, and when the ECP CLK PSEL bit is cleared the ECLK pin is pulled down.

The pull on ECLK pin is active only when it is an input pin, that is, ECP CLK FUN bit is 0 and ECP CLK DIR bit is 0

TMS570LS20x/10x Safety MCUs Exceptions

This chapter describes the exception conditions on the TMS570LS20x/10x Safety MCUs.

Topic ... Page

3.1 Resets

3.1.1 Reset Conditions

The TMS570LS20x/10x Safety MCUs can be reset by any of the conditions listed in Table 3-1. Each reset condition is flagged in the system exception status register (SYSESR).

Table 3-1. Causes of TMS570LS20x/10x Safety MCUs 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 PO RST bit in the SYSESR register, SYSESR[15].
Voltage Monitor reset	The TMS570LS20x/10x Safety MCUs have 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 PO RST 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 TMS570LS20x/10x Safety MCUs also get reset whenever the other IC detects a fault condition. This reset is flagged by the EXT RST 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 OSC RST 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 SW RST bit of 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 MMUGCR register. This reset is flagged by the CPU RST bit of the SYSESR register, SYSESR[5].
Debug reset	The ICEPICK logic implemented on the TMS570LS20x/10x Safety MCUs allow a system reset to be generated via the debug logic. This reset is flagged by the DBG RST bit of the SYSESR register, SYSESR[13].

The device nRST pin is an indicator for the TMS570LS20x/10x Safety MCUs to be under a reset condition. This pin will be driven output low whenever the device is under reset. As a result the EXT RST bit in the SYSESR register, SYSESR[3], gets set for all reset conditions. The device nRST pin is driven output low for at least 8 VCLK cycles for any reset condition. This time is 1048 cycles of the main oscillator (OSCIN) if the device power-on reset is asserted (nPORRST pin driven low).

Table 3-2 shows the system exception control and status registers.

Table 3-2. System Exception 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
0xE0 SYSECR Page 137	Reserved															
	RESET [1]	RESET [0]	Reserved													

0xE4 SYSESR Page 138	Reserved							
	PO RST	OSC RST	DBG RST	Reserved	CPU RST	SW RST	EXT RST	Reserved

3.2 Aborts

3.2.1 Precise and Imprecise Aborts

Two general types of aborts are possible: precise (or synchronous) and imprecise (or asynchronous) abort.

- A precise abort is defined as an abort in which the initiator of a transaction can determine the exact transaction which generated the abort. The initiator may then “rewind” the system back to the pre-abort state. Since the transaction which generated the abort is known, information about this transaction such as the address, read vs. write, etc. is recorded for later debug. A fault on a CPU instruction fetch is an example of a precise abort.
- An imprecise abort is defined as an abort in which the initiator of a transaction cannot determine the exact transaction which has generated the abort. In this case, the initiator cannot restore the state of the system to that before the abort. A fault on a buffered write transaction is an example of an imprecise abort.

An imprecise abort may be defined as internal or external.

- An internal imprecise abort occurs due to buffering or other imprecision inside a bus master. Such imprecision is generally comprehended by the bus master.
- An external imprecise abort occurs due to buffering or other imprecision in the data path of a bus master, but outside the bus master. The bus master cannot directly comprehend such an abort and the system impact is often fatal.

The TMS570 platform architecture applies techniques at the system level to mitigate the impact of external imprecise aborts. System level adoption of write status sidebands to the datapath allow bus masters to comprehend external 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 external imprecise abort may be registered at the point of imprecision for later analysis.

3.2.2 System Abort Conditions

The TMS570 platform architecture defines the following error conditions:

- Access to an illegal address (a non-implemented address)
- Access to a protected address (protection violation)
- Parity / ECC / Timeout Error on a valid access

3.2.2.1 Accesses to Illegal Addresses

The definition of an illegal address is dependent on the target slave memory frame being addressed, as described in Table 3-3.

Table 3-3. Definition of a Non-Implemented Address

Slave Memory Frame	Definition of Non-Implemented Address (Illegal Address)
eSRAMx	The eSRAM memory frame is capable of addressing up to 64MB. This TMS570LS20x/10x Safety MCUs only have 160KB of eSRAM in the eSRAM0 frame. Any access beyond 160KB generates an error response.
Reserved RAM frame	This TMS570LS20x/10x Safety MCUs does not have the RAM frames nCSRAM2 and nCSRAM3 implemented. Any access to these frames generates an error response.
Reserved memory	An error response is generated for any access to reserved memory.
CRC slave	The CRC module generates an error response for any access to the non-implemented addresses in the CRC slave frame.

Table 3-3. Definition of a Non-Implemented Address

Slave Memory Frame	Definition of Non-Implemented Address (Illegal Address)
Peripheral memory chip select frame	An access to a non-implemented peripheral memory address generates an error response. Each peripheral memory select can address up to 128KB. For any peripheral memory smaller than 128KB, an error response is generated for accesses to addresses starting from the nearest power of 2, with a minimum resolution of 1KB. For example, the MiBSPI1 supports 128 buffers. Each buffer is 64 bits wide. So the MiBSPI1 RAM only occupies 128 * 8 bytes = 1KB. Any access over an offset of 1KB to the MiBSPI1 memory frame will generate an error response.
Peripheral select frame	Each peripheral select frame contains 4 quadrants of 256 bytes each. An error response is generated for any access to the non-implemented peripheral select quadrants. Error handling for accesses within a frame/quadrant is module specific.

3.2.2.2 Accesses to protected address

Accesses to protected memory are defined as illegal accesses. The different types of illegal accesses are described below:

- Memory access protection.
  Memory access permissions are controlled by the MPU. 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.
- Peripheral register or Peripheral memory access right violation.
  Accesses to the peripherals are protected either by the MPU or by the peripheral frame protection register in the system module.
  - *Peripheral Register Access Violation:*
    The PCR module registers PPROTSETx contain one protection bit per peripheral register quadrant. These protection bits define the access rights to the peripherals on the VBUS. When the CPU attempts to write to any peripheral register frame for which it does not have the correct permissions, a protection violation is detected and an error response is generated.

    In addition, individual peripherals on the VBUS have certain register bits that are only writable in the privileged operating modes. If the CPU attempts to write to these peripheral register bits in a non-privileged operating mode, the write operation is simply ignored and no error response is generated.
  - *Peripheral Memory Access Violation:*
    The PCR module registers PMPROTSETx have one bit per peripheral memory frame. These protection bits define the access rights to the peripheral memory frames on the VBUS. When the CPU attempts to write to any peripheral memory frame for which it does not have the correct permission rights, a protection right access is detected and an error response is generated.

The peripheral register and peripheral memory protection control registers are shown in Table 3-4. Not all peripheral memory selects and peripheral register selects are implemented on the TMS570LS20x/10x Safety MCUs. Refer to Table 1-3 and Table 1-4 for the available peripheral memory selects and peripheral register selects.

Writing a 1 to each of the applicable PROT SET bits enables the access protection for the corresponding peripheral memory or register frames. Any protected peripheral register or memory frame can only be written to in privileged CPU mode. User mode write accesses generate an error response. Reads are still permitted in both user and privileged modes.

Table 3-4. Peripheral Register and Memory Protection Control Registers

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 PMPROTSET0 Page 157	PCS 31 PROT SET	PCS 30 PROT SET	PCS 29 PROT SET	PCS 28 PROT SET	PCS 27 PROT SET	PCS 26 PROT SET	PCS 25 PROT SET	PCS 24 PROT SET	PCS 23 PROT SET	PCS 22 PROT SET	PCS 21 PROT SET	PCS 20 PROT SET	PCS 19 PROT SET	PCS 18 PROT SET	PCS 17 PROT SET	PCS 16 PROT SET
	PCS 15 PROT SET	PCS 14 PROT SET	PCS 13 PROT SET	PCS 12 PROT SET	PCS 11 PROT SET	PCS 10 PROT SET	PCS 9 PROT SET	PCS 8 PROT SET	PCS 7 PROT SET	PCS 6 PROT SET	PCS 5 PROT SET	PCS 4 PROT SET	PCS 3 PROT SET	PCS 2 PROT SET	PCS 1 PROT SET	PCS 0 PROT SET
0x04 PMPROTSET1 Page 158	PCS 63 PROT SET	PCS 62 PROT SET	PCS 61 PROT SET	PCS 60 PROT SET	PCS 59 PROT SET	PCS 58 PROT SET	PCS 57 PROT SET	PCS 56 PROT SET	PCS 55 PROT SET	PCS 54 PROT SET	PCS 53 PROT SET	PCS 52 PROT SET	PCS 51 PROT SET	PCS 50 PROT SET	PCS 49 PROT SET	PCS 48 PROT SET
	PCS 47 PROT SET	PCS 46 PROT SET	PCS 45 PROT SET	PCS 44 PROT SET	PCS 43 PROT SET	PCS 42 PROT SET	PCS 41 PROT SET	PCS 40 PROT SET	PCS 39 PROT SET	PCS 38 PROT SET	PCS 37 PROT SET	PCS 36 PROT SET	PCS 35 PROT SET	PCS 34 PROT SET	PCS 33 PROT SET	PCS 32 PROT SET
0x10 PMPROTCLR0 Page 159	PCS 31 PROT CLR	PCS 30 PROT CLR	PCS 29 PROT CLR	PCS 28 PROT CLR	PCS 27 PROT CLR	PCS 26 PROT CLR	PCS 25 PROT CLR	PCS 24 PROT CLR	PCS 23 PROT CLR	PCS 22 PROT CLR	PCS 21 PROT CLR	PCS 20 PROT CLR	PCS 19 PROT CLR	PCS 18 PROT CLR	PCS 17 PROT CLR	PCS 16 PROT CLR
	PCS 15 PROT CLR	PCS 14 PROT CLR	PCS 13 PROT CLR	PCS 12 PROT CLR	PCS 11 PROT CLR	PCS 10 PROT CLR	PCS 9 PROT CLR	PCS 8 PROT CLR	PCS 7 PROT CLR	PCS 6 PROT CLR	PCS 5 PROT CLR	PCS 4 PROT CLR	PCS 3 PROT CLR	PCS 2 PROT CLR	PCS 1 PROT CLR	PCS 0 PROT CLR
0x14 PMPROTCLR1 Page 160	PCS 63 PROT CLR	PCS 62 PROT CLR	PCS 61 PROT CLR	PCS 60 PROT CLR	PCS 59 PROT CLR	PCS 58 PROT CLR	PCS 57 PROT CLR	PCS 56 PROT CLR	PCS 55 PROT CLR	PCS 54 PROT CLR	PCS 53 PROT CLR	PCS 52 PROT CLR	PCS 51 PROT CLR	PCS 50 PROT CLR	PCS 49 PROT CLR	PCS 48 PROT CLR
	PCS47 PROT CLR	PCS46 PROT CLR	PCS45 PROT CLR	PCS44 PROT CLR	PCS43 PROT CLR	PCS42 PROT CLR	PCS41 PROT CLR	PCS40 PROT CLR	PCS39 PROT CLR	PCS38 PROT CLR	PCS37 PROT CLR	PCS36 PROT CLR	PCS35 PROT CLR	PCS34 PROT CLR	PCS33 PROT CLR	PCS32 PROT CLR
0x20 PPROTSET0 Page 161	PS7 QUAD 3 PROT SET	PS7 QUAD 2 PROT SET	PS7 QUAD 1 PROT SET	PS7 QUAD 0 PROT SET	PS6 QUAD 3 PROT SET	PS6 QUAD 2 PROT SET	PS6 QUAD 1 PROT SET	PS6 QUAD 0 PROT SET	PS5 QUAD 3 PROT SET	PS5 QUAD 2 PROT SET	PS5 QUAD 1 PROT SET	PS5 QUAD 0 PROT SET	PS4 QUAD 3 PROT SET	PS4 QUAD 2 PROT SET	PS4 QUAD 1 PROT SET	PS4 QUAD 0 PROT SET
	PS3 QUAD 3 PROT SET	PS3 QUAD 2 PROT SET	PS3 QUAD 1 PROT SET	PS3 QUAD 0 PROT SET	PS2 QUAD 3 PROT SET	PS2 QUAD 2 PROT SET	PS2 QUAD 1 PROT SET	PS2 QUAD 0 PROT SET	PS1 QUAD 3 PROT SET	PS1 QUAD 2 PROT SET	PS1 QUAD 1 PROT SET	PS1 QUAD 0 PROT SET	PS0 QUAD 3 PROT SET	PS0 QUAD 2 PROT SET	PS0 QUAD 1 PROT SET	PS0 QUAD 0 PROT SET
0x24 PPROTSET1 Page 163	PS15 QUAD 3 PROT SET	PS15 QUAD 2 PROT SET	PS15 QUAD 1 PROT SET	PS15 QUAD 0 PROT SET	PS14 QUAD 3 PROT SET	PS14 QUAD 2 PROT SET	PS14 QUAD 1 PROT SET	PS14 QUAD 0 PROT SET	PS13 QUAD 3 PROT SET	PS13 QUAD 2 PROT SET	PS13 QUAD 1 PROT SET	PS13 QUAD 0 PROT SET	PS12 QUAD 3 PROT SET	PS12 QUAD 2 PROT SET	PS12 QUAD 1 PROT SET	PS12 QUAD 0 PROT SET
	PS11 QUAD 3 PROT SET	PS11 QUAD 2 PROT SET	PS11 QUAD 1 PROT SET	PS11 QUAD 0 PROT SET	PS10 QUAD 3 PROT SET	PS10 QUAD 2 PROT SET	PS10 QUAD 1 PROT SET	PS10 QUAD 0 PROT SET	PS9 QUAD 3 PROT SET	PS9 QUAD 2 PROT SET	PS9 QUAD 1 PROT SET	PS9 QUAD 0 PROT SET	PS8 QUAD 3 PROT SET	PS8 QUAD 2 PROT SET	PS8 QUAD 1 PROT SET	PS8 QUAD 0 PROT SET
0x28 PPROTSET2 Page 164	PS23 QUAD 3 PROT SET	PS23 QUAD 2 PROT SET	PS23 QUAD 1 PROT SET	PS23 QUAD 0 PROT SET	PS22 QUAD 3 PROT SET	PS22 QUAD 2 PROT SET	PS22 QUAD 1 PROT SET	PS22 QUAD 0 PROT SET	PS21 QUAD 3 PROT SET	PS21 QUAD 2 PROT SET	PS21 QUAD 1 PROT SET	PS21 QUAD 0 PROT SET	PS20 QUAD 3 PROT SET	PS20 QUAD 2 PROT SET	PS20 QUAD 1 PROT SET	PS20 QUAD 0 PROT SET
	PS19 QUAD 3 PROT SET	PS19 QUAD 2 PROT SET	PS19 QUAD 1 PROT SET	PS19 QUAD 0 PROT SET	PS18 QUAD 3 PROT SET	PS18 QUAD 2 PROT SET	PS18 QUAD 1 PROT SET	PS18 QUAD 0 PROT SET	PS17 QUAD 3 PROT SET	PS17 QUAD 2 PROT SET	PS17 QUAD 1 PROT SET	PS17 QUAD 0 PROT SET	PS16 QUAD 3 PROT SET	PS16 QUAD 2 PROT SET	PS16 QUAD 1 PROT SET	PS16 QUAD 0 PROT SET

Table 3-4. Peripheral Register and Memory Protection Control Registers

0x2C PPROTSET3 Page 165	PS31 QUAD 3 PROT SET	PS31 QUAD 2 PROT SET	PS31 QUAD 1 PROT SET	PS31 QUAD 0 PROT SET	PS30 QUAD 3 PROT SET	PS30 QUAD 2 PROT SET	PS30 QUAD 1 PROT SET	PS30 QUAD 0 PROT SET	PS29 QUAD 3 PROT SET	PS29 QUAD 2 PROT SET	PS29 QUAD 1 PROT SET	PS29 QUAD 0 PROT SET	PS28 QUAD 3 PROT SET	PS28 QUAD 2 PROT SET	PS28 QUAD 1 PROT SET	PS28 QUAD 0 PROT SET
	PS27 QUAD 3 PROT SET	PS27 QUAD 2 PROT SET	PS27 QUAD 1 PROT SET	PS27 QUAD 0 PROT SET	PS26 QUAD 3 PROT SET	PS26 QUAD 2 PROT SET	PS26 QUAD 1 PROT SET	PS26 QUAD 0 PROT SET	PS25 QUAD 3 PROT SET	PS25 QUAD 2 PROT SET	PS25 QUAD 1 PROT SET	PS25 QUAD 0 PROT SET	PS24 QUAD 3 PROT SET	PS24 QUAD 2 PROT SET	PS24 QUAD 1 PROT SET	PS24 QUAD 0 PROT SET
0x40 PPROTCLR0 Page 166	PS7 QUAD 3 PROT CLR	PS7 QUAD 2 PROT CLR	PS7 QUAD 1 PROT CLR	PS7 QUAD 0 PROT CLR	PS6 QUAD 3 PROT CLR	PS6 QUAD 2 PROT CLR	PS6 QUAD 1 PROT CLR	PS6 QUAD 0 PROT CLR	PS5 QUAD 3 PROT CLR	PS5 QUAD 2 PROT CLR	PS5 QUAD 1 PROT CLR	PS5 QUAD 0 PROT CLR	PS4 QUAD 3 PROT CLR	PS4 QUAD 2 PROT CLR	PS4 QUAD 1 PROT CLR	PS4 QUAD 0 PROT CLR
	PS3 QUAD 3 PROT CLR	PS3 QUAD 2 PROT CLR	PS3 QUAD 1 PROT CLR	PS3 QUAD 0 PROT CLR	PS2 QUAD 3 PROT CLR	PS2 QUAD 2 PROT CLR	PS2 QUAD 1 PROT CLR	PS2 QUAD 0 PROT CLR	PS1 QUAD 3 PROT CLR	PS1 QUAD 2 PROT CLR	PS1 QUAD 1 PROT CLR	PS1 QUAD 0 PROT CLR	PS0 QUAD 3 PROT CLR	PS0 QUAD 2 PROT CLR	PS0 QUAD 1 PROT CLR	PS0 QUAD 0 PROT CLR
0x44 PPROTCLR1 Page 167	PS15 QUAD 3 PROT CLR	PS15 QUAD 2 PROT CLR	PS15 QUAD 1 PROT CLR	PS15 QUAD 0 PROT CLR	PS14 QUAD 3 PROT CLR	PS14 QUAD 2 PROT CLR	PS14 QUAD 1 PROT CLR	PS14 QUAD 0 PROT CLR	PS13 QUAD 3 PROT CLR	PS13 QUAD 2 PROT CLR	PS13 QUAD 1 PROT CLR	PS13 QUAD 0 PROT CLR	PS12 QUAD 3 PROT CLR	PS12 QUAD 2 PROT CLR	PS12 QUAD 1 PROT CLR	PS12 QUAD 0 PROT CLR
	PS11 QUAD 3 PROT CLR	PS11 QUAD 2 PROT CLR	PS11 QUAD 1 PROT CLR	PS11 QUAD 0 PROT CLR	PS10 QUAD 3 PROT CLR	PS10 QUAD 2 PROT CLR	PS10 QUAD 1 PROT CLR	PS10 QUAD 0 PROT CLR	PS9 QUAD 3 PROT CLR	PS9 QUAD 2 PROT CLR	PS9 QUAD 1 PROT CLR	PS9 QUAD 0 PROT CLR	PS8 QUAD 3 PROT CLR	PS8 QUAD 2 PROT CLR	PS8 QUAD 1 PROT CLR	PS8 QUAD 0 PROT CLR
0x48 PPROTCLR2 Page 168	PS23 QUAD 3 PROT CLR	PS23 QUAD 2 PROT CLR	PS23 QUAD 1 PROT CLR	PS23 QUAD 0 PROT CLR	PS22 QUAD 3 PROT CLR	PS22 QUAD 2 PROT CLR	PS22 QUAD 1 PROT CLR	PS22 QUAD 0 PROT CLR	PS21 QUAD 3 PROT CLR	PS21 QUAD 2 PROT CLR	PS21 QUAD 1 PROT CLR	PS21 QUAD 0 PROT CLR	PS20 QUAD 3 PROT CLR	PS20 QUAD 2 PROT CLR	PS20 QUAD 1 PROT CLR	PS20 QUAD 0 PROT CLR
	PS19 QUAD 3 PROT CLR	PS19 QUAD 2 PROT CLR	PS19 QUAD 1 PROT CLR	PS19 QUAD 0 PROT CLR	PS18 QUAD 3 PROT CLR	PS18 QUAD 2 PROT CLR	PS18 QUAD 1 PROT CLR	PS18 QUAD 0 PROT CLR	PS17 QUAD 3 PROT CLR	PS17 QUAD 2 PROT CLR	PS17 QUAD 1 PROT CLR	PS17 QUAD 0 PROT CLR	PS16 QUAD 3 PROT CLR	PS16 QUAD 2 PROT CLR	PS16 QUAD 1 PROT CLR	PS16 QUAD 0 PROT CLR
0x4C PPROTCLR3 Page 169	PS31 QUAD 3 PROT CLR	PS31 QUAD 2 PROT CLR	PS31 QUAD 1 PROT CLR	PS31 QUAD 0 PROT CLR	PS30 QUAD 3 PROT CLR	PS30 QUAD 2 PROT CLR	PS30 QUAD 1 PROT CLR	PS30 QUAD 0 PROT CLR	PS29 QUAD 3 PROT CLR	PS29 QUAD 2 PROT CLR	PS29 QUAD 1 PROT CLR	PS29 QUAD 0 PROT CLR	PS28 QUAD 3 PROT CLR	PS28 QUAD 2 PROT CLR	PS28 QUAD 1 PROT CLR	PS28 QUAD 0 PROT CLR
	PS27 QUAD 3 PROT CLR	PS27 QUAD 2 PROT CLR	PS27 QUAD 1 PROT CLR	PS27 QUAD 0 PROT CLR	PS26 QUAD 3 PROT CLR	PS26 QUAD 2 PROT CLR	PS26 QUAD 1 PROT CLR	PS26 QUAD 0 PROT CLR	PS25 QUAD 3 PROT CLR	PS25 QUAD 2 PROT CLR	PS25 QUAD 1 PROT CLR	PS25 QUAD 0 PROT CLR	PS24 QUAD 3 PROT CLR	PS24 QUAD 2 PROT CLR	PS24 QUAD 1 PROT CLR	PS24 QUAD 0 PROT CLR

3.2.2.3 Illegal Transaction Detection and Response

Almost all illegal transactions provide an error response to the initiator **except** in the following cases:

- Write access to non-cacheable buffered memory. In this case, the CPU cannot identify the precise instruction that caused the transaction error. This is called an *imprecise abort*. For these imprecise transactions on the AMBA-AXI instruction and data buses, the SCR provides the system module with the address for an imprecise transaction.

Note:
For further explanation of imprecise aborts, please refer to the *ARM Architecture Reference Manual* (ARM reference number DDI0100). Please also refer to the Cortex-R4 Technical Reference Manual version r1p3 to verify the actual CPU behavior in case of an imprecise abort.

- Illegal writes to VBUS peripherals. These writes are buffered in the AXI-to-VBUS (A2V) bridge, which provides the CPU with an OKAY response before the actual write transaction error is detected. In this case, the A2V bridge provides the address, system mode (user/privileged), and the initiator ID of the illegal transaction to be logged in the system module.

Table 3-5 is a summary of the TMS570LS20x/10x Safety MCUs system responses when an illegal transaction is detected.

Table 3-5. Illegal Transaction Detection Responses

Access Type	System mode	Illegal Transaction Response
1) Error response to precise CPU transactions		
Non-cacheable, non-bufferable CPU accesses (NCNB)	User/Privileged	The system generates an external abort for an error response from the slave. The virtual address of the access and the abort status are logged in the MPU fault address register (FAR) and the MPU fault status register (FSR), respectively.
Non-cacheable bufferable CPU reads (NCB)		
2) Error response to imprecise CPU transactions		
Non-cacheable, bufferable writes (NCB)	All modes	The system generates an imprecise abort. The CPU enters the abort routine. The address of an illegal NCB is logged in the IMPFTADD register in the system module register. The status is given by the IMPFASTS register.
TCM writes	All modes	The CPU enters the abort routine. The auxiliary Fault Status Register (FSR) indicates which TCM bank caused the error. This information is overwritten by subsequent TCM write errors. See *ARM CPU Technical Reference Manual* for more details.
3) Error response to DMA transactions		
All DMA transactions are performed as non-cacheable, non-bufferable data accesses in user mode	User	The DMA generates an error interrupt to the host CPU for an error response from the slave to the DMA. For more information, refer to the DMA chapter.
4) Illegal writes to VBUS on which the initiator receives an okay response		
VBUS writes	All modes	The system generates an imprecise abort. The CPU enters the abort routine. The address of an illegal transaction on the VBUS is logged in the IMPFTADD register in the system module register. The status is given by the IMPFASTS register.

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 (SSIR). The SSI registers also allow the application to provide a label for that software interrupt. This label is an 8-bit value that is then 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. Refer to the System and Peripheral Control Register section for details on the related registers.

..

Table 3-6. System Software Interrupt 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
0xB0 SSIR1 Page 122	Reserved															
	SSKEY1								SSDATA1							
0xB4 SSIR2 Page 123	Reserved															
	SSKEY2								SSDATA2							
0xB8 SSIR3 Page 124	Reserved															
	SSKEY3								SSDATA3							
0xBC SSIR4 Page 125	Reserved															
	SSKEY4								SSDATA4							
0xF4 SSIVEC Page 144	Reserved															
	SSIDATA								SSIVECT[7–0]							
0xF8 SSIF Page 145	Reserved															
	Reserved												SSI_ FLAG4	SSI_ FLAG3	SSI_ FLAG2	SSI_ FLAG1

Tightly-Coupled RAM (TCRAM) Wrapper

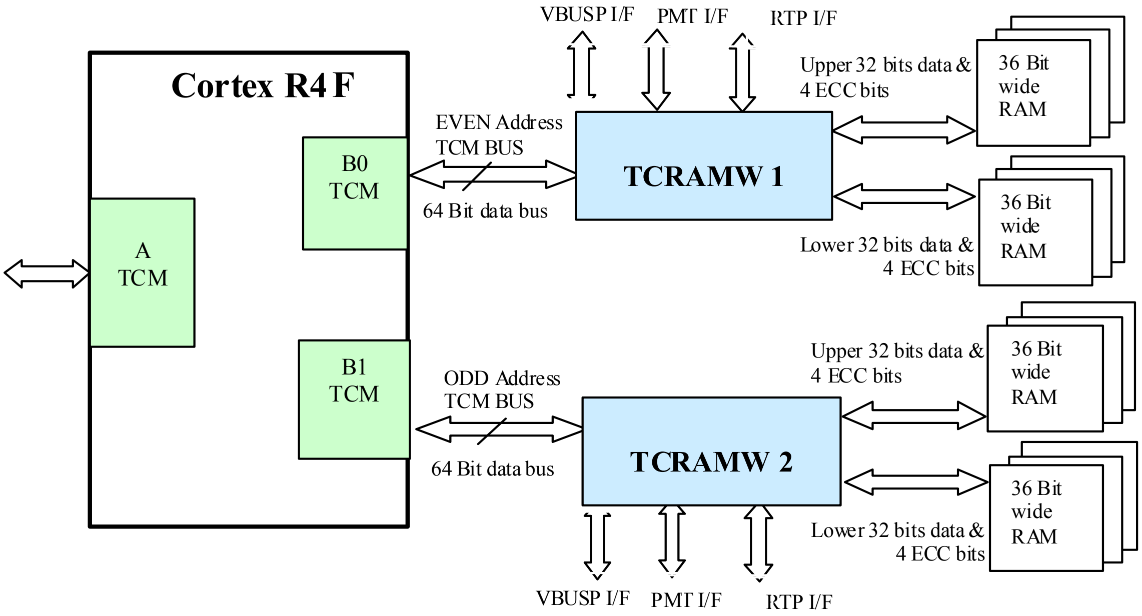
This chapter describes the tightly-coupled RAM wrapper of the Texas Instruments TMS570 microcontroller family.

Topic **Page**

4.1 Overview

The Tightly-Coupled RAM wrapper (TCRAMW) is used to connect the CPU BTCM interface to the RAM array using a 64-bit bus.

Figure 4-1. TCRAM Wrapper Connections



Some of the functions of the TCRAM wrapper are:

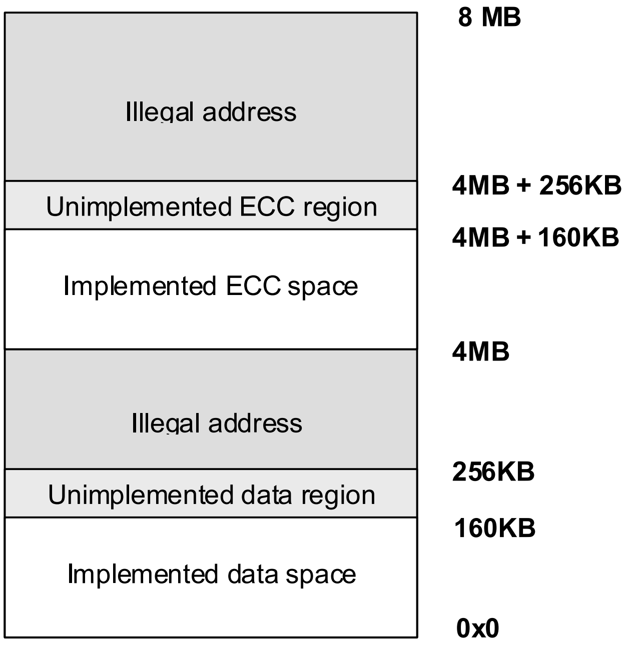
- Controls the RAM read and write operations as well as decoding the addresses within the RAM space
- Supports the Internal SECDED scheme implemented in the Cortex-R4F CPU
  - Uses the Event bus signals of the Cortex-R4F CPU to monitor the SECDED Operation and maintains the SECDED status in a user-accessible memory mapped register
  - The ECC checking logic inside the Cortex-R4F CPU is disabled after reset, and needs to be enabled by programming the auxiliary control register via a CP15 instruction
- Supports RAM trace port interface
  - Generates and traces out all RAM reads and writes to the Ram Trace Port (RTP) module.
- Supports read and write accesses in 64-bit, 32-bit, 16-bit, or 8-bit access sizes
  - Does not support bit-wise operations
- Uses redundant address decoding scheme for safety
  - Checks the decoding of address lines and generation of proper memory selects for the RAM banks
- Checks integrity of the TCM interface control bus by using the Cortex-R4F's TCM address parity scheme
  - Generates an address parity error and stores error address in memory-mapped register
- Supports auto-initialization of RAM
  - Auto-initialization can be triggered by configuring the SYS module registers
  - Refer section 1.5, *Memory Module Hardware Initialization* for more information
- The TCRAM wrapper supports two types of wait states:
  - Address pipeline which introduces a wait-state latency.
  - Extension data phase, which allows more time for data propagation.

Both types of wait states can be used and are programmable through RAM Control Register (RAMGCR) in the system module.

4.2 RAM Memory Map

The TCRAM wrapper decodes the entire 8MB of the BTCM frame. The TMS570LS20216 device supports 160KB of RAM on the CPU BTCM interface. Any access to the unimplemented space results in an error response from the TCRAM wrapper.

Figure 4-2. RAM Memory Map



Each RAM data word is 64 bits wide although the word is made up of two 36-bit wide banks connected in parallel, as shown in Figure 4-1, *TCRAM Wrapper Connections*. The 4 most significant bits from each 36-bit wide bank contributes one nibble to the 8 bits of ECC code needed to implement the Single Error Correction Double Error Detection (SECDED) scheme in the Cortex-R4F CPU.

An 8-bit ECC address is placed at every double-word (64 bits) boundary. The entire 64-bit TCM read data bus is appended with an 8-bit ECC value for a read from the 160KB RAM data space.

The ECC memory can also be directly read by a bus master by accessing a memory-mapped offset address starting from 4MB, as shown in Figure 4-2. ECC bits are read out in all byte lanes for a read from the ECC memory.

The ECC memory can be written only if the access is a 64-bit access and if ECC memory write permission is enabled via the RAMCTRL register.

Reads and writes to the ECC memory are not traced out to the RAM Trace Port.

Note: No ECC Error Generated for Accesses to ECC Memory

Reading the ECC memory sends the ECC values on both the TCM read data bus as well as the ECC bus. This may make the Cortex-R4F CPU detect a multi-bit error event as a response if ECC protection is enabled in the Cortex-R4F CPU. In order to prevent such un wanted errors for an ECC memory read, the TCRAM wrapper blocks the error signal generation for accesses to the ECC memory.

4.3 TCRAM Wrapper Support for Memory Fault Detection and Correction

The TCRAM wrapper module has safety/integrity support built-in by way of the following features:

1. Single Error Correction Double Error Detection (SECDED) support to the Cortex-R4F CPU
2. Redundant address decoding
3. Address bus parity

4.3.1 SECDED Support

The TCRAM wrapper sends out the ECC for each data read by the Cortex-R4F CPU on a dedicated TCM ECC read port. It also stored the TCM ECC Write port contents to the ECC memory when the master writes to the ECC memory. It is the master's responsibility to maintain integrity between the data and its ECC for all operations.

The TCRAM wrapper monitors the event bus of the TCM interface of the CPU.

Features dedicated for SECDED Support:

- Dedicated single error counter register (RAMOCCUR)
  - Used to count the single-bit error corrections by the Cortex-R4F CPU's SECDED block
- Error status register (RAMERRSTATUS)
  - Used to indicate the status of errors flagged by the TCRAM wrapper
  - 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
- Configurable threshold for single-bit corrections (RAMTHRESHOLD)
  - Prevents single-bit error corrections from exceeding a certain limit
  - Option to generate Single-bit error interrupt when threshold is exceeded (RAMINTCTRL)
- Register to hold the RAM address on which a double-bit error is detected (RAMUERRADDR)
- Register to hold the RAM address on which a single-bit error is detected (RAMSERRADDR)
  - This register is only updated when the single-bit error threshold is set to be 1

The TCRAM wrapper ignore the CPU's Event bus if an error is indicated for an access to ECC memory.

Note: Enabling monitoring of Cortex-R4F CPU's Event Bus

On reset the Event Export bit (X bit) of the Performance Monitor Control Register (PMNC) in the Cortex-R4F CPU is disabled. This bit must be set for the Event bus to indicate the data error so that the TCRAM wrapper could capture the Events in the registers dedicated for SECDED support.

4.3.2 Redundant Address Decode Logic

The address decode logic to generate the selects for the RAM memory and the ECC memory are duplicated in order to be able to detect decode logic errors. The primary and the redundant select signals are constantly compared by the safety logic and any mismatch is indicated by an address error signal mapped to the Error Signaling Module group 2 channel 6 for B0TCM, and channel 8 for B1TCM. The faulty address is captured in the RAMUERRADDR register.

In addition to the redundant address decode logic, any error in the RAM bank address decode logic is detected by generating two different memory selects and address signals to the RAM banks. Each 36-bit wide bank receives one of the memory selects and the address from the TCRAM wrapper. 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 scheme will detect this data error.

The TCRAM wrapper also supports a testing mechanism to ensure proper operation of the redundant address decode and compare logic itself. This logic is checked by providing a test stimulus and is triggered by programming the RAMTEST register. The address of an error identified during testing of the 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 selects for the RAM memory and the ECC memory.

4.3.3 Address Bus Parity Checking

The TCRAM wrapper checks the address bus integrity by calculating a syndrome for the entire BTCM control bus comprised of the TCEN0, TCEN1, TCADDR, TCBYTEWR, TCSEQ, TCACCTYPE and PARLVRAM signals output from the Cortex-R4F CPU along with the address parity bit generated by the Cortex-R4F CPU. The Cortex-R4F CPU generates a single parity bit in the cycle after it issues an access. This parity bit would always reach the TCRAM wrapper a cycle after the address reaches the TCRAM wrapper. Any mismatch between the address and the parity bit would result in an Address Parity Failure error signal. This error signal is mapped to the Error Signaling Module's Group 2 Channel 10 for B0TCM and Channel 12 for B1TCM.

The RAM address which fails the parity checking is captured in the RAMPERRADDR register. The TCRAM wrapper also indicates whether the parity checking failure occurred on a read access or a write access. This is indicated by two separate bits: RADDR PAR FAIL and WADDR PAR FAIL in the RAMERRSTATUS register.

The RAMERRSTATUS and RAMPERRADDR registers must be cleared in order to continue capturing subsequent errors.

The parity scheme used for checking the address parity is defined by the global system parity selection. This can be selected by configuring the DEVPARSEL field in the system module DEVCR1 register.

The global parity scheme can be overridden by configuring the address parity override field in the RAMCTRL register.

Note: No change of parity scheme on the fly

The TCRAM wrapper does not support on the fly change to the parity scheme being used. The application must ensure that the parity polarity (odd or even) is not changed when there is an ongoing access to the RAM.

4.4 Emulation / Debug Mode Behavior

The following describes the behavior of the TCRAM wrapper when in debug mode:

- The RAMOCCUR register continues to count the single-bit error corrections performed by the Cortex-R4F 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.

4.5 Trace Module Support

The TCRAM wrapper traces out the following signals to the RAM Trace Port (RTP) module, thereby providing RAM dataport trace capability.

- 18-bit address line
- 64-bit data bus
- Byte strobe information
- Current access master identification number
- Access type: Opcode or data fetch
- Read or Write access

No data is traced for an access to ECC memory.

4.6 Auto Initialization Support

The RAM memory can be initialized by using the dedicated auto-initialization hardware. The TCRAM wrapper 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, 0x0C.

4.7 Control and Status Registers

The TCRAM wrapper registers are accessed in the Cortex-R4F 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 0xFFFFF800 and 0xFFFFF900 for even and odd RAM ECC.

Table 4-1. TCRAM Wrapper Control and Status Register Map

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
0x00 RAM CTRL Page 56	Reserved	EMU TRACE DIS	Reserved		ADDR PARITY OVERRIDE				Reserved				ADDR PARITY DISABLE			
	Reserved							ECC WR EN	Reserved				ECC DETECT EN			
0x04 RAM THRESHOLD Page 58	Reserved															
	THRESHOLD(15–0)															
0x08 RAM OCCUR Page 59	Reserved															
	SINGLE ERROR OCCURRENCES(15–0)															
0x0C RAM INTCTRL Page 60	Reserved															
	Reserved															SERR EN
0x10 RAM ERRSTATUS Page 61	Reserved															
	Reserved						WADDR PAR FAIL	RADDR PAR FAIL	Reserved		DERR	ADDR COMP LOGIC FAIL	Res	ADDR DEC FAIL	Res	SERR
0x14 RAM SERRADDR Page 63	Reserved															
	SINGLE ERROR ADDRESS(17–3)													Reserved		
0x1C RAM UERRADDR Page 64	Reserved															
	UNCORRECTABLE ERROR ADDRESS(22–3)													Reserved		

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30 RAM TEST Page 65	Reserved															
	Reserved							TRIG-GER	TEST MODE		Reserved		TEST ENABLE			
0x38 RAM ADDRDECVECT Page 67	Reserved					ECC SELECT	Reserved									
	RAM CHIP SELECT(15–0)															
0x3C RAM PERADDR Page 68	Reserved															
	ADDRESS PARITY ERROR ADDRESS(22–3)													Reserved		

4.7.1 TCRAM Wrapper Control Register (RAMCTRL)

The RAMCTL register, shown in Figure 4-3 and described in Table 4-2, controls the safety features supported by the TCRAM wrapper.

Figure 4-3. TCRAM Wrapper Control Register (RAMCTRL) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserve d	EMU TRACE DIS	Reserved		ADDR PARITY OVERRIDE				Reserved				ADDR PARITY DISABLE			
R-0	R/WP-0	R-0		R/WP-0				R-0				R/WP-0101			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ECC WR EN	Reserved				ECC DETECT EN			
R-0							R/WP-0	R-0				R/WP-1010			

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-2. TCRAM Wrapper Control Register (RAMCTRL) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return zero and 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 for an emulation mode access.
		0	Data is allowed to be traced out to the trace modules for emulation mode accesses.
		1	Data is blocked from being traced out to the trace modules for emulation mode accesses.
29–28	Reserved		Reads return zero and writes have no effect.
27–24	ADDR PARITY OVERRIDE		Address Parity Override. This field, when set to 0xD, 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.
		1101	Parity scheme is opposite to the device global parity scheme.
		All others	Parity scheme is the same as the device global parity scheme.
23–20	Reserved		Reads return zero and writes have no effect.

Table 4-2. TCRAM Wrapper Control Register (RAMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
19–16	ADDR PARITY DISABLE		Address Parity Detect Disable. This field, when set to 0xA, 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 parity error address register is cleared before enabling address parity checking.**
		1010	Address parity checking is disabled
		All others	Address parity checking is enabled
15–9	Reserved		Reads return zero and 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.**
		0	ECC memory writes are disabled.
		1	ECC memory writes are enabled.
7–4	Reserved		Reads return zero and 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 wrapper. If this field is set to any value other than 0x5, then the TCRAM wrapper 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 0x5. The ECC detection is enabled by default as the ECC DETECT EN default value is 0xA.
		0101	ECC detection is disabled.
		All others	ECC detection is enabled.

4.7.2 TCRAM Wrapper Single-Bit Error Correction Threshold Register (RAMTHRESHOLD)

The RAMTHRESHOLD register, shown in Figure 4-4 and described in Table 4-3, allows the application to configure the number of single-bit error corrections by the SECDED logic inside the Cortex-R4F CPU before generating a single-bit error interrupt.

Figure 4-4. TCRAM Wrapper Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) [offset = 0x04]

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
THRESHOLD(15–0)															
R/WP-0															

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-3. TCRAM Wrapper Single-Bit Error Correction Threshold Register (RAMTHRESHOLD) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	THRESHOLD		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.

4.7.3 TCRAM Wrapper Single-Bit Error Occurrences Counter Register (RAMOCCUR)

The RAMOCCUR register, shown in Figure 4-5 and described in Table 4-4, indicates the number of single-bit error corrections performed by the SEC logic inside the Cortex-R4F CPU.

Figure 4-5. TCRAM Wrapper Single-Bit Error Occurrences Counter Register (RAMOCCUR) [offset = 0x08]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">SINGLE ERROR OCCURRENCES(15–0)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-4. TCRAM Wrapper Single-Bit Error Occurrences Counter Register (RAMOCCUR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	SINGLE ERROR OCCURRENCES		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 0x0 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 wrapper tried to update it, then the TCRAM wrapper 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.**

4.7.4 TCRAM Wrapper Interrupt Control Register (RAMINTCTRL)

The RAMINTCTRL register, shown in Figure 4-6 and described in Table 4-5, enables the generation of an interrupt to the CPU whenever the number of single-bit error corrections (RAMOCCUR) reaches the programmed threshold (RAMTHRESHOLD).

Figure 4-6. TCRAM Wrapper Interrupt Control Register (RAMINTCTRL) [offset = 0x0C]

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
Reserved															SERR EN
R-0															R/WP-0

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-5. TCRAM Wrapper Interrupt Control Register (RAMINTCTRL) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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.

4.7.5 TCRAM Wrapper Error Status Register (RAMERRSTATUS)

The RAMERRSTATUS register, shown in Figure 4-7 and described in Table 4-6, indicates the status of the various error conditions monitored by the TCRAM wrapper.

Figure 4-7. TCRAM Wrapper Error Status Register (RAMERRSTATUS) [offset = 0x10]

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
Reserved						WADDR PAR FAIL	RADDR PAR FAIL	Reserved		DERR	ADDR COMP LOGIC FAIL	Res	ADDR DEC FAIL	Res	SERR
R-0						R/WP-0	R/WP-0	R-0		R/WP-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-6. TCRAM Wrapper Error Status Register (RAMERRSTATUS) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9	WADDR PAR FAIL		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		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		Reads return zero and writes have no effect.
5	DERR		This bit indicates a multi-bit error detected by the Cortex-R4F SECDED logic.
4	ADDR COMP LOGIC FAIL		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		Reads return zero and writes have no effect.

Table 4-6. TCRAM Wrapper Error Status Register (RAMERRSTATUS) Field Descriptions (Continued)

Bit	Name	Value	Description
2	ADDR DEC FAIL		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		Reads return zero and writes have no effect.
0	SERR		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.

4.7.6 TCRAM Wrapper Single-Bit Error Address Register (RAMSERRADDR)

The RAMSERRADDR register, shown in Figure 4-8 and described in Table 4-7, captures the address for which the Cortex-R4F CPU detected a single-bit error.

Figure 4-8. TCRAM Wrapper Single-Bit Error Address Register (RAMSERRADDR) [offset = 0x14]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SINGLE ERROR ADDRESS(17–3)													Reserved		
R-0													R-0		

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-7. TCRAM Wrapper Single-Bit Error Address Register (RAMSERRADDR) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return zero and writes have no effect.
17–3	SINGLE ERROR ADDRESS		This register captures the bits 17–3 of the address for which the Cortex-R4F 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 register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2–0	Reserved		Reads return zero and writes have no effect.

4.7.7 TCRAM Wrapper Uncorrectable Error Address Register (RAMUERRADDR)

The RAMUERRADDR register, shown in Figure 4-9 and described in Table 4-8, captures the address for which the Cortex-R4F CPU detected a multi-bit error.

Figure 4-9. TCRAM Wrapper Uncorrectable Error Address Register (RAMUERRADDR) [offset = 0x1C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									UNCORRECTABLE ERROR ADDRESS(22–16)						
R-0									R-0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNCORRECTABLE ERROR ADDRESS(15–3)													Reserved		
R-0													R-0		

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-8. TCRAM Wrapper Uncorrectable Error Address Register (RAMUERRADDR) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads return zero and writes have no effect.
22–3	UNCORRECT-ABLE ERROR ADDRESS		This register captures the address for which there was an uncorrectable error or an address error. The uncorrectable error is indicated by the Cortex-R4F CPU's SECDED logic and the address error is indicated by the TCRAM wrapper's redundant address decode and compare logic. For the SECDED multi-bit or double-bit uncorrectable error this register stores the 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. 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		Reads return zero and writes have no effect.

4.7.8 TCRAM Wrapper Test Mode Control Register (RAMTEST)

The RAMTEST register, shown in Figure 4-10 and described in Table 4-9, controls the test mode of the TCRAM wrapper.

Figure 4-10. TCRAM Wrapper Test Mode Control Register (RAMTEST) [offset = 0x30]

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
Reserved							TRIGGER	TEST MODE		Reserved		TEST ENABLE			
R-0							R/WP-0	R/WP-00		R-0		R/WP-0101			

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-9. TCRAM Wrapper Test Mode Control Register (RAMTEST) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and writes have no effect.
8	TRIGGER		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 RAMERRSTATUS and RAMUERRADDR registers are in the cleared state.
7–6	TEST MODE		Test Mode. This field selects either equality of inequality testing schemes. If TEST MODE is set to 0x2, 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 0x1, 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		Reads return zero and 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.

Table 4-9. TCRAM Wrapper Test Mode Control Register (RAMTEST) Field Descriptions (Continued)

Bit	Name	Value	Description
		1010	Test mode is enabled.
		All others	Test mode is disabled.

4.7.9 *TCRAM Wrapper Test Mode Vector Register (RAMADDRDECVECT)*

The RAMADDRDECVECT register, shown in Figure 4-11 and described in Table 4-10, is used for testing the redundant address decode and compare logic of the TCRAM wrapper.

Figure 4-11. TCRAM Wrapper Test Mode Vector Register (RAMADDRDECVECT) [offset = 0x38]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAM CHIP SELECT(15–0)															
R/WP-0x0000															

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-10. TCRAM Wrapper Test Mode Vector Register (RAMADDRDEVECT) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26	ECC SELECT		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		Reads return zero and writes have no effect.
15–0	RAM CHIP SELECT		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.

4.7.10 TCRAM Wrapper Parity Error Address Register (RAMPERRADDR)

The RAMPERRADDR register, shown in Figure 4-12 and described in Table 4-11,stores the address for which an address-parity error was detected.

Figure 4-12. TCRAM Wrapper Parity Error Address Register (RAMPERRADDR) [offset = 0x3C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									ADDRESS PARITY ERROR ADDRESS(22–16)						
R-0									R-U						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS PARITY ERROR ADDRESS(15–3)															
R-U													R-0		

R = Read in all modes; WP = write in privileged mode only; *n* = value after reset

Table 4-11. TCRAM Wrapper Parity Error Address Register (RAMPERRADDR) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads return zero and writes have no effect.
22–3	ADDRESS PARITY ERROR ADDRESS		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.
2–0	Reserved		Reads return zero and writes have no effect.

Chapter 5

SPNU489C-February 2012

System and Peripheral Control Registers

This chapter describes the system and peripheral control registers of the Texas Instruments TMS570 architecture.

Topic **Page**

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 0xFFFF FF00 and the end address is 0xFFFF FFFF. The start address of the secondary system module frame is 0xFFFF E100 and the end address is 0xFFFF E1FF. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Figure 5-1 contains a summary figure of the system control registers.

Figure 5-1. Frame 1 System Control Registers Summary

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
0x00 SYSPC1 Page 77	Reserved															
	Reserved															ECP- CLK- FUN
0x04 SYSPC2 Page 78	Reserved															
	Reserved															ECP CLK DIR
0x08 SYSPC3 Page 79	Reserved															
	Reserved															ECP CLK DIN
0x0C SYSPC4 Page 80	Reserved															
	Reserved															ECP CLK DOUT
0x10 SYSPC5 Page 81	Reserved															
	Reserved															ECP CLK SET
0x14 SYSPC6 Page 82	Reserved															
	Reserved															ECP CLK CLR
0x18 SYSPC7 Page 83	Reserved															
	Reserved															ECP CLK ODE

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

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
0x1C SYSPC8 Page 84	Reserved															
	Reserved															ECP CLK PUE
0x20 SYSPC9 Page 85	Reserved															
	Reserved															ECP CLK PS
0x30 CSDIS Page 86	Reserved															
	Reserved								CLK SR7 OFF	CLK SR6 OFF	CLK SR5 OFF	CLK SR4 OFF	CLK SR3 OFF	CLK SR2 OFF	CLK SR1 OFF	CLK SR0 OFF
0x34 CSDISSET Page 87	Reserved															
	Reserved								CLK SR7 OFF SET	CLK SR6 OFF SET	CLK SR5 OFF SET	CLK SR4 OFF SET	CLK SR3 OFF SET	CLK SR2 OFF SET	CLK SR1 OFF SET	CLK SR0 OFF SET
0x38 CSDISCLR Page 88	Reserved															
	Reserved								CLK SR7 OFF CLR	CLK SR6 OFF CLR	CLK SR5 OFF CLR	CLK SR4 OFF CLR	CLK SR3 OFF CLR	CLK SR2 OFF CLR	CLK SR1 OFF CLR	CLK SR0 OFF CLR
0x3C CDDIS Page 89	Reserved															
	Reserved									RTI1 CLK OFF	VCLKA 2 OFF	VCLKA OFF	VCLK2 OFF	VCLKP OFF	HCLK OFF	GCLK OFF
0x40 CDDISSET Page 91	Reserved															
	Reserved									RTI1 CLK OFF SET	VCLKA 2 OFF SET	VCLKA OFF SET	VCLK2 OFF SET	VCLKP OFF SET	HCLK OFF SET	GCLK OFF SET
0x44 CDDISCLR Page 93	Reserved															
	Reserved									RTI1 CLK OFF CLR	VCLKA 2 OFF CLR	VCLKA OFF CLR	VCLK2 OFF CLR	VCLKP OFF CLR	HCLK OFF CLR	GCLK ENA CLR

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

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
0x48 GHVSRC Page 95	Reserved				GHVWAKE(3–0)				Reserved				GHVLPM(3–0)			
	Reserved												GHVSRC(3–0)			
0x4C VCLKASRC Page 97	Reserved															
	Reserved				VCLKA2S(3–0)				Reserved				VCLKA1S(3–0)			
0x50 RCLKSRC Page 99	Reserved															
	Reserved						RTI1DIV(1–0)		Reserved				RTI1SRC(3–0)			
0x54 CSVSTAT Page 101	Reserved															
	Reserved								CLK SR7V	CLK SR6V	CLK SR5V	CLK SR4V	CLK SR3V	CLK SR2V	CLK SR1V	CLK SR0V
0x58 MSTGCR Page 102	Reserved															
	Reserved						ROM_DIV(1–0)		Reserved				MSTGENA(3–0)			
0x5C MINITGCR Page 104	Reserved															
	Reserved												MINITGENA(3–0)			
0x60 MSIENA Page 105	MSIENA[31–16]															
	MSIENA[15–0]															
0x64 MSTFAIL	Reserved															
	Reserved															

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

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
0x68 MSINIGSTAT Page 107	Reserved															
	Reserved							MINI DONE	Reserved							MST DONE
0x6C MINISTAT Page 108	MIDONE[31–16]															
	MIDONE[15–0]															
0x70 PLLCTL1 Page 109	ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
	PLLMUL[15:0]															
0x74 PLLCTL2 Page 111	FM ENA	SPREADINGRATE[8:0]									Reserved	BWADJ[8:4]				
	BWADJ[3:0]				ODPLL			SPR_AMOUNT[8:0]								
0x78 Reserved	Reserved															
	Reserved															
0x7C DIEIDL Page 113	DIEIDL(31–16)															
	DIEIDL(15–0)															
0x80 DIEIDH Page 114	DIEIDH(31–16)															
	DIEIDH(15–0)															
0x8C CLKTEST Page 117	Reserved												CLK_TEST_EN(3–0)			
	Reserved				SEL_GIO_PIN(3–0)				Reserved				SEL_ECP_PIN(3–0)			

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

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
0x90-0xA0 Reserved	Reserved															
	Reserved															
0xAC IMPFTADD Page 121	IMPFTADD(31–16)															
	IMPFTADD(15–0)															
0xB0 SSIR1 Page 122	Reserved															
	SSKEY1								SSDATA1							
0xB4 SSIR2 Page 123	Reserved															
	SSKEY2								SSDATA2							
0xB8 SSIR3 Page 124	Reserved															
	SSKEY3								SSDATA3							
0xBC SSIR4 Page 125	Reserved															
	SSKEY4								SSDATA4							
0xC0 RAMGCR Page 126	Reserved												RAM_DFT_EN(3–0)			
	Reserved	WST_AENA3	Reserved	WST_DENA3	Reserved	WST_AENA2	Reserved	WST_DENA2	Reserved	WST_AENA1	Reserved	WST_DENA1	Reserved	WST_AENA0	Reserved	WST_DENA0
0xC4 BMMCR1 Page 128	Reserved															
	Reserved												MEM SW			

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

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**
0xC8 BMMCR2 Page 129	Reserved															
	Reserved								PRTY_ PBM	PRTY_ HPI	PRTY_ RAM3	PRTY_ RAM2	PRTY_ CRC	PRTY_ PBRG	PRTY_ FLASH	PRTY_ RAM0
0xD0 CLKCNTL Page 132	Reserved				VCLK2R[3–0]				Reserved				VCLKR[3–0]			
	Reserved							PENA	Reserved							
0xD4 ECPCNTRL Page 134	Reserved								ECP-COS	Reserved						
	ECPDIV															
0xD8 DSPGCR	Reserved															
	Reserved															
0xDC DEVCR1 Page 136	Reserved															
	Reserved												DEVPARSEL			
0xE0 SYSECR Page 137	Reserved															
	RESET [1]	RESET [0]	Reserved													
0xE4 SYSESR Page 138	Reserved															
	PO RST	OSC RST	WD RST	Reserved							CPU RST	SW RST	EXT RST	Reserved		Unuse d
0xE8 Reserved	Reserved															
	Reserved															

Figure 5-1. Frame 1 System Control Registers Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xEC GLBSTAT Page 140	Reserved															
	Reserved						FBSLIP	RFSLIP	Reserved							OSC FAIL
0xF0 DEVID Page 142	CP15	ID														TECH
	TECH			I/O	PPAR	Program parity		RECC	Version					1	0	1
0xF4 SSIVEC Page 144	Reserved															
	SSIDATA								SSIVECT[7–0]							
0xF8 SSIF Page 145	Reserved															
	Reserved												SSI_FLAG4	SSI_FLAG3	SSI_FLAG2	SSI_FLAG1
0xFC SSIR1 Page 122	Reserved															
	SSKEY1								SSDATA1							

5.1.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in Figure 5-2 and described in Table 5-1, controls the function of the ECLK pin.

Figure 5-2. SYS Pin Control Register 1 (SYSPC1) [offset = 0x00]

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
Reserved															ECP CLK FUN
R-0															R/W-0

R = Read in all modes; W = write in all modes; *n* = value after reset

Table 5-1. SYS Pin Control Register 1 (SYSPC1) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 guaranteed until 1 ECLK cycle has elapsed after switching into functional mode.**

5.1.2 SYS Pin Control Register 2 (SYSPC2)

The SYSPC2 register, shown in Figure 5-3 and described in Table 5-2, controls whether the pin is an input or an output when in GIO mode.

Figure 5-3. SYS Pin Control Register 2 (SYSPC2) [offset = 0x04]

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
Reserved															ECP CLK DIR
R-0															R/W-0

R = Read; W = Write; -n = value after reset

Table 5-2. SYS Pin Control Register 2 (SYSPC2) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 (offset 0x00).**

5.1.3 SYS Pin Control Register 3 (SYSPC3)

The SYSPC3 register, shown in Figure 5-4 and described in Table 5-3, displays the logic state of the ECLK pin when it is in GIO mode.

Figure 5-4. SYS Pin Control Register 3 (SYSPC3) [offset = 0x08]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="15">Reserved</td><td>ECP
CLK
DIN</td></tr>
<tr><td colspan="15">R-0</td><td>R-U</td></tr>
</table>

R = Read only; -n = value after reset; -U = undefined

Table 5-3. SYS Pin Control Register 3 (SYSPC3) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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).

5.1.4 SYS Pin Control Register 4 (SYSPC4)

The SYSPC4 register, shown in Figure 5-5 and described in Table 5-4, controls the logic level output function of the ECLK pin when it is configured as an output in GIO mode.

Figure 5-5. SYS Pin Control Register 4 (SYSPC4) [offset = 0x0C]

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
Reserved															ECP CLK DOUT
R-0															R/W-0

R = Read; W = Write; -n = value after reset

Table 5-4. SYS Pin Control Register 4 (SYSPC4) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 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 setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00). The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register (offset 0x04).**

5.1.5 SYS Pin Control Register 5 (SYSPC5)

The SYSPC5 register, shown in Figure 5-6 and described in Table 5-5, controls the set function of the ECLK pin when it is configured as an output in GIO mode.

Figure 5-6. SYS Pin Control Register 5 (SYSPC5) [offset = 0x10]

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
Reserved															ECP CLK SET
R-0															R/W-0

R = Read; W = Write; -n = value after reset

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

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00). The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register (offset 0x04).**

5.1.6 SYS Pin Control Register 6 (SYSPC6)

The SYSPC6 register, shown in Figure 5-7 and described in Table 5-6, controls the clear function of the ECLK pin when it is configured as an output in GIO mode.

Figure 5-7. SYS Pin Control Register 6 (SYSPC6) [offset = 0x14]

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
Reserved															ECP CLK CLR
R-0															R/W-0

R = Read; W = Write; -n = value after reset

Table 5-6. SYS Pin Control Register 6 (SYSPC6) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00). The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register (offset 0x04).

5.1.7 SYS Pin Control Register 7 (SYSPC7)

The SYSPC7 register, shown in Figure 5-8 and described in Table 5-7, controls the open drain function of the ECLK pin.

Figure 5-8. SYS Pin Control Register 7 (SYSPC7) [offset = 0x18]

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
Reserved															ECP CLK ODE
R-0															R/W-0

R = Read; W = Write; -n = value after reset

Table 5-7. SYS Pin Control Register 7 (SYSPC7) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 (offset 0Ch) 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 setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00).**

5.1.8 SYS Pin Control Register 8 (SYSPC8)

The SYSPC8 register, shown in Figure 5-9 and described in Table 5-8, controls the pull enable function of the ECLK pin when it is configured as an input in GIO mode.

Figure 5-9. SYS Pin Control Register 8 (SYSPC8) [offset = 0x1C]

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
Reserved															ECP CLK PUE
R-0															R/W-D

R = Read; W = Write; -n = value after reset; D = Device Specific

Table 5-8. SYS Pin Control Register 8 (SYSPC8) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 (offset 0x20).** **Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00). The ECLK pin is placed in input mode by setting the ECPCLKDIR bit to 0 in the SYSPC2 register (offset 0x04).**

5.1.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in Figure 5-10 and described in Table 5-9, controls the pull up/pull down configuration of the ECLK pin when it is configured as an input in GIO mode.

Figure 5-10. SYS Pin Control Register 9 (SYSPC9) [offset = 0x20]

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
Reserved															ECP CLK PS
R-0															R/W-0

R = Read; W = Write; -n = value after reset

Table 5-9. SYS Pin Control Register 9 (SYSPC9) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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 setting the ECPCLKPUE bit to 0 in the SYSPC8 register (offset 0x1C).** **Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register (offset 0x00). The ECLK pin is placed in input mode by setting the ECPCLKDIR bit to 0 in the SYSPC2 register (offset 0x04).**

5.1.10 Clock Source Disable Register (CSDIS)

The CSDIS register shown in Figure 5-11 and described in Table 5-10, controls and displays the state of the device clock sources.

Figure 5-11. Clock Source Disable Register (CSDIS) [offset = 0x30]

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
Reserved								CLK SR7 OFF	CLK SR6 OFF	CLK SR5 OFF	CLK SR4 OFF	CLK SR3 OFF	CLK SR2 OFF	CLK SR1 OFF	CLK SR0 OFF
R-0								R/WP-1	R/WP-1	R/WP-D	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device-specific reset value

Table 5-10. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLKSR[7–0] OFF		Clock source[7–0] off.
		0	Clock source[7–0] is enabled.
		1	Clock source[7–0] is disabled.
			Note: On wakeup, only clock sources 0, 4 and 5 are enabled.

Table 5-11. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source1	PLL1 (FMzPLL)
Clock Source 2	Not Implemented
Clock Source 3	Not Implemented
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High Frequency LPO (Low Power Oscillator) clock
Clock Source 6	PLL2 (FPLL)
Clock Source 7	Not Implemented

Note: Non implemented clock sources should not be enabled or used.

5.1.11 Clock Source Disable Set Register (CSDISSET)

The CSDISSET register shown in Figure 5-12 and described in Table 5-12, sets clock sources to the disabled state.

Figure 5-12. Clock Source Disable Set Register (CSDISSET) [offset = 0x34]

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
Reserved								SET CLK SR7 OFF	SET CLK SR6 OFF	SET CLK SR5 OFF	SET CLK SR4 OFF	SET CLK SR3 OFF	SET CLK SR2 OFF	SET CLK SR1 OFF	SET CLK SR0 OFF
R-0								R/WP-1	R/WP-1	R/WP-0	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device-specific reset value

.

Table 5-12. Clock Source Disable Set Register (CSDISSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	SETCLKSR[7–0] OFF		Set clock source[7–0] to the disabled state.
		0	*Read*– Clock source[7–0] is enabled. *Write*– Clock source[7–0] is unchanged.
		1	*Read*– Clock source[7–0] is disabled. *Write*– Clock source[7–0] 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 register (offset 0x30), the CSDISSET register (offset 0x34) and the CSDISCLR register (offset 0x38).**

Note: A list of the available clock sources is shown in the Clock Sources Table.

5.1.12 Clock Source Disable Clear Register (CSDISCLR)

The CSDISCLR register shown in Figure 5-13 and described in Table 5-12, clears clock sources to the enabled state.

Figure 5-13. Clock Source Disable Clear Register (CSDISCLR) [offset = 0x38]

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
Reserved								CLR CLK SR7 OFF	CLR CLK SR6 OFF	CLR CLK SR5 OFF	CLR CLK SR4 OFF	CLR CLK SR3 OFF	CLR CLK SR2 OFF	CLR CLK SR1 OFF	CLR CLK SR0 OFF
R-0								R/WP-1	R/WP-1	R/WP-D	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device-specific reset value

.

Table 5-13. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLRCLKSR[7–0] OFF		Enables clock source[7–0].
		0	*Read*– Clock source[7–0] is enabled. *Write*– Clock source[7–0] is unchanged.
		1	*Read*– Clock source[7–0] is disabled. *Write*– Clock source[7–0] is cleared to the enabled state. **Note: After a new clock source disable bit is cleared via the CSDISCLR register, the new status of the bit will be reflected in the CSDIS register (offset 0x30), the CSDISSET register (offset 0x34) and the CSDISCLR register (offset 0x38).**

Note: A list of the available clock sources is shown in the Clock Sources Table.

5.1.13 Clock Domain Disable Register (CDDIS)

The CDDIS register shown in Figure 5-14 and described in Table 5-14, controls the state of the clock domains.

Note: All the clock domains are enabled on wakeup.
The application should guarantee that when HCLK and VCLK_sys are turned off through the HCLKOFF bit, the GCLK domain is also turned off.

Figure 5-14. Clock Domain Disable Register (CDDIS) [offset = 0x3C]

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
Reserved									RTI1 CLK OFF	VCLKA2 OFF	VCLKA1 OFF	VCLK2 OFF	VCLKP OFF	HCLK OFF	GCLK OFF
R-0									R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-14. Clock Domain Disable Register (CDDIS) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	RTI1CLKOFF		RTI1CLK (Real Time Interrupt) domain off.
		0	The RTI1CLK domain is enabled.
		1	The RTI1CLK domain is disabled.
5–4	VCLKA[2–1]OFF		VCLKA[2–1] (Flexray and DCAN) domains off.
		0	The VCLKA[2–1] domains are enabled.
		1	The VCLKA[2–1] domains are disabled.
3	VCLK2OFF		VCLK2 (NHET) domain off.
		0	The VCLK2 domain is enabled.
		1	The VCLK2 domain is disabled.
2	VCLKPOFF		VCLK_periph (Peripheral) domain off.
		0	The VCLK_periph domain is enabled.
		1	The VCLK_periph domain is disabled.
1	HCLKOFF		HCLK and VCLK_sys (System Module) domains off.
		0	The HCLK and VCLK_sys domains are enabled.

Table 5-14. Clock Domain Disable Register (CDDIS) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	The HCLK and VCLK_sys domains are disabled.
0	GCLKOFF		GCLK (Cortex R4 CPU) domain off.
		0	The GCLK domain is enabled.
		1	The GCLK domain is disabled.

5.1.14 Clock Domain Disable Set Register (CDDISSET)

This CDDISSET register shown in Figure 5-15 and described in Table 5-15, sets clock domains to the disabled state.

Figure 5-15. Clock Domain Disable Set Register (CDDISSET) [offset = 0x40]

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
Reserved									SET RTI1 CLK OFF	SET VCLKA2 OFF	SET VCLKA1 OFF	SET VCLK2 OFF	SET VCLKP OFF	SET HCLK OFF	SET GCLK OFF
R-0									R/WP-1	R/WP-D	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device Specific

Table 5-15. Clock Domain Disable Set Register (CDDISSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	SETRTI1CLK OFF		Set RTI1CLK (Real Time Interrupt) domain.
		0	*Read*– The RTI1CLK domain is enabled. *Write*– The RTI1CLK domain is unchanged.
		1	*Read*– The RTI1CLK domain is disabled. *Write*– The RTI1CLK domain is set to the disabled state.
5	SETVCLKA2OFF		Set VCLKA2 (Flexray) domain.
		0	*Read*– The VCLKA2 domain is enabled. *Write*– The VCLK2 domain is unchanged.
		1	*Read*– The VCLKA2 domain is disabled. *Write*– The VCLKA2 domain is set to the disabled state.
4	SETVCLKA1OFF		Set VCLKA1 (DCAN) domain.
		0	*Read*– The VCLKA1 domain is enabled. *Write*– The VCLKA1 domain is unchanged.
		1	*Read*– The VCLKA1 domain is disabled. *Write*– The VCLKA1 domain is set to the disabled state.

Table 5-15. Clock Domain Disable Set Register (CDDISSET) Field Descriptions (Continued)

Bit	Name	Value	Description
3	SETVCLK2OFF		Set VCLK2 (NHET) domain.
		0	*Read*– The VCLK2 domain is enabled. *Write*– The VCLK2 domain is unchanged.
		1	*Read* – The VCLKA domain is disabled. *Write*– The VCLK2 domain is set to the disabled state.
2	SETVCLKPOFF		Set VCLK_periph (Peripheral) domain.
		0	*Read*– The VCLK_periph domain is enabled. *Write*– The VCLK_periph domain is unchanged.
		1	*Read* – The VCLK_periph domain is disabled. *Write*– The VCLK_periph domain is set to the disabled state.
1	SETHCLKOFF		Set HCLK and VCLK_sys (System Module) domains.
		0	*Read*– The HCLK and VCLK_sys domains are enabled. *Write*– The HCLK and VCLK_sys domains are unchanged.
		1	*Read*– The HCLK and VCLK_sys domains are disabled. *Write*– The HCLK and VCLK_sys domain are set to the disabled state.
0	SETGCLKOFF		Set GCLK (Cortex R4 CPU) domain.
		0	*Read*– The GCLK domain is enabled. *Write*– The GCLK domain is unchanged.
		1	*Read*– The GCLK domain is disabled. *Write*– The GCLK domain is set to the disabled state.

5.1.15 Clock Domain Disable Clear Register (CDDISCLR)

The CDDISCLR register shown in Figure 5-16 and described in Table 5-16, clears clock domains to the enabled state.

Figure 5-16. Clock Domain Disable Clear Register (CDDISCLR) [offset = 0x44]

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
Reserved									CLR RTI1 CLK OFF	CLR VCLKA2 OFF	CLR VCLKA1 OFF	CLR VCLK2 OFF	CLR VCLKP OFF	CLR HCLK OFF	CLR GCLK OFF
R-0									R/WP-1	R/WP-D	R/WP-0	R/WP-1	R/WP-1	R/WP-1	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = Device Specific

Table 5-16. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–6	CLRRTI1CLK OFF		Clear RTI1CLK (Real Time Interrupt) domain.
		0	*Read*– The RTI1CLK domain is enabled. *Write*– The RTI1CLK domain is unchanged.
		1	*Read*– The RTI1CLK domain is disabled. *Write*– The RTI1CLK domain is cleared to the enabled state.
5	CLRVCLKA2 OFF		Clear VCLKA2 (Flexray) domain.
		0	*Read*– The VCLKA2 domain is enabled. *Write*– The VCLKA2 domain is unchanged.
		1	*Read* – The VCLKA2 domain is disabled. *Write*– The VLCKA2 domain is cleared to the enabled state.
4	CLRVCLKA1OFF		Clear VCLKA1 (DCAN) domain.
		0	*Read*– The VCLKA1 domain is enabled. *Write*– The VCLKA1 domain is unchanged.
		1	*Read*– The VCLKA1 domain is disabled. *Write*– The VCLKA1 domain is cleared to the enabled state.

Table 5-16. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
3	CLRVCLK2OFF		Clear VCLK2 (NHET) domain.
		0	*Read*– The VCLK2 domain is enabled. *Write*– The VCLK2 domain is unchanged.
		1	*Read*– The VCLK2 domain is disabled. *Write*– The VCLK2 domain is cleared to the enabled state.
2	CLRVCLKPOFF		Clear VCLK_periph (Peripheral) domain.
		0	*Read*– The VCLK_periph domain is enabled. *Write*– The VCLK_periph is unchanged.
		1	*Read*– The VCLKP domain is disabled. *Write*– The VCLKP domain is cleared to the enabled state.
1	CLRHCLKOFF		Clear HCLK and VCLK_sys (System Module) domains.
		0	*Read*– The HCLK and VCLK_sys domains are enabled. *Write*– The HCLK and VCLK_sys domains are unchanged.
		1	*Read* – The HCLK and VCLK_sys domains are disabled. *Write*– The HCLK and VCLK_sys domains are cleared to the enabled state.
0	CLRGCLKOFF		Clear GCLK (Cortex R4 CPU) enable.
		0	*Read*– The GCLK domain is enabled. *Write*– The GCLK domain is unchanged.
		1	*Read* – The GCLK domain is disabled. *Write*– The GCLK domain is cleared to the enabled state.

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

The GHVSRC register is shown in Figure 5-17 and described in Table 5-17, controls the clock source configuration for the GCLK, HCLK, VCLK and VCLK2 clock domains.

Figure 5-17. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) [offset = 0x48]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				GHVWAKE(3–0)				Reserved				HVLPM(3–0)			
R-0				R/WP-0000				R-0				R/WP-0000			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												GHVSRC(3–0)			
R-0												R/WP-0000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-17. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
27–24	GHVWAKE[3–0]		GCLK, HCLK, VCLK, VCLK2 source on wakeup.
		0000	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0001	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0010	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0011	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0100	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0101	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0110	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		0111	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		1000–1111	Reserved - These values should not be used.
23–20	Reserved		Reads return zero and writes have no effect.

Table 5-17. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions (Continued)

Bit	Name	Value	Description
19–16	HVLPM[3–0]		HCLK, VCLK, VCLK2 source on wakeup when GCLK is turned off.
		0000	Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0001	Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0010	Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0011	Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0100	Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0101	Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0110	Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup.
		0111	Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup.
		1000–1111	Reserved - These values should not be used.
15–4	Reserved		Reads return zero and writes have no effect.
3–0	GHVSRC[3–0]		GCLK, HCLK, VCLK, VCLK2 current clock 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.**
		0000	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2.
		0001	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2.
		0010	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2.
		0011	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2.
		0100	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2.
		0101	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2.
		0110	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2.
		0111	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2.
		1000–1111	Reserved - These values should not be used.

Note: Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in the Clock Sources Table.

5.1.17 Peripheral Asynchronous Clock Source Register (VCLKASRC)

The VCLKASRC register shown in Figure 5-18 and described in Table 5-18, sets the clock source for the asynchronous peripheral clock domains to be configured to run from a specific clock source.

Figure 5-18. Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 0x4C]

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
Reserved				VCLKA2S(3–0)				Reserved				VCLKA1S(3–0)			
R-0				R/WP-1001				R-0				R/WP-1001			

R = Read in all modes; WP = Write in privileged mode only; -n values after reset

Table 5-18. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return zero and writes have no effect.
11–8	VCLKA2S[3–0]		Peripheral asynchronous clock2 source.
		0000	Clock source0 is the source for peripheral asynchronous clock2.
		0001	Clock source1 is the source for peripheral asynchronous clock2.
		0010	Clock source2 is the source for peripheral asynchronous clock2.
		0011	Clock source3 is the source for peripheral asynchronous clock2.
		0100	Clock source4 is the source for peripheral asynchronous clock2.
		0101	Clock source5 is the source for peripheral asynchronous clock2.
		0110	Clock source6 is the source for peripheral asynchronous clock2.
		0111	Clock source7 is the source for peripheral asynchronous clock2.
		1000–1111	VCLK is the source for peripheral asynchronous clock2.
7–4	Reserved		Reads return zero and writes have no effect.

Table 5-18. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	VCLKA1S[3–0]		Peripheral asynchronous clock1 source.
		0000	Clock source0 is the source for peripheral asynchronous clock1.
		0001	Clock source1 is the source for peripheral asynchronous clock1.
		0010	Clock source2 is the source for peripheral asynchronous clock1.
		0011	Clock source3 is the source for peripheral asynchronous clock1.
		0100	Clock source4 is the source for peripheral asynchronous clock1.
		0101	Clock source5 is the source for peripheral asynchronous clock1.
		0110	Clock source6 is the source for peripheral asynchronous clock1.
		0111	Clock source7 is the source for peripheral asynchronous clock1.
		1000–1111	VCLK is the source for peripheral asynchronous clock1.

Note: Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in the Clock Sources Table.

5.1.18 RTI Clock Source Register (RCLKSRC)

The RCLKSRC register shown in Figure 5-19 and described in Table 5-19, 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.

Figure 5-19. RTI Clock Source Register (RCLKSRC) [offset = 0x50]

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
Reserved						RTI1DIV(1–0)		Reserved				RTI1SRC(3–0)			
R-0						R/WP-01		R-0				R/WP-1001			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-19. RTI Clock Source Register (RCLKSRC) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9–8	RTI1DIV[1–0]		RTI clock1 divider.
		00	RTI1CLK divider value is 1.
		01	RTI1CLK divider value is 2.
		10	RTI1CLK divider value is 4.
		11	RTI1CLK divider value is 8.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	RTI1SRC[3–0]		RTI clock1 source.
		0000	Clock source0 is the source for RTI1CLK.
		0001	Clock source1 is the source for RTI1CLK.
		0010	Clock source2 is the source for RTI1CLK.
		0011	Clock source3 is the source for RTI1CLK.
		0100	Clock source4 is the source for RTI1CLK.
		0101	Clock source5 is the source for RTI1CLK.
		0110	Clock source6 is the source for RTI1CLK.

Table 5-19. RTI Clock Source Register (RCLKSRC) Field Descriptions (Continued)

Bit	Name	Value	Description
		0111	Clock source7 is the source for RT1ICLK.
		1000–1111	VCLK is the source for RTI1CLK.

Note: A list of the available clock sources is shown in the Clock Sources Table.

5.1.19 *Clock Source Valid Status Register (CSVSTAT)*

The CSVSTAT register shown in Figure 5-20 and described in Table 5-20, indicates the status of usable clock sources.

Figure 5-20. Clock Source Valid Status Register (CSVSTAT) [offset = 0x54]

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
Reserved								CLK SR7 V	CLK SR6 V	CLK SR5 V	CLK SR4 V	CLK SR3 V	CLK SR2 V	CLK SR1 V	CLK SR0 V
R-0								R-D	R-D	R-D	R-D	R-D	R-D	R-D	R-D

R = Read all modes; -n = Value after power-up reset; D = Device Specific

Table 5-20. Clock Source Valid Register (CSVSTAT) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CLKSR[7–0]V		Clock source[7–0] valid.
		0	Clock source[7–0] is not valid.
		1	Clock source[7–0] is valid. **Note: If the valid bit of the source of a clock domain is not set (i.e., the clock source is not fully stable), the respective clock domain is disabled by the Global Clock Module (GCM).**

Note: A list of the available clock sources is shown in the Clock Sources Table.

5.1.20 Memory Self-Test Global Control Register (MSTGCR)

The MSTGCR register shown in Figure 5-21 and described in Table 5-21, controls several aspects of the PBIST (Programmable Built-In Self Test) memory controller. More information about PBIST can be found in the PBIST chapter.

Figure 5-21. Memory Self-Test Global Control Register (MSTGCR) [offset = 0x58]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">MBIST_ALGSEL(7:0)</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R/WP-00000000</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="2">ROM_DIV(1:0)</td><td colspan="4">Reserved</td><td colspan="4">MSTGENA(3:0)</td></tr>
<tr><td colspan="6">R-0</td><td colspan="2">R/WP-00</td><td colspan="4">R-0</td><td colspan="4">R/WP-0101</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-21. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zero and writes have no effect.
23-16	MBIST_ALGSEL		Selects different test algorithm for PBIST
		00000000	No Algorithm selected.
		00000001	Checkerboard has been selected.
		00000010	March 13N with background of all 0s and all 1s has been selected.
		00000100	March 11N with background of hex 5 and As has been selected.
		00001000	March 13N with backgrounds of hex 3 and Cs, hex 0F and F0s, and 69 and 96s has been selected.
		00010000	PMOS Open Address Decode has been selected.
		00100000	No Algorithm selected.
		01000000	No Algorithm selected.
		10000000	No Algorithm selected.
15-10	Reserved		Reads return zero and writes have no effect.
9–8	ROM_DIV[1:0]		Prescaler divider bits for BIST ROM clock source.
		00	The BIST ROM clock source is HCLK divided by 1. PBIST will reset for 16 VBUS cycles.
		01	The BIST ROM clock source is HCLK divided by 2. PBIST will reset for 32 VBUS cycles.

Table 5-21. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions (Continued)

Bit	Name	Value	Description
		10	The BIST ROM clock source is HCLK divided by 4. PBIST will reset for 64 VBUS cycles.
		11	The BIST ROM clock source is HCLK divided by 8. PBIST will reset for 96 VBUS cycles.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	MSTGENA[3:0]		Memory self-test controller global enable key **Note: Enabling the MSTGENA key will generate a reset to the state machine of the selected PBIST controller.**
		1010	Memory self-test controller is enabled.
		Others	Memory self-test controller is disabled. **Note: It is recommended that a value of 0101b 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.**

5.1.21 *Memory Hardware Initialization Global Control Register (MINITGCR)*

The MINITGCR register shown in Figure 5-22 and described in Table 5-22, enables automatic hardware memory initialization.

Figure 5-22. Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 0x5C]

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
Reserved												MINITGENA(3–0)			
R-0												R/WP-0101			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-22. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	MINITGENA[3–0]		Memory hardware initialization global enable key.
		1010	Global memory hardware initialization is enabled
		Others	Global memory hardware initialization is disabled **Note: It is recommended that a value of 0101b be used to disable memory hardware initialization. This value will give maximum protection from an event that would inadvertently enable the controller.** **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.**

5.1.22 Memory Self-Test/Initialization Enable Register (MSIENA)

The MSIENA register shown in Figure 5-23 and described in Table 5-23 enables PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization. See the device data sheet for memory bit mapping.

Figure 5-23. Memory Self-Test/Initialization Enable Register (MSIENA) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIENA[31–16]															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIENA[15–0]															
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-23. Memory Self-Test/Initialization Enable Register (MSIENA) Field Descriptions

Bit	Name	Value	Description
31–0	MSIENA[31–1]		Memory initialization enable.

		0	*In memory self-test mode (MSTGENA = 1010):*
			Reserved.
			In memory Initialization mode (MINITGENA = 1010):
			Memory module [31–1] auto hardware initialization is disabled.

		1	*In memory self-test mode (MSTGENA = 1010):*
			Reserved.
			In memory Initialization mode (MINITGENA = 1010):
			Memory module[31–1] auto hardware initialization is enabled.

			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.
31–0	MSIENA[0]		Memory Self-Test (PBIST) controller and memory initialization enable. In memory self-test mode, this bit should be set before enabling the global memory self-test controller key (MSTGENA) in the MSTGCR register (offset 0x58). 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 a 1010b to any other value) will reset all the MSIENA[31–0] bits to their default values.

Table 5-23. Memory Self-Test/Initialization Enable Register (MSIENA) Field Descriptions (Continued)

Bit	Name	Value	Description

		0	*In memory self-test mode (*MSTGENA *= 1010):*
			PBIST controller is disabled.
			*In memory Initialization mode (*MINITGENA *= 1010):*
			Memory module [0] auto hardware initialization is disabled.

		1	*In memory self-test mode (*MSTGENA *= 1010):* PBIST controller is enabled. *In memory Initialization mode (*MINITGENA *= 1010):* Memory module[0] auto hardware initialization is enabled. **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.**

5.1.23 MSTC Global Status Register (MSTCGSTAT)

The MSTCGSTAT register shown in Figure 5-24 and described in Table 5-24, shows the status of the memory hardware initialization and the memory self-test.

Figure 5-24. MSTC Global Status Register (MSTCGSTAT) [offset = 0x68]

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
Reserved							MINI DONE	Reserved							MST DONE
R-0							R-0	R-0							R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-24. MSTC Global status register (MSTCGSTAT) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and writes have no effect.
8	MINIDONE		Memory hardware initialization complete status. **Note: Disabling the MINITGENA key (By writing any value other than 1010b) will clear the MINIDONE status bit to 0.** **Note: Individual memory initialization status is shown in the MINISTAT register (offset 0x6C).**
		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*– A write of 1 has no effect.
7–1	Reserved		Reads return zero and writes have no effect.
0	MSTDONE		Memory self-test run complete status. **Note: Disabling the MSTGENA key (by writing from a 1010b 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.

5.1.24 Memory Hardware Initialization Status Register (MINISTAT)

The MINISTAT register shown in Figure 5-25 and described in Table 5-25 indicates the status of hardware memory initialization.

Figure 5-25. Memory Hardware Initialization Status Register (MINISTAT) [offset = 0x6C]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">MIDONE[31–16]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">MIDONE[15–0]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-25. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions

Bit	Name	Value	Description
31–0	MIDONE[31–0]		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 a 1010b to any other value) will reset all the individual status bits to 0.**

5.1.25 PLL Control Register 1 (PLLCTL1)

The PLLCTL1 register shown in Figure 5-26 and described in Table 5-26, controls the output frequency of PLL1 (Clock Source 1 - FMzPLL). It also controls the behavior of the device if a PLL slip or oscillator failure is detected. More information about the FMzPLL can be found in the PLL chapter.

Figure 5-26. PLL Control Register 1 (PLLCTL1) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
R/WP-0	R/WP-01		R/WP-1111					R/WP-0	R-0	R/WP-000010					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLMUL[15:0]															
R/WP–0101111100000000															

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = device specific

Table 5-26. PLL Control Register 1 (PLLCTL1) Field Descriptions

Bit	Name	Value	Description
31	Reset on PLL Slip (ROS)		
		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	Bypass on PLL Slip (BPOS)		
		10	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		other	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	PLL Output Clock Divider (PLLDIV)		R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post\_ODCLK}/R$
		0x00	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/1$
		0x01	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/2$
		:	continues in sequence
		0x1F	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/32$

Table 5-26. PLL Control Register 1 (PLLCTL1) Field Descriptions (Continued)

Bit	Name	Value	Description
23	Reset On Oscillator Fail (ROF)		
		0	Do not reset system when oscillator is out of range.
		1	Reset when oscillator is out of range.
22	Reserved		Read/Write, but value has no effect on PLL operation.
21-16	Reference Clock Divider (REFCLKDIV)		NR = REFCLKDIV + 1 $f_{INT\ CLK} = f_{OSCIN}$/NR
		0x00	$f_{INT\ CLK} = f_{OSCIN}/1$
		0x01	$f_{INT\ CLK} = f_{OSCIN}/2$
		:	continues in sequence
		0x3F	$f_{INT\ CLK} = f_{OSCIN}/64$
15-0	PLL Multiplication Factor (PLLMUL)		Valid multiplication factors are from 92 to 184. NF = (PLLMUL / 256) + 1 $f_{VCO\ CLK} = f_{INT\ CLK}$x NF
		0x5B00	$f_{VCO\ CLK} = f_{INT\ CLK}$x 92
		0x5C00	$f_{VCO\ CLK} = f_{INT\ CLK}$x 93
		:	continues in sequence
		0xB700	$f_{VCO\ CLK} = f_{INT\ CLK}$x 184

5.1.26 PLL Control Register 2 (PLLCTL2)

The PLLCTL2 register shown in Figure 5-27 and described in Table 5-27, controls the modulation characteristics and the output divider of the PLL.

> **Note: Modulation Frequency and Depth Setting Constraints**
> There are several combinations of the modulation depth and modulation frequency that are not allowed. Some of these settings effect the PLL even when frequency modulation is not enabled. Refer the device data sheet to identify these combinations to avoid PLL malfunction.

Figure 5-27. PLL Control Register 2 (PLLCTL2) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FM ENA	SPREADINGRATE[8:0]									Reserved	BWADJ[8:4]				
R/WP-0	R/WP-111111111									R-0	R/WP-00000				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWADJ[3:0]				ODPLL			SPR_AMOUNT[8:0]								
R/WP-0111				R/WP-001			R/WP-000000000								

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = device specific

Table 5-27. PLL Control Register 2 (PLLCTL2) Field Descriptions

Bit	Name	Value	Description
31	Frequency Modulation Enable (FMENA)		
		0	Disable frequency modulation
		1	Enable frequency modulation
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1 $f_{mod} = f_s = f_{INT\ CLK}/(2*NS)$
		0x000	$f_{mod} = f_s = f_{INT\ CLK}/(2*1)$
		0x001	$f_{mod} = f_s = f_{INT\ CLK}/(2*2)$
		:	continues in sequence
		0x1FF	$f_{mod} = f_s = f_{INT\ CLK}/(2*512)$
21	Reserved		Read/Write, but value has no effect on PLL operation.

Table 5-27. PLL Control Register 2 (PLLCTL2) Field Descriptions (Continued)

Bit	Name	Value	Description
20-12	Bandwidth Adjustment (BWADJ)		NB = BWADJ + 1 $f_{BW} = f_{nom\_BW}/NB$
		0x007	$f_{BW} = f_{nom\_BW}/8$ (must be set to this value in non-modulation mode)
		0x008	$f_{BW} = f_{nom\_BW}/9$
		:	continues in sequence
		0x0FF	$f_{BW} = f_{nom\_BW}/256$
11-9	Internal PLL Output Divider (ODPLL)		OD = ODPLL + 1 $f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$ These bits must be changed before the PLL is enabled
		0x0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/1$
		0x1	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/2$
		:	continues in sequence
		0x7	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/8$
8-0	Spreading Amount (SPR_AMOUNT)		NV = SPR_AMOUNT + 1
		0x000	NV = 1
		0x001	NV = 2
		:	continues in sequence
		0x1FF	NV = 512

5.1.27 Die Identification Register Lower Word (DIEIDL)

The DIEIDL register shown in Figure 5-28 and described in Table 5-28, contains information about the die lot number, wafer number and X, Y wafer coordinates.

Figure 5-28. Die Identification Register, Lower Word (DIEIDL) [offset = 0x7C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOT # (LOWER 10 BITS)										WAFER #					
R-D										R-D					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y WAFER COORDINATE								X WAFER COORDINATE							
R-D								R-D							

R = Read only; -X = Value unchanged after reset; -D = device specific

Table 5-28. Die Identification Register, Lower Word (DIEIDL) Field Descriptions

Bit	Name	Value	Description
31–22	LOT # (Lower 10 Bits)	-	These read only bits contain the lower 10 bits of the device lot number.
21–16	WAFER #	-	These read only bits contain the wafer number of the device.
15–8	Y WAFER COORDINATE	-	These read only bits contain the Y wafer coordinate of the device
7–0	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.

5.1.28 Die Identification Register Upper Word (DIEIDH)

The DIEIDH register shown in Figure 5-29 and described in Table 5-29., contains information about the die lot number.

Figure 5-29. Die Identification Register, Upper Word (DIEIDH) [offset = 0x80]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		LOT # (UPPER 14 BITS)													
R-D															

R = Read only; -D = Value is device dependent

Table 5-29. Die Identification Register, Upper Word (DIEIDH) Field Descriptions

Bit	Name	Value	Description
31–14	RESERVED	-	These read only bits are reserved.
13–0	LOT # (Upper 14 Bits)	-	This read-only register contains the upper 14 bits of 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.

5.1.29 LPO/Clock Monitor Control Register (LPOMONCTL)

The LPOMONCTL register shown in Figure 5-30 and described in Table 5-30, controls the Low Frequency (Clock Source 4) and High Frequency (Clock Source 5) Low Power Oscillator's (LPO) trim values.

Note: This register is for test and debug use only!
Modifying the contents of this register will effect the behavior of clock monitor circuitry.

Figure 5-30. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BIAS ENABLE	Reserved							
R-0							R/WP-1	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HFTRIM(3–0)				Reserved				LFTRIM(3–0)			
R-0				R/WP-1000				R-0				R/WP-1000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-30. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and writes have no effect.
24	BIAS ENABLE		Low Power Oscillator (LPO) Bias enable.
		0	The bias circuit is disabled and the LPO is in low power mode.
		1	The LPO is enabled.
11–8	HFTRIM[3–0]		High frequency oscillator trim value. This four-bit value is used to center the HF oscillator's frequency. **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 values below are the ratio: f / fo
		0000	50.00%
		0001	56.25%
		0010	62.50%
		0011	68.75%
		0100	75.00%
		0101	81.25%
		0110	87.50%
		1000	100.00%

Table 5-30. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (Continued)

Bit	Name	Value	Description
		1001	106.25%
		1010	112.50%
		1011	118.75%
		1100	125.00%
		1101	131.25%
		1110	137.50%
		1111	143.75%
7–4	Reserved		Reads return zero and writes have no effect.
3–0	LFTRIM[3–0]		Low frequency oscillator trim value. This four-bit value is used to center the LF oscillator's frequency. **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 values below are the ratio: f / fo
		0000	50.00%
		0001	56.25%
		0010	62.50%
		0011	68.75%
		0100	75.00%
		0101	81.25%
		0110	87.50%
		1000	100.00%
		1001	106.25%
		1010	112.50%
		1011	118.75%
		1100	125.00%
		1101	131.25%
		1110	137.50%
		1111	143.75%

5.1.30 Clock Test Register (CLKTEST)

The CLKTEST register shown in Figure 5-31 and described in Table 5-31, 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!

Figure 5-31. Clock Test Register (CLKTEST) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					ALTLIMP CLOCK ENABLE	RANGE DET CTRL	RANGE DET ENA SSEL	Reserved				CLK_TEST_EN(3–0)			
R-0					R/WP-0	R/WP-0	R/WP-0	R-0				R/WP-1010			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SEL_ECP_PIN(3–0)			
R-0												R/WP-0000			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-31. Clock Test Register (CLKTEST) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads return zero and 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	RANGEDETENSSEL		Selects range detection enable. This bit resets asynchronously on nRST.
		0	The range detect enable is generated by the hardware in the clock monitor wrapper.

Table 5-31. Clock Test Register (CLKTEST) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	The range detect enable is controlled by the RANGEDETCTRL bit (Bit 25) of the CLKTEST register.
20–16	CLK_TEST_EN[3–0]		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 1in the SYSPC1 register (offset 0x00).**
		0101	Clock going to ECLK pin is enabled.
		Others	Clock going to ECLK pin is disabled.
15–4	Reserved		Reads return zero and writes have no effect.
3–0	SEL_ECP_PIN[3–0]		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 (offset 0x54).** **Note: The ECLK pin does not support output frequencies greater than 80 MHz. See the device data sheet for more information.**
		0000	Oscillator clock
		0001	PLL clock
		0010	Not Implemented
		0011	External clock applied on pin GIOB[0]
		0100	Low frequency LPO (Low Power Oscillator) clock
		0101	High frequency clock LPO (Low Power Oscillator) clock
		0110	Not Implemented
		0111	Not Implemented
		1000	GCLKMCLK
		1001	RTI1CLK
		1010	Not Implemented
		1011	VCLKA1
		1100	VCLKA2
		1101–1111	Not Implemented

Note:
Non implemented clock sources should not be enabled or used.

5.1.31 Imprecise Fault Status Register (IMPFASTS)

The IMPFASTS register shown in Figure 5-32 and described in Table 5-32, displays information about imprecise aborts that have occurred. An imprecise abort is an abort in which the initiator of a transaction cannot determine the exact transaction that has generated the abort. Please see the ARM Cortex R4 User Guide for more information about imprecise aborts.

Figure 5-32. Imprecise Fault Status Register (IMPFASTS) [offset = 0xA8]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">MASTERID(7–0)</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">Reserved</td><td>EMIFA</td><td>NCBA</td><td>VBUSA</td><td colspan="7">Reserved</td><td>ATYPE</td></tr>
<tr><td colspan="5">R-0</td><td>R-0</td><td>R-0</td><td>R-0</td><td colspan="7">R-0</td><td>RC-0</td></tr>
</table>

R = Read only; C = Clear by reading; -0 value after power-up reset

.

Table 5-32. Imprecise Fault Status Register (IMPFASTS) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zero and writes have no effect.
23–16	MASTERID[7–0]	0x00–0xFF	Master ID. This register indicates which master is responsible for the imprecise abort. The master ID value depends on device implementation– please refer to the device-specific data sheet for valid MASTERID value. **Notes:** • **These bits are only updated when an imprecise abort occurs** • **These bits are cleared to 0x00 only on power-up reset. The value of these bits remains unchanged after all other resets.**
15–11	Reserved		Reads return zero and 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-up reset. The value of this register remains unchanged after all other resets.**
		0	A NCBA is not responsible for the last imprecise abort.

Table 5-32. Imprecise Fault Status Register (IMPFASTS) Field Descriptions (Continued)

Bit	Name	Value	Description
		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-up 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		Reads return zero and 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-up reset. The value of this bit remains unchanged after all other resets**
		0	The last abort generated was a precise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were not updated.
		1	The last abort generated was an imprecise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were updated. **Note: Once ATYPE is set, the IMPFAWADD and IMPFASTS bits are not updated by subsequent ABORT signals.**

Note:
The DMA, DMM, and the peripheral master port will also generate an imprecise abort to the CPU when writing to the peripheral region or to the EMIF region. This will be indicated in the Master ID field of this register.

5.1.32 Imprecise Fault Address Register (IMPFTADD)

This IMPFTADD register shown in Figure 5-33 and described in Table 5-33, shows the address at which an imprecise abort occurred.

Figure 5-33. Imprecise Fault Write Address Register (IMPFTADD) [offset = 0xAC]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPFTADD															
R-0															

R = Read only; -n = value after power-up reset

Table 5-33. Imprecise Fault Write Address R (IMPFTADD) Field Descriptions

Bit	Name	Value	Description
31–0	IMPFTADD[31–0]	0–0xFFFF FFFF	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 0x0000 0000 only on power-up reset. The value of this register remains unchanged after all other resets.**

5.1.33 System Software Interrupt Request 1 Register (SSIR1)

The SSIR1 register shown in Figure 5-34 and described in Table 5-34, is used for software interrupt generation.

Figure 5-34. System Software Interrupt Request 1 Register (SSIR1) [offset = 0xB0]

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
SSKEY1								SSDATA1							
R/W-0								R/W-0							

R = Read; W = Write in all modes; -*n* = Value after reset

.

Table 5-34. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY1[7–0]	0x00–0xFF	System software interrupt request key. A 0x75 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 (0x75). The SSDATA1 field can only be written into if the write data into this field, the SSKEY1 field, matches the key (0x75).
7–0	SSDATA1[7–0]	0x00–0xFF	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 (0x75); therefore, byte writes cannot be performed on the SSDATA1 field.

Note:
This register is mirrored at offset 0xFC for compatibility reasons.

5.1.34 System Software Interrupt Request 2 Register (SSIR2)

The SSIR2 register shown in Figure 5-35 and described in Table 5-35, is used for software interrupt generation.

Figure 5-35. System Software Interrupt Request 2 Register (SSIR2) [offset = 0xB4]

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
SSKEY2								SSDATA2							
R/W-0								R/W-0							

R = Read; W = Write; -n = Value after reset

.

Table 5-35. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY2[7–0]	0x00-0xFF	System software interrupt2 request key. A 0x84 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 (0x84). The SSDATA2 field can only be written into if the write data into this field, the SSKEY2 field, matches the key (0x84).
7–0	SSDATA2[7–0]	0x00–0xFF	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 (0x84); therefore, byte writes cannot be performed on the SSDATA2 field.

5.1.35 System Software Interrupt Request 3 Register (SSIR3)

The SSIR3 register shown in Figure 5-36 and described in Table 5-36, is used for software interrupt generation.

Figure 5-36. System Software Interrupt Request 3 Register (SSIR3) [offset = 0xB8]

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
SSKEY3								SSDATA3							
R/W--0								R/W--0							

R = Read; W = Write; U = Undefined; -*n* = Value after reset

.

Table 5-36. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY3[7–0]	0x00–0xFF	System software interrupt request key. A 0x93 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 (0x93). The SSDATA3 field can only be written into if the write data into this field, the SSKEY3 field, matches the key (0x93).
7–0	SSDATA3[7–0]	0x00–0xFF	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 (0x93); therefore, byte writes cannot be performed on the SSDATA3 field.

5.1.36 System Software Interrupt Request 4 Register (SSIR4)

The SSIR4 register shown in Figure 5-37 and described in Table 5-37, is used for software interrupt generation.

Figure 5-37. System Software Interrupt Request 4 Register (SSIR4) [offset = 0xBC]

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
SSKEY4								SSDATA4							
R/W--0								R/W--0							

R = Read; W = Write; U = Undefined; -*n* = Value after reset

.

Table 5-37. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSKEY4[7–0]	0x00–0xFF	System software interrupt2 request key. A 0xA2 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 (0xA2). The SSDATA4 field can only be written into if the write data into this field, the SSKEY4 field, matches the key (0xA2).
7–0	SSDATA4[7–0]	0x00–0xFF	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 (0xA2); therefore, byte writes cannot be performed on the SSDATA4 field.

5.1.37 RAM Control Register (RAMGCR)

The RAMGCR register shown in Figure 5-38 and described in Table 5-38, 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 5-38. RAM Control Register (RAMGCR) [offset = 0xC0]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">RAM_DFT_EN(3:0)</td></tr>
<tr><td colspan="12">R-0</td><td colspan="4">R/WP-0101</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="13">Reserved</td><td>WST AENA 0</td><td>Reserve d</td><td>WST DENA 0</td></tr>
<tr><td colspan="13">R-0</td><td>R/WP-0</td><td>R-0</td><td>R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-38. RAM Control Register (RAMGCR) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads return zero and writes have no effect.
19–16	RAM_DFT_EN[3–0]		Functional mode RAM DFT (Design For Test) port enable key. **Note: For TI internal use only!**
		1010	RAM DFT port is enabled.
		Others	RAM DFT port is disabled **Note: It is recommended that a value of 0101b 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.**
31–3	Reserved		Reads return zero, writes should not be performed to these bits.
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		Reads return zero and writes have no effect.
0	WST_DENA0		eSRAM data phase wait state enable bit.

Table 5-38. RAM Control Register (RAMGCR) Field Descriptions (Continued)

Bit	Name	Value	Description
		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.

5.1.38 Bus Matrix Module Control Register 1 (BMMCR1)

The BMMCR1 register shown in Figure 5-39 and described in Table 5-39, allows RAM and Program (Flash) memory addresses to be swapped.

Figure 5-39. Bus Matrix Module Control Register 1 (BMMCR) [offset = 0xC4]

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
Reserved												MEMSW			
R-0												R/WP-1010			

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-39. Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	MEMSW[3–0]		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 MPM_ENA bit in the MMUGCR register (offset 0xCC).**
		1010	Default memory map: Program memory (Flash) starts at address 0x0000 0000. eSRAM starts at address 0x0800 0000.
		0101	Swapped memory map–: eSRAM starts at address 0x0000 0000. Program memory (Flash) starts at address 0x0800 0000.
		Any other value	The device memory map is unchanged.

5.1.39 Bus Matrix Module Control Register2 (BMMCR2)

The BMMCR2 register shown in Figure 5-40 and described in Table 5-40, allows the Bus Matrix Module (BMM) arbitration priority to be configured.

Figure 5-40. Bus Matrix Module Control Register2 (BMMCR2) [offset = 0xC8]

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
Reserved								PRTY_PBM	PRTY_HPI	PRTY_RAM3	PRTY_RAM2	PRTY_CRC	PRTY_PRG	PRTY_FLASH	PRTY_RAM0
R-0								R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-40. Bus Matrix Module Control Register2 (BMMCR2) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7	PRTY_PBM		Arbitration priority to PBM - Peripheral bus matrix.
		0	Fixed priority is used.
		1	Round robin priority is used.
6	PRTY_HPI		Arbitration priority to Coprocessor Interface.
		0	Fixed priority is used.
		1	Round robin priority is used.
5	PRTY_RAM3		Arbitration priority to eSRAM3.
		0	Fixed priority is used.
		1	Round robin priority is used.
4	PRTY_RAM2		Arbitration priority to eSRAM2.
		0	Fixed priority is used.
		1	Round robin priority is used.
3	PRTY_CRC		Arbitration priority to CRC.
		0	Fixed priority is used.
		1	Round robin priority is used.

Table 5-40. Bus Matrix Module Control Register2 (BMMCR2) Field Descriptions (Continued)

Bit	Name	Value	Description
2	PRTY_PRG		Arbitration priority to peripheral bridge.
		0	Fixed priority is used.
		1	Round robin priority is used.
1	PRTY_FLASH		Arbitration priority to FLASH.
		0	Fixed priority is used.
		1	Round robin priority is used.
0	PRTY_RAM0		Arbitration priority to eSRAM0.
		0	Fixed priority is used.
		1	Round robin priority is used.

5.1.40 MMU Global Control Register (MMUGCR)

The MMUGCR register shown in Figure 5-41 and described in Table 5-41, allows the memory protection mode to be configured on certain TMS570 devices. It also allows a reset to the Cortex R4 CPU to be generated.

Figure 5-41. MMU Global Control Register (MMUGCR) [offset = 0xCC]

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
Reserved															CPU RESET
R-0															R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = value after power-up reset

.

Table 5-41. MMU Global Control Register (MMUGCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and writes have no effect.
0	CPU RESET		This bit’s function on this device is to trigger a Cortex R4 CPU reset. When this bit is changed from a 0 to a 1 or from a 1 to a 0 a CPU reset is triggered. **Note: On certain TMS570 devices, this bit controls the memory protection configuration to use either the MMU or MPU. This device only supports an MPU; therefore the only function this bit has on this device is to trigger a CPU reset.** **Note: The CPU reset is specific to the CPU only, not to the system (system reset remains inactive during CPU reset).**
		0	The system is configured to use the MMU, or the MPU if the system has only an MPU.
		1	The system is configured to use the MPU.

5.1.41 Clock Control Register (CLKCNTL)

The CLKCNTL register shown in Figure 5-42 and described in Table 5-42, 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. Frequency of VCLK2 must also be an integer multiple of the frequency of VCLK. That is, f(VCLK2) = f(HCLK)/2 and f(VCLK) = f(HCLK)/3 is not allowed. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. The application must configure the VCLK2 ratio, read back the contents of the CLKCNTL register, and then configure the VCLK ratio.

Figure 5-42. Clock Control Register (CLKCNTRL) [offset = 0xD0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				VCLK2R				Reserved				VCLKR			
R-0				R/WP-0001				R-0				R/WP-0001			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							PENA	Reserved							
R-0							R/WP-0	R-0							

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-42. Clock Control Register (CLKCNTRL) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
27–24	VCLK2R[3–0]		VBUS clock2 ratio.
		0000	The VCLK2 speed is HCLK divided by 1.
		. . .	. . .
		1111	The VCLK2 speed is HCLK divided by 16.
23–20	Reserved		Reads return zero and writes have no effect.
19–16	VCLKR[3–0]		VBUS clock ratio.
		0000	The VCLK speed is HCLK divide by 1.
		. . .	. . .
		1111	The VCLK speed is HCLK divided by 16.
15–9	Reserved		Reads return zero and writes have no effect.
8	PENA		Peripheral enable bit. The application must set this bit before accessing any peripheral

Table 5-42. Clock Control Register (CLKCNTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset. **Note: The peripherals' registers/memory can only be accessed after this bit is set to 1.**
7–0	Reserved		Reads return zero and writes have no effect.

5.1.42 ECP Control Register (ECPCNTL)

The ECP register shown in Figure 5-43 and described in Table 5-43, 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 (offset 0x00) before a clock source will be visible on the ECLK pin.

Figure 5-43. ECP Control Register (ECPCNTL) [offset = 0xD4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							ECPS-SEL	ECP COS	Reserved						
R-0							R/W-0	R/W-0	R-0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECPDIV15–0															
R/W-0															

R = Read; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-43. ECP Control Register (ECPCNTL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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 (offset 0x8C).**
		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.
22–16	Reserved		Reads return zero and writes have no effect.

Table 5-43. ECP Control Register (ECPCNTL) Field Descriptions (Continued)

Bit	Name	Value	Description
15–0	ECPDIV(15–0)	0–FFFF	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: $\mathrm{ECLK} = \frac{\mathrm{VCLK or OSCIN}}{(\mathrm{ECPDIV} + 1)}$ **Note: The ECLK pin does not support output frequencies greater than 80 MHz. See the device data sheet for other ECLK limitations.**

5.1.43 DEV Parity Control Register1 (DEVCR1)

This register is shown in Figure 5-44 and described in Table 5-44.

Figure 5-44. DEV Parity Control Register1 (DEVCR1) [offset = 0xDC]

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
Reserved												DEVPARSEL			
R-0												R/WP-1010			

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-44. DEV Parity Control Register1 (DEVCR1) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and writes have no effect.
3–0	DEVPARSEL		Device parity select bit key. Note: After an odd (**DEVPARSEL**=0101) or even (**DEVPARSEL**=1010) 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.
		1010	The device parity is odd.
		0101	The device parity is even.

5.1.44 System Exception Control Register (SYSECR)

The SYSECR register shown in Figure 5-45 and described in Table 5-45 is used to generate a software reset.

Figure 5-45. System Exception Control Register (SYSECR) [offset = 0xE0]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>RESET1</td><td>RESET0</td><td colspan="14">Reserved</td></tr>
<tr><td>R/WP-0</td><td>R/WP-1</td><td colspan="14">R-0</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-45. System Exception Control Register (SYSECR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–14	RESET[1–0]		Software reset bits. Setting RESET1 or clearing RESET0 causes a system software reset.
		01	No reset will occur.
		1x, x0	A global system reset will occur.
13–0	Reserved		Reads return zero and writes have no effect.

5.1.45 System Exception Status Register (SYSESR)

The SYSESR register shown in Figure 5-46 and described in Table 5-46, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. All bits in this register can be cleared by writing a ‘1’ to the bit.

Figure 5-46. System Exception Status Register (SYSESR) [offset = 0xE4]

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
PO RST	OSC RST	WD RST	Reserved							CPU RST	SW RST	EXT RST	Reserved		MPMOD E
R/WC-X	R/WC-X	R/WC-X	R-0							R/WC-X	R/WC-X	R/WC-X	R-0		R-0

R = Read in all modes; WC = Write clear in all modes; -X = Value unchanged after reset; -n = Value after reset

Table 5-46. System Exception Status Register (SYSESR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15	PORST		Power-up reset. This bit is set when VCCOR (VCC Out of Range) is detected.
		0	No reset has occurred due to the V_{CC} being out of range.
		1	A reset was caused by V_{CC} being out of range (nPORST was asserted).
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 (offset 0x70).**
		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 analog or digital watchdog.
		0	No reset has occurred because of the AWD or DWD.
		1	A reset was caused by the AWD or DWD.
12–6	Reserved		Reads return zero and writes have no effect.

Table 5-46. System Exception Status Register (SYSESR) Field Descriptions (Continued)

Bit	Name	Value	Description
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 MMUGCR register (offset 0xCC).**
		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 (offset 0xE0).**
		0	No software reset has occurred.
		1	A software reset occurred.
3	EXTRST		External reset flag. This bit is set when a reset is caused by the external reset pin nRST.
		0	The external reset pin has not asserted a reset.
		1	A reset has been caused by the external reset pin.
2–1	Reserved		Reads return zero and writes have no effect.
0	MPMODE		This indicates the current memory protection unit (MPU) mode.
		0	MPU is disabled.
		1	MPU is enabled.

5.1.46 Global Status Register (GLBSTAT)

The GLBSTAT register shown in Figure 5-47 and described in Table 5-47, 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 (offset 70h).

Figure 5-47. Global Status Register (GLBSTAT) [offset = 0xEC]

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
Reserved						FBSLIP	RFSLIP	Reserved							OSC FAIL
R-x						RC-0	RC-0								R/C-0

R = Read; C = Clear;

.

Table 5-47. Global Status Register (GLBSTAT) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9	FBSLIP		PLL over cycle slip detection.
		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. **NOTE:** During power on, the FBSLIP and/or the RFSLIP bits may be inadvertently set along with the CLK source valid bit for the PLL (0xFFFFFF54). The signal to the SYS slip reset and the error signalling module is not affected. **Workaround:** 1. Write PLLCTL1[30:29] (2 bit key) to binary 10 (disable slip output). This clears the PLL clock source valid signal. 2. Restore PLLCTL1[30:29] to binary 01 (enable slip output). 3. Clear the slip bits in GLBSTAT register (0xFFFFFFEC).

Table 5-47. Global Status Register (GLBSTAT) Field Descriptions (Continued)

Bit	Name	Value	Description
8	RFSLIP		PLL under cycle slip detection.
		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. **NOTE:** During power on, the FBSLIP and/or the RFSLIP bits may be inadvertently set along with the CLK source valid bit for the PLL (0xFFFFFF54). The signal to the SYS slip reset and the error signalling module is not affected. **Workaround:** 1. Write PLLCTL1[30:29] (2 bit key) to binary ‘10’ (disable slip output). This clears the PLL clock source valid signal. 2. Restore PLLCTL1[30:29] to binary ‘01’ (enable slip output). 3. Clear the slip bits in GLBSTAT register (0xFFFFFFEC).
7–1	Reserved		Reads return zero and writes have no effect.
0	OSCFAIL		Oscillator fail flag bit.
		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.

5.1.47 Device Identification Register (DEV)

The DEV is a read-only register. It contains device-specific information that is hard-coded during device manufacture. This register is shown in Figure 5-48 and described in Table 5-48.

Figure 5-48. Device Identification Register (DEV) [offset = 0xF0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CP15	ID(														TECH
R-K	R-K														R-K

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TECH			I/O	PPAR	Program parity		RECC	Version					Platform ID		
R-K			R-K	R-K	R-K		R-K	R-K					R-K		

R = Read only; -*K* = Constant value

.

Table 5-48. Device Identification Register (DEV) Field Descriptions

Bit	Name	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	ID	0–0x3FFF	Device ID. The device ID is unique by device configuration and is defined in the device data sheet.
16–13	TECH		These bits define the process technology by which the device was manufactured.
		0000	Device manufactured in the C05 process technology.
		0001	Device manufactured in the F05 process technology.
		0010	Device manufactured in the C035 process technology.
		0011	Device manufactured in the F035 process technology.
12	I/O		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.

Table 5-48. Device Identification Register (DEV) Field Descriptions (Continued)

Bit	Name	Value	Description
11	PPAR		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	Program parity		These bits indicate which parity is present for the program memory.
		00	No memory protection is present.
		01	The program memory (Flash) has single bit parity.
		10	The program memory (Flash) has ECC.
		11	This combination is reserved.
8	RECC		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	0–0x1F	Version. These bits provide the revision of the device.
2–0	Platform ID	101	The device is part of the TMS570Px family. The TMS570Px ID is always 101b

5.1.48 Software Interrupt Vector Register (SSIVEC)

The SSIVEC register shown in Figure 5-49 and described in Table 5-49, contains information about software interrupts.

Figure 5-49. Software Interrupt Vector Register (SSIVEC) [offset = 0xF4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">SSIDATA</td><td colspan="8">SSIVECT</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R-0</td></tr>
</table>

R = Read only; -*n* = Value after reset

.

Table 5-49. Software Interrupt Vector Register (SSIVEC) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	SSIDATA[7–0]	0–0xFF	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[7–0]		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.**
		0x00	No software interrupt is pending.
		0x01	A software interrupt has been generated by writing the correct key value to The SSIR1 register.
		0x02	A software interrupt has been generated by writing the correct key value to the SSIR2 register.
		0x03	A software interrupt has been generated by writing the correct key value to the SSIR3 register.
		0x04	A software interrupt has been generated by writing the correct key value to the SSIR4 register.
		0x05–0xFF	Reserved

5.1.49 System Software Interrupt Flag Register (SSIF)

The SSIF register shown in Figure 5-50 and described in Table 5-50, contains software interrupt flag status information.

Figure 5-50. System Software Interrupt Flag Register (SSIF) [offset = 0xF8]

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
Reserved												SSI_ FLAG4	SSI_ FLAG3	SSI_ FLAG2	SSI_ FLAG1
R-0												R/C-0	R/C-0	R/C-0	R/C-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

.

Table 5-50. System Software Interrupt Flag Register (SSIF) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return zero and 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.

5.2 Secondary System Control Registers (SYS2)

This section describes the secondary frame of system registers. The start address of the second system module frame is 0xFFFF E100 and the end address is 0xFFFF E1FF. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Figure 5-51. Secondary Frame System Control Registers Summary

<table>
<tr><th>Offset
Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x00
PLLCTL3</td><td colspan="9">Reserved</td><td>OSC_
DIV</td><td colspan="6">Reserved</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="4">PLL_MUL (3:0)</td><td colspan="5">Reserved</td><td colspan="3">PLL_DIV (2:0)</td></tr>
<tr><td rowspan="2">0x04
Reserved</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td rowspan="2">0x08
STCCLKDIV</td><td colspan="5">Reserved</td><td colspan="3">CLKDIV</td><td colspan="8">Reserved</td></tr>
<tr><td colspan="16">Reserved</td></tr>
</table>

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

5.2.1 PLL Control Register 3 (PLLCTL3)

The PLLCTL3 register is shown in Figure 5-52 and described in Table 5-51. This register controls the settings of PLL2 (Clock Source 6 - FPLL).

Figure 5-52. PLL Control Register 3 (PLLCTL3) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									OSC_ DIV	Reserved					
R-W-000000000									R-WP-0	R-W-000000					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PLL_MUL (3:0)				Reserved					PLL_DIV (2:0)		
R-W-0000				R-WP-0011				R-W-00000					R-WP-111		

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset;

.

Table 5-51. PLL Control Register 3 (PLLCTL3) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads are zero and writes have no effect
22	OSC_DIV		Configures the OSC clock division factor (NR) $f_{INTCLK} = f_{OSCin}$ / NR NR = OSC_DIV + 1
		0	F_{INTCLK} / 1
		1	F_{INTCLK} / 2
21–12	Reserved		Reads are zero and writes have no effect
11–8	PLL_MUL[3-0]		Configures PLL reference clock multiplication factor (NF) between 1 and 15 $f_{OutputCLK} = f_{INTCLK} * NF$ NF = PLL_MUL[3-0] + 1
		0000	$f_{OutputCLK} = 1 * f_{INTCLK}$
		0001	$f_{OutputCLK} = 2 * f_{INTCLK}$
		0010	$f_{OutputCLK} = 2 * f_{INTCLK}$
		0010	$f_{OutputCLK} = 3 * f_{INTCLK}$
		...	
		1101	$f_{OutputCLK} = 15 * f_{INTCLK}$
			$f_{OutputCLK} = 1 * f_{INTCLK}$ (special case)
		1111	**NOTE: (PLL_MUL+1) sets the multiply factor for the PLL except in the special case of PLL_MUL = 1111b which is logically equivalent to the setting of 0000b.**

Table 5-51. PLL Control Register 3 (PLLCTL3) Field Descriptions (Continued)

Bit	Name	Value	Description
7–3	Reserved		Reads are zero and writes have no effect
2–0	PLL_DIV[2-0]		Configures PLL division factor (R) $f_{PLLCLK} = f_{OutputCLK} / R$ R = PLL_DIV[2-0] + 1
		000	$f_{PLLCLK} = f_{OutputCLK} / 1$
		001	$f_{PLLCLK} = f_{OutputCLK} / 2$
		010	$f_{PLLCLK} = f_{OutputCLK} / 3$
		...	
		110	$f_{PLLCLK} = f_{OutputCLK} / 7$
		111	$f_{PLLCLK} = f_{OutputCLK} / 8$

5.2.2 *CPU Logic BIST Clock Divider (STCCLKDIV)*

The STCCLKDIV register is shown in Figure 5-53 and described in Table 5-52. This register is used to set the internal Cortex R4 CPU Logic Built In Self-Test (LBIST) divider. This divider is used to reduce the LBIST clock speed without the need to change the main device clock HCLK/PLL speed.

> **Note: Maximum Cortex R4 CPU Self-Test (LBIST) Speed.**
> The maximum speed that LBIST can be executed is slower than the device's maximum rated frequency. The maximum speed can be found in the device data sheet.

Figure 5-53. CPU Logic BIST Clock Prescaler (STCCLKDIV) [offset = 0x08]

31–27	26–24	23–16
	CLKDIV	Reserved
R-0	RWP-000	R-0

15–0
Reserved
R-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset; D = device specific

.

Table 5-52. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved	0	
26-24	CLKDIV		Clock divider/prescaler for CPU clock during CPU Self-Test - LBIST
		000	STC CLK = HCLK / 1
		001	STC CLK = HCLK / 2
		001	STC CLK = HCLK / 3
		...	
		110	STC CLK = HCLK / 7
		111	STC CLK = HCLK / 8
23-0	Reserved	0	

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 0xFFFF E000 and the end address is 0xFFFF E0FF. Figure 5-54 summarizes the registers in the PCR, which are used to configure protection to the peripherals in PCS and PS regions. The following sections provide detailed information about these registers. Additional information about the PCR control registers can be found in the Exceptions section of the Architecture Overview chapter.

Figure 5-54. PCR Register Summary

Offset Address Register	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0x00 PMPROTSET0 Page 157	PCS 31 PROT SET	PCS 30 PROT SET	PCS 29 PROT SET	PCS 28 PROT SET	PCS 27 PROT SET	PCS 26 PROT SET	PCS 25 PROT SET	PCS 24 PROT SET	PCS 23 PROT SET	PCS 22 PROT SET	PCS 21 PROT SET	PCS 20 PROT SET	PCS 19 PROT SET	PCS 18 PROT SET	PCS 17 PROT SET	PCS 16 PROT SET
	PCS 15 PROT SET	PCS 14 PROT SET	PCS 13 PROT SET	PCS 12 PROT SET	PCS 11 PROT SET	PCS 10 PROT SET	PCS 9 PROT SET	PCS 8 PROT SET	PCS 7 PROT SET	PCS 6 PROT SET	PCS 5 PROT SET	PCS 4 PROT SET	PCS 3 PROT SET	PCS 2 PROT SET	PCS 1 PROT SET	PCS 0 PROT SET
0x04 PMPROTSET1 Page 158	PCS 63 PROT SET	PCS 62 PROT SET	PCS 61 PROT SET	PCS 60 PROT SET	PCS 59 PROT SET	PCS 58 PROT SET	PCS 57 PROT SET	PCS 56 PROT SET	PCS 55 PROT SET	PCS 54 PROT SET	PCS 53 PROT SET	PCS 52 PROT SET	PCS 51 PROT SET	PCS 50 PROT SET	PCS 49 PROT SET	PCS 48 PROT SET
	PCS 47 PROT SET	PCS 46 PROT SET	PCS 45 PROT SET	PCS 44 PROT SET	PCS 43 PROT SET	PCS 42 PROT SET	PCS 41 PROT SET	PCS 40 PROT SET	PCS 39 PROT SET	PCS 38 PROT SET	PCS 37 PROT SET	PCS 36 PROT SET	PCS 35 PROT SET	PCS 34 PROT SET	PCS 33 PROT SET	PCS 32 PROT SET
0x08 Reserved	Reserved															
	Reserved															
0x0C Reserved	Reserved															
	Reserved															
0x10 PMPROTCLR0 Page 159	PCS 31 PROT CLR	PCS 30 PROT CLR	PCS 29 PROT CLR	PCS 28 PROT CLR	PCS 27 PROT CLR	PCS 26 PROT CLR	PCS 25 PROT CLR	PCS 24 PROT CLR	PCS 23 PROT CLR	PCS 22 PROT CLR	PCS 21 PROT CLR	PCS 20 PROT CLR	PCS 19 PROT CLR	PCS 18 PROT CLR	PCS 17 PROT CLR	PCS 16 PROT CLR
	PCS 15 PROT CLR	PCS 14 PROT CLR	PCS 13 PROT CLR	PCS 12 PROT CLR	PCS 11 PROT CLR	PCS 10 PROT CLR	PCS 9 PROT CLR	PCS 8 PROT CLR	PCS 7 PROT CLR	PCS 6 PROT CLR	PCS 5 PROT CLR	PCS 4 PROT CLR	PCS 3 PROT CLR	PCS 2 PROT CLR	PCS 1 PROT CLR	PCS 0 PROT CLR
0x14 PMPROTCLR1 Page 160	PCS 63 PROT CLR	PCS 62 PROT CLR	PCS 61 PROT CLR	PCS 60 PROT CLR	PCS 59 PROT CLR	PCS 58 PROT CLR	PCS 57 PROT CLR	PCS 56 PROT CLR	PCS 55 PROT CLR	PCS 54 PROT CLR	PCS 53 PROT CLR	PCS 52 PROT CLR	PCS 51 PROT CLR	PCS 50 PROT CLR	PCS 49 PROT CLR	PCS 48 PROT CLR
	PCS47 PROT CLR	PCS46 PROT CLR	PCS45 PROT CLR	PCS44 PROT CLR	PCS43 PROT CLR	PCS42 PROT CLR	PCS41 PROT CLR	PCS40 PROT CLR	PCS39 PROT CLR	PCS38 PROT CLR	PCS37 PROT CLR	PCS36 PROT CLR	PCS35 PROT CLR	PCS34 PROT CLR	PCS33 PROT CLR	PCS32 PROT CLR

Figure 5-54. PCR Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18 Reserved	Reserved															
	Reserved															
0x1C Reserved	Reserved															
	Reserved															
0x20 PPROTSET0 Page 161	PS7 QUAD 3 PROT SET	PS7 QUAD 2 PROT SET	PS7 QUAD 1 PROT SET	PS7 QUAD 0 PROT SET	PS6 QUAD 3 PROT SET	PS6 QUAD 2 PROT SET	PS6 QUAD 1 PROT SET	PS6 QUAD 0 PROT SET	PS5 QUAD 3 PROT SET	PS5 QUAD 2 PROT SET	PS5 QUAD 1 PROT SET	PS5 QUAD 0 PROT SET	PS4 QUAD 3 PROT SET	PS4 QUAD 2 PROT SET	PS4 QUAD 1 PROT SET	PS4 QUAD 0 PROT SET
	PS3 QUAD 3 PROT SET	PS3 QUAD 2 PROT SET	PS3 QUAD 1 PROT SET	PS3 QUAD 0 PROT SET	PS2 QUAD 3 PROT SET	PS2 QUAD 2 PROT SET	PS2 QUAD 1 PROT SET	PS2 QUAD 0 PROT SET	PS1 QUAD 3 PROT SET	PS1 QUAD 2 PROT SET	PS1 QUAD 1 PROT SET	PS1 QUAD 0 PROT SET	PS0 QUAD 3 PROT SET	PS0 QUAD 2 PROT SET	PS0 QUAD 1 PROT SET	PS0 QUAD 0 PROT SET
0x24 PPROTSET1 Page 163	PS15 QUAD 3 PROT SET	PS15 QUAD 2 PROT SET	PS15 QUAD 1 PROT SET	PS15 QUAD 0 PROT SET	PS14 QUAD 3 PROT SET	PS14 QUAD 2 PROT SET	PS14 QUAD 1 PROT SET	PS14 QUAD 0 PROT SET	PS13 QUAD 3 PROT SET	PS13 QUAD 2 PROT SET	PS13 QUAD 1 PROT SET	PS13 QUAD 0 PROT SET	PS12 QUAD 3 PROT SET	PS12 QUAD 2 PROT SET	PS12 QUAD 1 PROT SET	PS12 QUAD 0 PROT SET
	PS11 QUAD 3 PROT SET	PS11 QUAD 2 PROT SET	PS11 QUAD 1 PROT SET	PS11 QUAD 0 PROT SET	PS10 QUAD 3 PROT SET	PS10 QUAD 2 PROT SET	PS10 QUAD 1 PROT SET	PS10 QUAD 0 PROT SET	PS9 QUAD 3 PROT SET	PS9 QUAD 2 PROT SET	PS9 QUAD 1 PROT SET	PS9 QUAD 0 PROT SET	PS8 QUAD 3 PROT SET	PS8 QUAD 2 PROT SET	PS8 QUAD 1 PROT SET	PS8 QUAD 0 PROT SET
0x28 PPROTSET2 Page 164	PS23 QUAD 3 PROT SET	PS23 QUAD 2 PROT SET	PS23 QUAD 1 PROT SET	PS23 QUAD 0 PROT SET	PS22 QUAD 3 PROT SET	PS22 QUAD 2 PROT SET	PS22 QUAD 1 PROT SET	PS22 QUAD 0 PROT SET	PS21 QUAD 3 PROT SET	PS21 QUAD 2 PROT SET	PS21 QUAD 1 PROT SET	PS21 QUAD 0 PROT SET	PS20 QUAD 3 PROT SET	PS20 QUAD 2 PROT SET	PS20 QUAD 1 PROT SET	PS20 QUAD 0 PROT SET
	PS19 QUAD 3 PROT SET	PS19 QUAD 2 PROT SET	PS19 QUAD 1 PROT SET	PS19 QUAD 0 PROT SET	PS18 QUAD 3 PROT SET	PS18 QUAD 2 PROT SET	PS18 QUAD 1 PROT SET	PS18 QUAD 0 PROT SET	PS17 QUAD 3 PROT SET	PS17 QUAD 2 PROT SET	PS17 QUAD 1 PROT SET	PS17 QUAD 0 PROT SET	PS16 QUAD 3 PROT SET	PS16 QUAD 2 PROT SET	PS16 QUAD 1 PROT SET	PS16 QUAD 0 PROT SET
0x2C PPROTSET3 Page 165	PS31 QUAD 3 PROT SET	PS31 QUAD 2 PROT SET	PS31 QUAD 1 PROT SET	PS31 QUAD 0 PROT SET	PS30 QUAD 3 PROT SET	PS30 QUAD 2 PROT SET	PS30 QUAD 1 PROT SET	PS30 QUAD 0 PROT SET	PS29 QUAD 3 PROT SET	PS29 QUAD 2 PROT SET	PS29 QUAD 1 PROT SET	PS29 QUAD 0 PROT SET	PS28 QUAD 3 PROT SET	PS28 QUAD 2 PROT SET	PS28 QUAD 1 PROT SET	PS28 QUAD 0 PROT SET
	PS27 QUAD 3 PROT SET	PS27 QUAD 2 PROT SET	PS27 QUAD 1 PROT SET	PS27 QUAD 0 PROT SET	PS26 QUAD 3 PROT SET	PS26 QUAD 2 PROT SET	PS26 QUAD 1 PROT SET	PS26 QUAD 0 PROT SET	PS25 QUAD 3 PROT SET	PS25 QUAD 2 PROT SET	PS25 QUAD 1 PROT SET	PS25 QUAD 0 PROT SET	PS24 QUAD 3 PROT SET	PS24 QUAD 2 PROT SET	PS24 QUAD 1 PROT SET	PS24 QUAD 0 PROT SET

Figure 5-54. PCR Register Summary (Continued)

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
0x30 Reserved	Reserved															
	Reserved															
0x34 Reserved	Reserved															
	Reserved															
0x38 Reserved	Reserved															
	Reserved															
0x3C Reserved	Reserved															
	Reserved															
0x40 PPROTCLR0 Page 166	PS7 QUAD 3 PROT CLR	PS7 QUAD 2 PROT CLR	PS7 QUAD 1 PROT CLR	PS7 QUAD 0 PROT CLR	PS6 QUAD 3 PROT CLR	PS6 QUAD 2 PROT CLR	PS6 QUAD 1 PROT CLR	PS6 QUAD 0 PROT CLR	PS5 QUAD 3 PROT CLR	PS5 QUAD 2 PROT CLR	PS5 QUAD 1 PROT CLR	PS5 QUAD 0 PROT CLR	PS4 QUAD 3 PROT CLR	PS4 QUAD 2 PROT CLR	PS4 QUAD 1 PROT CLR	PS4 QUAD 0 PROT CLR
	PS3 QUAD 3 PROT CLR	PS3 QUAD 2 PROT CLR	PS3 QUAD 1 PROT CLR	PS3 QUAD 0 PROT CLR	PS2 QUAD 3 PROT CLR	PS2 QUAD 2 PROT CLR	PS2 QUAD 1 PROT CLR	PS2 QUAD 0 PROT CLR	PS1 QUAD 3 PROT CLR	PS1 QUAD 2 PROT CLR	PS1 QUAD 1 PROT CLR	PS1 QUAD 0 PROT CLR	PS0 QUAD 3 PROT CLR	PS0 QUAD 2 PROT CLR	PS0 QUAD 1 PROT CLR	PS0 QUAD 0 PROT CLR
0x44 PPROTCLR1 Page 167	PS15 QUAD 3 PROT CLR	PS15 QUAD 2 PROT CLR	PS15 QUAD 1 PROT CLR	PS15 QUAD 0 PROT CLR	PS14 QUAD 3 PROT CLR	PS14 QUAD 2 PROT CLR	PS14 QUAD 1 PROT CLR	PS14 QUAD 0 PROT CLR	PS13 QUAD 3 PROT CLR	PS13 QUAD 2 PROT CLR	PS13 QUAD 1 PROT CLR	PS13 QUAD 0 PROT CLR	PS12 QUAD 3 PROT CLR	PS12 QUAD 2 PROT CLR	PS12 QUAD 1 PROT CLR	PS12 QUAD 0 PROT CLR
	PS11 QUAD 3 PROT CLR	PS11 QUAD 2 PROT CLR	PS11 QUAD 1 PROT CLR	PS11 QUAD 0 PROT CLR	PS10 QUAD 3 PROT CLR	PS10 QUAD 2 PROT CLR	PS10 QUAD 1 PROT CLR	PS10 QUAD 0 PROT CLR	PS9 QUAD 3 PROT CLR	PS9 QUAD 2 PROT CLR	PS9 QUAD 1 PROT CLR	PS9 QUAD 0 PROT CLR	PS8 QUAD 3 PROT CLR	PS8 QUAD 2 PROT CLR	PS8 QUAD 1 PROT CLR	PS8 QUAD 0 PROT CLR
0x48 PPROTCLR2 Page 168	PS23 QUAD 3 PROT CLR	PS23 QUAD 2 PROT CLR	PS23 QUAD 1 PROT CLR	PS23 QUAD 0 PROT CLR	PS22 QUAD 3 PROT CLR	PS22 QUAD 2 PROT CLR	PS22 QUAD 1 PROT CLR	PS22 QUAD 0 PROT CLR	PS21 QUAD 3 PROT CLR	PS21 QUAD 2 PROT CLR	PS21 QUAD 1 PROT CLR	PS21 QUAD 0 PROT CLR	PS20 QUAD 3 PROT CLR	PS20 QUAD 2 PROT CLR	PS20 QUAD 1 PROT CLR	PS20 QUAD 0 PROT CLR
	PS19 QUAD 3 PROT CLR	PS19 QUAD 2 PROT CLR	PS19 QUAD 1 PROT CLR	PS19 QUAD 0 PROT CLR	PS18 QUAD 3 PROT CLR	PS18 QUAD 2 PROT CLR	PS18 QUAD 1 PROT CLR	PS18 QUAD 0 PROT CLR	PS17 QUAD 3 PROT CLR	PS17 QUAD 2 PROT CLR	PS17 QUAD 1 PROT CLR	PS17 QUAD 0 PROT CLR	PS16 QUAD 3 PROT CLR	PS16 QUAD 2 PROT CLR	PS16 QUAD 1 PROT CLR	PS16 QUAD 0 PROT CLR

Figure 5-54. PCR Register Summary (Continued)

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
0x4C PPROTCLR3 Page 169	PS31 QUAD 3 PROT CLR	PS31 QUAD 2 PROT CLR	PS31 QUAD 1 PROT CLR	PS31 QUAD 0 PROT CLR	PS30 QUAD 3 PROT CLR	PS30 QUAD 2 PROT CLR	PS30 QUAD 1 PROT CLR	PS30 QUAD 0 PROT CLR	PS29 QUAD 3 PROT CLR	PS29 QUAD 2 PROT CLR	PS29 QUAD 1 PROT CLR	PS29 QUAD 0 PROT CLR	PS28 QUAD 3 PROT CLR	PS28 QUAD 2 PROT CLR	PS28 QUAD 1 PROT CLR	PS28 QUAD 0 PROT CLR
	PS27 QUAD 3 PROT CLR	PS27 QUAD 2 PROT CLR	PS27 QUAD 1 PROT CLR	PS27 QUAD 0 PROT CLR	PS26 QUAD 3 PROT CLR	PS26 QUAD 2 PROT CLR	PS26 QUAD 1 PROT CLR	PS26 QUAD 0 PROT CLR	PS25 QUAD 3 PROT CLR	PS25 QUAD 2 PROT CLR	PS25 QUAD 1 PROT CLR	PS25 QUAD 0 PROT CLR	PS24 QUAD 3 PROT CLR	PS24 QUAD 2 PROT CLR	PS24 QUAD 1 PROT CLR	PS24 QUAD 0 PROT CLR
0x50 Reserved	Reserved															
	Reserved															
0x54 Reserved	Reserved															
	Reserved															
0x58 Reserved	Reserved															
	Reserved															
0x5C Reserved	Reserved															
	Reserved															
0x60 PCSPWRDWNSET0 Page 170	PCS31 PWRDWN SET	PCS30 PWRDWN SET	PCS29 PWRDWN SET	PCS28 PWRDWN SET	PCS27 PWRDWN SET	PCS26 PWRDWN SET	PCS25 PWRDWN SET	PCS24 PWRDWN SET	PCS23 PWRDWN SET	PCS22 PWRDWN SET	PCS21 PWRDWN SET	PCS20 PWRDWN SET	PCS19 PWRDWN SET	PCS18 PWRDWN SET	PCS17 PWRDWN SET	PCS16 PWRDWN SET
	PCS15 PWRDWN SET	PCS14 PWRDWN SET	PCS13 PWRDWN SET	PCS12 PWRDWN SET	PCS11 PWRDWN SET	PCS10 PWRDWN SET	PCS9 PWRDWN SET	PCS8 PWRDWN SET	PCS7 PWRDWN SET	PCS6 PWRDWN SET	PCS5 PWRDWN SET	PCS4 PWRDWN SET	PCS3 PWRDWN SET	PCS2 PWRDWN SET	PCS1 PWRDWN SET	PCS0 PWRDWN SET
0x64 PCSPWRDWNSET1 Page 171	PCS63 PWRDWN SET	PCS62 PWRDWN SET	PCS61 PWRDWN SET	PCS60 PWRDWN SET	PCS59 PWRDWN SET	PCS58 PWRDWN SET	PCS57 PWRDWN SET	PCS56 PWRDWN SET	PCS55 PWRDWN SET	PCS54 PWRDWN SET	PCS53 PWRDWN SET	PCS52 PWRDWN SET	PCS51 PWRDWN SET	PCS50 PWRDWN SET	PCS49 PWRDWN SET	PCS48 PWRDWN SET
	PCS47 PWRDWN SET	PCS46 PWRDWN SET	PCS45 PWRDWN SET	PCS44 PWRDWN SET	PCS43 PWRDWN SET	PCS42 PWRDWN SET	PCS41 PWRDWN SET	PCS40 PWRDWN SET	PCS39 PWRDWN SET	PCS38 PWRDWN SET	PCS37 PWRDWN SET	PCS36 PWRDWN SET	PCS35 PWRDWN SET	PCS34 PWRDWN SET	PCS33 PWRDWN SET	PCS32 PWRDWN SET

Figure 5-54. PCR Register Summary (Continued)

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
0x68 Reserved	Reserved															
	Reserved															
0x6C Reserved	Reserved															
	Reserved															
0x70 PCSPWRDWN CLR0 Page 172	PCS31 PWRD WN CLR	PCS30 PWRD WN CLR	PCS29 PWRD WN CLR	PCS28 PWRD WN CLR	PCS27 PWRD WN CLR	PCS26 PWRD WN CLR	PCS25 PWRD WN CLR	PCS24 PWRD WN CLR	PCS23 PWRD WN CLR	PCS22 PWRD WN CLR	PCS21 PWRD WN CLR	PCS20 PWRD WN CLR	PCS19 PWRD WN CLR	PCS18 PWRD WN CLR	PCS17 PWRD WN CLR	PCS16 PWRD WN CLR
	PCS15 PWRD WN CLR	PCS14 PWRD WN CLR	PCS13 PWRD WN CLR	PCS12 PWRD WN CLR	PCS11 PWRD WN CLR	PCS10 PWRD WN CLR	PCS9 PWRD WN CLR	PCS8 PWRD WN CLR	PCS7 PWRD WN CLR	PCS6 PWRD WN CLR	PCS5 PWRD WN CLR	PCS4 PWRD WN CLR	PCS3 PWRD WN CLR	PCS2 PWRD WN CLR	PCS1 PWRD WN CLR	PCS0 PWRD WN CLR
0x74 PCSPWRDWN CLR1 Page 173	PCS63 PWRD WN CLR	PCS62 PWRD WN CLR	PCS61 PWRD WN CLR	PCS60 PWRD WN CLR	PCS59 PWRD WN CLR	PCS58 PWRD WN CLR	PCS57 PWRD WN CLR	PCS56 PWRD WN CLR	PCS55 PWRD WN CLR	PCS54 PWRD WN CLR	PCS53 PWRD WN CLR	PCS52 PWRD WN CLR	PCS51 PWRD WN CLR	PCS50 PWRD WN CLR	PCS49 PWRD WN CLR	PCS48 PWRD WN CLR
	PCS47 PWRD WN CLR	PCS46 PWRD WN CLR	PCS45 PWRD WN CLR	PCS44 PWRD WN CLR	PCS43 PWRD WN CLR	PCS42 PWRD WN CLR	PCS41 PWRD WN CLR	PCS40 PWRD WN CLR	PCS39 PWRD WN CLR	PCS38 PWRD WN CLR	PCS37 PWRD WN CLR	PCS36 PWRD WN CLR	PCS35 PWRD WN CLR	PCS34 PWRD WN CLR	PCS33 PWRD WN CLR	PCS32 PWRD WN CLR
0x78 Reserved	Reserved															
	Reserved															
0x7C Reserved	Reserved															
	Reserved															
0x80 PSPWRDWN SET0 Page 174	PS7 QUAD 3 PWRD WN SET	PS7 QUAD 2 PWRD WN SET	PS7 QUAD 1 PWRD WN SET	PS7 QUAD 0 PWRD WN SET	PS6 QUAD 3 PWRD WN SET	PS6 QUAD 2 PWRD WN SET	PS6 QUAD 1 PWRD WN SET	PS6 QUAD 0 PWRD WN SET	PS5 QUAD 3 PWRD WN SET	PS5 QUAD 2 PWRD WN SET	PS5 QUAD 1 PWRD WN SET	PS5 QUAD 0 PWRD WN SET	PS4 QUAD 3 PWRD WN SET	PS4 QUAD 2 PWRD WN SET	PS4 QUAD 1 PWRD WN SET	PS4 QUAD 0 PWRD WN SET
	PS3 QUAD 3 PWRD WN SET	PS3 QUAD 2 PWRD WN SET	PS3 QUAD 1 PWRD WN SET	PS3 QUAD 0 PWRD WN SET	PS2 QUAD 3 PWRD WN SET	PS2 QUAD 2 PWRD WN SET	PS2 QUAD 1 PWRD WN SET	PS2 QUAD 0 PWRD WN SET	PS1 QUAD 3 PWRD WN SET	PS1 QUAD 2 PWRD WN SET	PS1 QUAD 1 PWRD WN SET	PS1 QUAD 0 PWRD WN SET	PS0 QUAD 3 PWRD WN SET	PS0 QUAD 2 PWRD WN SET	PS0 QUAD 1 PWRD WN SET	PS0 QUAD 0 PWRD WN SET

Figure 5-54. PCR Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x84 PSPWRDWN SET1 Page 175	PS15 QUAD 3 PWRD WN SET	PS15 QUAD 2 PWRD WN SET	PS15 QUAD 1 PWRD WN SET	PS15 QUAD 0 PWRD WN SET	PS14 QUAD 3 PWRD WN SET	PS14 QUAD 2 PWRD WN SET	PS14 QUAD 1 PWRD WN SET	PS14 QUAD 0 PWRD WN SET	PS13 QUAD 3 PWRD WN SET	PS13 QUAD 2 PWRD WN SET	PS13 QUAD 1 PWRD WN SET	PS13 QUAD 0 PWRD WN SET	PS12 QUAD 3 PWRD WN SET	PS12 QUAD 2 PWRD WN SET	PS12 QUAD 1 PWRD WN SET	PS12 QUAD 0 PWRD WN SET
	PS11 QUAD 3 PWRD WN SET	PS11 QUAD 2 PWRD WN SET	PS11 QUAD 1 PWRD WN SET	PS11 QUAD 0 PWRD WN SET	PS10 QUAD 3 PWRD WN SET	PS10 QUAD 2 PWRD WN SET	PS10 QUAD 1 PWRD WN SET	PS10 QUAD 0 PWRD WN SET	PS9 QUAD 3 PWRD WN SET	PS9 QUAD 2 PWRD WN SET	PS9 QUAD 1 PWRD WN SET	PS9 QUAD 0 PWRD WN SET	PS8 QUAD 3 PWRD WN SET	PS8 QUAD 2 PWRD WN SET	PS8 QUAD 1 PWRD WN SET	PS8 QUAD 0 PWRD WN SET
0x88 PSPWRDWN SET2 Page 175	PS23 QUAD 3 PWRD WN SET	PS23 QUAD 2 PWRD WN SET	PS23 QUAD 1 PWRD WN SET	PS23 QUAD 0 PWRD WN SET	PS22 QUAD 3 PWRD WN SET	PS22 QUAD 2 PWRD WN SET	PS22 QUAD 1 PWRD WN SET	PS22 QUAD 0 PWRD WN SET	PS21 QUAD 3 PWRD WN SET	PS21 QUAD 2 PWRD WN SET	PS21 QUAD 1 PWRD WN SET	PS21 QUAD 0 PWRD WN SET	PS20 QUAD 3 PWRD WN SET	PS20 QUAD 2 PWRD WN SET	PS20 QUAD 1 PWRD WN SET	PS20 QUAD 0 PWRD WN SET
	PS19 QUAD 3 PWRD WN SET	PS19 QUAD 2 PWRD WN SET	PS19 QUAD 1 PWRD WN SET	PS19 QUAD 0 PWRD WN SET	PS18 QUAD 3 PWRD WN SET	PS18 QUAD 2 PWRD WN SET	PS18 QUAD 1 PWRD WN SET	PS18 QUAD 0 PWRD WN SET	PS17 QUAD 3 PWRD WN SET	PS17 QUAD 2 PWRD WN SET	PS17 QUAD 1 PWRD WN SET	PS17 QUAD 0 PWRD WN SET	PS16 QUAD 3 PWRD WN SET	PS16 QUAD 2 PWRD WN SET	PS16 QUAD 1 PWRD WN SET	PS16 QUAD 0 PWRD WN SET
0x8C PSPWRDWN SET3 Page 177	PS31 QUAD 3 PWRD WN SET	PS31 QUAD 2 PWRD WN SET	PS31 QUAD 1 PWRD WN SET	PS31 QUAD 0 PWRD WN SET	PS30 QUAD 3 PWRD WN SET	PS30 QUAD 2 PWRD WN SET	PS30 QUAD 1 PWRD WN SET	PS30 QUAD 0 PWRD WN SET	PS29 QUAD 3 PWRD WN SET	PS29 QUAD 2 PWRD WN SET	PS29 QUAD 1 PWRD WN SET	PS29 QUAD 0 PWRD WN SET	PS28 QUAD 3 PWRD WN SET	PS28 QUAD 2 PWRD WN SET	PS28 QUAD 1 PWRD WN SET	PS28 QUAD 0 PWRD WN SET
	PS27 QUAD 3 PWRD WN SET	PS27 QUAD 2 PWRD WN SET	PS27 QUAD 1 PWRD WN SET	PS27 QUAD 0 PWRD WN SET	PS26 QUAD 3 PWRD WN SET	PS26 QUAD 2 PWRD WN SET	PS26 QUAD 1 PWRD WN SET	PS26 QUAD 0 PWRD WN SET	PS25 QUAD 3 PWRD WN SET	PS25 QUAD 2 PWRD WN SET	PS25 QUAD 1 PWRD WN SET	PS25 QUAD 0 PWRD WN SET	PS24 QUAD 3 PWRD WN SET	PS24 QUAD 2 PWRD WN SET	PS24 QUAD 1 PWRD WN SET	PS24 QUAD 0 PWRD WN SET
0x90 Reserved	Reserved															
	Reserved															
0x94 Reserved	Reserved															
	Reserved															
0x98 Reserved	Reserved															
	Reserved															

Figure 5-54. PCR Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x9C Reserved	Reserved															
	Reserved															
0xA0 PSPWRDWN CLR0 Page 178	PS7 QUAD 3 PWRDWN CLR	PS7 QUAD 2 PWRDWN CLR	PS7 QUAD 1 PWRDWN CLR	PS7 QUAD 0 PWRDWN CLR	PS6 QUAD 3 PWRDWN CLR	PS6 QUAD 2 PWRDWN CLR	PS6 QUAD 1 PWRDWN CLR	PS6 QUAD 0 PWRDWN CLR	PS5 QUAD 3 PWRDWN CLR	PS5 QUAD 2 PWRDWN CLR	PS5 QUAD 1 PWRDWN CLR	PS5 QUAD 0 PWRDWN CLR	PS4 QUAD 3 PWRDWN CLR	PS4 QUAD 2 PWRDWN CLR	PS4 QUAD 1 PWRDWN CLR	PS4 QUAD 0 PWRDWN CLR
	PS3 QUAD 3 PWRDWN CLR	PS3 QUAD 2 PWRDWN CLR	PS3 QUAD 1 PWRDWN CLR	PS3 QUAD 0 PWRDWN CLR	PS2 QUAD 3 PWRDWN CLR	PS2 QUAD 2 PWRDWN CLR	PS2 QUAD 1 PWRDWN CLR	PS2 QUAD 0 PWRDWN CLR	PS1 QUAD 3 PWRDWN CLR	PS1 QUAD 2 PWRDWN CLR	PS1 QUAD 1 PWRDWN CLR	PS1 QUAD 0 PWRDWN CLR	PS0 QUAD 3 PWRDWN CLR	PS0 QUAD 2 PWRDWN CLR	PS0 QUAD 1 PWRDWN CLR	PS0 QUAD 0 PWRDWN CLR
0xA4 PSPWRDWN CLR1 Page 179	PS15 QUAD 3 PWRDWN CLR	PS15 QUAD 2 PWRDWN CLR	PS15 QUAD 1 PWRDWN CLR	PS15 QUAD 0 PWRDWN CLR	PS14 QUAD 3 PWRDWN CLR	PS14 QUAD 2 PWRDWN CLR	PS14 QUAD 1 PWRDWN CLR	PS14 QUAD 0 PWRDWN CLR	PS13 QUAD 3 PWRDWN CLR	PS13 QUAD 2 PWRDWN CLR	PS13 QUAD 1 PWRDWN CLR	PS13 QUAD 0 PWRDWN CLR	PS12 QUAD 3 PWRDWN CLR	PS12 QUAD 2 PWRDWN CLR	PS12 QUAD 1 PWRDWN CLR	PS12 QUAD 0 PWRDWN CLR
	PS11 QUAD 3 PWRDWN CLR	PS11 QUAD 2 PWRDWN CLR	PS11 QUAD 1 PWRDWN CLR	PS11 QUAD 0 PWRDWN CLR	PS10 QUAD 3 PWRDWN CLR	PS10 QUAD 2 PWRDWN CLR	PS10 QUAD 1 PWRDWN CLR	PS10 QUAD 0 PWRDWN CLR	PS9 QUAD 3 PWRDWN CLR	PS9 QUAD 2 PWRDWN CLR	PS9 QUAD 1 PWRDWN CLR	PS9 QUAD 0 PWRDWN CLR	PS8 QUAD 3 PWRDWN CLR	PS8 QUAD 2 PWRDWN CLR	PS8 QUAD 1 PWRDWN CLR	PS8 QUAD 0 PWRDWN CLR
0xA8 PSPWRDWN CLR2 Page 180	PS23 QUAD 3 PWRDWN CLR	PS23 QUAD 2 PWRDWN CLR	PS23 QUAD 1 PWRDWN CLR	PS23 QUAD 0 PWRDWN CLR	PS22 QUAD 3 PWRDWN CLR	PS22 QUAD 2 PWRDWN CLR	PS22 QUAD 1 PWRDWN CLR	PS22 QUAD 0 PWRDWN CLR	PS21 QUAD 3 PWRDWN CLR	PS21 QUAD 2 PWRDWN CLR	PS21 QUAD 1 PWRDWN CLR	PS21 QUAD 0 PWRDWN CLR	PS20 QUAD 3 PWRDWN CLR	PS20 QUAD 2 PWRDWN CLR	PS20 QUAD 1 PWRDWN CLR	PS20 QUAD 0 PWRDWN CLR
	PS19 QUAD 3 PWRDWN CLR	PS19 QUAD 2 PWRDWN CLR	PS19 QUAD 1 PWRDWN CLR	PS19 QUAD 0 PWRDWN CLR	PS18 QUAD 3 PWRDWN CLR	PS18 QUAD 2 PWRDWN CLR	PS18 QUAD 1 PWRDWN CLR	PS18 QUAD 0 PWRDWN CLR	PS17 QUAD 3 PWRDWN CLR	PS17 QUAD 2 PWRDWN CLR	PS17 QUAD 1 PWRDWN CLR	PS17 QUAD 0 PWRDWN CLR	PS16 QUAD 3 PWRDWN CLR	PS16 QUAD 2 PWRDWN CLR	PS16 QUAD 1 PWRDWN CLR	PS16 QUAD 0 PWRDWN CLR
0xAC PSPWRDWN CLR3 Page 181	PS31 QUAD 3 PWRDWN CLR	PS31 QUAD 2 PWRDWN CLR	PS31 QUAD 1 PWRDWN CLR	PS31 QUAD 0 PWRDWN CLR	PS30 QUAD 3 PWRDWN CLR	PS30 QUAD 2 PWRDWN CLR	PS30 QUAD 1 PWRDWN CLR	PS30 QUAD 0 PWRDWN CLR	PS29 QUAD 3 PWRDWN CLR	PS29 QUAD 2 PWRDWN CLR	PS29 QUAD 1 PWRDWN CLR	PS29 QUAD 0 PWRDWN CLR	PS28 QUAD 3 PWRDWN CLR	PS28 QUAD 2 PWRDWN CLR	PS28 QUAD 1 PWRDWN CLR	PS28 QUAD 0 PWRDWN CLR
	PS27 QUAD 3 PWRDWN CLR	PS27 QUAD 2 PWRDWN CLR	PS27 QUAD 1 PWRDWN CLR	PS27 QUAD 0 PWRDWN CLR	PS26 QUAD 3 PWRDWN CLR	PS26 QUAD 2 PWRDWN CLR	PS26 QUAD 1 PWRDWN CLR	PS26 QUAD 0 PWRDWN CLR	PS25 QUAD 3 PWRDWN CLR	PS25 QUAD 2 PWRDWN CLR	PS25 QUAD 1 PWRDWN CLR	PS25 QUAD 0 PWRDWN CLR	PS24 QUAD 3 PWRDWN CLR	PS24 QUAD 2 PWRDWN CLR	PS24 QUAD 1 PWRDWN CLR	PS24 QUAD 0 PWRDWN CLR

5.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0)

This register is shown in Figure 5-55 and described in Table 5-53.

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

Figure 5-55. Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PROT SET	PCS30 PROT SET	PCS29 PROT SET	PCS28 PROT SET	PCS27 PROT SET	PCS26 PROT SET	PCS25 PROT SET	PCS24 PROT SET	PCS23 PROT SET	PCS22 PROT SET	PCS21 PROT SET	PCS20 PROT SET	PCS19 PROT SET	PCS18 PROT SET	PCS17 PROT SET	PCS16 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PROT SET	PCS14 PROT SET	PCS13 PROT SET	PCS12 PROT SET	PCS11 PROT SET	PCS10 PROT SET	PCS9 PROT SET	PCS8 PROT SET	PCS7 PROT SET	PCS6 PROT SET	PCS5 PROT SET	PCS4 PROT SET	PCS3 PROT SET	PCS2 PROT SET	PCS1 PROT SET	PCS0 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

Table 5-53. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions

Bit	Name	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.

5.3.2 *Peripheral Memory Protection Set Register 1 (PMPROTSET1)*

This register is shown in Figure 5-56 and described in Table 5-54.

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

Figure 5-56. Peripheral Memory Protection Set Register 1 (PMPROTSET1) [offset = 0x04]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PROT SET	PCS62 PROT SET	PCS61 PROT SET	PCS60 PROT SET	PCS59 PROT SET	PCS58 PROT SET	PCS57 PROT SET	PCS56 PROT SET	PCS55 PROT SET	PCS54 PROT SET	PCS53 PROT SET	PCS52 PROT SET	PCS51 PROT SET	PCS50 PROT SET	PCS49 PROT SET	PCS48 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PROT SET	PCS46 PROT SET	PCS45 PROT SET	PCS44 PROT SET	PCS43 PROT SET	PCS42 PROT SET	PCS41 PROT SET	PCS40 PROT SET	PCS39 PROT SET	PCS38 PROT SET	PCS37 PROT SET	PCS36 PROT SET	PCS35 PROT SET	PCS34 PROT SET	PCS33 PROT SET	PCS32 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 5-54. Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions

Bit	Name	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.

5.3.3 *Peripheral Memory Protection Clear Register 0 (PMPROTCLR0)*

This register is shown in Figure 5-57 and described in Table 5-55.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-57. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PROT CLR	PCS30 PROT CLR	PCS29 PROT CLR	PCS28 PROT CLR	PCS27 PROT CLR	PCS26 PROT CLR	PCS25 PROT CLR	PCS24 PROT CLR	PCS23 PROT CLR	PCS22 PROT CLR	PCS21 PROT CLR	PCS20 PROT CLR	PCS19 PROT CLR	PCS18 PROT CLR	PCS17 PROT CLR	PCS16 PROT CLR

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PROT CLR	PCS14 PROT CLR	PCS13 PROT CLR	PCS12 PROT CLR	PCS11 PROT CLR	PCS10 PROT CLR	PCS9 PROT CLR	PCS8 PROT CLR	PCS7 PROT CLR	PCS6 PROT CLR	PCS5 PROT CLR	PCS4 PROT CLR	PCS3 PROT CLR	PCS2 PROT CLR	PCS1 PROT CLR	PCS0 PROT CLR

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-55. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31–0]PROTCLR		Peripheral memory frame protection set.
		0	*Read*– The peripheral memory frame[31–0] can be written to and read from in both user and privileged modes. *Write*– The bit is unchanged.
		1	*Read*– The peripheral memory frame[31–0] 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.

5.3.4 Peripheral Memory Protection Clear Register 1 (PMPROTCLR1)

This register is shown in Figure 5-58 and described in Table 5-56.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-58. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) [offset = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PROT CLR	PCS62 PROT CLR	PCS61 PROT CLR	PCS60 PROT CLR	PCS59 PROT CLR	PCS58 PROT CLR	PCS57 PROT CLR	PCS56 PROT CLR	PCS55 PROT CLR	PCS54 PROT CLR	PCS53 PROT CLR	PCS52 PROT CLR	PCS51 PROT CLR	PCS50 PROT CLR	PCS49 PROT CLR	PCS48 PROT CLR
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PROT CLR	PCS46 PROT CLR	PCS45 PROT CLR	PCS44 PROT CLR	PCS43 PROT CLR	PCS42 PROT CLR	PCS41 PROT CLR	PCS40 PROT CLR	PCS39 PROT CLR	PCS38 PROT CLR	PCS37 PROT CLR	PCS36 PROT CLR	PCS35 PROT CLR	PCS34 PROT CLR	PCS33 PROT CLR	PCS32 PROT CLR
R/WP-0															

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-56. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[63–32]PROTCLR		Peripheral memory frame protection clear.
		0	*Read*– The peripheral memory frame[63–32] can be written to and read from in both user and privileged modes. *Write*– The bit is unchanged.
		1	*Read*– The peripheral memory frame[63–32] 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.

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 5-57 protects the slave.

The above arrangement is true for all the peripheral select (PS0 to PS31), presented in section 5.3.6—section 5.3.12. This register holds bits for PS0 to PS7 and is shown in Figure 5-59 and described in Table 5-57.

Note:
Writes to non implemented bits have no effect and reads are 0.

Figure 5-59. Peripheral Protection Set Register 0 (PPROTSET0) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PROT SET	PS7 QUAD2 PROT SET	PS7 QUAD1 PROT SET	PS7 QUAD0 PROT SET	PS6 QUAD3 PROT SET	PS6 QUAD2 PROT SET	PS6 QUAD1 PROT SET	PS6 QUAD0 PROT SET	PS5 QUAD3 PROT SET	PS5 QUAD2 PROT SET	PS5 QUAD1 PROT SET	PS5 QUAD0 PROT SET	PS4 QUAD3 PROT SET	PS4 QUAD2 PROT SET	PS4 QUAD1 PROT SET	PS4 QUAD0 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PROT SET	PS3 QUAD2 PROT SET	PS3 QUAD1 PROT SET	PS3 QUAD0 PROT SET	PS2 QUAD3 PROT SET	PS2 QUAD2 PROT SET	PS2 QUAD1 PROT SET	PS2 QUAD0 PROT SET	PS1 QUAD3 PROT SET	PS1 QUAD2 PROT SET	PS1 QUAD1 PROT SET	PS1 QUAD0 PROT SET	PS0 QUAD3 PROT SET	PS0 QUAD2 PROT SET	PS0 QUAD1 PROT SET	PS0 QUAD0 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-57. Peripheral Protection Set Register 0 (PPROTSET0)) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[7–0]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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET0 and PMPROTCLR0 registers is set to 1.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-60 and described in Table 5-58.

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

Figure 5-60. Peripheral Protection Set Register 1 (PPROTSET1) [offset = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PROT SET	PS15 QUAD2 PROT SET	PS15 QUAD1 PROT SET	PS15 QUAD0 PROT SET	PS14 QUAD3 PROT SET	PS14 QUAD2 PROT SET	PS14 QUAD1 PROT SET	PS14 QUAD0 PROT SET	PS13 QUAD3 PROT SET	PS13 QUAD2 PROT SET	PS13 QUAD1 PROT SET	PS13 QUAD0 PROT SET	PS12 QUAD3 PROT SET	PS12 QUAD2 PROT SET	PS12 QUAD1 PROT SET	PS12 QUAD0 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PROT SET	PS11 QUAD2 PROT SET	PS11 QUAD1 PROT SET	PS11 QUAD0 PROT SET	PS10 QUAD3 PROT SET	PS10 QUAD2 PROT SET	PS10 QUAD1 PROT SET	PS10 QUAD0 PROT SET	PS9 QUAD3 PROT SET	PS9 QUAD2 PROT SET	PS9 QUAD1 PROT SET	PS9 QUAD0 PROT SET	PS8 QUAD3 PROT SET	PS8 QUAD2 PROT SET	PS8 QUAD1 PROT SET	PS8 QUAD0 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-58. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET1 and PMPROTCLR1 registers is set to 1.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-61 and described in Table 5-59.

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

Figure 5-61. Peripheral Protection Set Register 2 (PPROTSET2) [offset = 0x28]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PROT SET	PS23 QUAD2 PROT SET	PS23 QUAD1 PROT SET	PS23 QUAD0 PROT SET	PS22 QUAD3 PROT SET	PS22 QUAD2 PROT SET	PS22 QUAD1 PROT SET	PS22 QUAD0 PROT SET	PS21 QUAD3 PROT SET	PS21 QUAD2 PROT SET	PS21 QUAD1 PROT SET	PS21 QUAD0 PROT SET	PS20 QUAD3 PROT SET	PS20 QUAD2 PROT SET	PS20 QUAD1 PROT SET	PS20 QUAD0 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PROT SET	PS19 QUAD2 PROT SET	PS19 QUAD1 PROT SET	PS19 QUAD0 PROT SET	PS18 QUAD3 PROT SET	PS18 QUAD2 PROT SET	PS18 QUAD1 PROT SET	PS18 QUAD0 PROT SET	PS17 QUAD3 PROT SET	PS17 QUAD2 PROT SET	PS17 QUAD1 PROT SET	PS17 QUAD0 PROT SET	PS16 QUAD3 PROT SET	PS16 QUAD2 PROT SET	PS16 QUAD1 PROT SET	PS16 QUAD0 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-59. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET2 and PMPROTCLR2 registers is set to 1.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-62 and described in Table 5-60.

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

Figure 5-62. Peripheral Protection Set Register 3 (PPROTSET3) [offset = 0x2C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PROT SET	PS31 QUAD2 PROT SET	PS31 QUAD1 PROT SET	PS31 QUAD0 PROT SET	PS30 QUAD3 PROT SET	PS30 QUAD2 PROT SET	PS30 QUAD1 PROT SET	PS30 QUAD0 PROT SET	PS29 QUAD3 PROT SET	PS29 QUAD2 PROT SET	PS29 QUAD1 PROT SET	PS29 QUAD0 PROT SET	PS28 QUAD3 PROT SET	PS28 QUAD2 PROT SET	PS28 QUAD1 PROT SET	PS28 QUAD0 PROT SET

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PROT SET	PS27 QUAD2 PROT SET	PS27 QUAD1 PROT SET	PS27 QUAD0 PROT SET	PS26 QUAD3 PROT SET	PS26 QUAD2 PROT SET	PS26 QUAD1 PROT SET	PS26 QUAD0 PROT SET	PS25 QUAD3 PROT SET	PS25 QUAD2 PROT SET	PS25 QUAD1 PROT SET	PS25 QUAD0 PROT SET	PS24 QUAD3 PROT SET	PS24 QUAD2 PROT SET	PS24 QUAD1 PROT SET	PS24 QUAD0 PROT SET

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-60. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31–0]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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET3 and PMPROTCLR3 registers is set to 1.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-63 and described in Table 5-61.

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

Figure 5-63. Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PROT CLR	PS7 QUAD2 PROT CLR	PS7 QUAD1 PROT CLR	PS7 QUAD0 PROT CLR	PS6 QUAD3 PROT CLR	PS6 QUAD2 PROT CLR	PS6 QUAD1 PROT CLR	PS6 QUAD0 PROT CLR	PS5 QUAD3 PROT CLR	PS5 QUAD2 PROT CLR	PS5 QUAD1 PROT CLR	PS5 QUAD0 PROT CLR	PS4 QUAD3 PROT CLR	PS4 QUAD2 PROT CLR	PS4 QUAD1 PROT CLR	PS4 QUAD0 PROT CLR

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PROT CLR	PS3 QUAD2 PROT CLR	PS3 QUAD1 PROT CLR	PS3 QUAD0 PROT CLR	PS2 QUAD3 PROT CLR	PS2 QUAD2 PROT CLR	PS2 QUAD1 PROT CLR	PS2 QUAD0 PROT CLR	PS1 QUAD3 PROT CLR	PS1 QUAD2 PROT CLR	PS1 QUAD1 PROT CLR	PS1 QUAD0 PROT CLR	PS0 QUAD3 PROT CLR	PS0 QUAD2 PROT CLR	PS0 QUAD1 PROT CLR	PS0 QUAD0 PROT CLR

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-61. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET0 and PMPROTCLR0 registers is cleared to 0.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-64 and described in Table 5-62.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-64. Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PROT CLR	PS15 QUAD2 PROT CLR	PS15 QUAD1 PROT CLR	PS15 QUAD0 PROT CLR	PS14 QUAD3 PROT CLR	PS14 QUAD2 PROT CLR	PS14 QUAD1 PROT CLR	PS14 QUAD0 PROT CLR	PS13 QUAD3 PROT CLR	PS13 QUAD2 PROT CLR	PS13 QUAD1 PROT CLR	PS13 QUAD0 PROT CLR	PS12 QUAD3 PROT CLR	PS12 QUAD2 PROT CLR	PS12 QUAD1 PROT CLR	PS12 QUAD0 PROT CLR

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PROT CLR	PS11 QUAD2 PROT CLR	PS11 QUAD1 PROT CLR	PS11 QUAD0 PROT CLR	PS10 QUAD3 PROT CLR	PS10 QUAD2 PROT CLR	PS10 QUAD1 PROT CLR	PS10 QUAD0 PROT CLR	PS9 QUAD3 PROT CLR	PS9 QUAD2 PROT CLR	PS9 QUAD1 PROT CLR	PS9 QUAD0 PROT CLR	PS8 QUAD3 PROT CLR	PS8 QUAD2 PROT CLR	PS8 QUAD1 PROT CLR	PS8 QUAD0 PROT CLR

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-62. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET1 and PMPROTCLR1 registers is cleared to 0.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-65 and described in Table 5-63.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-65. Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 0x48]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PROT CLR	PS23 QUAD2 PROT CLR	PS23 QUAD1 PROT CLR	PS23 QUAD0 PROT CLR	PS22 QUAD3 PROT CLR	PS22 QUAD2 PROT CLR	PS22 QUAD1 PROT CLR	PS22 QUAD0 PROT CLR	PS21 QUAD3 PROT CLR	PS21 QUAD2 PROT CLR	PS21 QUAD1 PROT CLR	PS21 QUAD0 PROT CLR	PS20 QUAD3 PROT CLR	PS20 QUAD2 PROT CLR	PS20 QUAD1 PROT CLR	PS20 QUAD0 PROT CLR

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PROT CLR	PS19 QUAD2 PROT CLR	PS19 QUAD1 PROT CLR	PS19 QUAD0 PROT CLR	PS18 QUAD3 PROT CLR	PS18 QUAD2 PROT CLR	PS18 QUAD1 PROT CLR	PS18 QUAD0 PROT CLR	PS17 QUAD3 PROT CLR	PS17 QUAD2 PROT CLR	PS17 QUAD1 PROT CLR	PS17 QUAD0 PROT CLR	PS16 QUAD3 PROT CLR	PS16 QUAD2 PROT CLR	PS16 QUAD1 PROT CLR	PS16 QUAD0 PROT CLR

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-63. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET2 and PMPROTCLR2 registers is cleared to 0.

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 PPROTSET0, in section 5.3.5. This register is shown in Figure 5-66 and described in Table 5-64.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-66. Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PROT CLR	PS31 QUAD2 PROT CLR	PS31 QUAD1 PROT CLR	PS31 QUAD0 PROT CLR	PS30 QUAD3 PROT CLR	PS30 QUAD2 PROT CLR	PS30 QUAD1 PROT CLR	PS30 QUAD0 PROT CLR	PS29 QUAD3 PROT CLR	PS29 QUAD2 PROT CLR	PS29 QUAD1 PROT CLR	PS29 QUAD0 PROT CLR	PS28 QUAD3 PROT CLR	PS28 QUAD2 PROT CLR	PS28 QUAD1 PROT CLR	PS28 QUAD0 PROT CLR

R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PROT CLR	PS27 QUAD2 PROT CLR	PS27 QUAD1 PROT CLR	PS27 QUAD0 PROT CLR	PS26 QUAD3 PROT CLR	PS26 QUAD2 PROT CLR	PS26 QUAD1 PROT CLR	PS26 QUAD0 PROT CLR	PS25 QUAD3 PROT CLR	PS25 QUAD2 PROT CLR	PS25 QUAD1 PROT CLR	PS25 QUAD0 PROT CLR	PS24 QUAD3 PROT CLR	PS24 QUAD2 PROT CLR	PS24 QUAD1 PROT CLR	PS24 QUAD0 PROT CLR

R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-64. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31–0]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 can be read in both user and privileged modes. *Write*– The bit in PMPROTSET3 and PMPROTCLR3 registers is cleared to 0.

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.

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

This register is shown in Figure 5-67 and described in Table 5-65.

Figure 5-67. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PWR DWN SET	PCS30 PWR DWN SET	PCS29 PWR DWN SET	PCS28 PWR DWN SET	PCS27 PWR DWN SET	PCS26 PWR DWN SET	PCS25 PWR DWN SET	PCS24 PWR DWN SET	PCS23 PWR DWN SET	PCS22 PWR DWN SET	PCS21 PWR DWN SET	PCS20 PWR DWN SET	PCS19 PWR DWN SET	PCS18 PWR DWN SET	PCS17 PWR DWN SET	PCS16 PWR DWN SET

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PWR DWN SET	PCS14 PWR DWN SET	PCS13 PWR DWN SET	PCS12 PWR DWN SET	PCS11 PWR DWN SET	PCS10 PWR DWN SET	PCS9 PWR DWN SET	PCS8 PWR DWN SET	PCS7 PWR DWN SET	PCS6 PWR DWN SET	PCS5 PWR DWN SET	PCS4 PWR DWN SET	PCS3 PWR DWN SET	PCS2 PWR DWN SET	PCS1 PWR DWN SET	PCS0 PWR DWN SET

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-65. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions

Bit	Name	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.

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

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

This register is shown in Figure 5-68 and described in Table 5-71.

Figure 5-68. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PWR DWN SET	PCS62 PWR DWN SET	PCS61 PWR DWN SET	PCS60 PWR DWN SET	PCS59 PWR DWN SET	PCS58 PWR DWN SET	PCS57 PWR DWN SET	PCS56 PWR DWN SET	PCS55 PWR DWN SET	PCS54 PWR DWN SET	PCS53 PWR DWN SET	PCS52 PWR DWN SET	PCS51 PWR DWN SET	PCS50 PWR DWN SET	PCS49 PWR DWN SET	PCS48 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PWR DWN SET	PCS46 PWR DWN SET	PCS45 PWR DWN SET	PCS44 PWR DWN SET	PCS43 PWR DWN SET	PCS42 PWR DWN SET	PCS41 PWR DWN SET	PCS40 PWR DWN SET	PCS39 PWR DWN SET	PCS38 PWR DWN SET	PCS37 PWR DWN SET	PCS36 PWR DWN SET	PCS35 PWR DWN SET	PCS34 PWR DWN SET	PCS33 PWR DWN SET	PCS32 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-66. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions

Bit	Name	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.

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

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

This register is shown in Figure 5-69 and described in Table 5-67.

Figure 5-69. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS31 PWR DWN CLR	PCS30 PWR DWN CLR	PCS29 PWR DWN CLR	PCS28 PWR DWN CLR	PCS27 PWR DWN CLR	PCS26 PWR DWN CLR	PCS25 PWR DWN CLR	PCS24 PWR DWN CLR	PCS23 PWR DWN CLR	PCS22 PWR DWN CLR	PCS21 PWR DWN CLR	PCS20 PWR DWN CLR	PCS19 PWR DWN CLR	PCS18 PWR DWN CLR	PCS17 PWR DWN CLR	PCS16 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS15 PWR DWN CLR	PCS14 PWR DWN CLR	PCS13 PWR DWN CLR	PCS12 PWR DWN CLR	PCS11 PWR DWN CLR	PCS10 PWR DWN CLR	PCS9 PWR DWN CLR	PCS8 PWR DWN CLR	PCS7 PWR DWN CLR	PCS6 PWR DWN CLR	PCS5 PWR DWN CLR	PCS4 PWR DWN CLR	PCS3 PWR DWN CLR	PCS2 PWR DWN CLR	PCS1 PWR DWN CLR	PCS0 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

Table 5-67. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[31–0] PWRDNCLR0		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.

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

Note:
Only those bits that have a slave at the corresponding bit position are implemented. Therefore, the size of this register is device-dependent. Writes to non implemented bits have no effect and reads are 0.

This register is shown in Figure 5-70 and described in Table 5-68.

Figure 5-70. Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS63 PWR DWN CLR	PCS62 PWR DWN CLR	PCS61 PWR DWN CLR	PCS60 PWR DWN CLR	PCS59 PWR DWN CLR	PCS58 PWR DWN CLR	PCS57 PWR DWN CLR	PCS56 PWR DWN CLR	PCS55 PWR DWN CLR	PCS54 PWR DWN CLR	PCS53 PWR DWN CLR	PCS52 PWR DWN CLR	PCS51 PWR DWN CLR	PCS50 PWR DWN CLR	PCS49 PWR DWN CLR	PCS48 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS47 PWR DWN CLR	PCS46 PWR DWN CLR	PCS45 PWR DWN CLR	PCS44 PWR DWN CLR	PCS43 PWR DWN CLR	PCS42 PWR DWN CLR	PCS41 PWR DWN CLR	PCS40 PWR DWN CLR	PCS39 PWR DWN CLR	PCS38 PWR DWN CLR	PCS37 PWR DWN CLR	PCS36 PWR DWN CLR	PCS35 PWR DWN CLR	PCS34 PWR DWN CLR	PCS33 PWR DWN CLR	PCS32 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-68. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PCS[63–32] PWRDNSET		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.

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 5.3.5.

This arrangement is the same for bits of PS8 to PS31, presented in section 5.3.18—section 5.3.24. This register holds bits for PS0 to PS7. This register is shown in Figure 5-71 and described in Table 5-69.

Note:

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

Figure 5-71. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PWR DWN SET	PS7 QUAD2 PWR DWN SET	PS7 QUAD1 PWR DWN SET	PS7 QUAD0 PWR DWN SET	PS6 QUAD3 PWR DWN SET	PS6 QUAD2 PWR DWN SET	PS6 QUAD1 PWR DWN SET	PS6 QUAD0 PWR DWN SET	PS5 QUAD3 PWR DWN SET	PS5 QUAD2 PWR DWN SET	PS5 QUAD1 PWR DWN SET	PS5 QUAD0 PWR DWN SET	PS4 QUAD3 PWR DWN SET	PS4 QUAD2 PWR DWN SET	PS4 QUAD1 PWR DWN SET	PS4 QUAD0 PWR DWN SET
R/WP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PWR DWN SET	PS3 QUAD2 PWR DWN SET	PS3 QUAD1 PWR DWN SET	PS3 QUAD0 PWR DWN SET	PS2 QUAD3 PWR DWN SET	PS2 QUAD2 PWR DWN SET	PS2 QUAD1 PWR DWN SET	PS2 QUAD0 PWR DWN SET	PS1 QUAD3 PWR DWN SET	PS1 QUAD2 PWR DWN SET	PS1 QUAD1 PWR DWN SET	PS1 QUAD0 PWR DWN SET	PS0 QUAD3 PWR DWN SET	PS0 QUAD2 PWR DWN SET	PS0 QUAD1 PWR DWN SET	PS0 QUAD0 PWR DWN SET
R/WP-1															

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-69. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions

Bit	Name	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

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 5.3.17. This register is shown in Figure 5-72 and described in Table 5-70.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-72. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 0x84]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PWR DWN SET	PS15 QUAD2 PWR DWN SET	PS15 QUAD1 PWR DWN SET	PS15 QUAD0 PWR DWN SET	PS14 QUAD3 PWR DWN SET	PS14 QUAD2 PWR DWN SET	PS14 QUAD1 PWR DWN SET	PS14 QUAD0 PWR DWN SET	PS13 QUAD3 PWR DWN SET	PS13 QUAD2 PWR DWN SET	PS13 QUAD1 PWR DWN SET	PS13 QUAD0 PWR DWN SET	PS12 QUAD3 PWR DWN SET	PS12 QUAD2 PWR DWN SET	PS12 QUAD1 PWR DWN SET	PS12 QUAD0 PWR DWN SET

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PWR DWN SET	PS11 QUAD2 PWR DWN SET	PS11 QUAD1 PWR DWN SET	PS11 QUAD0 PWR DWN SET	PS10 QUAD3 PWR DWN SET	PS10 QUAD2 PWR DWN SET	PS10 QUAD1 PWR DWN SET	PS10 QUAD0 PWR DWN SET	PS9 QUAD3 PWR DWN SET	PS9 QUAD2 PWR DWN SET	PS9 QUAD1 PWR DWN SET	PS9 QUAD0 PWR DWN SET	PS8 QUAD3 PWR DWN SET	PS8 QUAD2 PWR DWN SET	PS8 QUAD1 PWR DWN SET	PS8 QUAD0 PWR DWN SET

R/WP-1

R/WP = Read in all modes, any time, write in privileged mode only, *-n* = Value after reset

.

Table 5-70. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions

Bit	Name	Value	Description
31–0	PS[15–8]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 PSPWRDWNSET1 and PSPWRDWNCLR1 registers is set to 1

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 5.3.17. This register is shown in Figure 5-66 and described in Table 5-64.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-73. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PWR DWN SET	PS23 QUAD2 PWR DWN SET	PS23 QUAD1 PWR DWN SET	PS23 QUAD0 PWR DWN SET	PS22 QUAD3 PWR DWN SET	PS22 QUAD2 PWR DWN SET	PS22 QUAD1 PWR DWN SET	PS22 QUAD0 PWR DWN SET	PS21 QUAD3 PWR DWN SET	PS21 QUAD2 PWR DWN SET	PS21 QUAD1 PWR DWN SET	PS21 QUAD0 PWR DWN SET	PS20 QUAD3 PWR DWN SET	PS20 QUAD2 PWR DWN SET	PS20 QUAD1 PWR DWN SET	PS20 QUAD0 PWR DWN SET

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PWR DWN SET	PS19 QUAD2 PWR DWN SET	PS19 QUAD1 PWR DWN SET	PS19 QUAD0 PWR DWN SET	PS18 QUAD3 PWR DWN SET	PS18 QUAD2 PWR DWN SET	PS18 QUAD1 PWR DWN SET	PS18 QUAD0 PWR DWN SET	PS17 QUAD3 PWR DWN SET	PS17 QUAD2 PWR DWN SET	PS17 QUAD1 PWR DWN SET	PS17 QUAD0 PWR DWN SET	PS16 QUAD3 PWR DWN SET	PS16 QUAD2 PWR DWN SET	PS16 QUAD1 PWR DWN SET	PS16 QUAD0 PWR DWN SET

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-71. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions

Bit	Name	Value	Description
31–0	PS[23–16]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 PSPWRDWNSET2 and PSPWRDWNCLR2 registers is set to 1

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 5.3.17. This register is shown in Figure 5-74 and described in Table 5-72.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-74. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PWR DWN SET	PS31 QUAD2 PWR DWN SET	PS31 QUAD1 PWR DWN SET	PS31 QUAD0 PWR DWN SET	PS30 QUAD3 PWR DWN SET	PS30 QUAD2 PWR DWN SET	PS30 QUAD1 PWR DWN SET	PS30 QUAD0 PWR DWN SET	PS29 QUAD3 PWR DWN SET	PS29 QUAD2 PWR DWN SET	PS29 QUAD1 PWR DWN SET	PS29 QUAD0 PWR DWN SET	PS28 QUAD3 PWR DWN SET	PS28 QUAD2 PWR DWN SET	PS28 QUAD1 PWR DWN SET	PS28 QUAD0 PWR DWN SET

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PWR DWN SET	PS27 QUAD2 PWR DWN SET	PS27 QUAD1 PWR DWN SET	PS27 QUAD0 PWR DWN SET	PS26 QUAD3 PWR DWN SET	PS26 QUAD2 PWR DWN SET	PS26 QUAD1 PWR DWN SET	PS26 QUAD0 PWR DWN SET	PS25 QUAD3 PWR DWN SET	PS25 QUAD2 PWR DWN SET	PS25 QUAD1 PWR DWN SET	PS25 QUAD0 PWR DWN SET	PS24 QUAD3 PWR DWN SET	PS24 QUAD2 PWR DWN SET	PS24 QUAD1 PWR DWN SET	PS24 QUAD0 PWR DWN SET

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; *-n* = Value after reset

.

Table 5-72. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions

Bit	Name	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

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 5.3.17. This register is shown in Figure 5-75 and described in Table 5-73.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-75. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = 0xA0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS7 QUAD3 PWR DWN CLR	PS7 QUAD2 PWR DWN CLR	PS7 QUAD1 PWR DWN CLR	PS7 QUAD0 PWR DWN CLR	PS6 QUAD3 PWR DWN CLR	PS6 QUAD2 PWR DWN CLR	PS6 QUAD1 PWR DWN CLR	PS6 QUAD0 PWR DWN CLR	PS5 QUAD3 PWR DWN CLR	PS5 QUAD2 PWR DWN CLR	PS5 QUAD1 PWR DWN CLR	PS5 QUAD0 PWR DWN CLR	PS4 QUAD3 PWR DWN CLR	PS4 QUAD2 PWR DWN CLR	PS4 QUAD1 PWR DWN CLR	PS4 QUAD0 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS3 QUAD3 PWR DWN CLR	PS3 QUAD2 PWR DWN CLR	PS3 QUAD1 PWR DWN CLR	PS3 QUAD0 PWR DWN CLR	PS2 QUAD3 PWR DWN CLR	PS2 QUAD2 PWR DWN CLR	PS2 QUAD1 PWR DWN CLR	PS2 QUAD0 PWR DWN CLR	PS1 QUAD3 PWR DWN CLR	PS1 QUAD2 PWR DWN CLR	PS1 QUAD1 PWR DWN CLR	PS1 QUAD0 PWR DWN CLR	PS0 QUAD3 PWR DWN CLR	PS0 QUAD2 PWR DWN CLR	PS0 QUAD1 PWR DWN CLR	PS0 QUAD0 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

.

Table 5-73. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions

Bit	Name	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.

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 5.3.17. This register is shown in Figure 5-76 and described in Table 5-74.

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

Figure 5-76. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) [offset = 0xA4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS15 QUAD3 PWR DWN CLR	PS15 QUAD2 PWR DWN CLR	PS15 QUAD1 PWR DWN CLR	PS15 QUAD0 PWR DWN CLR	PS14 QUAD3 PWR DWN CLR	PS14 QUAD2 PWR DWN CLR	PS14 QUAD1 PWR DWN CLR	PS14 QUAD0 PWR DWN CLR	PS13 QUAD3 PWR DWN CLR	PS13 QUAD2 PWR DWN CLR	PS13 QUAD1 PWR DWN CLR	PS13 QUAD0 PWR DWN CLR	PS12 QUAD3 PWR DWN CLR	PS12 QUAD2 PWR DWN CLR	PS12 QUAD1 PWR DWN CLR	PS12 QUAD0 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS11 QUAD3 PWR DWN CLR	PS11 QUAD2 PWR DWN CLR	PS11 QUAD1 PWR DWN CLR	PS11 QUAD0 PWR DWN CLR	PS10 QUAD3 PWR DWN CLR	PS10 QUAD2 PWR DWN CLR	PS10 QUAD1 PWR DWN CLR	PS10 QUAD0 PWR DWN CLR	PS9 QUAD3 PWR DWN CLR	PS9 QUAD2 PWR DWN CLR	PS9 QUAD1 PWR DWN CLR	PS9 QUAD0 PWR DWN CLR	PS8 QUAD3 PWR DWN CLR	PS8 QUAD2 PWR DWN CLR	PS8 QUAD1 PWR DWN CLR	PS8 QUAD0 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

Table 5-74. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions

Bit	Name	Value	Description
31–0	PS[15–8]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 PSPWRDWNSET1 and PSPWRDWNCLR1 registers is cleared to 0.

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 5.3.17. This register is shown in Figure 5-77 and described in Table 5-75.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-77. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = 0xA8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS23 QUAD3 PWR DWN CLR	PS23 QUAD2 PWR DWN CLR	PS23 QUAD1 PWR DWN CLR	PS23 QUAD0 PWR DWN CLR	PS22 QUAD3 PWR DWN CLR	PS22 QUAD2 PWR DWN CLR	PS22 QUAD1 PWR DWN CLR	PS22 QUAD0 PWR DWN CLR	PS21 QUAD3 PWR DWN CLR	PS21 QUAD2 PWR DWN CLR	PS21 QUAD1 PWR DWN CLR	PS21 QUAD0 PWR DWN CLR	PS20 QUAD3 PWR DWN CLR	PS20 QUAD2 PWR DWN CLR	PS20 QUAD1 PWR DWN CLR	PS20 QUAD0 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS19 QUAD3 PWR DWN CLR	PS19 QUAD2 PWR DWN CLR	PS19 QUAD1 PWR DWN CLR	PS19 QUAD0 PWR DWN CLR	PS18 QUAD3 PWR DWN CLR	PS18 QUAD2 PWR DWN CLR	PS18 QUAD1 PWR DWN CLR	PS18 QUAD0 PWR DWN CLR	PS17 QUAD3 PWR DWN CLR	PS17 QUAD2 PWR DWN CLR	PS17 QUAD1 PWR DWN CLR	PS17 QUAD0 PWR DWN CLR	PS16 QUAD3 PWR DWN CLR	PS16 QUAD2 PWR DWN CLR	PS16 QUAD1 PWR DWN CLR	PS16 QUAD0 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

Table 5-75. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions

Bit	Name	Value	Description
31–0	PS[23–16]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 PSPWRDWNSET2 and PSPWRDWNCLR2 registers is cleared to 0.

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 5.3.17. This register is shown in Figure 5-78 and described in Table 5-76.

> **Note:**
> Only those bits that have a slave at the corresponding bit position are implemented. Writes to non implemented bits have no effect and reads are 0.

Figure 5-78. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) [offset = 0xAC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PS31 QUAD3 PWR DWN CLR	PS31 QUAD2 PWR DWN CLR	PS31 QUAD1 PWR DWN CLR	PS31 QUAD0 PWR DWN CLR	PS30 QUAD3 PWR DWN CLR	PS30 QUAD2 PWR DWN CLR	PS30 QUAD1 PWR DWN CLR	PS30 QUAD0 PWR DWN CLR	PS29 QUAD3 PWR DWN CLR	PS29 QUAD2 PWR DWN CLR	PS29 QUAD1 PWR DWN CLR	PS29 QUAD0 PWR DWN CLR	PS28 QUAD3 PWR DWN CLR	PS28 QUAD2 PWR DWN CLR	PS28 QUAD1 PWR DWN CLR	PS28 QUAD0 PWR DWN CLR

R/WP-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS27 QUAD3 PWR DWN CLR	PS27 QUAD2 PWR DWN CLR	PS27 QUAD1 PWR DWN CLR	PS27 QUAD0 PWR DWN CLR	PS26 QUAD3 PWR DWN CLR	PS26 QUAD2 PWR DWN CLR	PS26 QUAD1 PWR DWN CLR	PS26 QUAD0 PWR DWN CLR	PS25 QUAD3 PWR DWN CLR	PS25 QUAD2 PWR DWN CLR	PS25 QUAD1 PWR DWN CLR	PS25 QUAD0 PWR DWN CLR	PS24 QUAD3 PWR DWN CLR	PS24 QUAD2 PWR DWN CLR	PS24 QUAD1 PWR DWN CLR	PS24 QUAD0 PWR DWN CLR

R/WP-1

R = Read in all modes; WP = Write in privileged mode only; -*n* = Value after reset

Table 5-76. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions

Bit	Name	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.

Chapter 6

SPNU489C-February 2012

Programmable Built-In Self Test (PBIST) Module

Topic Page

6.1 Overview

The PBIST (Programmable Built-In Self Test) architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances. This document describes how the PBIST architecture is used to test embedded memories in based TI microcontrollers. This document is targeted towards describing PBIST from an application self test perspective.

6.1.1 Features of PBIST

- Extensive instruction set allows multiple kinds of algorithms
- Supports multiple configurability with memory-mapped configuration registers
- Status registers available that allows status checking
- Simple register set allows easy programming
- Supports algorithm development before and after silicon
- External and CPU controlled modes of operation
- Supports ROM-based testing
- Peripheral Bus compatible Config bus interface
- Supports ROM testing
- Supports multiple clock domains for testing different memories
- Clock gating to conserve power
- Data-width is soft programmable
- Supports any number of background patterns
- The algorithms, background patterns and other information is generated and stored in the on-chip ROM.
- Supports Test execution speed up to 100MHz peripheral clock.

6.2 PBIST vs. Embedded CPU

The PBIST architecture consists of a small CPU with an instruction set targeted specifically towards testing RAM memories. This CPU includes both control and instruction registers necessary to execute the individual memory algorithms. Once an algorithm is loaded into the instruction registers, it can be run on multiple memories of different sizes or types by changing the control register to minimize test load overhead. The memory configuration information is stored into on-chip ROM. The test algorithm code is also loaded into on-chip ROM.

The functionality of PBIST controller is similar to using an embedded CPU to test the memories in the terms of the flexibility to add, remove or modify algorithms without changing the silicon. It differs in these ways:

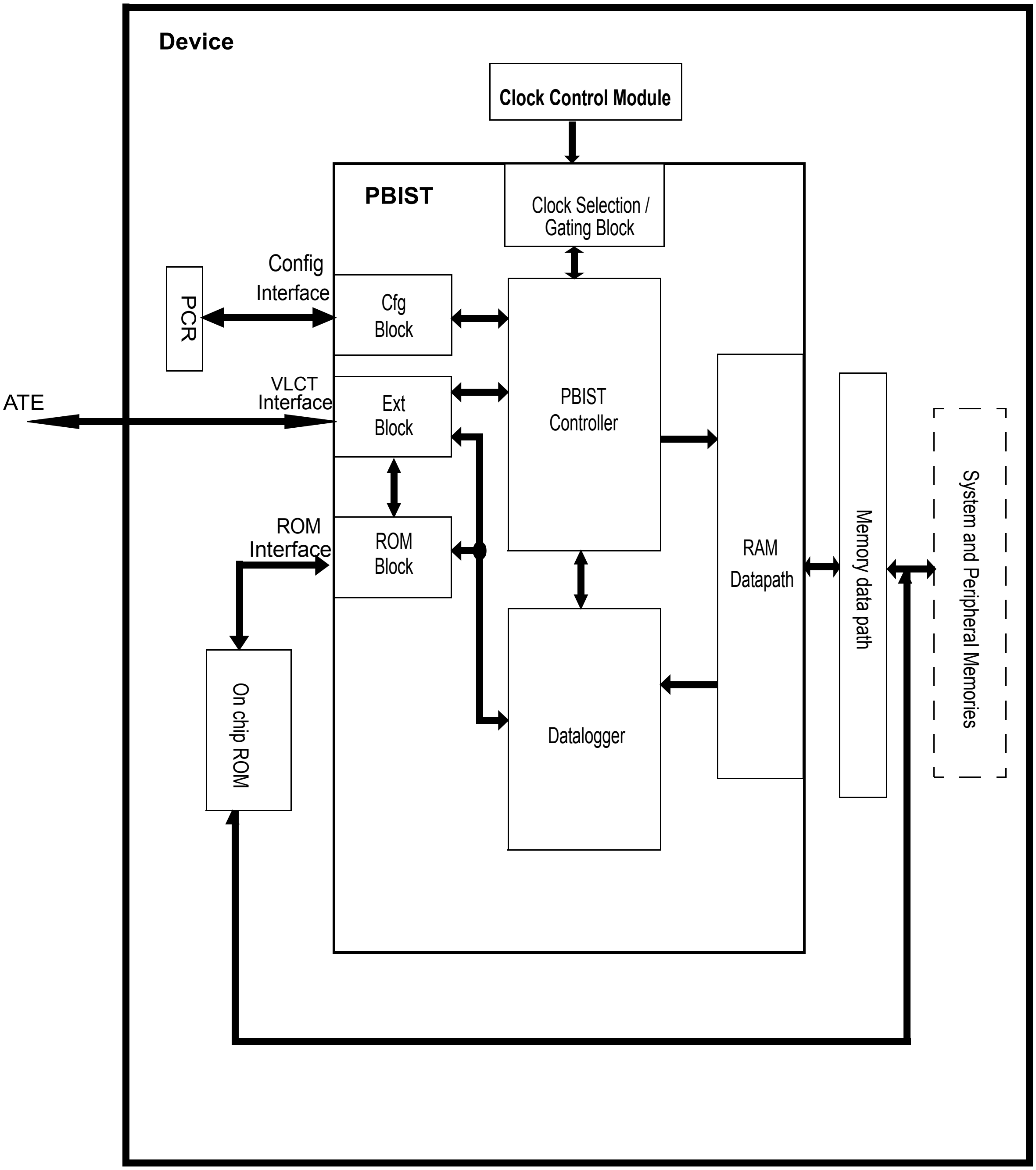
- PBIST does not need to be tightly coupled to the memory to provide algorithmic coverage, while embedded CPUs do. Embedded CPUs have difficulty covering memory instances used outside the CPU subsystem.
- PBIST is easily programmable. Embedded CPUs are designed for their targeted use and are often not easily programmed for memory test algorithms.
  - Embedded CPUs require more time to develop the algorithm code.
  - The algorithm code for embedded CPUs is significantly larger than that needed for PBIST.
  - For embedded CPUs, the code must be optimized for each memory type and size.
  - The embedded CPU is often much larger than the PBIST engine and therefore more susceptible to process defects that will render it unable to test the memories on some percentage of the devices.
  - For embedded CPU testing algorithms need to be re-coded for every new version of the CPU

6.3 PBIST block diagram

The Figure 6-1 illustrates the basic PBIST blocks and its wrapper logic for the device.

The ATE interface is used only during the production tests. For memory self test during application, PCR (Peripheral Bus) and ROM interfaces are used.

Figure 6-1. PBIST Block Diagram



6.3.1 The On-chip ROM

The On-chip ROM contains the information regarding the algorithms and memories to be tested. It contains the algorithm header, microcode, the PBIST RAM grouping information and header information.

- The algorithm header contains the following:
  - Size of the algorithm micro code.
  - Valid bit to enable the execution of the next algorithm on the onchip ROM. The 1st algorithm in the onchip ROM is always executed by default.
  - Retention mode enable bit to indicate the PBIST to stop executing the next algorithm until driven by an retention resume signal
  - MISR mode enable bit to perform an MISR check after execution of each selected individual RAM group for an algorithm
  - Cumulative MISR mode to perform an MISR check after cumulative execution of all selected RAM group for an algorithm.
  - Six bit unique id for the algorithm. An total of 64 algorithms can be stored in the Onchip ROM
- The algorithm microcode section contains the binary code to be loaded into the PBIST controller register file for execution of the algorithm.
- An RAM group header contains the following:
  - Size of the RAM group information.
  - Valid bit to enable the execution of the next RAM group information on the onchip ROM. The 1st RAM group information in the onchip ROM is always executed by default.
- The RAM group information contains the following:
  - The At-speed clock source for the RAM test
  - An unique RAM select id through RGS (RAM group select - Hardware memory group, user can map any failure to particular ‘RAM GROUP’ since failure reporting is in terms of RGS) and RDS (return data select - Hardware mux select to choose between the data returned from the various RAMs within a RGS).
  - Read data compare width up to 64 bits can be compared.
  - Read compare latency to facilitate pipelining to meet timing on the data path between the PBIST controller and the memory under test.
  - RAM latency programmation capability to generate wait states between RAM’s accesses.
  - 2 address constants to indicate the start addresses for each port during the RAM group test.
  - 2 loop count constants to account for the memory depth for each port during the RAM group test.
  - 2 increment constant to account for RAM mux factors during row march or column march type of test.
  - Write data path chip select register
  - RAM group test start enable

15 RAM groups are supported in this device with up to 31 individual RAM information per RAM group.

6.3.2 PBIST interface to the register file/ configuration registers:

There are 3 interfaces to the PBIST register file and configuration registers

1. The ROM interface: This is the interface for the PBIST controller to the On-chip ROM to load the algorithms into the internal register file and configure the PBIST control registers (configuration registers) which will be the primary source of the algorithm microcode and RAM information for RAM/ROM test during application memory self test. The ROM information will be hard-coded during device development.
2. VLCT interface: This interface will be used as a complement to the ROM interface to develop additional algorithms if required in addition to the onchip production algorithms. It will also provide selection capability

of algorithm targeted to different production test example Production memory test, Production Iddq testing etc. This interface is used for production tests only and not required during application self test.

3. Config interface (Peripheral Bus interface): This interface provides access to PBIST registers. This will provide the capability to select the algorithm and RAM groups for PBIST test from on the onchip ROM based on the applications requirements.

The simplest way to set up an application self test can be achieved by programming the algorithm info and RAM group info override registers (OVER) and triggering the ROM based memory self test.

6.3.3 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 test sequence is that for each selected algorithm the PBIST controller executes the 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 Table 6-1and Table 6-2 for RAM grouping and Algorithm informations.

6.3.4 Clock selection/Gating Logic

This logic selects the at-speed functional clock associated to each Memory/RAM group and its associated clock gating for low power consumption.

6.3.5 Reset Control

Reset to the PBIST module is controlled from the MSTGCR register in the system module. Changing MSTGCR from 0x5 to 0xA would generate a reset to the module, resetting the PBIST logic, data paths and registers.

Note:
MSTGCR = 0x5 does not represent the disabled state of the PBIST controller. It is possible to run the PBIST when MSTGCR = 0x5. However, a reset is required before starting the PBIST controller which is possible from changing MSTGCR = 0x5 to 0xA.

The PBIST module can be accessed only after the clock to PBIST is activated by writing a 0x3 to PACT[1:0] register.

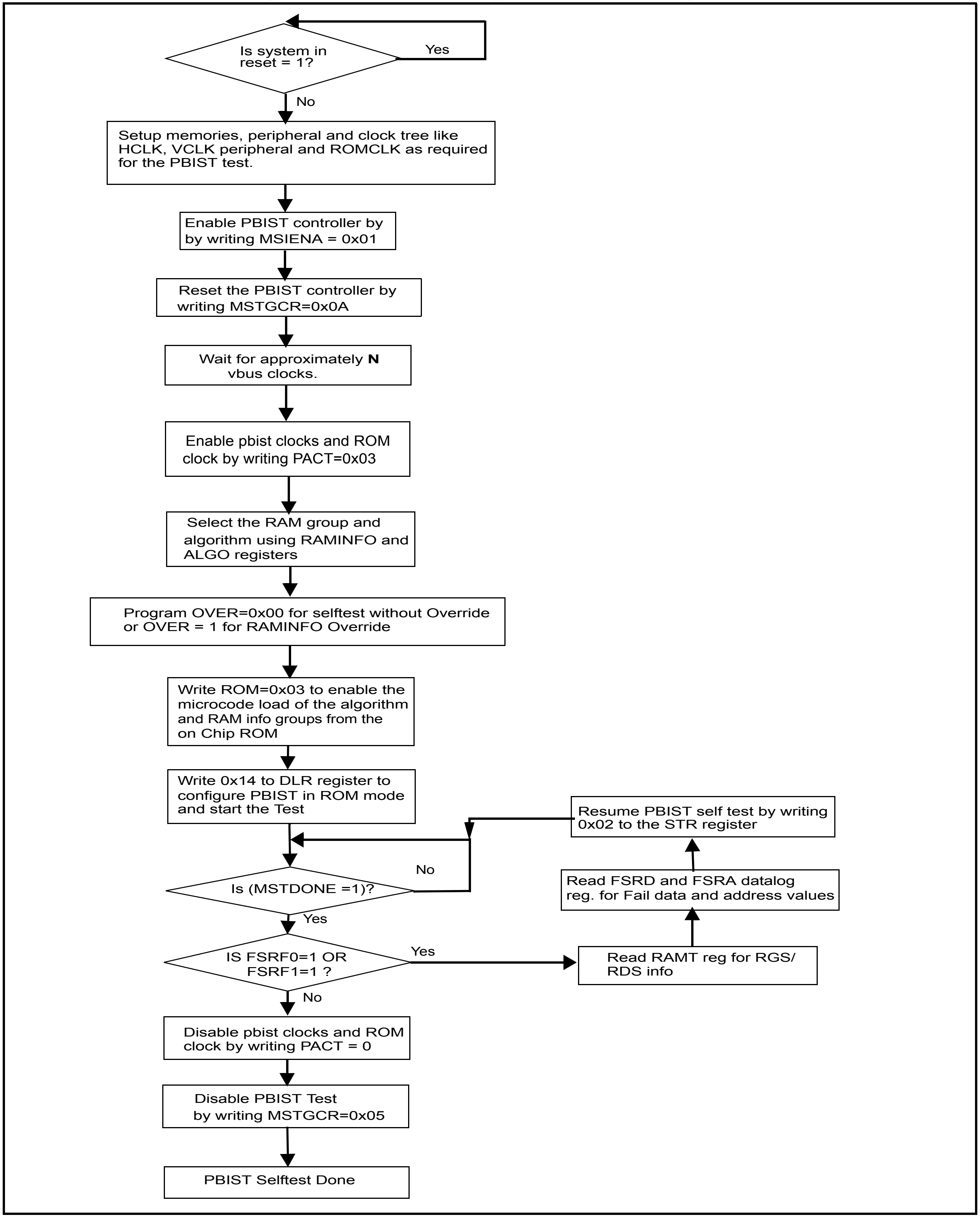
6.3.6 Testing ROM

When running algorithms on ROM memories, it is required to program the ROM clock divide prescaler in MSTGCR register in system module. This is required as ROM may not support testing at maximum device frequency.

6.4 PBIST self test flow

The Figure 6-1 illustrates the PBIST Memory Self Test flow.

Figure 6-2. PBIST Memory Self Test Flow Diagram


Is system in reset = 1?
Yes
No
Setup memories, peripheral and clock tree like HCLK, VCLK peripheral and ROMCLK as required for the PBIST test.
Enable PBIST controller by by writing MSIENA = 0x01
Reset the PBIST controller by writing MSTGCR=0x0A
Wait for approximately N vbus clocks.
Enable pbist clocks and ROM clock by writing PACT=0x03
Select the RAM group and algorithm using RAMINFO and ALGO registers
Program OVER=0x00 for selftest without Override or OVER = 1 for RAMINFO Override
Write ROM=0x03 to enable the microcode load of the algorithm and RAM info groups from the on Chip ROM
Write 0x14 to DLR register to configure PBIST in ROM mode and start the Test
Resume PBIST self test by writing 0x02 to the STR register
No
Is (MSTDONE =1)?
Read FSRD and FSRA datalog reg. for Fail data and address values
Yes
IS FSRF0=1 OR FSRF1=1 ?
Yes
Read RAMT reg for RGS/ RDS info
No
Disable pbist clocks and ROM clock by writing PACT = 0
Disable PBIST Test by writing MSTGCR=0x05
PBIST Selftest Done

All of the Memory self test and RAM grouping information will be stored on the On-chip ROM. The information to setup the device to run at-speed, setup PBIST controller, select Memory self test algorithms and RAM groups and self test kickoff though the PBIST configuration and system registers will be done by the user program.

6.4.1 PBIST Selftest Program Sequence

1. Wait for the device reset to get deasserted. Once the reset is deasserted configure the PLL registers. Program the HCLK to PBIST ROM clock ratio by configuring bits 9:8 in the MSTGCR register of the system module. This will be device specific based on the maximum ROM operating frequency.
2. If required to run at high frequency select PLL as clock Source. Once PLL valid is asserted, configure the GHVSRC register in the system module to select PLL as the clock source.
3. Enable PBIST Controller by setting bit 1 of MSIENA register in system module. Without this bit set the MSTDONE flag will not get set after test completion.
4. Enable the PBIST self test by writing a value of 0x0A 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 0x03 to PACT register to enable the Pbist internal clocks and ROM interface clock.
7. Program the ALGO register to decide which algorithm from the instruction ROM must be selected. (Default value of ALGO register. is all 1's, means 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 (OVER =0x00), user should make sure that only the algorithms that run on similar RAMs are selected. If a single port algorithm is selected in ALGO register, the RINFOL and RINFOU register must select only the single port RAM's. Same applies for two port RAM's.

8. Program OVER=0x01 to run PBIST selftest without RAM override. Program OVER=0x00 to run Pbist selftest with RAM Override.
9. Write a value of 0x03 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 0x14 to configure the PBIST run in ROM mode and to enable the config access. With this the PBIST selftest is kicked-off.
11. Wait for the PBIST selftest done by polling MSTDONE bit of MSTCGSTAT register in System Module.
12. Once Self test is completed, check the Fail Status registers FSRF0 and FSRF1.

    In case there is a failure (FSRF0 or FSRF1=0x01)

    i Read RAMT register which indicates the RGS and RDS values of the failure RAM

    ii Read FSRC0 and FSRC1 registers which contains the failure count

    iii Read FSRA0 and FSRA1 registers which contains the address of first failure

    iv Read FSRDL0 and FSRDL1 registers which contains the failure data.

    v .Write a value of 0x02 to the STR register to resume the test.

In case there is no failure (FSRF0 and FSRF1=0x00) the Memory self test in completed.

i) Disable the PBIST internal and ROM clocks by writing a 0x00 to the PACT register.

ii) Disable the PBIST self test by writing a value of 0x05 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 completly lost. User software must take care of data backup if required. Typically the PBIST tests are carried out at the beginning of Application software.

Note:
Memory test fail information is reported in terms of RGS:RDS and not RAM GROUP. There is no relationship between the RAM GROUP and RGS. They can be same or different.

6.5 Memory test algorithms on the On-chip ROM

This section provides a brief description of the test algorithms that are available on the device for application 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 row and column decode is such that the targeted bits (and only the targeted bits) are affected on a write
     - The row and column decode is such that only the targeted bits are presented to the boundary of the memory on a specified read.
     - This level of coverage was initially managed via walking a 1 through a sea of 0s and inversely walking a 0 through a sea of 1s. However, this takes a significantly long time to execute. The layout of the array allows this to be done in a more parallel manner because:
     - The bits that align to the specific external bus I/O are packed together in sticks. This allows the sticks to be tested in parallel.
     - Write disturb failures will remain until the disturbed location is rewritten.
   - The basic operation of the march is to initialize the array to a know pattern, then march a different pattern to the memory with verification.
   - 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 algorithms is to target the global column decode Logic.

6.6 RAM Grouping and Algorithm

Table 6-1 gives the list of RAM groups and their types supported on the device. Table 6-2 maps the different algorithm supported in application mode for the RAM groups with the background patterns used for the particular algorithm.

Table 6-1. RAM Grouping

RAM Group	Module	Memory Type	RGS/RDS
1	PBIST ROM	ROM	0/1
2	STC ROM	ROM	13/1
3	DCAN1	Single Port	1/0..2
4	DCAN2	Single Port	2/0..2
5	DCAN3	Single Port	3/0..2
6	ESRAM	Single Port	4/21..22
7	MIBSPI	Single Port	5/0..5
8	VIM	Single Port	6/0
9	MibADC	Two Port	7/0..1
10	DMA	Two Port	8/0..5
11	NHET	Two Port	9/0..11
12	HET TU	Two Port	10/0..5
13	RTP	Two Port	11/0..8
14	Flexray	Single Port	12/0..7

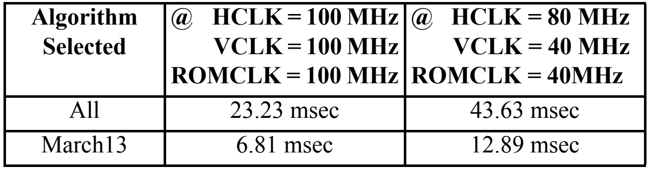
Table 6-2. Algorithm Mapping

Algorithm No	Module	Memory Type	Valid RAM Groups	Available background patterns
1	triple_read_slow_read	ROM	1,2	
2	triple_read_fast_read	ROM	1,2	
3	march13n_red	Single Port	3,4,5,6,7,8,14	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3
4	march13n_red	Two Port	9,10,11,12,13	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3
5	down1A_red	Single Port	3,4,5,6,7,8,14	0xFFFFFFFF, 0xAAAAAAAA
6	down1A_red	Two Port	9,10,11,12,13	0xFFFFFFFF, 0xAAAAAAAA
7	mapcolumn	Single Port	3,4,5,6,7,8,14	0xFFFFFFFF, 0x00000000
8	mapcolumn	Two Port	9,10,11,12,13	0xFFFFFFFF, 0x00000000
9	precharge	Single Port	3,4,5,6,7,8,14	0xFFFFFFFF, 0x00000000
10	precharge	Two Port	9,10,11,12,13	0xFFFFFFFF, 0x00000000
11	dtxn2	Single Port	3,4,5,6,7,8,14	0xFFFFFFFF, 0x00000000
12	dtxn2	Two Port	9,10,11,12,13	0xFFFFFFFF, 0x00000000
13	pmos_open	Single Port	3,4,5,6,7,8,14	0xFFFFFFFF, 0x00000000
14	pmos_open	Two Port	9,11,12,13	0xFFFFFFFF, 0x00000000
15	pmos_open_slice1	Two Port	10	0xFFFFFFFF, 0x00000000
16	pmos_open_slice2	Two Port	10	0xFFFFFFFF, 0x00000000

Note:
March13 is the most recommended algorithm for the memory self test.

Table 6-3 gives the Typical PBIST execution times covering all the RAM groups in the device at different clock rates.

Table 6-3. Typical PBIST Execution Times



Algorithm Selected	@ HCLK = 100 MHz
	VCLK = 100 MHz
	ROMCLK = 100 MHz
		@ HCLK = 80 MHz
		VCLK = 40 MHz
		ROMCLK = 40MHz

All	23.23 msec	43.63 msec
March13	6.81 msec	12.89 msec

6.7 PBIST Control Registers

PBIST uses configuration registers for programming the algorithm and it's 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 0xFFFF E400.

Note:
There is no watchdog functionality implemented in the PBIST controller. If a bad code is executed, the PBIST will run forever. The PBIST controller does not guard against this situation.

Note:
Registers are accessible only when the clock to PBIST controller is active. The clock is activated by first writing 0x3 to PACT register.

Note:
Registers from offset 0x0000 to 0x0100 are used by PBIST microcontroller for microcode execution and should not be modified by the application self test.

Table 6-4. Registers

Offset Address† Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0100 Variable Address Register 0 (A0) Page 203	Reserved															
	A0															
0x0104 Variable Address Register 1 (A1) Page 203	Reserved															
	A1															
0x0108 Variable Address Register 2 (A2) Page 203	Reserved															
	A2															
0x010C Variable Address Register 3 (A3) Page 203	Reserved															
	A3															
0x0110 Variable Loop Count Register 0 (L0) Page 204	Reserved															
	L0															

Table 6-4. Registers (Continued)

0x0114 Variable Loop Count Register 1 (L1) Page 204	Reserved
	L1
0x0118 Variable Loop Count RegisterPage 204 2 (L2) Page 204	Reserved
	L2
0x011C Variable Loop Count Register 3 (L3) Page 204	Reserved
	L3
0x0120 DD0 Data Register Page 205	D1
	D0
0x0124 DE0 Data Register Page 206	E1
	E0
0x0128 Reserved	Reserved
	Reserved
0x012C Reserved	Reserved
	Reserved
0x0130 Constant Address Register 0 (CA0) Page 207	Reserved
	CA0
0x0134 Constant Address Register 1 (CA1) Page 207	Reserved
	CA1

Table 6-4. Registers (Continued)

0x0138 Constant Address Register 2 (CA2) Page 207	Reserved
	CA2
0x013C Constant Address Register 3 (CA3) Page 207	Reserved
	CA3
0x0140 Constant Loop Count Register 0 (CL0) Page 208	Reserved
	CL0
0x0144 Constant Loop Count Register 1 (CL1) Page 208	Reserved
	CL1
0x0148 Constant Loop Count Register 2 (CL2) Page 208	Reserved
	CL2
0x014C Constant Loop Count Register 3 (CL3) Page 208	Reserved
	CL3
0x0150 Constant Increment Register 0 (I0) Page 209	Reserved
	I0
0x0154 Constant Increment Register 1 (I1) Page 209	Reserved
	I1
0x0148 Constant Increment Register 2 (I2) Page 209	Reserved
	I2

Table 6-4. Registers (Continued)

<table>
<tr><td rowspan="2">0x015C
Constant Increment
Register 3 (I3)
Page 209</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">I3</td></tr>
<tr><td rowspan="2">0x0160
RAMT
Page 210</td><td colspan="8">RGS</td><td colspan="8">RDS</td></tr>
<tr><td colspan="8">DWR</td><td colspan="2">SMS</td><td colspan="4">PLS</td><td colspan="2">RLS</td></tr>
<tr><td rowspan="2">0x0164
DLR
Page 211</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="5">Reserved</td><td>DLR10</td><td>DLR9</td><td>DLR8</td><td>DLR7</td><td>DLR6</td><td>DLR5</td><td>DLR4</td><td>DLR3</td><td>DLR2</td><td>DLR1</td><td>DLR0</td></tr>
<tr><td rowspan="2">0x0168
CMS
Page 214</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">CMS</td></tr>
<tr><td rowspan="2">0x016C
STR
Page 215</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="11">Reserved</td><td colspan="5">STR</td></tr>
<tr><td rowspan="2">0x0170
Reserved</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td rowspan="2">0x0174
Reserved</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td rowspan="2">0x0178
CSR
Page 216</td><td colspan="8">CSR3</td><td colspan="8">CSR2</td></tr>
<tr><td colspan="8">CSR1</td><td colspan="8">CSR0</td></tr>
<tr><td rowspan="2">0x017C
FDLY
Page 217</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">FDLY</td></tr>
</table>

Table 6-4. Registers (Continued)

<table>
<tr><td rowspan="2">0x0180
PACT
Page 218</td><td colspan="5">Reserved</td></tr>
<tr><td colspan="3">Reserved</td><td>PACT1</td><td>PACT0</td></tr>
<tr><td rowspan="2">0x0184
PBIST ID
Page 219</td><td colspan="5">Reserved</td></tr>
<tr><td>Reserved</td><td colspan="4">PBISTID[7:0]</td></tr>
<tr><td rowspan="2">0x0188
OVER
Page 220</td><td colspan="5">Reserved</td></tr>
<tr><td colspan="2">Reserved</td><td>OVER2</td><td>OVER1</td><td>OVER0</td></tr>
<tr><td rowspan="2">0x018C
Reserved</td><td colspan="5">Reserved</td></tr>
<tr><td colspan="5">Reserved</td></tr>
<tr><td rowspan="2">0x0190
FSRF0
Page 222</td><td colspan="5">Reserved</td></tr>
<tr><td>Reserved</td><td colspan="4">FSRF0</td></tr>
<tr><td rowspan="2">0x0194
FSRF1
Page 223</td><td colspan="5">Reserved</td></tr>
<tr><td>Reserved</td><td colspan="4">FSRF1</td></tr>
<tr><td rowspan="2">0x0198
FSRC0
Page 224</td><td colspan="5">Reserved</td></tr>
<tr><td>Reserved</td><td colspan="4">FSRC0</td></tr>
<tr><td rowspan="2">0x019C
FSRC1
Page 225</td><td colspan="5">Reserved</td></tr>
<tr><td>Reserved</td><td colspan="4">FSRC1</td></tr>
<tr><td rowspan="2">0x01A0
FSRA0
Page 226</td><td colspan="5">Reserved</td></tr>
<tr><td colspan="5">FSRA0 (15-0)</td></tr>
</table>

Table 6-4. Registers (Continued)

0x01A4 FSRA1 Page 227	Reserved	
	FSRA1 (15-0)	
0x01A8 FSRDL0 Page 228	Reserved	
	FSRDL0 (15-0)	
0x01AC Reserved	Reserved	
	Reserved	
0x01B0 FSRDL1 Page 229	Reserved	
	FSRDL1 (15-0)	
0x01B4 Reserved	Reserved	
	Reserved	
0x01B8 Reserved	Reserved	
	Reserved	
0x01BC Reserved	Reserved	
	Reserved	
0x01C0 ROM Page 230	Reserved	
	Reserved	ROM
0x01C4 ALGO Page 231	ALGO3 (31-24	ALGO2 (23-16)
	ALGO1 (15-8)	ALGO0 (7-0)

Table 6-4. Registers (Continued)

0x01C8 RINFOL Page 232	RINFOL3 (31-24	RINFOL2 (23-16)
	RINFOL1 (15-8)	RINFOL0 (7-0)

0x01CC RINFOU Page 233	RINFOU3 (31-24	RINFOU2 (23-16)
	RINFOU1 (15-8)	RINFOU0 (7-0)

6.7.1 Variable Address Register (A0, A1, A2, A3)

Functionality of the register is described in Figure 6-3 and Table 6-5.

Figure 6-3. *Variable Address Register (A0, A1, A2, A3)* [offset = 0x0100 - 0x010C]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">A0</td></tr>
<tr><td colspan="16">RW-x</td></tr>
</table>

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-5. *Variable Address Register (A0, A1, A2, A3)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	A0, A1, A2, A3		Variable Address 0, 1, 2, 3

6.7.2 *Variable Loop Count Register (L0, L1, L2, L3)*

Functionality of the register is described in Figure 6-4 and Table 6-6.

Figure 6-4. ***Variable Loop Count Register (L0, L1, L2, L3)*** **[offset = 0x110 - 11C]**

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
L0															
RW-x															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-6. ***Variable Loop Count Register (L0, L1, L2, L3)***

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	L0, L1, L2, L3		Variable Loop Counter 0, 1, 2, 3

6.7.3 *DD0 Data Register (DD0)*

Functionality of the register is described in Figure 6-5 and Table 6-7.

Figure 6-5. *DD0 Data Register (DD0)* [offset = 0x0120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D1															
RW-x															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0															
RW-x															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-7. *DD0 Data Register (DD0)*

Bit	Name	Value	Description
31-16	D1		DD1 Data Register Upper 16
15-0	D0		DD0 Data Register Lower 16

6.7.4 DDE Data Register (DE0)

Functionality of the register is described in Figure 6-6 and Table 6-8.

Figure 6-6. *DE0 Data Register (DE0)* [offset = 0x0124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E1															
RW-x															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E0															
RW-x															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-8. *DE0 Data Register (DE0)*

Bit	Name	Value	Description
31-16	E1		DE1 Data Register Upper 16
15-0	E0		DE0 Data Register Lower 16

6.7.5 *Constant Address Register (CA0, CA1, CA2, CA3)*

Functionality of the register is described in Figure 6-7 and Table 6-9.

Figure 6-7. *Constant Address Register (CA0, CA1, CA2, CA3)* [offset = 0x0130 - 0x013C]

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
CA0															
RW-X															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-9. *Constant Address Register (CA0, CA1, CA2, CA3)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	CA0, CA1, CA2, CA3		Constant Address 0, 1, 2, 3

6.7.6 *Constant Loop Count Register (CL0, CL1, CL2, CL3)*

Functionality of the register is described in Figure 6-8 and Table 6-10.

Figure 6-8. *Constant Loop Count Register (CL0, CL1, CL2, CL3)* [offset = 0x140 - 0x014C]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CL0</td></tr>
<tr><td colspan="16">RW-X</td></tr>
</table>

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-10. *Constant Loop Count Register (CL0, CL1, CL2, CL3)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	CL0, CL1, CL2, CL3		Constant Loop Counter 0, 1, 2, 3

6.7.7 *Constant Increment Register (I0, I1, I2, I3)*

Functionality of the register is described in Figure 6-9 and Table 6-11.

Figure 6-9. *Constant Increment Register (I0, I1, I2, I3)* [offset = 0x150 - 0x0154]

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
I0															
RW-X															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-11. *Constant Increment Register 0 (I0)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	I0, I1, I2, I3		Constant Increment Counter 0, 1, 2, 3

6.7.8 RAM Configuration Register (RAMT)

This register is divided into following internal registers, none of which have a default value after reset. Figure 6-10 and Table 6-12 illustrate this register.

Figure 6-10. RAM Configuration Register (RAMT) [offset = 0x0160]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RGS								RDS							
RW-X								RW-X							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWR								SMS		PLS				RLS	
RW-X								RW-X		RW-X				RW-X	

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-12. *RAM Configuration Register (RAMT)*

Bit	Name	Value	Description
31-24	RGS		Ram Group Select
23-16	RDS		Return Data Select
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

This register provides the information regarding which memory is currently being tested. In case of PBIST test failure, this register can be read to find out which memory failed.

Bit fields RGS and RDS map to memory / memory bank. This is device specific. Please refer to device data sheet for the mapping of RGS, RDS to memories.

Note:
Registers A0 - A3, L0 - L3, DD0, DE0, CA0 - CA3, I0 - I3, RAMT registers are used by PBIST controller for microcode execution. These registers should not be modifed by application software, as incorrect data in these may result in PBIST failure.

6.7.9 Datalogger Register (DLR)

This register puts the PBIST controller into the appropriate comparison modes for data logging. Figure 6-11 and Table 6-13 illustrate this register.

Figure 6-11. *Datalogger Register (DLR)* [offset = 0x0164]

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
Reserved					DLR10	DLR9	DLR8	DLR7	DLR6	DLR5	DLR4	DLR3	DLR2	DLR1	DLR0
R-0					R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-13. *Datalogger Register (DLR)*

Bit	Name	Value	Description
31-11	Reserved		Reads return zeros and writes have no effect.
10	DLR10		Retention testing
9	DLR9		GO / NO-GO testing
8	DLR8		MISR testing
7	DLR7		Time Stamp
6	DLR6		Column Fail Masking
5	DLR5		Emulation cache access
4	DLR4		Config access
3	DLR3		TCK gated (Diag_clock gated)
2	DLR2		ROM-based testing
1	DLR1		IDDQ testing
0	DLR0		Distributed Compare mode

- **DLR0: Distributed Compare mode**

  In this mode, the internal PBIST datalogger is not used for data comparison. The expected data is always driven out on the write data buses during reads. This expected data is XOR-ed with the return data locally at each individual memory. With distributed comparing enabled, PBIST expects a 1 or 0 in the return data to indicate whether that compare passed (1) or failed (0). The expected data is always all 1s, and this is compared with results in the return data. Also in this mode, each bit in DWR indicates a separate memory that is part of the parallel test.

- **DLR1: IDDQ testing mode**

  This mode is used to enable IDDQ testing of the memories. With this mode enabled, the new WRITE_IDDQ A0/A1 instruction can be used to write back inverted read data from the memories. Data comparison is disabled in this mode and the FSRD register stores the return data. This implies the test will never fail. For config/VLCT based testing, this DLR bit should be written explicitly to enable this mode. For ROM-based testing, if bit [23] of the algorithm header were set, it would automatically enable this mode only for that particular algorithm.

- **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.

- **DLR3: TCK Gated mode**

  This mode is identical to the existing one. When this bit is asserted, the pbist_done and pbist_fail signals are driven to the tester directly from inside the datalogger instead of the VLCT interface block. This mode exists because the tester clocks might not always be running when PBIST is done with its memory testing. This mode refers to diag_clock and not tck.

- **DLR4: Config access mode**

  This mode, when set, indicates the CPU is being used to access PBIST. After reset, this bit is 0, which defaults to the VLCT interface.

- **DLR5: Emulation cache access mode**

  In this mode, all failure checking is turned off. PBIST is set in single-step mode automatically and it waits for reads from Config of the FSRDL0/1 register. Every time a read occurs, PBIST fetches the corresponding data from the address locations, and sends it back to CPU on the pbist_to_cfgbi_rdata bus.

- **DLR6: Column Fail Masking mode**

  In this mode, the register stores the cumulative bit fails and masks them from the next failure detection. This allows detection of columns failing without multiple failures for the same column triggering unnecessary failures.

- **DLR7: Time stamp mode**

  PBIST supports a unique timestamping methodology. In any testing mode, the PBIST controller stops when a failure occurs. If the testing operation is restarted, the back-to-back testing may have been interrupted and may hide failures. To avoid this problem, use the time stamp mode. In this mode, PBIST saves the time stamp of the last failure. When you restart, the PBIST controller starts from the top and skips error-checking until the point of the first failure. It then enables data logging, and testing continues. At the end of the test, you will have detected all failures only once.

- **DLR8: MISR testing mode (ROM testing)**

  If MISR mode is set, then failure checking is turned off internally, and all read data is fed into the MISRs to generate a signature. This mode should be set when testing chip-level ROMs. It can also be used generally with other memory testing algorithms. The final MISR signature is compared with D1:D0, which holds the expected signature. A pass or fail is then asserted accordingly. For ROM-based testing, the information in the algorithm header would automatically enable this mode for the ROM test algorithms.

- **DLR9: GO / NO-GO testing mode**

  This is the default mode of testing. This bit in DLR must be set to 0 before performing testing in any other mode. When in this mode, ROM-based testing is kicked off as soon as the tmode_pbist_ext is asserted. If the intention is to perform go/no-go testing via config, write to both this bit and bit [2] of the DLR register simultaneously.

- **DLR10: Retention testing mode**

  After an algorithm is executed while in retention mode, the retention signal is asserted and PBIST waits for pbist_resume signal to be asserted before continuing. This mode makes the most sense with ROM-based testing although it can also be used with the config and VLCT interfaces. For ROM-based testing, if bit [24] of the algorithm header were set, it would automatically enable this mode only for that particular algorithm.

Most of these modes can be set in conjunction with other DLR modes.

6.7.10 Clock-Mux Select Register (CMS)

This 4-bit clock select register selects between 4 clock inputs. Writing to this register must be done carefully. Default value of the register is 0000, which selects clock1. Functionality of the register is described in Figure 6-12 and Table 6-14.

Figure 6-12. *Clock-Mux Select Register (CMS)* [offset = 0x0168]

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
Reserved												CMS			
R-0												R/W-0000			

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-14. *Clock-Mux Select Register (CMS)*

Bit	Name	Value	Description
31-4	Reserved		Reads return zeros and writes have no effect.
3-0	CMS		Selects the clock input
		0000	Clock1/main_clk/pbist_clk1
		0001	Clock2/cfg-clk/pbist_clk2
		0010	Clock3/pbist_clk3
		0011	Clock4/rom_clk/pbist_clk4
		0100	External clk
		1000	External clk

Note:
For ROM based testing, the appropriate clock is selected for each memory, based on the RAM group information. The application software should not modify this register.

6.7.11 Program Control Register (STR)

This register starts the PBIST controller in appropriate modes. Figure 6-13 and Table 6-15 illustrate this register.

Figure 6-13. *Program Control Register (STR)* [offset = 0x016C]

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
Reserved											STR				
R-0											R/W-00000				

R = Read in all modes; W = Write in all modes; *-n* = Value after reset

Table 6-15. *Program Control Register (STR)*

Bit	Name	Value	Description
31-5	Reserved		Reads return zeros and writes have no effect.
4-0	STR		PBIST Controller Mode
			Starts the PBIST controller in appropriate mode
		00001	Start / Time Stamp mode restart
		00010	Resume / Emulation read
		00100	Stop
		01000	Step / Step for emulation mode
		10000	Check MISR mode

Note:
The above bits are mutually exclusive. Incorrect software programming might give unexpected results.

6.7.12 Chip Select Register (CSR)

PBIST implements a chip select of up to 32 RAMs. When a READ or WRITE command is executed, the corresponding Chip Select is driven active and enables that RAM. This is 32-bit chip select register internally divided into four configuration registers to support doing a MVI to this register from within an algorithm, 8 bits at a time. CSR is normally the same as the 32-bit decoded value of the 5-bit RDS. For a memory to be selected, the corresponding CSR bit pointed to by the RDS register should be set to 1. More than 1 CSR could be turned on at the same time to excite more than one memory simultaneously, but the return data will always be selected from one memory pointed to by the unique RDS. Figure 6-14 and Table 6-16 illustrate this register.

Figure 6-14. Chip Select Register (CSR) [offset = 0x0178]

31-24	23-16
CSR3	CSR2
R/W-0	R/W-0

15-8	7-0
CSR1	CSR0
R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-16. *Chip Select Register (CSR)*

Bit	Name	Value	Description
31-24	CSR3		Chip Select 3
23-16	CSR2		Chip Select 2
15-8	CSR1		Chip Select 1
7-0	CSR0		Chip Select 0

Note:
This register should not be modifed by application software, as incorrect data in these may result in PBIST failure.

6.7.13 *Fail Delay Register (FDLY)*

This register is used with external ATE interface. When failure occurs, instead of immediately asserting the pbist_fail signal, the controller waits for the delay programmed through this register. The default value of the register is 01001000 (72 cycles). PBIST waits for the delay specified in this register before processing a new failure. This delay might be necessary as tester clocks are not fast enough and delay is required to capture the fail signal and logout data. Figure 6-15 and Table 6-17 illustrate this register.

Figure 6-15. *Fail Delay Register (FDLY)* [offset = 0x017C]

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
Reserved								FDLY							
R-0								R/W-01001000							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-17. *Fail Delay Register (FDLY)*

Bit	Name	Value	Description
31-8	Reserved		Reads return zeros and writes have no effect.
7-0	FDLY		Delay in asserting the FAIL signal

Note:
This register is not required during application self test.

6.7.14 PBIST Activate/ROM 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. A value of '1' in bit [0] would select the clock pointed to by the CMS register. Bit [1] is for turning on the clock going to the instruction ROM.

Figure 6-16. *PBIST Activate/ROM Clock Enable Register (PACT)* [offset = 0x0180]

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
Reserved														PACT1	PACT0
R-0														R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-18. *PBIST Activate/ROM Clock Enable Register (PACT)*

Bit	Name	Value	Description
31-2	Reserved		Reads return zeros and writes have no effect.
1	PACT1		PBIST Activate
		0	Clock to on chip ROM is disabled
		1	Normal PBIST operation for ROM based testing
0	PACT0		ROM Clock Enable Register
		0	Disable internal PBIST clocks
		1	Enable internal PBIST clocks

- **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.

- **PACT1**

  Setting this bit turns on the clock going to the instruction ROM.

Note:
This register must be programmed to 0x11, during application self test.

6.7.15 PBIST ID Register

Functionality of the register is described in Figure 6-17 and Table 6-19.

Figure 6-17. *PBIST ID Register* [offset = 0x184]

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
Reserved								PBIST ID							
R-0								RW-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-19. *PBIST ID Register*

Bit	Name	Value	Description
31-8	Reserved		Reads return zeros and writes have no effect.
7-0	PBIST ID		This is a unique ID assigned to each PBIST controller in a device with multiple PBIST controllers.

Note:
The PBIST ID should not be changed from its default value 0x00.

6.7.16 Override Register (OVER)

Functionality of the register is described in Figure 6-18 and Table 6-20.

Figure 6-18. *Override Register (OVER)* [offset = 0x0188]

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
Reserved													OVER2	OVER1	OVER0
R-0													R/W-0	R/W-0	R/W-1

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-20. *Override Register (OVER)*

Bit	Name	Value	Description
31-3	Reserved		Reads return zeros and writes have no effect.
2	OVER2		Multiple Memory Enable
		0	Disables multiple memories during testing for noise-injection/IR drop.
		1	Enables multiple memories during testing for noise-injection/IR drop.
1	OVER1		READ Override Bit
		0	READ commands are executed normally.
		1	READ commands are internally converted to READ Without Compares.
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 will override the RAM selection from the RAM info registers RINFOL and RINFOU.

- **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, user can program the RAM info groups to selectively run each algorithm on predefined memory groups.

Note:
When this override bit = 0 each algorithm selected in ALGO register will run on each RAM selected in RINFOL and RINFOU register. It must be ensured that: (1) only same type of RAMs are selected (Single Port or Dual Port). AND (2) Only RAMs that are valid for ALL algoriothms are selected. Otherwise the PBIST test will fail.

- **OVER1**

  This bit is the READ Override bit. By default, this bit is 0, which means READ commands are executed. When this bit is set to 1, all READs are internally converted to READ Without Compares. This bit should not be set during application self test.
- **OVER2**

  If this bit is set to 1, it enables multiple memories during testing for noise-injection/IR drop. The default value is 0. When this bit is set to 1, all bits of CSR go high. This way, data will be read by all memories within the same RGS. This bit should not be set during application self test.

6.7.17 Fail Status Fail Register 0 (FSRF0)

This register indicate if a failure occurred during a PBIST test. This register is set whenever a failure occurs. FSRF0 indicates Port 0 failures. Figure 6-19 and Table 6-21 illustrate this register.

Figure 6-19. *Fail Status Fail Register 0 (FSRF0)* [offset = 0x0190]

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
Reserved															FSRF0
R-0															R-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-21. *Fail Status Fail Register 0 (FSRF0)*

Bit	Name	Value	Description
31-1	Reserved		Reads return zeros and 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	Indicates a failure on Port 0

6.7.18 Fail Status Fail Register 1 (FSRF1)

This register indicate if a failure occurred during a PBIST test. This register is set whenever a failure occurs. FSRF1 indicates Port 1 failures. Figure 6-20 and Table 6-22 illustrate this register.

Figure 6-20. *Fail Status Fail Register 1 (FSRF1)* [offset = 0x0194]

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
Reserved															FSRF1
R-0															R-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-22. *Fail Status Fail Register 1 (FSRF1)*

Bit	Name	Value	Description
31-1	Reserved		Reads return zeros and writes have no effect.
0	FSRF1		Fail Status 1. This bit would be cleared by reset of the module using MSTGCR register in system module.
		0	No Failure Occurred
		1	Indicates a failure on Port 1

6.7.19 Fail Status Count Register 0 (FSRC0)

This register keeps count of the number of failures. Whenever a failure occurs, the value gets incremented by one and when the failure is processed, the value is decremented by one. FSRC0 is for Port 0. Figure 6-21 and Table 6-23 illustrate this register.

Figure 6-21. *Fail Status Count 0 Register (FSRC0)* [offset = 0x0198]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">FSRC0</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R-0</td></tr>
</table>

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-23. *Fail Status Count 0 Register (FSRC0)*

Bit	Name	Value	Description
31-8	Reserved		Reads return zeros and writes have no effect.
7-0	FSRC0		Fail Status Count 0. Indicates the number of failures on port 0.

6.7.20 *Fail Status Count Register 1 (FSRC1)*

This register keeps count of the number of failures. Whenever a failure occurs, the value gets incremented by one and when the failure is processed, the value is decremented by one. FSRC1 is for Port 1. Figure 6-22 and Table 6-24 illustrate this register.

Figure 6-22. *Fail Status Count Register 1 (FSRC1)* [offset = 0x019C]

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
Reserved								FSRC1							
R-0								R-0							

R = Read in all modes; W = Write in all modes; *-n* = Value after reset

Table 6-24. *Fail Status Count Register 1 (FSRC1)*

Bit	Name	Value	Description
31-8	Reserved		Reads return zeros and writes have no effect.
7-0	FSRC1		Fail Status Count 1. Indicates the number of failures on port 1.

6.7.21 Fail Status Address 0 Register (FSRA0)

This register captures the memory address of the first failure on port 0. Figure 6-23 and Table 6-25 illustrate this register.

Figure 6-23. *Fail Status Address 0 Register (FSRA0)* [offset = 0x01A0]

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
FSRA0															
R-0															

R = Read in all modes; W = Write in all modes; *-n* = Value after reset

Table 6-25. *Fail Status Address 0 Register (FSRA0)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	FSRA0		Fail Status Address 0. Contains the address of the first failure.

6.7.22 *Fail Status Address 1 Register (FSRA1)*

This register captures the memory address of the first failure on port 1. Figure 6-24 and Table 6-26 illustrate this register.

Figure 6-24. *Fail Status Address 1 Register (FSRA1)* [offset = 0x01A4]

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
FSRA1															
R-0															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-26. *Fail Status Address 1 Register (FSRA1)*

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	FSRA1		Fail Status Address 1. Contains the address of the first failure.

6.7.23 *Fail Status Data Register 0 (FSRDL0)*

This internal register is used to capture the failure data in case of failures. Also, depending on the PBIST controller mode, this register also hold the return data from the memories (IDDQ and EMU modes) or the MISR signature (MISR and CMISR modes). FSRDL0 corresponds to Port 0. Figure 6-25 and Table 6-27 illustrate this register.

Figure 6-25. *Fail Status Data Register 0 (FSRDL0)* [offset = 0x01A8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSRDL0[31:16]															
R-0xAAAA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSRDL0[15:0]															
R-0xAAAA															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-27. *Fail Status Data Register 0 (FSRDL0)*

Bit	Name	Value	Description
31-0	FSRDL0		Failure data on port 0

6.7.24 *Fail Status Data Register 1 (FSRDL1)*

This internal register is used to capture the failure data in case of failures. Also, depending on the pBIST controller mode, this register also hold the return data from the memories (IDDQ and EMU modes) or the MISR signature (MISR and CMISR modes). FSRDL1 corresponds to Port 1. Figure 6-26 and Table 6-28 illustrate this register.

Figure 6-26. ***Fail Status Data Register 1 (FSRDL1)*** **[offset = 0x01B0]**

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">FSRDL1[31:16]</td></tr>
<tr><td colspan="16">R-0xAAAA</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">FSRDL1[15:0]</td></tr>
<tr><td colspan="16">R-0xAAAA</td></tr>
</table>

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-28. ***Fail Status Data Register 1 (FSRDL1)***

Bit	Name	Value	Description
31-0	FSRDL1		Failure data on port 1

6.7.25 ROM Mask Register (ROM)

This two-bit register sets appropriate ROM access modes for the PBIST controller. The default value is 0x11. It can be programmed according to Table 6-29.

Figure 6-27. *ROM Mask Register (ROM)* [offset = 0x01C0]

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
Reserved														ROM	
R-0														R/W-0x11	

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-29. *ROM Mask Register (ROM)*

Bit	Name	Value	Description
31-2	Reserved		Reads return zeros and writes have no effect.
1-0	ROM		ROM Mask
		00	No information is used from ROM
		01	Only RAM Group information from ROM
		10	Only Algorithm information from ROM
		11	Both Algorithm and RAM information from ROM. This option should be selected for application self test.

6.7.26 ROM Algorithm Mask Register (ALGO)

Figure 6-28 and Table 6-30 illustrate this register.

Figure 6-28. *Algorithm Mask Register (ALGO)* [offset = 0x01C4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">ALGO3 (31-24)</td><td colspan="8">ALGO2 (23-16)</td></tr>
<tr><td colspan="8">RW-0xFF</td><td colspan="8">RW-0xFF</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">ALGO1 (15-8)</td><td colspan="8">ALGO0 (7-0)</td></tr>
<tr><td colspan="8">RW-0xFF</td><td colspan="8">RW-0xFF</td></tr>
</table>

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-30. *Algorithm Mask Register (ALGO)*

Bit	Name	Value	Description
31		0	Algorithm 32 is not selected
		1	Selects algorithm 32 for PBIST run
30		0	Algorithm 31 is not selected
		1	Selects algorithm 31 for PBIST run
...			
0		0	Algorithm 1 is not selected
		1	Selects algorithm 1for PBIST run
[31:0]		0x0000	No algorithms are selected

Note:
This algorithm mapping is device specific. Please refer to the device datasheet for list and bit mapping for available algorithms.

6.7.27 RAM Info Mask Lower Register (RINFOL)

This register is 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 should be set to ‘1’. The default value of this register is all ‘1’s, which means all the RAM Info Groups would be selected. Figure 6-29 and Table 6-31 illustrate this register.

The information from this register is used only when bit 0 in OVER register is not set.

Figure 6-29. *RAM Info Mask Lower Register (RINFOL)* [offset = 0x01C8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RINFOL3 (31-24)								RINFOL2 (23-16)							
RW-0xFF								RW-0xFF							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RINFOL1 (15-8)								RINFOL0 (7-0)							
RW-0xFF								RW-0xFF							

R = Read in all modes; W = Write in all modes; WP = Write in privilege mode only; WC = Write in sci-compatible mode only; *-n* = Value after reset

Table 6-31. *RAM Info Mask Lower Register (RINFOL) (RINFOL)*

Bit	Name	Value	Description
31		0	RAM Group 32 is not selected
		1	Selects group 32 for PBIST run
30		0	RAM Group 31 is not selected
		1	Selects RAM group 31 for PBIST run
...			
0		0	RAM Group 1 is not selected
		1	Selects RAM Group 1for PBIST run
[31:0]		0x0000	None of the RAM Groups 1 to 32 are selected

Note:
This RAM info group mapping is device specific. Please refer to the device datasheet for list and bit mapping for RAM info groups and valid algorithms.

6.7.28 RAM Info Mask Upper Register (RINFOU)

This register is 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 should be set to ‘1’. The default value of this register is all ‘1’s, which means all the RAM Info Groups would be selected. Figure 6-30 and Table 6-32 illustrate this register.

Figure 6-30. *RAM Info Mask Upper Register (RINFOU)* [offset = 0x01CC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RINFOU3 (31-24)								RINFOU2 (23-16)							
R/W-0xFF								R/W-0xFF							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RINFOU1 (15-8)								RINFOU0 (7-0)							
R/W-0xFF								R/W-0xFF							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 6-32. *RAM Info Mask Upper Register (RINFOU)*

Bit	Name	Value	Description
31		0	RAM Group 64 is not selected
		1	Selects group 64 for PBIST run
30		0	RAM Group 63 is not selected
		1	Selects RAM group 63 for PBIST run
...			
0		0	RAM Group 33 is not selected
		1	Selects RAM Group 33 for PBIST run
[31:0]		0x0000	None of RAM Groups 33 to 64 are selected

6.8 PBIST Configuration Example

The following examples assumes that the PLL is locked and selected as clock source with HCLK = 160MHz and VCLK = 80MHz.

6.8.1 Example1 : PBIST Configuration for DCAN1 RAM with No RAM Override

The example explains the configurations for running March13, Down1A and Mapcolumn algorithm on DCAN1

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module
   MSTGCR[9:8] = 1
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 32 VCLK cycles
5. Enable the Pbist internal clocks and ROM interface clock.
   PACT = 0x3
6. Disable RAM Override
   OVER = 0x0;
7. Select the Algorithm (Refer Table 6-2)
   ALGO = 0x00000054 (Algo 3,5,7 for DCAN1)
8. Program the RAM group Info to select DCAN1 and MIBSPI (Refer Table 6-1)
   RINFOL = 0x00000004 (Select RAM Group 3)
   RINFOU = 0x00000000 (since this device supports only 15 RAM Group)
9. Select both Algorithm and RAM information from on chip PBIST ROM
   ROM = 0x3
10. Configure PBIST to run in ROM Mode and Kickoff PBIST test
    DLR = 0x14
11. Wait for PBIST test to complete by polling MSTDONE bit in System Moule.
    while(MSTDONE !=1);
12. Once Self test is completed, check the Fail Status registers FSRF0 and FSRF1.
    In case there is a failure (FSRF0 or FSRF1=0x01)

    i Read RAMT register which indicates the RGS and RDS values of the failure RAM

    ii Read FSRC0 and FSRC1 registers which contains the failure count

    iii Read FSRA0 and FSRA1 registers which contains the address of first failure

    iv Read FSRDL0 and FSRDL1 registers which contains the failure data.

    v .Resume the Test if required using Program Control register STR = 2

    In case there is no failure (FSRF0 and FSRF1=0x00) the Memory self test is completed.

    i) Disable the PBIST internal and ROM clocks
    PACT = 0.

    ii) Disable the PBIST self test
    MSTGCR[3:0] = 0x5

6.8.2 Example2 : PBIST Configuration for ALL RAM Groups with RAM Override

The example explains the configurations for running March13, Down1A and Mapcolumn algorithm on all RAM groups.

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module
   MSTGCR[9:8] = 1
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 32 VCLK cycles
5. Enable the Pbist internal clocks and ROM interface clock.
   PACT = 0x3
6. Enable RAM Override
   OVER = 0x1;
7. Select the Algorithm (Refer Table 6-2)
   ALGO = 0x000000FC
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 Moule.
    while(MSTDONE !=1);
11. Once Self test is completed, check the Fail Status registers FSRF0 and FSRF1.
    In case there is a failure (FSRF0 or FSRF1= 0x01)

    i Read RAMT register which indicates the RGS and RDS values of the failure RAM

    ii Read FSRC0 and FSRC1 registers which contains the failure count

    iii Read FSRA0 and FSRA1 registers which contains the address of first failure

    iv Read FSRDL0 and FSRDL1 registers which contains the failure data.

    v Resume the Test if required using Program Control register STR = 2

    In case there is no failure (FSRF0 and FSRF1 = 0x00) the Memory self test is completed.

    i) Disable the PBIST internal and ROM clocks
       PACT = 0.

    ii) Disable the PBIST self test
        MSTGCR[3:0] = 0x5

Phase-Locked Loop (PLL) Clock Modules

This device contains two separate Phase-Locked Loop (PLL) clock modules, the Frequency Modulated zero-pin PLL (FMzPLL) and the Flexray PLL (FPLL). The FMzPLL is available as Clock Source 1 and is intended to be used as the device's main clock source. The FPLL is available as Clock Source 6 and is intended to be used as the Flexray clock source. This chapter describes the functionality of both the Frequency Modulated zero-pin Phase-Locked Loop (FMzPLL) and the Flexray Phased-Locked Loop (FPLL).

Topic **Page**

7.1 Device Clock Overview

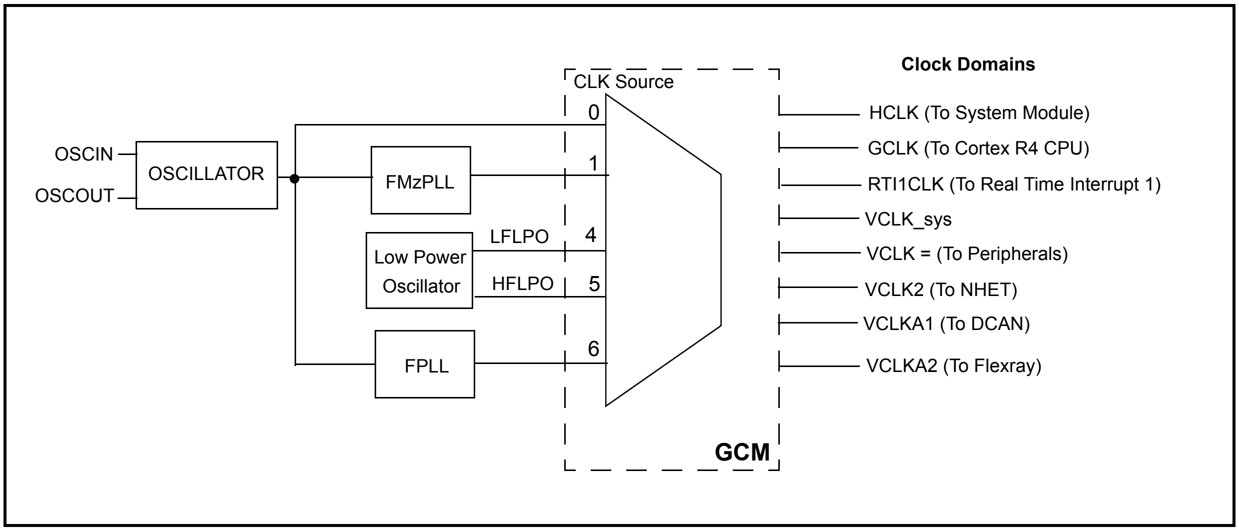
The TMS570 platform architecture supports a total of seven separate clock sources. This device implements five of the possible seven clock sources including the oscillator, the FMzPLL, the Low Frequency Low Power Oscillator (LFLPO), the High Frequency Low Power Oscillator (HFLPO) and the FPLL. The clock source implementation is shown in the Clock Sources Table below.

Table 7-1. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	External Oscillator
Clock Source1	PLL1 (FMzPLL)
Clock Source 2	Not Implemented
Clock Source 3	Not Implemented
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High Frequency LPO (Low Power Oscillator) clock
Clock Source 6	PLL2 (FPLL)
Clock Source 7	Not Implemented

The Global Clock Module (GCM) is used to configure and provide the clocks generated by the PLLs (and other clock sources) to the different modules in the device. A block diagram of the global clock module that is implemented on this device is shown below. The GCM can be configured by using the Primary System Control Registers located between addresses 0xFFFF FF30 and 0xFFFF FF54. See the System and Peripheral Control Registers Chapter for more information.

Figure 7-1. Global Clock Module Block Diagram



The Oscillator serves as the main input to clock source 0, the FMzPLL and the FPLL. It is comprised of the two external device pins OSCIN and OSCOUT. In most applications a crystal or resonator (between 5MHz and 15MHz) is placed between these two pins. The oscillator is provides the necessary feedback to the external crystal or resonator for oscillation and also converts it’s sinusoidal input wave into a square wave before it is supplied to the rest of the device.

Note: Vendor Validation of Crystals/Resonators

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 dvice for optimum start-up and operation over temperature and voltage extremes. The vendor also factors in margins for variations in the microcontroller process.

7.2 FMzPLL Introduction/Feature Overview

This section provides an overview of the Frequency Modulated zero-pin Phase-Locked Loop (FMzPLL) module. The FMzPLL is used to multiply the input frequency to a higher (device operation) frequency than can be conveniently achieved with an external crystal or resonator. Additionally, the FMzPLL allows the flexibility to be able to generate many different frequency options from a given crystal or resonator.

Frequency modulation can be superimposed on the FMzPLL output 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 EMI or noise sensitive applications.

7.2.1 Features

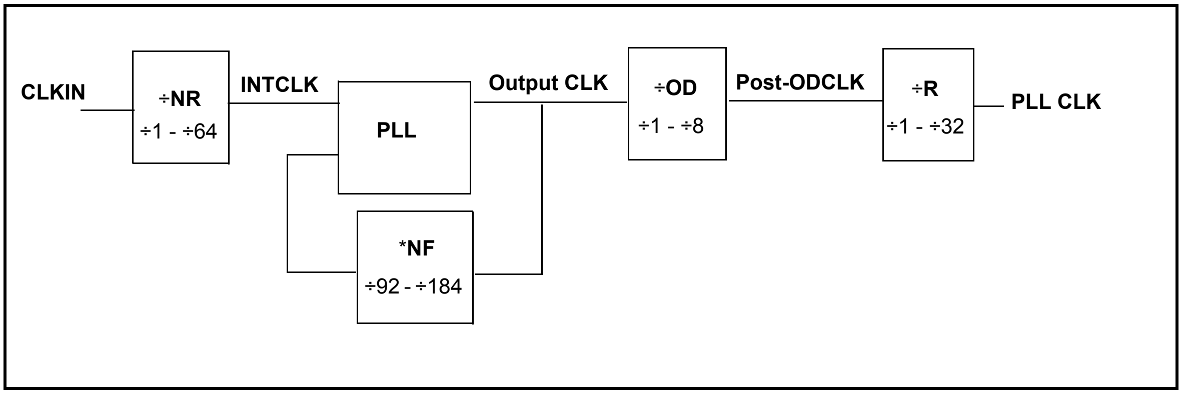
The main features of the FMzPLL clock module are:

- The FMzPLL module can be operated in either modulation or non-modulation mode.
- The FMzPLL prescale allows a wide range of input frequencies for proper operation.
- The phase-frequency detector assures lock to the fundamental reference frequency.
- The FMzPLL provides multiple frequency configuration options.
- When the FMzPLL is used with modulation enabled, the modulation frequency, modulation depth and bandwidth settings are programmable.
- It provides a user-option to reset the device if the external resonator or crystal fails.
- It provides a user-option to either reset the device or bypass the FMzPLL if a PLL Slip is detected.
- A PLL calculator tool (F035 FMzPLL Calculator) is available to assist the user in FMzPLL setup

7.3 FMzPLL Operation

The FMzPLL clock module generates the PLL clock from an external external resonator/crystal reference (CLKIN). The oscillator circuit drives an external crystal/resonator, and the FMzPLL divides (NR), the reference input for a lower frequency input into the PLL (INTCLK); though the input divider has a range from 1 to 64 (integer) INTCLK has a valid range of 1.63MHz to 6.53MHz. The FMzPLL multiplies (NF) this internal frequency by 92 - 184 with a valid range on Output CLK of 120MHz to 500MHz. The FMzPLL 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. Optionally, the frequency can be modulated (controlled jitter is introduced). See section 7.3.1.4 for more information about frequency modulation.

Figure 7-2. FMzPLL Block Diagram



Note:
ODPLL must be changed before enabling the FMzPLL

The programming algorithm for the FMzPLL is shown below:

1. Choose the Output CLK frequency as integer divider of output frequency as close to 240MHz as possible. The Output CLK frequency should not exceed 500MHz or fall below 120MHz.
2. Choose the multiplication factor as close to 120 as possible. Multiplier (NF) may not exceed 184 or fall below 92.
3. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum device frequency.
4. Select the output divider R to further reduce the FMzPLL output frequency provided to the device.

Note: F035 FMzPLL Calculator
The F035 FMzPLL Calculator is also available to assist in configuring the FMzPLL

There will be some delay before changes to the FMzPLL settings will take effect. It is best to disable the FMzPLL prior to changing the settings. All delays shown below are valid once the FMzPLL is enabled. If the FMzPLL is already enabled when the control registers are written, there may be additional delay cycles.

- When NR or NF is changed (or the input clock is enabled), the FMzPLL output is held static for 4096 internal clock cycles (t = 4096*NR/f_{OSCIN})

In general, the smallest delay is achieved when the registers are configured prior to enabling the FMzPLL; when registers are written while the FMzPLL is active, some delays that may be concurrent are treated as sequential. As a specific example, when all parameters are changed once at the beginning of the code, the

FMzPLL output is held static for about 4650*NR/f_{OSCIN}. More information about FMzPLL setup can be found in the FMzPLL Configuration Example.

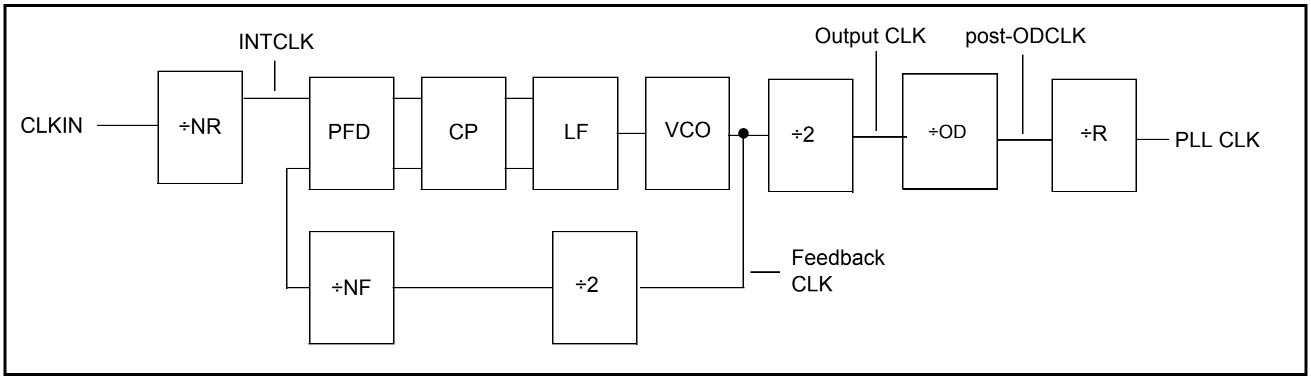
7.3.1 *Phase-Locked Loop (PLL) Description*

The basic PLL block consists of the following six logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)
- Frequency Modulation
- Slip Detector

Figure 7-3 further illustrates the sub-blocks of the FMzPLL blocks shown in Figure 7-2. 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 greater than the internal frequency (OSCIN/NR). In the forward path (from VCO to PLL CLK), the ÷2 block creates a clean duty cycle.

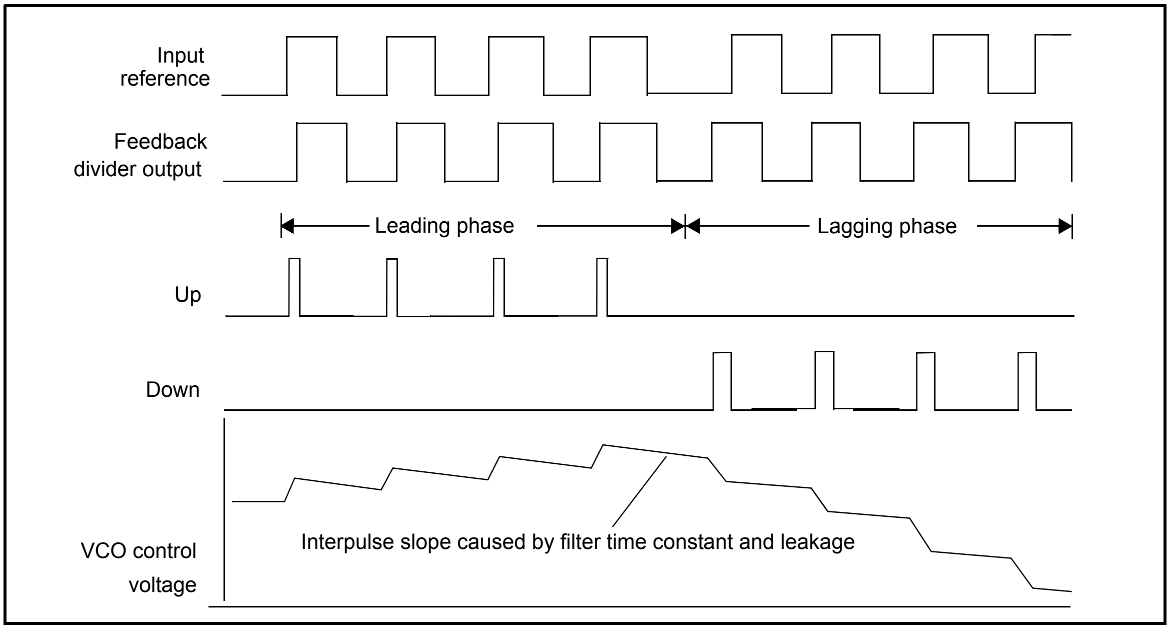
Figure 7-3. Basic PLL Circuit



7.3.1.1 *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 7-4.

Figure 7-4. 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 as shown in Figure 7-4. 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 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 bitfield 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 (e.g. 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.

7.3.1.2 Charge Pump and Loop Filter

The charge pump (CP) adds or removes 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.

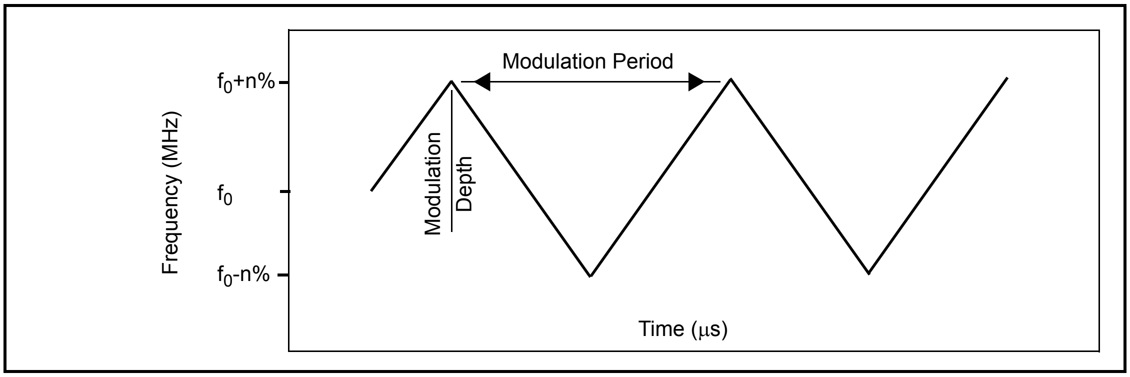
7.3.1.3 Voltage-Controlled Oscillator

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.

7.3.1.4 Frequency Modulation

When the FMzPLL is used in the modulating mode the output clock of the FMzPLL changes frequency in a controlled way, centered around the unmodulated output frequency. The VCO frequency is modulated at the loop filter and creates the triangular frequency modulation as shown in Figure 7-5.

Figure 7-5. Frequency vs. Time



> **Note: Modulation Frequency and Depth Setting Constraint**
> There are several combinations of the modulation depth and modulation frequency that are not allowed. Some of these settings effect the FMzPLL even when frequency modulation is not enabled. Refer to the device datasheet to identify these combinations to avoid FMzPLL malfunction.

The programming algorithm for the frequency modulation settings for the FMzPLL is shown below: These settings are controlled by the bits in the PLLCTL2 register.

1. Determine modulation frequency divider NS based on desired spreading (modulation) frequency f_{mod} (f_S).

$$NS = \mathrm{round}\left(\frac{f_{OSCIN}}{NR}\frac{1}{2 \times f_S}\right)$$

$$f_{mod} = f_S = \frac{f_{OSCIN}}{NR}\frac{1}{2 \times NS}$$

2. Determine modulation depth divider (NV) based on desired Modulation Depth.

$$NV = \mathrm{round}\left(5.02154 \times 10^7 \times \frac{Depth \times f_S}{f_{OutputCLK}}\right)$$

3. Determine the bandwidth divider NB

$$NB = \max(\mathrm{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{f_{OutputCLK}\frac{f_{OSCIN}}{NR}}}{f_S}\right),8)$$

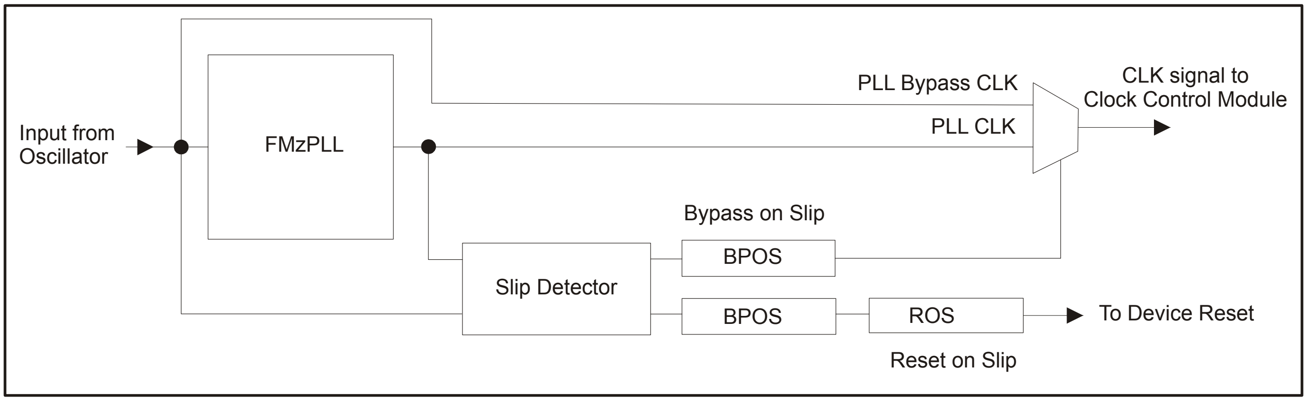
- When NB or NS is changed, the FMzPLL output is held static for 512 internal clock cycles (t = 512*NR/ f_{OSCIN}). This timing item cannot be carried out concurrently with a PLL re-lock since the re-lock occurs with NB forced to the default value for faster locking.
- When NV is changed or modulation enabled, the FMzPLL output is held static for 1 - 2 modulation period (t = $1/f_S$~ $2/f_S$)

As these different fields (NV, NF, NB, etc.) are changed, the FMzPLL asserts different delays in order to insure a valid output. It is recommended that all pertinent parameters be setup at one time in order to minimize system delays. When multiple parameters are changed simultaneously, it is possible for all delays to be concurrent. As a specific example, when all parameters are changed once at the beginning of the code and modulation is enabled, the FMzPLL output is held static for about 4650*NR/f_{OSCIN} + 1 modulation period. More information about FMzPLL frequency modulation setup can be found in the FMzPLL Configuration Example.

7.3.1.5 FMzPLL Slip Detector

The FMzPLL Slip Detector monitors the input and output of the FMzPLL and reports any 2-cycle slips. Upon a slip detection three actions can be taken: nothing, device reset, or the FMzPLL can be bypassed so that the device is supplied with the oscillator input. The behavior of the device after a PLL slip is detected is configured by the Reset on Slip (ROS) and Bypass on Slip (BPOS) bits in the PLLCTL1 register. The Reset on Slip bit setting determines whether a reset is asserted after a PLL slip is detected. The Bypass on Slip bits determine whether the device will automatically bypass the FMzPLL and use the oscillator to provide the device clock after a PLL slip is detected. See the PLLCTL1 register for more details on the configuration of the ROS and BPOS bits. Figure 7-6 below shows a block diagram of the PLL Slip Detector and the reset/ bypass logic.

Figure 7-6. PLL Slip Detection & Reset/Bypass Block Diagram



7.3.1.6 Oscillator Fail Detector

The Clock Monitor Module (CMM) implements circuitry that monitors the oscillator input frequency to ensure that it remains within a specified operational range. (This range is specified in the LPO and Clock Detection section of the device datasheet.) If the frequency of the oscillator ever falls out of the specified frequency window, the clock monitor can either reset the device or switch the device's Clock Source 1 to limp mode. In limp mode all modules driven by Clock Source 1 are switched from the FMzPLL output to be driven by the internal High Frequency LPO (Low Power Oscillator) output. The behavior of the device after an oscillator failure is configured by the Reset on Oscillator Fail (ROF) bit in the PLLCTL1 register. If the ROF bit in the PLLCTL1 register is set when the oscillator fails, a system reset will occur, and the OSC_RST history bit is set in the SYSESR register. The only way OSCFAIL can be cleared (and to re-enable OSCIN as the clock source) is via a power-on-reset.

7.4 FMzPLL Control Registers

7.4.1 Control registers

The FMzPLL clock module has two registers (PLLCTL1 and PLLCTL2) located within the system module, plus it has four bits located in other global control registers of the system module.

The FMzPLL is off at power-on. It may be turned on by clearing the CLK_SR1_OFF bit in the CSDIS register of the System module.

The following sections describe the two registers used to configure the FMzPLL (Clock Source 1). These registers support 8, 16, and 32-bit write accesses.

Figure 7-7. Module Registers

Offset Address† Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x70 PLLCTL1 Page 248	ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
	PLLMUL[15:0]															
0x74 PLLCTL2 Page 250	FMENA	SPREADINGRATE[8:0]									Reserved	BWADJ[8:4]				
	BWADJ[3:0]				ODPLL[2:0]			SPR_AMOUNT[8:0]								

7.4.1.1 PLL Control 1 Register (PLLCTL1)

Figure 7-8 illustrates this register and Table 7-2 provides the bit descriptions.

Figure 7-8. PLL Control 1 Register (PLLCTL1)[Location = 0xFFFF FF70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROS	BPOS[1:0]		PLLDIV[4:0]					ROF	Reserved	REFCLKDIV[5:0]					
R/WP-0	R/WP-01		R/WP-1111					R/WP-0	R-0	R/WP-000010					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLMUL[15:0]															
R/WP–0101111100000000															

R = Read, W = Write; P = Privilege Mode, -*n* = Value after reset

Table 7-2. PLL Control 1 Register (PLLCTL1) Field Descriptions

Bit	Name	Value	Description
31	Reset on PLL Slip (ROS)		
		0	Do not reset system when slip is detected
		1	Reset when slip is detected
			Note: BPOS (PLLCTL1.29:30) must be enabled in order to use Reset On Slip (ROS) functionality
30-29	Bypass of PLL Slip(BPOS)		
		10	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		other	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 (PLLCTL1.31) is set to 1, the device will be reset if a PLL slip is detected and BPOS will not bypass the PLL.
28-24	PLL Output Clock Divider (PLLDIV)		R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post\_ODCLK}/R$ This divider is depicted as '÷R' in the FMzPLL Block Diagram.
		0x00	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/1$
		0x01	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/2$
		:	continues in sequence
		0x1F	$f_{PLL\ CLK} = f_{post\text{-}ODCLK}/32$
			Note: This divider is outside of the PLL macro and can be changed at any time without requiring a re-lock.

Table 7-2. PLL Control 1 Register (PLLCTL1) Field Descriptions (Continued)

Bit	Name	Value	Description
23	Reset On Oscillator Fail (ROF)		
		0	Do not reset system when oscillator is out of range.
		1	Reset when oscillator is out of range.
22	Reserved		Write 0.
21-16	Reference Clock Divider (REFCLKDIV)		NR = REFCLKDIV + 1 $f_{INT\ CLK} = f_{OSCIN}/NR$ This divider is depicted as '÷NR' in the FMzPLL Block Diagram.
		0x00	$f_{INT\ CLK} = f_{OSCIN}/1$
		0x01	$f_{INT\ CLK} = f_{OSCIN}/2$
		:	continues in sequence
		0x3F	$f_{INT\ CLK} = f_{OSCIN}/64$
			Note: This value changes the operating point of the PLL; changing this value while the PLL is active requires a re-lock.
15-0	PLL Multiplication Factor (PLLMUL)		Valid multiplication factors are from 92 to 184. NF = (PLLMUL / 256) + 1 $f_{VCO\ CLK} = f_{INT\ CLK}$x NF This multiplier is depicted as '*NF' in the FMzPLL Block Diagram.
		0x5B00	$f_{VCO\ CLK} = f_{INT\ CLK}$x 92
		0x5B80	$f_{VCO\ CLK} = f_{INT\ CLK}$x 92.5
		0x5C00	$f_{VCO\ CLK} = f_{INT\ CLK}$x 93
		:	continues in sequence
		0xB700	$f_{VCO\ CLK} = f_{INT\ CLK}$x 184
			The PLL checks the range of the multiplier. IF PLLMUL/256 > 183 OR PLLMUL/256<91, then PLLMUL/256 = 129
			Note: This value changes the operating point of the PLL; changing this value while the PLL is active requires a re-lock.

7.4.1.2 PLL Control 2 Register (PLLCTL2)

The PLLCTL2 register controls the frequency modulated mode of operation of the FMzPLL. The frequency modulation option is available for applications that have critical EMC considerations. Figure 7-9 illustrates this register and Table 7-3 provides the bit descriptions.

Figure 7-9. PLL Control 2 Register (PLLCTL2)[0xFFFF FF74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FM ENA	SPREADINGRATE[8:0]									Reserved	BWADJ[8:4]				
R/WP-0	R/WP-111111111									R-0	R/WP-00000				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWADJ[3:0]				ODPLL			SPR_AMOUNT[8:0]								
R/WP-0111				R/WP-001			R/WP-000000000								

R = Read, W = Write; P = Privilege Mode, -*n* = Value after reset

Table 7-3. PLL Control 2 Register (PLLCTL2) Field Descriptions

Bit	Name	Value	Description
31	Frequency Modulation Enable (FMENA)		
		0	Disable frequency modulation
		1	Enables frequency modulation
			Note: The PLL disables the clock output for 1 modulation period when this value is changed while the PLL is active.
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1 $f_{mod} = f_s = f_{INT\ CLK}/(2*NS)$ NS is the modulation frequency divider that is described in the Frequency Modulation Section. $f_{INT\ CLK}$ is the frequency of ‘INTCLK’ as depicted in the FMzPLL Block Diagram.
		0x000	$f_{mod} = f_s = f_{INT\ CLK}/(2*1)$
		0x001	$f_{mod} = f_s = f_{INT\ CLK}/(2*2)$
		:	continues in sequence
		0x1FF	$f_{mod} = f_s = f_{INT\ CLK}/(2*512)$
			Note: The PLL disables the clock output for 512*NR oscillator cycles if SPREADINGRATE is changed while FMENA is 1. If FMENA is 0, changing SPREADINGRATE has no impact on the clock availability.
21	Reserved		Read/Write, but value has no effect on PLL operation.

Table 7-3. PLL Control 2 Register (PLLCTL2) Field Descriptions (Continued)

Bit	Name	Value	Description
20-12	Bandwidth Adjustment (BWADJ)		NB = BWADJ + 1 $f_{BW} = f_{nom\_BW}/NB$ NB is the bandwidth divider that is described in the Frequency Modulation Section.
		0x007	$f_{BW} = f_{nom\_BW}/8$ (must be set to this value in non-modulation mode)
		0x008	$f_{BW} = f_{nom\_BW}/9$
		:	continues in sequence
		0x0FF	$f_{BW} = f_{nom\_BW}/256$
			The wrapper checks to insure that the BWADJ is programmed within a valid range. IF BWADJ < 7 OR BWADJ > 255, then BWADJ = 7.
			Note: The PLL disables the clock output for 512*NR oscillator cycles if BWADJ is changed while the PLL is active.
11-9	Internal PLL Output Divider (ODPLL)		OD = ODPLL + 1 $f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$ These bits must be changed before the PLL is enabled. This divider is depicted as '÷OD' in the FMzPLL Block Diagram.
		0x0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/1$
		0x1	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/2$
		:	continues in sequence
		0x7	$f_{post\text{-}ODCLK} = f_{VCO\ CLK}/8$
			Note: ODPLL cannot be changed while the PLL is active
8-0	Spreading Amount (SPR_AMOUNT)		NV = SPR_AMOUNT + 1 NV is the modulation depth divider that is described in the Frequency Modulation Section.
		0x000	NV = 1
		0x001	NV = 2
		:	continues in sequence
		0x1FF	NV = 512
			Note: The PLL disables the clock output for 1 modulation period if SPR_AMOUNT is changed while FMENA is 1. If FMENA is 0, changing SPR_AMOUNT has no impact on the clock availability.

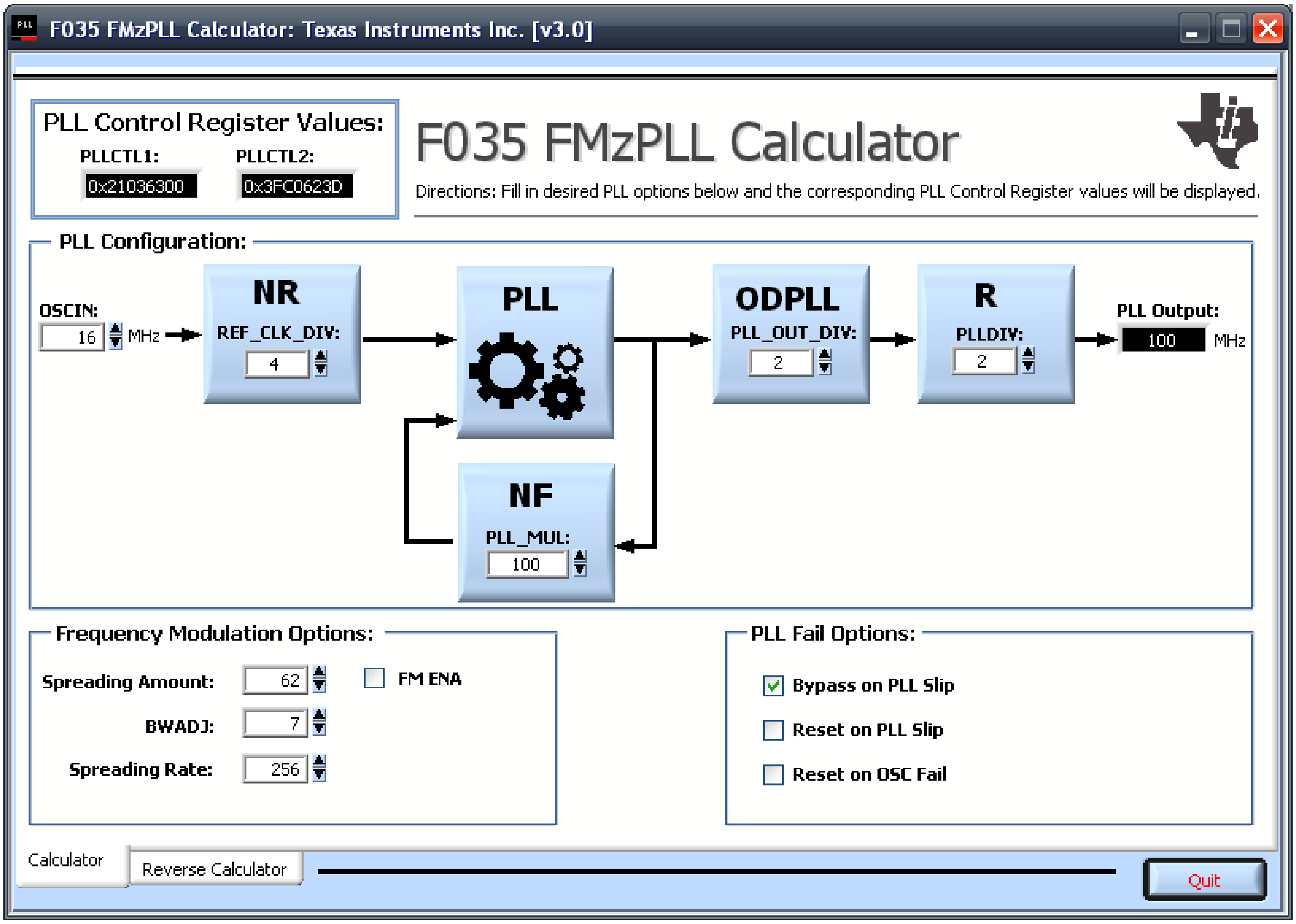
7.5 FMzPLL Calculator (F035 FMzPLL Calculator)

The F035 FMzPLL calculator is provided to assist with setting up the PLL. This calculator can be installed and run on a Windows XP based PC. It provides the user with control of the OSCIN speed, multiplier setting, divider settings, frequency modulation settings and also allows the user to set PLL/OSC fail options. Once the user has configured the desired options, the calculator displays the PLL output speed and the corresponding PLLCTL1 and PLLCTL2 register settings. These register settings can then be used to setup the PLL. The tool also contains a reverse calculator mode that will display all PLL options when values for PLLCTL1 and PLLCTL2 are entered. The F035 FMzPLL Calculator is shown below in Figure 7-10.

Note: F035 FMzPLL Calculator Warning

The FMzPLL calculator does not check for all FMzPLL configuration errors. It is the programmer's responsibility to determine if the FMzPLL is configured properly for device operation. Refer to the device datasheet for more information on how to avoid improper device configuration and FMzPLL malfunction.

Figure 7-10. F035 FMzPLL Calculator



F035 FMzPLL Calculator: Texas Instruments Inc. [v3.0]

PLL Control Register Values:

PLLCTL1: 0x21036300

PLLCTL2: 0x3FC0623D

F035 FMzPLL Calculator

Directions: Fill in desired PLL options below and the corresponding PLL Control Register values will be displayed.

PLL Configuration:

OSCIN: 16 MHz

NR REF_CLK_DIV: 4

PLL

ODPLL PLL_OUT_DIV: 2

R PLLDIV: 2

PLL Output: 100 MHz

NF PLL_MUL: 100

Frequency Modulation Options:

Spreading Amount: 62

☐ FM ENA

BWADJ: 7

Spreading Rate: 256

PLL Fail Options:

☑ Bypass on PLL Slip

☐ Reset on PLL Slip

☐ Reset on OSC Fail

7.6 FMzPLL Configuration Example

This section provides an example of how to program the PLL.

Suppose that using a 7MHz crystal, the application requires a

- 140MHz GCLK (and HCLK) frequency
- 134 kHz spreading frequency
- 1% spreading depth.

Then, using the algorithm from Section 7.3

1. Choose Output CLK frequency as integer divider of output frequency near to 240MHz. Output CLK frequency should not exceed 500MHz or fall below 120MHz.

   The integer values for 140MHz are 140MHz or 280MHz. For this example, select 280MHz since it is closer to 240MHz.

2. Choose the multiplication factor near to 120. Multiplier (NF) may not exceed 184 or fall below 92.

   7MHz/NR *(approximately 120) = 280MHz

   7MHz/3*120 = 280MHz

3. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum frequency of output divider R (device specific frequency).

   If the R-divider can accept 280MHz, then either OD or R can be set to 2 (and the other divider set to 1). If R-divider cannot operate at 280MHz, then the OD-divider must be set to 2.

4. Choose the modulation frequency divider NS

$$NS = \mathrm{round}\left(\frac{f_{OSCIN}}{NR}\frac{1}{2 \times f_S}\right) = \mathrm{round}\left(\frac{7[MHz]}{3}\frac{1}{2 \times 134[KHz]}\right) = \{8, 9\}$$

$$f_S = \frac{f_{OSCIN}}{NR}\frac{1}{2 \times NS} = \frac{7[MHz]}{3}\frac{1}{2 \times \{8, 9\}} = \{145.8, 129.6\}\ [KHz]$$

   For this example, select 129KHz (with NS = 9).

5. Choose the modulation depth divider NV

$$NV = \mathrm{round}\left(5.02154 \times 10^7 \times \frac{Depth \times f_S}{f_{OutputCLK}}\right) = \mathrm{round}\left(5.02154 \times 10^7 \times \frac{0.01 \times 0.1296[MHz]}{280[MHz]}\right) = 232$$

6. Choose the bandwidth divider NB

$$NB = \max\left\{\mathrm{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{f_{OutputCLK} \times \frac{f_{OSCIN}}{NR}}}{f_S}\right), 8\right\}$$

$$NB = \mathrm{ceiling}\left(\frac{0.00406562 \times 10 \times \sqrt{280[MHz] \times \frac{7[MHz]}{3}}}{0.1296[MHz]}\right) = 9$$

7. Setting only these fields (that is, not BPOS, ROF, or ROS) yields

   PLLCTL1 = 0x00027700

   PLLCTL2 = 0x820082E7

   INTCLK is 2.33MHz which falls within the permissible frequency range of 1.63 to 6.53 MHz

NF is centered in the range from 92 - 184 at 120

Output CLK has a frequency of 280MHz, falling within the permissible range of 120MHz to 500MHz and near the target of 240MHz.

OD is selected so that post-ODCLK meets the device specification

NB is selected to be large enough to allow modulation frequency f_s.

7.7 FPLL Introduction/Feature Overview

This section provides an overview of the Flexray phase-locked loop (FPLL) module. The FPLL is used to multiply the input frequency to a higher frequency, usually 80MHz for Flexray operation. The OSCIN frequency is limited to 10MHz, 16MHz or 20MHz to achieve the 80MHz required by the Flexray module in most applications. The FPLL is Clock Source 6 on this device.

7.7.1 Features

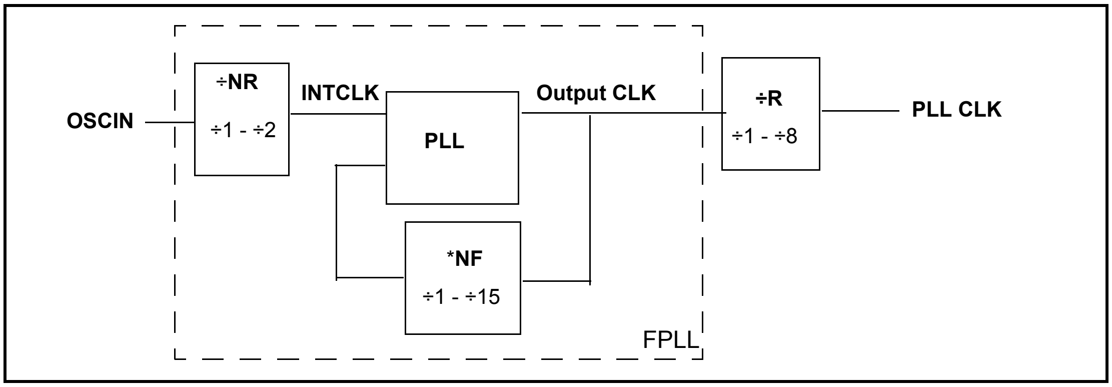
The main features of the FPLL clock module are:

- The phase-frequency detector assures lock to the fundamental reference frequency.
- The FPLL provides a wide range of scalability with the following configurability:
  - Reference Input divider (NR - *OSC_DIV*) has an integer range of 1 to 2.
  - Multiplier (NF - *PLL_MUL*) has an integer range of 1 to 15.
  - Prescaler (R - *PLL_DIV*) has an integer range from 1 to 8.

7.8 FPLL Operation

The FPLL clock module generates the PLL clock from an external resonator/crystal reference. The FPLL divides the reference input (NR) for a lower frequency input into the PLL; the reference input divider can be programmed to either 1 or 2. The internal clock to the PLL (INTCLK) has a valid range of 10MHz to 100MHz. The PLL multiplies this internal frequency by 1 - 15 (NF) with a valid range on the output clock of 10MHz to 250MHz. The PLL output CLK is subsequently divided by a prescale value (R), the value of R is an integer from 1 – 8. Figure 7-11 illustrates the functional blocks in the FPLL.

Figure 7-11. FPLL Block Diagram



The FPLL output frequency (PLL CLK) can be calculated as:

$$f_{PLL\ CLK} = f_{OSCIN} *NF/(NR*R)$$

Note:
The FPLL output frequency (Output CLK) should be set so that input to the prescaler R does not exceed the maximum device frequency. All the divider/multiplication values must be integer values in the range given.

There will be some delay before changes to the PLL setting take effect. It is best to disable the PLL prior to changing the settings. The delay shown below is valid from when the PLL is enabled. If the PLL is already enabled when the control registers are written, there may be an additional delay. When NR or NF is changed (or when the input clock is digitally re-enabled after exiting a low power mode in which the oscillator was stopped), the PLL output is held static for 4096 internal clock cycles ($t = 4096*NR/f_{OSCIN}$)

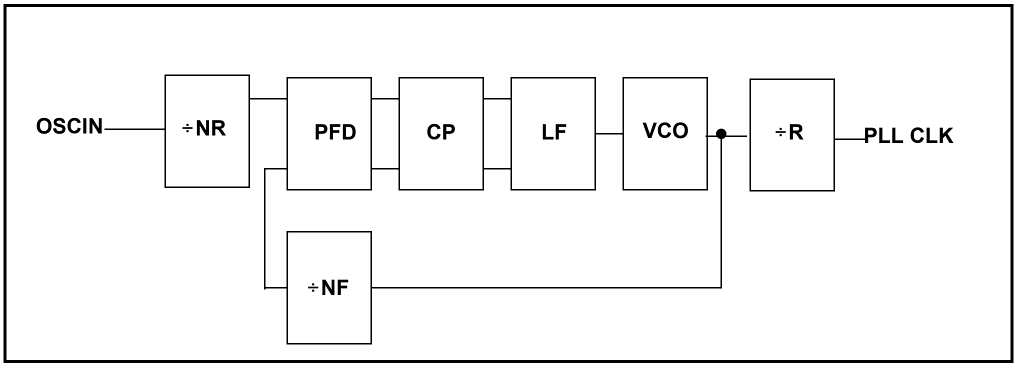
7.8.1 Phase-Locked Loop (PLL)

The PLL block consists of four logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)

Figure 7-12 further illustrates the sub-blocks of the FMzPLL blocks shown in Figure 7-11. 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 NF; this feedback divider requires the VCO to generate a frequency NF times greater than the internal frequency (INTCLK = OSCIN / NR).

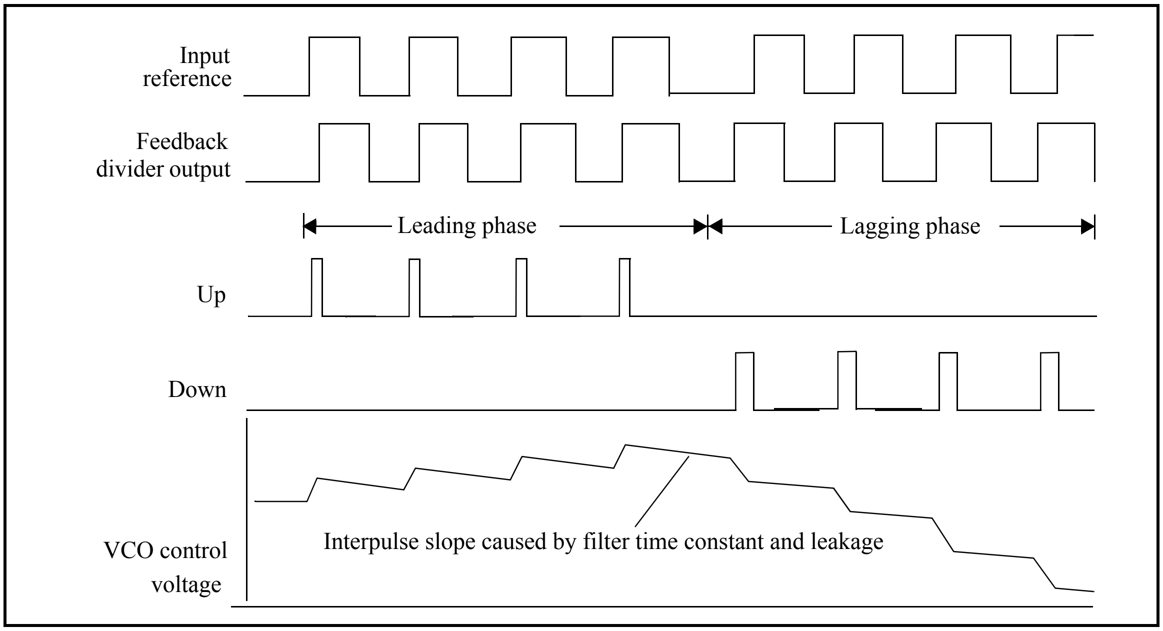
Figure 7-12. Basic PLL Circuit



7.8.1.1 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 7-13.

Figure 7-13. 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 10ns, then an up pulse of approximately 10ns is generated (see Figure 7-13). On the other hand, when the reference input lags the feedback input by 10ns, then a down pulse of approximately 10ns 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 the phase-frequency detector will not inadvertently lock to a harmonic or subharmonic of the reference frequency. This important property also ensures that the output frequency of the VCO is always exactly NF times the reference frequency.

7.8.1.2 Charge Pump and Loop Filter

The charge pump (CP) adds or removes 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.

7.8.1.3 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.

7.9 FPLL Control Register

7.9.1 Control register

This section describes the FPLL control register that controls the operation of the Flexray PLL (Clock Source 6). Figure 7-14 illustrates this register and Table 7-4 provides the bit descriptions

Figure 7-14. PLL Control Register 3 (PLLCTL3) [Location = 0xFFFF E100]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									OSC_ DIV	Reserved					
R-W-000000000									R-WP-0	R-W-000000					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PLL_MUL (3:0)				Reserved					PLL_DIV (2:0)		
R-W-0000				R-WP-0011				R-W-00000					R-WP-111		

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 7-4. PLL Control Register 3 (PLLCTL3) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads are zero and writes have no effect
22	OSC_DIV		Configures the OSC clock division factor (NR) $f_{INTCLK} = f_{OSCin} / NR$ NR = OSC_DIV + 1 This divider is depicted as '÷NR' in the FPLL Block Diagram.
		0	$F_{INTCLK} / 1$
		1	$F_{INTCLK} / 2$
21–12	Reserved		Reads are zero and writes have no effect
11–8	PLL_MUL[3-0]		Configures PLL reference clock multiplication factor (NF) between 1 and 15 $f_{OutputCLK} = f_{INTCLK} * NF$ NF = PLL_MUL[3-0] + 1 This multiplier is depicted as '*NF' in the FPLL Block Diagram.
		0000	$f_{OutputCLK} = 1 * f_{INTCLK}$
		0001	$f_{OutputCLK} = 2 * f_{INTCLK}$
		0010	$f_{OutputCLK} = 2 * f_{INTCLK}$
		0010	$f_{OutputCLK} = 3 * f_{INTCLK}$
		...	
		1101	$f_{OutputCLK} = 15 * f_{INTCLK}$

Table 7-4. PLL Control Register 3 (PLLCTL3) Field Descriptions (Continued)

Bit	Name	Value	Description
		1111	$f_{OutputCLK}$ = 1 * f_{INTCLK} (special case) **NOTE: (PLL_MUL+1) sets the multiply factor for the PLL except in the special case of PLL_MUL = 1111b which is logically equivalent to the setting of 0000b.**
7–3	Reserved		Reads are zero and writes have no effect
2–0	PLL_DIV[2-0]		Configures PLL division factor (R) f_{PLLCLK} = $f_{OutputCLK}$ / R R = PLL_DIV[2-0] + 1 This divider is depicted as '÷R' in the FPLL Block Diagram.
		000	f_{PLLCLK} = $f_{OutputCLK}$ / 1
		001	f_{PLLCLK} = $f_{OutputCLK}$ / 2
		010	f_{PLLCLK} = $f_{OutputCLK}$ / 3
		...	
		110	f_{PLLCLK} = $f_{OutputCLK}$ / 7
		111	f_{PLLCLK} = $f_{OutputCLK}$ / 8

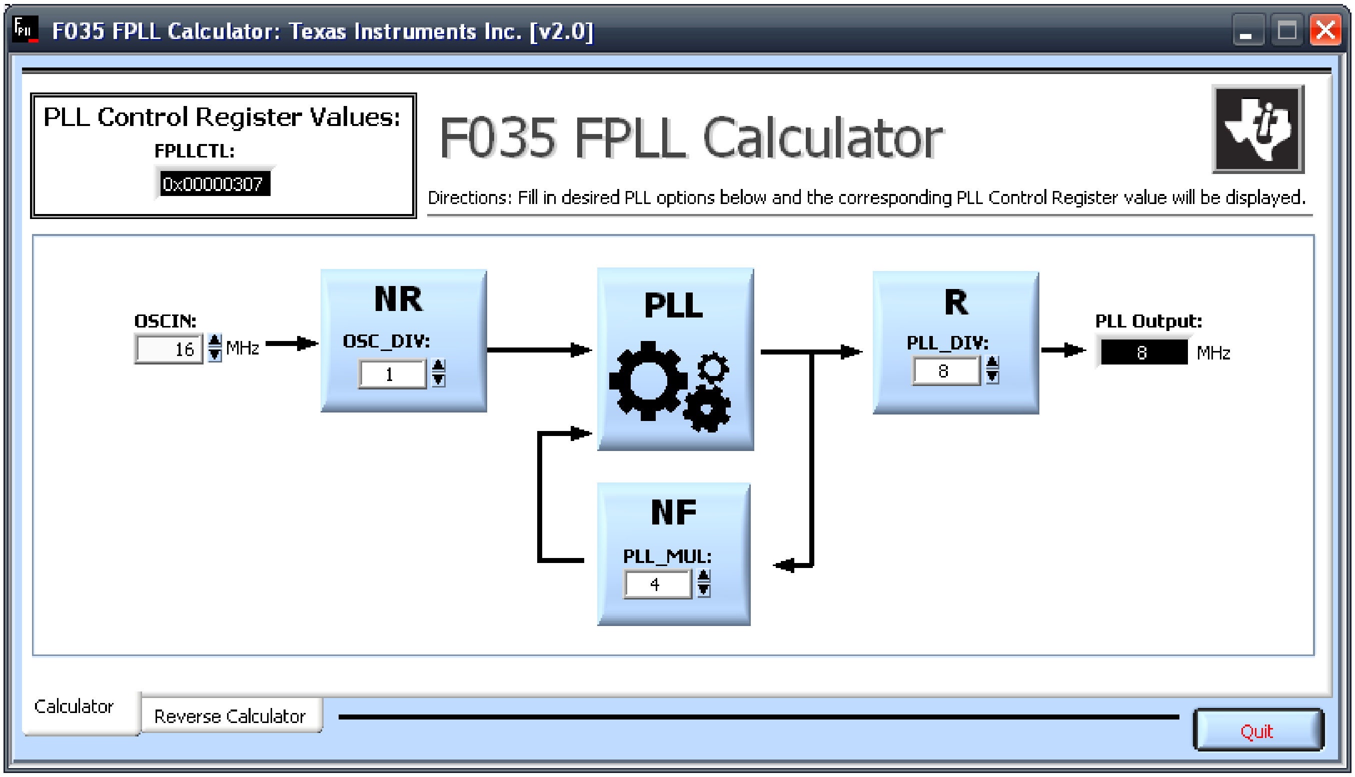
7.10 FPLL Calculator

The FPLL calculator is provided to assist with setting up the FPLL. This calculator can be installed and run on a Windows XP based PC. It provides the user with control of the OSCIN speed, input divider setting, multiplier setting, and output divider setting. Once the user has configured the desired options, the calculator displays the PLL output speed and the corresponding PLLCTL3 register settings. These register settings can then be used to setup the PLL. The tool also contains a reverse calculator mode that will display all PLL options when values for the PLLCTL3 register is entered. The FPLL Calculator user interface is shown below in Figure 7-15.

Note: F035 FPLL Calculator Warning

The FPLL calculator does not check for all FPLL configuration errors. It is the programmer's responsibility to determine if the FPLL is configured properly for device operation. Refer to the device datasheet for more information on FPLL configuration.

Figure 7-15. F035 FPLL Calculator



F035 FPLL Calculator: Texas Instruments Inc. [v2.0]

PLL Control Register Values:

FPLLCTL: 0x00000307

F035 FPLL Calculator

Directions: Fill in desired PLL options below and the corresponding PLL Control Register value will be displayed.

OSCIN: 16 MHz

OSC_DIV: 1

PLL_MUL: 4

PLL_DIV: 8

PLL Output: 8 MHz

7.11 FPLL Configuration Example

This section provides an example of how to program the FPLL, the formula to calculate the FPLL CLK is below:

$\mathbf{f_{PLL\ CLK} = f_{OSCIN} *NF/(NR*R)}$

Suppose that using a 16MHz crystal, the application requires a 80MHz PLL output frequency

1, The INTCLK of the FPLL has to be in the range from 10MHz to 100MHz, so with a 16MHz crystal the OSC_DIV (Oscillator Division factor) has to be configured to 1 (NR=1).

$f_{INTCLK} = f_{OSCin}$ / NR

f_{INTCLK}= 16MHz/1= 16MHz

2, The internal PLL frequency get multiplied with the factor PLL_MUL. The Output CLK of the PLL has to be in the range of 100MHz to 250MHz and the maximum Output CLK frequency does not exceed the maximum frequency of output divider PLL_DIV (device maximum frequency).

$f_{OutputCLK} = f_{INTCLK}$ * NF

To get a multiplication by 10 the value 1001b needs to be written into the PLL_MUL bitfield. So the Output CLK is calculated as following:

$f_{OutputCLK}$ = 16MHz * 10 = 160MHz

3, To get the desired FPLL frequency the divider PLL_DIV needs to be configured:

$f_{PLLCLK} = f_{OutputCLK}$ / R

The f_{PLLCLK} should be 80MHz, and the $f_{OutputCLK}$ is 160MHz, so the divider R needs to be 2. To get this division rate the PLL_DIV bitfield needs to be configured to the value 001b.

f_{PLLCLK} = 160MHz / 2 = 80MHz

Based on the above, the value of the FPLL control register is 0x00000901.

F035 Flash Module

The flash electrically erasable programmable read-only memory module is a type of nonvolatile memory which has fast read access times and is able to be reprogrammed in the field or in the application. This section describes the F035 flash module.

Topic Page

8.1 Overview

The F035 Flash is primarily used to provide the instruction memory to the Cortex-R4F CPU. It can also be used as static data memory such as calibration tables. The flash can be electrically programed and erased many times to ease code development.

8.1.1 Features

- Supports up to 4 banks and 512 kbyte each.
- Read, program and erase with a single 3.3v supply voltage.
- Supports error detection and correction.
  - Single Error Correction and Double Error Detection (SECDED)
  - ECC (Error Correction Code) is evaluated in the CPU which provides memory bus protection
  - 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 time (refer to the device datasheet for details).

8.1.2 Definition of Terms

Terms used in this document have the following meaning:

- 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.
- 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 which share input/output buffers, data paths, sense amplifiers, and control logic. Flash bank can store both program instructions and data.
- Flash module: Flash banks, charge pump, and flash wrapper.
- Flash wrapper (FWM): Power and mode control logic, data path, wait logic, and write/erase state machines.
- 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 which must be erased simultaneously.
- Standard read mode: The mode assumed when the pipeline mode is disabled. Physically, 128 bits of data (+ 16 bit ECC) is read at a time. However, only 32 bits of data is used while the other 96 bits of data is 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.

8.1.3 F035 Flash Tools

Texas Instruments provides the following tools for F035 flash:

- *nowECC* tool - to generate the flash ECC from the flash data.
- *nowFlash* tool - to erase/program/verify the flash content through JTAG.

  An erased cell reads 1 and a programmed cell reads 0.
- Code Composer Studio V4.x - the development environment with integrated flash programming capabilities.

8.2 Default flash configuration

At power up, the flash module state exhibits the following properties:

- ECC inside Cortex-R4F is disabled
- Wait states are set to 1
- 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 by initializing the FRDCNTL register to achieve the optimum system performance. This needs to be done before switching to the final device operating frequency. The required data and address waitstates for timing can be found in the device datasheet.

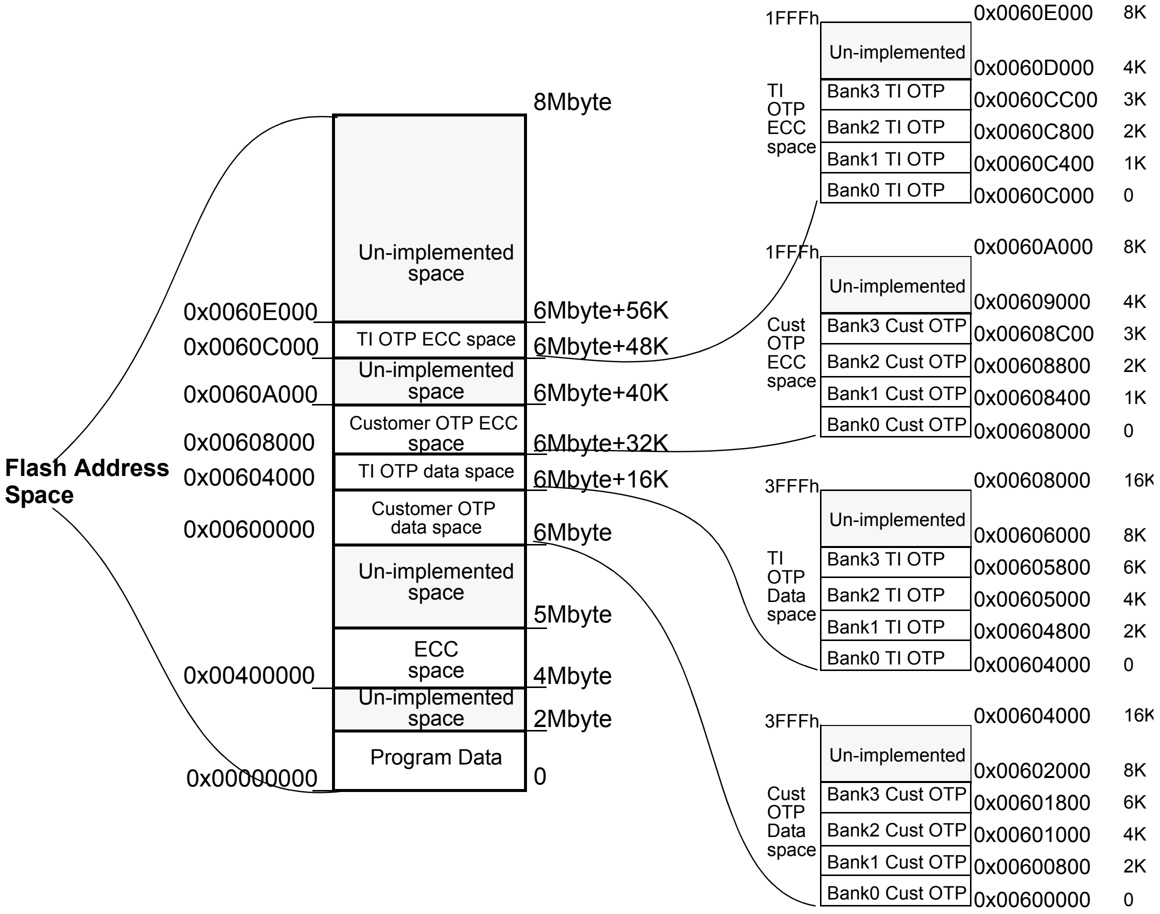
8.3 Memory Map

The flash module contains the program memory, which is usually mapped starting at location zero, and one Customer OTP sector and one TI OTP sector per bank. The Customer OTP sectors may be programmed by the customer, but can not 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 state machine for erase and program operations.

Usually, the flash memory is located from 0x0 to 0x007FFFFF, total 8 Mbytes. Figure 8-1 shows the typical memory map of a F035 device. Please refer to the device specific datasheet for the Bank/Sectoring information.

In order to make it possible to read the ECC bits without triggering an ECC error the flash contents are also mirrored at address 0x20000000. Section 8.4 discusses the mirrored memory in more detail.

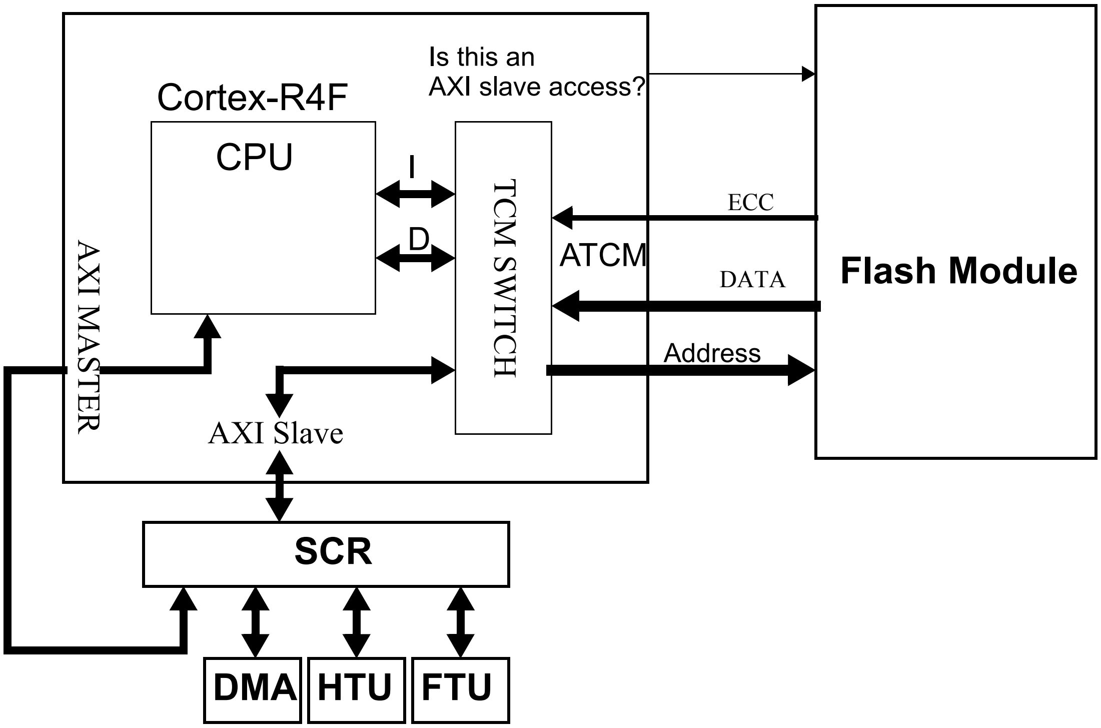
Figure 8-1. Flash Memory Map of F035 Device



8.4 CPU ATCM access and AXI Slave Access

Figure 8-2 shows the diagram of the two different read paths to access the flash - CPU ATCM (Tightly-Coupled Memory interface A) access and AXI (Advanced eXtensible Interface) slave access. During normal operation, the CPU accesses flash through the ATCM address space. We call this type of access a CPU ATCM access. A CPU ATCM access is a CPU read of any address from 0x0 to 0x007FFFFF. An AXI slave access is defined as an access that goes through the SCR and into the AXI slave port. The request routes to the ATCM from the AXI slave port. Some examples of AXI slave accesses are: CPU read of mirrored address (from 0x20000000 to 0x207FFFFF), DMA read, HTU read and FTU read. The Cortex-R4F CPU will see the mirrored address as outside of the ATCM space, and generate an access from the AXI master port. The request from the AXI master port will go through the SCR and into the AXI slave port. From here the request is routed to the ATCM, but it is seen as an AXI slave access.

Figure 8-2. CPU ATCM Access and AXI Slave Access Diagram



Note:
AXI slave access will NOT trigger ECC error if the flash content is all 1's or all 0's (When EOCV and EZCV bits are set) or EDACMODE = 5.

8.4.1 Illegal Address Generation

Any un-implemented flash space not decoded to a flash bank will cause an illegal address signal to be generated. The illegal address signal forces the CPU to take either a data or an instruction abort.

To optimize performance, this device uses the ECC logic internal to the Cortex-R4F to protect the flash memory. If the ECC in the Cortex-R4F is disabled (refer to Example 2), reading any of implemented flash memory area will not generate ECC or ESM (Error Signaling Module) errors. If the ECC in the Cortex-R4F is enabled (refer to Example 1), the user should follow the recommendation in section 8.5.3.6 to access the ECC memory area.

8.5 Operation

The following sections discuss various modes of operation related to reading, power modes, data protection, and wait state generation.

8.5.1 Flash Read Modes

In addition to *standard read mode*, the flash module also has *pipeline mode*, which affects the technique used to fetch the next memory word. Using this mode allows increased clock speeds and CPU throughput.

8.5.1.1 Standard Read Mode

Standard read mode is defined as the mode in effect when pipeline mode is inactive. Standard read mode is the default read mode after reset. During standard read mode, each read access to flash is decoded by the flash wrapper to fetch the data from the addressed location. The data is returned after the RWAIT number of cycles. RWAIT defines the number of random access wait states. After reset the RWAIT has the reset value of one wait state. Address wait states are not used in standard read mode.

No pipeline buffers are used in standard read mode so every access is used immediately and every access creates a unique flash bank access.

Standard read mode is the recommended mode for lower frequency operation of which RWAIT can be set to zero to provide single cycle access operation. The flash wrapper can operate at higher frequency using non-pipeline mode at the expense of adding wait states. At higher frequencies, it is recommended to enable pipeline mode. Please see the device specific datasheet to determine maximum frequency allowed in standard read mode (non-pipeline mode).

Pipeline mode is disabled by clearing the ENPIPE bit in the flash control register FRDCNTL.0.

8.5.1.2 Pipeline Mode

Pipeline mode removes the flash memory access time for sequential addresses from the critical timing path, which increases clock speeds and CPU throughput.

In pipeline mode, both data wait states (RWAIT) and Address Setup Wait State are required for high frequency operation. In pipeline mode there is always at least one data wait state even when they are set to 0. The Address Setup Wait State is only available in pipeline mode and can be enabled by setting the ASWSTEN bit. Please see the device specific datasheet to determine maximum frequency allowed in pipeline mode versus the number of data and address wait states required.

Pipeline mode is enabled by setting the ENPIPE bit in the flash control register FRDCNTL.0.

8.5.1.3 Read Margin Modes

Read margin modes 0 and 1 are used to test cells for marginality. Read Margin 0 is used to test cells for marginality of programmed cells. Read Margin 1 is used to test cells for marginality of erased cells.

Read margin modes should only be entered when executing from RAM. All banks must be powered up before entering a read margin mode. When entering a read margin mode, or changing from one read margin mode to the other, allow 1us delay before the first flash access to allow the flash banks to adjust to the new mode.

Read margin modes 0 or 1 are enabled by setting the RM0 or RM1 bit in the special read control register FSPRD. The recommended procedures to enter, change and leave read margin mode are shown below:

To enter read margin mode.

1. Set the banks to remain powered up by writing 0xFF to bits 0:15 of FBFALLBACK register.
2. Do one dummy read from each bank to turn on the banks.
3. Start execution out of RAM.
4. Turn on read margin 0 or 1 by writing 1 or 2 to FSPRD register.
5. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
6. Do two dummy reads from each bank.

7. Wait 1 us

   Any reads will now be with the selected read margins. The application can now return to flash if desired.

To change read margin mode to the other

1. Start execution out of RAM.
2. Switch read margin mode to the other by writing 2 or 1 to FSPRD register.
3. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us

   Any reads will now be with the selected read margins. The application can now return to flash if desired

To leave read margin mode:

1. Start execution out of RAM.
2. Turn off read margin 0 and 1. Write 0 to FSPRD register.
3. Flush the data buffer by reading two RAM locations separated by at least 32 bytes.
4. Do two dummy reads from each bank.
5. Wait 1 us

   Any reads will now be a standard read. The application can now return to flash if desired. The application can also set the FBFALLBACK register to the original values.

This read-margin mode is normally entered during power-up and a full CRC check of the flash contents is performed. In this way a potential read mode failure can be identified prior to data loss or gain causing a bit flip.

Note:
The read-margin modes are intended for diagnostic capabilities and the flash content is guaranteed for the lifetime specified in the datasheet.

8.5.2 *Erase/Program Flash*

The F035 Flash should be programmed, erased and verified only by using the F035 Flash API library. These functions are written, compiled and validated by Texas Instruments. The flash module contains a Command State Machine (CSM) to perform program, erase, and validate operations. This section only provides a high level description for these operations, please refer to the F035 Flash API library documentation for more information.

A typical flow to program flash is:

Compact → ***Erase*** → ***Program*** → ***Verify***

8.5.2.1 *Compact*

A device may contain depleted (over erased) columns and/or marginally erased bits. The application code should validate the target flash before erasing it. Otherwise, the leakage current caused by the depleted bits might confuse the sense amplifier in other operation. The CSM provides a “Validate sector” command to perform compaction on the target sector as needed. This command is implemented by the following Flash API function:

Flash_Compact_B().

This function validates the flash data and ECC together. For example, if this function is called to validate the flash data of sector 0 in bank1, it validates both this sector and the corresponding ECC area.

8.5.2.2 *Erase*

The target flash is ready for erasing after it is validated by the compact function. After erasing, the target flash reads as all '0xFFFFFFFF's. This state is called as 'blank'. The erase function must be executed before programming. The user should NOT skip erase on sectors that read as 'blank' because these sectors may require additional erasing or compaction due to marginally erased bits or depleted columns. The CSM provides an “Erase Sector” command and an “ Erase Bank” command to erase the target sector or bank. Similar to the compact function, the erase function erases the data and the ECC together. These commands are implemented by the following Flash API functions:

Flash_Erase_Bank_B() ; //erase the target bank (Support disabling preconditioning)

Flash_Erase_B(); //erase the target sector (Support disabling preconditioning)

Flash_Erase_Sector_B(); //erase the target sector (Do not support disabling preconditioning)

where preconditioning means “Programming 0’s prior to applying erase pulses”. The application must make sure that the target flash is 'blank' before disabling preconditioning. The Flash API provides the following function to determine if the flash bank is 'blank' before disabling preconditioning erasing:

Flash_Blank_B().

This function can also be used to verify the flash has been properly erased.

8.5.2.3 *Program*

The CSM provides a “Program customer OTP” command and a “Program Data” command to program the customer OTP area and program data area; The CSM also provides a “Program Customer OTP Check Bits” command and a “Program Check Bits” command to program the customer OTP ECC and data ECC. These commands are implemented by the following Flash API functions:

Flash_Prog_B(); //Program flash ECC and data.

OTP_Prog_B(); //Program customer OTP ECC and data.

Different from the compact and erase function, the program function programs the flash data and ECC separately. The user must generate the flash ECC data (e. g. by calling *the nowECC* tool, which is developed by TI to generate ECC data from a compiled outfile) and program the ECC into the target flash ECC area.

8.5.2.4 *Verify*

After programming, the user must perform verify using Read Margin Mode. The Verify functions are implemented in the Flash API functions Flash_Verify_B() and Flash_PSA_Verify_B().

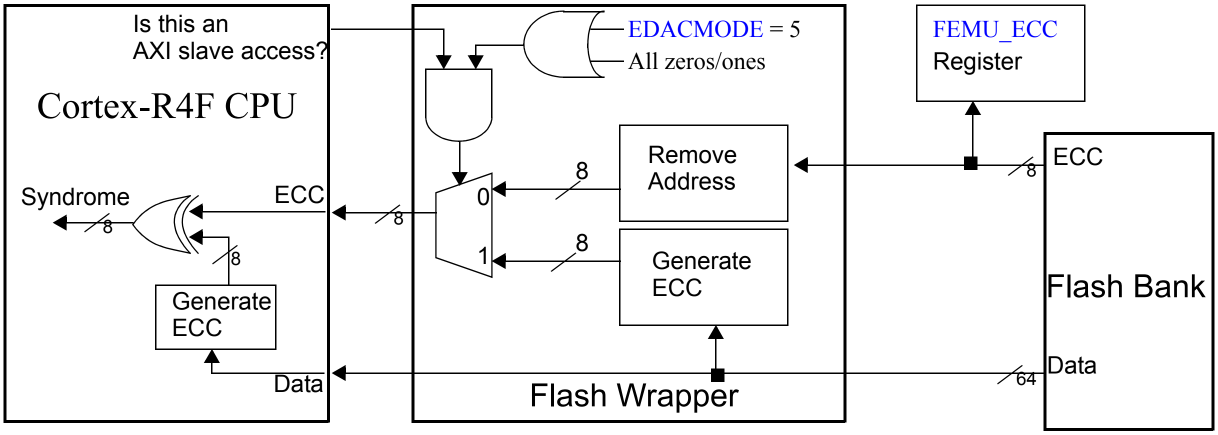
8.5.3 ECC Protection

The Cortex-R4F CPU implemented in this device contains an embedded ECC (Error Correction Code) logic. When enabled, the ECC logic provides the capability of SECDED (Single Error Correction and Double Error Detection). The ECC logic requires a total of 8 ECC bits for each 64 bits of data to be stored in the flash memory.

Figure 8-3 illustrates how the ECC logic works during a read operation. This is done by following three steps:

6. The flash wrapper reads the 8 ECC bits and 64 bits of data from flash bank. ECC bits stored in flash also include address information which is used by the flash wrapper to ensure the correct bank is being addressed.
7. The flash wrapper processes the data and ECC bits.
   a. During a normal read, the flash wrapper removes the address component from the 8 read ECC bits and send them to the Cortex-R4F CPU. Please see section 8.5.3.1 for detail information.
   b. Under a few special conditions, the flash wrapper generates a correct ECC value from the data and sends it to the Cortex-R4F CPU. In this case, the ECC bits sent to CPU are always correct for the 64 bits of read data and the ECC bits read from flash bank are discarded. Please see section 8.5.3.2 for detail information.
8. Third, on the Cortex-R4F CPU side, the ECC logic inside Contex-R4F generates eight ECC bits based on the 64 bits read data value. These eight calculated ECC bits are then XORed with the pre-determined ECC bits associated with the data read from the flash. The 8-bit output is the syndrome. The syndrome is decoded to determine one of three conditions:
   c. No error occurred
   d. A correctable error occurred
   e. A non-correctable error occurred

Figure 8-3. Cortex-R4F Flash ECC Read Path



Note:
Since ECC is calculated for an entire 64-bit data, a non 64-bit read such as a byte read or a half-word read will still force the entire 64-bit data to be read and calculated but only the byte or half-word will be actually used by the CPU.

8.5.3.1 ECC Generation for CPU ATCM Accesses

In this mode, the flash wrapper always

1. Reads the eight ECC bits in flash.
2. Strips out the address component from the ECC value.

3. Sends the ECC value to the CPU.

8.5.3.2 ECC Generation for AXI slave Accesses

As shown in Figure 8-3, during AXI slave access, the flash wrapper will generate a correct ECC value from the data but not check the flash bank ECC values under any of the following three special conditions.

1. EDACMODE = 5; Or,
2. the flash data is all ones and the EOCV bit in register FEDACCTRL1 are set to 1; Or,
3. the flash data is all zeros and the EZCV bit in register FEDACCTRL1 are set to 1.

In these cases, the ECC logic inside the CPU only checks the data bus from the flash to the CPU, but does not check the flash data ECC bits. It can be used when the flash ECC is not programmed with correct ECC (e.g. flash is erased).

Under all other conditions, the AXI slave access will follow the steps in section 8.5.3.1 to generate and send ECC bits to CPU.

8.5.3.3 Enabling / Disabling ECC inside CPU

Since the ECC is provided by the Cortex-R4F CPU, globally the ECC logic is enabled and disabled by accessing CPU registers.

Example 1. Enabling ECC Detection inside Cortex-R4F

```
_ACTM_Enable_ECC
          MRC p15,#0,r1,c9,c12,#0;   Read PMNC register in privileged mode
          ORR r1, r1, #0x00000010;  Bit4 enable export of the events to the event bus (send
                                     ECC error to the flash wrapper)
          MCR p15,#0,r1,c9,c12,#0;   Write PMNC register in privileged mode

          MRC p15, #0,r1,c1,c0,#1;   Read Auxiliary control register
          ORR r1, r1, #0x02000000;   Bit25 is ATCM ECC check enable
          MCR p15, #0, r1, c1, c0, #1; Write Auxiliary control register

          MRC p15, #0,r1,c1,c0,#1;   Read Auxiliary control register
          ORR r1, r1, #0x00000001;   Bit0 is ATCM external error into CPU enable
          MCR p15, #0, r1, c1, c0, #1; Write Auxiliary control register

          MOV PC, lr
```

Example 2. Disabling ECC Detection inside Cortex-R4F

```
_ACTM_Disable_ECC
          MRC p15, #0, r1, c1, c0, #1; Read Auxiliary control register
          MVN R0,#0x1 <<25;            Clearing Bit25 disables ATCM ECC check
          AND R1 ,R1, R0
          DMB
          MCR p15, #0, r1, c1, c0, #1; Write Auxiliary control register
          ISB

          MRC p15, #0, r1, c1, c0, #1; Read Auxiliary control register
          MVN R0,#0x1 <<0;            Clearing Bit0 disables ATCM external error into CPU
          AND R1 ,R1, R0
          DMB
          MCR p15, #0, r1, c1, c0, #1; Write Auxiliary control register
          ISB

          MOV PC, lr
```

8.5.3.4 Blocking Single and Multiple Bit Errors from the CPU

Once the ECC logic in CPU detects a flash ECC error, it will notify the flash wrapper. Then, the flash error will trigger an ESM (Error Signaling Module - see ESM user guide for detail information) error. This section discusses how to prevent flash ECC errors showing as ESM errors. This feature can be used to prevent unintended ESM errors due to speculative reads. The single and double bit flash ECC errors are blocked under following conditions.

1. If EDACEN is set to 0x5, CPU single and double error signals are blocked for all flash memory accesses.

Note:
This mode is a legacy from previous devices. Texas Instruments does NOT recommend to use this mode when the ECC logic in the CPU is enabled.

2. If EDACEN is set to any value other than 0xD, ECC errors generated from the OTP memory regions are blocked. Only if EDACEN is set to 0xD are OTP memory generate ESM errors.
3. At any given time, ECC errors in four different sectors can be blocked during CPU ATCM access. This is done by specifying the bank/sector to be excluded and its inverse value in the sector disable register FEDACSDIS and FEDACSDIS2. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ESM errors. After reset the sector register is reset to all zeros which means that no sector is excluded from ESM errors.

In normal operation, to prevent the ESM errors due to speculative reads of ECC and OTP space, EDACEN should be NOT set to 0xD; to prevent the ESM errors due to speculative reads of flash sectors with wrong ECC value (sectors used for special purpose), ESM errors of these sectors should be blocked through programming the FEDACSDIS and FEDACSDIS2 registers.

Speculative Reads: The Cortex-R4F CPU might do a data read because it sees a conditional instruction in the pipeline that may generate a read. If the read is not used, it is discarded by the CPU, but the ECC checking has already occurred. Since the address in the register used for the read is not necessarily pointing to a location that would normally be read, this speculative read could occur anywhere in the ATCM space.

Table 8-1 summarizes the effects of registers (EDACMODE and EDACEN) on ECC generation and CPU events.

Table 8-1. Effects of Registers on ECC Generation and ESM Errors

AXI Slave Access	FWM Generate ECC from Data		ESM Respond to Flash ECC Error		
	EDACMODE		EDACEN		
	=5	=5	=5	=D	Other
OTP	N	Y	N	Y	N
OTP all 1s or 0s	Y	Y	N	Y	N
OTP ECC	N	Y	N	N	N
MAIN ECC	N	Y	N	N	N
All 1s or 0s	Y	Y	N	Y	Y
Disabled sector[1]	N	Y	N	Y	Y
Program Memory	N	Y	N	Y	Y
CPU ATCM Access	**FWM Generate ECC from Data**		**ESM Respond to Flash ECC Error**		
			EDACEN		
			=5	=D	Other
OTP	N		N	Y	N
OTP all 1s or 0s	N		N	Y	N
OTP ECC	N		N	N	N
MAIN ECC	N		N	N	N
All 1s or 0s	N		N	Y	Y
Disabled sector[1]	N		N	Y	N
Program Memory	N		N	Y	Y

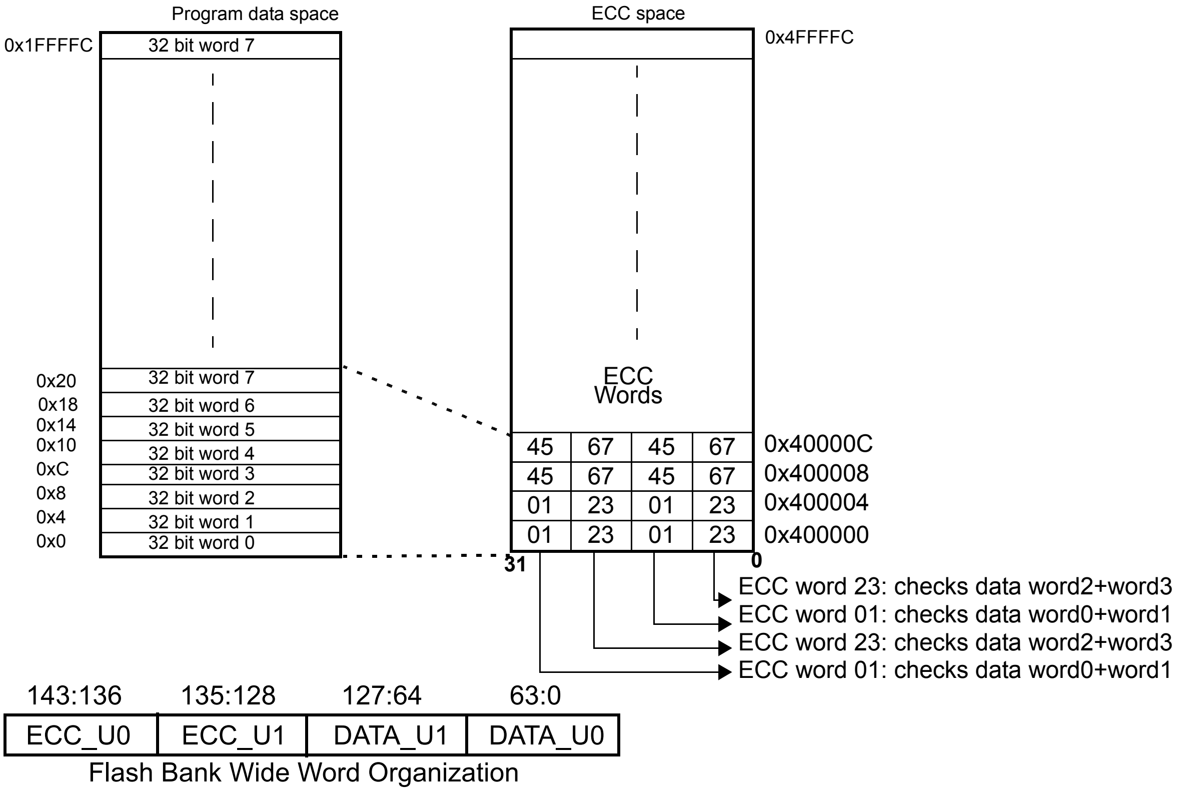
[1] Sectors defined in FEDACSDIS or FEDACSDIS2.

8.5.3.5 ECC Memory Map

In this device ECC bits are mapped to a 4M byte offset from the flash memory base address. Each 64-bit data has its corresponding 8 ECC bits mapped to an address in the ECC space. Refer to Figure 8-4 for an illustration.

As shown in Figure 8-4, the flash bank in this device is 144 bits wide, including 128 bits for normal data width and two sets of 8 ECC bits. Each 8 ECC bits check 64 data bits and 19 address bits. Logically, these 8 ECC bits repeat themselves four times as shown in Figure 8-4. In other words, physically, flash ECC occupies only 1/8 of the corresponding flash data; logically, the size of flash ECC is half of the corresponding flash data.

Figure 8-4. ECC Word Memory Mapping in CPU Address Space



Only 16-bit and 8-bit reads are supported when reading from the ECC space. The flash wrapper cannot support 32-bit reads of the ECC bits because only 16 ECC bits are implemented in one flash row physically. During both 16-bit and 8-bit reads, the flash wrapper will duplicate the physical 16 ECC bits as shown in Figure 8-4.

8.5.3.6 Reading the ECC Bits

ECC values should be read from the RD_ECC field of the FEMU_ECC register during AXI slave access (refer to Figure 8-3). The ECC bits can be read directly by the CPU, but this could create an ECC error. If required to be read directly by the CPU the user can disable ECC checking in the CPU, but this will mean no ECC protection during this operation.

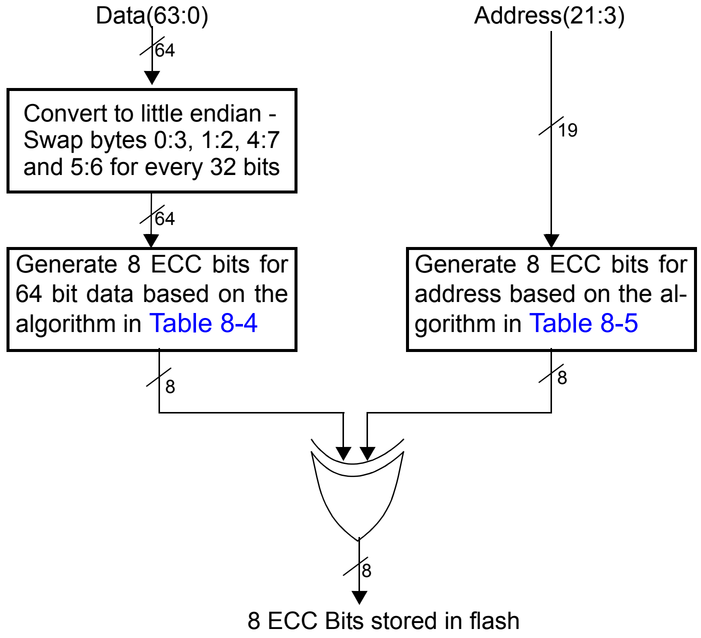
To load the FEMU_ECC register with ECC values, application software should read the data from the program data memory and not the special ECC memory region. Make sure the read is an AXI slave access read. Also make sure the read is a data read and not an opcode or instruction fetch. After this the FEMU_ECC register will contain the ECC bits for the specific address read during the AXI access.

8.5.3.7 ECC Generation Algorithm

The F035 Flash stores 8 ECC bits for every 64 bits of data. This ECC is derived from both address and data values to protect both fields. As shown in Figure 8-5, it takes four steps to generate the 8 ECC bits.

1. Convert each 32-bit word in to little endian format
2. Generate the 8 ECC bits for data only
3. Generate the 8 ECC bits for address only
4. XOR the results from 2 and 3 together

Figure 8-5. Flash ECC Calculation


Data(63:0)
Address(21:3)
64
Convert to little endian - Swap bytes 0:3, 1:2, 4:7 and 5:6 for every 32 bits
19
64
Generate 8 ECC bits for 64 bit data based on the algorithm in Table 8-4
Generate 8 ECC bits for address based on the algorithm in Table 8-5
8
8
8
8 ECC Bits stored in flash

This device is a big endian (BE32) device. However, the ECC logic in the Cortex-R4F generates ECC bits based on little endian format. Therefore, the first step to calculate the ECC is to "Convert each 32-bit word in to little endian format". For every 32-bit word, swap bytes 0:3, 1:2, 4:7 and 5:6 before calculating the ECC. For example, assume the application software has 0x01234567 at address 0x2415D8 and 0x89ABCDEF at address 0x2415DC, the data should be re-formatted as follows:

Table 8-2. Data in Flash (BE32)

ADDR 21:0	Data 31:0	ADDR ECC	Data ECC	Final ECC
0x2415D8	0x1234567	0x97	0x0C	0x9B
0x2415DC	0x89ABCDEF			

Table 8-3. Data in Little Endian Format

ADDR 21:0	Data 31:0
0x2415D8	0x67453412
0x2415DC	0xEFCDAB89

The second step is to "Generate the 8 ECC bits for data only". Table 8-4 shows how to encode the ECC bits for 64 bits of data (in little endian format) without address information. Each ECC bit is calculated as the parity bit for the corresponding data bits marked with 'x' in the same column. The second row shows the corresponding parity used for each ECC bit calculation. For even parity, XOR all data bits with an 'x' across each column to get the corresponding ECC bit. For odd parity, XNOR all data bits with an 'x' across each column to get the corresponding ECC bit. For example, ECC bit 7 is xor_reduce(D55:40,D31:24,D7:0). This could also be calculated as: Sum(D55:40,D31:24,D7:0) modulo 2.

For example, the 64 bits of data in Table 8-3 will generate the 8 bit ECC value 0x0C for the data only.

Table 8-4. ECC Encoding inside Cortex-R4F for the Data Bits

ECC bits[1]	ECC[7]	ECC[6]	ECC[5]	ECC[4]	ECC[3]	ECC[2]	ECC[1]	ECC[0]
Parity[2]	Even	Even	Even	Even	Odd	Odd	Even	Even

Participating Data bits	63		x	x	x		x		x
	62		x	x	x				
	61		x	x		x	x		x
	60		x	x		x		x	x
	59		x	x		x			
	58		x	x			x	x	x
	57		x	x			x		
	56		x	x				x	
	55	x			x	x	x		x
	54	x			x	x		x	x
	53	x			x	x			
	52	x			x		x	x	x
	51	x			x		x		
	50	x			x			x	
	49	x				x		x	
	48	x				x	x	x	x
	47	x		x	x		x		x
	46	x		x	x				
	45	x		x		x	x		x
	44	x		x		x		x	x
	43	x		x		x			
	42	x		x			x	x	x
	41	x		x			x		
	40	x		x				x	
	39		x		x	x	x		x
	38		x		x	x		x	x
	37		x		x	x			
	36		x		x		x	x	x
	35		x		x		x		
	34		x		x			x	
	33		x			x		x	
	32		x			x	x	x	x
	31	x	x	x	x		x		
	30	x	x	x	x				x
	29	x	x	x		x	x		
	28	x	x	x		x		x	
	27	x	x	x		x			x
	26	x	x	x			x	x	
	25	x	x	x			x		x
	24	x	x	x				x	x
	23				x	x	x		
	22				x	x		x	
	21				x	x			x
	20				x		x	x	
	19				x		x		x
	18				x			x	x
	17					x		x	x
	16					x	x	x	
	15			x	x		x		
	14			x	x				x
	13			x		x	x		
	12			x		x		x	
	11			x		x			x
	10			x			x	x	
	9			x			x		x
	8			x				x	x
	7	x	x		x	x	x		
	6	x	x		x	x		x	
	5	x	x		x	x			x
	4	x	x		x		x	x	
	3	x	x		x		x		x
	2	x	x		x			x	x
	1	x	x			x		x	x
	0	x	x			x	x	x	

Notes: 1) Each ECC[x] bit represents the parity bit for the corresponding data bits marked with x in the same column.

2) The ECC bit is generated as either an XOR(Even) or an XNOR(Odd) of the data bits marked with x in the same column.

The third step is to "Generate the 8 ECC bits for address only". The ECC calculation or algorithm for the address is done as shown in Table 8-5. XOR all address bits with an 'x' across each column to get the corresponding ECC(x) bit. For example, ECC bit 7 is xor_reduce(A9,A8,A7,A6,A5,A4,A3). This could also be calculated as: (A9+A8+A7+A6+A5+A4+A3) modulo 2. For example, the address in Table 8-2 will generate the 8 bit ECC value of 0x97 for the address only.

Table 8-5. ECC Encoding for the Address Bits

ECC bits[1]		ECC[7]	ECC[6]	ECC[5]	ECC[4]	ECC[3]	ECC[2]	ECC[1]	ECC[0]
Parity[2]		Even	Even	Even	Even	Even	Even	Even	Even
ADDR	HEX	0007F	7FF80	07F80	19F83	6A78D	2A9B5	0BAD1	554EA
Participating Address bits	21		x			x			x
	20		x			x	x		
	19		x		x				x
	18		x		x	x	x	x	
	17		x	x					x
	16		x	x		x	x	x	
	15		x	x	x			x	x
	14		x	x	x		x	x	
	13		x	x	x	x			x
	12		x	x	x	x		x	
	11		x	x	x	x	x		
	10		x	x	x	x	x	x	x
	9	x						x	x
	8	x					x		x
	7	x					x	x	
	6	x				x			x
	5	x				x	x		
	4	x			x				x
	3	x			x	x	x	x	

Finally, the 8-bit data ECC and the 8-bit address ECC will be XORed together to calculate the final 8-bit ECC value stored in flash. For example, the final 8-bit ECC result for the data / address in Table 8-2 is 0x9B.

8.5.3.8 *ECC Syndrome Decoding*

As shown in Figure 8-3, during a flash read, when the ECC in the CPU is enabled, the address component is removed from the combined addr+data ECC value stored in flash. After that, the 8-bit ECC value will be XORed with the 8-bit ECC generated by the CPU, creating an 8-bit syndrome. The syndrome is decoded to determine if an error has occurred:

- No error

  This is the normal condition. No further action is taken.

- Single error correction

  The ECC algorithm is able to correct a single bit error. This erroneous bit could be one of the 64 data bits or one of the 8 ECC bits. The syndrome is decoded and generates a signal to invert the failing bit. In this case, the data read by the CPU is automatically corrected but the data in flash will not be corrected unless the application code fixes the erroneous bit by reprogramming the flash.

  There are two types of interrupts the ECC logic can generate when a single-bit error is detected and corrected:

  1. Interrupt on single error interrupt

     The address and failing bit of each correctable error is latched and an interrupt can be generated to inform the program that a failure has occurred.

     When a correctable error is detected and the “Correctable errors interrupt enable (EZFEN)” bit is set, an interrupt is generated and the “SBE FLG” flag is set. The address and error position are frozen from being updated until this flag is cleared by the application software.

     Note: Instructions to clear the status flag should be placed at the end of the interrupt subroutine. This is to avoid the address and position registers latching new values during the interrupt subroutine.

2. Interrupt on error profiling

   When the profiling interrupt is enabled by setting EPEN bit, each time a correctable error is detected by the ECC logic, the correctable error occurrence counter (SEC_OCCUR bits in the FCOR_ERR_CNT register) is incremented. A profiling interrupt is generated when the correctable error occurrence counter is equal to the threshold value. The threshold is a 16-bit user programmable value (SEC_THRESHOLD) stored in the FEDACCTRL2 register. The error occurrence counter is frozen in suspend mode (during JTAG debug mode).

- Non-correctable error detection

  The ECC logic is able to detect double bit errors, but can not correct them. Address errors will also be treated as an un-correctable error. When an un-correctable error is detected, the CPU will assert a multiple bit error signal to the flash wrapper.

  Once the error signal is generated, the un-correctable error address registeriFUNC_ERR_ADD is frozen until it is read by the CPU.

8.5.3.9 JTAG Debug Mode

During JTAG debug mode, reads from the Flash memory are sill corrected by ECC if the ECC logic in the CPU is enabled. If a correctable error is detected then it is corrected but the error interrupt is not generated and the error occurrence counter is not incremented if in error profiling mode. If an un-correctable error is detected then the data is returned without generating an un-correctable error signal.

8.5.4 Data Security

Data security against either accidental or deliberate access by unauthorized agents is built into the flash module. Level 1 security allows each sector to be individually protected from any access other than read.

At power up, all of the sector-enable bits in FBSE are initialized to 0 so that the flash memory location cannot be modified. The sector should only be enabled when erasing or programming flash memory. The sector-enable bits can only be modified in privileged mode **AND** when the PROTL1DIS bit in the FBPROT register is 1. The sector-enable bits of the sectors within the bank selected by bits BANK[2:0] in register FMAC can be modified by writing the corresponding FBSE bit.

8.5.5 Automatic Power-down of Flash Banks

The flash module provides a mechanism to automatically power down flash banks after they have not been accessed for some user programmable time. Special timers automatically sequence the power up and power down of each bank independently of each other. The charge pump module has its own independent power up/down timers as well.

8.5.5.1 Active Grace Period

The active grace period (AGP) can be used to optimize the flash module power consumption vs access time. Faster access times are associated with higher power modes of operation. At one extreme, the power control logic could attempt to reduce power consumption by putting the banks and charge pump into a low-power mode immediately at the end of every flash access. However, if accesses are only a few cycles apart, this can actually increase power consumption versus leaving the flash powered, because the banks and charge pump consume more power during flash startup and access.

The active grace periods (supported for each bank independently in addition to the charge pump module) allow the banks and/or charge pump to be maintained in active mode for a specified period following an access. This is done in anticipation of another read within the AGP time, to allow the subsequent read to have a faster access and spend less time dissipating power than if the bank went into one of the low power modes immediately. If the next access does not occur within the AGP time, the power control logic can automatically put the bank and/or charge pump into a low-power mode to reduce power consumption during long periods of inactivity.

The AGP value is programmed by a set of programmable counters (FBAC and FPAC2) which keep the flash bank or charge pump in active mode until the counter expires, at which time the bank or charge pump reverts to its fallback power mode as defined in the FBFALLBACK and FPAC1 registers. The application software can program the fallback power mode to be standby or sleep mode to reduce power consumption, or program it to be active mode to keep the bank active regardless of counter settings (default). The charge pump AGP counter remains in its initialized state when any one of the banks is active, including the AGP counter of the bank. The charge pump AGP counter begins counting when all banks have become inactive. The application software can also check the current power mode of flash bank and charge pump by reading the FBPRDY register. See register descriptions for detail information.

8.5.5.2 Wakeup of Flash Banks/Pumps

Any access to a flash bank causes the bank and charge pump to go into active mode, regardless of their current state. Also, any erase, program, or validate sector command causes the charge pump to become active.

If the charge pump is in sleep mode when the flash access begins, the power mode control logic automatically sequences the charge pump to standby mode, then to active mode. Also, if any bank is active or in standby mode, the charge pump is active, independent of the charge pump fallback power mode.

The CPU can override the power control functions of the flash module by setting all of the AGP counters to zero. In this case, the power mode control logic still sequences the pump through standby mode automatically if needed, and it activates the pump automatically if any bank is put into any power mode other than sleep mode.

8.6 Control Registers

This section details the flash module registers, summarized in Figure 8-6. 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 0xFFF87000.

Figure 8-6. Flash Control Register Summary

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
0xFFF87000 FRDCNTL Page 288	Reserved															
	Reserved				RWAIT				Reserved			ASWSTEN	Reserved			ENPIPE
0xFFF87004 FSPRD Page 289	Reserved															
	Reserved														RM1	RM0
0xFFF87008 FEDACCTRL1 Page 290	Reserved							SUSP_IGNR	Reserved				EDACMODE			
	Reserved						EZFEN	EPEN	reserved		EOCV	EZCV	EDACEN			
0xFFF8700C FEDACCTRL2 Page 293	Reserved															
	SEC_THRESHOLD															
0xFFF87010 FCOR_ERR_CNT Page 294	Reserved															
	COR_ERR_CNT															
0xFFF87014 FCOR_ERR_ADD Page 295	Reserved					COR_ERR_ADD[26:16]										
	COR_ERR_ADD[15:3]													word offset[2:0]		
0xFFF87018	Reserved															
	Reserved															

Figure 8-6. Flash Control Register Summary (Continued)

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
0xFFF8701C FEDACSTATUS Page 296	Reserved															
	Reserved					ADD PAR ERR	Res	ECC MUL ERR	Reserved						SBE-FLG	ERR PRF FLG
0xFFF87020 FUNC_ERR_ADD Page 298	UNC_ERR_ADD[31:16]															
	UNC_ERR_ADD[15:3]													00		
0xFFF87024 FEDACSDIS Page 299	BankID1_inverse			reserved	SectorID1_inverse				BankID1			reserved	SectorID1			
	BankID0_inverse			reserved	SectorID0_inverse				BankID0			reserved	SectorID0			
0xFFF87028 - 0xFFF8702C	Reserved															
	Reserved															
0xFFF87030 FBPROT Page 300	Reserved															
																PROT L1DIS
0xFFF87034 FBSE Page 301	Reserved															
	BSE[15:0]															
0xFFF87038	Reserved															
	Reserved								Reserved for Flash API							
0xFFF8703C FBAC Page 302	reserved								OTPPROTDIS[7:0]							
	BAGP[7:0]								VREADST[7:0]							

Figure 8-6. Flash Control Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFF87040 FBFALLBACK Page 303	Reserved															
	Reserved								BANKPWR3 [1:0]		BANKPWR2 [1:0]		BANKPWR1 [1:0]		BANKPWR0 [1:0]	
0xFFF87044 FBPRDY Page 304	Reserved															
	PUMP RDY	Reserved											BANKRDY[3:0]			
0xFFF87048 FPAC1 Page 305	reserved					PSLEEP[10:0]										
	Reserved															PUMP PWR
0xFFF8704C FPAC2 Page 306	Reserved															
	PAGP[15:0]															
0xFFF87050 FMAC Page 307	Reserved															
	Reserved													BANK[2:0]		
0xFFF87054	Reserved															
	Reserved for Flash API															
0xFFF87058 - 0xFFF8705C	Reserved															
	Reserved															
0xFFF87060 FEMU_ECC Page 308	Reserved								RD_ECC[7:0]							
	Reserved								EMU_ECC[7:0]							

Figure 8-6. Flash Control Register Summary (Continued)

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
0xFFF87064	Reserved															
	Reserved for Flash API															
0xFFF87068 - 0xFFF87078	Reserved															
	Reserved															
0xFFF8707C FPAR_OVR Page 309	Reserved															
	BUS_PAR_DIS				PAR_OVR_KEY			ADD_INV_PAR	Reserved							
0xFFF87080 - 0xFFF870BC	Reserved															
	Reserved															
0xFFF870C0 FEDACSDIS2 Page 310	BankID3_inverse			reserved	SectorID3_inverse				BankID3			reserved	SectorID3			
	BankID2_inverse			reserved	SectorID2_inverse				BankID2			reserved	SectorID2			

8.6.1 *Flash Option Control Register (FRDCNTL)*

FRDCNTL supports pipeline mode. There is only one FRDCNTL register for the entire F035 Flash.

Figure 8-7. Flash Option Control Register (FRDCNTL) [0xFFF87000]

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
Reserved				RWAIT				Reserved			ASW-STEN	Reserved			EN PIPE
R-0				RWP-0001				R-0			RWP-0	R-0			RWP-0

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-6. Flash Option Control Register (FRDCNTL) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads return zeros and writes have no effect.
11–8	RWAIT	0x0–0xF	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 they are set to 0. **Note:** The required wait states for each HCLK frequency can be found in device datasheet.
7–5	Reserved		Reads return zeros and 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 device datasheet.
3–1	Reserved		Reads return zeros and writes have no effect.
0	ENPIPE		Enable Pipeline Mode
		0	Pipeline mode is disabled.
		1	Pipeline mode is enabled.

8.6.2 Flash Special Read Control Register (FSPRD)

Figure 8-8. Flash Special Read Control Register (FSPRD - 0xFFF87004)

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
Reserved														RM1	RM0
R-0														RWP-0x0	

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-7. Flash Special Read Control Register (FSPRD) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zeros and writes have no effect.
1	RM1		Read Margin 1
		0	Read Margin 1 mode is enabled.
		1	Read Margin 1 mode is enabled.
0	RM0		Read Margin 0
		0	Read Margin 0 mode is disabled.
		1	Read Margin 0 mode is enabled.
			Note: If both RM0 and RM1 are set then Read Margin 0 is taken.

8.6.3 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)

Figure 8-9. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EZFEN	EPEN	Reserved		EOCV	EZCV	EDACEN			
R-0						RWP-0	RWP-0	R-0		RWP-0	RWP-0	RWP-0x5			

-*n* = Value after reset, R=Read, WP=Write in Privilege Mode

This register controls ECC detection.

Table 8-8. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zeros and 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 FWM.
		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		Reads return zeros and writes have no effect.
19–16	EDACMODE		Error Correction Mode.
		0101	During all AXI slave access, flash wrapper will calculate the ECC value from the flash data and send it to the CPU. This is useful while programming the flash and the flash bank contains incomplete words with invalid ECC bytes. **Note:** These bits have no impact on non AXI slave access**.**
		all other values	The flash wrapper will send the ECC bits in the flash bank to the CPU directly after stripping out the address component.

Table 8-8. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
			When this field is “0101” the integrity of the flash wrapper is slightly lowered because during AXI slave access, an error in a stored bit is not reflected in the ECC sent to the CPU.
15–10	Reserved		Reads return zeros and writes have no effect.
9	EZFEN		Correctable errors interrupt enable
		0	Single error interrupt is disabled.
		1	Single error interrupt is enabled.
8	EPEN		Error Profiling Enable.
		0	Error profiling is disabled.
		1	Error profiling is enabled. The correctable error interrupt is generated when number of CPU accesses of correctable bit errors detected and corrected has reached the threshold value defined in the FEDACCTRL2 register
7–6	Reserved		Reads return zeros and writes have no effect.
5	EOCV		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 ones) will generate ECC errors. The error counter for profiling will increment if all ones are detected.
		1	One condition valid is enabled. During AXI slave accesses, reading of an erased location (64 data bits and the corresponding 8 ECC bits are all ones) will NOT generate ECC errors. The error counter for profiling will NOT increment if all ones are detected. **Note:** This bit only applies to AXI slave accesses. This bit has no impact during normal CPU ATCM reads.
4	EZCV		Zero Condition Valid.
		0	Zero condition valid is disabled. Reading of all zeros (64 data bits and the corresponding 8 ECC bits are all zeros) will generate ECC errors. The error counter for profiling will increment if all zeros are detected.
		1	Zero condition valid is enabled. During AXI slave accesses, reading of all zeros (64 data bits and the corresponding 8 ECC bits are all zeros) will NOT generate ECC errors. The error counter for profiling will NOT increment if all zeros are detected. **Note:** This bit only applies to AXI slave accesses. This bit has no impact during normal CPU ATCM reads.

Table 8-8. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Name	Value	Description
3–0	EDACEN		Error Detection and Correction Enable
		0101	CPU single and double error signals are blocked. **Note:** It is NOT recommended to use mode with ECC in CPU enabled. If the ECC in CPU is enabled, the CPU will still check and correct ECC errors, which would cause reading the wrong data and generating abort.
		1101	Error Detection and Correction is enabled, ECC errors generated from the OTP memory regions and disabled sectors are NOT blocked.
		all other values	Error Detection and Correction is enabled, ECC errors generated from the OTP memory regions and disabled sectors are blocked. **Note:** It is recommended writing “1010” to enable EDACEN to guard against soft errors from flipping EDACEN to a disable state.

8.6.4 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2 - 0xFFF8700C)

Figure 8-10. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2 - 0xFFF8700C)

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
SEC_THRESHOLD															
RWP-0															

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-9. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–0	SEC_THRESHOLD		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 zero disables the threshold so that it never triggers the profile interrupt.

8.6.5 Flash Error Correction Counter Register (FCOR_ERR_CNT - 0xFFF87010)

Figure 8-11. Flash Error Correction Counter Register (FCOR_ERR_CNT - 0xFFF87010)

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">COR_ERR_CNT</td></tr>
<tr><td colspan="16">RWP-0</td></tr>
</table>

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-10. Flash Error Correction Counter Register (FCOR_ERR_CNT) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–0	COR_ERR_CNT		Correctable Error Counter This 16 bit counter contains the number of correctable error occurrences. A write to this register with any value will reset the counter to all zeros. The counter resets to 0 when it is greater than or equal to the threshold value and continues to increment if it detects an error again. This counter is frozen in emulation mode. The register only counts when the error profiling bit EPEN in the FEDACCTRL1 register is set to one.

8.6.6 Flash Correctable Error Address (FCOR_ERR_ADD - 0xFFF87014)

Figure 8-12. Flash Correctable Error Address (FCOR_ERR_ADD - 0xFFF87014)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COR_ERR_ADD[15:3]													word offset[2:0]		
R-u													R-0		

-n **=** Value after reset, R=Read, WP=Write in Privilege Mode
-u = unchanged value on internal reset, cleared on power up

This register is not changed after the reset. This register will not get updated when SBE_FLG is ‘1’. The error address is captured during errors when the EZFEN enable bit is set. During error profiling mode when EPEN is set, the error address is **NOT** captured if a correctable error is detected.

Table 8-11. Flash Correctable Error Address (FCOR_ERR_ADD) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zeros and writes have no effect.
26–3	COR_ERR_ADD		Error Address COR_ERR_ADD records the CPU logical address of which a correctable error (ECC single bit error) is detected by the ECC logic in the CPU. This register is frozen from changing during emulation mode.
2–0	word offset	0	The last 3 digit of the correctable error address. Since ECC is checked on 64 bit data, the address captured is aligned to a 64-bit boundary with bit[2:0] tied to 0. These bits always read 0; writes have no effect

8.6.7 Flash Error Status Register (FEDACSTATUS - 0xFFF8701C)

Figure 8-13. Flash Error Status Register (FEDACSTATUS - 0xFFF8701C)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">Reserved</td><td>ADD PAR ERR</td><td>Res</td><td>ECC MUL ERR</td><td colspan="6">Reserved</td><td>SBE-FLG</td><td>ERR PRF FLG</td></tr>
<tr><td colspan="5">R-0</td><td>RCP-u</td><td>R-0</td><td>RCP-u</td><td colspan="6">R-0</td><td>RCP-u</td><td>RCP-u</td></tr>
</table>

-n = Value after reset, R=Read, CP=Clear in Privilege Mode by writing a 1,
-u = unchanged value on internal reset, cleared on power up

All these error status bits can be cleared by writing a one to the bit. Writing a zero has no effect.

These error bits are not set in the emulation mode but they can be cleared in the emulation mode by writing ‘1’s to the bits. By setting the SUSP_IGNR bit to ‘1’ these error bits can be set in suspend mode.

Bits 0 to 1 show correctable errors while bits 8 and 10 show uncorrectable errors.

When the correctable errors are detected, the current address is stored in the FCOR_ERR_ADD register and frozen. The correctable errors in bits 1:0 must be cleared before the end of the interrupt service routine or else the interrupt will re-issue. The FCOR_ERR_ADD registers will not update while SBE FLG (bit 1) is set.

When the uncorrectable errors are detected, the current address is stored in the FUNC_ERR_ADD register and frozen. The FUNC_ERR_ADD will not change again unless it is first unfrozen by being read. Additional uncorrectable errors are blocked from setting additional error bits until the FUNC_ERR_ADD is unfrozen. All the uncorrectable error bits will assert a UERR to the ESM module while unfrozen. Then, the UERR signal is frozen until the FUNC_ERR_ADD is read.

Flash wrapper has higher priority than the CPU to set the error bit. This could cause a condition where the error bit is cleared before it can be read by the CPU. To avoid the freezing of the error bits and error address under this situation, the application should read the FUNC_ERR_ADD to un-freeze the error bits and error address if an interrupt is generated but no error bit is set.

Table 8-12. Flash Error Status Register (FEDACSTATUS) Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		Reads return zeros and writes have no effect.
10	ADD PAR ERR		Address Parity Error. A parity error was detected on the incoming address bus. The full 32 bit address will be stored in FUNC_ERR_ADD.
9	Reserved		Reads return zeros and writes have no effect.

Bit	Name	Value	Description
8	ECC MUL ERR		Multiple bit ECC Status Flag
		0	Un-correctable double bit or multiple bit ECC Error is not detected
		1	Un-correctable double bit or multiple bit ECC Error is detected **Note:** Address error in flash bank is also treated as multiple bit ECC Error.
7–2	Reserved		Reads return zeros and writes have no effect.
1	SBE FLG		Single bit ECC Status Flag. Only apply when EZFEN bit is set.
		0	Single bit ECC error is not detected
		1	Single bit ECC error is detected
0	ERR PRF FLG		ERR_PRF_FLG flag. Only apply when EPEN is set.
		0	Profiling error is not detected.
		1	Profiling error is detected. The number of occurrences of correctable error detected and corrected by ECC logic in the CPU has reached the programmed threshold value stored in FEDACCTRL2 register

8.6.8 Flash Un-correctable Error Address (FUNC_ERR_ADD - 0xFFF87020)

Figure 8-14. Flash Un-correctable Error Address (FUNC_ERR_ADD - 0xFFF87020)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNC_ERR_ADD[31:16]															
R-u															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNC_ERR_ADD[15:3]													00		
R-u													R-0		

-n = Value after reset, R=Read, WP=Write in Privilege Mode
-u = unchanged value on internal reset, cleared on power up

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, this register can be un-frozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at powerup.

Table 8-13. Flash Un-correctable Error Address (FUNC_ERR_ADD) Field Descriptions

Bit	Name	Value	Description
31–3	UNC_ERR_ADD		Un-correctable Error Address UNC_ERR_ADD records the CPU logical address of which an un-correctable 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	word offset	0	The last 3 digit of the address. Since ECC is checked on 64 bit data, the address captured is aligned to a 64-bit boundary with bit[2:0] tied to 0. If the ECC error was due to an address bit then this value will be questionable.

8.6.9 Flash Error Detection Sector Disable (FEDACSDIS - 0xFFF87024)

Figure 8-15. Flash Error Detection Sector Disable (FEDACSDIS - 0xFFF87024)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BankID1_inverse			reserved	SectorID1_inverse				BankID1			reserved	SectorID1			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BankID0_inverse			reserved	SectorID0_inverse				BankID0			reserved	SectorID0			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

-n = Value after reset, R=Read, WP=Write in Privilege Mode

The sectors specified in this register can block single and multiple bit ECC errors detected by the CPU when read from the main address (offset 0). The user must specify the bank/sector to be excluded and its inverse value in this register. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ECC checking. FEDACSDIS2 serves the same purpose. Bank/sector ID is the Bank NO./Sector NO. in the device datasheet.

Table 8-14. Flash Error Detection Sector Disable (FEDACSDIS) Field Descriptions

Bit	Name	Value	Description
31–29	BankID1_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
28	Reserved		Reads return zeros and writes have no effect.
27–24	SectorID1_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
23–21	BankID1		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
20	Reserved		Reads return zeros and writes have no effect.
19–16	Sector1D1		This is the value of the sector ID to be disabled from error detection checking.
15–13	BankID0_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
12	Reserved		Reads return zeros and writes have no effect.
11–8	SectorID0_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
7–5	BankID0		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
4	Reserved		Reads return zeros and writes have no effect.
3–0	Sector1D0		This is the value of the sector ID to be disabled from error detection checking.

8.6.10 Flash Bank Protection Register (FBPROT - 0xFFF87030)

Figure 8-16. Flash Bank Protection Register (FBPROT - 0xFFF87030)

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
															PROTL 1DIS
.R-0															RWP-0

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-15. Flash Bank Protection Register (FBPROT) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zeros and writes have no effect.
0	PROTL1DIS		PROTL1DIS: Level 1 Protection Disabled Level 1 Protection Disable bit. Setting this bit disables protection from writing to the OTPPROTDIS bits as well as the Sector Enable registers FBSE for all banks. Clearing this bit enables protection and disables write access to the OTPPROTDIS register bits and FBSE register.
		0	Level 1 protection is disabled.
		1	Level 1 protection is enabled.

8.6.11 Flash Bank Sector Enable Register (FBSE- 0xFFF87034)

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[2:0] bits of 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 8-17. Flash Bank Sector Enable Register (FBSE - 0x34)

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*
BSE[15:0]															
RWP-0															

-n = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 8-16. Flash Bank Sector Enable Register (FBSE)

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15-0	BSE[15:0]		Bank Sector Enable Each bit corresponds to a flash sector in the bank specified by the FMAC register. This bit can be set only when PROTL1DIS = 1 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.

8.6.12 Flash Bank Access Control Register (FBAC - 0x3C)

Figure 8-18. Flash Bank Access Control Register (FBAC - 0xFFF8703C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved								OTPPROTDIS[7:0]							
R-0								RWP-00000000							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAGP[7:0]								VREADST[7:0]							
RWP-0								RWP-00001111							

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 8-17. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zeros and writes have no effect.
23-16	OTPPROTDIS[7:0]		OTP Sector Protection Disable. Each bit corresponds to a flash bank. This bit can be set only when PROTL1DIS = 1 and in privilege mode.
		0	Programming of the OTP sector is disabled.
		1	Programming of the OTP sector is enabled.
15-8	BAGP[7:0]		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[7:0]		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.

8.6.13 Flash Bank Fallback Power Register (FBFALLBACK - 0xFFF87040)

Figure 8-19. Flash Bank Fallback Power Register (FBFALLBACK - 0xFFF87040)

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
Reserved								BANKPWR3 [1:0]		BANKPWR2 [1:0]		BANKPWR1 [1:0]		BANKPWR0 [1:0]	
RWP-0xF								RWP-11		RWP-11		RWP-11		RWP-11	

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

Table 8-18. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15–8	Reserved		Not used in this device.
7-6	BANKPWR3[1:0]		Bank 3 Fallback Power Mode
		00	Bank sleep mode
		01	Bank standby mode
		10	Reserved
		11	Bank active mode
5-4	BANKPWR2[1:0]		Bank 2 Fallback Power Mode - See BANKPWR3[1:0] for details.
3-2	BANKPWR1[1:0]		Bank 1 Fallback Power Mode - See BANKPWR3[1:0] for details.
1-0	BANKPWR0[1:0]		Bank 0 Fallback Power Mode - See BANKPWR3[1:0] for details.

8.6.14 Flash Bank/Pump Ready Register (FBPRDY - 0xFFF87044)

FBRDY allows the user to determine if the associated bank or the pump is ready for read access.

Figure 8-20. Flash Bank/Pump Ready Register (FBPRDY - 0xFFF87044)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>PUMP RDY</td><td colspan="7">Reserved</td><td colspan="4">Reserved</td><td colspan="4">BANKRDY[3:0]</td></tr>
<tr><td>R-1</td><td colspan="7">R-0</td><td colspan="4">R-0xF</td><td colspan="4">R-0xF</td></tr>
</table>

-*n* = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 8-19. Flash Bank/Pump Ready Register (FBPRDY) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15	PUMPRDY		Pump Ready Pump Ready is a read-only bit which allows software to determine if the pump is ready for flash access before attempting the actual access. If an access is made to a bank when the pump is not ready, wait states are asserted until it becomes ready.
		0	Pump is not ready
		1	Pump is ready, in active power state.
14-8	Reserved		Reads return zeros and writes have no effect.
7-4	Reserved		Reads return Ones and writes have no effect.
3-0	BANKRDY[3:0]		Bank Ready This is a read-only register which allows software to determine if the corresponding bank is ready for flash access before the access is attempted. **Note:** User should wait for both the pump and the targeted bank to be ready before attempting an access.
		0	The corresponding bank is not ready. For example, BANKRDY[1]=0 means BANK1 is not ready for access.
		1	The corresponding bank is ready, in active power mode. For example, BANKRDY[1]=1 means BANK1 is ready for access.

8.6.15 Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048)

Figure 8-21. Flash Pump Access Control Register 1 (FPAC1 - 0xFFF87048)

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*
reserved					PSLEEP[10:0]										
R-0					RWP-0x64										

15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Reserved															PUMP PWR
R-0															RWP-1

-n = Value after reset, R = Read, WP = Write in Privilege Mode

.

Table 8-20. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zeros and writes have no effect.
26-16	PSLEEP[10:0]		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		Reads return zeros and writes have no effect.
0	PUMPPWR		Flash Charge Pump Fallback Power Mode
		0	Sleep (all pump circuits disabled)
		1	Active (all pump circuits active)

8.6.16 Flash Pump Access Control Register 2 (FPAC2 - 0xFFF8704C)

Figure 8-22. Flash Pump Access Control Register 2 (FPAC2 - 0xFFF8704C)

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
PAGP[15:0]															
RWP-0															

-n = Value after reset, R = Read, WP = Write in Privilege Mode

Table 8-21. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15-0	PAGP[15:0]		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 which is a divide by 16 of HCLK.

8.6.17 Flash Module Access Control Register (FMAC - 0xFFF87050)

Figure 8-23. Flash Module Access Control Register (FMAC - 0xFFF87050)

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="13">Reserved</td><td colspan="3">BANK[2:0]</td></tr>
<tr><td colspan="13">R-0</td><td colspan="3">RWP-0</td></tr>
</table>

-n = Value after reset, R = Read, WP = Write in Privilege Mode

Table 8-22. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return zeros and writes have no effect.
2-0	BANK[2:0]		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[2:0] can identify up to 8 flash banks. If less than 8 banks are configured and if BANK[2:0] is selected for an un-implemented bank then the BANK[2:0] will set itself to the highest implemented bank. To determine the number of implemented banks, write “111” to this register and then read it back for the number of implemented banks -1.

8.6.18 Flash Emulation ECC Register (FEMU_ECC - 0xFFF87060)

Figure 8-24. Flash Emulation ECC Register (FEMU_ECC - 0xFFF87060)

31–24	23–16
Reserved	RD_ECC[7:0]
R-0	R-0

15–8	7–0
Reserved	EMU_ECC[7:0]
R-0	RWP-00000011 †

-*n* = Value after reset, R=Read, WP=Write in Privilege Mode; †-see text

Table 8-23. Flash Emulation ECC Register (FEMU_ECC) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Reads return zeros and writes have no effect.
23–17	RD_ECC		This field will contain the ECC value of the address read during an AXI slave access. This allows for reading the ECC values without getting an ECC error from the CPU. Application code can read out the ECC value in this field after reading the mirrored program data flash. Care should be taken to avoid overwriting this value with a DMA action.
15–8	Reserved		Reads return zeros and writes have no effect.
7-0	EMU_ECC		This field will contain the ECC bits written to the flash wrapper module, which is a function of both the address and the data of the write and put into this field. This field is read only and resets to all zeros.

8.6.19 Flash Parity Override (FPAR_OVR - 0xFFF8707C)

Figure 8-25. Flash Parity Override (FPAR_OVR - 0xFFF8707C)

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
BUS_PAR_DIS				PAR_OVR_KEY			ADD_INV_PAR	Reserved							
RWP-0101				RWP-010			RWP-0	R-0							

-n = Value after reset, R=Read, WP=Write in Privilege Mode

Table 8-24. Flash Parity Override (FPAR_OVR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
15 –12	BUS_PAR_DIS		Disable address bus parity key
		1010	The address bus parity checking is disabled.
		Other	Enable the parity checking on the Address bus.
11 –9	PAR_OVR_KEY		Parity overwrite key
		101	The selected ADD_INV_PAR field becomes active
		Other	The module uses the global system parity bit in the system register DEVCR1.
11 –8	ADD_INV_PAR		Address Odd Parity
		1	The incoming address bus will invert system parity bit in the system register DEVCR1 for parity calculations. **Note:** This is only valid when PAR_OVR_KEY is set to '101'.
		0	System parity bit in the system register DEVCR1 is used for parity calculations. This bit is set to the system parity bit value on reset.
7–0	Reserved		Reads return zeros and writes have no effect.

8.6.20 Flash Error Detection Sector Disable (FEDACSDIS2 - 0xFFF870C0)

Figure 8-26. Flash Error Detection Sector Disable (FEDACSDIS2 - 0xFFF870C0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BankID3_inverse			reserved	SectorID3_inverse				BankID3			reserved	SectorID3			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BankID2_inverse			reserved	SectorID2_inverse				BankID2			reserved	SectorID2			
RWP-0			R-0	RWP-0				RWP-0			R-0	RWP-0			

-n = Value after reset, R=Read, WP=Write in Privilege Mode

The sectors specified in this register can block single and multiple bit ECC errors detected by the CPU when read from the main address (offset 0). The user must specify the bank/sector to be excluded and its inverse value in this register. Only when the programmed bank/sector ID value and its calculated inverted value matches the programmed inverse value will the sector selected be excluded from ECC checking errors. FEDACSDIS serves the same purpose.

Table 8-25. Flash Error Detection Sector Disable (FEDACSDIS2) Field Descriptions

Bit	Name	Value	Description
31–29	BankID3_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
28	Reserved		Reads return zeros and writes have no effect.
27–24	SectorID3_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
23–21	BankID3		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
20	Reserved		Reads return zeros and writes have no effect.
19–16	Sector1D3		This is the value of the sector ID to be disabled from error detection checking.
15–13	BankID2_Inverse		This is the inverse value of the bank ID which contains the sector to be disabled from error detection checking.
12	Reserved		Reads return zeros and writes have no effect.
11–8	SectorID2_Inverse		This is the inverse value of the sector ID to be disabled from error detection checking.
7–5	BankID2		This is the value of the bank ID which contains the sector to be disabled from error detection checking.
4	Reserved		Reads return zeros and writes have no effect.
3–0	Sector1D2		This is the value of the sector ID to be disabled from error detection checking.

TEXAS INSTRUMENTS

Chapter 9

SPNU489C-February 2012

CPU Self Test Controller (STC) Module

This document describes the behavior and specification of the CPU SelfTest Controller and briefly present the Logic BIST concept present in the device.

Topic Page

9.1 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.

Software based Selftest program for the cores are available but offers slightly less test coverage. Due to the complexity of the soft cores it is very difficult to achieve the required coverage and also the program size will be much larger. For these complex cores, on-chip logic BIST support for the self test is preferred solution.

The main features of this solution includes:

- Implement Logic BIST controller along with On-chip Self-Test controller for the synthesizable CPU cores which enables to achieve high test coverage
- Ability to divide the complete test run into independent test sets (intervals).
- Capable of running the complete test as well as running few intervals at a time.
- Ability to continue from the last executed interval (test set) as well as ability to restart from the beginning (First test set).
- Complete isolation of the self tested CPU core with 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 (example: DMA access to CPU TCM) will be held till 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 selftest run for debugging purposes
- Supports single CPU as well as dual CPU architecture with two LBIST controllers.
- Generically parameterizable options for the number of shadow scan channels to support different LBIST controller configurations with different shadow scan chains.
- The number of patterns per interval is fixed to 32.
- STC Clock divider support in system module to achieve desired clock rate.

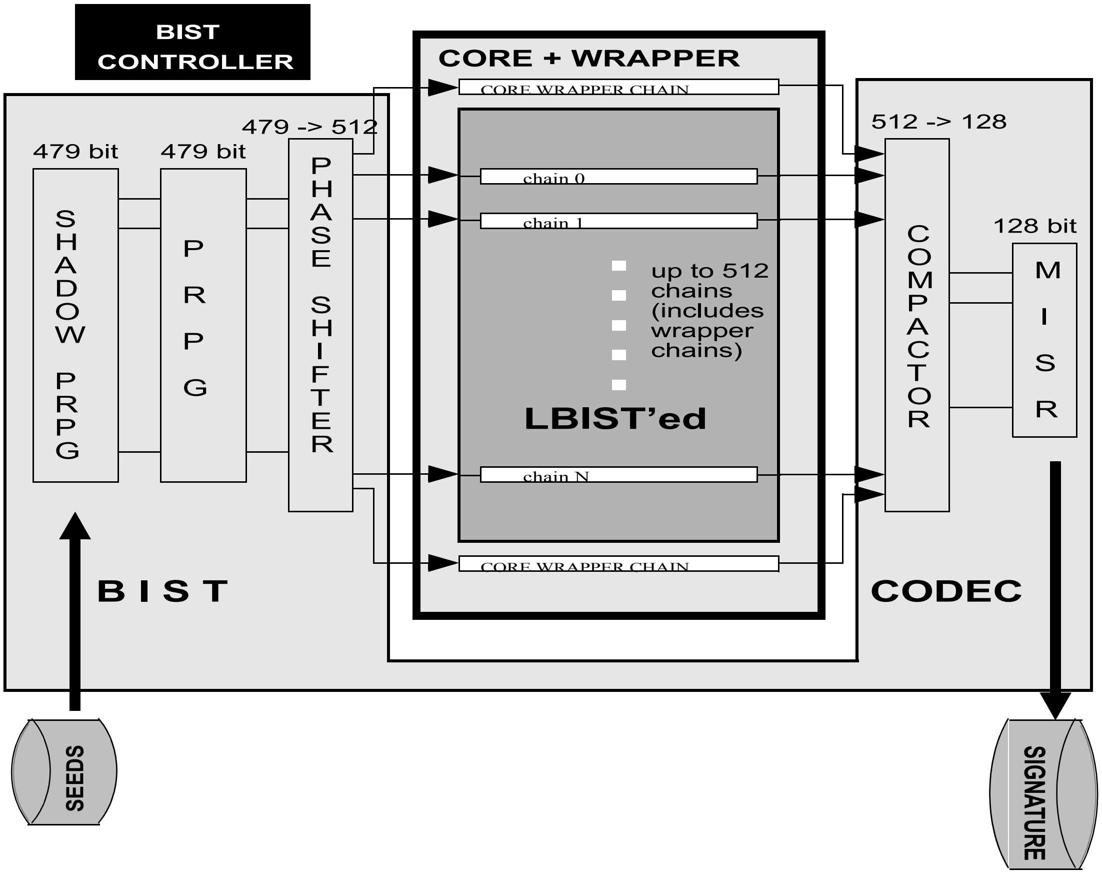
Note: There are minor differences in logic BIST (LBIST) and Deterministic Logic BIST (DBIST) implementations and how different tool vendors name it. This document refers to the DBIST implementation. But the terms LBIST and DBIST are used interchangeably.

9.2 Deterministic Logic BIST concept

Deterministic Logic BIST is a methodology which moves the test pattern generation to on-chip. Logic BIST will be implemented on functional partitions (BISTe'd CORES) that are speed critical and have high gate count. A conceptual diagram of Deterministic Logic BIST implementation is shown in Figure 9-1. The architecture contains a Linear feedback shift register (LFSR) which is initialized to a particular value or seeds. The DBIST tools uses deterministic test patterns to produce the seeds or initial values. This functions as a pseudo-random pattern generator (PRPG). The test patterns are compressed in PRPG seeds which are applied through the PRPG shadow registers. The patterns pass through a phase shifter (LFSR) which basically converts the patterns generated by the PRPG into a 2-dimensional arrays of parallel scan chains which are scanned into the design under test (BISTe'd CORE). The idea is to have as many parallel scan chains as possible which are kept short which increases the controllability and observability of the design with the minimum set of patterns. The Deterministic Logic BIST tool supports up to 512 parallel scan chains. The captured data from the LBIST core are loaded in parallel into signature analyzer which compacts the scan outputs into a signature.

The signature analyzer is a modified LFSR know as multiple input signature register (MISR), A compactor is a combinational logic block which reduces the scan chain outputs coming from the Logic BIST core so that the MISR can be smaller. The shadow PRPG, PRPG, Phase shifter, compactor and MISR together constitute an LBIST CODEC. THE LBIST CODEC in turn is controlled by an LBIST controller. A wrapper is also generated around the BIST'ed CORE this contains all the Input bounding cells suppress X propagation to the LBIST'ed core so that the MISR can generate a proper signature.

Figure 9-1. LBIST conceptual diagram



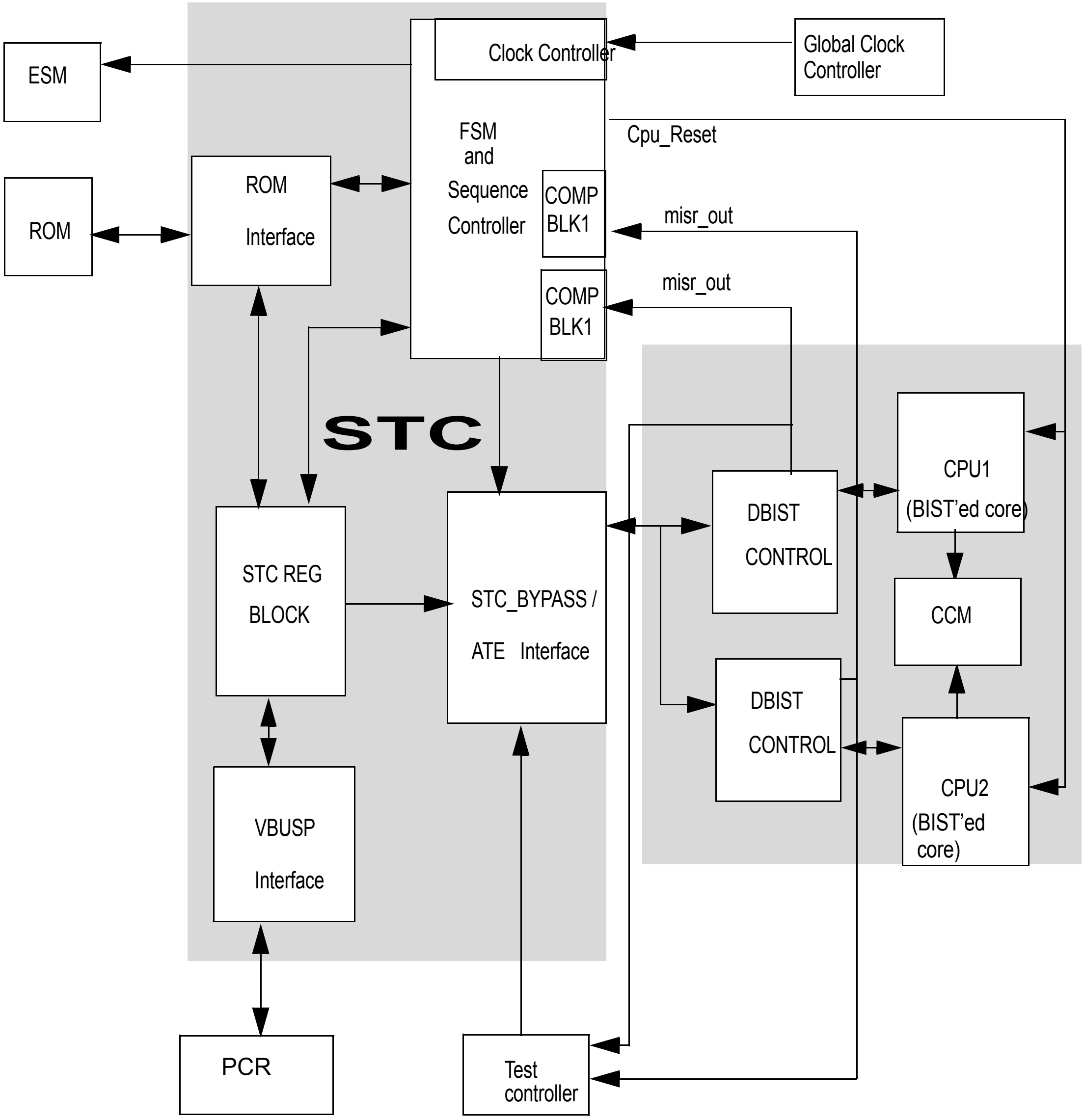
9.3 STC Block diagram

STC module provides an interface to the LBIST controller implemented on the core.

The CPU STC is composed of 5 blocks of logic.

- ROM Interface
- FSM and Sequence control
- Register file
- Peripheral Bus Interface (VBUSP Interface)
- STC Bypass/ATE interface

Figure 9-2. Block diagram for Dual CPU architecture with CCM



9.4 Module Description

9.4.1 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 is stored in ROM.

9.4.1.1 FSM and Sequence control

This block generates the signals and data to LBIST controller based on the seed, test_type, scan chain depth. The sequence of operation per interval are defined in the Flow chart in Section 9.5. The timing protocol and flow diagrams are covered in Section 9.11.

9.4.1.2 Clock Control

The CLOCK CNTRL sub-block is handles the clock selection and clock generation for ROM, LBIST controller as well as BIST'ed core (CPU) clocks.

9.4.2 Register Block

9.4.2.1 Control Registers

This block handles the control of the Self Test Controller. It includes control registers to specify the test model type like Stuck-at or transition delay methodologies. This also controls the reseeding (Reloading the existing seed of the PRPG) in the LBIST controller.

This also handles the buffering the LBIST seed data and pipelines according to the number of shadow scan chains.

9.4.2.2 Configuration Registers

This contains various configuration and status registers which provide the result of selftest run. These registers are memory mapped and accessible through Peripheral Bus (VBUSP) interface.

9.4.3 STC Bypass / ATE Interface

This is a production test interface. This module allows bypassing the self test FSM. The LBIST signal interface are brought out directly to the module ports and these are accessible to ATE (tester) at the device level. The intent on the block is to provide capability for fault isolation for parts failing the CPU self test run.

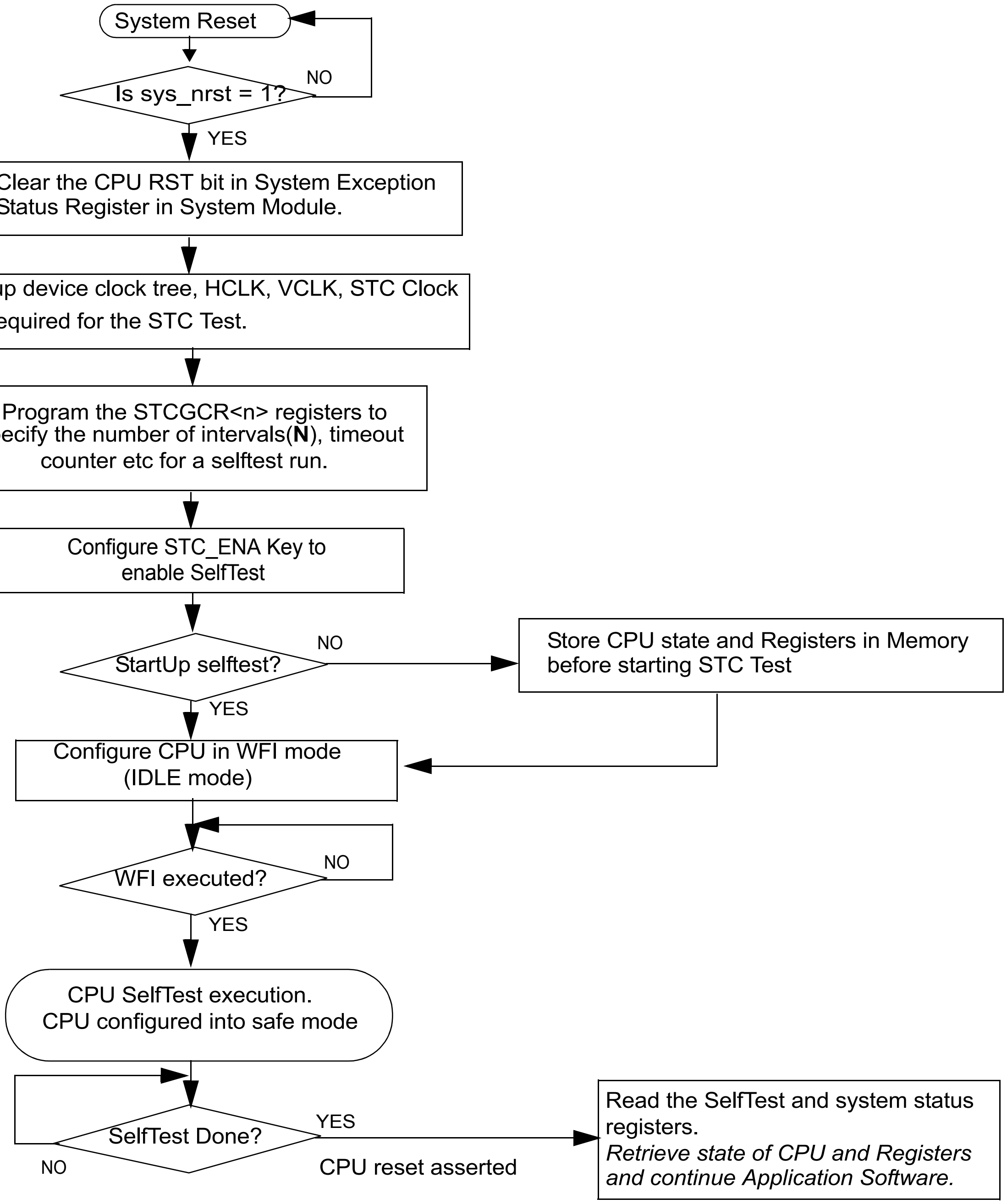
This modules receives two sets of LBIST signals; one from device test controller and another similar set from self test FSM (test sequencer). The selftest enable key is used to select one of the above mentioned datapath to the CPU LBIST controller.

9.4.4 Peripheral Bus (VBUSP) Interface

STC control registers are accessed through peripheral (VBUSP) interface. During application programming, configuration registers are programmed through the peripheral interface, to enable and run the self test controller.

9.5 Application Self Test Flow Chart

Figure 9-3. Application Self Test Flow Chart


System Reset
Is sys_nrst = 1?
NO
YES
Clear the CPU RST bit in System Exception Status Register in System Module.
Setup device clock tree, HCLK, VCLK, STC Clock as required for the STC Test.
Program the STCGCR<n> registers to specify the number of intervals(**N**), timeout counter etc for a selftest run.
Configure STC_ENA Key to enable SelfTest
StartUp selftest?
NO
Store CPU state and Registers in Memory before starting STC Test
YES
Configure CPU in WFI mode (IDLE mode)
WFI executed?
NO
YES
CPU SelfTest execution. CPU configured into safe mode
SelfTest Done?
YES
NO
CPU reset asserted
Read the SelfTest and system status registers.
Retrieve state of CPU and Registers and continue Application Software.

STC generates a CPU reset after completion of the Test. Before initiating the self test user should take backup of the registers that are used in the application that get reset by CPU reset and restore after the self test.

Following are some of the Registers that has to be backuped before and restored after Self Test.

1. All 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 etc.
3. CP13 Coprocessor Registers - FPU configuration registers, General Purpose Registers.
4. Hardware Break Point and watch point registers like BVR, BSR, WVR, WSR etc.

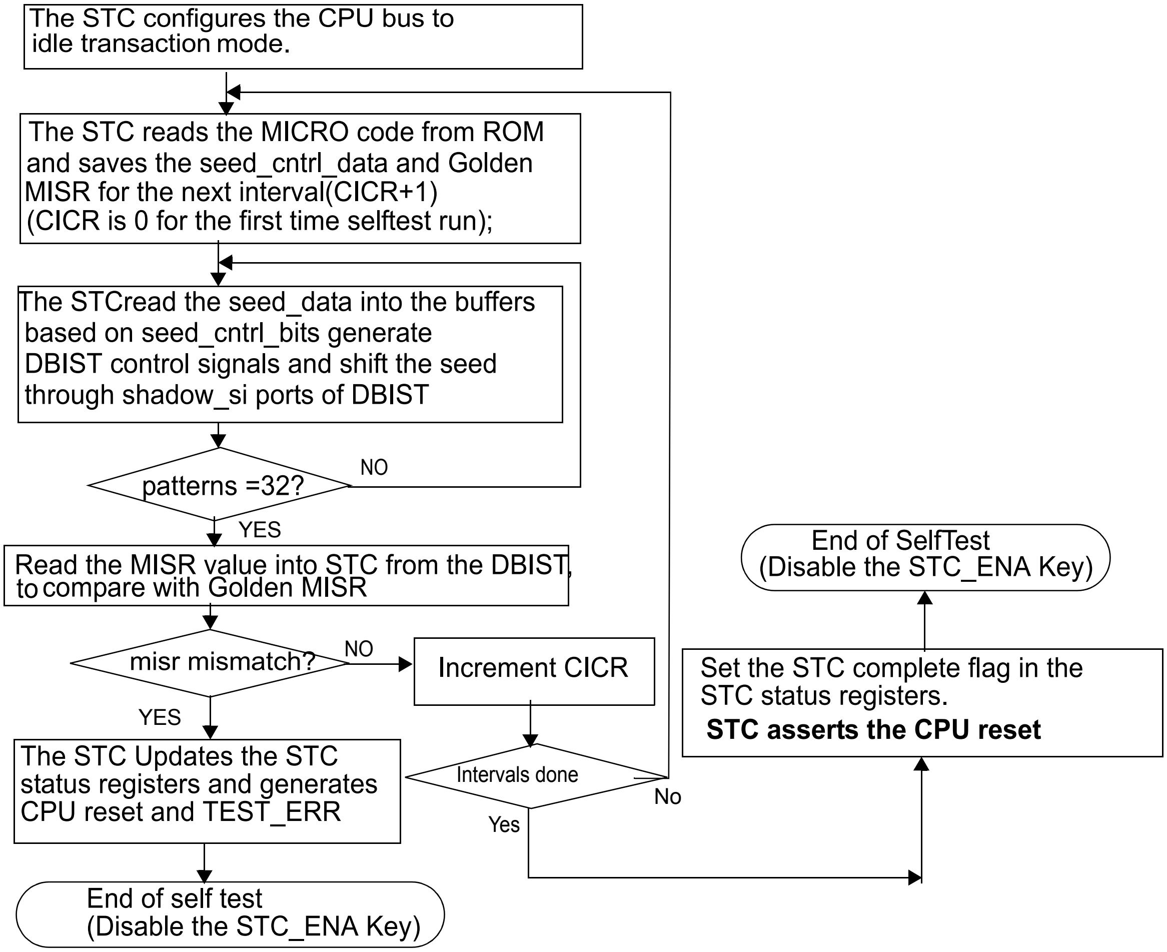
For more info on the CPU register description reset please refer to the *Cortex-R4 Technical Reference Manual version r1p3.*

Note:

Check all reset source flags after running CPU selftest. If a flag in addition to CPU reset is set , clear CPU reset flag and service the other reset source accordingly.

9.6 SelfTest Execution Flow

Figure 9-4. SelfTest Execution Flow


The STC configures the CPU bus to idle transaction mode.
The STC reads the MICRO code from ROM and saves the seed_cntrl_data and Golden MISR for the next interval(CICR+1) (CICR is 0 for the first time selftest run);
The STCread the seed_data into the buffers based on seed_cntrl_bits generate DBIST control signals and shift the seed through shadow_si ports of DBIST
patterns =32?
NO
YES
Read the MISR value into STC from the DBIST, to compare with Golden MISR
misr mismatch?
NO
YES
Increment CICR
Intervals done
No
Yes
The STC Updates the STC status registers and generates CPU reset and TEST_ERR
End of self test (Disable the STC_ENA Key)
Set the STC complete flag in the STC status registers.
STC asserts the CPU reset
End of SelfTest (Disable the STC_ENA Key)

9.7 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 DBIST controller is shifted into the STC. This is compared with the golden MISR value stored in the ROM.

At the end of CPU self test, the STC controller updates the status flags in Global Status Register (STCGSTAT) and resets the CPU. In case of a MISR mismatch or a test timeout, error signals are generated to ESM module. TEST_ERR signal is asserted, when a MISR miscompare occurs during the self test. TIMEOUT_ERR is asserted when a time out occurs during the self test, meaning the test could not complete within the time specified in Timeout counter Preload register STCTPR. However at the device level, these two errors may be combined and mapped to a single ESM channel.

9.8 STC Test Coverage and Duration

The test coverage and number of test execution cycles (STCCLK) for each test interval is shown in Table 9-1.

Table 9-1. STC Test Coverage and Duration

Intervals	Test Coverage	Test Cycle
0	0	0
1	57,14	1555
2	65,82	3108
3	70,56	4661
4	73,56	6214
5	76,06	7767
6	78,07	9320
7	79,62	10873
8	80,92	12426
9	82,1	13979
10	82,94	15532
11	83,76	17085
12	84,51	18638
13	85,12	20191
14	85,62	21744
15	86,19	23297
16	86,56	24850
17	86,97	26403
18	87,33	27956
19	87,67	29509
20	88,01	31062
21	88,31	32615
22	88,58	34168
23	88,87	35721
24	89,11	37274
25	89,34	38827
26	89,59	40380
27	89,82	41933
28	90,05	43486
29	90,26	45039
30	90,46	46592
31	90,64	48145
32	90,84	49698

Table 9-2 gives the Typical STC execution times for 32 intervals at different clock rates.

Table 9-2. Typical STC Execution Times

Number of Intervals	@ HCLK = 160 MHz
	VCLK = 80 MHz
	STCCLK = 53.3 MHz
		@ HCLK = 100 MHz
		VCLK = 100 MHz
		STCCLK = 50MHz
			@ HCLK = 133 MHz
			VCLK = 66.5 MHz
			STCCLK = 44.33MHz

32	0.926 msec	0.989 msec	1.11 msec

9.9 STC Control Registers

STC control registers are accessed through Peripheral Bus (VBUSP) interface. Read and write access in 8,16 and 32 bit are supported. The base address for the control registers is 0xFFFF E600.

All reserved bits are read as zeros. Write to reserved fields has no effect.

> **Note:**
> In suspend mode all register write accesses to the privilege mode only write registers can also be performed in user module.

Table 9-3. 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
0x00 STCGCR0 Page 324	INTCOUNT															
	Reserved															RS_CNT
0x04 STCGCR Page 325	Reserved															
	Reserved												STC_ENA			
0x08 STCTPR Page 326	RTOD[31:16]															
	RTOD[15:0]															
0x0C STC_CADDR Page 327	ADDR[31:16]															
	ADDR[15:0]															
0x10 STCCICR Page 328	Reserved															
	N[15:0]															
0x14 STCGSTAT Page 329	Reserved															
	Reserved														TEST_FAIL	TEST_DONE

Table 9-3. Registers (Continued)

0x18 STCFSTAT Page 330	Reserved			
	Reserved	TO_ERR	CPU2_FAIL	CPU1_FAIL
0x2C CPU1_CURMISR3 Page 331	MISR[31:16]			
	MISR[15:0]			
0x30 CPU1_CURMISR2 Page 331	MISR[63:48]			
	MISR[47:32]			
0x34 CPU1_CURMISR1 Page 331	MISR[95:80]			
	MISR[79:64]			
0x38 CPU1_CURMISR0 Page 331	MISR[127:112]			
	MISR[111:96]			
0x3C CPU2_CURMISR3 Page 333	MISR[31:16]			
	MISR[15:0]			
0x40 CPU2_CURMISR2 Page 333	MISR[63:48]			
	MISR[47:32]			
0x44 CPU2_CURMISR1 Page 333	MISR[95:80]			
	MISR[79:64]			

Table 9-3. Registers (Continued)

0x48 CPU2_CURMISR0 Page 333	MISR[127:112]
	MISR[111:96]

9.9.1 STC global control register0 (STCGCR0)

This register is described in Figure 9-5 and Table 9-4. The offset address is 0x00.

Figure 9-5. STC global control register0 (STCGCR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTCOUNT															
RWP -1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RS_CNT
RWP -1															RWP -0

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 9-4. STC global control register0 (STCGCR0) Field Descriptions

Bit	Name	Value	Description
31–16	INTCOUNT		Number of intervals of selftest run This register specifies the number of intervals to run for the selftest run. This correspond to the number of intervals to be run from the value reflected in the current interval counter.
15-1	Reserved		Reads return 0 and writes have no effect.
0	RS_CNT		Restart or Continue This bit specifies the selftest controller whether to continue the run from next interval onwards or to restart from interval 0. This bit gets reset after the completion of selftest run.
		0	Continue STC run from previous interval.
		1	Restart STC run from interval 0.

Note:
On a powerup reset or system reset this register gets reset to its default values.

9.9.2 STC Global Control Register1 (STCGCR1)

This register is described in Figure 9-6 and Table 9-5. The offset address is 0x04.

Figure 9-6. STCGCR1 (STC Global Control Register1)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												STC_ENA			
RWP -0												RWP -0	RWP -1	RWP -0	RWP -1

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset) or System reset

Table 9-5. STCGCR1 (STC Global Control Register1) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect.
3-1	STC_ENA		Self test run enable key
		1010	Self test run enabled.
		all others	Self test run disabled

Note:

On a powerup reset or system reset this register resets to its default values. Also this register automatically resets to its default values at the completion of a self test run.

9.9.3 Self Test Run Timeout Counter Preload Register (STCTPR)

This register is described in Figure 9-7 and Table 9-6. The offset address is 0x08.

Figure 9-7. Self Test Run Timeout Counter Preload Register (STCTPR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTOD[31:16]															
RWP-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTOD[15:0]															
RWP -1															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset or system reset)

Table 9-6. Self Test Run Timeout Counter Preload Register (STCTPR)

Bit	Name	Value	Description
31-0	RTOD		Self Test timeout count preload This register contains the total number of VBUS clock cycles it will take before an self test timeout error (TIMEOUT_ERR) will be triggered after the initiation of the self test run. This is a fail safe feature to not hang-up the system on account of any run away self test issues, The above 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. **Note: This register gets reset to its default value with Power on or system reset assertion.**

9.9.4 STC Current ROM Address Register (STC_CADDR)

This register is described in Figure 9-8 and Table 9-7. The offset address is 0x0C.

Figure 9-8. STC Current ROM Address Register (STC_CADDR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR[31:16]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR15:0]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST (power on reset) or system reset

Table 9-7. STC Current ROM Address Register (STC_CADDR) Field Descriptions

Bit	Name	Value	Description
31–0	ADDR		Current ROM Address This register reflects the current address ROM address (for micro code load) which is the current value of the STC program counter.

Note:
When the global configuration register restart bit STCGCR0[0] is set to a 1 on the start of a self test run or on a powerup reset or system reset this register resets to all zeroes.

9.9.5 STC Current Interval Count Register (STCCICR)

This register is described in Figure 9-9 and Table 9-8. The offset address is 0x10.

Figure 9-9. STC Current Interval Count Register (STCCICR)

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
N[15:0]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 9-8. STC Current Interval Count Register (STCCICR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect.
15-0	N		Interval Number This specifies the Last executed Interval number.

Note:
When the global configuration register restart bit STCGCR0[0] is set to a 1 or on a powerup reset the Current interval counter resets to default value else it always maintains its value.

9.9.6 SelfTest Global Status Register (STCGSTAT)

This register is described in Figure 9-10 and Table 9-9. The offset address is 0x14.

Figure 9-10. SelfTest Global Status Register (STCGSTAT)

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
Reserved														TEST_FAIL	TEST_DONE
R-0														RWP-0	RWP-0

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 9-9. SelfTest Global Status Register (STCGSTAT) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and 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 **Note:** The above 2 status bits can be cleared to their default values on a write 1 to the above bits. Additionally when the STC_ENA Key is enabled from a non 1010 value to a 1010 value the above 2 status flags get cleared to their default values. 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. An reset is generated to the CPU on which the STC run is being performed when TEST_DONE goes high (The test is completed). This register gets reset to its default value with Power on reset assertion.

9.9.7 SelfTest Fail Status Register (STCFSTAT)

This register is described in Figure 9-11 and Table 9-10. The offset address is 0x18.

Figure 9-11. SelfTest Fail Status Register (STCFSTAT)

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
Reserved													TO_ERR	CPU2_FAIL	CPU1_FAIL
R -0													RWP-0	RWP-0	RWP-0

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after nPORST or system reset

Table 9-10. SelfTest Fail Status Register (STCFSTAT) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TO_ERR		Timeout Error
		0	No time out error occurred
		1	SelfTest 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

Note:

The above 3 status bits can be cleared to their default values on a write 1 to the above bits. Additionally when the STC_ENA Key in STCGCR1 is enabled from a non 1010 value to a 1010 value the above 3 status flags get cleared to their default values. This register gets reset to its default value with Power on reset assertion.

9.9.8 CPU1 Current MISR Register (CPU1_CURMISR[3:0])

This register is described in Figure 9-12 through Figure 9-15 and Table 9-11. The offset address for CPU1_CURMISR3 is 0x2C.

Figure 9-12. CPU1 Current MISR Register (CPU1_CURMISR3)

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

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR2 is 0x30.

Figure 9-13. CPU1 Current MISR Register (CPU1_CURMISR2)

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*
MISR[63:48]															
R -0															
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
MISR[47:32]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR1 is 0x34.

Figure 9-14. CPU1 Current MISR Register (CPU1_CURMISR1)

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*
MISR[95:80]															
R -0															
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
MISR[79:64]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU1_CURMISR0 is 0x38.

Figure 9-15. CPU1 Current MISR Register (CPU1_CURMISR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[127:112]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISR[111:96]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

Table 9-11. CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions

Bit	Name	Value	Description
127-0	MISR		MISR data from CPU1 This register contains the MISR data from the CPU1 for the current interval. This value will be compared with the GOLDEN MISR value copied from ROM.

Note:
This register gets reset to its default value with Power on or system reset assertion.

9.9.9 *CPU2_CURMISR[3:0] (CPU2 Current MISR Register)*

This register is described in Figure 9-16 through Figure 9-19 and Table 9-12. The offset address for CPU2_CURMISR3 is 0x3C.

Figure 9-16. CPU2 Current MISR Register (CPU2_CURMISR3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[31:16]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISR[15:0]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU2_CURMISR2 is 0x40.

Figure 9-17. CPU2 Current MISR Register (CPU2_CURMISR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[63:48]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISR[47:32]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU2_CURMISR1 is 0x44.

Figure 9-18. CPU2 Current MISR Register (CPU2_CURMISR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[95:80]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISR[79:64]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

The offset address for CPU2_CURMISR0 is 0x48.

Figure 9-19. CPU2 Current MISR Register (CPU2_CURMISR0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISR[127:112]															
R -0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISR[111:96]															
R -0															

U = Undefined; R = Read, WP = Write in privilege mode,-*n* = Value after reset

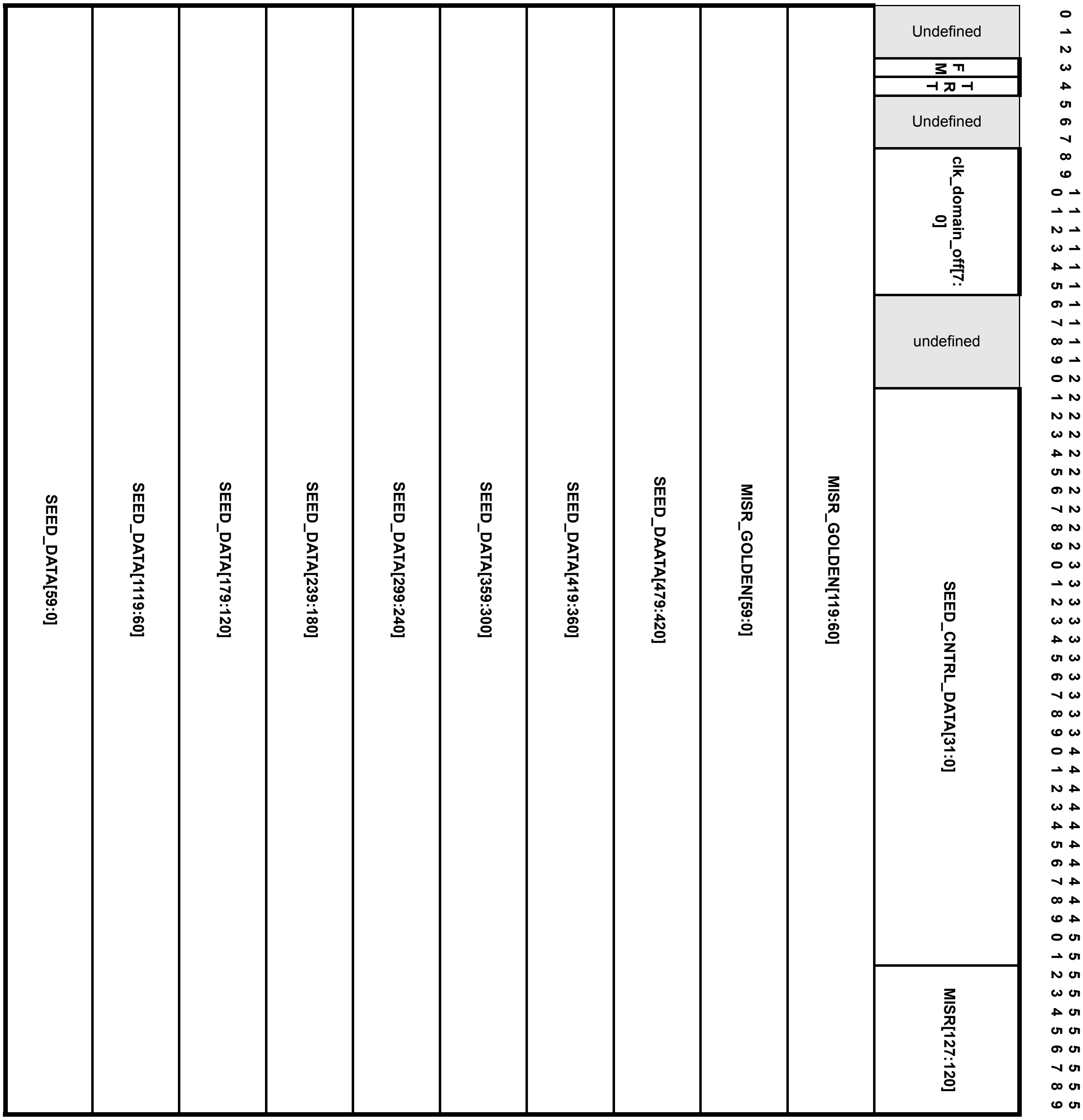
Table 9-12. CPU2 Current MISR Register (CPU2_CURMISR[3:0]) Field Descriptions

Bit	Name	Value	Description
127-0	MISR		MISR data from CPU2 This register contains the MISR data from the CPU2 for the current interval. This value will be compared with the GOLDEN MISR value copied from ROM.

Note:
This register gets reset to its default value with Power on or system reset assertion.

9.10 ROM Organization

Figure 9-20. ROM Organization for a Interval (Their might be a maximum of 32 Seed_datas/interval)



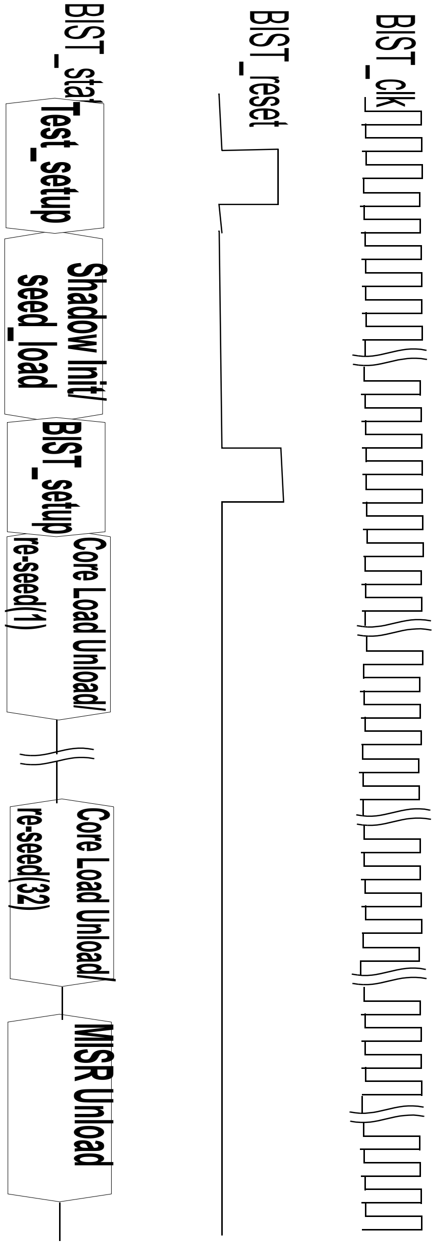
The ROM contains the data to be processed by STC for the self test run. This includes the SEED_DATA, Golden MISR compare and the interval specific configuration data for the STC and the DBIST controller. The ROM space is divided into chunks and each chunk contains the data corresponding to one DBIST interval. The size required for each interval may vary depending on the number of seeds required for that particular interval. The first selftest run starts reading the ROM from address 0. Then STC continues to read the ROM till it finishes the particular selftest of ‘N’ intervals. STC retains its ROM address (STC_CADDR) for the next selftest run unless there is power on reset or an self test restart.

Golden MISR contains the Golden signature data of the current Interval. This value is used to compare with actual MISR value to generate the pass/fail information of the interval.

9.11 Timing Diagrams

9.11.1 BIST Operation: For a self test run for One Interval

Figure 9-21. BIST Operation



The above diagram shows the protocol for a self test run with one interval:

1. Test setup procedure: Once the self test run is initiated and the interval configuration and MISR values are stored in the STC an reset is generated for 2 controller clock cycles to clear the scan_en counter of the BIST controller to its reset state.
2. Shadow Init seed load procedure: The shadow PRPG of the BIST controller is loaded with its initial seed value by the STC from the ROM.
   The seed in the shadow PRPG is loaded into the PRPG at the end of this procedure

3. BIST setup procedure: An reset is generated to the BIST controller to clear the values of the BIST controllers MISR
4. Core Load_unload procedure: While the PRPG is generating the scan patterns to the BISTe'd core the next seed value is loaded into the shadow PRPG (if re-seed = 1) else the shadow PRPG maintains its previous value. Step 4 is repeated for 32 times which constitutes an interval.
5. The BIST controller scan's out the MISR value to the STC on which it is compared with its golden value to indicate the pass or fail of an interval.

Clock considerations:

1. During a pattern shift (At-speed and Stuck-At) the slowest clock of the BISTe'd core is activated as the shift clock for all the clock domains in the BISTe'd core.
2. For an Stuck-At interval the slowest clock of the BISTe'd core is the source for all the functional clock domains of the BISTe'd core during capture.
3. Captures for At-speed interval are done per individual BISTe'd core domain clock during which all other clock domains are gated off.

These GTM interface signals are muxed with the STC FSM generated respective signals in the Bypass Block.

9.12 STC Configuration Example

The following examples assumes that the PLL is locked and selected as clock source with HCLK = 160MHz and VCLK = 80MHz

9.12.1 Example 1 : Self test run for 32 Interval

This example explains the configurations for running STC Test for maximum Test Intervals 32

1. Maximum STC clock rate support at 160MHz HCLK is 53.33 MHz. Divide HCLK by 3 to achieve this clock rate. STCCLKDIV[26:24] register in the secondary system module frame at location 0xFFFF E108 is used.
   STCCLKDIV[26:24] = 2
2. Clear CPU RST status bit in System Exception Status Register in system Module.
   SYSESR[5] = 1
3. Configure the Test Interval count in STC module
   STCGCR0[31:16] = 32
4. Configure SelfTest Run Time out counter preload Register.
   STCTPR[31:0] = 0xFFFFFFFF
5. Enable CPU Self Test
   STCGCR1[3:0]= 0xA;
6. Take Backup of CPU state and configuration registers that gets reset on CPU reset.
7. Configure CPU in Idle Mode by executing CPU Idle Instruction
   asm(“ WFI”)
8. On CPU reset verify CPU RST status bit in 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 TEST_DONE TEST_DONE bit before evaluating TEST_FAIL bit.

   If TEST_DONE =0 the Self test is not completed. Restart the STC test.

   If (TEST_DONE =1 and TEST_FAIL = 1) the Self test is completed and Failed.

   – Read 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, configurations registers and continue application software.

Asynchronous External Memory Interface (EMIF)

This document describes the operation of the asynchronous external memory interface (EMIF) in the TMS570LS20x/10x Safety MCUs device..

Topic

Page

10.1 Introduction

This document describes the operation of the asynchronous external memory interface (EMIF) in the TMS570LS20x/10x Safety MCUs device.

10.1.1 Purpose of the Peripheral

The purpose of this EMIF is to provide a means to connect to an asynchronous SRAM device. Section 10.3 contains examples of operating the EMIF in this configuration

10.1.2 Features

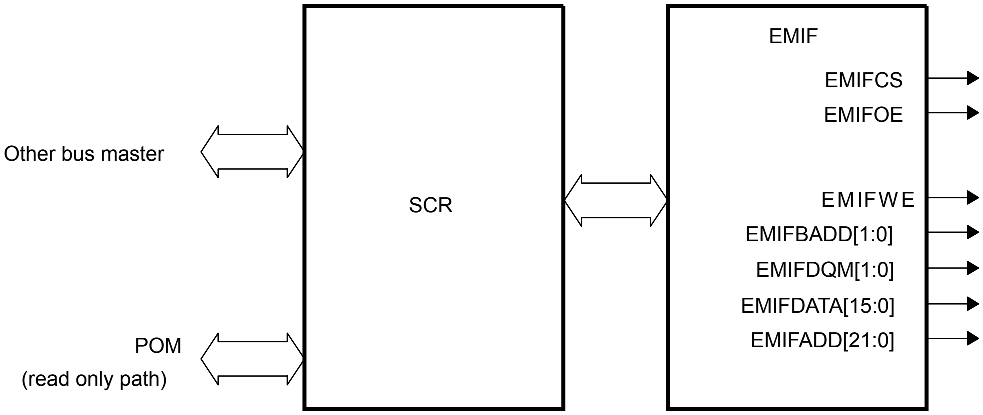
The EMIF includes many features to enhance the ease and flexibility of connecting to external asynchronous devices. The EMIF features includes support for:

- 4 addressable chip select spaces of up to 32MB each
- 16-bit data bus width
- Programmable cycle timings such as setup, strobe, and hold times as well as turnaround time
- Select strobe mode
- Extended Wait mode
- Data bus parking
- Little-endian operating mode

10.1.3 Functional Block Diagram

Figure 10-1 illustrates the connections between the EMIF and its internal requesters, along with the external EMIF pins. Section 10.2.2 contains a description of the entities internal to the device that can send requests to the EMIF, along with their prioritization. Section 10.2.3 describes the EMIF's external pins and summarizes their purpose when interfacing with SDRAM and asynchronous devices.

Figure 10-1. EMIF Functional Block Diagram



10.2 Peripheral Architecture

This section provides details about the architecture and operation of the EMIF.

10.2.1 Clock Control

The EMIF's internal clock is sourced from VCLK_P.

10.2.2 EMIF Requests

The on chip master can access external memory by EMIF through two separate data paths. The POM (Parameter Overlay Memory) module can issue only read VBUSP transactions to EMIF. The other master (CPU, etc.) can issue either read or write VBUSP transactions to EMIF to interface to external memory. The other master can also read or write EMIF configuration registers. There is one SCRP (switch central resource VBUSP) which has a fixed priority scheme. The POM module will have the highest priority.

10.2.3 Signal Descriptions

Table 10-1 describes the function of each of the EMIF pins.

Table 10-1. EMIF Pins

Pins(s)	I/O	Description
EM_ A[21:0]	O	EMIF address bus. These pins are used in conjunction with the EM_BA pins to form the address that is sent to the device.
EM_BA[1:0]	O	EMIF bank address. These pins are used in conjunction with the EM_A pins to form the address that is sent to the device.
EM_CS[3:0]	O	Active-low chip enable pin for asynchronous devices. These pins are meant to be connected to the chip-select pin of the attached asynchronous device.
EM_D[15:0]	I/O	EMIF data bus.
EM_RW	O	Read/Write select pin. This pin is high for the duration of an asynchronous read access cycle and low for the duration of an asynchronous write cycle.
EM_OE	O	Active-low pin enable for asynchronous devices. This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EM_DQM[1:0]	I/O	Byte High (bit 1) and Byte Low (bit 0) enable
EM_WE	O	Active-low write enable. This pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle.

10.2.4 Pin Multiplexing

The EMIF does not share pins with any other IP.

10.2.5 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including Flash and ASRAM. It can be operated in two major modes:

- Normal mode
- Select Strobe (SS) mode

The behavior of the EM_CS signal is the single difference between Normal mode and Select Strobe mode (see Table 10-2). In Normal mode, the EM_CS signal becomes active at the beginning of the setup period and remains active for the duration of the transfer. In Select Strobe mode, the EM_CS signal functions as a strobe signal, active only during the strobe period of an access.

Table 10-2. Behavior of $\overline{\text{EM\_CS}}$ Signal Between Normal Mode and Select Strobe Mode

Mode	Operation of EM_CS[3:0]
Normal	Active during the entire asynchronous access cycle
Select Strobe	Active only during the strobe period of an access cycle

10.2.5.1 Interfacing to Asynchronous Memory

Figure 10-2 shows the EMIF's external pins used in interfacing with an asynchronous device. Of special note is the connection between the EMIF and the external device's address bus. Because the device uses 32-bit data paths internally, the EMIF address pin EM_A[0] always provides the least-significant bit of a 32-bit word address.

When interfacing to an 8-bit asynchronous device, the EM_BA[1] and EM_BA[0] pins are used to provide the least-significant bits of the byte address.

Figure 10-3 shows the mapping between the EMIF and the connected device's data and address pins.

Figure 10-2. EMIF Asynchronous Interface

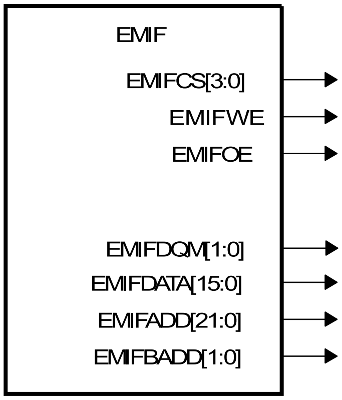
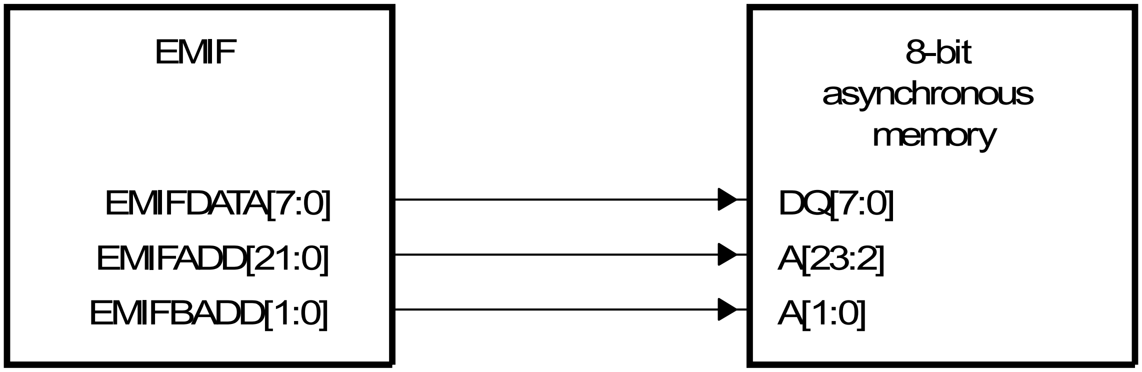


Figure 10-3. EMIF to 8-Bit Memory Interface



10.2.5.2 Programmable Asynchronous Parameters

The EMIF allows a high degree of programmability for shaping asynchronous accesses. The programmable parameters are:

- Setup: The time between the beginning of a memory cycle (address valid) and the activation of the output enable or write enable strobe
- Strobe: The time between the activation and deactivation of output enable or write enable strobe.
- Hold: The time between the deactivation of output enable or write enable strobe and the end of the cycle, which may be indicated by an address change or the deactivation of the EM_CS signal.

Separate parameters are provided for read and write cycles. Each parameter is programmed in terms of EMIF clock cycles.

10.2.5.3 Configuring the EMIF for Asynchronous Accesses

The operation of the EMIF's asynchronous interface can be configured by programming the appropriate memory-mapped registers. The reset value and bit position for each register field can be found in Section 4. The following tables list the programmable register fields and describe the purpose of each field. These registers should not be programmed while an asynchronous access is in progress. The transfer following a write to these registers will use the new configuration.

Table 10-3 describes the asynchronous configuration register (AnCR). There are four AnCRs. Each chip select space has a dedicated AnCR. This allows each chip select space to be programmed independently to interface to different asynchronous memory types.

Table 10-3. Description of the Asynchronous Configuration Register (AnCR)

Parameter	Description
SS	Select Strobe mode. This bit selects the EMIF's mode of operation in the following way: SS = 0h selects Normal mode. EM_CS is active for duration of access. SS = 1h selects Select Strobe mode. EM_CS acts as a strobe.
W_SETUP/R_SETUP	Read/Write setup widths. These fields define the number of EMIF clock cycles of setup time for the address pins (EM_A and EM_BA) and asynchronous chip enable (EM_CS) before the read strobe pin (READ_OE) or write strobe pin (WRITE_WE) falls, minus 1 cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EM_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_STROBE/R_STROBE	Read/Write strobe widths. These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin (READ_OE) or write strobe pin (WRITE_WE), minus 1 cycle. If Extended Wait mode is enabled by setting the EW bit in the asynchronous configuration register (AnCR), these fields must be set to a value greater than zero. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_HOLD/R_HOLD	Read/Write hold widths. These fields define the number of EMIF clock cycles of hold time for the address pins (EM_A and EM_BA) and asynchronous chip enable (EM_CS) after the read strobe pin (READ_OE) or write strobe pin (WRITE_WE) rises, minus 1 cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EM_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
TA	Minimum turnaround time. This field defines the minimum number of EMIF clock cycles between the end of one asynchronous access and the start of another, minus 1 cycle. This delay is not incurred when a read is followed by a read, or a write is followed by a write to the same chip select space. The purpose of this feature is to avoid contention on the bus. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
ASIZE	Asynchronous Device Bus Width. This field reflects the data bus width of the asynchronous interface in the following way: ASIZE = 0h indicates an 8-bit bus. Note that a request for a 32-bit word requires four external accesses with ASIZE = 0h. ASIZE = 1h indicates an 16-bit bus.

10.2.5.4 Read and Write Operations in Normal Mode

Normal mode is the asynchronous interface's default mode of operation. The Normal mode is selected when the SS bit in the asynchronous configuration register (AnCR) is cleared to 0. In this mode, the EM_CS signal operates as a chip enable signal, active throughout the duration of the memory access.

10.2.5.4.1 Asynchronous Read Operations (Normal Mode)

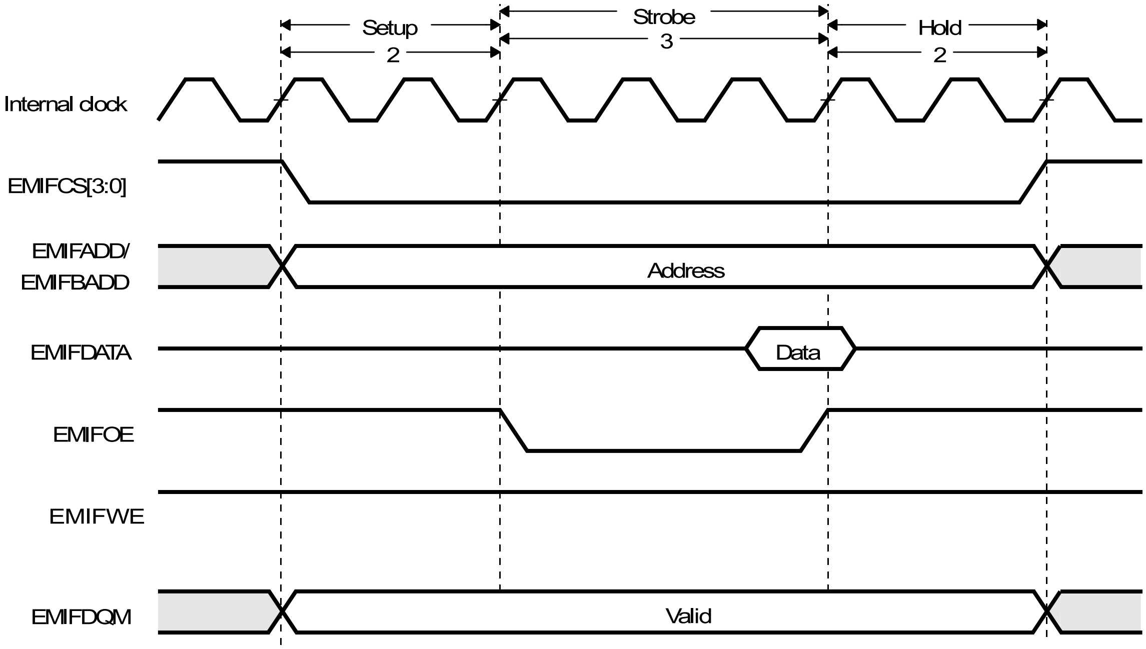
An asynchronous read is performed when any of the requesters mentioned in Section 10.2.2 request a read from the attached asynchronous memory. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Normal mode are described in Table 10-4 and an example timing diagram of a basic read operation is shown in Figure 10-4.

> **Note:**
> During the entirety of an asynchronous read operation, the WRITE_WE and EM_RW pins are driven high.

Table 10-4. Asynchronous Read Operation in Normal Mode

Time Interval	Pin Activity in WE Strobe Mode
Turnaround Period	Once the EMIF receives a read request, the EMIF waits for the programmed number of turn-around cycles period before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous configuration register (AnCR). There are two exceptions to this rule: If the current read operation was directly proceeded by another read operation to the same CS space, no turnaround cycles are inserted. If the current read operation was not directly proceeded by a read operation to the same CS space and the TA field has been cleared to 0, one turn-around cycle will be inserted. After the EMIF has waited for the turnaround cycles to complete, it proceeds to the setup period of the operation.
Start of setup period	At the beginning of the setup period: The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in AnCR. The address pins EM_A and EM_BA become valid EM_CS falls to enable the external device (if not already low from a previous operation)
Start of strobe period	At the beginning of the strobe period READ_OE falls
Start of hold period	At the beginning of the hold period: READ_OE rises The EMIF samples the data on the EM_D bus.
End of hold period	At the end of the hold period: The address pins EM_A and EM_BA become invalid EM_CS rises (if no more operations are required to complete the current request) The EMIF will be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 10-4. Timing Waveform of an Asynchronous Read Cycle in Normal Mode



10.2.5.4.2 Asynchronous Write Operations (Normal Mode)

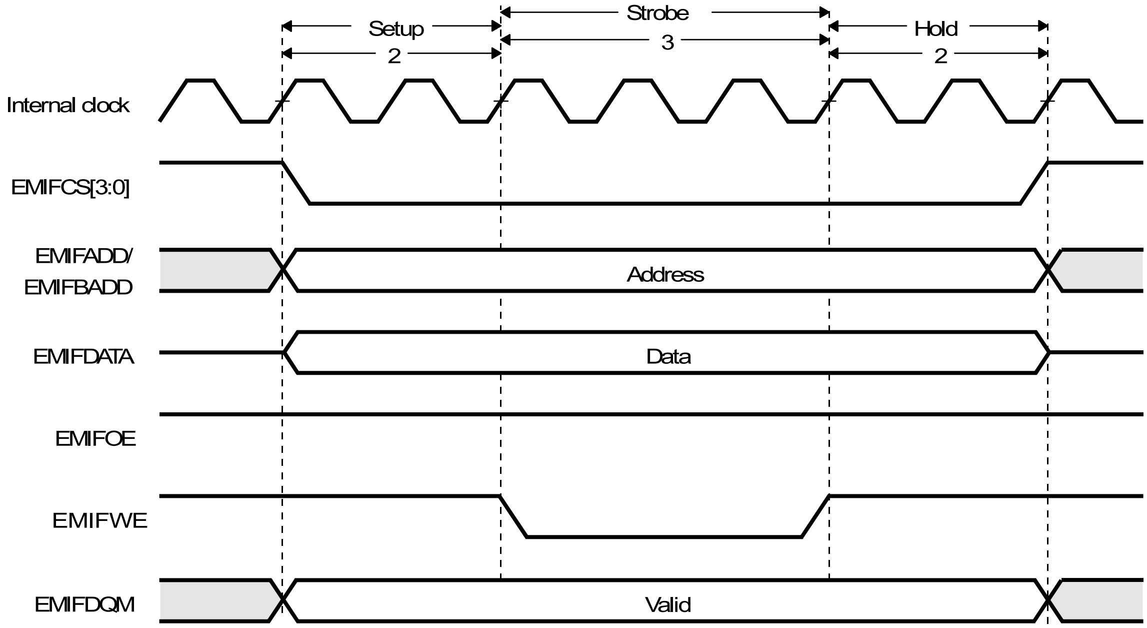
An asynchronous write is performed when any of the requesters mentioned in Section 10.2.2 request a write to asynchronous memory. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Normal mode are described in Table 10-5 and an example timing diagram of a basic write operation is shown in Figure 10-5.

Note:
During the entirety of an asynchronous write operation, the $\overline{\text{EM\_OE}}$ pin is driven high.

Table 10-5. Asynchronous Write Operation in Normal Mode

Time Interval	Pin Activity in WE Strobe Mode
Turnaround Period	Once the EMIF receives a write request, the EMIF waits for the programmed number of turn-around cycles period before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous configuration register (AnCR). There are two exceptions to this rule: If the current write operation was directly proceeded by another write operation to the same CS space, no turnaround cycles are inserted. If the current write operation was not directly proceeded by a write operation to the same CS space and the TA field has been cleared to 0, one turnaround cycle will be inserted. After the EMIF has waited for the turnaround cycles to complete, it proceeds to the setup period of the operation.
Start of setup period	At the beginning of the setup period: The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in AnCR. The address pins EM_A and EM_BA and the data pins EM_D become valid. The EM_RW pin falls to indicate a write (if not already low from a previous operation). EM_CS falls to enable the external device (if not already low from a previous operation).
Start of strobe period	At the beginning of the strobe period of a write operation: EM_WE falls
Start of hold Period	At the beginning of the hold period EM_WE rises
End of hold Period	At the end of the hold period: The address pins EM_A and EM_BA become invalid The data pins become invalid The EM_RW pin rises (if no more operations are required to complete the current request) EM_CS rises (if no more operations are required to complete the current request) The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 10-5. Timing Waveform of an Asynchronous Write Cycle in Normal Mode



10.2.5.5 Read and Write Operations in Select Strobe Mode

Select Strobe mode is the EMIF's second mode of operation. The SS mode is selected when the SS bit in the asynchronous configuration register (AnCR) is set to 1. In this mode, the EM_CS pin functions as a strobe signal and is therefore only active during the strobe period of an access cycle.

10.2.5.5.1 Asynchronous Read Operations (Select Strobe Mode)

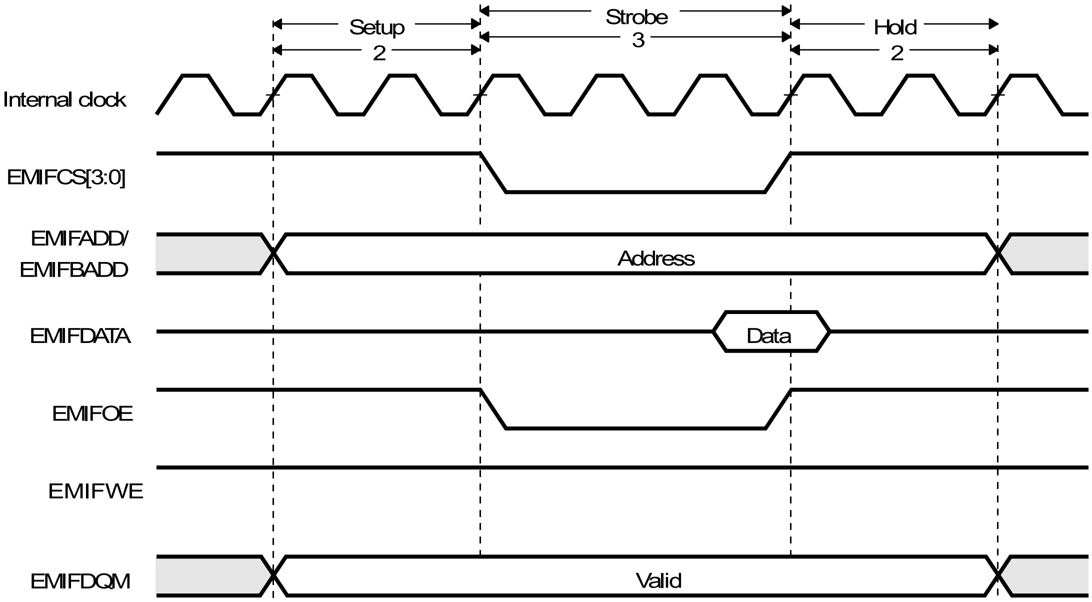
An asynchronous read is performed when any of the requesters mentioned in Section 10.2.2 request a read from the attached asynchronous memory. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe mode are described in Table 10-6 and an example timing diagram of a basic read operation is shown in Figure 10-6.

Note:
During the entirety of an asynchronous read operation, the EM_WE and EM_RW pins are driven high.

Table 10-6. Asynchronous Read Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround Period	Once the EMIF receives a read request, the EMIF waits for the programmed number of turn-around cycles before period proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous configuration register (AnCR). There are two exceptions to this rule: If the current read operation was directly proceeded by another read operation to the same CS space, no turnaround cycles are inserted. If the current read operation was not directly proceeded by a read operation to the same CS space and the TA field has been cleared to 0, one turnaround cycle will be inserted. After the EMIF has waited for the turnaround cycles to complete, it proceeds to the setup period of the operation.
Start of setup period	At the beginning of the setup period: The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in AnCR. The address pins EM_A and EM_BA become valid.
Start of strobe period	At the beginning of the strobe period: EM_CS and EM_OE fall at the start of the strobe period
Start of hold period	At the beginning of the hold period: period EM_CS and EM_OE rise The EMIF samples the data on the EM_D bus
End of hold period	At the end of the hold period: The address pins EM_A and EM_BA become invalid The EMIF may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted. If this is the case, the EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 10-6. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode



10.2.5.5.2 Asynchronous Write Operations (Select Strobe Mode)

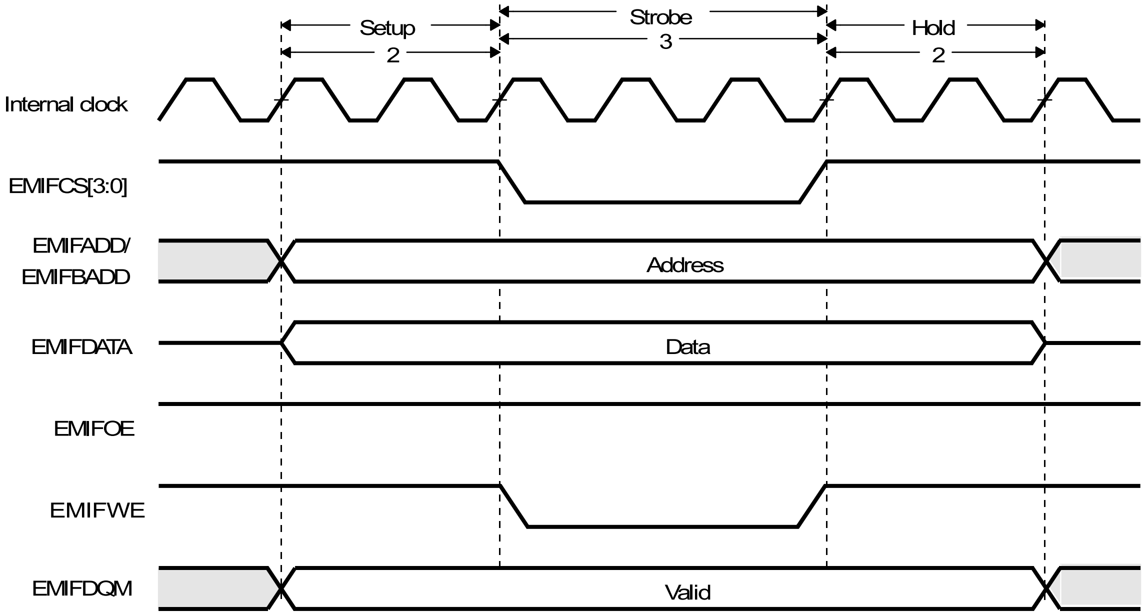
An asynchronous write is performed when any of the requesters mentioned in Section 10.2.2 request a write to memory in the asynchronous bank of the EMIF. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe mode are described in Table 10-7 and an example timing diagram of a basic write operation is shown in Figure 10-7.

Note: During the entirety of an asynchronous write operation, the EM_OE pin is driven high.

Table 10-7. Asynchronous Write Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround Period	Once the EMIF receives a write request, the EMIF waits for the programmed number of turn-around cycles period before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous configuration register (AnCR). There are two exceptions to this rule: If the current write operation was directly proceeded by another write operation to the same CS space, no turnaround cycles are inserted. If the current write operation was directly proceeded by a write operation to the same CS space and the TA field has been cleared to 0, one turnaround cycle will be inserted. After the EMIF has waited for the turnaround cycles to complete, it proceeds to the setup period of the operation.
Start of setup period	At the beginning of the setup period: The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in AnCR. The address pins EM_A and EM_BA and the data pins EM_D become valid. The EM_RW pin falls to indicate a write (if not already low from a previous operation).
Start of strobe period	At the beginning of the strobe period: EM_CS and EM_WE fall
Start of hold period	At the beginning of the hold period: EM_CS and EM_WE rise
End of hold period	At the end of the hold period: The address pins EM_A and EM_BA become invalid The data pins become invalid The EM_RW pin rises (if no more operations are required to complete the current request) The EMIF may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIF immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIF returns to its previous state unless another asynchronous request has been submitted. If this is the case, the EMIF instead enters directly into the turn-around period for the pending read or write operation.

Figure 10-7. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode



10.2.5.6 Data Bus Parking

The EMIF always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIF issues a read command to the external memory does it stop driving the data bus. After the EMIF latches the last read data, it immediately parks the data bus again.

10.2.5.7 Reset and Initialization Considerations

The EMIF and its registers will be reset when any of the following events occur:

1. The $\overline{\text{nPORST}}$ pin on the device is asserted
2. Wake up from hibernate..

When a reset occurs, the EMIF will immediately abandon any access request that is in progress and reset all registers and internal logic to their default state.

Following device power-up and deassertion of the $\overline{\text{nPORST}}$ pin, the internal clock to the EMIF can be turned on and the EMIF memory-mapped registers are programmed to their default values.

The following steps are required to configure the EMIF module after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see the device-specific data sheet).
2. Program the VDD3P3V_PWDN register to power up the IO pins for the EMIF (see the device-specific data manual).
3. Program the Power and Sleep Controller (PSC) to enable the EMIF module. For details on the PSC, see SPRU978.
4. Program the wait cycle register (AWCCR) to select the polarity of the wait signal and the maximum number of extended wait states (steps 4-7 are described in this document).
5. Program the chip select space configuration registers (AnCR) to select the interface parameters for the memory accesses.
6. Program the interrupt configuration registers (EIRR, EIMR, EIMSR, and EIMCR) to select the desired interrupt configuration.

The EMIF module is now ready to perform memory accesses.

10.2.5.8 Interrupt Support

The EMIF supports a single interrupt to the CPU. The interrupt is not multiplexed with another interrupt and is therefore always available.

There are two conditions that may cause the EMIF to generate an interrupt to the CPU. These two conditions are:

- An asynchronous time out

Only when the interrupt is enabled by setting the appropriate bit (ATMSET) in the EMIF interrupt mask set register (EIMSR) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIF interrupt mask clear register (EIMCR). The bit fields in both the EIMSR and EIMCR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the EIMSR and EIMCR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIF interrupt raw register (EIRR) and the EMIF interrupt mask register (EIMR) indicate the status of each interrupt. The appropriate bit (AT) in EIRR is set when the interrupt condition occurs, whether or not the interrupt has been enabled. Whereas, the appropriate bit (ATM) in EIMR is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in EIRR clears the EIRR bit as well as the corresponding bit in EIMR.

Table 10-8 contains a brief summary of the interrupt status and control bit fields. See Section 10.4 for complete details on the register fields.

Table 10-8. Interrupt Monitor and Control Bit Fields

Register Name	Bit Name	Description
EMIF interrupt raw register (EIRR)	AT	This bit is always set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the ATM bit in EIMR.
EMIF interrupt mask register (EIMR)	ATM	This bit is only set when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the ATMSET bit in EIMSR.
EMIF interrupt mask set register (EIMSR)	ATMSET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
EMIF interrupt mask clear register (EIMCR)	ATMCLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.

10.2.5.9 Power Management

Power dissipation to the EMIF may be managed by gating the input clock to the EMIF off. The input clock is turned off outside of the EMIF through the use of the CGM Module. The software must endure that all traffic from the CPU or other master accessing the EMIF has stopped before disabling the EMIF.

10.2.5.10 Emulation Considerations

The operation of the EMIF is not affected when a breakpoint is reached or an emulation halt occurs.

10.2.5.11 Possible Race Condition

A race condition may exist when certain masters write data to the EMIF. For example, if master A passes a software message via a buffer in EMIF memory and does not wait for indication that the write completes, when master B attempts to read the software message it may read stale data and therefore receive an incorrect message. In order to confirm that a write from master A has landed before a read from master B is performed, master A must wait for the write completion status from the EMIF before indicating to master B that the data is ready to be read. If master A does not wait for indication that a write is complete, it must perform the following workaround:

1. Perform the required write.
2. Perform a dummy read from the EMIF memory from the same CS space where the write was issued.

Indicate to master B that the data is ready to be read after completion of the read in step 2. The completion of the read in step 2 ensures that the previous write was done.

The EDMA and ATA peripherals do not need to implement the above workaround. If a peripheral is not listed here, then the above workaround is required. Refer to the device-specific data manual for more information.

10.2.6 Endianess Support

Since the device internal data bus is 32-bits wide and the device operates in little-endian mode, data from the internal data bus is written to or read from the external 8-bit memory in a very specific sequence to maintain the natural order of little-endian operations. That is, a stream of data starting at any address n will always be accessed in the correct or incrementing order and the EMIF will always access address n prior to n+1. Table 10-9 shows the EMIF data ordering for memory accesses.

Table 10-9. 8-Bit Asynchronous Memory Data Ordering

Internal Bus		External Bus	
Data	**Address**	**EM_BA[1:0]**	**EM_D[7:0]**
xxxx xxDEh	0h	0h	DEh
xxxx BCxxh	0h	1h	BCh
xx9A xxxxh	0h	2h	9Ah
78xx xxxxh	0h	3h	78h
xxxx BCDEh	0h	0h	DEh
	0h	1h	BCh
789A xxxxh	0h	2h	9Ah
	0h	3h	78h
789A BCDEh	0h	0h	DEh
	0h	1h	BCh
	0h	2h	9Ah
	0h	3h	78h

10.3 Example Configuration

The EMIF allows a high degree of programmability for shaping asynchronous accesses. As previously stated, the shape and duration of the asynchronous access is determined by controlling the widths of the SETUP, STROBE, HOLD, and turnaround periods. The widths of these periods are configured by programming the asynchronous configuration register (AnCR) for the corresponding chip select space. See Section 10.2.5.3 and Section 10.4.2 for more information.

The programmability inherent to the EMIF, provides the EMIF with the flexibility to interface with a variety of asynchronous memory types. By programming the W_SETUP/R_SETUP, W_STROBE/R_STROBE, W_HOLD/R_HOLD, TA, and ASIZE fields in AnCR, the EMIF can be configured to meet the data sheet specification for most asynchronous memory devices.

This section presents examples describing how to interface the EMIF to asynchronous SRAM.

10.3.1 Interfacing to Asynchronous SRAM (ASRAM)

The following example describes how to interface the EMIF to the ISSI IS61WV20488BLL device.

10.3.1.1 3.1.1 Connecting to ASRAM

Figure 10-8 shows how to connect the EMIF to the IS61WV20488BLL device.

Figure 10-8. Connecting the EMIF to the IS61WV20488BLL

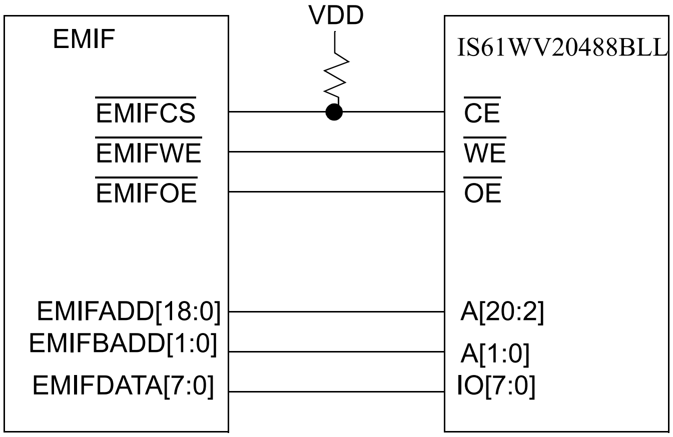
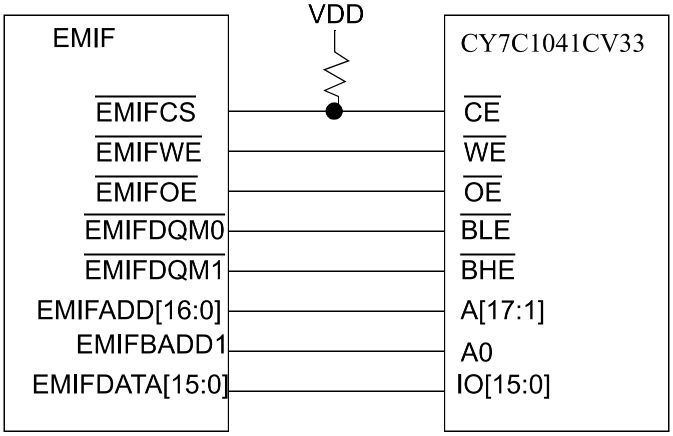


Figure 10-9 shows how to connect the EMIF to the CY7C1041CV33 device.

Figure 10-9. Connecting the EMIF to the CY7C1041CV33



10.3.1.2 Meeting AC Timing Requirements for ASRAM

When configuring the EMIF to interface to ASRAM, you must consider the AC timing requirements of the ASRAM as well as the AC timing requirements of the EMIF. These can be found in the data sheet for each respective device. The read and write asynchronous cycles are programmed separately in the asynchronous configuration register (AnCR).

For a read access, Table 10-10, Table 10-11 and TTable 10-12 list the AC timing specifications that must be considered.

Table 10-10. EMIF Input Timing Requirements

Parameter	Description
tSU	Data Setup time, data valid before EM_OE high
tH	Data Hold time, data valid after EM_OE high

Table 10-11. ASRAM Output Timing Characteristics

Parameter	Description
tACC	Address Access time
tOH	Output data Hold time for address change
COD	Output Disable time from chip enable

Table 10-12. ASRAM Input Timing Requirement for a Read

Parameter	Description
tRC Read Cycle time	

Figure 10-10 shows an asynchronous read access and describes how the EMIF and ASRAM AC timing requirements work together to define the values for R_SETUP, R_STROBE, and R_HOLD.

From Figure 10-10, the following equations may be derived. tcyc is the period at which the EMIF operates. The R_SETUP, R_STROBE, and R_HOLD fields are programmed in terms of EMIF cycles where as the data sheet specifications are typically given in nano seconds. This explains the presence of tcyc in the denominator of the following equations. A minus 1 is included in the equations because each field in AnCR is programmed in terms of EMIF clock cycles, minus 1 cycle. For example, R_SETUP is equal to R_SETUP width in EMIF clock cycles minus 1 cycle.

$$\text{R\_SETUP} + \text{R\_STROBE} \geq \frac{(t_{ACC}(m) + t_{SU})}{t_{cyc}} - 1$$

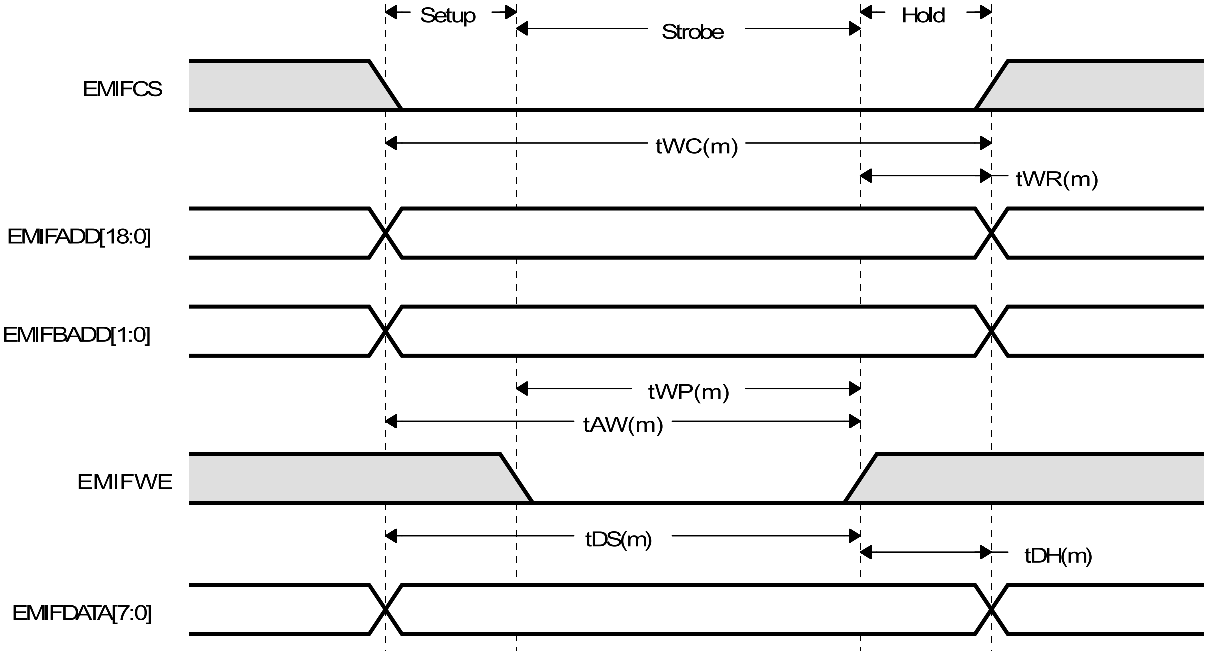
$$\text{R\_SETUP} + \text{R\_STROBE} + \text{R\_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

$$\text{R\_HOLD} \geq \frac{(t_H - t_{OH}(m))}{t_{cyc}} - 1$$

The EMIF offers an additional parameter, TA, that defines the turnaround time between read and write cycles. This parameter protects against the situation when the output turn-off time of the memory is longer than the time it takes to start the next write cycle. If this is the case, the EMIF will drive data at the same time as the memory, causing contention on the bus. By examining Figure 10-10, the equation for TA can be derived as:

$$TA \geq \frac{t_{COD}(m)}{t_{CYC}} -1$$

Figure 10-10. Timing Waveform of an ASRAM Read



For a write access, Table 10-13 lists the AC timing specifications that must be satisfied.

Table 10-13. ASRAM Input Timing Requirements for a Write

Parameter	Description
tWP	Write Pulse width
tAW	Address valid to end of Write
tDS	Data Setup time
tWR	Write Recovery time
tDH	Data Hold time
tWC	Write Cycle time

Figure 10-11 shows an asynchronous write access and describes how the EMIF and ASRAM AC timing requirements work together to define values for W_SETUP, W_STROBE, and W_HOLD.

From Figure 10-11, the following equations may be derived. tcyc is the period at which the EMIF operates. The W_SETUP, W_STROBE, and W_HOLD fields are programmed in terms of EMIF cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of tcyc in the denominator of the following equations. A minus 1 is included in the equations because each field in AnCR is programmed in terms of EMIF clock cycles, minus 1 cycle. For example, W_SETUP is equal to W_SETUP width in EMIF clock cycles minus 1 cycle.

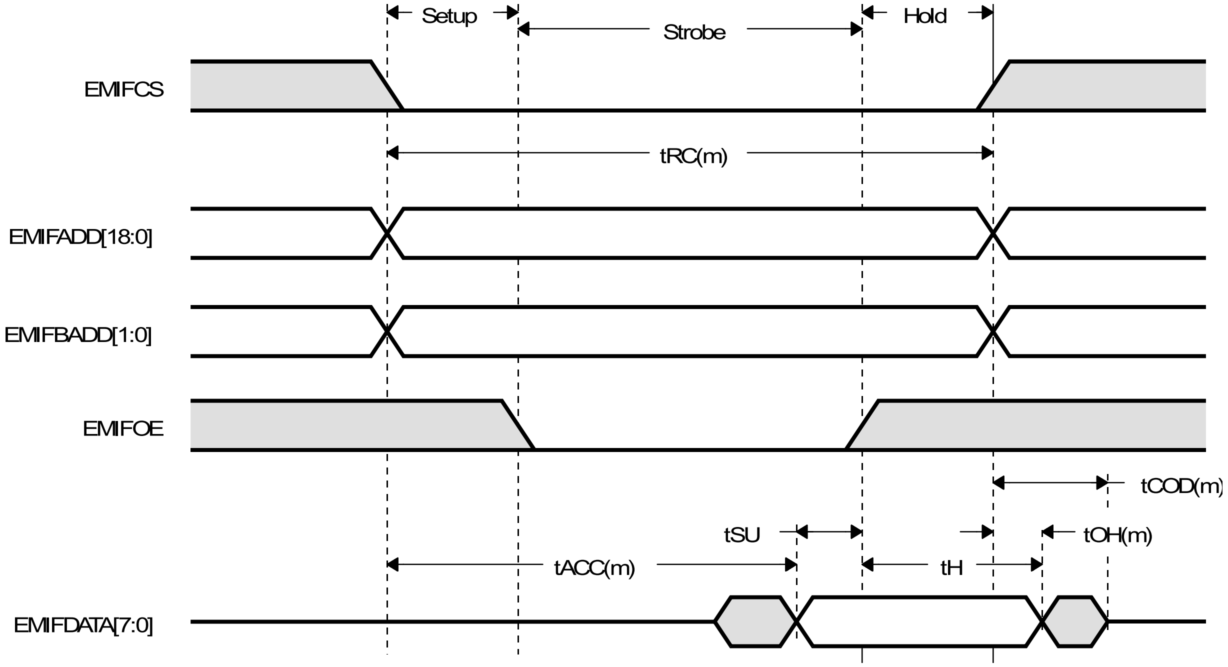
$$\text{W\_STROBE} \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$\text{W\_SETUP} + \text{W\_STROBE} \geq \max\left(\frac{t_{AW}(m)}{t_{cyc}}, \frac{t_{DS}(m)}{t_{cyc}}\right) - 1$$

$$\text{W\_HOLD} \geq \max\left(\frac{t_{WR}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1$$

$$\text{W\_SETUP} + \text{W\_STROBE} + \text{W\_HOLD} \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 10-11. Timing Waveform of an ASRAM Write



10.3.1.3 Taking Into Account PCB Delays

The equations described in Section 10.3.1.2 are for the ideal case, when board design does not contribute delays. Board characteristics, such as impedance, loading, length, number of nodes, etc., affect how the device driver behaves. Signals driven by the EMIF will be delayed when they reach the ASRAM and conversely. Table 10-14 lists the delays shown in Figure 10-12 and Figure 10-13 due to PCB affects. The PCB delays are board specific and must be estimated or determined though the use of IBIS modeling. The signals denoted (ASRAM) are the signals seen at the ASRAM. For example, EM_CS represents the signal at the EMIF and EM_CS (ASRAM) represents the delayed signal seen at the ASRAM.

Table 10-14. ASRAM Timing Requirements With PCB Delays

Parameter	Description
	Read Access
tEM_CS	Delay on EM_CS from EMIF to ASRAM. EM_CS is driven by EMIF.
tEM_A	Delay on EM_A from EMIF to ASRAM. EM_A is driven by EMIF.
tEM_OE	Delay on EM_OE from EMIF to ASRAM. EM_OE is driven by EMIF.
tEM_D	Delay on EM_D from ASRAM to EMIF. EM_D is driven by ASRAM.
	Write Access
tEM_CS	Delay on EM_CS from EMIF to ASRAM. EM_CS is driven by EMIF.
tEM_A	Delay on EM_A from EMIF to ASRAM. EM_A is driven by EMIF.
tEM_WE	Delay on EM_WE from EMIF to ASRAM. EM_WE is driven by EMIF.
tEM_D	Delay on EM_D from EMIF to ASRAM. EM_D is driven by EMIF.

From Figure 10-12, the following equations may be derived. tcyc is the period at which the EMIF operates. The R_SETUP, R_STROBE, and R_HOLD fields are programmed in terms of EMIF cycles where as the data sheet specifications are typically given in nano seconds. This is explains the presence of tcyc in the denominator of the following equations. A minus 1 is included in the equations because each field in AnCR is programmed in terms of EMIF clock cycles, minus 1 cycle. For example, R_SETUP is equal to R_SETUP width in EMIF clock cycles minus 1 cycle.

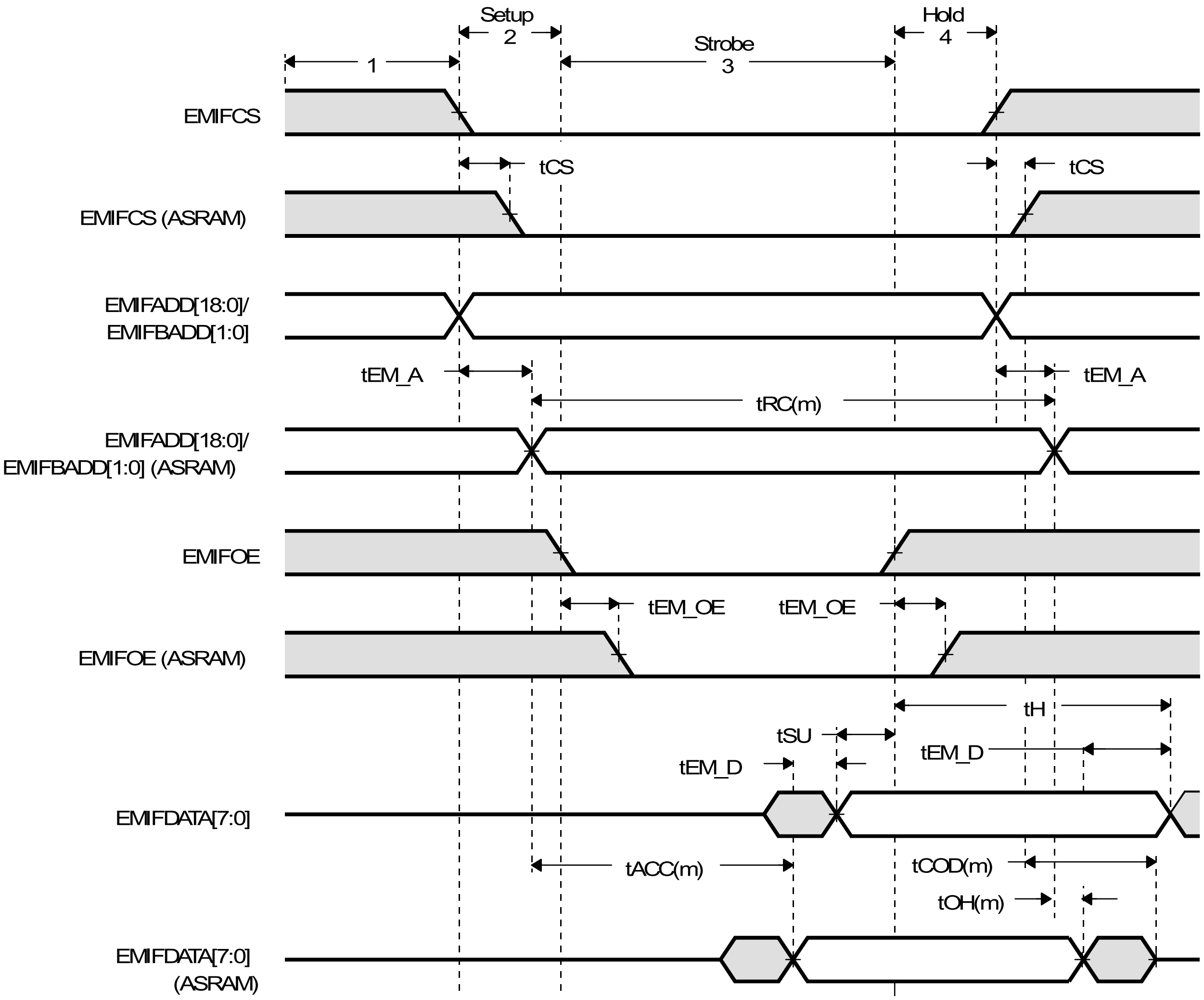
$$\text{R\_SETUP} + \text{R\_STROBE} \geq \frac{\left(t_{EM\_A} + t_{ACC}(m) + t_{SU} + t_{EM\_D}\right)}{t_{cyc}} - 1$$

$$\text{R\_SETUP} + \text{R\_STROBE} + \text{R\_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

$$\text{R\_HOLD} \geq \frac{\left(t_{H} - t_{EM\_D} - t_{OH}(m) - t_{EM\_A}\right)}{t_{cyc}} - 1$$

$$\text{TA} \geq \frac{\left(t_{EM\_CS} + t_{COD}(m) + t_{EM\_D}\right)}{t_{cyc}} - 1$$

Figure 10-12. Timing Waveform of an ASRAM Read with PCB Delays



From Figure 10-13, the following equations may be derived. tcyc is the period at which the EMIF operates. The W_SETUP, W_STROBE, and W_HOLD fields are programmed in terms of EMIF cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of tcyc in the denominator of the following equations. A minus 1 is included in the equations because each field in AnCR is programmed in terms of EMIF clock cycles, minus 1 cycle. For example, W_SETUP is equal to W_SETUP width in EMIF clock cycles minus 1 cycle.

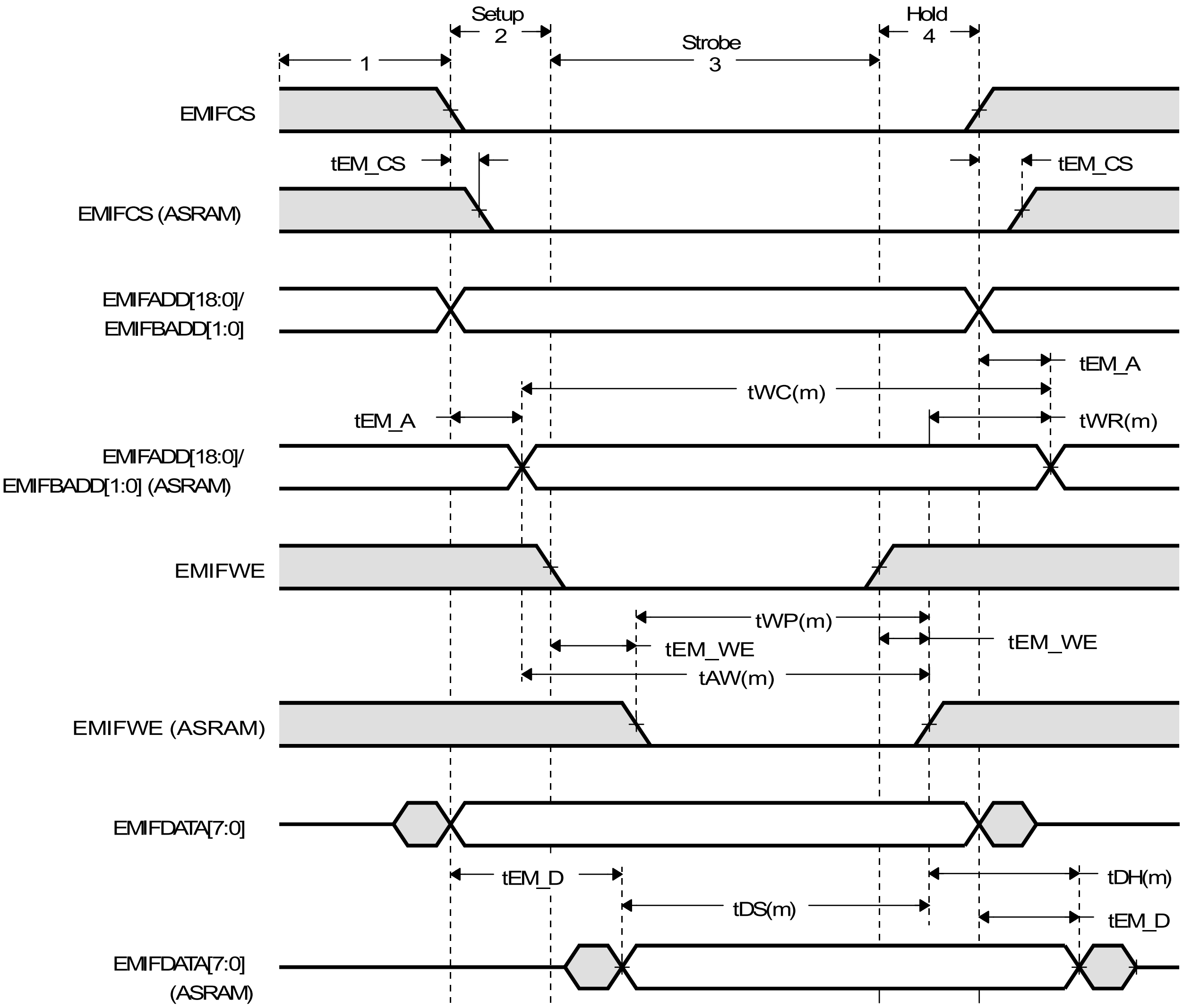
$$\text{W\_STROBE} \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$\text{W\_SETUP} + \text{W\_STROBE} \geq \max\left(\frac{(t_{EM\_A} + t_{AW}(m) - t_{EM\_WE})}{t_{cyc}}, \frac{(t_{EM\_D} + t_{DS}(m) - t_{EM\_WE})}{t_{cyc}}\right) - 1$$

$$\text{W\_HOLD} \geq \max\left(\frac{(t_{EM\_WE} + t_{WR}(m) - t_{EM\_A})}{t_{cyc}}, \frac{(t_{EM\_WE} + t_{DH}(m) - t_{EM\_D})}{t_{cyc}}\right) - 1$$

$$\text{W\_SETUP} + \text{W\_STROBE} + \text{W\_HOLD} \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 10-13. Timing Waveform of an ASRAM Read with PCB Delays



10.3.1.4 Example Using IS61WV20488BLL

This section takes you through the configuration steps required to implement ISSI's IS61WV20488BLL ASRAM with the EMIF. The following assumptions are made:

- SRAM is connected to chip select space 3 (EM_CS[3])
- EMIF clock speed is 100 MHz (tcyc = 10 nS)

Table 10-15 lists the data sheet specifications for the EMIF and Table 10-16 lists the data sheet specifications for the ASRAM.

Table 10-15. EMIF Timing Requirements for IS61WV20488BLL Example

Parameter	Description	Min	Max	Units
tSU	Data Setup time, data valid before EM_OE high	5		nS
tH	Data Hold time, data valid after EM_OE high	0		nS

Table 10-16. ASRAM Timing Requirements for IS61WV20488BLL Example

Parameter	Description	Min	Max	Units
tACC	Address Access time		10	nS
tOH	Output data Hold time for address change	2		nS
tRC	Read cycle time	10		nS
tWP	Write Pulse width	8		nS
tAW	Address valid to end of Write	8		nS
tDS	Data Setup time	6		nS
tWR	Write Recovery time	0		nS
tDH	Data Hold time	0		nS
tWC	Write Cycle time	10		nS
tCOD	Output Disable time from chip enable		4	nS

Table 10-17 lists the values of the PCB board delays. The delays were estimated using the rule that there is 180 pS of delay for every 1 inch of trace.

Table 10-17. Measured PCB Delays for IS61WV20488BLL Example

Parameter	Description	Delay (ns)
	Read Access	
tEM_CS	Delay on EM_CS from EMIF to ASRAM. EM_CS is driven by EMIF.	0.36
tEM_A	Delay on EM_A from EMIF to ASRAM. EM_A is driven by EMIF.	0.27
tEM_OE	Delay on EM_OE from EMIF to ASRAM. EM_OE is driven by EMIF.	0.36
tEM_D	Delay on EM_D from ASRAM to EMIF. EM_D is driven by ASRAM.	0.45
	Write Access	
tEM_CS	Delay on EM_CS from EMIF to ASRAM. EM_CS is driven by EMIF.	0.36
tEM_A	Delay on EM_A from EMIF to ASRAM. EM_A is driven by EMIF.	0.27
tEM_WE	Delay on EM_WE from EMIF to ASRAM. EM_WE is driven by EMIF.	0.36
tEM_D	Delay on EM_D from EMIF to ASRAM. EM_D is driven by EMIF.	0.45

Inserting these values into the equations defined above allows you to determine the values for SETUP, STROBE, HOLD, and TA. For a read:

$$\text{R\_SETUP} + \text{R\_STROBE} \geq \frac{(t_{EM\_A} + t_{ACC}(m) + t_{SU} + t_{EM\_D})}{t_{cyc}} - 1 \geq \frac{(0.27 + 10 + 5 + 0.45)}{10} - 1 \geq 0.57$$

$$\text{R\_SETUP} + \text{R\_STROBE} + \text{R\_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3 \geq \left(\frac{10}{10}\right) - 3 \geq -2$$

$$\text{R\_HOLD} \geq \frac{(t_{H} - t_{EM\_D} - t_{OH}(m) - t_{EM\_A})}{t_{cyc}} - 1 \geq \frac{(0 - 0.45 - 2 - 0.27)}{10} - 1 \geq -1.27$$

$$\text{TA} \geq \frac{(t_{EM\_CS} + T_{COD}(m) + t_{EM\_D})}{t_{cyc}} - 1 \geq \frac{(0.36 + 4 + 0.45)}{10} - 1 \geq -0.52$$

Therefore if R_SETUP = 0, then R_STROBE = 0, R_HOLD = 0, and TA = 0. For a write:

$$\text{W\_STROBE} \geq \frac{t_{WP}(m)}{t_{cyc}} - 1 \;\geq\; \left(\frac{8}{10}\right) - 1 \geq -0.2$$

$$\text{W\_SETUP} + \text{W\_STROBE} \geq \max\left(\frac{(t_{EM\_A} + t_{AW}(m) - t_{EM\_WE})}{t_{cyc}}, \frac{(t_{EM\_D} + t_{DS}(m) - t_{EM\_WE})}{t_{cyc}}\right) - 1$$

$$\geq \max\left(\frac{(0.27 + 8 - 0.36)}{10}, \frac{(0.45 + 6 - 0.36)}{10}\right) - 1 \geq -0.21$$

$$\text{W\_HOLD} \geq \max\left(\frac{(t_{EM\_WE} + t_{WR}(m) - t_{EM\_A})}{t_{cyc}}, \frac{(t_{EM\_WE} + t_{DH}(m) - t_{EM\_D})}{t_{cyc}}\right) - 1$$

$$\geq \max\left(\frac{(0.36 + 0 - 0.27)}{10}, \frac{(0.36 + 0 - 0.45)}{10}\right) - 1 \geq -0.99$$

$$\text{W\_SETUP} + \text{W\_STROBE} + \text{W\_HOLD} \geq \frac{t_{WC}(m)}{t_{cyc}} - 3 \;\geq\; \left(\frac{10}{10}\right) - 3 \geq -2$$

Therefore, W_SETUP = 0, W_STROBE = 0, and W_HOLD = 0.

Since the value of the W_SETUP/R_SETUP, W_STROBE/R_STROBE, W_HOLD/R_HOLD, and TA fields are equal to EMIF clock cycles minus 1 cycle, the A2CR should be configured as in Table 10-18. In this example, the EM_WAIT signal is not implemented; therefore, the asynchronous wait cycle configuration register (AWCCR) does not need to be programmed.

Table 10-18. Configuring A2CR for IS61WV20488BLL Example

Parameter	Setting
SS	Select Strobe mode. • SS = 0. Places EMIF in Normal Mode.
EW	Extended Wait mode enable. • EW = 0. Disabled Extended wait mode.
W_SETUP/R_SETUP	Read/Write setup widths. • W_SETUP = 0 • R_SETUP = 0
W_STROBE/R_STROBE	Read/Write strobe widths. • W_STROBE = 0 • R_STROBE = 0
W_HOLD/R_HOLD	Read/Write hold widths. • W_HOLD = 0 • R_HOLD = 0
TA	Minimum turnaround time. • TA = 0
ASIZE	Asynchronous Device Bus Width. • ASIZE = 0, indicating an 8-bit data bus width • ASIZE = 1, indicating an 16-bit data bus width

10.4 Registers

The external memory interface (EMIF) is controlled by programming its internal memory-mapped registers (MMRs). Table 10-19 lists the memory-mapped registers for the EMIF. The base address for the control registers is 0xFFFF E800. All other register offset addresses not listed in Table 10-19 should be considered as reserved locations and the register contents should not be modified.

Note:
The EMIF MMRs only support word (4 byte) accesses. Performing a byte (8-bit) or halfword (16-bit) write to a register results in unknown behavior.

Table 10-19. External Memory Interface (EMIF) Registers

Offset	Acronym	Register Description	Section
0h	RCSR	Revision Code and Status Register	Section 10.4.1
10h	A1CR	Asynchronous 1 Configuration Register (CS0 space)	Section 10.4.2
14h	A2CR	Asynchronous 2 Configuration Register (CS1 space)	Section 10.4.2
18h	A3CR	Asynchronous 3 Configuration Register (CS2 space)	Section 10.4.2
1Ch	A4CR	Asynchronous 4 Configuration Register (CS3 space)	Section 10.4.2
40h	EIRR	EMIF Interrupt Raw Register	Section 10.4.3
44h	EIMR	EMIF Interrupt Mask Register	Section 10.4.4
48h	EIMSR	EMIF Interrupt Mask Set Register	Section 10.4.5
4Ch	EIMCR	EMIF Interrupt Mask Clear Register	Section 10.4.6

10.4.1 Revision Code and Status Register (RCSR)

The revision code and status register (RCSR) is shown in Figure 10-14 and described in Table 10-20.

Figure 10-14. Revision Code and Status Register (RCSR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BE	FR	MODULE_ID													
R-0	R-1	R-Fh													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAJOR_REVISION								MINOR_REVISION							
R-2h								R-1h							

R = Read only, -*n* = value after reset

Table 10-20. Revision Code and Status Register (RCSR) Field Descriptions

Bit	Name	Value	Description
31	BE	0	Big Endian. Reflects the endianness mode of the EMIF. Little-endian mode
30	FR	0 1	Full Rate. Reflects whether the EMIF is set to operate at full or half rate. Half rate Full rate
29-16	MODULE_ID	0-3FFFh	Module identification. EMIF: Fh = asynchronous mode
15-8	MAJOR_REVISION	0-FFh	Major Revision. EMIF code revisions are indicated by a revision code taking the format MAJOR_REVISION.MINOR_REVISION. Major revision = 2h
7-0	MINOR_REVISION	0-FFh	Minor Revision. EMIF code revisions are indicated by a revision code taking the format MAJOR_REVISION.MINOR_REVISION. Minor revision = 1h

10.4.2 Asynchronous Configuration Registers (A1CR-A4CR)

The asynchronous configuration register (AnCR) is used to configure the shaping of the address and control signals during an access to asynchronous memory. It is also used to program the width of asynchronous interface and to select from various modes of operation. This register can be written prior to any transfer, and any asynchronous transfer following the write will use the new configuration. The AnCR is shown in Figure 10-15 and described in Table 21. There are four AnCRs. Each chip select space has a dedicated AnCR. This allows each chip select space to be programmed independently to interface to different asynchronous memory types.

Figure 10-15. Asynchronous n Configuration Register (AnCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS	Reserved	W_SETUP				W_STROBE						W_HOLD			R_SETUP
R/W-0	R-0	R/W-Fh				R/W-3Fh						R/W-7h			R/W-Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R_SETUP			R_STROBE						R_HOLD			TA		ASIZE	
R/W-Fh			R/W-3Fh						R/W-7h			R/W-3h		R/W-0	

R/W = Read/Write; -n = value after reset

A The W_STROBE and R_STROBE bits must not be cleared to 0 when operating in Extended Wait mode.

Table 10-21. Asynchronous n Configuration Register (AnCR) Field Descriptions

Bit	Name	Value	Description
31	SS		Select Strobe bit. This bit defines whether the asynchronous interface operates in Normal mode or Select Strobe mode. See Section 2.5 for details on the two modes of operation.
		0	Normal mode is enabled.
		1	Select Strobe mode is enabled.
30	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
29-26	W_SETUP	0-Fh	Write setup width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.
25-20	W_STROBE	0-3Fh	Write strobe width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.
19-17	W_HOLD	0-7h	Write hold width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.
16-13	R_SETUP	0-Fh	Read setup width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.
12-7	R_STROBE	0-3Fh	Read strobe width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.
6-4	R_HOLD	0-7h	Read hold width in EMIF clock cycles, minus 1 cycle. See Section 10.2.5.3 for details.

Table 10-21. Asynchronous n Configuration Register (AnCR) Field Descriptions

Bit	Name	Value	Description
3-2	TA	0-3h	Minimum Turn-Around time. This field defines the minimum number of EMIF clock cycles between the end of one asynchronous access and the start of another, minus 1 cycle. This delay is not incurred by a read followed by a read or a write followed by a write to the same CS space. See Section 2.5.3 for details.
1-0	ASIZE	0-3h	Asynchronous data bus width.
		0	8-bit data bus
		1	16-bit data bus
		2h-3h	Reserved

10.4.3 EMIF Interrupt Raw Register (EIRR)

The EMIF interrupt raw register (EIRR) is used to monitor and clear the EMIF's hardware-generated interrupts. The bits in EIRR will be set when an interrupt condition occurs regardless of the status of the EMIF interrupt mask set register (EIMSR) and EMIF interrupt mask clear register (EIMCR). Writing a 1 to these bit fields will clear them as well as the corresponding bit field in the EMIF interrupt mask register (EIMR). The EIRR is shown in Figure 10-16 and described in Table 22.

Figure 10-16. EMIF Interrupt Raw Register (EIRR)

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
Reserved															AT
R-0															R/W1C-0

R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

Table 10-22. EMIF Interrupt Raw Register (EIRR) Field Descriptions

Bit	Name	Value	Description
31-1	Reserved		Reads return 0 and writes have no effect.
0	AT		Asynchronous Timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle the EM_WAIT pin did not go inactive within the number of cycles defined by the MEWC field in the asynchronous wait cycle configuration register (AWCCR).
		0	Indicates that an asynchronous timeout has not occurred. Writing a 0 has no effect.
		1	Indicates that an asynchronous timeout has occurred. Writing a 1 will clear this bit and the ATM bit in the EMIF interrupt mask register (EIMR).

10.4.4 EMIF Interrupt Mask Register (EIMR)

Like the EMIF interrupt raw register (EIRR), the EMIF interrupt mask register (EIMR) is used to monitor and clear the status of the EMIF's hardware-generated interrupts. The main difference between the two registers is that when the bit fields in EIMR are set, an active-high pulse will be sent to the CPU interrupt controller. Also, the bit fields in EIMR are only set to 1 if the associated interrupt has been enabled in the EMIF interrupt set register (EISR). The EIMR is shown in Figure 10-17 and described in Table 23.

Figure 10-17. EMIF Interrupt Mask Register (EIMR)

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
Reserved															ATM
R-0															R/W1C-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 10-23. EMIF Interrupt Mask Register (EIMR) Field Descriptions

Bit	Name	Value	Description
31-1	Reserved		Reads return 0 and writes have no effect.
0	ATM		Asynchronous Timeout Masked. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle the EM_WAIT pin did not go inactive within the number of cycles defined by the MEWC field in the asynchronous wait cycle configuration register (AWCCR), provided that the ATMSET bit is set to 1 in the EMIF interrupt mask set register (EIMSR).
		0	Indicates that an asynchronous timeout interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that an asynchronous timeout interrupt has been generated. Writing a 1 will clear this bit and the AT bit in the EMIF interrupt mask register (EIMR).

10.4.5 EMIF Interrupt Mask Set Register (EIMSR)

The EMIF interrupt mask set register (EIMSR) is used to enable the interrupts. If a bit is set to 1, the corresponding bit in the EMIF interrupt masked register (EIMR) will be set and an interrupt will be generated when the associated interrupt condition occurs. If a bit is cleared to 0, the the corresponding bit in EIMR will always read 0 and no interrupts will be generated when the associated interrupt condition occurs. Writing a 1 to the WRMSET and ATMSET bits enables each respective interrupt. The EIMSR is shown in Figure 10-18 and described in Table 24.

Figure 10-18. EMIF Interrupt Mask Set Register (EIMSR)

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
Reserved															ATMSET
R-0															R/W-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 10-24. EMIF Interrupt Mask Set Register (EIMSR) Field Descriptions

Bit	Name	Value	Description
31-1	Reserved		Reads return 0 and writes have no effect.
0	ATMSET		Asynchronous Timeout Mask Set. This bit determines whether or not the asynchronous timeout interrupt is enabled. Writing a 1 to this bit sets this bit, sets the ATMCLR bit in the EMIF interrupt mask clear register (EIMCR), and enables the asynchronous timeout interrupt. To clear this bit, a 1 must be written to the ATMCLR bit in EIMCR.
		0	Indicates that the asynchronous timeout interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the asynchronous timeout interrupt is enabled. Writing a 1 sets this bit and the ATMCLR bit in the EMIF interrupt mask clear register (EIMCR).

10.4.6 EMIF Interrupt Mask Clear Register (EIMCR)

The EMIF interrupt mask clear register (EIMCR) is used to disable the interrupts. If a bit is read as 1, the corresponding bit in the EMIF interrupt masked register (EIMR) will be set and an interrupt will be generated when the associated interrupt condition occurs. If a bit is read as 0, the corresponding bit in EIMR will always read 0 and no interrupt will be generated when the corresponding interrupt condition occurs. Writing a 1 to the ATMCLR and WRMCLR bits disables each respective interrupt. The EIMCR is shown in Figure 10-19 and described in Table 25.

Figure 10-19. EMIF Interrupt Mask Clear Register (EIMCR)

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
Reserved															ATMCLR
R-0															R/W-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 10-25. EMIF Interrupt Mask Clear Register (EIMCR) Field Descriptions

Bit	Name	Value	Description
31-1	Reserved		Reads return 0 and writes have no effect.
0	ATMCLR		Asynchronous Timeout Mask Clear. This bit determines whether or not the asynchronous timeout interrupt is enabled. Writing a 1 to this bit clears this bit, clears the ATMSET bit in the EMIF interrupt mask set register (EIMSR), and disables the asynchronous timeout interrupt. To set this bit, a 1 must be written to the ATMSET bit in EIMSR.
		0	Indicates that the asynchronous timeout interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the asynchronous timeout interrupt is enabled. Writing a 1 clears this bit and the ATMSET bit in the EMIF interrupt mask set register (EIMSR).

Chapter 11

SPNU489C-February 2012

Parameter Overlay Module (POM)

This reference guide provides details for the Parameter Overlay Module (POM).

Topic Page

11.1 Introduction

In many applications it is important to be able to change certain parameters in the program without having to re-flash the device and immediately test these changes either in a hardware-in-the-loop simulation or in a real environment.

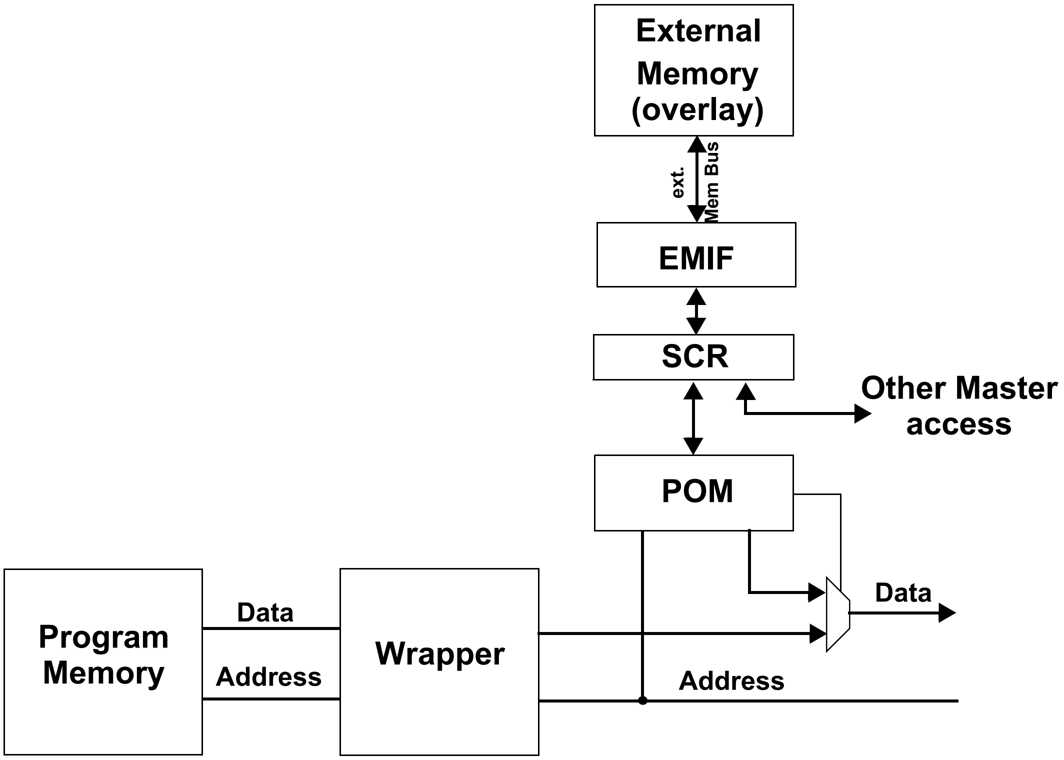
The POM provides a mechanism to redirect accesses to non-volatile memory into a volatile memory external to the device. The data requested by the CPU will be fetched from the overlay memory instead of the main non-volatile memory. The overlay memory can be accessed directly by other masters in the system to provide an easy update path of the stored data. Other masters can be for example the main CPU, DMA, DMM or JTAG.

11.1.1 Main Features

- Redirects program memory accesses to internal/external memory interface (overlay)
- Up to 4 MByte of external overlay memory
- Provides up to 32 programmable memory regions to replace non-volatile memory
  - Programmable region start address
  - Programmable region size (64 Bytes up to 256kBytes in power of 2 steps)
- Overlay memory is memory mapped
  - Writable by any master (e.g. CPU, DMA, DMM, etc.)
- Memory Patch functionality for production devices

11.2 Block diagram

Figure 11-1. System block diagram for external memory configuration



11.3 Module Operation

The POM has up to 32 programmable regions. Whenever the CPU requests data from the non-volatile memory which address falls into one of the programmed regions, the POM will request the data from the overlay memory. Waitstates will be automatically inserted until the data is available from the overlay memory. This ensures that the overlay memory has the same (in the case that the latency from overlay memory is less than the program memory latency) or slower access time than the program memory.

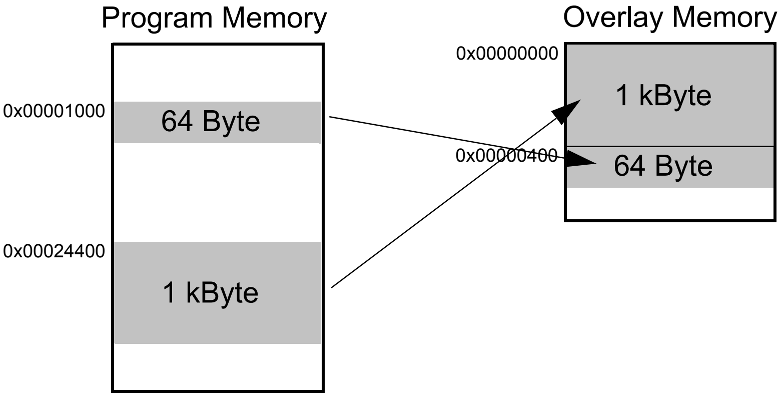
The POM does not provide a feature to write into the overlay memory. The write has to be performed directly to the memory mapped address space of the overlay memory.

When the module is disabled, no redirection of access will be performed.

11.3.1 Decode Regions

There are 32 decode regions. Regions are defined by a start address and a region size. The start address is 22 bits wide to cover the maximum 4 MByte address space defined for program memory. The region size ranges from 64 Bytes to 256 kBytes with a step size of power of 2. If a region size of 0 is selected, the region is disabled. To support overlay memory which has not the same size as the program memory, an overlay startaddress which is 22 bit wide, can be specified. The size of the overlay region is the same as the program memory region. The startaddress of both program memory and overlay memory region have to be a multiple of the programmed region size. When the address of the access falls into one of the programmed regions, the POM will start requesting the data from the overlay memory address, based on the overlay startaddress. If the address falls into multiple overlapping regions, the region with the lowest number has highest priority and only one read request from overlay memory will be initiated. This avoids that regions with the same program memory address, but different overlay memory addresses, request different data.

Figure 11-2. Region definition example



11.4 Control Registers

11.4.1 Control registers

This section describes the Parameter Overlay Module registers. The registers support only 32-bit writes. The offset is relative to the associated peripheral select.

Table 11-1. POM Registers

Offset Address Register[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 POMGLBCTRL Page 383	Reserved															
	Reserved												ON/OFF			
0x04 POMREV Page 384	SCHEME		Reserved		FUNC											
	RTL					MAJOR			CUSTOM		MINOR					
0x08 Reserved Page 385	Reserved															
	Reserved															DNM
0x200, 0x210, ..., POMPRGSTARTx[a] Page 385	Reserved										STARTADDRESS[21:16]					
	STARTADDRESS[15:0]															
0x204, 0x214, ..., POMOVLSTARTx Page 387	Reserved										STARTADDRESS[21:16]					
	STARTADDRESS[15:0]															
0x208, 0x218, ..., POMREGSIZEx Page 388	Reserved															
	Reserved												SIZE			
0xF00 POMITCTRL Page 389	Reserved															
	Reserved															

Table 11-1. POM Registers (Continued)

0xFA0 POMCLAIMSET Page 390	Reserved		
	Reserved	SET1	SET0
0xFA4 POMCLAIMCLR Page 391	Reserved		
	Reserved	CLR1	CLR0
0xFB0 POMLOCKACCESS Page 391	Reserved		
	Reserved		
0xFB4 POMLOCKSTATUS Page 393	Reserved		
	Reserved		
0xFB8 POMAUTHSTATUS Page 394	Reserved		
	Reserved		
0xFC8 POMDEVID Page 395	Reserved		
	Reserved		
0xFCC POMDEVTYPE Page 396	Reserved		
	Reserved	Sub Type	Major Type
0xFD0 POMPERIPHERALID4 Page 397	Reserved		
	Reserved	4KB Count	JEP106 Continuation Code
0xFD4 POMPERIPHERALID5 Page 398	Reserved		
	Reserved		

Table 11-1. POM Registers (Continued)

0xFD8 POMPERIPHERALI D6 Page 399	Reserved			
	Reserved			
0xFDC POMPERIPHERALI D7 Page 400	Reserved			
	Reserved			
0xFE0 POMPERIPHERALI D0 Page 401	Reserved			
	Reserved	Part Number		
0xFE4 POMPERIPHERALI D1 Page 402	Reserved			
	Reserved	JEP106 Identity Code	Part Number	
0xFE8 POMPERIPHERALI D2 Page 403	Reserved			
	Reserved	Revision	JEDEC	JEP106 Identity Code
0xFEC POMPERIPHERALI D3 Page 404	Reserved			
	Reserved			
0xFF0 POMCOMPONENTI D0 Page 405	Reserved			
	Reserved	Preamble		
0xFF4 POMCOMPONENTI D1 Page 406	Reserved			
	Reserved	Component Class	Preamble	
0xFF8 POMCOMPONENTI D2 Page 407	Reserved			
	Reserved	Preamble		

Table 11-1. POM Registers (Continued)

0xFFC POMCOMPONENTID3 Page 408	Reserved	
	Reserved	Preamble

a. x = 0..31

11.4.2 POM Global Control Register (POMGLBCTRL)

This register contains a key to enable the POM module. Logic remains reset until this key is set.

Figure 11-3. POM Global Control Register (POMGLBCTRL) [offset = 0x00h]

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
Reserved												ON/OFF			
R-0												R/WP-0101			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-2. POM Global Control Register (POMGLBCTRL) Field Descriptions

Bit	Name	Value	Description
31-4	Reserved		Reads return 0 and writes have no effect.
3-0	ON/OFF		Turn functionality of POM on or off.
		all other	POM is held in reset. NOTE: The key should be written to 0101, to avoid single bit flips inadvertently turning on the module.
		1010	POM is functional.

11.4.3 POM Revision ID (POMREV)

This register contains the revision ID of the POM module.

Figure 11-4. POM Revision ID (POMREV) [offset = 0x04h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="2">SCHEME</td><td colspan="2">Reserved</td><td colspan="12">FUNC</td></tr>
<tr><td colspan="2">R-0x1</td><td colspan="2">R-0</td><td colspan="12">R-0x0A03</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">RTL</td><td colspan="3">MAJOR</td><td colspan="2">CUSTOM</td><td colspan="6">MINOR</td></tr>
<tr><td colspan="5">R-0x0</td><td colspan="3">R-0x1</td><td colspan="2">R-0x0</td><td colspan="6">R0x4</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-3. POM Revision ID (POMREV) Field Descriptions

Bit	Name	Value	Description
31–30	SCHEME	0x1	Used to distinguish between different ID schemes.
29-28	Reserved		Reads return 0 and writes have no effect.
27-16	FUNC	0x0A03	Indicates the SW compatible module family
15-11	RTL	0x0	RTL version number
10-8	MAJOR	0x1	Major revision number
7-6	CUSTOM	0x0	Indicates a device specific implementation
5-0	MINOR	0x4	Minor revision number

11.4.4 Reserved

This register is for TI internal use only.

Figure 11-5. Reserved [offset = 0x08h]

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
Reserved															DNM
R-0															R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-4. POM Clock Gate Control Register (POMCLKCTRL) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and writes have no effect.
0	DNM		Do not modify this bit. Leave it in its reset state. Modifying the bit while the POM module is switched on can result in unexpected behavior.

11.4.5 POM Program Region Start Address Register x (POMPROGSTARTx)

This register contains the start address of the region in program memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 11-6. POM Program Region Start Register x (POMPROGSTARTx) [offset = 0x200, 0x210, ... 0x3F0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										STARTADDRESS[21:16]					
R-0										R/WP-0					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-5. POM Program Region Start Address Register x (POMPROGSTARTx) Field Descriptions

Bit	Name	Value	Description
31–22	Reserved		Reads return 0 and writes have no effect.
21-0	STARTADDRESS		Defines the startaddress of the program memory region. The startaddress has to be a multiple of the region size.

NOTE: If the region start address is programmed to a non-region size boundary, the region will begin at the next lower region size boundary.

11.4.6 POM Overlay Region Start Address Register x (POMOVLSTARTx)

This register contains the start address of the region in overlay memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 11-7. POM Overlay Region Start Register x (POMOVLSTARTx) [offset = 0x204, 0x214, ..., 0x3F4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										STARTADDRESS[21:16]					
R-0										R/WP-0					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-6. POM Overlay Region Start Address Register x (POMOVLSTARTx) Field Descriptions

Bit	Name	Value	Description
31–22	Reserved		Reads return 0 and writes have no effect.
21-0	STARTADDRESS		Defines the startaddress of overlay memory region. The startaddress has to be a multiple of the region size.

NOTE: If the region start address is programmed to a non-region size boundary, the region will begin at the next lower region size boundary.

11.4.7 POM Region Size Register x (POMREGSIZEx)

This register contains the region size for both program memory and overlay memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 11-8. POM Region Size Register x (POMREGSIZEx) [offset = 0x208, 0x218, ..., 0x3F8]

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
Reserved												SIZE			
R-0												R/WP-0			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-7. POM Region Size Register x (POMREGSIZEx) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect.
3-0	SIZE	0000b 0001b 0010b : 1101b 1110b 1111b	Region disabled 64 Bytes 128 Bytes : 256 kBytes Reserved Reserved

NOTE: If the region is enabled by writing a non-zero value to the SIZE bitfield, it will take up to 3 VCLK cycles until the write takes effect. If during this time an access to the programmed region is taking place and the POM is already enabled in the POMGLBCTRL register, it either decodes the previous setting of the SIZE field or the access will be directed to the program memory when the region was disabled.

11.4.8 POM Integration Control Register (POMITCTRL)

This is a CoreSight register and is for debug purpose only. The integration functionality is not implemented. The register reads 0x00000000.

Figure 11-9. POM Integration Control Register (POMITCTRL) [offset = 0xF00]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-8. POM Integration Control Register (POMITCTRL) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.9 POM Claim Set Register (POMCLAIMSET)

This is a CoreSight register and is for debug purpose only. This register allows different masters to claim the control for the POM module. There is no control functionality for POM register accesses implemented. The only functionality of these bits is to indicate that another master is controlling the module.

Figure 11-10. POM Claim Set Register (POMCLAIMSET) [offset = 0xFA0]

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
Reserved														SET1	SET0
R-0														R/WP-1	R/WP-1

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-9. POM Claim Set Register (POMCLAIMSET) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and writes have no effect.
1	SET1		The module is claimed
		0	Read: This claim tag bit is not implemented Write: no effect
		1	Read: This claim tag is implemented Write: Set claim tag
0	SET0		The module is claimed
		0	Read: This claim tag bit is not implemented Write: no effect
		1	Read: This claim tag is implemented Write: Set claim tag

11.4.10 POM Claim Clear Register (POMCLAIMCLR)

This is a CoreSight register and is for debug purpose only. This register allows different masters to claim the control for the POM module. There is no control functionality for POM register accesses implemented. The only functionality of these bits is to indicate that another master is controlling the module.

Figure 11-11. POM Claim Clear Register (POMCLAIMCLR) [offset = 0xFA4]

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
Reserved														CLR1	CLR0
R-0														R/WP-1	R/WP-1

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-10. POM Claim Clear Register (POMCLAIMCLR) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and writes have no effect.
1	CLR1		The module is claimed Read: Current claim tag value Write 0: no effect Write 1: Clear claim tag
0	CLR0		The module is claimed Read: Current claim tag value Write 0: no effect Write 1: Clear claim tag

11.4.11 POM Lock Access Register (POMLOCKACCESS)

This is a CoreSight register and is for debug purpose only. The register reads 0x00000000.

Figure 11-12. POM Lock Access Register (POMLOCKACCESS) [offset = 0xFB0]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-11. POM Lock Access Register (POMLOCKACCESS) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.12 POM Lock Status Register (POMLOCKSTATUS)

This is a CoreSight register and is for debug purpose only. The register reads 0x00000000.

Figure 11-13. POM Lock Status Register (POMLOCKSTATUS) [offset = 0xFB4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-12. POM Lock Status Register (POMLOCKSTATUS) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.13 POM Authentication Status Register (POMAUTHSTATUS)

This is a CoreSight register and is for debug purpose only. The register reads 0x00000000.

Figure 11-14. POM Authentication Status Register (POMAUTHSTATUS) [offset = 0xFB8]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-13. POM Authentication Status Register (POMAUTHSTATUS) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.14 POM Device ID Register (POMDEVID)

This is a CoreSight register and is for debug purpose only. The register reads 0x00000000.

Figure 11-15. POM Device ID Register (POMDEVID) [offset = 0xFC8]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-14. POM Device ID Register (POMDEVID) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.15 POM Device Type Register (POMDEVTYPE)

This is a CoreSight register and is for debug purpose only. The device type register defines the CoreSight module class.

Figure 11-16. POM Device Type Register (POMDEVTYPE) [offset = 0xFCC]

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
Reserved								Sub Type				Major Type			
R-0								R-0x0				R-0x4			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-15. POM Device Type Register (POMDEVTYPE) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-4	Sub Type	0x0	Other
3-0	Major Type	0x4	Debug Control

11.4.16 POM Peripheral ID 4 Register (POMPERIPHERALID4)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 11-17. POM Peripheral ID 4 Register (POMPERIPHERALID4) [offset = 0xFD0]

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
Reserved								4KB Count				JEP106 Continuation Code			
R-0								R-0x0				R-0x0			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-16. POM Peripheral ID 4 Register (POMPERIPHERALID4) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-4	4KB Count	0x0	only 4KB implemented
3-0	JEP Continuation Code	0x0	JEP106 Code

11.4.17 POM Peripheral ID 5 Register (POMPERIPHERALID5)

This is a CoreSight register and is for debug purpose only. This register reads 0x00000000.

Figure 11-18. POM Peripheral ID 5 Register (POMPERIPHERALID5) [offset = 0xFD4]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-17. POM Peripheral ID 5 Register (POMPERIPHERALID5) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.18 POM Peripheral ID 6 Register (POMPERIPHERALID6)

This is a CoreSight register and is for debug purpose only. This register reads 0x00000000.

Figure 11-19. POM Peripheral ID 6 Register (POMPERIPHERALID6) [offset = 0xFD8]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-18. POM Peripheral ID 6 Register (POMPERIPHERALID6) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.19 POM Peripheral ID 7 Register (POMPERIPHERALID7)

This is a CoreSight register and is for debug purpose only. This register reads 0x00000000.

Figure 11-20. POM Peripheral ID 7 Register (POMPERIPHERALID7) [offset = 0xFDC]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-19. POM Peripheral ID 7 Register (POMPERIPHERALID7) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.20 POM Peripheral ID 0 Register (POMPERIPHERALID0)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 11-21. POM Peripheral ID 0 Register (POMPERIPHERALID0) [offset = 0xFE0]

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
Reserved								Part Number							
R-0								R-0							

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-20. POM Peripheral ID 0 Register (POMPERIPHERALID0) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-0	Part Number	0x0	Reads 0, since POMREV defines the module

11.4.21 POM Peripheral ID 1 Register (POMPERIPHERALID1)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 11-22. POM Peripheral ID 1 Register (POMPERIPHERALID1) [offset = 0xFE4]

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
Reserved								JEP106 Identity Code				Part Number			
R-0								R-0x7				R-0			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-21. POM Peripheral ID 1 Register (POMPERIPHERALID1) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-4	JEP106 Identity Code	0x7	Part of TI JEDEC number
3-0	Part Number	0x0	Reads 0, since POMREV defines the module

11.4.22 POM Peripheral ID 2 Register (POMPERIPHERALID2)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 11-23. POM Peripheral ID 2 Register (POMPERIPHERALID2) [offset = 0xFE8]

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
Reserved								Revision				JEDEC	JEP106 Identity Code		
R-0								R-0x0				R-1	R-0x1		

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-22. POM Peripheral ID 2 Register (POMPERIPHERALID2) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-4	Revision	0x0	Reads 0, since POMREV defines the module
3	JEDEC	0x1	Indicates JEDEC assigned value
2-0	JEP106 Identity Code	0x1	JEDEC+JEP106 Identity Code (POMPERIPHERALID2)+JEP106 Identity Code (POMPERIPHERALID1) form TI JEDEC ID of 0x97

11.4.23 POM Peripheral ID 3 Register (POMPERIPHERALID3)

This is a CoreSight register and is for debug purpose only. This register reads 0x00000000.

Figure 11-24. POM Peripheral ID 3 Register (POMPERIPHERALID3) [offset = 0xFEC]

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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-23. POM Peripheral ID 3 Register (POMPERIPHERALID3) Field Descriptions

Bit	Name	Value	Description
31–0	Reserved		Reads return 0 and writes have no effect.

11.4.24 POM Component ID 0 Register (POMCOMPONENTID0)

This is a CoreSight register and is for debug purpose only.

Figure 11-25. POM Component ID 0 Register (POMCOMPONENTID0) [offset = 0xFF0]

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
Reserved								Preamble							
R-0								R-0x0D							

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-24. POM Component ID 0 Register (POMCOMPONENTID0) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-0	Preamble	0x0D	Preamble

11.4.25 POM Component ID 1 Register (POMCOMPONENTID1)

This is a CoreSight register and is for debug purpose only.

Figure 11-26. POM Component ID 1 Register (POMCPOMPONENTID1) [offset = 0xFF4]

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
Reserved								Component Class				Preamble			
R-0								R-0x9				R-0x0			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-25. POM Component ID 1 Register (POMCOMPONENTID1) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-4	Component Class	0x9	CoreSight Component
3-0	Preamble	0x0	Preamble

11.4.26 POM Component ID 2 Register (POMCOMPONENTID2)

This is a CoreSight register and is for debug purpose only.

Figure 11-27. POM Component ID 2 Register (POMCPOMPONENTID2) [offset = 0xFF8]

31–16
Reserved
R-0

15–8	7–0
Reserved	Preamble
R-0	R-0x05

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-26. POM Component ID 2 Register (POMCOMPONENTID2) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-0	Preamble	0x05	Preamble

11.4.27 POM Component ID 3 Register (POMCOMPONENTID3)

This is a CoreSight register and is for debug purpose only.

Figure 11-28. POM Component ID 3 Register (POMCPOMPONENTID3) [offset = 0xFFC]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">Preamble</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R-0xB1</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset; -x = indeterminate;

Table 11-27. POM Component ID 3 Register (POMCOMPONENTID3) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return 0 and writes have no effect.
7-0	Preamble	0xB1	Preamble

Chapter 12

SPNU489C-February 2012

General-Purpose Input/Output (GIO) Module

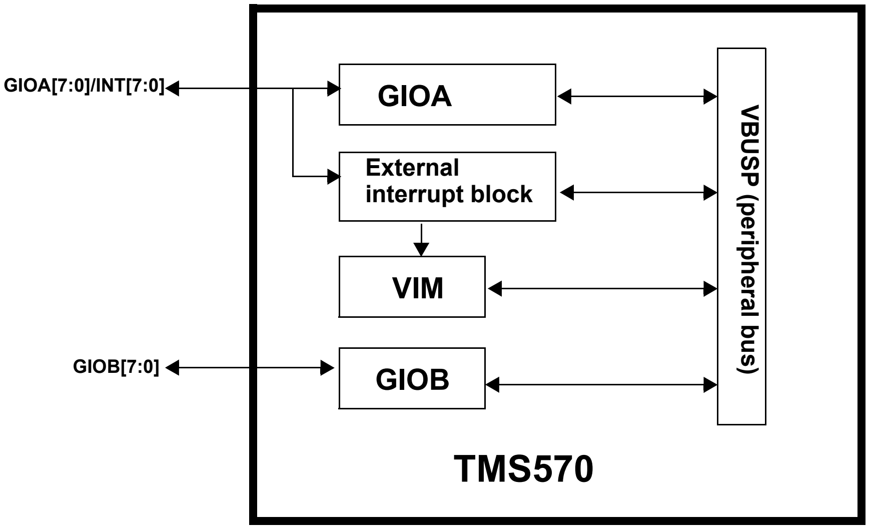
The general-purpose input/output (GIO) module provides the TMS570 family of devices with input/output (I/O) capability. The I/O pins are bidirectional and bit-programmable. The GIO module also supports external interrupt capability.

Topic Page

12.1 Overview

The GIO module on this device supports two ports, GIOA and GIOB. Each module has 8 I/O pins. The I/O pins are bidirectional and bit-programmable. The GIOA module also supports external interrupt capability.

Figure 12-1. Device Level Overview



The GIO module has the following features:

- Each I/O pin is controlled by bits in these registers:
  - Data direction (GIODIR)
  - Data input (GIODIN)
  - Data output (GIODOUT)
  - Data set (GIODSET)
  - Data clear (GIODCLR)
  - Open drain (GIOPDR)
  - Pull disable (GIOPULDIS)
  - Pull select (GIOPSL)
- The interrupts have the following characteristics:
  - Programmable interrupt detection either on both edges or on a single edge (set in GIOINTDET)
  - Programmable edge-detection polarity, either rising or falling edge (set in GIOPOL register)
  - Individual interrupt flags (set in GIOFLG register)
  - Individual interrupt enables, set and cleared through GIOENASET and GIOENACLR registers respectively
  - Programmable interrupt priority, set through GIOLVLSET and GIOLVLCLR registers
- Internal pullup/pulldown allows unused I/O pins to be left unconnected.

12.2 Quick Start Guide

The GIO module comprises two separate components: an input/output (I/O) block and an external interrupt block. Figure 12-2 and Figure 12-3 shows what the user should do after power reset to configure the GIO module as I/O and external interrupt.

Figure 12-2. I/O Function Quick Start Flow Chart

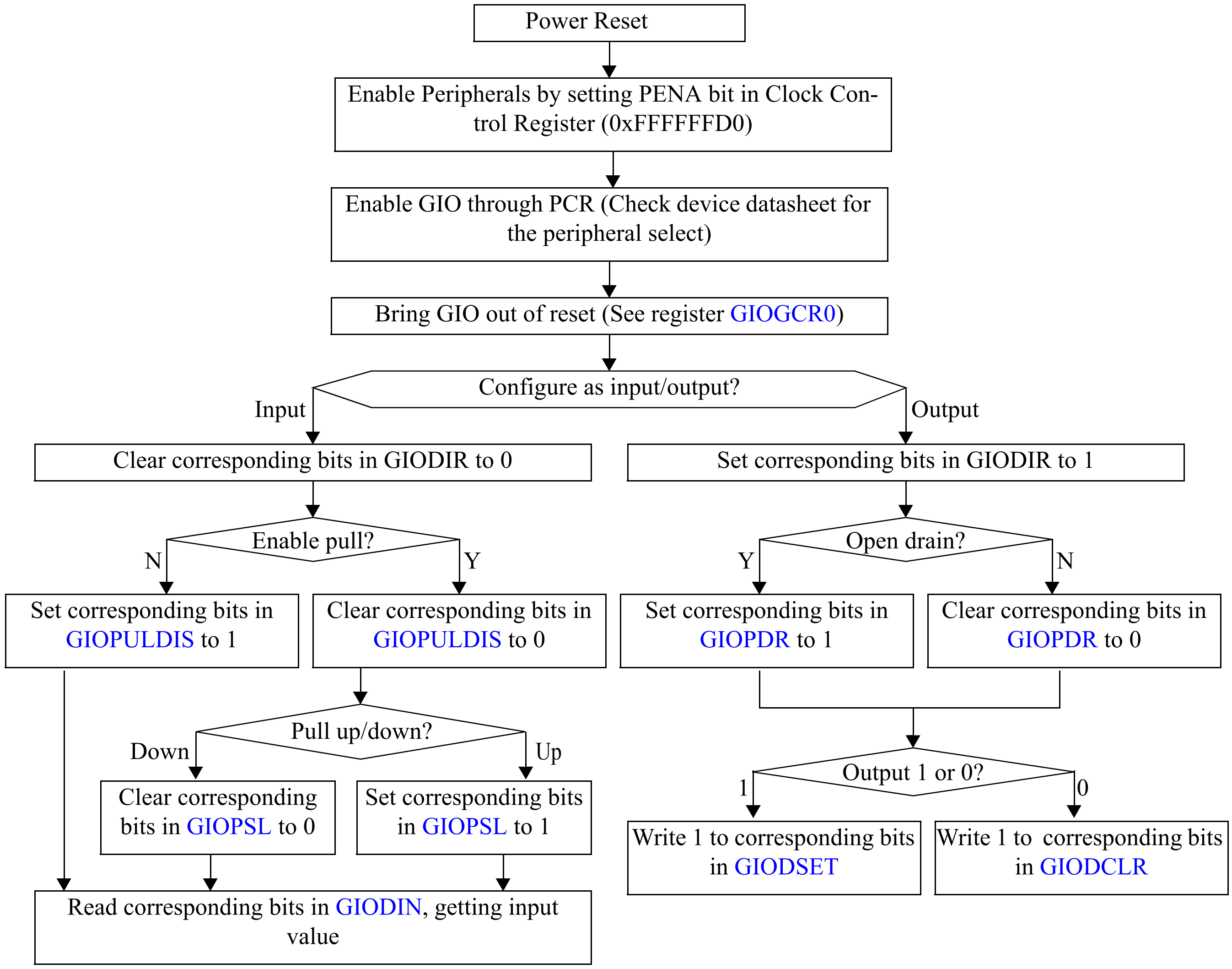
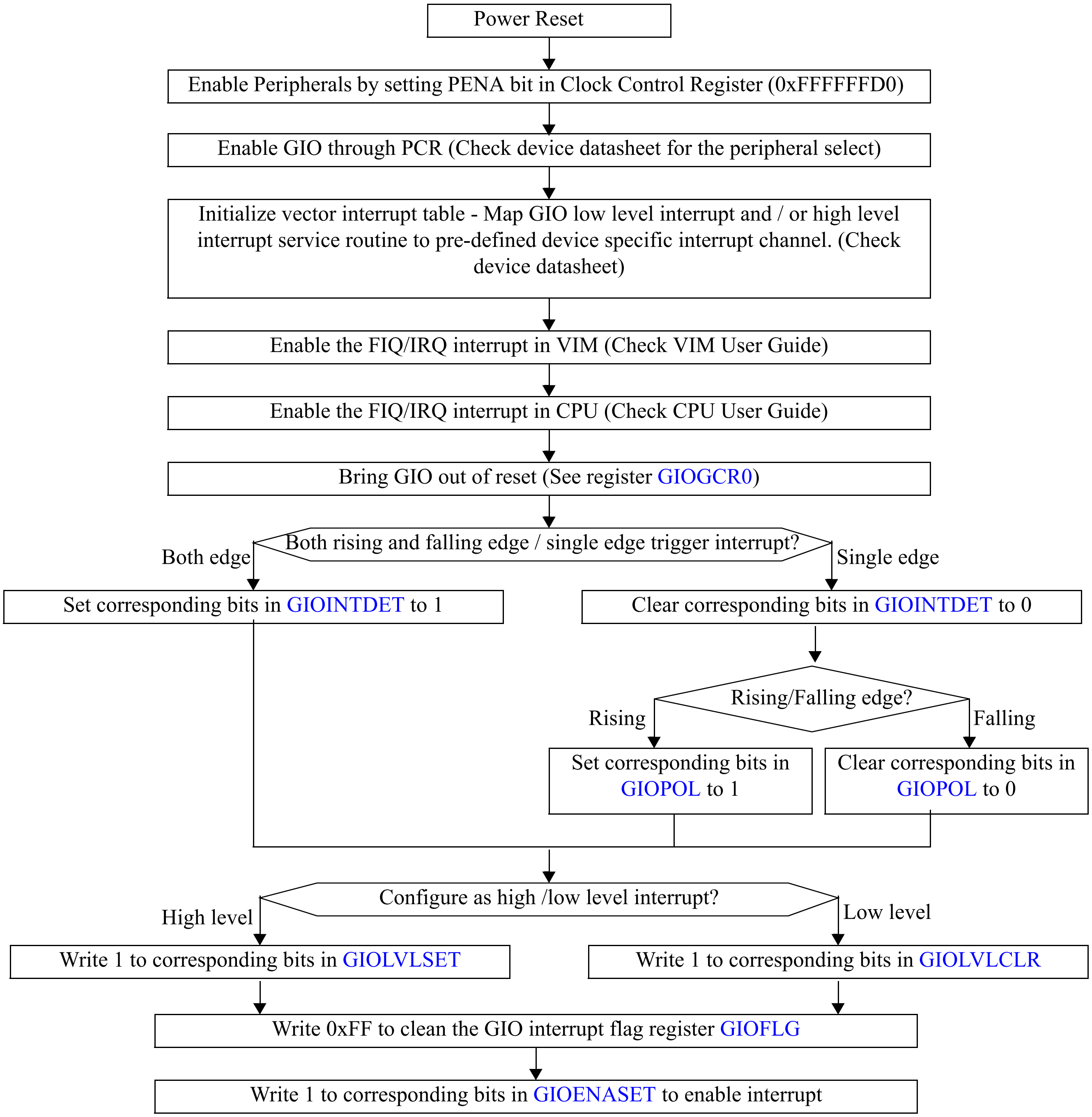

Power Reset
Enable Peripherals by setting PENA bit in Clock Control Register (0xFFFFFFD0)
Enable GIO through PCR (Check device datasheet for the peripheral select)
Bring GIO out of reset (See register GIOGCR0)
Configure as input/output?
Input
Output
Clear corresponding bits in GIODIR to 0
Set corresponding bits in GIODIR to 1
Enable pull?
N
Y
Open drain?
Y
N
Set corresponding bits in GIOPULDIS to 1
Clear corresponding bits in GIOPULDIS to 0
Set corresponding bits in GIOPDR to 1
Clear corresponding bits in GIOPDR to 0
Pull up/down?
Down
Up
Clear corresponding bits in GIOPSL to 0
Set corresponding bits in GIOPSL to 1
Output 1 or 0?
1
0
Write 1 to corresponding bits in GIODSET
Write 1 to corresponding bits in GIODCLR
Read corresponding bits in GIODIN, getting input value

Figure 12-3. External Interrupt Function Quick Start Flow Chart

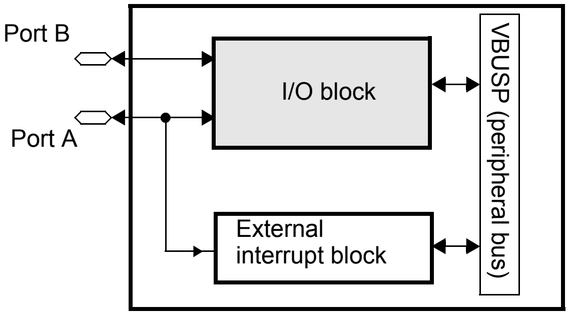

Power Reset
Enable Peripherals by setting PENA bit in Clock Control Register (0xFFFFFFD0)
Enable GIO through PCR (Check device datasheet for the peripheral select)
Initialize vector interrupt table - Map GIO low level interrupt and / or high level interrupt service routine to pre-defined device specific interrupt channel. (Check device datasheet)
Enable the FIQ/IRQ interrupt in VIM (Check VIM User Guide)
Enable the FIQ/IRQ interrupt in CPU (Check CPU User Guide)
Bring GIO out of reset (See register GIOGCR0)
Both rising and falling edge / single edge trigger interrupt?
Both edge
Single edge
Set corresponding bits in GIOINTDET to 1
Clear corresponding bits in GIOINTDET to 0
Rising/Falling edge?
Rising
Falling
Set corresponding bits in GIOPOL to 1
Clear corresponding bits in GIOPOL to 0
Configure as high /low level interrupt?
High level
Low level
Write 1 to corresponding bits in GIOLVLSET
Write 1 to corresponding bits in GIOLVLCLR
Write 0xFF to clean the GIO interrupt flag register GIOFLG
Write 1 to corresponding bits in GIOENASET to enable interrupt

In GIO interrupt service routine, user shall read the GIO offset register GIOOFFA / GIOOFFB (depending on high/low level interrupt) to clear the flag and find the pending interrupt GIO channel.

12.3 Functional Description of GIO Module

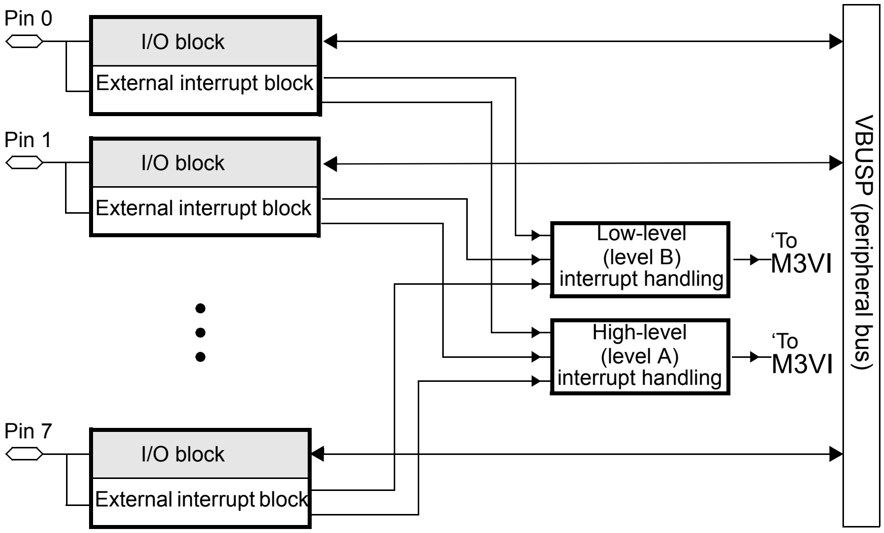
The GIO module (see Figure 12-4) comprises two separate components: an input/output (I/O) block and an external interrupt block. The GIO module has 2 ports (A and B) with 8 pins (0 through 7) per port. The pins for port B handle the standard I/O functions only; see Figure 12-4. Port A can handle interrupts in addition to the standard I/O functions.

Figure 12-4. GIO Module Diagram



The pins on port A, shown in Figure 12-5, are all interrupt-capable pins and can be used to handle either general I/O functions or external interrupt signals. The pins are connected to both an I/O block and an external interrupt block. Each of the eight I/O blocks is connected to the VBUSP bus, whereas the interrupt blocks are physically attached to a single high-level-interrupt-handling block and to a single low-level-interrupt-handling block. The high-level-interrupt-handling block, which is also denoted as level A, and the low-level-interrupt-handling block, which is also denoted as level B, each send one signal to the vectored interrupt manager (M3VIM) for processing. The interrupt priority of level A and level B interrupt handling blocks can be re-programmed in the M3VIM.

Figure 12-5. GIO Port A Module Diagram

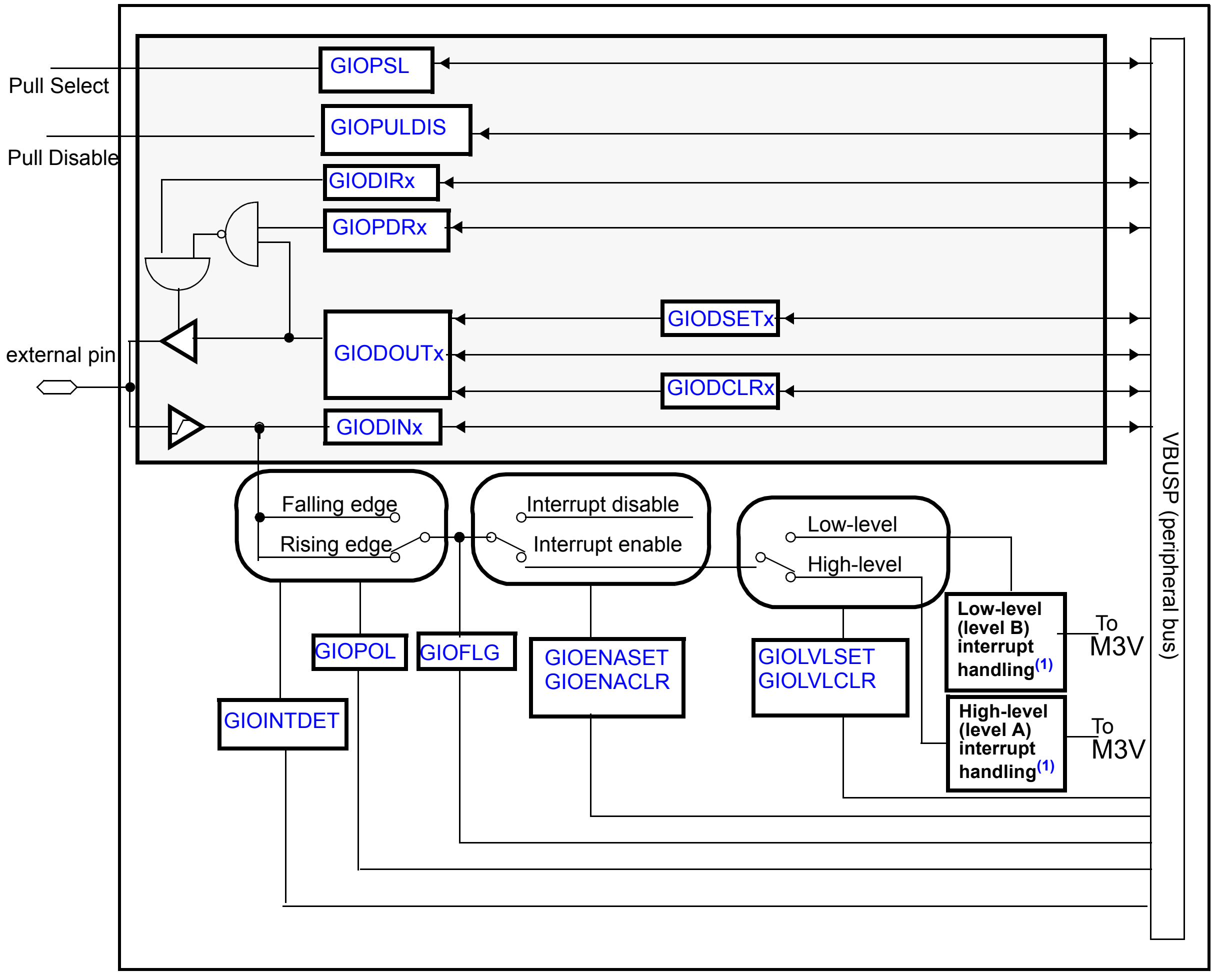


12.3.1 GIO Block Diagram

The GIO block diagram (see Figure 12-6) represents the flow of information through a pin. The shaded area corresponds to the I/O block; the unshaded area corresponds to the external interrupt block.

Because port B is not interrupt-capable, the block diagram for that port reduces to the shaded portion of the block diagram in Figure 12-6.

Figure 12-6. 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.

12.3.2 I/O Blocks

Each pin serviced by port A and B contains its individual I/O block.

12.3.2.1 I/O Function

The GIO module sends data to the external pin through the output buffer and receives data from the external pin through the input buffer (see Figure 12-6). The associated registers are:

- GIODIR[7:0] —Controls the direction that information is sent. The GIODIR[7:0] register determines whether or not values in the data output register are sent to the external pin. The input buffer is enabled except in the case when the pin direction is set as an input (GIODIR[7:0]) AND the pull control is disabled (GIOPULDIS[7:0]) AND pull down is selected as the pull bias (GIOPSL[7:0]). Refer to Table 12-1. When the input buffer is enabled, information that is sent to the external pin is also received in the input buffer.
  GIODOUT[7:0] —Controls what information is sent to the external pin when it is configured as an output. When the output buffer is enabled, writing values to the data output register (GIODOUT[7:0]) applies a voltage to the output pin. 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 forces the pin to a high output voltage (V_{OH} or higher) if the open drain functionality is disabled (GIOPDR[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).

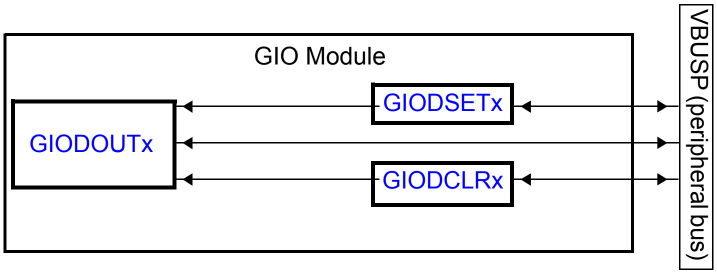
- GIODIN[7:0] —Receives the information from the external pin. A high voltage (V_{IH} or greater) applied to the pin causes a high value (1) in the data input register (GIODIN[7:0]). When a low voltage (V_{IL} or less) is applied to the pin, the data input register reads a low value (0). The V_{IH} and V_{IL} values are device specific and can be found in the device datasheet.
- GIOPDR[7:0] —Controls the open drain configuration of the pin. If the pin is set as open drain, a high value (1) written to the data output register (GIODOUT[7:0]) forces the pin to a high impedance state (z). Open drain functionality is enabled or disabled using the open drain register GIOPDR[7:0].
- GIOPULDIS[7:0] —Disables the pull control capability at pin. The pull functionality for the pin can be enabled or disabled in the GIOPULDIS[7:0] register.
- GIOPSL[7:0] —Selects the pull type at pin. Pull down or pull up can be selected using the GIOPSL[7:0] register.

12.3.2.2 Output Control Registers

When a GIO pin is configured as an output pin, the value in the data output register (GIODOUT[7:0]) specifies the voltage applied to the external pin. The GIO module provides three ways of communicating with the data output register (see Figure 12-7).

- The control register bit can be written directly by writing to the data output register (GIODOUT[7:0]).
- The data output register bit can be set to 1 using the data set register (GIODSET[7:0]).
- The data output register bit can be cleared to 0 using the data clear register (GIODCLR[7:0]).

Figure 12-7. Communication With the Data Output Register



The GIODSET[7:0] and GIODCLR[7:0] registers allow improved handling of data. The data set and data clear registers preclude any possibility of a read-modify-write (RMW) operation. RMW operations are possible when the CPU reads a register, performs some action (for example, an OR operation), and then writes the values back into the register. Under such conditions, it is possible that the contents of the original register (GIODOUT[7:0]) can change (for example, an interrupt procedure) between the time when the CPU originally reads the register and the time when the CPU writes the new value. If the contents of the register changes between the time the CPU reads the register and the time it writes the new value to that register, then the CPU has ended up using an outdated register value as the basis for its operations, and therefore generates an erroneous result from those operations that also ends up being written back into the register. RMW operations can be avoided by using the GIODSET[7:0] and GIODCLR[7:0] registers.

12.3.2.3 Pullup/Pulldown Function

GIO module pins can have either an active pullup or active pulldown that makes it possible to leave the pins unconnected externally when the pins are input pins. The pull capability is enabled by programming the GIOPULDIS[7:0] register. By enabling the pull capability, the default pull on the GIO pins, as indicated in the device datasheet, are enabled. The pins also have the capability to be programmed as either a pullup or a pulldown using the GIOPSL[7:0] register. Programming GIOPSL[7:0] overrides the default pull on the pins. See Table 12-1.

The default pullup/pulldown functionality of all GIO pins at system reset is pulldown.

Note:
It is possible to disable the GIO Input buffer altogether by configuring the pin direction to input via the GIODIR[A-B][7:0], disabling the pin's pull functionality using GIOPULDIS[7:0], and setting the pin's corresponding bit in the GIOPSL[7:0] to 0. Once an input buffer is disabled, the external signal can not be read from the buffer internally.

12.3.2.4 Open Drain Function

The GIO pins can be configured to include an open drain functionality using the GIOPDR[7:0] registers, when the pins are configured as output pins. When the open drain functionality is enabled (GIOPDR[7:0] = 1), a 0 written to the data output register GIODOUT[7:0] forces the pin to a low output voltage (V_{OL} or lower), whereas a 1 written to the data output register forces the pin to a high impedance state. The open drain functionality is disabled when the pin is configured as an input pin.

12.3.2.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 12-1.

Table 12-1. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

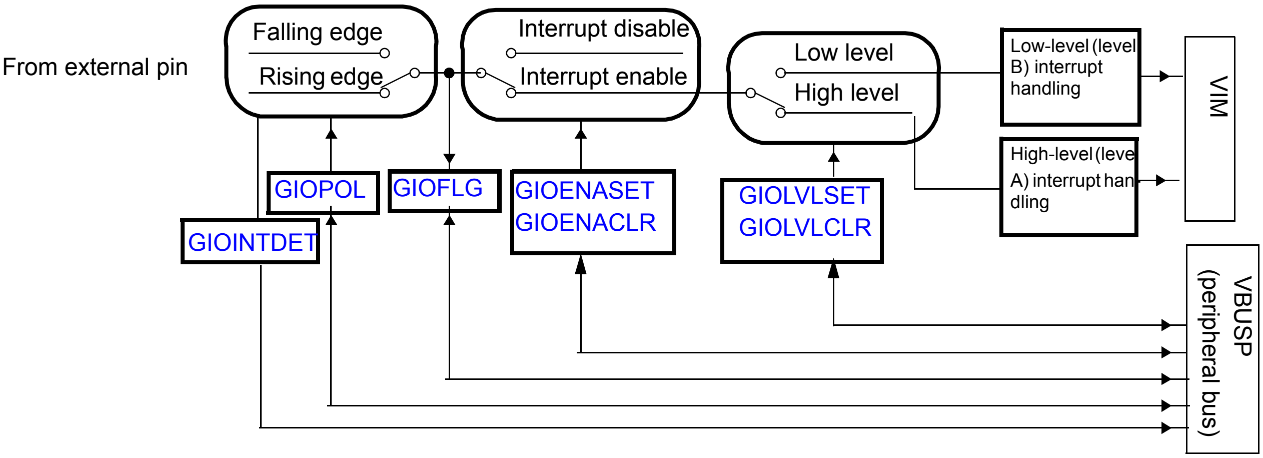
Device under Reset?	Pin Direction (GIODIR)	Pull Disable (GIOPULDIS)	Pull Select (GIOPSL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Depends on pull control
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Disabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

1 X = Don’t care
2 GIODIR = 0 for input, 1 for output
3 GIOPULDIS = 0 for enabling pull control
= 1 for disabling pull control
4 GIOPSL= 0 for pull-down functionality
= 1 for pull-up functionality
5 If open drain is enabled, output buffer will be disabled if output 1.

12.3.3 External Interrupt Block

Each interrupt-capable pin connects to the GIO module’s single low-level-interrupt-handling block and single high-level-interrupt-handling block. Depending on the priority, the interrupt signal is sent through the appropriate offset register to the vectored interrupt manager (M3VIM) in the system module (see Section 12.3.3.3). Figure 12-8 shows the external interrupt block.

Figure 12-8. External Interrupt Block



12.3.3.1 Edge Detection and the Flag Register

The edge-detection hardware and flag register, like the input buffer, are always enabled. The GIOINTDET register specifies whether interrupt detection is on both edges or on a single edge only. If interrupt detection is on a single edge only, then the GIOPOL register is used to define whether a rising edge or a falling edge is recognized as an interrupt.

A rising edge occurs when the voltage on a given pin transitions from a low value (V_{IL} or lower) to a high value (V_{IH} or higher). The voltage on the external pin must remain at the high level for at least one VCLK cycle to ensure recognition. (For an explanation of VCLK, please refer to TMS570 architecture user guide).

A falling edge occurs when the voltage on the external pin transitions from a high value (V_{IH} or higher) to a low value (V_{IL} or lower). The voltage on the external pin must remain at the low level for at least one VCLK cycle to ensure recognition. (GIOPOL behaves differently in a low-power state. See Section 12.4.2 for more information.)

The corresponding flag in the GIOFLG register is set when a transition appearing on the external pin matches the combination of edges chosen by the GIOINTDET and GIOPOL registers. For example, to set the flag on only a rising edge on pin 2 of GIO port A, clear the bit in the interrupt detection register (GIOINTDET[2] = 0), and set the bit in the polarity register (GIOPOL[2] = 1). Then, when the signal transition takes place, the GIO module will set the appropriate flag in the flag register (GIOFLG[2] = 1).

Note: Setting Flag With Interrupt Disabled

A flag can be set whether or not the interrupt is enabled. The flag register can then be polled instead of driving an interrupt. Additionally, the flag should not be set before enabling the interrupt; specifically, the flag register should be cleared before enabling the interrupt.

The edge-detection hardware responds to voltages on the external pin and does not discriminate between the source of these voltages. Therefore, the interrupt will respond to the correct edge even when generated from a pin whose output buffer is enabled.

12.3.3.2 Interrupts and Interrupt Levels

The interrupt flag is set when an edge transition on the external pin is detected and matches the edge as chosen by the GIOINTDET and GIOPOL registers. An interrupt can be generated from the set flag if the interrupt is enabled (see Figure 12-8).

The external interrupt can be enabled or disabled using the GIOENASET and GIOENACLR register bits respectively. These two registers are physically implemented as a single register. If the interrupt is enabled, the signal with an appropriate edge leads to an interrupt. If multiple interrupts occur simultaneously, the GIO module must prioritize the interrupts so that they can be handled in the proper order.

The order in which simultaneous GIO interrupts are processed is determined by the following criteria:

1. The GIO priority control registers (GIOLVLSET and GIOLVLCLR) provide a software-implemented prioritization scheme. Each pin can be set as either a high-level (level A) or low-level interrupt (level B). By default in the Vectored Interrupt Manager (M3VIM), interrupts with level A are higher priority than those of level B and therefore are serviced first. However, the priority of level A and level B can be re-programmed in the M3VIM. Refer to the M3VIM module user's guide. The handling of interrupts with the same priority is determined by the interrupt with the lowest bit value having the highest priority. This prioritization is hardwired into the module.

Note: Wakeup Condition
GIOA interrupts are also used to awaken the device from the low power modes. The wakeup interrupt is level-based rather than edge-based. Please refer to VIM user guide for wakeup configuration.

Table 12-2 shows an example of how interrupt priorities are determined. Four interrupts occur simultaneously on GIO port A pins 3–0, and are handled as following:

1. The interrupts on pin 1 and pin 0 are both sent to the Vectored Interrupt Manager (M3VIM) for servicing because they have the lowest bit values among the high-level and low-level interrupts. Assuming that the M3VIM is set to service level A interrupt priority before level B interrupt priority, pin 1 is serviced first because it is the high-priority interrupt of the two.
2. Next, the interrupts on pin 3 and pin 0 are sent to the M3VIM for servicing. The interrupt on pin 3 is serviced because it is the only remaining high-priority interrupt.
3. The interrupt on pin 0 is serviced third because it has the lowest bit value.

The interrupt on pin 2 is serviced last because it is the only remaining interrupt.

Table 12-2. Determining Interrupt Priority

Pin (bit)	3	2	1	0
Priority	High Level (Level A)	Low Level (Level B)	High Level (Level A)	Low Level (Level B)
Order	2	4	1	3

12.3.3.3 High-Level-Interrupt Block and Low-Level-Interrupt Block

The interrupt-handling blocks each contain two registers: an offset register and an emulation register. The high-level interrupts are denoted as level A and consequently, the registers for the high-level-interrupt-handling block are GIOOFFA and GIOEMUA; see Figure 12-9. Likewise, the registers for the low-level-interrupt-handling block (denoted as level B) are GIOOFFB and GIOEMUB; see Figure 12-10.

Figure 12-9. High-Level-Interrupt-Handling Block

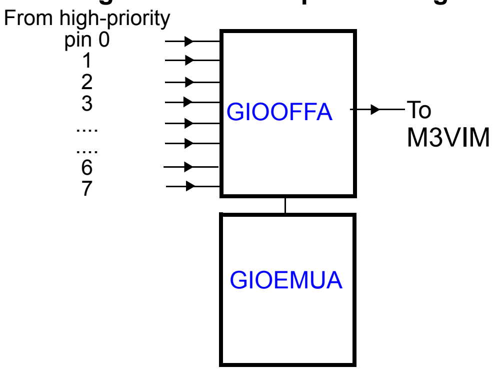
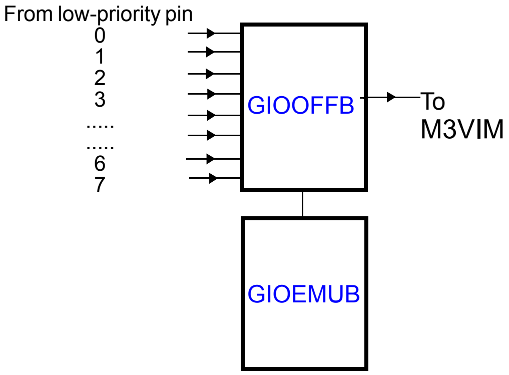


Figure 12-10. Low-Level-Interrupt-Handling Block



The read-only registers GIOOFFA and GIOOFFB generate a numerical offset value that represents the highest-priority pending external interrupt (see Table 12-3). The offset can be used to locate the position of the interrupt routine in the vector table. A read of the offset register clears the offset register and the corresponding flag bit in the GIOFLG register.

The high-level offset register, GIOOFFA, receives signals from each active interrupt that is configured as high-level. The GIOOFFA displays the high-level interrupt with the highest priority (that is, the interrupt that was generated by the lowest bit in the flag register). Similarly, the GIOOFFB displays the low-level interrupt with the highest priority.

The emulation control registers (GIOEMUA and GIOEMUB) mirror the offset registers. The emulation registers contain a numerical offset value that represent the highest priority pending external interrupt (see Table 12-3). A read of the emulation registers does not clear any bit in any register. These registers allow the device emulator to read and display the offset register values without affecting interrupt execution.

Table 12-3. GIO Offset A Values and Corresponding Interrupt

GIOOFF(5–0)	Pending External Interrupt
000000	No interrupt
000001	Interrupt 0
000010	Interrupt 1
000011	Interrupt 2
000100	Interrupt 3
. . .	. . .
001000	Interrupt 7

GIOOFF(5–0)	Pending External Interrupt
0001001–111111	Reserved

Table 12-4 illustrates how to identify the interrupt being serviced. In this example, interrupts are enabled for pins 0, 1 and 2 and all three pins are set to the same interrupt level (same priority). This example assumes that interrupts on pins 0 and 2 occur simultaneously. Reading the flag control register, GIOFLG, returns a value of xxxxx101, indicating that interrupt flags have been set for pins 0 and 2. The first five values are indeterminate because only pins 0, 1 and 2 are interrupt-enabled. Reading the offset B control register, GIOOFFB, returns a value of 00000001, indicating that the interrupt on pin 0 is currently being processed (see Table 12-3).

Table 12-4. Reading the Offset Register to Determine Serviced Interrupt

Bits	7	6	5	4	3	2	1	0
GIOENA	Reserved					1	1	1
GIOPOL	Reserved					1	0	0
GIOFLG	Reserved					1	0	1
GIOLVL	Reserved					0	0	0
GIOOFFA	0	0	0	0	0	0	0	0
GIOEMUA	0	0	0	0	0	0	0	0
GIOOFFB	0	0	0	0	0	0	0	1
GIOEMUB	0	0	0	0	0	0	0	1

12.3.3.4 Special Considerations for Interrupts

Please note that interrupts are subject to the following special considerations:

- To use the enabled pins as interrupts, the I/O function of the pins are typically set as input. If the pin's I/O function is set as output, the signal feeds directly into the input. In this case, interrupts are only generated when the device toggles the data output register, thereby creating the appropriate interrupt edge. See section 12.6.1, *Example: Setting Interrupts and Configuring Pins for Output*, page 445.
- The interrupt flag can be set even though the interrupt is not enabled. Therefore, you must clear the flag register before enabling the interrupts to ensure that a spurious interrupt is not generated. See section 12.6.3, *Example: Clearing Interrupt Flags and Setting Interrupts*, page 447.
- If interrupts are enabled when GIODIN[7:0] is read, the bits corresponding to the enabled interrupts must be masked to avoid ambiguous results. For example, if GIO port A pins [2:0] are configured as interrupts and pins [7:3] are configured as inputs (V_{IH} applied), then a read of GIODIN0 will read 11111xxx, where the x values are interrupt levels and not inputs. Mask the input register against (in this case) 11111000. See section 12.6.4, *Example: Reading Port B Input Register*, page 447.

12.4 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)
  - Module level power-down
  - Device level power-down

12.4.1 Emulation Mode

Emulation mode is used by debugger tools to stop the CPU at breakpoints to read registers. When the device is in emulation mode, it pulls the suspend signal high.

Note: Emulation Mode and Emulation Registers
Emulation mode is a mode of operation of the device and is separate from the GIO emulation registers (GIOEMUA and GIOEMUB). The contents of these emulation registers are identical to the contents of GIO offset registers (GIOOFFA and GIOOFFB). 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 M3VIM is unable to service interrupts.
- Any register can be read without affecting the state of the system.
- A write to a register affects the state of the system.

12.4.2 Power-Down Mode (Low-Power Mode)

In the 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. The GIO module has two power-down modes: module-level power down and device-level power down. In both these power-down modes (low-power modes), interrupt pins become level-sensitive rather than edge-sensitive. The polarity bit changes function from falling edge and rising edge to low and high. A corresponding level on an interrupt pin pulls the module out of low-power mode.

12.4.2.1 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 Register for details.

12.4.2.2 Device-Level Power Down

The entire device can be placed in one of the pre-defined low-power modes: doze, snooze, or sleep. See the device datasheet for details.

12.5 GIO Control Registers

Table 12-5 shows the summary of the GIO registers. It has two ports designated A–B (GIOA is interrupt capable ports), with eight pins per port. As an example, port B bits [2:0] of the GIODIR register would be represented as GIODIR[B][2:0]. Each port has a set of registers that control the I/O function of the pins for that port - they are GIODIR[A-B][7:0], GIODIN[A-B][7:0], GIODOUT[A-B][7:0], GIODSET[A-B][7:0], GIODCLR[A-B][7:0], GIOPDR[A-B][7:0], GIOPULDIS[A-B][7:0], and GIOPSL[A-B][7:0].

The registers are accessible in 8-, 16-, and 32-bit reads or writes. Consult the device-specific data sheet to verify the pin configuration. The start address for the GIO module is 0xFFF7BC00.

Table 12-5. GIO Control Register Summary

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
0xFFF7BC00 GIOGCR0 Page 425	Reserved															
	Reserved															RESET
0xFFF7BC08 GIOINTDET Page 426	Reserved															
	Reserved								GIOINTDET 0[7:0]							
0xFFF7BC0C GIOPOL Page 427	Reserved															
	Reserved								GIOPOL 0[7:0]							
0xFFF7BC10 GIOENASET Page 428	Reserved															
	Reserved								GIOENASET 0[7:0]							
0xFFF7BC14 GIOENACLR Page 428	Reserved															
	Reserved								GIOENACLR 0[7:0]							
0xFFF7BC18 GIOLVLSET Page 430	Reserved															
	Reserved								GIOLVLSET 0[7:0]							
0xFFF7BC1C GIOLVLCLR Page 430	Reserved															
	Reserved								GIOENACLR 0[7:0]							

[1] See the specific device data sheet to verify the base address of the GIO registers.

Table 12-5. GIO Control Register Summary (Continued)

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
0xFFF7BC20 GIOFLG Page 432	Reserved															
	Reserved								GIOFLG 0[7:0]							
0xFFF7BC24 GIOOFFA Page 433	Reserved															
	Reserved										GIOOFFA[5:0]					
0xFFF7BC28 GIOOFFB Page 434	Reserved															
	Reserved										GIOOFFB[5:0]					
0xFFF7BC2C GIOEMUA Page 435	Reserved															
	Reserved										GIOEMUA[5:0]					
0xFFF7BC30 GIOEMUB Page 436	Reserved															
	Reserved										GIOEMUB[5:0]					
0xFFF7BC34, 0xFFF7BC54 GIODIR[7:0] Page 437	Reserved															
	Reserved								GIODIR0[A-B][7:0]							
0xFFF7BC38, 0xFFF7BC58 GIODIN[7:0] Page 438	Reserved															
	Reserved								GIODIN0[A-B][7:0]							
0xFFF7BC3C, 0xFFF7BC5C GIODOUT[7:0] Page 439	Reserved															
	Reserved								GIODOUT0[A-B][7:0]							

[1] See the specific device data sheet to verify the base address of the GIO registers.

Table 12-5. GIO Control Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFF7BC40, 0xFFF7BC60 GIOSET[7:0] Page 440	Reserved															
	Reserved								GIOSET[A-B][7:0]							
0xFFF7BC44, 0xFFF7BC64 GIOCLR[7:0] Page 441	Reserved															
	Reserved								GIOCLR[A-B][7:0]							
0xFFF7BC48, 0xFFF7BC68 GIOPDR[7:0] Page 442	Reserved															
	Reserved								GIOPDR[A-B][7:0]							
0xFFF7BC4C, 0xFFF7BC6C GIOPULDIS[7:0] Page 443	Reserved															
	Reserved								GIOPULDIS[A-B][7:0]							
0xFFF7BC50, 0xFFF7BC70 GIOPSL[7:0] Page 444	Reserved															
	Reserved								GIOPSL[A-B][7:0]							

[1] See the specific device data sheet to verify the base address of the GIO registers.

12.5.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a zero (0) to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before normal operations can begin on this module. Figure 12-11 and Table 12-6 describe this register.

Figure 12-11. GIO Global Control Register (GIOGCR0) [0xFFF7BC00]

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
Reserved															RESET
R-0															R/WP-0

R = Read in all modes; WP = Write in privileged mode only; -n = Value after reset

.

Table 12-6. GIO Global Control Register (GIOGCR0) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zeros and writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

12.5.2 GIO Interrupt Detect Register (GIOINTDET)

The GIOINTDET register provides the flexibility to either ignore the polarity of the edges that are recognized as an interrupt, in which case both rising and falling edges are recognized, or recognizing the interrupt on specifically a rising or falling edge as determined by the GIOPOL register. To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. Figure 12-12 and Table 12-7 describe this register.

Figure 12-12. GIO Interrupt Detect Register (GIOINTDET) [0xFFF7BC08]

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
Reserved								GIOINTDET 0[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-7. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved	1	Reserved.
7–0	GIOINTDET 0[7:0]		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.

12.5.3 GIO Interrupt Polarity Register (GIOPOL)

The GIOPOL register controls the polarity—rising edge (low to high) or falling edge (high to low)—that sets the flag. To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. When the device is in low power mode, the interrupts are no longer triggered by an edge, but instead by a level. Therefore, in low power mode, the GIOPOL register controls the ***level***, high or low, which will trigger the interrupt. Figure 12-13 and Table 12-8 describe this register.

Figure 12-13. GIO Interrupt Polarity Register (GIOPOL) [0xFFF7BC0C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">GIOPOL 0[7:0]</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R/W-0</td></tr>
</table>

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-8. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		Reserved.
7–0	GIOPOL 0[7:0]		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 (doze, snooze, sleep or hibernate):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.

12.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR register controls which interrupt-capable pins are configured as interrupts. These two registers are physically implemented as a single register. If the interrupt is enabled, the signal with an appropriate edge leads to an interrupt.

12.5.4.1 GIOENASET Register

Figure 12-14 and Table 12-9 describe this register.

> **Note: Enabling Interrupt at the Device Level**
>
> GIO can be mapped to two different device level interrupts, the GIO high level (level A) and low level (level B) interrupts through programming registers GIOLVLSET and GIOLVLCLR. The corresponding bit to the mapped device level interrupt must be set within the vectored interrupt manager (M3VIM) in the interrupt mask register (REQMASK) to enable the appropriate interrupts. Additionally, the ARM CPU (CPSR bit 7 or 6) must be cleared to recognize interrupt requests (IRQ/FIQ); .

Figure 12-14. GIO Interrupt Enable Set Register (GIOENASET) [0xFFF7BC10]

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
Reserved								GIOENASET 0[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-9. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		*Reserved.*
7–0	GIOENASET 0[7:0]		Interrupt enable for pins GIOA[7:0]
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

12.5.4.2 GIOENACLR Register

This register disables the interrupt. Figure 12-15 and Table 12-10 describe this register.

Figure 12-15. GIO Interrupt Enable Clear Register (GIOENACLR) [0xFFF7BC14]

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
Reserved								GIOENACLR 0[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-10. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		*Reserved.*
7–0	GIOENACLR 0[7:0]		Interrupt disable for pins GIOA[7:0].
		0	*Read:* The interrupt is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* Writing a one to this bit disables the interrupt.

12.5.5 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 M3VIM. Each interrupt is individually configured.

- The high-level interrupts are recorded to GIOOFFA and GIOEMUA.
- The low-level interrupts are recorded to GIOOFFB and GIOEMUB.

12.5.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the M3VIM. An interrupt can be configured as a high level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a zero has no effect. Figure 12-16 and Table 12-11 describe this register.

Figure 12-16. GIO Interrupt Priority Register (GIOLVLSET) [0xFFF7BC18]

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
Reserved								GIOLVLSET 0[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-11. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		*Reserved.*
7-0	GIOLVLSET 0[7:0]		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 GIOOFFB and GIOEMUB. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA.

12.5.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low level interrupt going to the M3VIM. Figure 12-17 and Table 12-12 describe this register.

Figure 12-17. GIO Interrupt Priority Register (GIOLVLCLR) [0xFFF7BC1C]

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
Reserved								GIOLVLCLR 0[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-12. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved	1	*Reserved.*
7–0	GIOLVLCLR 0[7:0]		GIO low priority interrupt for pins GIOA[7:0].
		0	*Read:* The interrupt is a low-level interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The interrupt is set as a high level interrupt. The high-level interrupts are recorded to GIOOFFA and GIOEMUA. *Write:* The interrupt is set as a low level interrupt. The low-level interrupts are recorded to GIOOFFB and GIOEMUB.

12.5.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET and GIOPOL) has occurred. The flag is also cleared by reading the appropriate offset register; see Section 12.3.3.3. Figure 12-18 and Table 12-13 describe this register.

Figure 12-18. GIO Interrupt Flag Register (GIOFLG) [0xFFF7BC20]

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
Reserved								GIOFLG 0[7:0]							
R-0								R/WC-0							

R = Read in all modes; WC = Write clears the bit, -n = Value after reset

.

Table 12-13. GIO Interrupt Flag Register (GIOFLG) Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		Reserved.
7–0	GIOFLG 0[7:0]		GIO flag for pins GIOA[7:0].
		0	*Read:* A transition has not occurred since the last clear. *Write:* Writing a zero 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**

12.5.7 GIO Offset A Register (GIOOFFA)

The GIOOFFA 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 12-19 and Table 12-14 describe this register.

Note:
Reading this register clears it, GIOEMUA 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.

Figure 12-19. GIO Offset A Register (GIOOFFA) [0xFFF7BC24]

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
Reserved										GIOOFFA[5–0)					
R-0										R-0					

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-14. GIO Offset A Register (GIOOFFA) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOOFFA(5–0)		GIO offset A. 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.
		000000	No interrupt is pending.
		000001	Interrupt 0 (correspond to GIOA0) is pending with a high priority.
		. . .	. . .
		001000	Interrupt 7 (correspond to GIOA7) is pending with a high priority.
		001001–111111	Reserved

12.5.8 GIO Offset B Register (GIOOFFB)

The GIOOFFB 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 12-20 and Table 12-15 describe this register.

Note:

Reading this register clears it, GIOEMUB 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.

Figure 12-20. GIO Offset B Register (GIOOFFB) [0xFFF7BC28]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="10">Reserved</td><td colspan="6">GIOOFFB(5–0)</td></tr>
<tr><td colspan="10">R-0</td><td colspan="6">R-0</td></tr>
</table>

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 12-15. GIO Offset B Register (GIOOFFB) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOOFFB		GIO offset register B. 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.
		000000	No interrupt is pending.
		000001	Interrupt 0 (correspond to GIOA0) is pending with a low priority.
		. . .	. . .
		001000	Interrupt 7 (correspond to GIOA7) is pending is pending with a low priority.
		001001–111111	Reserved

12.5.9 GIO Emulation A Register (GIOEMUA)

The GIOEMUA register is a read-only register. The contents of this register are identical to the contents of GIOOFFA. The intention for the this register is that software can use it without clearing the flags. Figure 12-21 and Table 12-16 describe this register.

Note:
The corresponding flag in the GIOFLG register (Section 12.5.6) is not cleared when the GIOEMUA register is read.

Figure 12-21. GIO Emulation A Register (GIOEMUA) [0xFFF7BC2C]

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
Reserved										GIOEMUA					
R-0										R-0					

R = Read in all modes; n = Value after reset

Table 12-16. GIO Emulation A Register (GIOEMUA) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOEMUA		GIO emulation register A. These bits index the currently pending high-priority interrupt.
		000000	No interrupt is pending.
		000001	Interrupt 0 (correspond to GIOA0) is pending with a high priority.
		. . .	. . .
		001000	Interrupt 7 (correspond to GIOA7) is pending with a high priority.
		001001–111111	Reserved

12.5.10 GIO Emulation B Register (GIOEMUB)

The GIOEMUB register is a read-only register. The contents of this register are identical to the contents of GIOOFFB. The intention for the this register is that software can use it without clearing the flags. Figure 12-22 and Table 12-17 describe this register.

Note:
The corresponding flag in the GIOFLG register (Section 12.5.6) is not cleared when the GIOEMUB register is read.

Figure 12-22. GIO Emulation B Register (GIOEMUB) [0xFFF7BC30]

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
Reserved										GIOEMUB(5–0)					
R-0										R-0					

R = Read in all modes; -n = Value after reset

.

Table 12-17. GIO Emulation B Register (GIOEMUB) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and writes have no effect.
5–0	GIOEMUB		GIO emulation register B. These bits index the currently pending low-priority interrupt.
		000000	No interrupt is pending.
		000001	Interrupt 0 (correspond to GIOA0) is pending with a low priority.
		. . .	. . .
		001000	Interrupt 7 (correspond to GIOA7) is pending with a low priority.
		001001– 111111	Reserved

12.5.11 GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 12-23 and Table 12-18 describe this register.

Figure 12-23. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0])[0xFFF7BC34, 0xFFF7BC54]

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
Reserved								GIODIR[A-B][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-18. GIO Data Direction Registers [A-B][7:0] (GIODIR[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIODIR[A-B][7:0]		GIO data direction of ports [A-B], 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. **Note: The input buffer is always enabled except for the situation stated in Section 12.3.2.3.**

12.5.12 GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0])

Values in the GIODIN register reflect the current state (high = 1 or low = 0) on the pins of the port. Figure 12-24 and Table 12-19 describe this register.

Figure 12-24. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0])[0xFFF7BC38, 0xFFF7BC58]

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
Reserved								GIODIN[A-B][7:0]							
R-0								R-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 12-19. GIO Data Input Registers [A-B][7:0] (GIODIN[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODIN[A-B][7:0]		GIO data input for ports [A-B], pins[7:0].
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

12.5.13 GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0])[0xFFF7BC3C, 0xFFF7BC5C]

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 12-25 and Table 12-20 describe this register.

Note:

Values in the GIODSET register, Section 12.5.14, set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

Figure 12-25. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0])

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
Reserved								GIODOUT[A-B][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-20. GIO Data Output Registers [A-B][7:0] (GIODOUT[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODOUT[A-B][7:0]		GIO data output of ports [A-B], 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.**

12.5.14 GIO Data Set Register [A-B][7:0] (GIODSET[A-B][7:0])

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 12-26 and Table 12-21 describe this register.

Figure 12-26. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0])[0xFFF7BC40, 0xFFF7BC60]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">GIODSET[A-B][7:0]</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">R/W-0</td></tr>
</table>

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

.

Table 12-21. GIO Data Set Registers [A-B][7:0] (GIODSET[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODSET[A-B][7:0]		GIO data set for ports [A-B], pins[7:0]. This bit drives the output of GIO pin high.
		0	*Write:* Writing a zero 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.**

12.5.15 GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0])

Values in this register clear the data output register (Section 12.5.14) bit to 0 regardless of its current value. The contents of this register reflect the contents of GIODOUT. Figure 12-27 and Table 12-22 describe this register.

Figure 12-27. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0])[0xFFF7BC44, 0xFFF7BC64]

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
Reserved								GIODCLR[A-B][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 12-22. GIO Data Clear Registers [A-B][7:0] (GIODCLR[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIODCLR[A-B][7:0]		GIO data clear for ports [A-B], pins[7:0]. This bit drives the output of GIO pin low.
		0	*Write:* Writing a zero 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.**

12.5.16 GIO Open Drain Register [A-B][7:0] (GIOPDR[A-B][7:0])

Values in this register enable or disable the open drain capability of the data pins. Figure 12-28 and Table 12-23 describe this register.

Figure 12-28. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0])[0xFFF7BC48, 0xFFF7BC68]

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
Reserved								GIOPDR[A-B][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privilege modes; -n = Value after reset

.

Table 12-23. GIO Open Drain Registers [A-B][7:0] (GIOPDR[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7-0	GIOPDR[A-B][7:0]		GIO open drain for ports [A-B], 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.

12.5.17 GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0])

Values in this register enable or disable the pull control capability of the pins. Figure 12-29 and Table 12-24 describe this register.

Figure 12-29. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0])[0xFFF7BC4C, 0xFFF7BC6C]

31–16
Reserved
R-0

15–8	7–0
Reserved	GIOPULDIS[A-B][7:0]
R-0	R/W-0

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset depends upon the device

.

Table 12-24. GIO Pull Disable Registers [A-B][7:0] (GIOPULDIS[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIOPULDIS[A-B][7:0]		GIO pull disable for ports [A-B], pins[7:0]. Writes to this bit will only take effect when the GIO pin is 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.

12.5.18 GIO Pull Select Register [A-B][7:0] (GIOPSL[A-B][7:0])

Values in this register select the pull up or pull down functionality of the pins. Figure 12-30 and Table 12-25 describe this register.

Figure 12-30. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0])[0xFFF7BC50, 0xFFF7BC70]

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
Reserved								GIOPSL[A-B][7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in user and privileged modes; -n = Value after reset

Table 12-25. GIO Pull Select Registers [A-B][7:0] (GIOPSL[A-B][7:0]) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	GIOPSL[A-B][7:0]		GIO pull select for ports [A-B], 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.

12.6 Applications Examples

The application examples in this section assume a typical configuration of five I/O functional ports, in which only Port A is external interrupt-capable.

12.6.1 Example: Setting Interrupts and Configuring Pins for Output

The following code demonstrates how to set the interrupts. In the example, two of the interrupts are enabled, and the third pin of port A is configured for output. R2 keeps the same value, and the other registers act as temporary storage for addresses and values.

```
GIO_LOC    .word 0xFFF7BC00       ;device specific address for the GIO registers.
GIOENASET  .equ     0x0C            ;setting up equate statements
GIOPOL     .equ    0x08
GIOFLG     .equ    0x1C
GIOPRYSET  .equ    0x14
GIOPRYCLR  .equ    0x18
GIOOFFA    .equ    0x20
GIOEMUA    .equ    0x28
GIOOFFB    .equ    0x24
GIOEMUB    .equ    0x2C
GIODIRA    .equ    0x30
GIODINA    .equ    0x34
GIODOUTA   .equ    0x38
GIODSETA   .equ    0x3C
GIODCLRA   .equ    0x40
GIOPDRA    .equ    0x44
GIOPULDISA .equ    0x48

   LDR R2, GIO_LOC              ;loads GIO base address into register 2.
                                ; **SET THE POLARITY OF THE INTERRUPTS.**
   MOV R3, #GIOPOL              ; loads GIOPOL offset address into register 3.
   MOV R4, #0x02                ; loads a value of 0x02 into register 4.
                                ; Sets bit 1.
   STR R4, [R2, R3]             ; sets interrupt 0 to trigger on falling edge
                                ; and interrupt 1 to trigger on rising edge.
                                ; **SET THE PRIORITY ON BOTH INTERRUPTS AS LOW**
   MOV R3, #GIOPRYCLR           ; loads the GIOPRY address into register 3.
   MOV R4, #0x03                ; loads a value of 0 into register 4.
   STR R4, [R2, R3]             ; priority on interrupts 0 and 1 is set LOW.
                                ; ** SET PORT A FOR OUTPUT (EXCEPT FOR INTERRUPTS
                                ; WHICH MUST BE CONFIGURED AS INPUTS)**
   MOV R3, #GIODIRA             ; loads GIODIRA offset address into register 3.
   MOV R4, #0xFC                ; loads a binary value of 11111100
```

```
    STR R4, [R2, R3]                ; sets pins 0 and 1 as input to insure proper
                                    ; interrupt behavior pin 2 configured as
                                    ; output

    MOV R3, #GIOFLG                 ; loads GIOFLG offset address into register 3.

    MOV R4, #0xFF                   ; sets 32 bits.

    STR R4, [R2, R3]                ; clears all bits of the flag register.

 ; **ENABLE THE FIRST TWO INTERRUPTS 1:0.**

  MOV R3, #GIOENA ; loads GIOENA offset address into register 3.

  MOV R4, #0x03 ; sets first 2 bits 1:0

  STR R4, [R2, R3] ; enables the first two interrupts
```

12.6.2 Example: Toggling Output Buffers

The following code demonstrates how to toggle the output buffer port B using the GIODSET and GIODCLR buffers.

```
GIO_LOC    .word   0xFFF7BC00   ;device specific address for the GIO registers.

GIOENA     .equ    0x04       ;setting up equate statements

...

GIODIRB   .equ    0x58

GIODINB    .equ    0x5C

GIODOUTB   .equ    0x60

GIODSETB   .equ    0x64

GIODCLRB   .equ    0x68

    LDR R2, GIO_LOC                 ; loads absolute address of the GIO memory
                                    ; location (device specific).

                                    ; **SET ALL BITS IN GIODOUTB AS 1.**

    MOV R3, #GIODSETB               ; loads offset address of GIODSETB

    MOV R4, #0xFF                   ; loads a binary value of 11111111

    STR R4, [R2, R3]                ; sets all bits of GIODOUTB.

                                    ; **CONFIGURE PORT 1 AS OUTPUT.**

    MOV R3, #GIODIRB                ; loads offset address of GIODIRB.

    MOV R4, #0xFF                   ; loads a binary value of 11111111.

    STR R4, [R2, R3]                ; configures port 1 as output

                                    ; **TOGGLE BITS 0, 2, 4, 6.**

    MOV R3, #GIODSETB               ; loads offset address of GIODSETB

    MOV R4, #GIODCLRB               ; loads GIODCLRB offset address

    MOV R5, #0x55                   ; loads a binary value of 01010101

TOGGLE

    STR R5, [R2, R4]                ; clears GIODOUTB bits 0, 2, 4, 6

    STR R5, [R2, R3]                ; sets GIODOUTB bits 0, 2, 4, 6
      B TOGGLE        ; loops back to TOGGLE
```

12.6.3 Example: Clearing Interrupt Flags and Setting Interrupts

The following code demonstrates how interrupt flags should be cleared before interrupts are enabled. In this example, pins GIO(2:0) are set as interrupts.

```
GIO_LOC     .word   0xFFF7BC00   ;device specific address for GIO registers
GIOENASET   .equ    0x0C       ;setting up equate statements
...
GIOFLG      .equ    0x1C


   LDR R2, GIO_LOC             ;loads absolute address of the GIO memory
   MOV R3, #GIOFLG             ;loads GIOFLG offset address into R3.
   MOV R4, #0x07               ;loads a value of 00000111 into R4.
   STR R4, [R2, R3]            ;clears the interrupt-capable bits of the
                               ;GIOFLG control register.

   MOV R3, #GIOENA             ;loads GIOENA offset address into R3.
   MOV R4, #0x07               ;loads 00000111 into R4.
   STR R4, [R2, R3]            ;enables pins 2:0 of port A as interrupts.
```

12.6.4 Example: Reading Port B Input Register

The following code demonstrates how to read the input register of port A when interrupts are enabled. Pin 0 is set as an interrupt, and pins 2 and 1 are configured as inputs.

```
GIO_LOC.word 0xFFF7BC00        ;device specific address for GIO registers
GIOENA .equ    0x04            ;setting up equate statements
...
GIODIRB .equ 0x58
GIODINB .equ 0x5C
GIODOUTB.equ 0x60
GIODSETB.equ 0x64
GIODCLRB .equ 0x68
   LDR R2, GIO_LOC             ;loads absolute address of the GIO memory
                               ;**MASK OUTPUTS AND INTERRUPTS SO INPUT IS UNAMBIGUOUS.**
   MOV R3, #GIODINB            ;loads the offset address of GIODINB.
   LDR R4, [R2, R3]            ;loads the value in GIODINB register into R4.
   MOV R3, #0xFE               ;loads 11111110 into R3. This value is used
                               ;to mask the input so that only the input
                               ;values are read. The 1’s appear in the places
                               ;where the input register is reading input
                               ;voltages.
   AND R4, R4, R3              ;loads masked input value into R4.
```

Chapter 13
SPNU489C-February 2012

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 2.0 protocol specified in the *LIN Specification Package*. This module can be configured to operate in either SCI(UART) or LIN mode

Topic **Page**

13.1 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 functionality 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.

13.1.1 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.
- At 100MHz Peripheral Clock, 3.125 Mbits/s is the Max Baud Rate achievable.
- Capability to use Direct Memory Access (DMA) for transmit and receive data.
- 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.

13.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3 or 2.0 protocols.
- LIN2.1 protocol Master Complaint.
- 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
- Capability to use Direct Memory Access (DMA) for transmit and receive data.
- 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

13.2 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major component 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 (interface clock generated by the system module).

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 13-1 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 multibuffered receiver and transmitter. The SCI interface, the DMA control subblocks and the baud generator are modified as part of the hardware enhancements for LIN compatibility. Figure 13-2 shows the SCI/LIN block diagram.

Figure 13-1. Detailed SCI Block Diagram

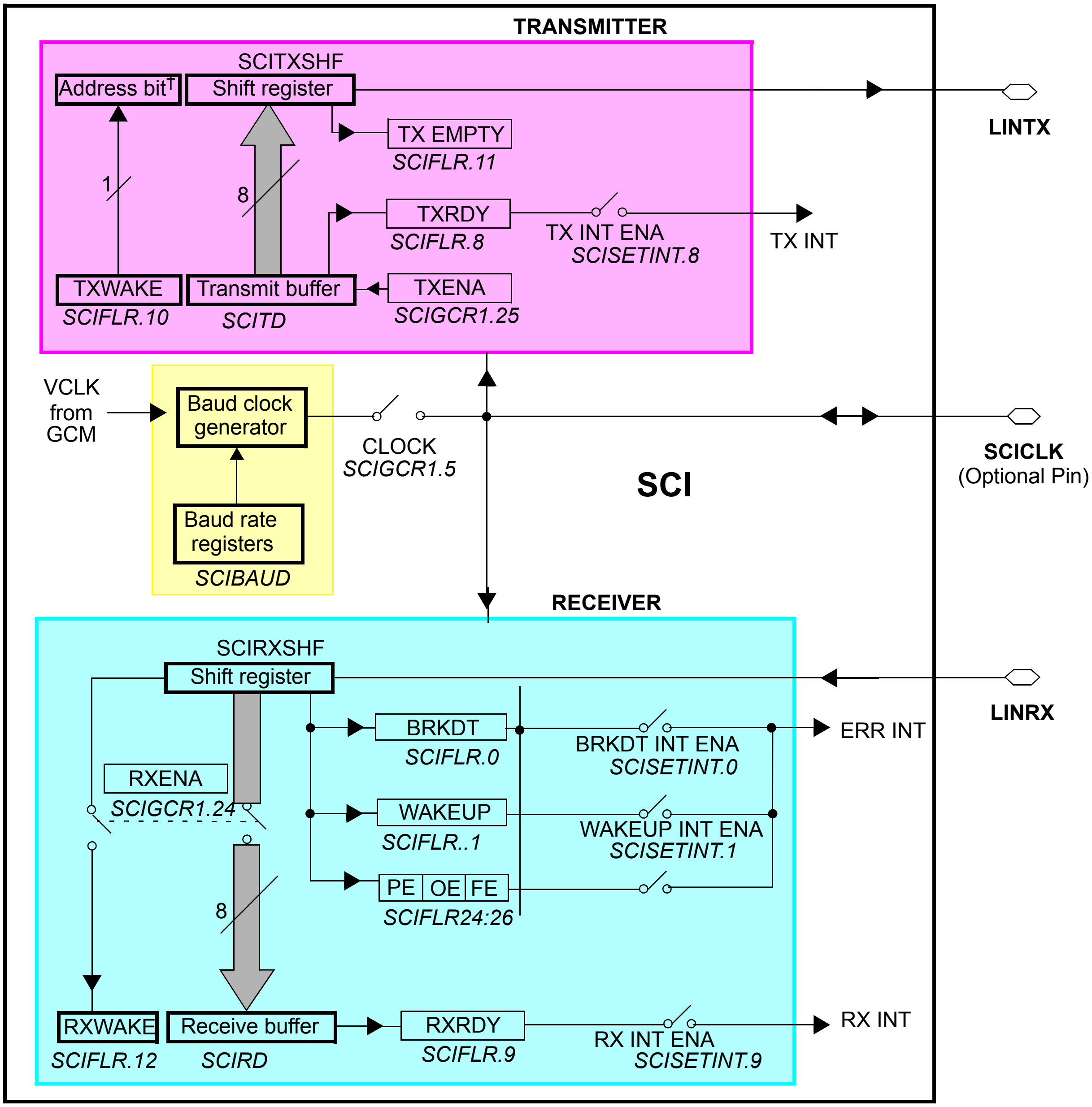
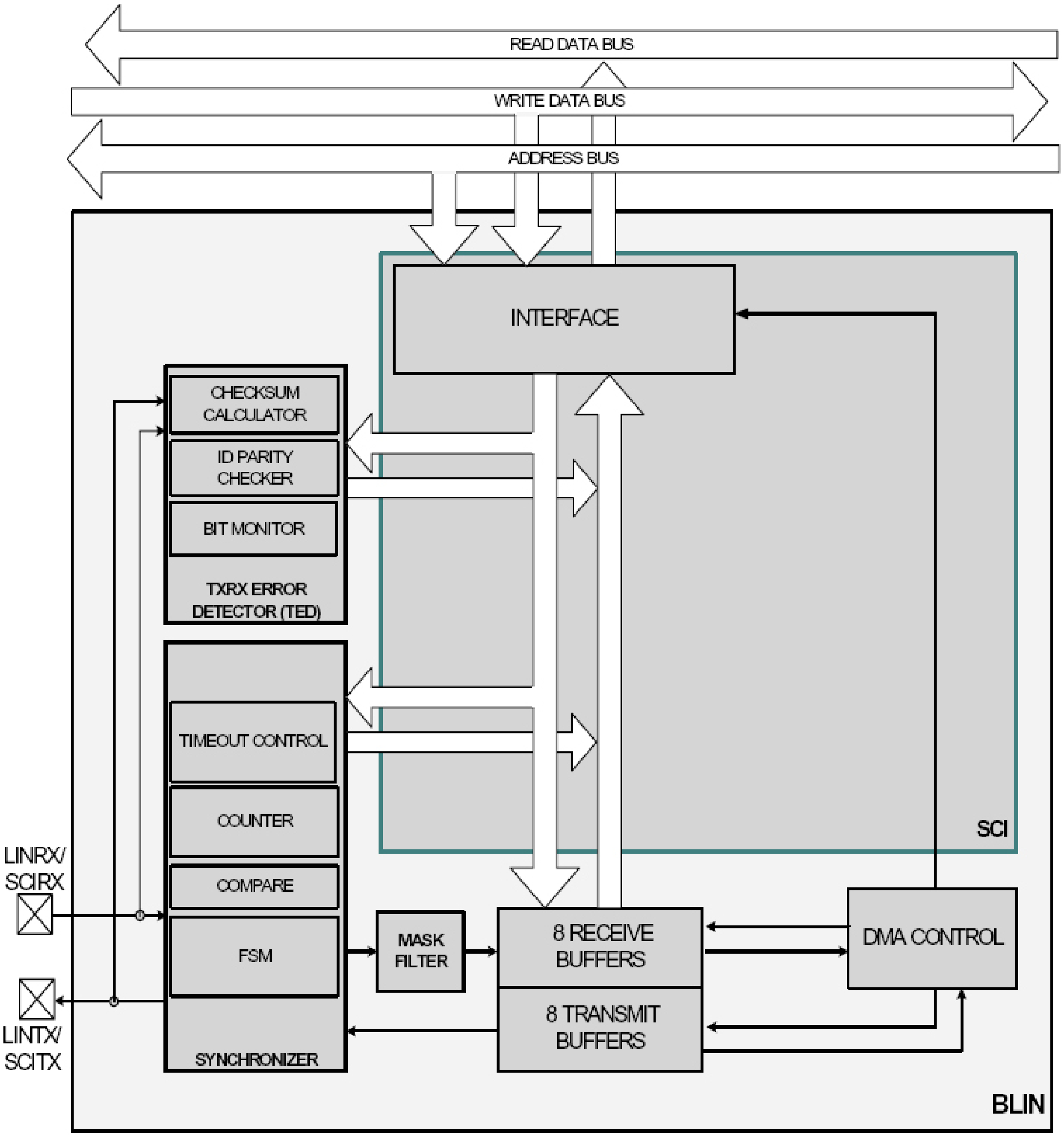


Figure 13-2. SCI/LIN Block Diagram



13.3 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

13.3.1 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

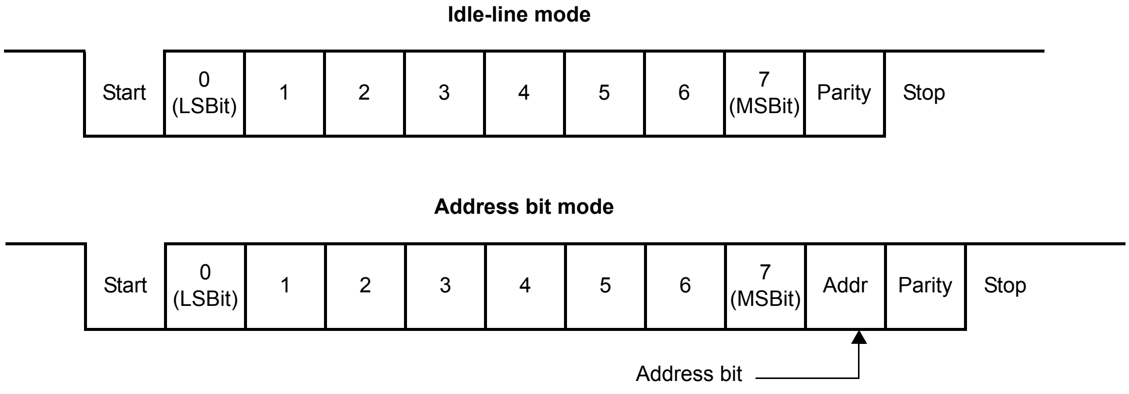
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start 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 13-3.

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 13-3 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 13-3 use one stop bit per frame.

Figure 13-3. Typical SCI Data Frame Formats



Idle-line mode

Address bit mode

13.3.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

13.3.2.1 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.

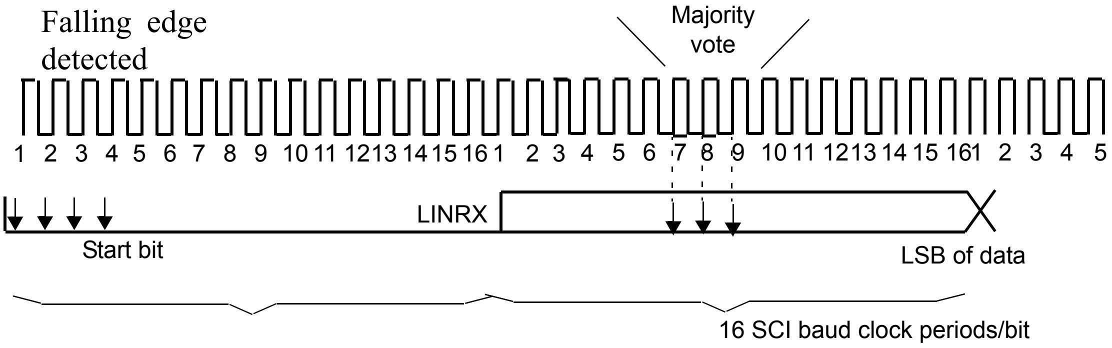
In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors 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 Stat 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 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 13-4 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 13-4. Asynchronous Communication Bit Timing



13.3.2.2 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.**

13.3.3 SCI Baud Rate

The SCI/LIN has an internally generated serial clock determined by the VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value of BRSR register to select the required baud rates. The additional 4-bit fractional divider M refines the baudrate selection.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$\textit{SCICLK Frequency} = \left(\frac{\text{VCLK Frequency}}{\left(P+1+\frac{M}{16}\right)}\right)$$

$$\textit{Asynchronous baud value} = \left(\frac{\text{SCICLK Frequency}}{16}\right)$$

For P = 0,

$$\textit{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{32}\right)$$

13.3.3.1 *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 (BRSR[27:24]), the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 13-2. 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 13-1), as shown in Table 13-2.

The baud rate will vary over a data field to average according to the BRSR[30:28] value by a "**d**" fraction of the peripheral internal clock: **0<d<1.** Refer Figure 13-5 for a simple Average "d' calculation based on "U" value(BRSR[30:28]).

The instantaneous bit time is expressed in terms of T_{VCLK} *as follows:*

$$\text{For all P other than 0, and all M and d (0 or 1), } T^{i}{}_{bit} = \left[16\left(P+1+\frac{M}{16}\right)+d\right]T_{VCLK}$$

$$\text{For P= } 0 \; T_{bit} = 32T_{VCLK}$$

The averaged bit time is expressed in terms of T_{VCLK} as follows*:*

$$\text{For all P other than } 0\text{, and all M and d (0<d <1), } T^{a}{}_{bit} = \left[16\left(P+1+\frac{M}{16}\right)+d\right]T_{VCLK}$$

$$\text{For P= } 0 \; T_{bit} = 32T_{VCLK}$$

.

Table 13-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration)

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
2	1	0	0	0	1	0	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0
4	1	0	1	0	1	0	1	0	1	0
5	1	1	1	0	1	0	1	0	1	1
6	1	1	1	0	1	1	1	0	1	1
7	1	1	1	1	1	1	1	0	1	1

1 Normal configuration = Start + 8 Data Bits + Stop Bit

Table 13-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration)

BRS [30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop 0	Stop 1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0	0	0
2	1	0	0	0	1	0	0	0	1	0	0	0	1
3	1	0	1	0	1	0	0	0	1	0	1	0	1
4	1	0	1	0	1	0	1	0	1	0	1	0	1
5	1	1	1	0	1	0	1	0	1	1	1	0	1
6	1	1	1	0	1	1	1	0	1	1	1	0	1
7	1	1	1	1	1	1	1	0	1	1	1	1	1

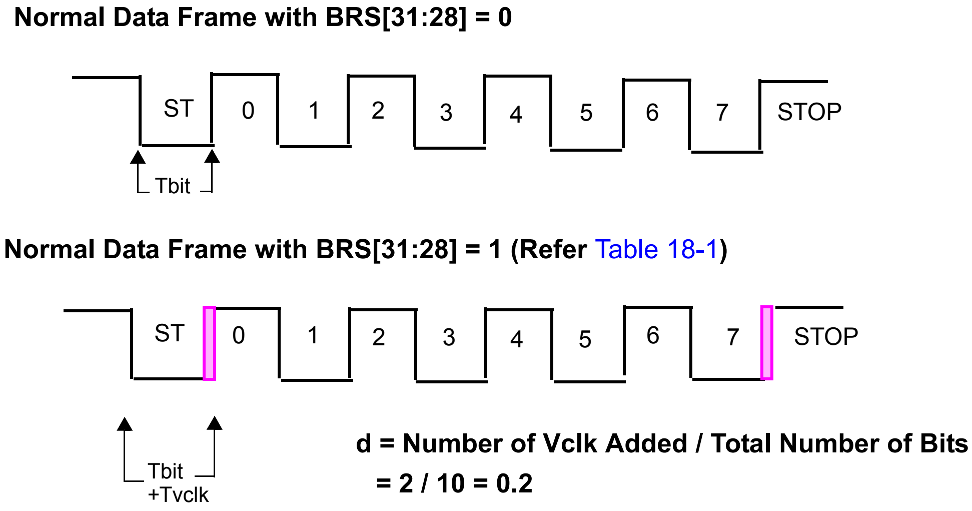
1 Maximum configuration = Start + 8 Data Bits + Addr Bit + Parity Bit + Stop Bit 0 + Stop Bit 1

Table 13-3. SCI Mode (Minimum Configuration)

BRS[30:28]	Start Bit	D[0]	Stop
0	0	0	0
1	1	0	0
2	1	0	0
3	1	0	1
4	1	0	1
5	1	1	1
6	1	1	1
7	1	1	1

1 Minimum configuration = Start + 1 Data Bits + Stop Bit

Figure 13-5. Superfractional Divider Example


Normal Data Frame with BRS[31:28] = 0
ST
0
1
2
3
4
5
6
7
STOP
Tbit
Normal Data Frame with BRS[31:28] = 1 (Refer Table 18-1)
ST
0
1
2
3
4
5
6
7
STOP
Tbit
+Tvclk
d = Number of Vclk Added / Total Number of Bits
= 2 / 10 = 0.2

13.3.4 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 Support two multi processor 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.

13.3.4.1 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 13-6 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:

Step1 : Write a 1 to the TXWAKE bit.

Step2 : 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.

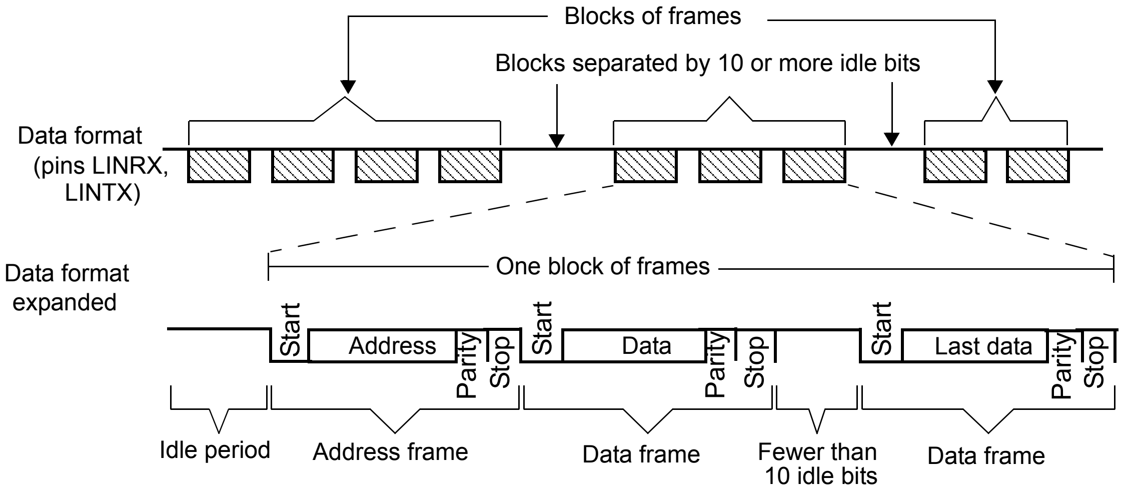
Step3 : Wait for the SCI to clear the TXWAKE flag.

Step4 : 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 13-6. Idle-Line Multiprocessor Communication Format



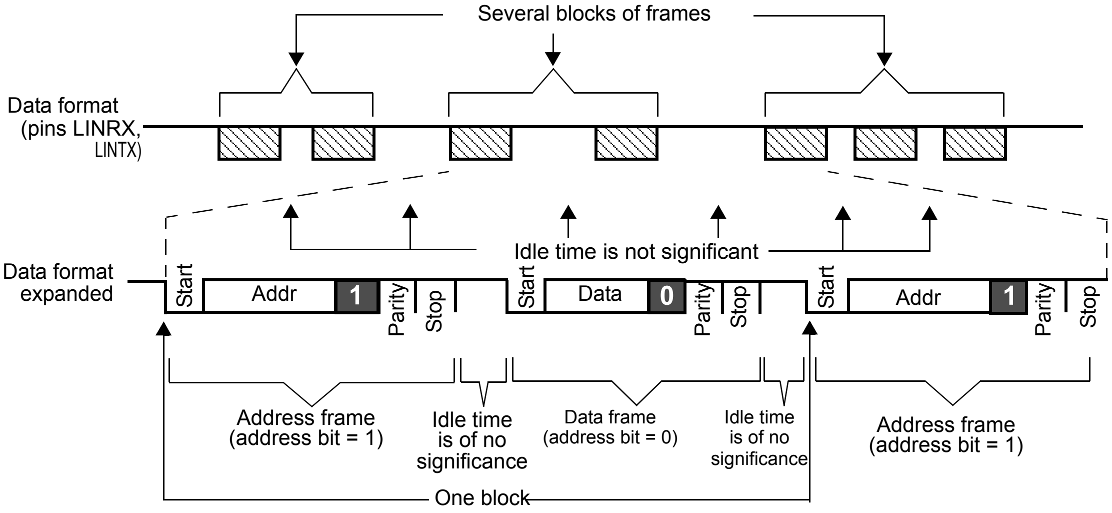
13.3.4.2 Address-Bit Multiprocessor Mode

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 13-7 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 13-7. Address-Bit Multiprocessor Communication Format



13.3.5 SCI Multi Buffered Mode

To reduce CPU load when Receiving or Transmitting data in interrupt mode or DMA mode, the SCI/LIN module has eight separate Receive and transmit buffers. Multi buffered mode is enabled by setting the MBUF MODE bit.

The multibuffer 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, as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is , frame error, and overrun error).

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag in SCIFLR register), and a DMA request (TXDMA) could occur after transmitting a response.

Figure 13-8 and Figure 13-9 shows the receive and transmit multibuffer functional block diagram.

Figure 13-8. Receive Buffers

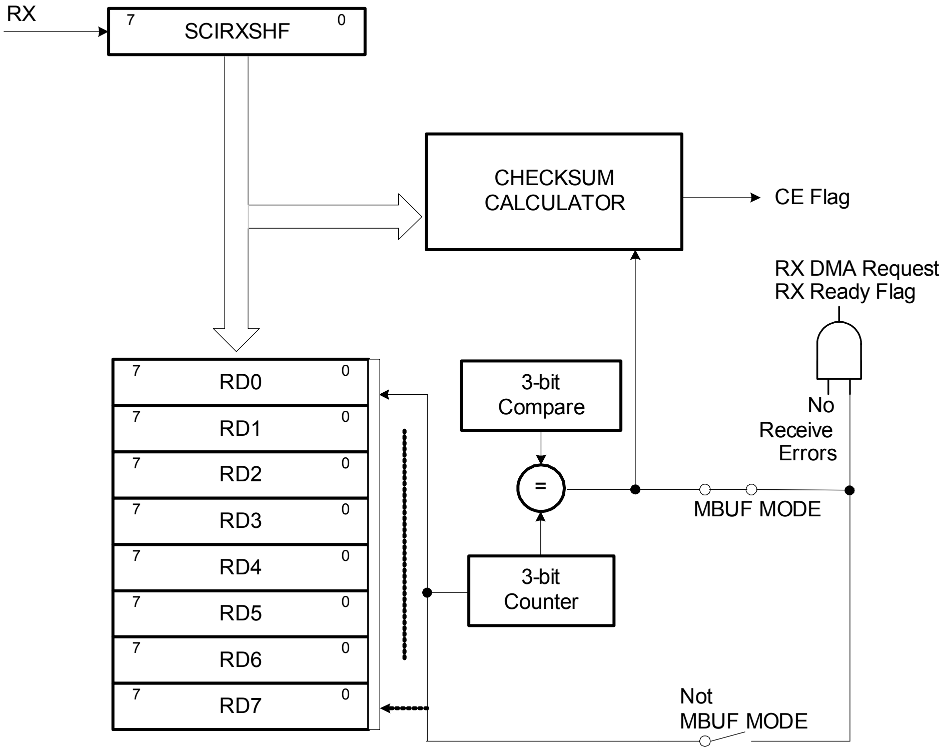
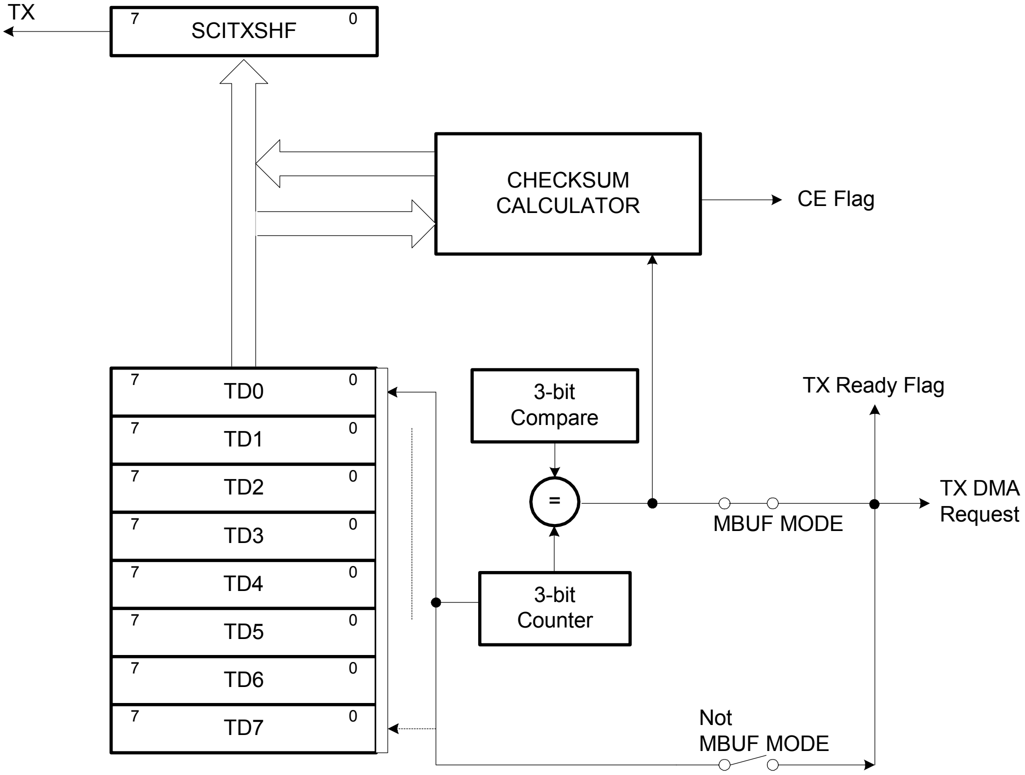


Figure 13-9. Transmit Buffers



13.4 SCI Interrupts

The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see Figure 13-10). 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 13-10. General Interrupt Scheme

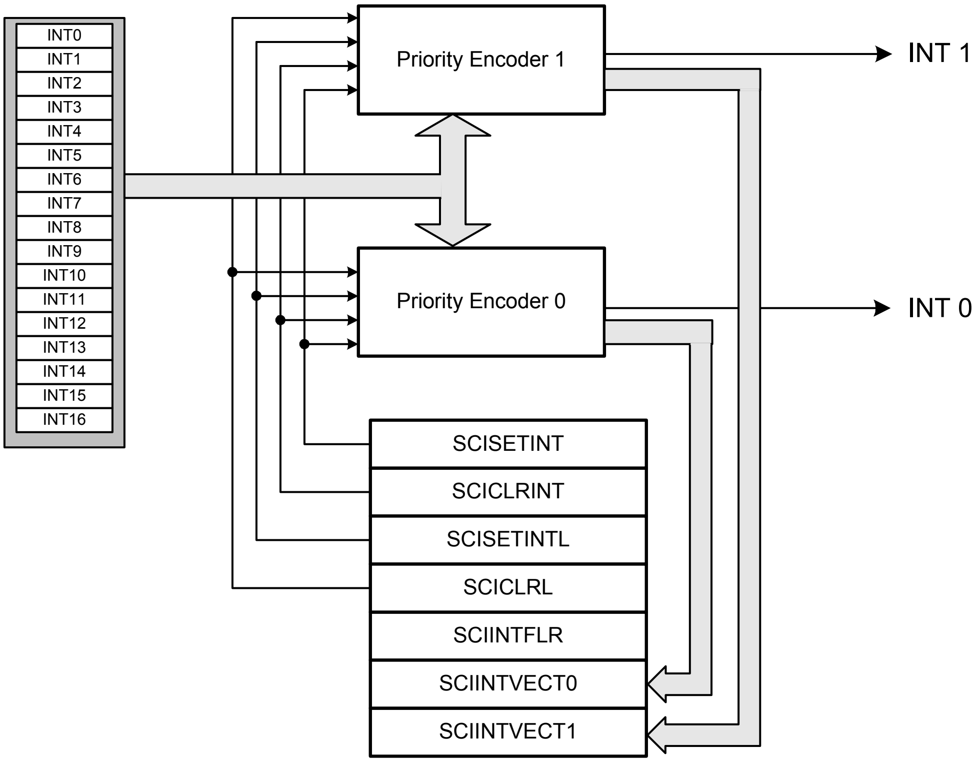
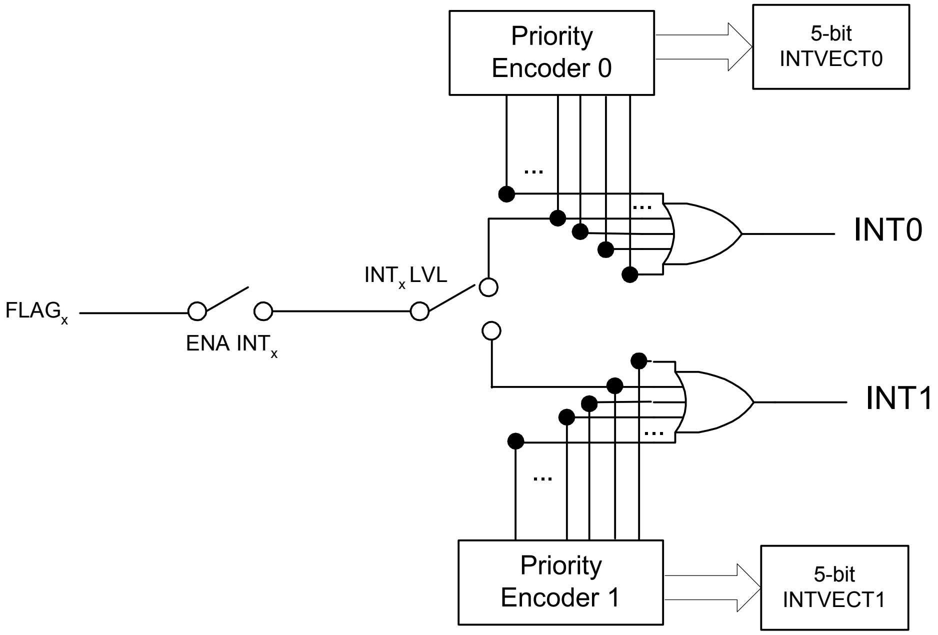


Figure 13-11. Interrupt Generation for Given Flags



13.4.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. 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 unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is 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.

13.4.2 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.

On a device with both SCI and a DMA controller, the bits SET RX DMA ALL and SET RX DMA must be cleared to select interrupt functionality.

13.4.3 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.

13.4.4 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)

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 13-4.

Table 13-4. 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.

13.5 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. The DMA transfers depending on whether multibuffer mode bit (MBUF MODE) is enabled or not. For DMA module configuration refer

13.5.0.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

In Multi Buffered SCI mode with DMA enabled, the receiver loads the RDy buffers for each received character. RX DMA request is triggered once the last character of the programmed number of characters (LENGTH) are received and copied to the corresponding RDy buffer successfully.

If the multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received.

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames or DMA requests for both. This is controlled by the SET RX DMA ALL bit.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received.

13.5.0.2 Transmit DMA Requests

DMA functionality is enabled/disabled by the CPU with SET TX DMA/CLR TX DMA bits, respectively.

In Multi Buffered SCI mode once TXRDY bit is set or after a transmission of programmed no. of characters (LENGTH) (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred.

13.6 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 is set to 1. Of particular importance is the SWnRST bit. This active-low bit is 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 (one) 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 SWnRST is held low the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting RESET bit.
- Clear SWnRST to 0 before configuring the SCI.
- Select the desired frame format by programming SCIGCR1.
- Configure the LINRX and LINTX pins for SCI functionality by setting the RX FUNC and TX FUNC bit.
- Select the baud rate to be used for communication by programming BRSR.
- Select internal clock by programming the CLOCK bit.
- Set the CONT bit to make SCI not to 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 to connect the transmitter to the receiver internally. (This feature is used to perform a self-test.)
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Set SWnRST to 1 after the SCI is configured.
- Perform Receive or Transmit data. (see Section 13.6.1 / Section 13.6.2)

13.6.1 Receiving Data

The SCI receiver is enabled to receive messages if 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.

13.6.1.1 Receiving Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 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 FE, OE, or PE 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.

User can receive data by

1) Polling Receive Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for RXRDY bit and read the data from SCIRD register once RXRDY is set high. CPU is unnecessarily overloaded by selecting Polling mode. To avoid this user can use either

Interrupt or DMA method. To use interrupt method SET RX INT bit should be set and to use DMA SET RX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment RXRDY is set.

13.6.1.2 Receiving Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. In this mode SCI sets the RXRDY bit when programmed number of data are received in the receive buffer, the complete frame. The error condition detection logic is same as Single Buffer Mode, except that it monitors for the complete frame. Like Single Buffer Mode the user can use either Interrupt, DMA or polling method to read the data. The received data has to be read from the LINRD0 and LINRD1 register, based on the number of bytes. For LENGTH less than or equal to 4, Read to LINRD0 register will clear the “RXRDY” flag. For LENGTH greater than 4, Read to LINRD1 register will clear the “RXRDY” flag.

13.6.2 Transmitting Data

The SCI transmitter is enabled if 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.

13.6.2.1 Transmitting Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits it. The flags TXRDY and TX EMPTY 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.

User can transmit data by

1) Polling Transmit Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for TXRDY bit to go high before writing the data to SCITD register. CPU is unnecessarily overloaded by doing this Polling method. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET TX INT bit should be set and to use DMA SET TX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment TXRDY 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/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can be done by either disabling the transmit interrupt(CLR TX INT) / DMA request (CLR TX DMA bit) 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.

13.6.2.2 Transmitting Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. Similar to Single Buffer Mode the software can use polling, Interrupt or DMA method to write the data to be transmitted. The data to be transmitted has to be written to LINTD0 and LINTD1 register, based on the number of bytes. SCI waits for data to be written to Byte 0(TD0) of LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically.

13.7 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.

13.8 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation, DMA controls 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 13.14.

13.8.1 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

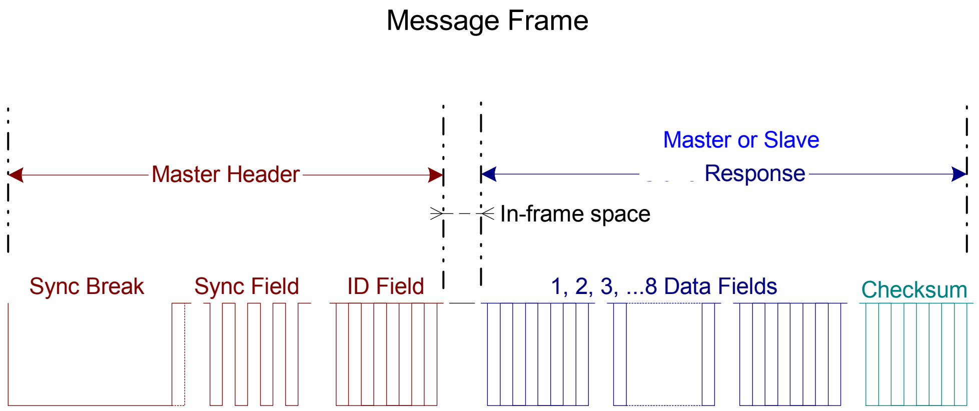
The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package* Revision 1.3 and 2.0 by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

13.8.2 Message Frame

The LIN protocol defines a message frame format, illustrated in Figure 13-12. 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.

Figure 13-12. LIN Protocol Message Frame Format: Master Header and Slave Response



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.

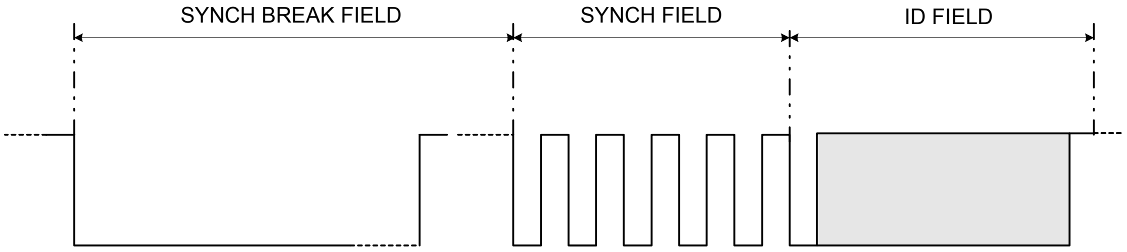
The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

13.8.2.1 Message Header

The header of a message is initiated by a master (see Figure 13-13) 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

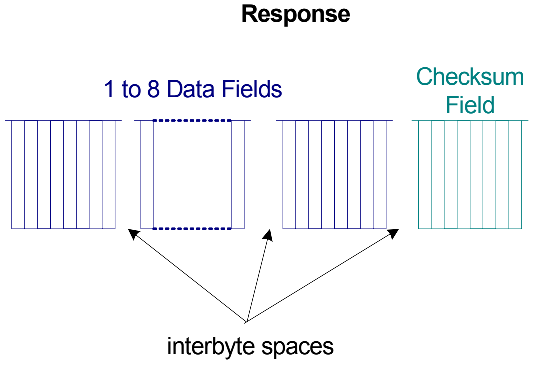
Figure 13-13. Header 3 fields: Synch Break, Synch, and ID



13.8.2.2 Response

The format of the response is as illustrated in Figure 13-14. 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 13-14. 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 13.8.6). 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 13-5, or by LENGTH value in SCIFORMAT[18:16] register; see Table 13-6. The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

Table 13-5. 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 13-6. 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

13.8.3 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 BRSR 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.

13.8.4 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 (BRSR). There is an additional 3-bit fractional divider value, field U in the BRSR, which further fine-tunes the data field baud rate.

The ranges for the prescaler values in the BRSR 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 BRSR 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** $\mathbf{T_{VCLK}}$“ (with **a** = 0,1) is added to each T_{bit} as explained in Section 13.8.4.2. 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:

$$1\text{kHz} \le F_{\text{LINCLK}} \le 20\text{kHz}$$

All transmitted bits are shifted in and out at T_{bit} periods.

13.8.4.1 Fractional Divider

The M field of the BRSR register modifies the integer prescaler P for finer 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:

$$\text{For all P other than 0, and all M, } T_{bit} = 16\left(P + 1 + \frac{M}{16}\right)T_{VCLK}$$

$$\text{For P= 0 : } T_{bit} = 32T_{VCLK}$$

Therefore, the LINCLK frequency is given by

$$F_{LINCLK} = \frac{F_{VCLK}}{16\left(P + 1 + \frac{M}{16}\right)} \quad \text{For all P other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For } P = 0$$

13.8.4.2 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)

13.8.4.3 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRSR[27:24], the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 13-7. 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 13-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
2	1	0	0	0	1	0	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0
4	1	0	1	0	1	0	1	0	1	0
5	1	1	1	0	1	0	1	0	1	1
6	1	1	1	0	1	1	1	0	1	1
7	1	1	1	1	1	1	1	0	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 BRSR[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:*

$$\text{For all P other than 0, and all M and d (0 or 1), } T^{i}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$$

$$\text{For P= 0 } T_{bit} = 32T_{VCLK}$$

The averaged bit time is expressed in terms of T_{VCLK} as follows*:*

$$\text{For all P other than 0, and all M and d (0<d <1), } T^{a}_{bit} = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK}$$

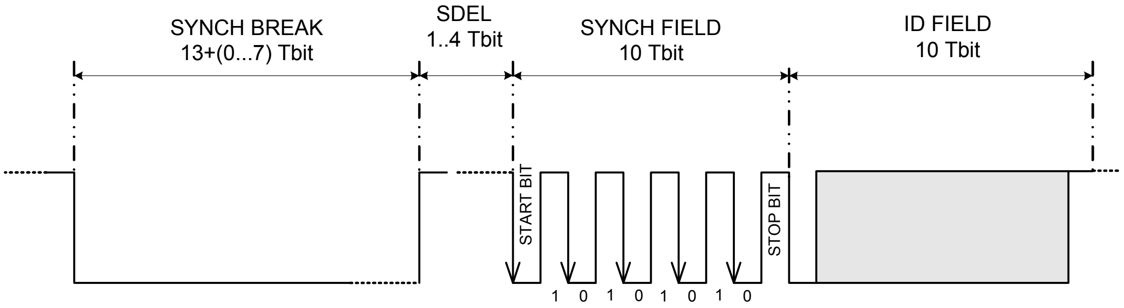
$$\text{For P= 0 } T_{bit} = 32T_{VCLK}$$

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.

13.8.5 Header Generation

Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU pr the DMA will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU or DMA 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 13-15.

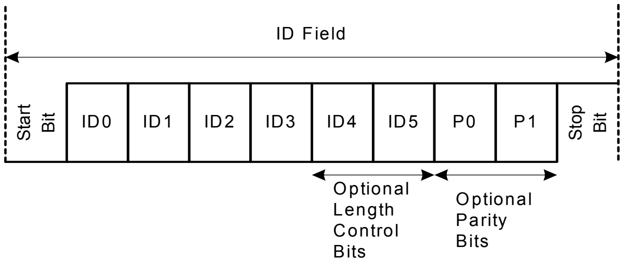
Figure 13-15. Message Header in Terms of T_{bit}



- 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 13-16 for an illustration of the ID field.

Figure 13-16. 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 multibuffer 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.

13.8.5.1 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.

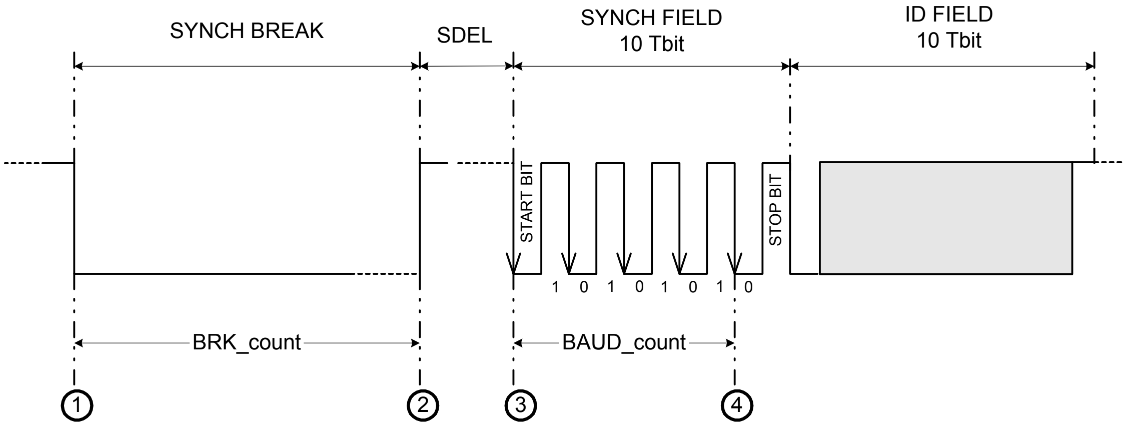
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

13.8.5.2 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 13-17). These values are always calculated during the Header reception for synch field validation (Figure 13-18).

Figure 13-17. Measurements for Synchronization



By measuring the values BRK_count and BAUD_count, a valid synch break sequence can be detected as described in Figure 13-18. The four numbered events in Figure 13-17 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:

$$\text{BAUD\_count} + \text{BAUD\_count} \gg 2 + \text{BAUD\_count} \gg 3 \leq \text{BRK\_count}$$

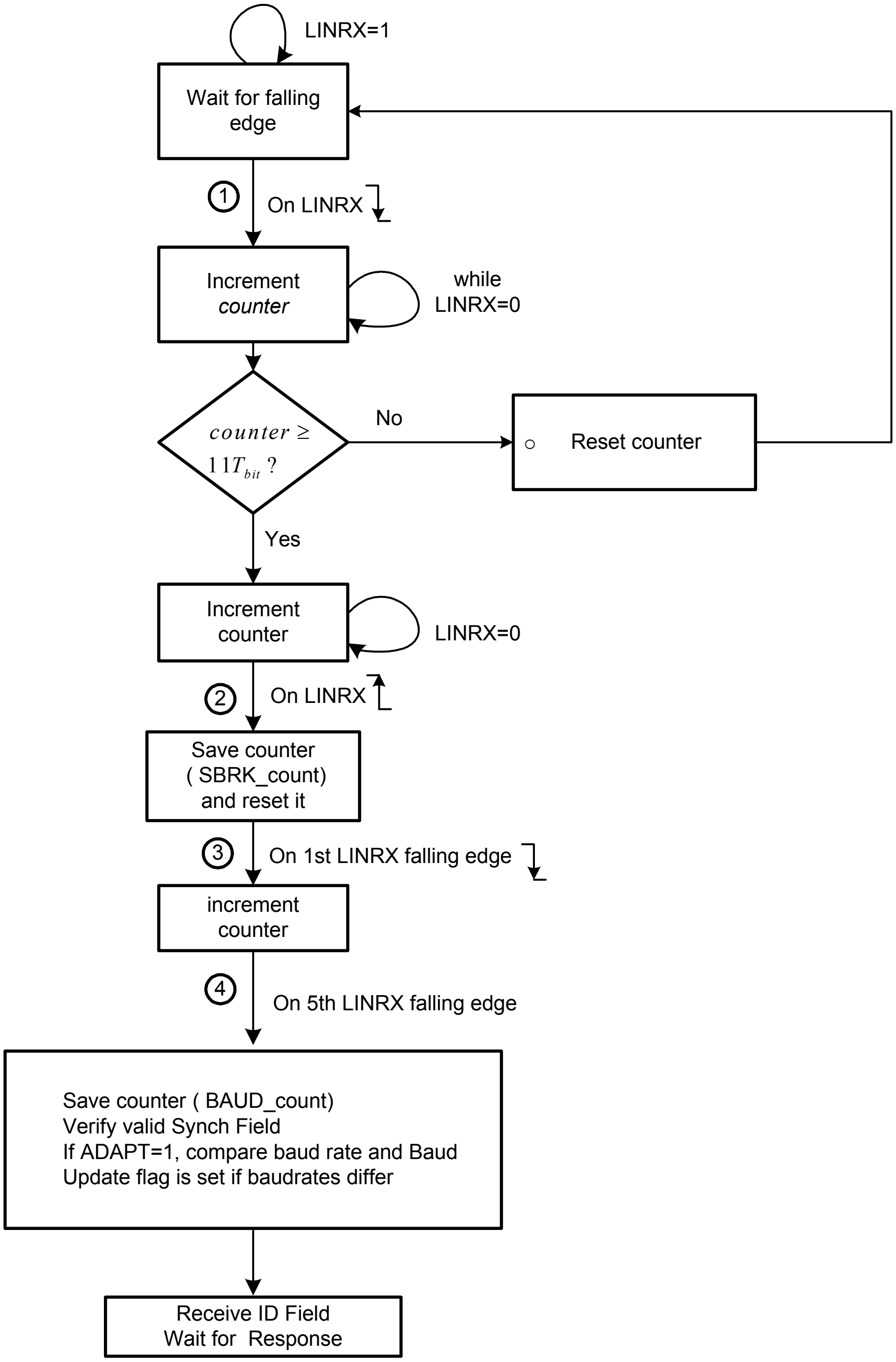
The BAUD_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a T_{bit} unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 13-18, 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 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.

Note:
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 13-18. 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.

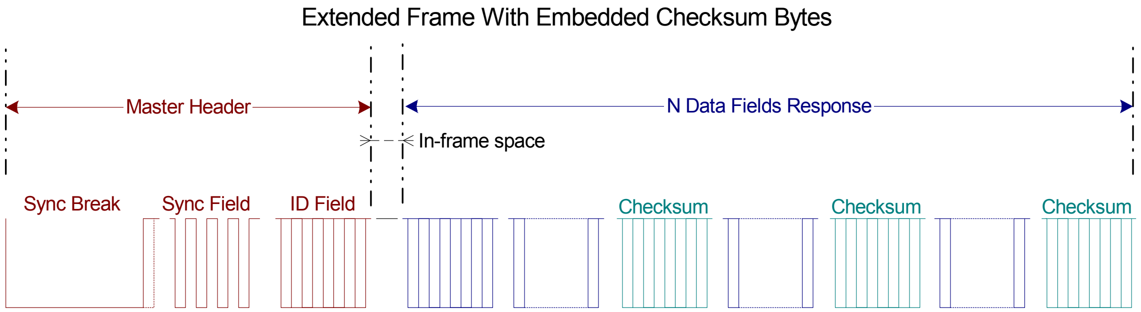
Note:
When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and Set the SWnRST bit to make sure that the internal state machines are back to their normal states

13.8.6 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 13-19. 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 13-19. Optional Embedded Checksum in Response for Extended Frames



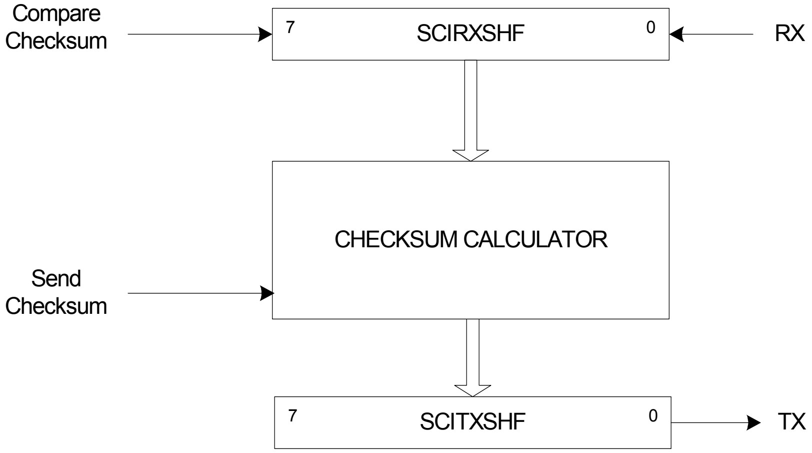
An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle 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 13-20.

Note:
The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

Figure 13-20. Checksum Compare and Send for Extended Frames



13.8.7 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

13.8.7.1 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 13-8. 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

13.8.7.2 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:
If NRE error detection occurs before the bus idle detection (e.g. incomplete header is received and the bus is held recessive until timeout), it can flag the bus idle timeout after 3.96 s (rather than a minimum of 4 s) at 1KHz baud rate (worst case).

13.8.7.3 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 13.12.3 for more details.

13.8.8 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:

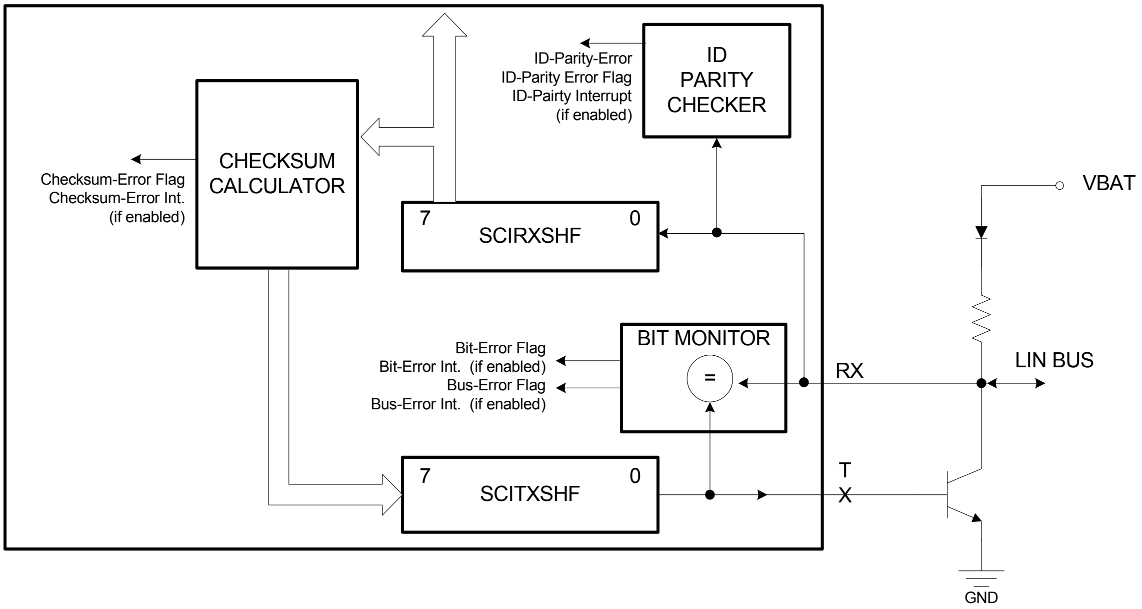
- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

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.

13.8.8.1 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 13-21.

Figure 13-21. TXRX Error Detector



13.8.8.2 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 (i.e. Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Synch Break can be generated (e.g. because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (e.g. 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.

13.8.8.3 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.

$$
\begin{aligned}
P0 &= ID0 \oplus ID1 \oplus ID2 \oplus ID4 \text{ (even parity)} \\
P1 &= \overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5} \text{ (odd parity)}
\end{aligned}
$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 13.8.9 for details.

13.8.8.4 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 13-22) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 13-23) 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 13-22. Classic Checksum Generation at Transmitting Node

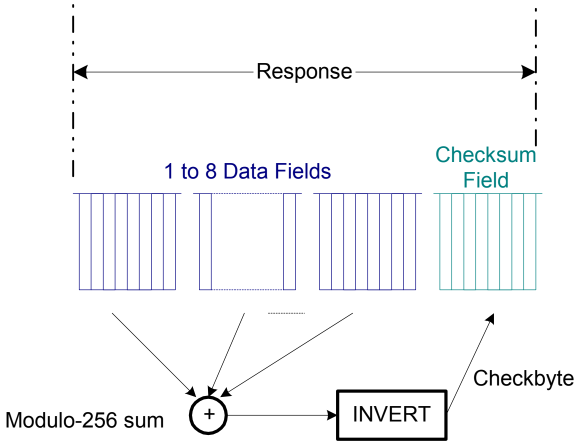
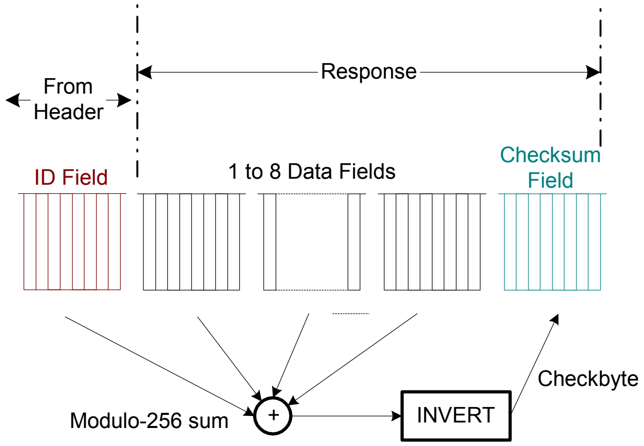


Figure 13-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node



13.8.9 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 13-24. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 13-16) 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 13-24. 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; i.e., compare five most significant bits (MSBs) and filter three least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 \ = \ 0x07$$

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.

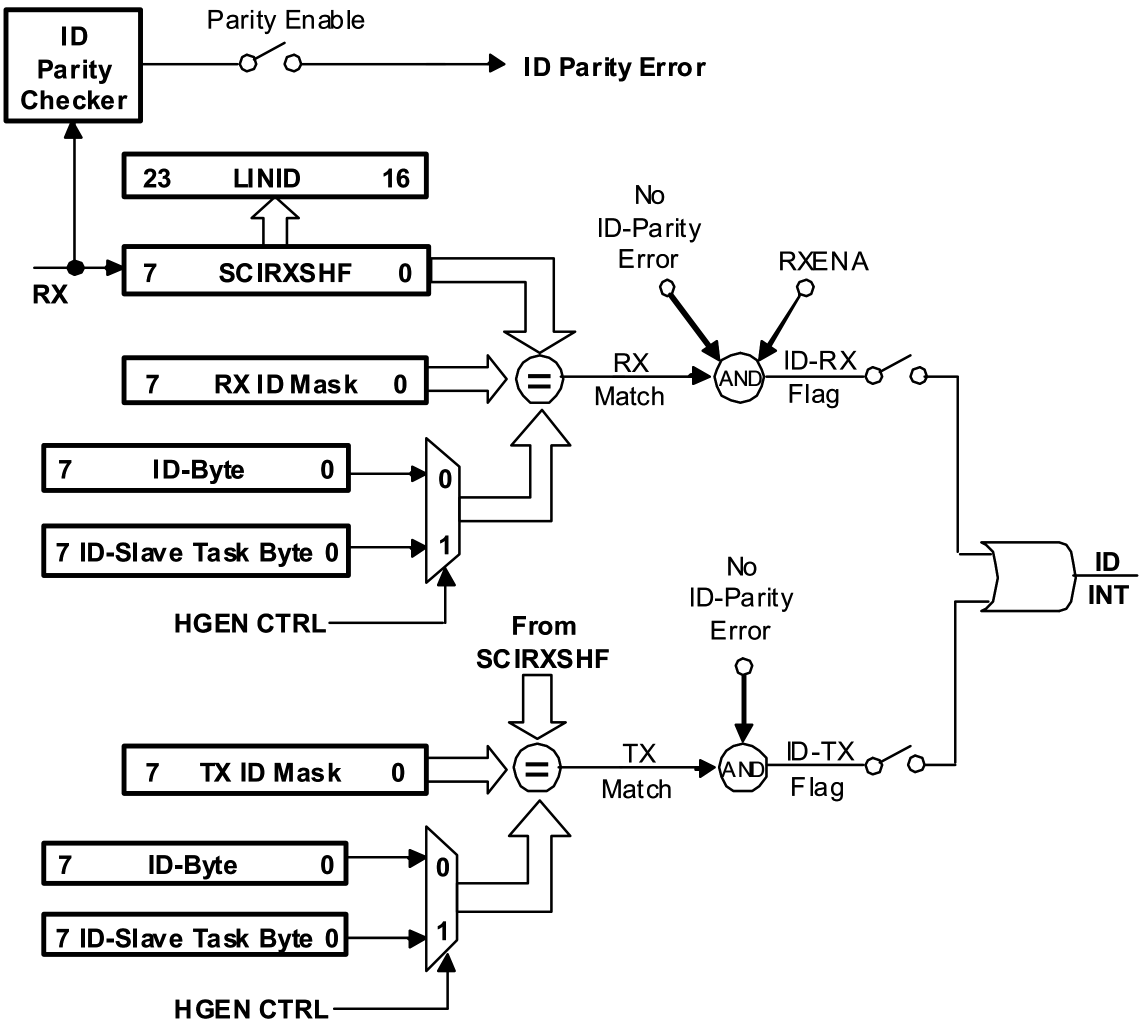
Note:
When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

In multibuffer 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 multibuffer 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 multibuffer 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 multibuffer 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 13-24. ID Reception, Filtering and Validation



13.8.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with N = 1–8) response in interrupt mode or DMA 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 multibuffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multibuffer mode is enabled, or to RD0 if multibuffer 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*), 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 13-8 illustrates the transmit buffers.

A receive interrupt, and a receive ready RXRDY flag set as well as a DMA request (RXDMA) 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. A DMA request can be generated for each received byte or for the entire response depending on whether the multibuffer mode is enabled or not(MBUF MODE bit).

13.8.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with N = 1–8) response in interrupt mode or DMA 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 preloaded in the TXy transmit buffers. Optionally, a DMA transfer could be done on a byte-per-byte basis when multibuffer mode is not enabled (MBUF MODE bit).

The multibuffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multibuffer mode is enabled, or from TD0 to SCITXSHF if multibuffer 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*), 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), and a DMA request (TXDMA) could occur after transmitting a response. A DMA request can be generated for each transmitted byte or for the entire response depending on whether multibuffer mode is enabled or not (MBUF MODE bit).

Figure 13-9 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 multibuffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

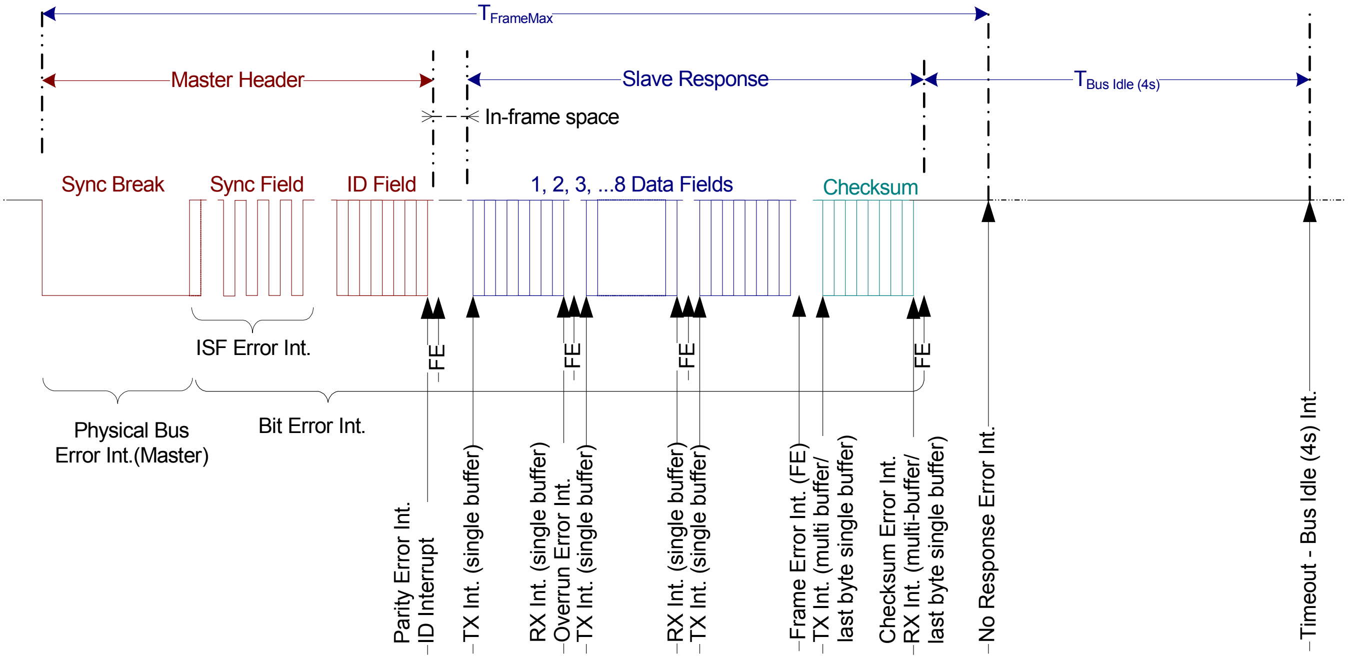
Note:
The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

13.9 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 13.4. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 13-4.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 13-25.

Figure 13-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence



13.10 LIN DMA Interface

LIN DMA Interface uses the SCI DMA interface logic. DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. There are two modes for DMA transfers depending on whether multibuffer mode is enabled or not via the multibuffer enable control bit (MBUF MODE).

13.10.1 LIN Receive DMA Requests

In LIN mode, when the multibuffer option is enabled, if a received response (up to eight data bytes) is transferred to the receive buffers (RDy), then a DMA request is generated. If the multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received. This DMA functionality is enabled/disabled by the user using SET RX DMA / CLR RX DMA bits, respectively.

13.10.2 LIN Transmit DMA Requests

In LIN mode with the multibuffer option enabled, after a transmission (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multibuffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred. This DMA functionality is enabled/disabled by the user using SET TX DMA / CLR TX DMA bits, respectively.

13.11 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 SWnRST is held low the entire time that the LIN is being configured, the order in which the registers are programmed is not important

- Enable LIN by setting RESET bit.
- Clear SWnRST to 0 before configuring the LIN.
- Configure the LINRX and LINTX pins as SCI functional by setting the RX FUNC and TX FUNC bit.
- Select LIN mode by programming LIN MODE bit.
- Select Master or Slave mode by programming the CLOCK bit.
- Select the desired frame format(Checksum, Parity, length control) by programming SCIGCR1.
- Select multi-buffer mode by programming MBUF MODE bit.
- Select the baud rate to be used for communication by programming BRSR.
- Set the Maximum baud rate to be used for communication by programming BRSR.
- Set the CONT bit to make LIN not to 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 to connect the transmitter to the receiver internally. (This feature is used to perform a self-test.)
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set SWnRST to 1 after the SCI is configured.
- Perform Receive or Transmit data. (see Section 13.8.9 / Section 13.8.10 / Section 13.11.2)

13.11.1 Receiving Data

The LIN receiver is enabled to receive messages if 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/LIN function pin.

ID RX flag is set after a valid LIN ID is received with RX Match, generated ID interrupt if enabled.

13.11.1.1 Receiving Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode SCI/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 FE, OE, or PE if any of these error conditions were detected in the received data. These error conditions are supported with configurable Interrupt capability.

User can read the Received data by

1) Polling Receive Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for RXRDY bit and read the data from RD0 byte of LINRD0 register once RXRDY is set high. CPU is unnecessarily overloaded by selecting Polling mode. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET RX INT bit should be set and to use DMA SET RX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment RXRDY is set. If checksum scheme is used once Compare Checksum CC bit is set, the checksum will be compared on the byte that is currently being received, expected to be the checkbyte be enabled during the last byte of the data. CC bit will be cleared once Checksum is received. A CE will immediately be flagged if there is a checksum error.

13.11.1.2 Receiving Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. In this mode SCI/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 same as Single Buffer mode, except that it monitors for the complete frame. Like Single Buffer mode the user can use either Interrupt, DMA or polling method to read the data. The received data has to be read from the LINRD0 and LINRD1 register, based on the number of bytes. For LENGTH less than or equal to 4, Read to LINRD0 register will clear the "RXRDY" flag. For LENGTH greater than 4, Read to LINRD1 register will clear the "RXRDY" flag. If checksum scheme is enabled by setting CC bit during the reception of the data, then the byte that is received after the reception of the programmed no. of data bytes indicated by LENGTH is treated as a checksum byte and compared.

13.11.2 Transmitting Data

The SCI transmitter is enabled if 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 TD0 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.

ID TX flag is set after a valid LIN ID is received with TX Match, generated ID interrupt if enabled.

13.11.2.1 Transmitting Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits it. The flags TXRDY and TX EMPTY 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.

User can transmit data by

1) Polling Transmit Ready Flag

2) Receive Interrupt

3) DMA

In polling method, software can poll for TXRDY bit to go high before writing the data to TD0 register. CPU is unnecessarily overloaded by doing this Polling method. To avoid this user can use either Interrupt or DMA method. To use interrupt method SET TX INT bit should be set and to use DMA SET TX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment TXRDY is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and TD0 are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can be done by either disabling the transmit interrupt(CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit). In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after the current transmission.

Note: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

13.11.2.2 Transmitting Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. Similar to Single Buffer mode the software can use polling, Interrupt or DMA method to write the data to be transmitted. The data to be transmitted has to be written to LINTD0 and LINTD1 register, based on the number of bytes. SCI/LIN waits for data to be written to Byte 0(TD0) of LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after transmission of programmed no. of data bytes indicated by LENGTH field.

13.12 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.

The LIN module enters low-power mode when a sleep command frame is received. A wakeup signal will terminate the sleep mode of the LIN bus. On receipt of the sleep command, the POWERDOWN bit must be set by the application software to make the module enter local low-power mode.

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.

13.12.1 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.

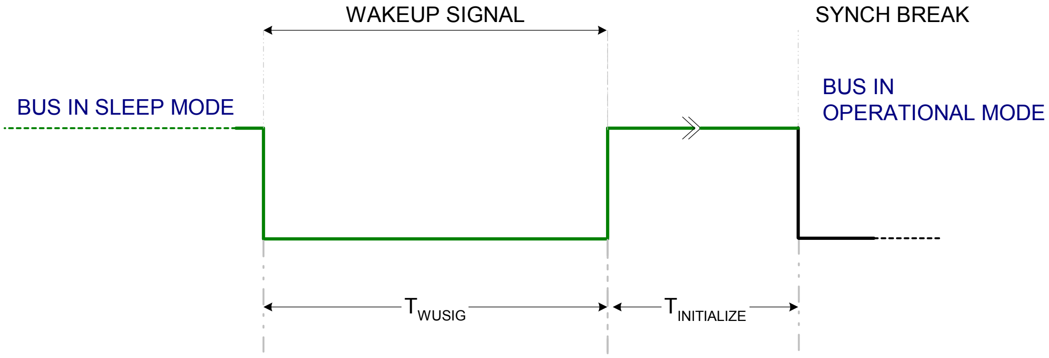
13.12.2 Wakeup

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 13-26. 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 13-26. Wakeup Signal Generation



$$0.25\text{ms} \le T_{WUSIG} \le 5\text{ms}$$

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 dominant level in 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).

13.12.3 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 x programmed prescaler.

Note:
The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

13.13 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.

13.14 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 Figure 13-27. Among the features that can be programmed are the LIN protocol mode, communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration.

Figure 13-27. SCI/LIN Control Registers Summary

Register Offset Address[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 SCIGCR0 Page 499	Reserved															
	Reserved															RESET
0x04 SCIGCR1 Page 500	Reserved						TX ENA	RX ENA	Reserved						CONT	LOOP BACK
	Reserved		STOP EXT FRAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP	SW nRST	LIN MODE	CLOCK	STOP	PARITY	PARITY ENA	TIMMING MODE	COMM MODE
0x08 SCIGCR2 Page 507	Reserved								Reserved						CC	SC
	Reserved							GEN WU	Reserved							POWER DOWN
0x0C SCISETINT Page 509	SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT	Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
	Reserved		SET ID	Reserved			SET RX INT	SET TX INT	SET TOA3 WUS INT	SET TOA WUS INT	Reserved	SET TIME-OUT INT	Reserved		SET WAKEUP INT	SET BRKDT INT
0x10 SCICLEARINT Page 514	CLR BE INT	CLR PBE INT	CLR CE INT	CLR ISFE INT	CLR NRE INT	CLR FE INT	CLR OE INT	CLR PE INT	Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
	Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT	CLR TOA3 WUS INT	CLR TOA WUS INT	Reserved	CLR TIME-OUT INT	Reserved		CLR WAKEUP INT	CLR BRKDT INT
0x14 SCI SETINTLVL Page 519	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	Reserved					SET RX DMA ALL INT LVL	Reserved	
	Reserved		SET ID INT LVL	Reserved			SET RX INT LVL	SET TX INT LVL	SET TOA3 WUS INT LVL	SET TOA WUS INT LVL	Reserved	SET TIME-OUT INT LVL	Reserved		SET WAKEUP INT LVL	SET BRKDT INT LVL

[1] The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18 SCICLEARINTLVL Page 523	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	Reserved					CLRRX DMA ALL INT LVL	Reserved	
	Reserved		CLR ID TX INT LVL	Reserved			CLR ID RX INT LVL	CLR TX INT LVL	CLR TOA3 WUS INT LVL	CLR TOA WUS INT LVL	Reserved	CLR TIME-OUT INT LVL	Reserved		CLR WAKE UP INT LVL	CLR BRKDT INT LVL
0x1C SCIFLR Page 527	BE	PBE	CE	ISFE	NRE	FE	OE	PE	Reserved							
	Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY	TOA3 WUS	TOA WUS	Reserved	TIME-OUT	BUSY	IDLE	WAKE UP	BRKDT
0x20 SCIINTVECT0 Page 539	Reserved															
	Reserved											INTVECT0[4:0]				
0x24 SCIINTVECT1 Page 540	Reserved															
	Reserved											INTVECT1[4:0]				
0x28 SCIFORMAT Page 541	Reserved													LENGTH[2:0]		
	Reserved													CHAR[2:0]		
0x2C BRS Page 543	Reserved	U(2–0)			M(3–0)				PRESCALER P [23:16]							
	PRESCALER P [15:0]															
0x30 SCIED Page 546	Reserved															
	Reserved								ED[7:0]							
0x34 SCIRD Page 547	Reserved															
	Reserved								RD[7:0]							

[1] The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x38 SCITD Page 548	Reserved															
	Reserved								TD[7:0]							
0x3C SCIPIO0 Page 549	Reserved															
	Reserved													TX FUNC	RX FUNC	CLK FUNC
0x40 SCIPIO1 Page 550	Reserved															
	Reserved													TX DIR	RX DIR	CLK DIR
0x44 SCIPIO2 Page 553	Reserved															
	Reserved													TX IN	RX IN	CLK IN
0x48 SCIPIO3 Page 554	Reserved															
	Reserved													TX OUT	RX OUT	CLK OUT
0x4C SCIPIO4 Page 556	Reserved															
	Reserved													TX SET	RX SET	CLK SET
0x50 SCIPIO5 Page 558	Reserved															
	Reserved													TX CLR	RX CLR	CLK CLR
0x54 SCIPIO6 Page 560	Reserved															
	Reserved													TX PDR	RX PDR	CLK PDR

[1] The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x58 SCIPIO7 Page 562	Reserved															
	Reserved													TX PD	RX PD	CLK PD
0x5C SCIPIO8 Page 563	Reserved															
	Reserved													TX PSL	RX PSL	CLK PSL
0x60 LINCOMPARE Page 564	Reserved															
	Reserved						SDEL[1:0]		Reserved					SBREAK[2:0]		
0x64 LINRD0 Page 566	RD0[7:0]								RD1[7:0]							
	RD2[7:0]								RD3[7:0]							
0x68 LINRD1 Page 567	RD4[7:0]								RD5[7:0]							
	RD6[7:0]								RD7[7:0]							
0x6C LINMASK Page 568	Reserved								RX ID MASK[7:0]							
	Reserved								TX ID MASK[7:0]							
0x70 LINID Page 569	Reserved								Received ID[7:0]							
	ID-SlaveTask BYTE(7–0)								ID BYTE[7:0]							
0x74 LINTD0 Page 570	TD0[7:0]								TD1[7:0]							
	TD2[7:0]								TD3[7:0]							

[1] The offset address is relative to the peripheral's beginning address.

Register Offset Address[1]	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
0x78 LINTD1 Page 571	TD4[7:0]								TD5[7:0]							
	TD6[7:0]								TD7[7:0]							
0x7C MBRS Page 572	Reserved															
	Reserved			MBR[12:0]												
0x80 Reserved	Reserved															
	Reserved															
0x84 Reserved	Reserved															
	Reserved															
0x88 Reserved	Reserved															
	Reserved															
0x8C Reserved	Reserved															
	Reserved															
0x90 IODFTCTRL Page 573	BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA	Reserved			PIN SAMPLE MASK		TX SHIFT[2:0]		
	Reserved				IODFTENA[3:0]				Reserved						LPB ENA	RXP ENA

[1] The offset address is relative to the peripheral's beginning address.

13.14.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. Figure 13-28 and Table 13-9 illustrate this register.

Figure 13-28. SCI Global Control Register 0 (SCIGCR0) [offset = 0x00]

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
Reserved															RESET
R-0															RWP-0

R = Read, W = Write in all modes; RWP = Read/Write in privileged mode only; S = Set, U = Undefined, -*n* = Value after reset

Table 13-9. SCI Global Control Register 0 (SCIGCR0) Field Description

Bit	Name	Value	Description
31–1	Reserved		Reads return 0 and 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.

13.14.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 13-29 and Table 13-10 illustrate this register.

Figure 13-29. SCI Global Control Register 1 (SCIGCR1) [offset = 0x04]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						TXENA	RXENA	Reserved						CONT	LOOP BACK
R-0						R/W-0	R/W-0	R-0						R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		STOP EXT F RAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP	SW nRST	LIN MODE	CLOCK	STOP	PARITY	PARITY ENA	TIMING MODE	COMM MODE
R-0	R-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/WL-0	R/W-0	R/W-0	RW-0	R/W-0	R/WC-0	R/WC-0	R/W-0	R/WC-0	R/W-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WP = Write in privileged mode only; WC = Write in sci-compatible mode only; -*n* = Value after reset

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description

Bit	Name	Value	Description
31–26	Reserved		Reads return 0 and 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).

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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 multibuffers.
		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 13-9) 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 guaranteed 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		Reads return 0 and writes have no effect.
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.

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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 LINID occurs. Mask of 0xFF in LINMASK register will result in no match.
		1	ID filtering uses ID-SlaveTask BYTE (recommended). Mask of 0xFF in 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 0xFF and the TX ID MASK must be set to 0xFF.**
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		Multibuffer mode. This bit is effective in LIN and SCI modes.This bit controls receive/transmit buffer usage, i.e., whether the RX/TX multibuffers are used or a single register, RD0/TD0, is used.
		0	The multi-buffer mode is disabled.
		1	The multi-buffer mode is enabled.

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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.
8	SLEEP		SCI sleep. This bit is effective in LIN and SCI modes. 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 13-9) 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 13.12 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 13-11 and Table 13-12. 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 SW nRESET = 0.**

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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**.

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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.
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.

Table 13-10. SCI Global Control Register 1 (SCIGCR1) Field Description (Continued)

Bit	Name	Value	Description
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 13-11. SCI Receiver Status Flags

SCI Flag	Register	Bit	Value After SW nRESET[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 SW nRESET = 0.

Table 13-12. SCI Transmitter Status Flags

SCI Flag	Register	Bit	Value After SW nRESET[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 SW nRESET = 0.

13.14.3 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 13-30 and Table 13-13 illustrate this register.

Figure 13-30. SCI Global Control Register 2 (SCIGCR2) [offset = 0x08

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="14">Reserved</td><td>CC</td><td>SC</td></tr>
<tr><td colspan="14">R-0</td><td>R/WL-0</td><td>R/WL-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">Reserved</td><td>GEN WU</td><td colspan="7">Reserved</td><td>POWER DOWN</td></tr>
<tr><td colspan="7">R-0</td><td>R/W-0</td><td colspan="7">R-0</td><td>R/W-0</td></tr>
</table>

R = Read, W = Write in all modes, WL = Write in LIN mode only; -*n* = Value after reset

Table 13-13. SCI Global Control Register 2 (SCIGCR2) Field Description

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect.
17	CC		Compare checksum. This bit is effective in LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. In non-multibuffer 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 (i.e., during the inter-frame space), then the immediate next byte will be treated as a checksum byte. A CE will immediately be flagged if there is a checksum error. This bit is automatically cleared once the checksum is compared. See Section 13.8.6 for more details.
		0	No checksum compare will occur.
		1	Compare checksum on expected checksum byte.

Table 13-13. SCI Global Control Register 2 (SCIGCR2) Field Description (Continued)

Bit	Name	Value	Description
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-multibuffer mode, the checksum byte will be sent after the current byte transmission. In multibuffer mode, the checksum byte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). See Section 13.8.6 for more details. This byte will be cleared after the checksum byte has been transmitted.
		0	No checksum byte will be sent.
		1	A checksum byte will be sent.
15–9	Reserved		Reads return 0 and 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		Reads return 0 and writes have no effect.
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 13.12 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.

13.14.4 SCI Set Interrupt Register (SCISETINT)

Figure 13-31 and Table 13-14 illustrate this register. Refer Figure 13-31 for details on when different interrupt flags get set in a frame during LIN Mode.

Figure 13-31. SCI Set Interrupt Register (SCISETINT) [offset = 0x0C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT	Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SET ID INT	Reserved			SET RX INT	SET TX INT	SET TOA3 WUS INT	SET TOA WUS INT	Reserve d	SET TIME-OUT INT	Reserved		SET WAKE UP INT	SET BRKDT INT
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; -*n* = Value after reset

Table 13-14. SCI Set Interrupt Register (SCISETINT) Field Description

Bit	Name	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 13-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	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.
23–19	Reserved		Reads return 0 and writes have no effect.

Table 13-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
18	SET RX DMA ALL		Set receive DMA all. This bit is effective in SCI-compatible mode only. This bit determines if a separate interrupt is generated for the address frames sent in multiprocessor communications. When this bit is 0, RX interrupt requests are generated for address frames and DMA requests are generated for data frames. When this bit is 1, RX DMA requests are generated for both address and data frames.
		0	*Read:* The DMA request is disabled for address frames (the receive interrupt request is enabled for address frames). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The DMA request is enabled for address and data frames
17	SET RX DMA		Set receiver DMA. This bit is effective in LIN or SCI-compatible mode. To enable receiver DMA requests, this bit must be set. If it is cleared, interrupt requests are generated depending on bit SCISETINT
		0	*Read:* Receive DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* Receive DMA request is enabled.
16	SET TX DMA		Set transmit DMA. This bit is effective in LIN or SCI-compatible mode. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on SET TX INT bit (SCISETINT).
		0	*Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* Transmit DMA request is enabled.
15–14	Reserved		Reads return 0 and 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 13.8.9 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		Reads return 0 and writes have no effect.

Table 13-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
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.
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		Reads return 0 and 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		Reads return 0 and writes have no effect.

Table 13-14. SCI Set Interrupt Register (SCISETINT) Field Description (Continued)

Bit	Name	Value	Description
1	SET WAKEUP INT		Set wakeup interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wake-up interrupt and thereby exit low-power mode. If enabled, the wake-up 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.

13.14.5 SCI Clear Interrupt Register (SCICLEARINT)

Figure 13-32 and Table 13-15 illustrate this register. //ML: Should we explain purpose of set/clear registers here?

Figure 13-32. SCI Clear Interrupt Register (SCICLEARINT) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR BE INT	SET PBE INT	CLR CE INT	CLR ISFE INT	CLR RE INT	CLR FE INT	CLR OE INT	CLR PE INT	Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT	CLR TOA3 WUS INT	CLR TOA WUS INT	Reserved	CLR TIME-OUT INT	Reserved		CLR WAKE UP INT	CLR BRKDT INT
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description

Bit	Name	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		Set 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.

Table 13-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
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.
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.
23–19	Reserved		Reads return 0 and writes have no effect.

Table 13-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
18	CLR RX DMA ALL		Clear receive DMA all. This bit is effective in SCI mode only. This bit clears the receive DMA request for address frames when set. Only receive data frames generate a DMA request.
		0	*Read:* Receive DMA request for address frames is disabled; Instead, RX interrupt requests are enabled for address frames. Receive DMA requests are still enabled for data frames. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive DMA request for address and data frames is enabled. *Write:* The receive DMA request for address and data frames is disabled.
17	CLR RX DMA		Clear receive DMA request. This bit is effective in LIN or SCI mode. This bit disables the receive DMA request when set.
		0	*Read:* The receive DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive DMA request is enabled. *Write:* The receive DMA request is disabled.
16	CLR TX DMA		Clear transmit DMA request. This bit is effective in LIN or SCI mode. This bit disables the transmit DMA request when set.
		0	*Read:* The transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The transmit DMA request is enabled. *Write:* The transmit DMA request is disabled.
15–14	Reserved		Reads return 0 and 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		Reads return 0 and 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.

Table 13-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
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.
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		Reads return 0 and 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		Reads return 0 and 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.

Table 13-15. SCI Clear Interrupt Register (SCICLEARINT) Field Description (Continued)

Bit	Name	Value	Description
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.

13.14.6 SCI Set Interrupt Level Register (SCISETINTLVL)

Figure 13-33 and Table 13-16 illustrate this register.

Figure 13-33. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 0x14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	Reserved					SET RX DMA ALL INT LVL	Reserved	
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SET ID INT LVL	Reserved			SET RX INT LVL	SET TX INT LVL	SET TOA3 WUS INT LVL	SET TOA WUS INT LVL	Reserved	SET TIME-OUT INT LVL	Reserved		SET WAKE UP INT LVL	SET BRKDT INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description

Bit	Name	Value	Description
31	SET BE INT LV		Set bit error 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.
30	SET PBE INT LVL		Set physical bus error 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.
29	SET CE INT LVL		Set checksum-error 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.
28	SET ISFE INT LVL		Set inconsistent-synch-field-error 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.

Table 13-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
27	SET NRE INT LVL		Set no-response-error 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.
26	SET FE INT LVL		Set framing-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.
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.
23–19	Reserved		Reads return 0 and writes have no effect.
18	SET RX DMA ALL INT LVL		Set receive DMA all interrupt levels. This bit is effective in SCI mode only.
		0	*Read:* The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The receive interrupt request for address frames is mapped to the INT1 line.
17–14	Reserved		Reads return 0 and 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		Reads return 0 and writes have no effect.

Table 13-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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		Reads return 0 and 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.

Table 13-16. SCI Set Interrupt Level Register (SCISETINTLVL) Field Description (Continued)

Bit	Name	Value	Description
0	SET BRKDT INT		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.

13.14.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 13-34 and Table 13-17 illustrate this register.

Figure 13-34. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 0x18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	Reserved					CLRRX DMA ALL INT LVL	Reserved	
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0					R/WC-0	R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLR ID IX INT LVL	Reserved			CLR ID RX INT LVL	CLR TX INT LVL	CLR TOA3 WUS INT LVL	CLR TOA WUS INT LVL	Reserved	CLR TIME-OUT INT LVL	Reserved		CLR WAKE UP INT LVL	CLR BRKDT INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0	R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description

Bit	Name	Value	Description
31	CLR BE INT LVL		Clear bit error 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:* 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 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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 13-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
27	CLR NRE INT LVL		Clear no-response-error 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
26	CLR FE INT LVL		Clear framing-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:* 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 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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
23–19	Reserved		Reads return 0 and writes have no effect.
18	CLR RX DMA ALL INT LVL		Clear receive DMA interrupt level. This bit is effective in SCI-compatible mode only.
		0	*Read:* The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive interrupt request for address frames is mapped to the INT1 line. *Write:* The receive interrupt request for address frames is mapped to the INT0 line.
17–14	Reserved		Reads return 0 and writes have no effect.

Table 13-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
13	CLR ID INT LVL		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
12–10	Reserved		Reads return 0 and writes have no effect.
9	CLR RX INT LVL		Clear 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:* 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 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:* 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 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:* 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 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
5	Reserved		Reads return 0 and writes have no effect.
4	CLR TIMEOUT INT LVL		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 13-17. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Description (Continued)

Bit	Name	Value	Description
3–2	Reserved		Reads return 0 and writes have no effect.
1	CLR WAKEUP INT LVL		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT		Clear 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:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

13.14.8 SCI Flags Register (SCIFLR)

Figure 13-35 and Table 13-18 illustrate this register.

Figure 13-35. SCI Flags Register (SCIFLR) [offset = 0x1C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BE	PBE	CE	ISFE	NRE	FE	OE	PE	Reserved							
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY	TOA3 WUS	TOA WUS	Reserved	TIME-OUT	BUSY	IDLE	WAKE UP	BRKDT
R-0	R/WL-0	R/WL-0	R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1	R/WL-0	R/WL-0	R-0	R/WL-0	R/W-0	R-0	R/WL-0	R/WC-0

R = Read in all modes; W = Write in all modes; WC = Write in sci-compatible mode only; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-18. SCI Flags Register (SCIFLR) Field Description

Bit	Name	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 13.8.8 for more information. The bit error flag is cleared by any of the following: • Setting of the SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 13.8.8 for more information. The physical bus error flag is cleared by the following: • Setting of the SW nRST 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 (e.g. because of a bus shortage to VBAT) or if no Synch Break Delimiter can be generated (e.g. 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.
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 13.8.8 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 SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 13.8.5.2 for more information. The inconsistent synch field error flag is cleared by the following: • Setting of the SW nRST 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:* This 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 13.8.7 for more information. The no-response error flag is cleared by the following: • Setting of the SW nRST 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 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	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 (Section 13.14.4). The framing error flag is cleared by the following: • Setting of the SW nRST 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 multibuffer 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. *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 SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 SW nRST 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		Reads return 0 and writes have no effect.
14	ID RX Flag		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 13.8.9 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 SW nRST 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:* This bit is cleared to 0.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
13	ID TX Flag		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 13.8.9 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 SW nRST 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 a TX match. *Write:* This bit is cleared to 0.
12	RXWAKE		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. RXWAKE is cleared by the following: • Setting of the SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
11	TX EMPTY		Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty. In multibuffer mode, this flag indicates the TDx registers and shift register (SCITXSHF) are empty. In non-multibuffer mode, this flag indicates the LINTD0 byte and the shift register (SCITXSHF) are empty.
			Note: The RESET bit, an active SW nRESET (SCIGCR1[7]) or a system reset sets this bit. This bit does not cause an interrupt request.
			SCI mode or LIN non-multibuffer 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 multibuffer mode:
		0	Multibuffer or shift register (or all) are loaded with data
		1	Multibuffer and shift registers are all empty.
10	TXWAKE		Transmitter wakeup method select. This bit is effective in SCI mode only. The TXWAKE 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 SW nRESET 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).

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
9	RXRDY		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,* RXRDY is set once a valid frame is received in multibuffer mode, a valid frame being a message frame received with no errors. In *non-multibuffer mode,* RXRDY 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 RXRDY flag bit is set if the interrupt-enable bit is set (SCISETINT[9]); RXRDY is cleared by the following: • Setting of the SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
8	TXRDY		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 multibuffer 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 interrupt enable bit TXINT is set. **Note:** **1) TXRDY is also set to 1 either by setting of the RESET bit, enabling SW nRST 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 multibuffers are full.
		1	The multibuffers are ready to receive the next character(s). For more information on transmit interrupt handling, see the SCI document for compatibility mode and Section 13.8.9 for LIN mode.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	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 SW nRST 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 13.12.3 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 3 wakeup signals and 1.5 seconds time. *Write:* The bit will be 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 SW nRST 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 13.12.3 for more information.
		0	*Read:* No timeout occurs after one wakeup signal (150 ms). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A timeout occur after one wakeup signal. *Write:* The bit will be cleared to 0.
5	Reserved		Reads return 0 and 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 4 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 SW nRST 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 13.8.7 for more information.
		0	*Read:* No bus idle has been detected since this bit was cleared. *Write:* A write of 0 to this bit has no effect.
		1	*Read:* A LIN bus idle has been detected. *Write:* The bit is cleared to 0.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 SW nRST 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.
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		Wake-up 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 SW nRST 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.

Table 13-18. SCI Flags Register (SCIFLR) Field Description (Continued)

Bit	Name	Value	Description
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 BRKDT INT ENA bit is set. The BRKDT bit is reset by the following: • Setting of the SW nRST 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:* Writing a 1 to this bit clears it to 0.

13.14.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 13-36 and Table 13-19 illustrate this register.

Figure 13-36. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 0x20]

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
Reserved											INTVECT0[4:0]				
R-0											R-0				

R = Read in all modes; -*n* = Value after reset

Table 13-19. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Description

Bit	Name	Value	Description
31–5	Reserved		Reads return 0 and writes have no effect.
4–0	INTVECT0[4:0]	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 13-4 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).**

13.14.10SCI Interrupt Vector Offset 1 (SCIINTVECT1)

Figure 13-37 and Table 13-20 illustrate this register.

Figure 13-37. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 0x24

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
Reserved											INTVECT1[4:0]				
R-0											R-0				

R = Read in all modes; -*n* = Value after reset

Table 13-20. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Description

Bit	Name	Value	Description
31–5	Reserved		Reads return 0 and writes have no effect.
5–0	INTVECT1[4:0]	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 13-4 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).**

13.14.11SCI Format Control Register (SCIFORMAT)

Figure 13-38 and Table 13-21 illustrate this register.

Figure 13-38. SCI Format Control Register (SCIFORMAT) [offset = 0x28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													LENGTH[2:0]		
R-0													R/W-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CHAR[2:0]		
R-0													R/WC-0		

R = Read in all modes; W = Write in all modes; WC = Write in SCI-compatible mode only; -*n* = Value after reset

Table 13-21. SCI Format Control Register (SCIFORMAT) Field Description

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18–16	LENGTH[2:0]	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.
		000	The response field has 1 byte/character.
		001	The response field has 2 bytes/characters.
		010	The response field has 3 bytes/characters.
		011	The response field has 4 bytes/characters.
		100	The response field has 5 bytes/characters.
		101	The response field has 6 bytes/characters.
		110	The response field has 7 bytes/characters.
		111	The response field has 8 bytes/characters.
15–3	Reserved		Reads return 0 and writes have no effect.

Table 13-21. SCI Format Control Register (SCIFORMAT) Field Description (Continued)

Bit	Name	Value	Description
2–0	CHAR[2:0]	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.**
		000	The character is 1 bit long.
		001	The character is 2 bits long.
		010	The character is 3 bits long.
		011	The character is 4 bits long.
		100	The character is 5 bits long.
		101	The character is 6 bits long.
		110	The character is 7 bits long.
		111	The character is 8 bits long.

13.14.12Baud Rate Selection Register (BRS)

This section describes the baud rate selection register. Figure 13-39 and Table 13-22 illustrate this register.

Figure 13-39. Baud Rate Selection Register (BRS) [offset = 0x2C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	U[2:0]			M[3:0]				PRESCALER P [23:16]							
R-0	R/W-0			R/W-0				R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESCALER P [15:0]															
R/W-0															

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-22. Baud Rate Selection Register (BRS) Field Description

Bit	Name	Value	Description
31	Reserved		This bit is always read as 0. Writes have no effect.
30–28	U[2:0]	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 superfine tuning of the fractional baud rate with seven more intermediate values for each of the M fractional divider values. See Section 13.8.4.1 for more details.
27-24	M[3:0]	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 13.8.4.1 for more details.

Table 13-22. Baud Rate Selection Register (BRS) Field Description (Continued)

Bit	Name	Value	Description
23–0	PRESCALER P [23:0]	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: *Asynchronous baud value* = $\left\lfloor \frac{\text{VCLK Frequency}}{16\left(P + 1 + \frac{M}{16}\right)} \right\rfloor$ *Isosynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{P + 1}\right)$ For P = 0, *Asynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{32}\right)$ *Isosynchronous baud value* = $\left(\frac{\text{VCLK Frequency}}{2}\right)$ Table 13-23 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode..

Table 13-23. 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.

13.14.13SCI 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.

13.14.13.1Receiver Emulation Data Buffer (SCIED)

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. Figure 13-40 and Table 13-24 illustrate this register.

Figure 13-40. Receiver Emulation Data Buffer (SCIED) [offset = 0x30

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
Reserved								ED[7:0]							
R-0								R-0							

R = Read in all modes; W = Write in SCI-compatible mode only; *-n* = Value after reset

Table 13-24. Receiver Emulation Data Buffer (SCIED) Field Description

Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	ED[7:0]	0–FFh	Emulator data. This bit is effective in compatibility SCI 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.

13.14.13.2Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. Figure 13-41 and Table 13-25 illustrate this register.

Figure 13-41. Receiver Data Buffer (SCIRD) [offset = 0x34

31–16
Reserved
R-0

15–8	7–0
Reserved	RD[7:0]
R-0	R-0

R = Read in all modes; W = Write in SCI-compatible mode only; -*n* = Value after reset

Table 13-25. Receiver Data Buffer (SCIRD) Field Description

Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	RD[7:0]		Receiver data. This bit is effective in compatibility SCI 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.

13.14.13.3Transmit Data Buffer Register (SCITD)

Data to be transmitted is written to the SCITD register. Figure 13-42 and Table 13-26 illustrate this register.

Figure 13-42. Transmit Data Buffer Register (SCITD) [offset = 0x38

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
Reserved								TD[7:0]							
R-0								R/W-0							

R = Read in all modes; W = Write in SCI-compatible mode only; -*n* = Value after reset

Table 13-26. Transmit Data Buffer Register (SCITD) Field Description

Bit	Name	Value	Description
31–8	Reserved		This bit is always read as 0. Writes have no effect.
7–0	TD[7:0]	0–FFh	Transmit data. This bit is effective in compatibility SCI 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 TX INT ENA 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.

13.14.14SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 13-43 and Table 13-27 illustrate this register.

Figure 13-43. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 0x3C

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
Reserved													TX FUNC	RX FUNC	CLK FUNC
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-27. SCI Pin I/O Control Register 0 (SCIPIO0) Field Description

Bit	Name	Value	Description
31–3	Reserved		This bit is always read as 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	CLK FUNC		Clock function. This bit is effective in LIN or SCI mode. This bit defines the function of pin SCICLK. In LIN mode, the SCICLK pin can only be a digital I/O pin.
		0	SCICLK is a general-purpose digital I/O pin.
		1	SCICLK is the SCI serial clock pin.

13.14.15SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 13-44 and Table 13-28 illustrate this register.

Figure 13-44. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 0x40]

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
Reserved													TX DIR	RX DIR	CLK DIR
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-28. SCI Pin I/O Control Register 1 (SCIPIO1)

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 13-29 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 13-30 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.

Table 13-28. SCI Pin I/O Control Register 1 (SCIPIO1)

Bit	Name	Value	Description
0	CLK DIR		Clock pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the SCICLK pin. The direction is defined differently depending upon the value of the CLK FUNC bit.
			If CLK FUNC = 0 (SCICLK = general-purpose I/O), then the CLK DIR bit has the following function:
		0	SCICLK pin is a general-purpose input pin.
		1	SCICLK pin is a general-purpose output pin.
			If CLK FUNC = 1 (SCICLK pin has SCI clock functionality) and an internal clock has been selected then the CLK DIR bit has the following functionality, in SCI compatibility mode only:
		0	The internal SCI clock is not output on SCICLK pin.
		1	The internal SCI clock is output on SCICLK pin.
			If CLK FUNC = 1 (SCICLK pin has SCI clock functionality) and an external clock has been selected (SCIGCR1[5] = 0), then the SCI expects an external clock signal not to exceed VCLK/16 on the SCICLK pin (see Table 13-31 for a description of the behavior of the pin with different bits set).

Table 13-29. 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 13-30. 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.

Table 13-31. SCICLK Pin Control

Function	CLK IN[1]	CLK OUT	CLK FUNC	CLK DIR
SCICLK (output)	X	X	1	1
SCICLK (input)	X	X	1	0
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

[1] CLK IN is a read-only bit. Its value always reflects the level of the SCICLK pin.

13.14.16SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 13-45 and Table 13-32 illustrate this register.

Figure 13-45. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 0x44]

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
Reserved													TX IN	RX IN	CLK IN
R-0													R-X	R-X	R-X

R = Read in all modes; -*n* = Value after reset; -*x* = Indeterminate

Table 13-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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	CLK IN		Clock pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the SCICLK pin.
		0	The SCICLK pin is at logic low (0).
		1	The SCICLK pin is at logic high (1).

13.14.17SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 13-46 and Table 13-33 illustrate this register.

Figure 13-46. SCI Pin I/O Control Register 3 (SCIPIO3) [offset =0x48

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
Reserved													TX OUT	RX OUT	CLK OUT
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 13-29 for an explanation of this bit's effect in combination with other bits.
		0	The output on the LINTX 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 SC 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 13-30 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)

Table 13-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Description (Continued)

Bit	Name	Value	Description
0	CLK OUT		Clock data out. This bit is effective in LIN or SCI mode. This bit contains the data to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.) See Table 13-31 for an explanation of this bit's effect in combination with other bits.
		0	Output value on SCICLK is a 0 (logic low).
		1	Output value on SCICLK is a 1 (logic high). (Output voltage is V_{OH} or higher if CLKPDR = 0 and output is in high impedance state if CLKPDR = 1)

13.14.18SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 13-47 and Table 13-34 illustrate this register.

Figure 13-47. SCI Pin I/O Control Register 4 (SCIPIO4) [offset =0x4C .

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
Reserved													TX SET	RX SET	CLK SET
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 13-29 for an explanation of this bit's effect in combination with other bits.
		0	*Read:* The output on LINTX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on LINTX 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 13-30 for an explanation of this bit's effect in combination with the other bits.
		0	*Read:* The output on LINRX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on LINRX is at logic high (1).

Table 13-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Description (Continued)

Bit	Name	Value	Description
0	CLK SET		Clock pin set. This bit is effective in LIN or SCI mode. This bit sets the data to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.) See Table 13-31 for an explanation of this bit's effect in combination with the other bits.
		0	*Read:* The output on SCICLK is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on SCICLK is at logic high (1).

13.14.19SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 13-48 and Table 13-35 illustrate this register.

Figure 13-48. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 0x50

31–16
Reserved
R-0

15–3	2	1	0
Reserved	TX CLR	RX CLR	CLK CLR
R-0	R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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 is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on LINTX is at logic high (1). *Write:* The output on LINTX 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 is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on LINRX is at logic high (1). *Write:* The output on LINRX is at logic low (0).

Table 13-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Description (Continued)

Bit	Name	Value	Description
0	CLK CLR		Clock pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.)
		0	*Read:* The output on SCICLK is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on SCICLK is at logic high (1). *Write:* The output on SCICLK is at logic low (0).

13.14.20SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 13-49 and Table 13-36 illustrate this register.

Figure 13-49. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 0x54

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
Reserved													TX PDR	RX PDR	CLK PDR
R-0													R/W-0	R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and writes have no effect.
2	TX PDR		Transmit pin open drain enable. This bit is effective in LIN or SC 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 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 high impedance if TXOUT =1.
1	RX PDR		Receive pin open train enable. This bit is effective in LIN or SC 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 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 high impedance if RXOUT =1.

Table 13-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Description (Continued)

Bit	Name	Value	Description
0	CLK PDR		Clock pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin SCICLK if the following conditions are met: • CLK FUNC = 0 (SCICLK pin is a general-purpose I/O.) • CLK DIR = 1 (SCICLK pin is a general-purpose output.)
		0	Open drain functionality is disabled; the output voltage is V_{OL} or lower if CLKOUT =0 and V_{OH} or higher if CLKOUT =1.
		1	Open drain functionality is enabled; the output voltage is V_{OL} or lower if CLKOUT =0 and high impedance if CLKOUT =1.

13.14.21SCI Pin I/O Control Register 7 (SCIPIO7)

Figure 13-50 and Table 13-37 illustrate this register.

Figure 13-50. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 0x58]

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
Reserved													TX PD	RX PD	CLK PD
R-0													R/W-n	R/W-n	R/W-n

R = Read in all modes; W = Write in all modes; -*n* = Value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 13-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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	CLK PD		Clock pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables the pull control capability on the input pin SCICLK.
		0	Pull control on the SCICLK pin is enabled.
		1	Pull control on the SCICLK pin is disabled.

13.14.22SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 13-51 and Table 13-38 illustrate this register.

Figure 13-51. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 0x5C

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
Reserved													TX PSL	RX PSL	CLK PSL
R-0													R/W-*n*	R/W-*n*	R/W-*n*

R = Read in all modes; W = Write in all modes; -*n* = Value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 13-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Description

Bit	Name	Value	Description
31–3	Reserved		Reads return 0 and 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	CLK PSL		CLK pin pull select. This bit is effective in LIN or SCI mode. It selects pull type in the input pin SCICLK.
		0	Pull control on the SCICLK pin is enabled.
		1	Pull control on the SCICLK pin is disabled.

13.14.23 LIN Compare Register (LINCOMPARE)

Figure 13-52 and Table 13-39 illustrate this register.

Figure 13-52. LIN Compare Register (LINCOMPARE) [offset = 0x60

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
Reserved						SDEL[1:0]		Reserved					SBREAK[2:0]		
R-0						RWP-0		R-0					RWP-0		

R = Read in all modes; WL = Write in LIN mode only; RWP = Read/Write in privileged mode only; -*n* = Value after reset

Table 13-39. LIN Compare Register (LINCOMPARE) Field Description

Bit	Name	Value	Description
31–10	Reserved		Reads return 0 and writes have no effect.
9–8	SDEL[1:0]		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 0x00. The formula to program the value (in T_{bits}) for the synchronization delimiter is $T_{SDEL} = (SDEL+1)T_{bit}$
		00	The synch delimiter has 1 T_{bit}.
		01	The synch delimiter has 2 T_{bit}.
		10	The synch delimiter has 3 T_{bit}.
		11	The synch delimiter has 4 T_{bit}.
7-3	Reserved		Reads return 0 and writes have no effect.

Table 13-39. LIN Compare Register (LINCOMPARE) Field Description (Continued)

Bit	Name	Value	Description
2–0	SBREAK[2:0]		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 0x0, which adds nothing to the automatically generated SYNCH BREAK.** The formula to program the value (in T_{bits}) for the SYNCH BREAK is $T_{SYNBRK} = 13T_{bit} + SBREAK \times T_{bit}$
		000	The synch break has no additional T_{bit}.
		001	The synch break has 1 additional T_{bit}.
		010	The synch break has 2 additional T_{bit}.
		011	The synch break has 3 additional T_{bit}.
		100	The synch break has 4 additional T_{bit}.
		101	The synch break has 5 additional T_{bit}.
		110	The synch break has 6 additional T_{bit}.
		111	The synch break has 7 additional T_{bit}.

13.14.24LIN Receive Buffer 0 Register (LINRD0)

Figure 13-53 and Table 13-40 illustrate this register.

Figure 13-53. LIN Receive Buffer 0 Register (LINRD0) [offset = 0x64

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD0[7:0]								RD1[7:0]							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD2[7:0]								RD3[7:0]							
R-0								R-0							

R = Read in all modes; -*n* = Value after reset

Table 13-40. LIN Receive Buffer 0 Register (LINRD0) Field Description

Bit	Name	Value	Description
31–24	RD0[7:0]	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[7:0]	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[7:0]	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[7:0]	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.

13.14.25LIN Receive Buffer 1 Register (LINRD1)

Figure 13-54 and Table 13-41 illustrate this register.

Figure 13-54. LIN Receive Buffer 1 Register (RD1) [offset = 0x68]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD4[7:0]								RD5[7:0]							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD6[7:0]								RD7[7:0]							
R-0								R-0							

R = Read in all modes; -*n* = Value after reset

Table 13-41. LIN Receive Buffer 1 Register (RD1) Field Description

Bit	Name	Value	Description
31–24	RD4[7:0]	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[7:0]	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[7:0]	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[7:0]	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.

13.14.26LIN Mask Register (LINMASK)

Figure 13-55 and Table 13-42 illustrate this register.

Figure 13-55. LIN Mask Register (LINMASK) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RX ID MASK[7:0]							
R-0								R/WL-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TX ID MASK[7:0]							
R-0								R/WL-0							

R = Read in all modes; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-42. LIN Mask Register (LINMASK) Field Description

Bit	Name	Value	Description
31–24	Reserved		Reads of these bits return 0 and writes have no effect.
23–16	RX ID MASK[7:0]	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		Reads of these bits return 0 and writes have no effect.
7–0	TX ID MASK[7:0]	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.

13.14.27LIN Identification Register (LINID)

Figure 13-56 and Table 13-43 illustrate this register.

Figure 13-56. LIN Identification Register (LINID) [offset = 0x70

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RECEIVED ID[7:0]							
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID-SlaveTask BYTE[7:0]								ID BYTE[7:0]							
R/WL-0								R/WL-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-43. LIN Identification Register (LINID) Field Description

Bit	Name	Value	Description
31–24	Reserved		Reads return 0 and writes have no effect.
23–16	Received ID[7:0]	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[7:0]	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[7:0]	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 HGENCTRL = 0.

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 0xFF and the TX ID MASK must be set to 0xFF.

13.14.28LIN Transmit Buffer 0 Register(LINTD0)

Figure 13-57 and Table 13-44 illustrate this register.

Figure 13-57. LIN Transmit Buffer 0 Register (LINTD0) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD0[7:0]								TD1[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD2[7:0]								TD3[7:0]							
R/W-0								R/W-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-44. LIN Transmit Buffer 0 Register (LINTD0) Field Description

Bit	Name	Value	Description
31-24	TD0[7:0]	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[7:0]	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[7:0]	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[7:0]	0–FFh	8-Bit transmit buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

13.14.29LIN Transmit Buffer 1 Register (LINTD1)

Figure 13-58 and Table 13-45 illustrate this register.

Figure 13-58. LIN Transmit Buffer 1 Register (LINTD1) [offset = 0x78

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD4[7:0]								TD5[7:0]							
R/W-0								R/W-0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TD6[7:0]								TD7[7:0]							
R/W-0								R/W-0							

R = Read in all modes; W = Write in all modes; -*n* = Value after reset

Table 13-45. LIN Transmit Buffer 1 Register (LINTD1) Field Description

Bit	Name	Value	Description
31-24	TD4[7:0]	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[7:0]	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[7:0]	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[7:0]	0–FFh	8-Bit transmit buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

13.14.30Maximum Baud Rate Selection Register (MBRS)

Figure 13-59 and Table 13-46 illustrate this register.

Figure 13-59. Maximum Baud Rate Selection Register (MBRS) [offset = 0x7C

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
Reserved			MBR[12:0]												
R-0			R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-1	R/WL-0	R/WL-1	R/WL-1	R/WL-0	R/WL-0

R = Read in all modes; WL = Write in LIN mode only; -*n* = Value after reset

Table 13-46. Maximum Baud Rate Selection Register (MBRS) Field Description

Bit	Name	Value	Description
31–13	Reserved		Reads return 0 and writes have no effect.
12–0	MBR[12:0]	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 13.8.5.2) 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 mistakingly 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. $$\mathrm{MBR} = \frac{0.9\mathrm{xVCLK}}{\mathrm{maxbaudrate}}$$

13.14.31Input/Output Error Enable (IODFTCTRL) Register

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. Figure 13-60 and Table 13-47 illustrate this register. After the basic SCI/LIN module configuration, enable the required Error mode to be created followed by IODFT Key enable.

Figure 13-60. Input/Output Error Enable Register (IODFTCTRL) [offset = 0x90]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA	Reserved			PIN SAMPLE MASK[1:0]		TX SHIFT [2:0]		
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R-0	R/W-0	R/WC-0	R/WC-0	R-0			R/W-0	R/W-0	R/W-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				IODFTENA[3:0]				Reserved						LPB ENA	RXP ENA
R-0				R/WP-0	R/WP-1	R/WP-0	R/WP-1	R-0						R/WP-0	R/WP-0

R = Read in all modes; W = Write in all modes; WL = Write in LIN mode only; WC = Write in sci-compatible mode only; WP = Write in privilege mode only.
-*n* = Value after reset

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

Table 13-47. Input/Output Error Enable Register (IODFTCTRL) Field Description

Bit	Name	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.

Table 13-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
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		Reads return 0 and 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.
25	PEN		Parity error enable. This bit is effective in compatibility 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	BRKD TENA		Break detect error enable. This bit is effective in SCI-compatibility 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_{BITS} so that a BRKDT error occurs.
23–21	Reserved		Reads return 0 and writes have no effect.

Table 13-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
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).**
		00	No mask is used.
		01	Invert the TX Pin value at TBIT_CENTER.
		10	Invert the TX Pin value at TBIT_CENTER + SCLK.
		11	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.
		000	No delay occurs.
		001	The value is delayed by 1 SCLK.
		010	The value is delayed by 2 SCLK.
		011	The value is delayed by 3 SCLK.
		100	The value is delayed by 4 SCLK.
		101	The value is delayed by 5 SCLK.
		110	The value is delayed by 6 SCLK.
		111	The value is delayed by 7 SCLK.
15–12	Reserved		Reads return 0 and writes have no effect.
11–8	IODFTENA		IO DFT enable key. Write access permitted in Privilege mode only/
		1010	IODFT is enabled.
		All other values	IODFT is disabled.
7–2	Reserved		Reads return 0 and writes have no effect.
1	LPBENA		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.

Table 13-47. Input/Output Error Enable Register (IODFTCTRL) Field Description (Continued)

Bit	Name	Value	Description
0	RXPENA		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 loop-back mode)
		0	Analog loopback through the transmit pin is enabled.
		1	Analog loopback through the receive pin is enabled.

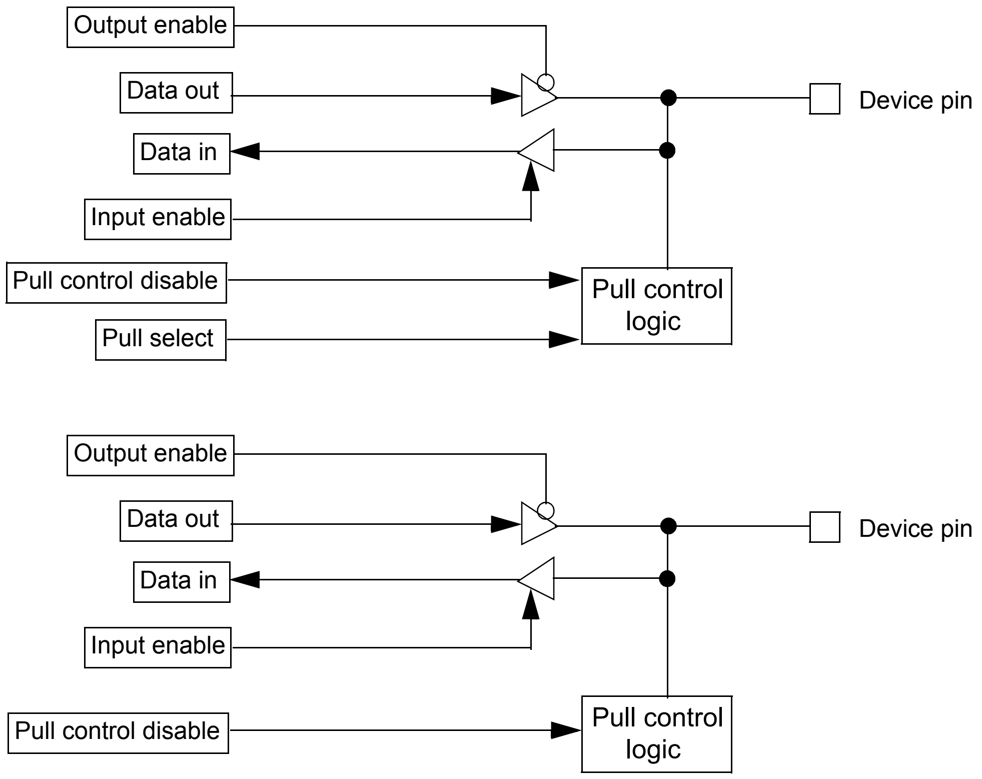
13.15 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

13.15.1 GPIO Functionality

Figure 13-61 illustrates the GPIO functionality.

Figure 13-61. GPIO Functionality



13.15.2 Under Reset

The following apply if a device is under reset:

- Pull control. The reset pull control on the pins is enabled or disabled depending on a device-specific option. This feature is configurable for each module separately.
- Input buffer. If the reset pull control is enabled, then the input buffer is also enabled. If the reset pull control is disabled then the input buffer is also disabled.
- Output buffer. The output buffer is disabled.

13.15.3 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 13.14.21). In this case, if the PSL (pull select) bit in the SCIPIO8 register (Section 13.14.22) 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 disabled only if the pin direction is set as input in the SCIPIO1 register (Section 13.14.15) AND the pull control is disabledAND pull down is selected as the pull bias (the PSL bit in the SCIPIO8 register; Section 13.14.22). In all other cases, the input buffer is enabled.

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. If the pin is configured as input or receive, the pulls are enabled or disabled depending on bit PD in the pull disable register SCIPIO7 (Section 13.14.21).

- 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 13.14.15) AND the open-drain feature is not enabled in the SCIPIO6 register (Section 13.14.20)

13.15.4 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 13.14.14).

13.15.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 13-48.

Table 13-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins[1]

Device under Reset?	Pin Direction (DIR)[2]	Pull Disable (PULDIS)[3]	Pull Select (PULSEL)[4]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Depends on pull control
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
[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

Chapter 14
SPNU489C-February 2012

Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP)

This reference guide provides the specifications for a 16-bit configurable synchronous multi-buffered multi-pin serial peripheral interface(MibSPI). This chapter also provides the specifications for MibSPI with Parallel Pin Option (MibSPIP). 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/MibSPIP, unless otherwise noted.

Topic **Page**

14.1 Overview

The MibSPI/MibSPIP 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
- 8-bit baud clock generator, supports max up to 20Mhz baud rate
- Serial clock (SPICLK) pin
- 4 SPISOMI/SPISIMO pins for data transfer, with programmable pin direction
- SPI enable ($\overline{\text{SPIENA}}$) pin
- 4 slave chip select ($\overline{\text{SPISCS}}$[3:0]) 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
CS - SPI Chip Select Pin
ENA - SPI Enable Pin.
This device contains two MibSPIs (MibSPI1 and MibSPI3) and one MibSPIP (MibSPIP5). All three modules support 4 SPICS, 1 SPICLK, 1 SPIENA pins. MibSPI 1 & 3 support 1 SPISOMI/SPISIMO pins; MibSPIP5 supports 4 SPISOMI/SPISIMO pins.

14.1.1 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)
- Multi-pin parallel modes

14.1.2 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, etc.) or by the internal tick counter. The internal tick counter supports periodic trigger events. Each buffer of the MibSPI can be associated with different DMA channels in different TGs, allowing the user to move data between internal memory and an external slave with minimal CPU interaction.

14.1.2.1 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, trigger sources for individual groups (14 external trigger sources and 1 internal trigger source supported)
- Memory fault detection via an internal parity circuit.
- Number of DMA-controlled buffers and number of DMA request channels (up to 8 for each of transmit and receive)
- Number of DMA transfers for each buffer (up to 65536 words for up to 8 buffers)
- Uninterrupted DMA buffer transfer (NOBREAK buffer).

14.1.2.2 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.

14.1.3 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.

14.2 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 (SPISCS[3:0]) pins, are used when communicating with multiple slave devices. When the a write occurs to SPIDAT1 in master mode, the SPISCS pins are automatically driven to select the specified slave.

Handshaking mechanism, provided by the 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.

14.2.1 Pin Configurations

The SPI supports data connections as shown in Table 14-1.

Table 14-1. Pin Configurations

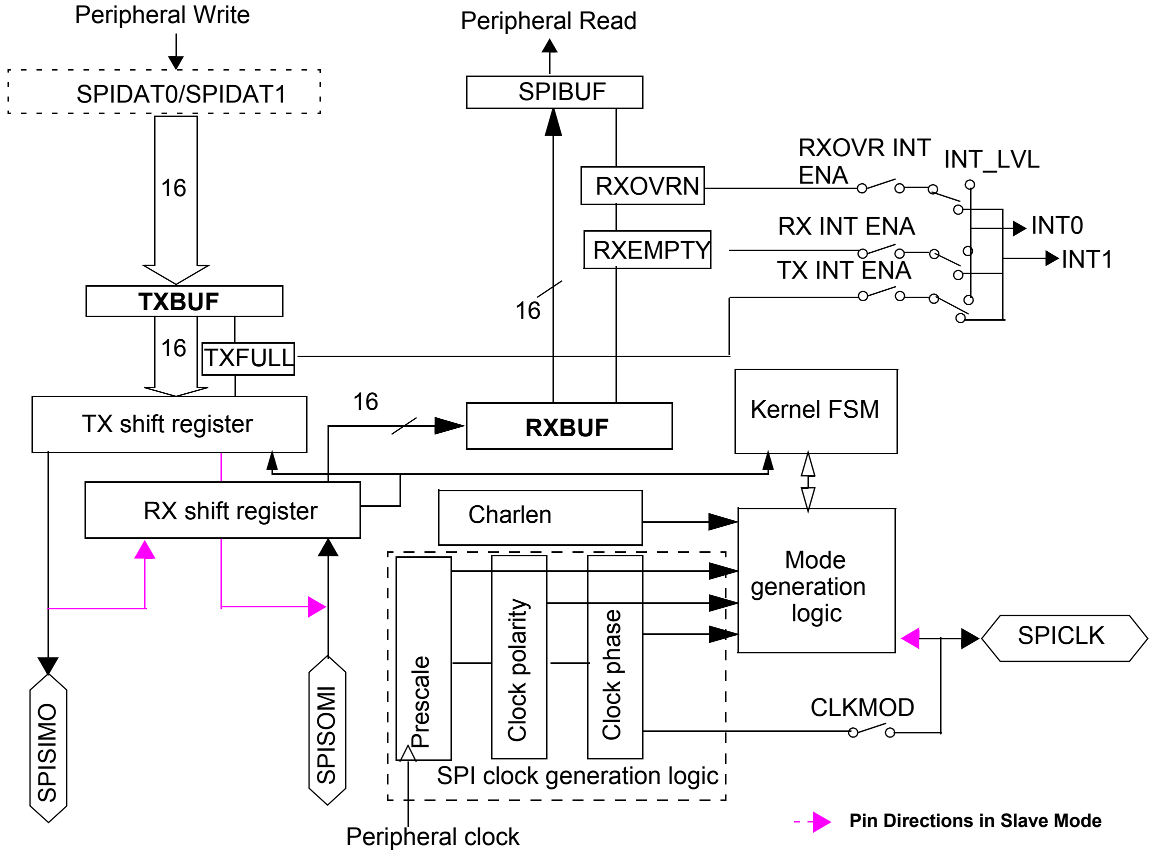
Pin	Master Mode		Slave Mode	
SPICLK	Drives the clock to external devices		Receives the clock from the external master	
SPISOMI	Receives data from the external slave		Sends data to the external master	
SPISIMO	Transmits data to the external slave		Receives data from the external master	
SPIENA	**SPIENA disabled:** GIO	**SPIENA enabled:** Receives ENA signal from the external slave	**SPIENA disabled:** GIO	**SPIENA enabled:** Receives ENA signal from the external master
SPICS[3:0]	**SPICS disabled:** GIO	**SPICS enabled:** Selects one or more slave devices	**SPICS disabled:** GIO	**SPICS enabled:** Receives the CS signal from the external master

Note:
1) When the SPICS[3:0] signals are disabled, the chip-select field in the transmit data is not used.
2) When the SPIENA signal is disabled, the SPIENA pin is ignored in master mode, and not driven as part of the SPI transaction in slave mode.

14.2.2 Data Handling

Figure 14-1 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 14-1. 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.

14.2.2.1 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. For devices with DMA, if DMA is enabled, a transmit DMA request (TX_DMA_REQ) is generated to cause the next word to be fetched. 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 transmit DMA request (if enabled) or a transmitter-empty interrupt (if enabled) is generated at the same time.

14.2.2.2 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 DMA request (if enabled) is generated 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. A receive DMA request is generated, if enabled. The receive complete interrupt line remains high.
- If SPIBUF is read by the CPU or DMA 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.

14.2.2.1 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 14-2).

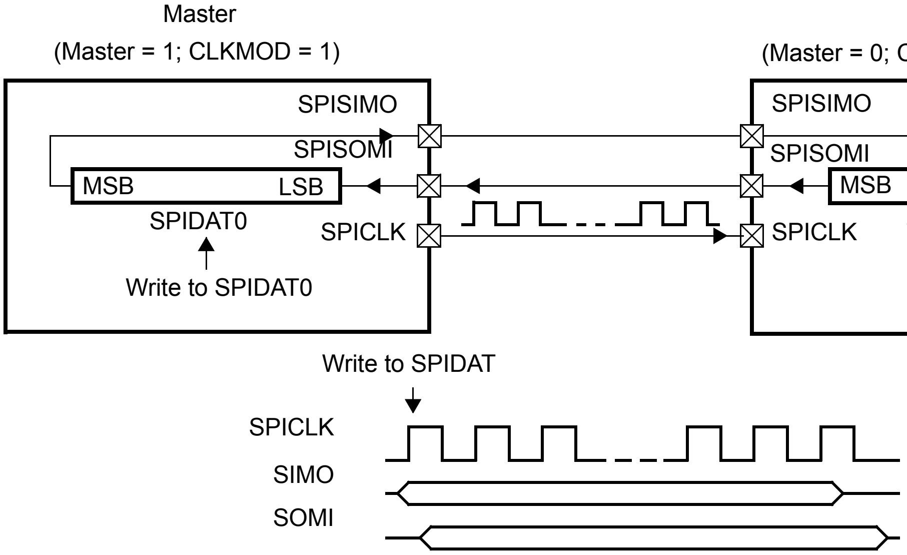
Data written to the shift register (SPIDAT0 / SPIDAT1) initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB of the SPIDAT0 register. When the selected number of bits have been transmitted, the received data in the shift register is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

See Section 14.2.2.1, *Data Sequencing when SPIDAT0 or SPIDAT1 Is Written* on page 583 and Section 14.2.2.2, *Data Sequencing when All Bits Shifted into RXSHIFT Register* on page 583 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 14-2. SPI Three-Pin Operation



14.2.3 Operation with $\overline{\text{SPISCS}}$

In master mode, each chip select signal is used to select a specific slave. In slave mode, chip select signal is used to enable/disable the transfer. Chip-select functionality is enabled by setting one of the $\overline{\text{SPISCS}}$ [3:0] pins as chip selects. It is disabled by setting all $\overline{\text{SPISCS}}$ [3:0] pins as GIOs in SPIPC0[3:0]

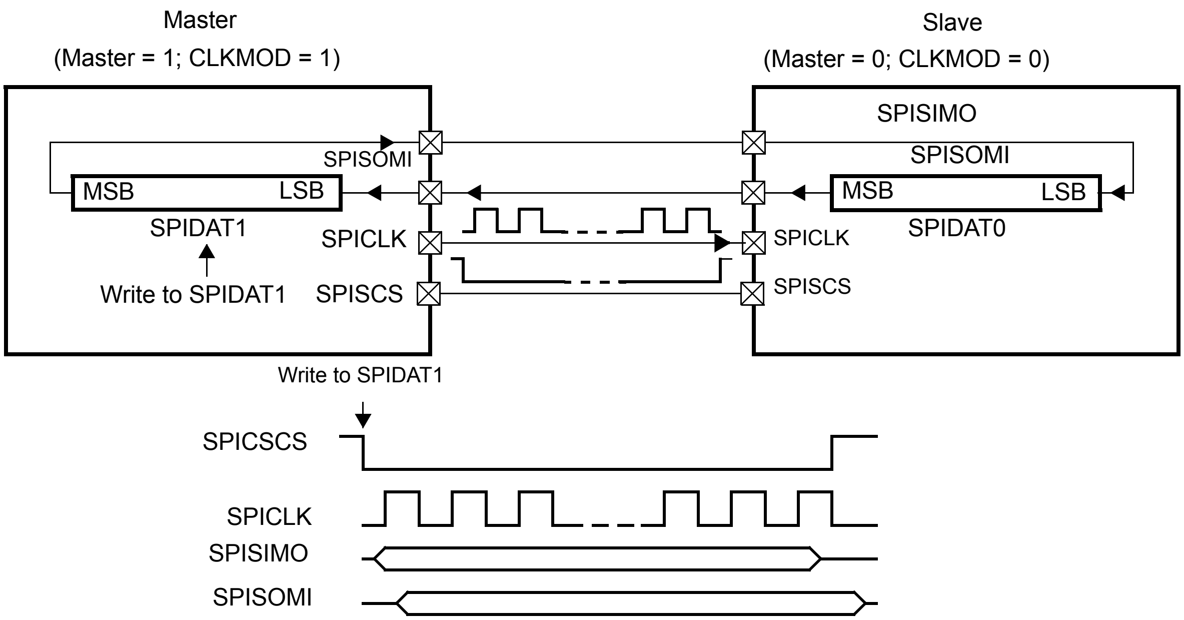
Multiple Chip Selects

The $\overline{\text{SPISCS}}$ [3:0] pins that are used must be configured as functional pins in the SPIPC0[3:0] register. The default pattern to be put on the $\overline{\text{SPISCS}}$ [3:0] 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 SPISCS[3:0] output pin. The drive state for SPISCS[3:0] 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 SPISCS input pin.

Figure 14-3. Operation with SPISCS



14.2.4 Operation with SPIENA

The SPIENA operates as a WAIT signal pin. For both the slave and the master, the SPIENA pin must be configured to be functional (SPIPC0[8] = 1). In this mode, an active low signal from the slave on the 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).

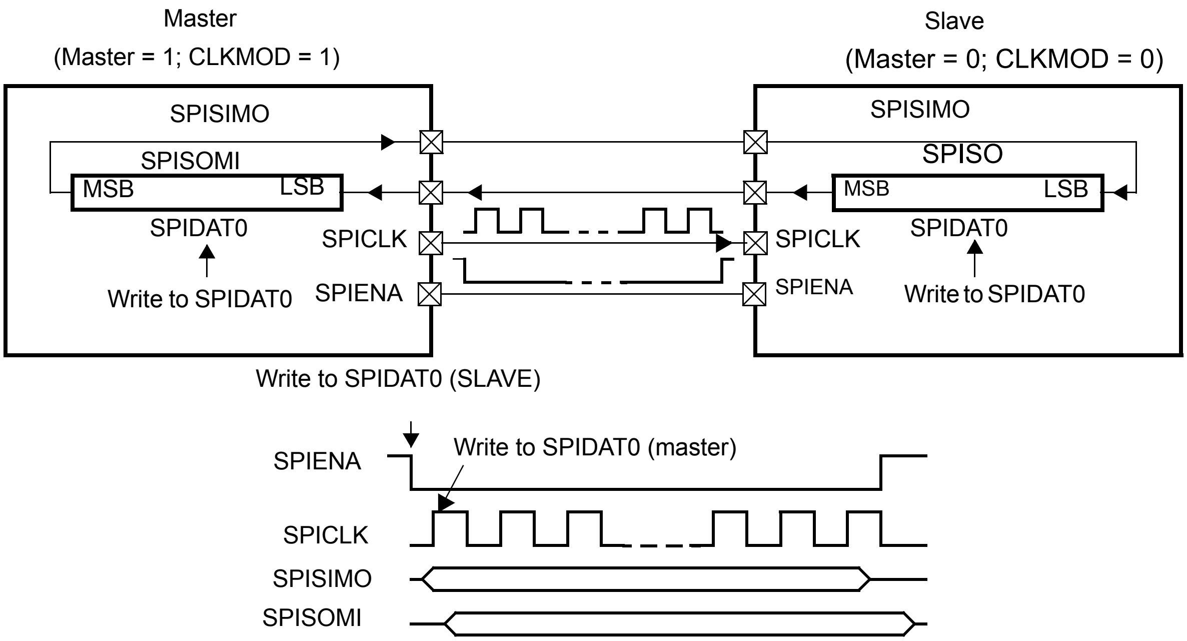
If the SPIENA pin is in high-z mode (ENABLE_HIGHZ = 1), the slave will put SPIENA into the high-impedance once it completes receiving a new character. If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will drive SPIENA to 1 once it completes receiving a new character. The slave will drive 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 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 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 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 14-4. Operation with SPIENA



14.2.5 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 SPISCS [3:0] pins must be configured as functional pins.

If the SPIENA pin is in high-z 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 (SPISCS is low).

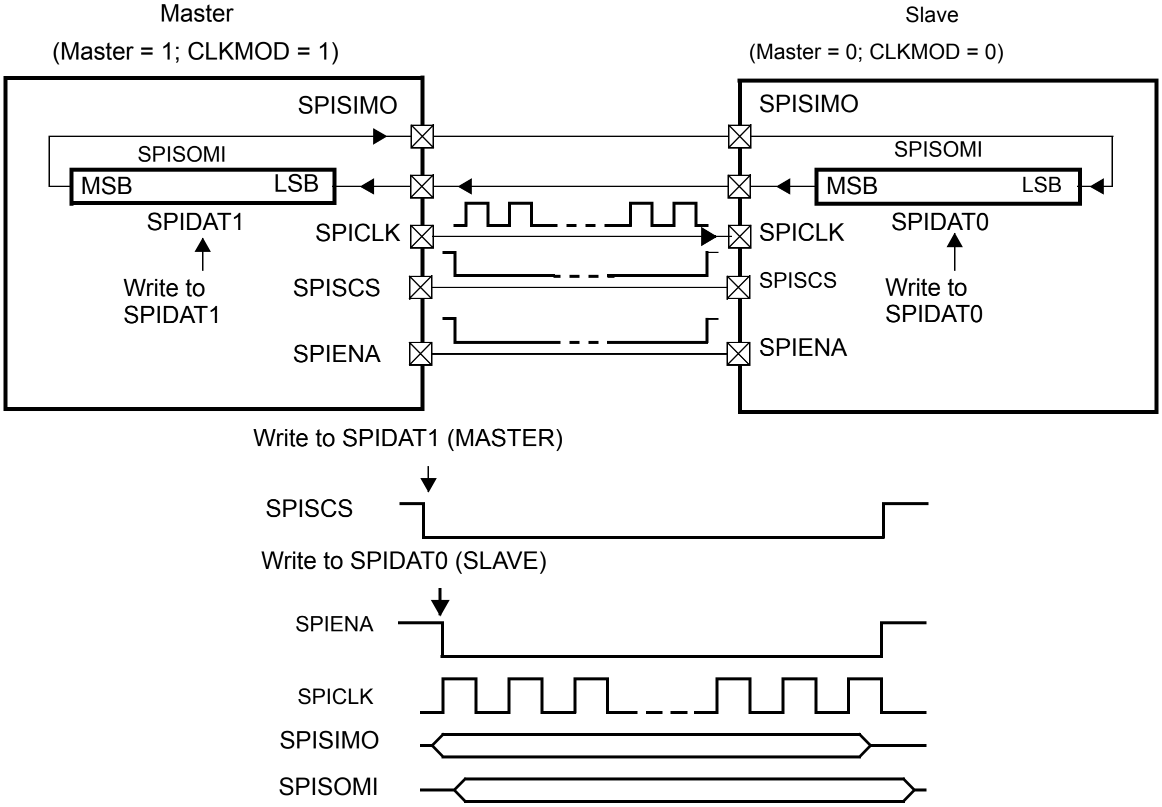
If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave SPI drives this pin high by default when it is in functional mode. The slave SPI will drive the SPIENA signal low when new data is written to the slave shift register (SPIDAT0/SPIDAT1) and the slave is selected by the master (SPISCS is low). If the slave is deselected by the master (SPISCS 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 SPISCS 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 SPISCS signals low. The master will drive the SPISCS signals high again after completing the transfer of the bits of the data.

In slave mode (CLKMOD = 0), the SPISCS pins will act as SPI functional inputs.

Figure 14-5. SPI Five-Pin Option with $\overline{\text{SPIENA}}$ and $\overline{\text{SPISCS}}$



14.2.6 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 14-6 shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer to be transmitted correctly.

Figure 14-6. Format for Transmitting an 8-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	x	x	x	x	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 14-7 shows how a 10-bit word (0x0A2) is stored in the buffer once it is received.

Figure 14-7. Format for Receiving an 8-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

14.2.7 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.

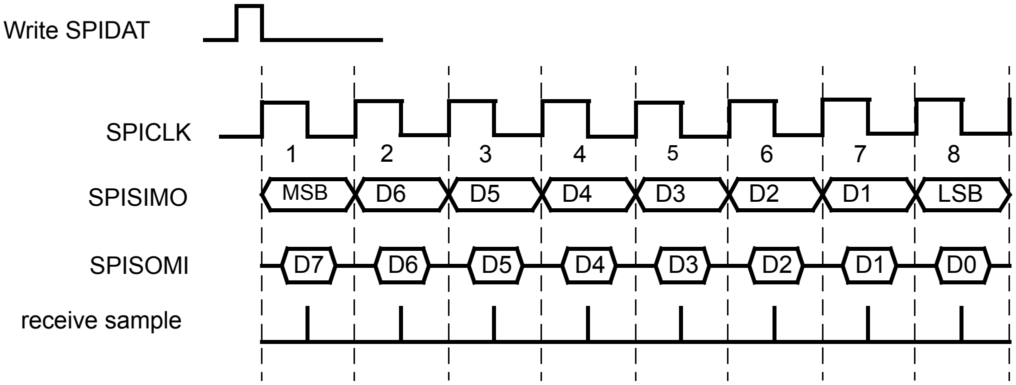
The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 14-2.

Table 14-2. 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.

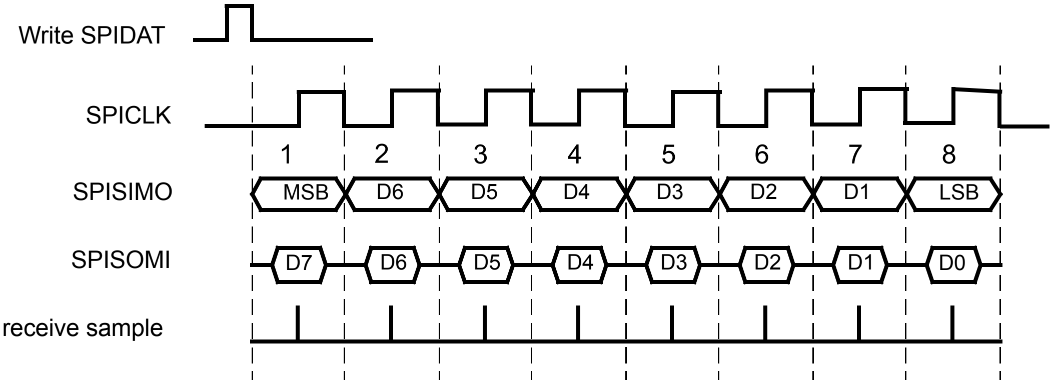
Figure 14-8 to Figure 14-11 illustrate the four possible configurations of SPICLK corresponding to each mode. Having four signal options allows the SPI to interface with many different types of serial devices.

Figure 14-8. Clock Mode with Polarity = 0 and Phase = 0



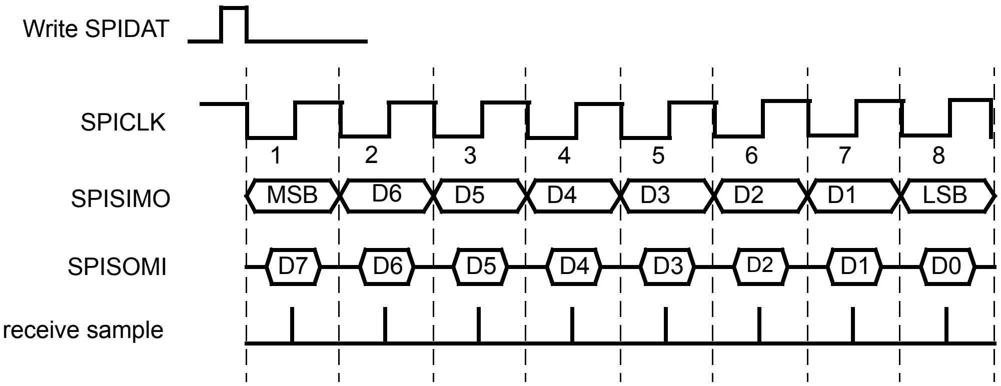
Data is output on the rising edge of SPICLK.
Input data is latched on the falling edge of SPICLK.

Figure 14-9. Clock Mode with Polarity = 0 and Phase = 1



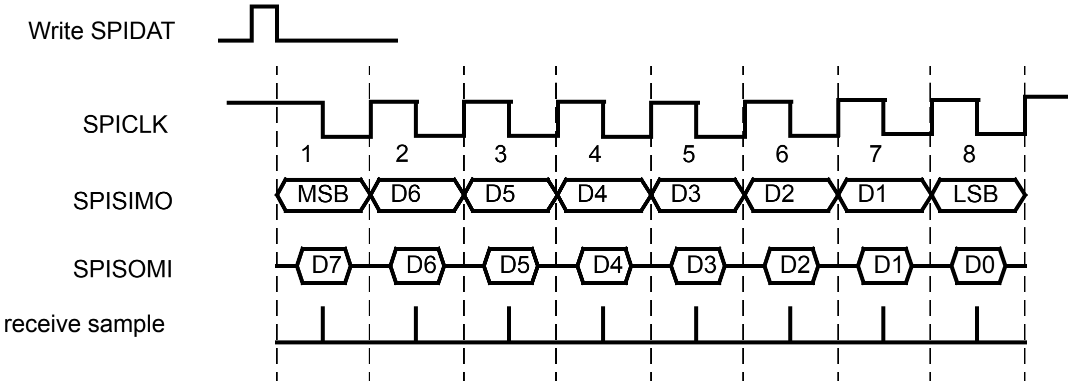
Data is output one-half cycle before the first rising edge of SPICLK and on subsequent falling edges of SPICLK
Input data is latched on the rising edge of SPICLK

Figure 14-10. 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 14-11. Clock Mode with Polarity = 1 and Phase = 1

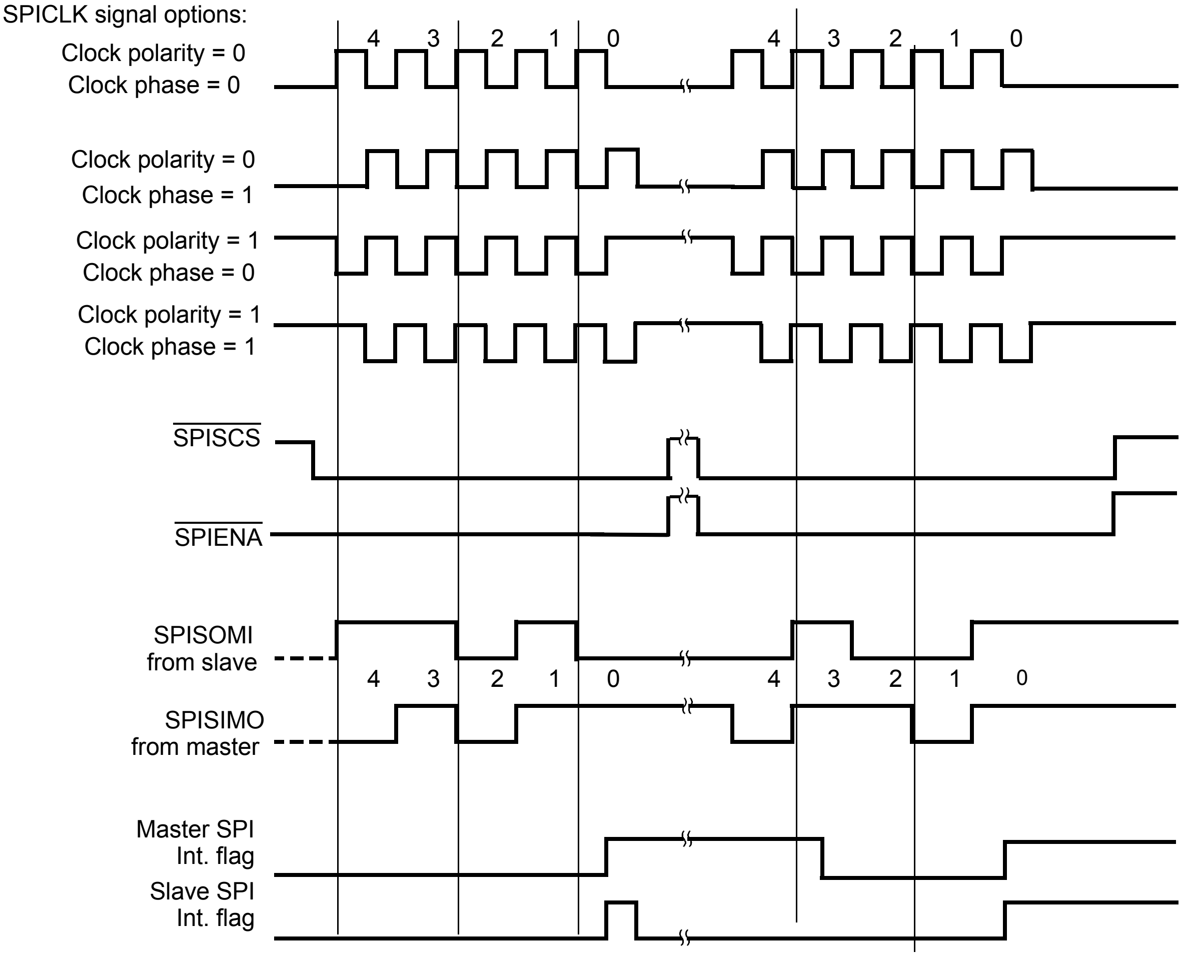


Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK.
Input data is latched on the falling edge of SPICLK.

14.2.8 Data Transfer Example

Figure 14-12 illustrates a SPI data transfer between two devices using a character length of five bits.

Figure 14-12. Five Bits per Character (5-Pin Option)



14.2.9 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, i.e., not active, can be configured via the register CSDEF. During a transmission, the value of the chip select control field (CSNR[7:0]) of the SPIDAT1 register (SPIDAT1[23:16]) is driven on the $\overline{\text{SPISCS}}$ [4:0] pins. When the transmission finishes the default chip-select value (defined by the CSDEF register) is put on the $\overline{\text{SPISCS}}$ [4:0] 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 $\overline{\text{SPISCS}}$ pins at the same time, which enables binary encoded chip selects from 0 to 16. To use encoded chip selects, all four 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* $\overline{\text{SPISCS}}$ pins can be used for encoding a *n*-bit address and the remaining pins can be connected to decoded-mode slaves.

14.2.10 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.

14.2.11 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.

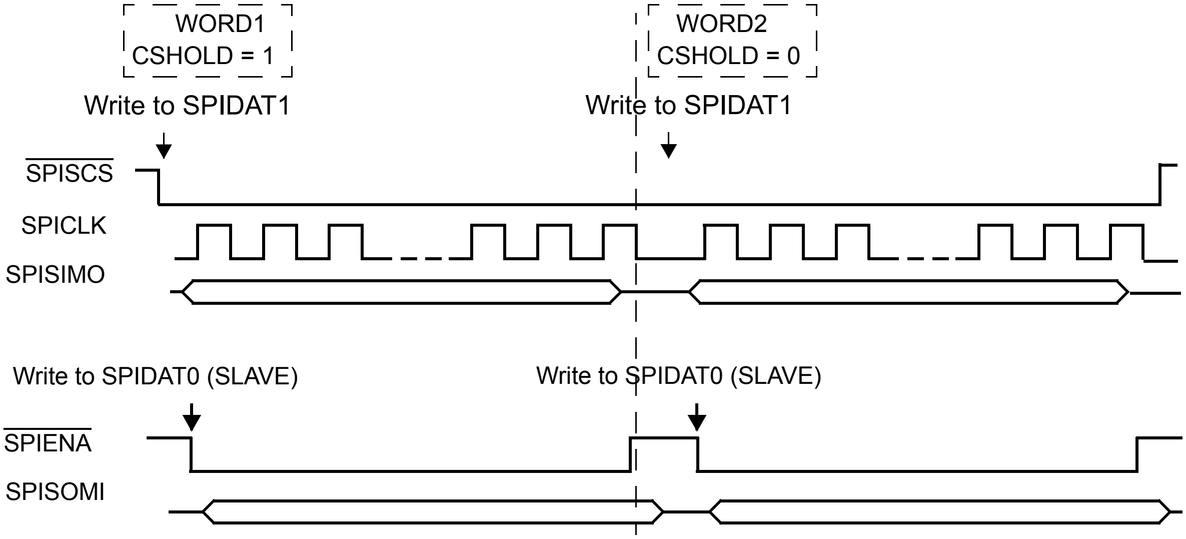
14.2.11.1 The CSHOLD Bit in Master Mode

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 14-13 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 14-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI)



14.2.12 Detection of Slave Desynchronization (Master Only)

When a slave supports generation of an enable signal (ENA), de-synchronization can be detected. With the enable signal a slave indicates to the master that it is ready to exchange data. A de-synchronization 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 de-synchronized slave and the consecutive data word. A configurable 8 bit time-out counter, 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 Desync 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 guaranteed to be for the current buffer.

14.2.13 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. 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.

14.2.14 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 (e.g. 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 $\overline{\text{SPISCS}}$ 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 $\overline{\text{SPISCS}}$(x) pins are configured as functional, and are being used for selecting the slave.

14.2.15 Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices)

In order to increase throughput, the parallel mode of the SPI enables the module to send data over more than one data line (Parallel 2, 4 or 8). When parallel mode is used, the data length must be set as 16 bits. Only module MIBSPIP5 supports Parallel Mode.

This feature increases throughput by 2 for 2 pins, by 4 for 4 pins, or by 8 for 8 pins.

Parallel mode supports the following features:

- Scalable data lines (1, 2, 4, 8) per direction. (SOMI and SIMO lines)
- All clock schemes are supported (clock phase and polarity)
- Parity is supported. The parity bit will be transmitted on bit0 of the SIMO/SOMI lines. The receive parity is expected on bit0 of the SOMI/SIMO pins.

Parallel mode can be programmed using the PMODEx[1:0] bits of SPIPMCTRL register. See Figure 14-28. for details about this register.

After reset the parallel mode selection bits are cleared (single SIMO/SOMI lines).

14.2.15.2 Parallel Mode Block Diagram

Figure 14-14 and Figure 14-15 show the parallel connections to the SPI shift register.

Figure 14-14. Block Diagram Shift Register, MSB First

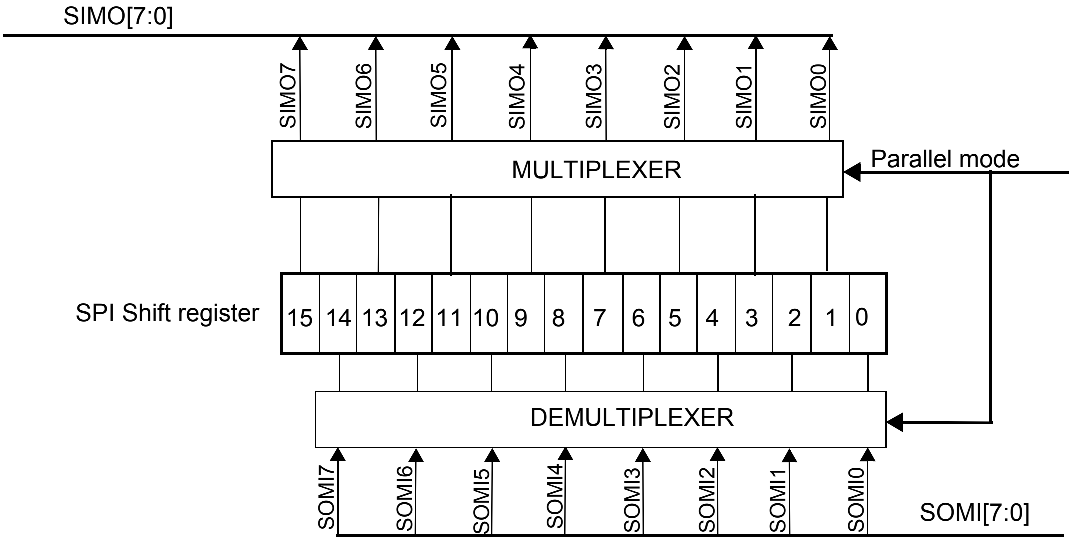
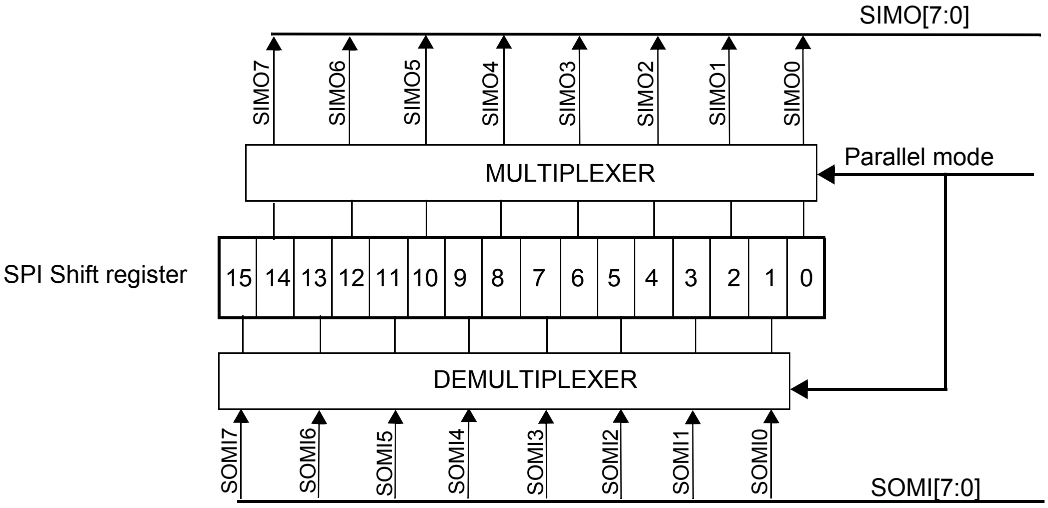


Figure 14-15. Block Diagram Shift Register, LSB First



14.2.15.3 Parallel Mode Pin Mapping, MSB First

Table 14-3 and Table 14-4 describe the SOMI and SIMO pin mapping when the SPI is used in parallel mode (1, 2, 4, 8) pin mode, MSB first.

Note:
MSB-first or LSB-first can be configured using the SHIFTDIRx bit of the SPIFMTx registers.

Table 14-3. Pin Mapping for SIMO Pin with MSB First

Parallel Mode	Shift Register Bit	SIMO[7:0]
1	15	0
2	15	1
	7	0
4	15	3
	11	2
	7	1
	3	0
8	15	7
	13	6
	11	5
	9	4
	7	3
	5	2
	3	1
	1	0

Table 14-4. Pin Mapping for SOMI Pin with MSB First

Parallel Mode	Shift Register Bit	SOMI[7:0]
1	0	0
2	0	0
	8	1
4	0	0
	4	1
	8	2
	12	3
8	0	0
	2	1
	4	2
	6	3
	8	4
	10	5
	12	6
	14	7

14.2.15.4 Parallel Mode Pin Mapping, MSB-First, LSB-First

Table 14-5–Table 14-6 describe the SIMO and SOMI pin mapping when SPI is used in parallel mode (1, 2, 4, 8) pin mode, LSB first.

Table 14-5. Pin Mapping for SIMO Pin with LSB First

Parallel Mode	Shift Register Bit	SIMO[7:0]
1	0	0
2	8	1
	0	0
4	12	3
	8	2
	4	1
	0	0
8	14	7
	12	6
	10	5
	8	4
	6	3
	4	2
	2	1
	0	0

Table 14-6. Pin Mapping for SOMI Pin with LSB First

Parallel Mode	Shift Register Bit	SOMI[7:0]
1	15	0
2	7	0
	15	1
4	3	0
	7	1
	11	2
	15	3
8	1	0
	3	1
	5	2
	7	3
	9	4
	11	5
	13	6
	15	7

14.2.15.5 2-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 2-data line mode (master mode) the shift register bits 15 and 7 will be connected to the pins SIMO[1] and SIMO[0], and the shift register bits 8 and 0 will be connected to the pins SOMI[1] and SOMI[0] or vice versa in slave mode. After writing to the SPIDAT0/SPIDAT1 register, the bits 15 and 7 will be output on SIMO[1] and SIMO[0] on the rising edge if SPICLK. With the falling clock edge of the SPICLK, the received data on SOMI[1] and SOMI[0] will be latched to the shift register bits 8 and 0. The subsequent rising edge of SPICLK will shift the data in the shift register by 1 bit to the left. (SIMO[1] will shift the data out from bit 15 to 8, SIMO[0] will shift the data out from bit 7 to 0). After eight SPICLK cycles, when the full data word is transferred, the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set. Figure 14-16 shows the clock /data diagram of the 2-data line mode. Figure 14-17 shows the timing of a two-pin parallel transfer.

Figure 14-16. 2-data Line Mode (Phase 0, Polarity 0)

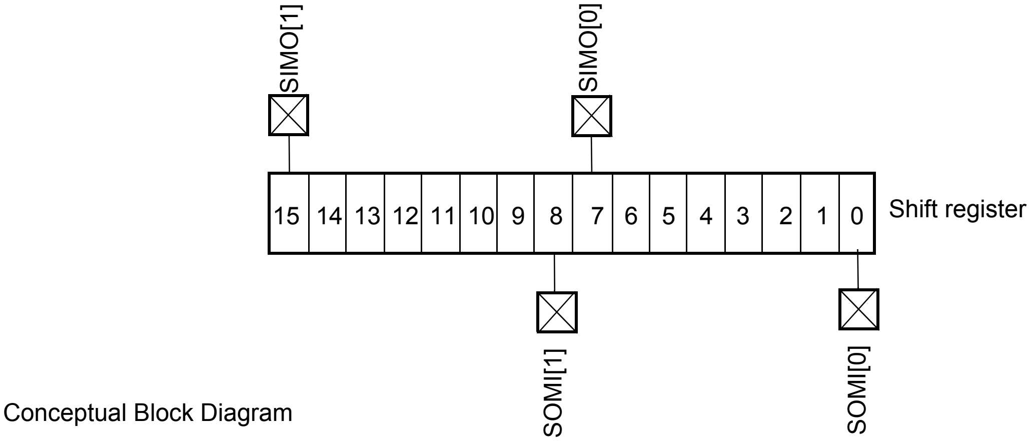
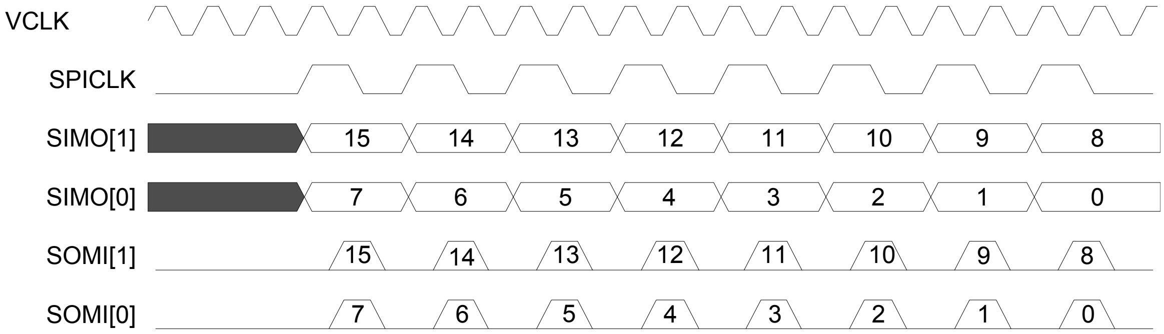


Figure 14-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0)



14.2.15.6 4-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 4-data line mode (master mode) the shift register bits 15, 11, 7, and 3 will be connected to the pins SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift register bits 12, 8, 4, and 0 will be connected to the pins SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode). After writing to SPIDAT1/SPIDAT0, the bits 15, 11, 7, and 3 will be output on SIMO[3], SIMO[2], SIMO[1], and SIMO[0] on the rising edge of SPICLK. With the falling clock edge of the SPICLK, the received data on SOMI[3], SOMI[2], SOMI[1] and SOMI[0] will be latched to shift register bits 12, 8, 4, and 0. The subsequent rising edge of SPICLK will shift data in the shift register by 1 bit to the left (SIMO[3] will shift the data out from bit 15 to 12, SIMO[2] will shift the data out from bit 11 to 8, SIMO[1] will shift the data out from bit 7 to 4, SIMO[0] will shift the data out from bit 3 to 0). After four SPICLK cycles, when the full data word is transferred, the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set.

Figure 14-19 shows the clock/data diagram of the four-data line mode. Figure 14-20, shows the timing diagram for four-data line mode.

Figure 14-18. 4-Data Line Mode (Phase 0, Polarity 0)

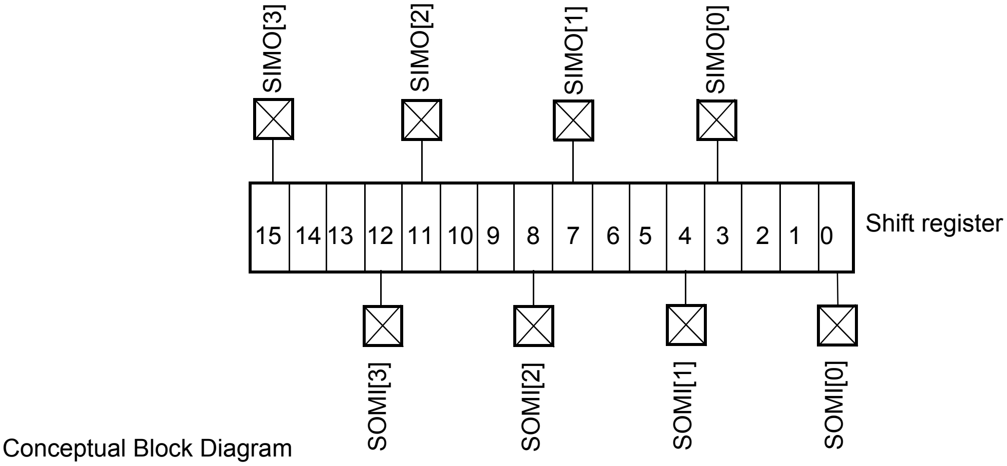


Figure 14-19. 4-Data Line Mode (Phase 0, Polarity 0)

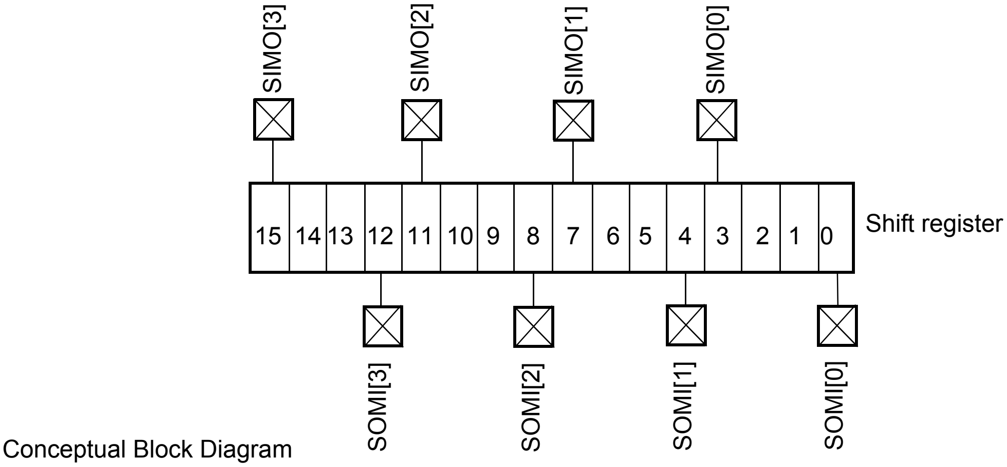
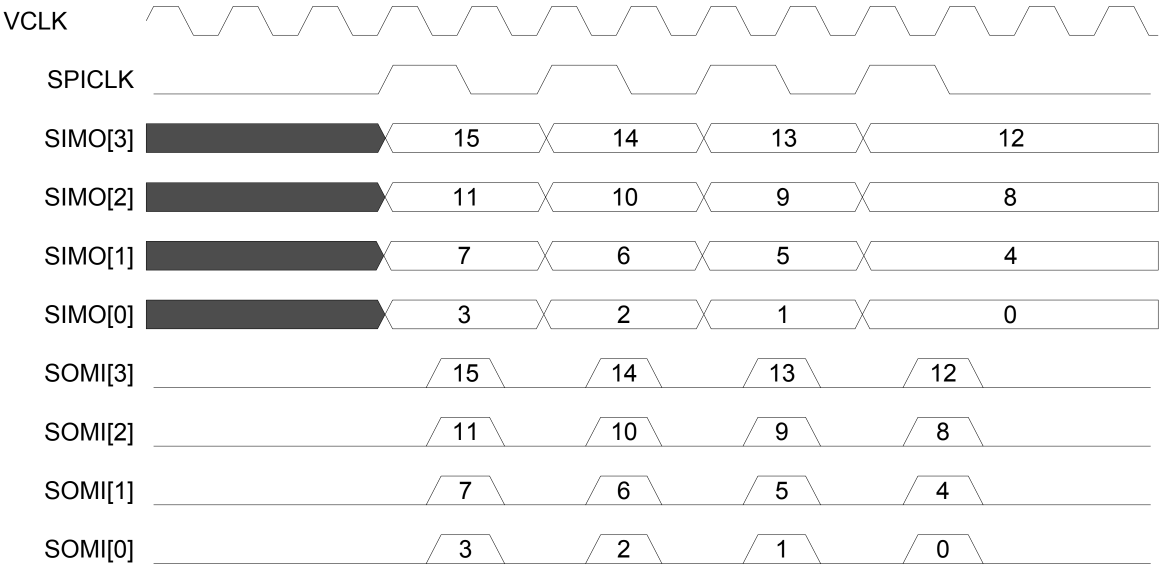


Figure 14-20. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



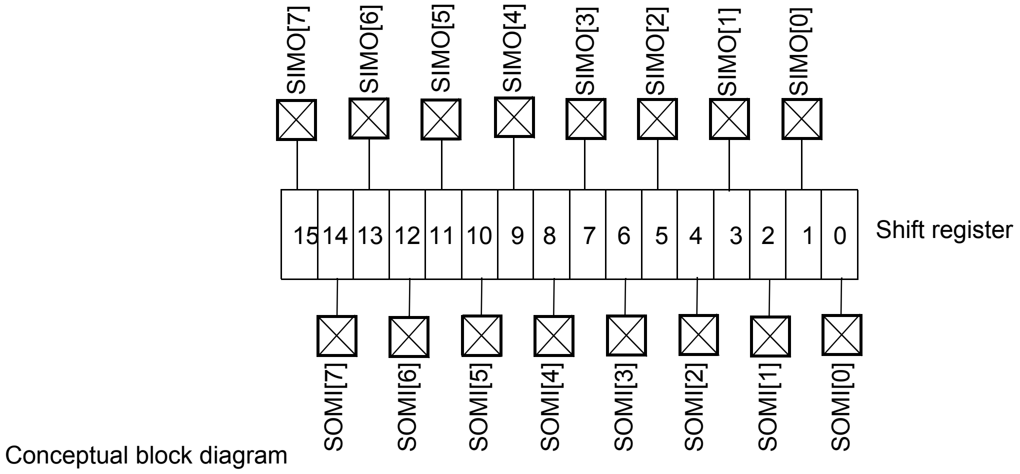
14.2.15.7 8-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 8-data line mode (master mode) the shift register bits 15, 13, 11, 9, 7, 5 and 3 will be connected to the pins SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift-register bits 14, 12, 10, 8, 6, 4, and 0 will be connected to the pins SOMI[7], SOMI[6], SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode).

After writing to SPIDAT0/SPIDAT1, the bits 15, 13, 11, 9, 7, 5, 3, and 1 will be output on SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], on the rising edge of SPICLK. On the falling clock edge of the SPICLK, the received data on SOMI[8], SOMI[7], SOMI[6],SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] will be latched to the shift register bits 14, 12, 10, 8, 6, 4, 2, and 0.

The subsequent rising edge of SPICLK will shift the data in the shift register by one bit to the left. After two SPICLK cycles, when the full data word is transferred the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set. Figure 14-21 shows the clock/data diagram of the 8-data line mode. Figure 14-22 shows the pin timings for 8-data line mode.

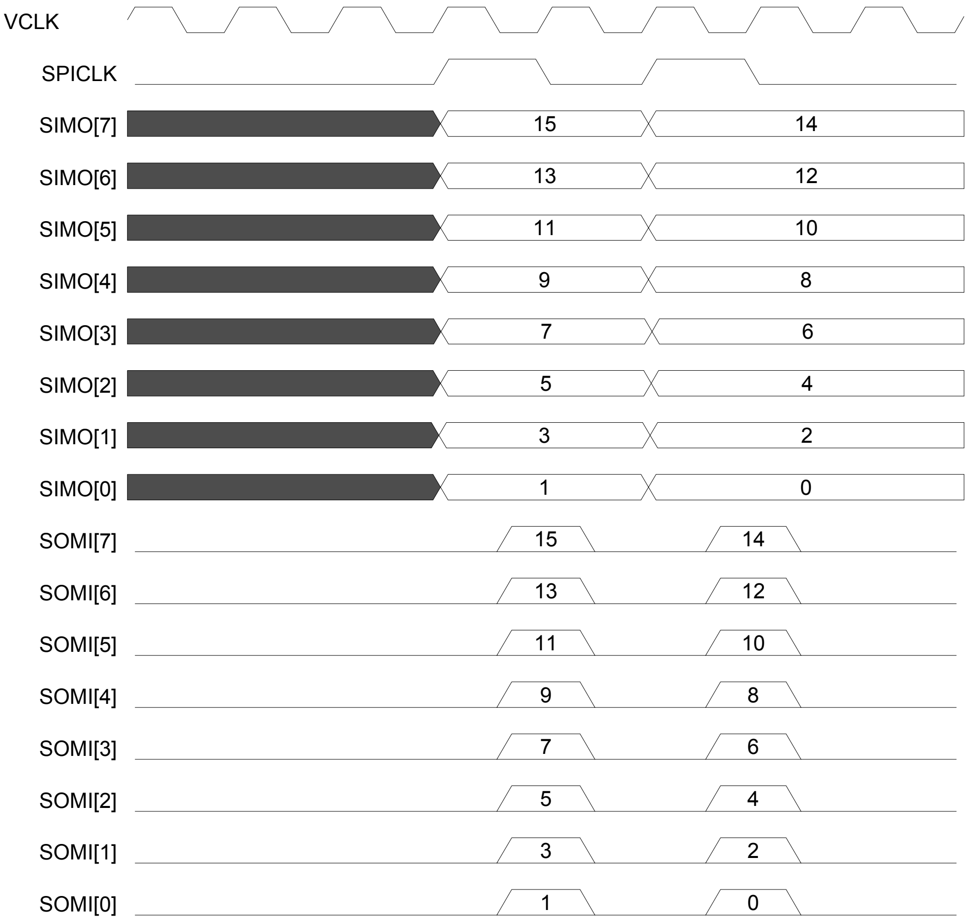
Figure 14-21. eight-data Line Mode (Phase 0, Polarity 0)



Note: Parity Support

Using the parity support in parallel mode may seriously affect throughput. For an eight-line mode to transfer 16 bits of data, only two SPICLK pulses are enough. If parity is enabled, one extra SPICLK pulse will be used to transfer and receive the parity bit. Parity will be transmitted and received on the 0th line regardless of 1/2/4/8-line modes. During the parity bit transfer, other data bits are not valid.

Figure 14-22. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



Note: Modulo Count Parallel Mode is not supported in this device.

14.2.16 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.

14.3 Test Features

14.3.1 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 fedback to the receiver, whereas the SIMO, SOMI, and CLK pin are disconnected, i.e., 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.

14.3.2 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 14-23 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 14.9.42, *I/O-Loopback Test Control Register (IOLPBKTSTCR)* on page 700 for details of the register field controlling these features.

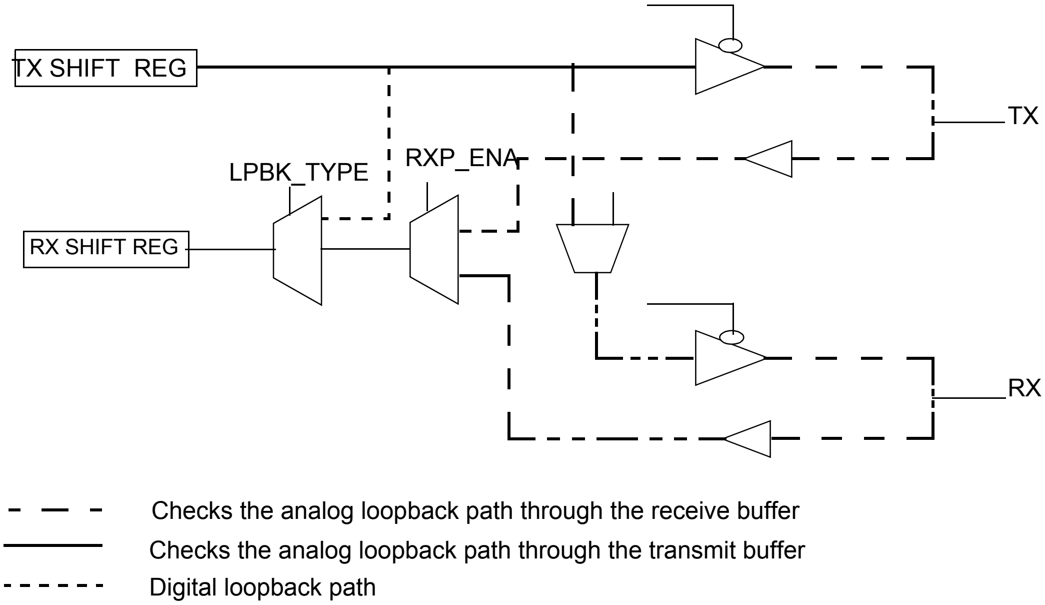
In Input/Output loopback test modes, even when the module is in slave mode, the SPICLK is generated internally. This SPICLK is used for all loopback-mode SPI transactions. Slave-mode features can be tested without the help of another master SPI, using the internally-generated SPICLK. Chip selects are 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 14-23. I/O Paths during I/O Loopback Modes


TX SHIFT REG
RX SHIFT REG
LPBK_TYPE
RXP_ENA
TX
RX
Checks the analog loopback path through the receive buffer
Checks the analog loopback path through the transmit buffer
Digital loopback path

1 This diagram is intended to illustrate loopback paths and therefore may omit some normal-mode paths.

14.3.2.1 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 SPISCS[3:0] to 0x0. The actual number of chip selects can be programmed to have any or all of the SPISCS 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[3:0] 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 0x5 to the IOLPBTSTENA bits or all of the TGs can be disabled.

14.4 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
- input-buffer enabling/disabling (controlled by the PULDIS and PSEL bits)

14.5 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.

14.6 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.

14.6.1 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:

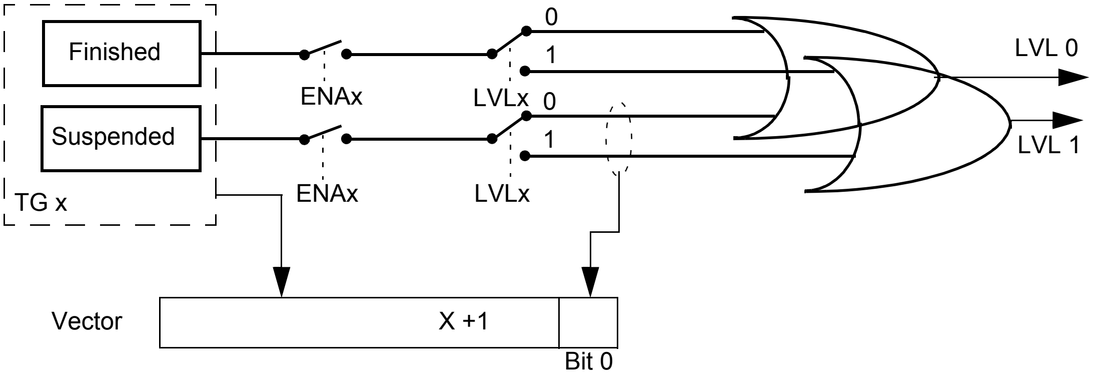
- Transmission error interrupt
- Receive overrun interrupt
- TG suspended interrupt
- TG completed interrupt

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 14-24 illustrates the TG interrupts.

Figure 14-24. TG Interrupt Structure

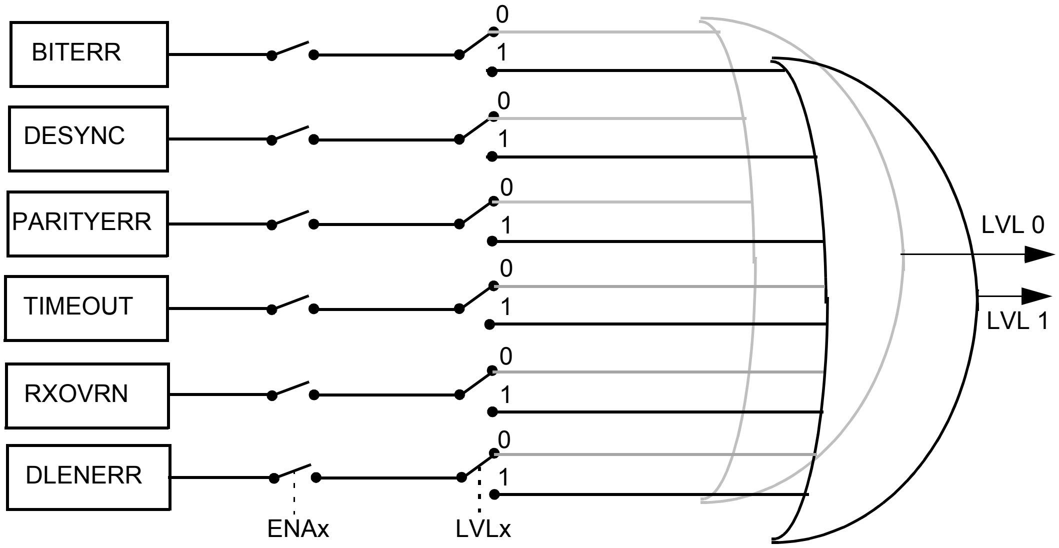


During transmission, if one of the following errors occurs, BITERR, DESYNC, PARITYERR, TIMEOUT, DLENERR, the corresponding flag in the SPIFLG register is set. If the enable bit is set, then an interrupt is generated. The level of the interrupts could be generated according to the bit field in SPILVL 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 14-25. 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.

14.7 DMA Interface

In order to reduce CPU overhead in handling SPI message traffic on a character-by-character basis, SPI can use the DMA controller to transfer the data. The DMA request enable bit (DMA REQ EN) controls the assertion of requests to the DMA controller module. When a character is being transmitted or received, the SPI will signal the DMA via the DMA request signals, TX_DMA_REQ and RX_DMA_REQ. The DMA controller will then perform the required data transfer.

For efficient behavior during DMA operations, the transmitter empty and receive-buffer full interrupts can be disabled. For specific DMA features, see the DMA controller specification.

The SPI generates a request on the TX_DMA_REQ line each time the TX data is copied to the TX shift register either from the TXBUF or from peripheral data bus (when TXBUF is empty).

The first TX_DMA_REQ pulse is generated when either of the following is true:

- DMA REQ EN (SPIINT0[16]) is set to 1 while SPIEN (SPIGCR1[24]) is already 1.
- SPIEN (SPIGCR1[24]) is set to 1 while DMA REQ EN (SPIINT0[16]) is already 1.

The SPI generates a request on the RX_DMA_REQ line each time the received data is copied to the SPIBUF.

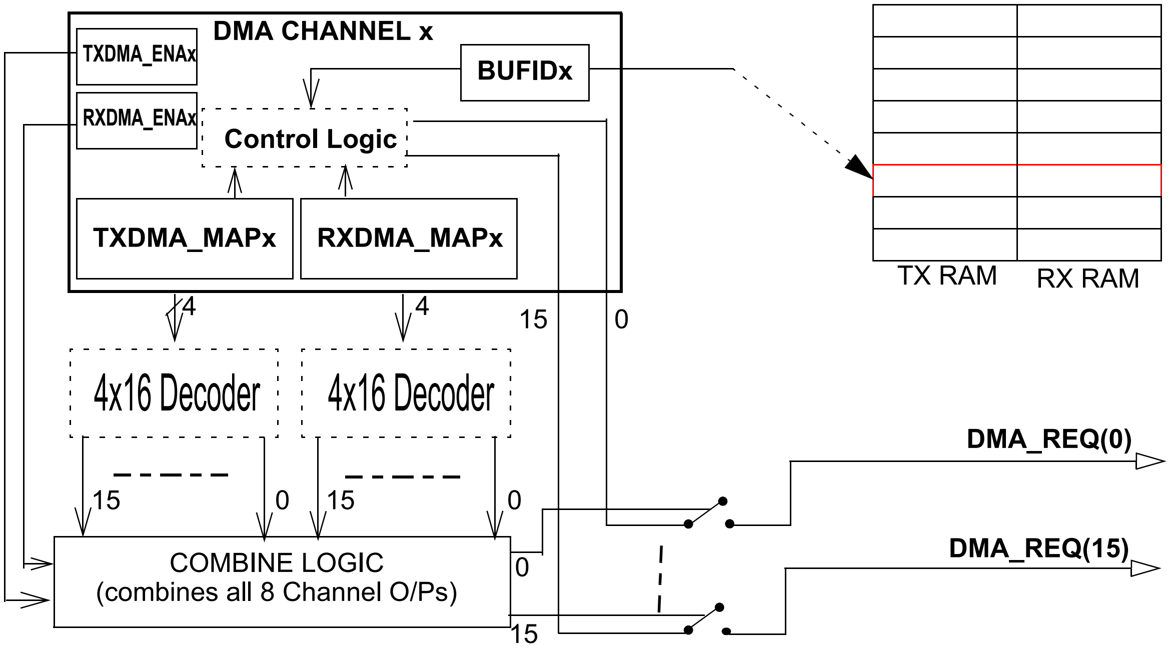
14.7.1 DMA in Multi-Buffer Mode

The MibSPI provides sophisticated programmable DMA control logic that completely eliminates the necessity of CPU intervention for data transfers, once programmed. When the multi-buffer mode is used, the DMA enable bit in the SPIINT0 register is ignored. DMA source or destination should be only the multi-buffer RAM and not SPIDAT0 / SPIDAT1 or SPIBUF register as in case of compatibility mode DMA.

The MibSPI offers up to eight DMA channels (for SEND and RECEIVE). All of the DMA channels are programmable individually and can be hooked to any buffer. The MibSPI provides up to 16 DMA request lines, and DMA requests from any channel can be programmed to be routed through any of these 16 lines. A DMA transfer can trigger both transmit and receive.

Each DMA channel has the capability to transfer a block of up to 32 data words without interruption using only one buffer of the array by configuring the DMAxCTRL register. Using the DMAxCOUNT and DMACTNTLEN register, up to 65535 (64K) words of data can be transferred without any interruption using just one buffer of the array. This enables the transfer of memory blocks from or into an external SPI memory.

Figure 15. DMA Channel and Request Line (logical) structure in Multibuffer Mode



14.8 Module Configuration

MibSPI/MibSPIP can be configured to function as Normal SPI and Multibuffered 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.

14.8.1 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 SPIENA is held low 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, CLK and optional CSx, ENA 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 CS 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.

– User must wait for TXFULL to reset or TXINT before writting next data to SPIDAT1 register.

– User must wait for RXEMPTY to reset or RXINT before reading the data from SPIBUF register.

14.8.2 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 SPIENA is held low 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, CLK and optional CSx, ENA 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.

– User can poll Transfer-group interrupt flag or wait for a transfer-completed interrupt to read and write new data to the buffers.

14.9 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 can be found in the device data sheet.

Table 14-7. SPI Registers

Offset Address[1] 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
0x00h SPIGCR0 page 617	Reserved															
	Reserved															nRESET
0x04h SPIGCR1 page 618	Reserved							SPIEN	Reserved							LOOP BACK
	Reserved							POWERDOWN	Reserved						CLK MOD	MASTER
0x08h SPIINT0 page 620	Reserved							ENABLE HIGHZ	Reserved							DMA REQ EN
	Reserved						TXINT ENA	RXINT ENA	Reserved	OVRN INT ENA	Reserved	BIT ERR ENA	DESYNC ENA	PAR ERR ENA	TIME OUT ENA	DLEN ERR ENA
0x0Ch SPILVL page 623	Reserved															
	Reserved						TX INT LVL	RX INT LVL	Reserved	OVRN INT LVL	Reserved	BITERR LVL	DESYNC LVL	PAR ERR LVL	TIME OUT LVL	DLEN ERR LVL
0x10h SPIFLG page 625	Reserved							BUF INIT ACTIVE	Reserved							
	Reserved						TX INT FLG	RX INT FLG	Reserved	OVRN INT FLG	Reserved	BIT ERR FLG	DESYNC FLG	PAR ERR FLG	TIME OUT FLG	DLEN ERR FLG
0x14h SPIPC0 page 630	SOMIFUN[7:0]								SIMOFUN[7:0]							
	Reserved				SOMI FUN	SIMO FUN	CLK FUN	ENA FUN	SCSFUN[7:0]							
0x18h SPIPC1 page 632	SOMIDIR[7:0]								SIMODIR[7:0]							
	Reserved				SOMI DIR	SIMO DIR	CLK DIR	ENA DIR	SCSDIR[7:0]							

[1] The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x1Ch SPIPC2 page 634	SOMIDIN[7:0]								SIMODIN[7:0]							
	Reserved				SOMI DIN	SIMO DIN	CLK DIN	ENA DIN	SCSDIN[7:0]							
0x20h SPIPC3 page 636	SOMIDOUT[7:0]								SIMODOUT[7:0]							
	Reserved				SOMI DOUT	SIMO DOUT	CLK DOUT	ENA DOUT	SCSDOUT[7:0]							
0x24h SPIPC4 page 638	SOMISET[7:0]								SIMOSET[7:0]							
	Reserved				SOMI SET	SIMO SET	CLK SET	ENA-SET	SCSSET[7:0]							
0x28h SPIPC5 page 640	SOMICLR[7:0]								SIMOCLR[7:0]							
	Reserved				SOMI CLR	SIMO CLR	CLK CLR	ENA CLR	SCSCLR[7:0]							
0x2Ch SPIPC6 page 642	SOMIPDR[7:0]								SOMICLR[7:0]							
	Reserved				SOMI PDR	SIMO PDR	CLK PDR	ENA PDR	SCSPDR[7:0]							
0x30h SPIPC7 page 644	SOMIDIS[7:0]								SIMODIS[7:0]							
	Reserved				SOMI PDIS	SIMO PDIS	CLK PDIS	ENA PDIS	SCSPDIS[7:0]							
0x34h SPIPC8 page 646	SOMIPSEL[7:0]								SIMOPSEL[7:0]							
	Reserved				SOMI PSL	SIMO PSL	CLK PSL	ENA PSL	SCSPSL[7:0]							
0x38h SPIDAT0 page 648	Reserved															
	TXDATA(15–0)															

[1] The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x3Ch SPIDAT1 page 649	Reserved			CS HOLD	Reserved	WDEL	DFSEL		CSNR(7–0)							
	TXDATA(15–0)															
0x40h SPIBUF page 651	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN ERR	LCSNR(7–0)							
	RXDATA(15–0)															
0x44h SPIEMU page 655	Reserved															
	RXDATA(15–0)															
0x48h SPIDELAY page 656	C2TDELAY(7–0)								T2CDELAY(7–0)							
	T2EDELAY(7–0)								C2EDELAY(7–0)							
0x4Ch SPIDEF page 660	Reserved															
	Reserved								CSDEF[7:0]							
0x50h–5Ch SPIFMT[0:3] page 661	Reserved		WDELAY[0:3](5–0)						PAR POL [0:3]	PAR-ITY [0:3] ENA	WAIT ENA [0:3]	SHIFT DIR[0:3]	Reserved	DIS CS TIM-ERS	POLAR ITY [0:3]	PHASE [0:3]
	PRESCALE[0:3]								Reserved			CHARLEN[0:3]				
0x60h–64h TGINTVECT[0:1] page 664	Reserved															
	Reserved										INTVECT[0:1]					SUS PEND [0:1]
0x6Ch SPIPMCTRL page 668	Reserved		MOD CLK POL3	MMODE 3(2–0)			PMODE 3(1–0)		Reserved		MOD CLK POL2	MMODE 2(2–0)			PMODE 2(1–0)	
	Reserved		MOD CLK POL1	MMODE 1(2–0)			PMODE 1(1–0)		Reserved		MOD CLK POL0	MMODE0(2–0)			PMODE 0(1–0)	

[1] The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x70h MIBSPIE page 672	Reserved															RX RAM ACCESS
	Reserved															MibSPI ENA
0x74h TGITENST page 674	SET INTEN RDY[15:0]															
	SET INTEN SUS[15:0]															
0x78h TGITENCR page 675	CLR INTEN RDY[15:0]															
	CLR INTEN SUS[15:0]															
0x7Ch TGITLVST page 676	SET INTLVL RDY[15:0]															
	SET INTLVL SUS[15:0]															
0x80h TGITLVCR page 677	CLR INTLVL RDY[15:0]															
	CLR INTLVL SUS[15:0]															
0x84h TGITFLG page 678	INTFLG RDY[15:0]															
	INTFLG SUS[15:0]															
0x88–0x8Ch	Reserved															
	Reserved															
0x90h TICKCNT page 680	TICK ENA	RE LOAD	CLKCTRL(1–0)		Reserved											
	TICKVALUE(15–0)															

[1] The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x94h LTGPEND page 682	Reserved			TG IN SERVICE(4–0)					Reserved							
	Reserved	LPEND(6–0)							Reserved							
0x98–D4h TG[0:15]CTRL page 683	TG ENA[0:15]	ONE SHOT [0:15]	PRST [0:15]	TGTD[0:15]	Reserved				TRGEVT[0:15](4–0)				TRIGSRC[0:15](4–0)			
	Reserved	PSTART[0:15](6–0)							Reserved	PCURRENT[0:15](6–0)						
0xD8h–F4h DMA[0:8]CTRL page 689	ONE SHOT [0:8]	BUFID[0:8](6–0)							RXDMA_MAP[0:8](3–0)				TXDMA_MAP(3–0)			
	RX DMA ENA[0:8]	TX DMA ENA[0:8]	NO BRK[0:8]	ICOUNT[0:8](4–0)					Reserved	COUNTBIT17[0:8]	COUNT[0:8](5–0)					
0xF8 - 0x114h DMAxCOUNT page 693	ICOUNTx(15–0)															
	COUNTx(15–0)															
0x118h DMACNTLEN page 694	Reserved															
	Reserved															LARGE COUNT
0x11Ch	Reserved															
	Reserved															
0x120h UERRCTRL page 695	Reserved															
	Reserved							PTEST-EN	Reserved				EDEN(3–0)			
0x124h UERRSTAT page 696	Reserved															
	Reserved														EDFLG1	EDFLG0

[1] The base address of these registers can be found in the device data sheet.

Offset Address[1] 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
0x128h UERRADDR1 page 697	Reserved															
	Reserved						UERRADDR1(9–0)									
0x12Ch UERRADDR0 page 698	Reserved															
	Reserved							UERRADDR0(8–0)								
0x130h RXOVRN_BUF_ADDR page 699	Reserved															
	Reserved						RXOVRN_BUF_ADDR(9–0)									
0x134h IOLPBKTSTCR page 700	Reserved							SCS FAIL FLG	Reserved			CTRL_ BITERR	CTRL_ DESYNC	CTRL_ PAR ERR	CTRL_ TIME OUT	CTRL_ DLEN ERR
	Reserved				IOLPBKTSTENA(3–0)				Reserved		ERR SCS PIN(2–0			CTRL SCS PIN ERR	LPBK_ TYPE	RXP_ ENA

[1] The base address of these registers can be found in the device data sheet.

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.

14.9.1 SPI Global Control Register 0 (SPIGCR0)

Figure 14-1. SPI Global Control Register 0 (SPIGCR0) [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	7	6	5	4	3	2	1	0
Reserved															nRE-SET
R-0															R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 14-8. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and writes have no effect.
0	nRESET		Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be done. After setting this bit, auto Initialization of multi-buffer RAM starts.
		0	SPI is in the reset state.
		1	SPI is out of the reset state.

14.9.2 SPI Global Control Register 1 (SPIGCR1)

Figure 14-2. SPI Global Control Register 1 (SPIGCR1) [offset = 04h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							SPIEN	Reserved							LOOP-BACK
R-0							R/W-0	R-0							R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							POWER-DOWN	Reserved						CLK-MOD	MAS-TER
R-0							R/W-0	R-0						R/W-0	R/W-0

R = Read in all modes; W = Write in all modes; WP = Write in privilege mode only; -n = value after reset

Table 14-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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 SPIDAT0 and SPIDAT1 registers •All the fields of the SPIFLG register •Contents of SPIBUF and the internal RXBUF registers
		0	The SPI is not activated for transfers.
		1	Activates SPI
23–17	Reserved		Reads return zero and 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 areis 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 disabled.
		1	Internal loop-back test mode enabled.

Table 14-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (Continued)

Bit	Name	Value	Description
15–9	Reserved		Reads return zero and 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		Reads return zero and 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 SPIENA and SPISCS[3:0] pins in functional mode.
		0	Clock is external. •SPIENA is an output. •SPISCS[3:0] are inputs.
		1	Clock is internally-generated. •SPIENA is an output. •SPISCS[3:0] are outputs.
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 set 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 is anare inputs, SPISOMI[7:0] pins is anare outputs
		1	SPISOMI[7:0] pins is anare inputs, SPISIMO[7:0] pins is anare outputs

14.9.3 SPI Interrupt Register (SPIINT0)

Figure 14-3. SPI Interrupt Register (SPIINT0) [offset = 08h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TXINTENA	RXINTENA	Reserved	RXOVRNINTENA	Reserved	BITERRENA	DESYNCENA	PARERRENA	TIMEOUTENA	DLENERRENA
R-0						R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = write, P = Privilege mode; -*n* = Value after reset

Table 14-10. SPI Interrupt Register (SPIINT0) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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		Reads return zero and writes have no effect.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. Enable DMA REQ only after setting the SPIEN bit to 1.
		0	DMA is not used.
		1	DMA requests will be generated. **Note: A DMA request will be generated on the TX DMA REQ line each time a word is copied to the shift register either from TXBUF or directly from SPIDAT0/SPIDAT1 writes.** **Note: A DMA request will be generated on the RX DMA REQ line each time a word is copied to the SPIBUF register either from RXBUF or directly from the shift register.**
15–10	Reserved		Reads return zero and writes have no effect.
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 (SPIFLG[9]) is set to 1.
		0	No interrupt will be generated upon TXINTFLG being set to 1.

Table 14-10. SPI Interrupt Register (SPIINT0) Field Descriptions (Continued)

Bit	Name	Value	Description
		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 (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		Reads return zero and writes have no effect.
6	RXOVRNINTENA		Overrun interrupt enable.
		0	Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved		Reads return zero and writes have no effect.
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables an interrupt on a 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.

Table 14-10. SPI Interrupt Register (SPIINT0) Field Descriptions (Continued)

Bit	Name	Value	Description
		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 $\overline{\text{SPISCS}}$ pins are used, if the incoming valid $\overline{\text{SPISCS}}$ 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.

14.9.4 SPI Interrupt Level Register (SPILVL)

Figure 14-4. SPI Interrupt Level Register (SPILVL) [offset = 0Ch]

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
Reserved						TXINTLVL	RX INTLVL	Reserved	RXOVRNINTLVL	Reserved	BITER-RLVL	DESYN-CLVL	PAR-ERRLVL	TIME-OUTLVL	DLEN-ERRLVL
R-0						R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write in any mode; -*n* = Value after reset

Table 14-11. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and 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		Reads return zero and 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		Reads return zero and 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.

Table 14-11. SPI Interrupt Level Register (SPILVL) Field Descriptions (Continued)

Bit	Name	Value	Description
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	DLEN ERR LVL		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.

14.9.5 SPI Flag Register (SPIFLG)

Note:
SPIFLAG bits are cleared on read. Software must check all flag bits when reading this register.

Figure 14-5. SPI Flag Register (SPIFLG) [offset = 10h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TXINTFLG	RXINTFLG	Reserved	RXOVRNINTFLG	Reserved	BITERRFLG	DESYNCFLG	PARERRFLG	TIMEOUTFLG	DLENERRFLG
R-0						R-0	R/WC-0	R-0	R/WC-0	R-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

R = Read; WC = Write/read Clear; -*n* = Value after reset

Table 14-12. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and 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 BUF INIT ACTIVE 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		Reads return zero and writes have no effect.

Table 14-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
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, one 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		Reads return zero and writes have no effect.

Table 14-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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 SPIFLG register 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 (e.g. 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 SPIFLG register 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		Reads return zero and writes have no effect.

Table 14-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
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. • Set 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.
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only.
		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. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.
2	PARITYERRFLG		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.
		0	No parity error detected.
		1	A parity error occurred. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.

Table 14-12. SPI Flag Register (SPIFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	TIMEOUTFLG		Time-out caused by nonactivation of ENA signal.
		0	No ENA-signal time-out occurred.
		1	An ENA signal time-out occurred. The SPI generates a time-out because the slave hasn't 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, i.e. the SPI doesn't re-start a data transfer from this buffer. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set SPIENA bit to 0.
0	DLEN ERR FLG		Data-length error flag. This flag can be cleared by one of the following methods. • Write a 1 to this bit. • Set 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.

14.9.6 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 14-6. SPI Pin Control Register 0 (SPIPC0) [offset = 14h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIFUN[7:0]								SIMOFUN[7:-0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIFUN0	SIMOFUN0	CLKFUN	ENAFUN	SCSFUN[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-UR/W-0	R/W-UR/W-0							

R = Read, W = write, -*n* = Value after reset

Table 14-13. SPI Pin Control (SPIPC0) Field Descriptions

Bit	Name	Value	Description
31-24	SOMIFUN[7:0]		Slave out, master in function. Determines whether SPISOMI[x] is to be used as a general-purpose I/O pin or as a SPI functional pin. **Note: Duplicate Control Bits for SPISOMI[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 SPISOMI[0] pin. The read value of Bit 24 always reflects the value of bit 11.**
		0	SPISOMI[x] pin is a GIO pin.
		1	SPISOMI[x] pin is a SPI functional pin.
23-16	SIMOFUN[7:0]		Slave in, master out function. Determines whether SPISIMO[x] is to be used as a general-purpose I/O pin or as a SPI functional pin. **Note: Duplicate Control Bits for SPISIMO[x].** **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.**
		0	The SPISIMOx pin is a GIO pin.
		1	The SPISIMOx pin is a SPI functional pin
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-13. SPI Pin Control (SPIPC0) Field Descriptions (Continued)

Bit	Name	Value	Description
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: Regardless of the number of parallel pins used, the 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: Regardless of the number of parallel pins used, the 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	Reserved		Always write a 0 to this bit. Reads are undefined.
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.
7-0	Reserved		Always write a 0 to these bits. Reads are undefined.
7-0	SCSFUN[7:0]		$\overline{\text{SPISCSx}}$ function. Determines whether each $\overline{\text{SPISCSx}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. If the slave $\overline{\text{SPISCSx}}$ pins are in functional mode and receive an inactive high signal, the slave SPI will place its output in high-z and disable shifting.
		0	The $\overline{\text{SPISCSx}}$ pin is a GIO pin.
		1	The $\overline{\text{SPISCSx}}$ pin is a SPI functional pin.

14.9.7 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 14-7. SPI Pin Control Register 1 (SPIPC1) [offset = 18h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMI DIR[7:0]								SIMO DIR[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMI0 DIR	SIMO0 DIR	CLKDIR	ENADIR	SCSDIR[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-0U	R/W-0U							

R = Read, W = Write; -*n* = Value after reset

Table 14-14. SPI Pin Control Register (SPIPC1) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDIR[7:0]		SPISOMIx direction. Controls the direction of SPISOMIx 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	SPISOMIx pin is an input.
		1	SPISOMIx pin is an output.
23–16	SIMODIR[7:0]		SPISIMOx direction. Controls the direction of SPISIMOx 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 SPISIMO0.** **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 pin. The read value of Bit 16 always reflects the value of bit 10.**
		0	SPISIMOx pin is an input.
		1	SPISIMOx pin is an output.

Table 14-14. SPI Pin Control Register (SPIPC1) Field Descriptions (Continued)

Bit	Name	Value	Description
15–12	Reserved		Reads return zero and writes have no effect.
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	SPISOMI0 pin is an input.
		1	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	SPISIMO0 pin is an input.
		1	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	SPICLK pin is an input.
		1	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 CLK-MOD bit (SPIGCR1[1]).
		0	$\overline{\text{SPIENA}}$ pin is an input.
		1	$\overline{\text{SPIENA}}$ pin is an output.
7–0	SCSDIR[7:0]		$\overline{\text{SPISCSx}}$ direction. These bits control the direction of each $\overline{\text{SPISCSx}}$ pin when it is used as a general-purpose I/O pin. Each pin could be configured independently from the others If the $\overline{\text{SPISCSx}}$ is used as a SPI functional pin; the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	$\overline{\text{SPISCSx}}$ pin is an input.
		1	$\overline{\text{SPISCSx}}$ pin is an output.

14.9.8 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 14-8. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDIN[7:0]								SIMODIN[7:0]							
R-U								R-U							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIDIN0	SIMODIN0	CLKDIN	ENADIN	SCSDIN[7:0]							
R-0				R-U	R-U	R-U	R-U	R-U							

R = Read; W = Write; U = Undefined; -*n* = Value after reset

Table 14-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Name	Value	Description
31-24	SOMIDIN[7:0]		SPISOMIx data in. The value of the SPISOMIx pins.
		0	The SPISOMIx pin is logic 0.
		1	The SPISOMIx pin is logic 1.
23–16	SIMODIN[7:0]		SPISIMOx data in. The value of the SPISIMOx pins.
		0	The SPISIMOx pin is logic 0.
		1	The SPISIMOx pin is logic 1.
15–12	Reserved		Reads return zero and 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

Table 14-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions (Continued)

Bit	Name	Value	Description
8	ENADIN		$\overline{\text{SPIENA}}$ data in. The 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[7:0]		$\overline{\text{SPISCSx}}$ data in. The value of the $\overline{\text{SPISCSx}}$ pins.
		0	The $\overline{\text{SPISCSx}}$ pin is logic 0.
		1	The $\overline{\text{SPISCSx}}$ pin is logic 1

14.9.9 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 14-9. SPI Pin Control Register 3 (SPIPC3) [offset = 20h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDOUT[7:0]								SOMIDOUT[7:0]							
R/W-U								R/W-U							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMID OUT0	SIMOD OUT0	CLKD-OUT	ENADOUT	SCSDOUT[7:0]							
R-0				R/W-U	R/W-U	R/W-U	R/W-U	R/W-U							

R = Read; W = Write; -*n* = Value after reset

Table 14-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDOUT[7:0]		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[7:0]		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 SPISOMI0. 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		Reads return zero and writes have no effect.

Table 14-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
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[7:0]		$\overline{\text{SPISCSx}}$ data out write. Only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose I/O pins and configured as an output pins. The value of these bits indicates the value sent to the pins.
		0	The $\overline{\text{SPISCSx}}$ pin is logic 0.
		1	The $\overline{\text{SPISCSx}}$ pin is logic 1.

14.9.10 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 14-10. SPI Pin Control Register 4 (SPIPC4) [offset = 24h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">SOMISET[7:0]</td><td colspan="8">SOMISET[7:0]</td></tr>
<tr><td colspan="8">R/W-U</td><td colspan="8">R/W-U</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="4">Reserved</td><td>SOMISET0</td><td>SIMOSET0</td><td>CLKSET</td><td>ENASET</td><td colspan="8">SCSSET[7:0]</td></tr>
<tr><td colspan="4">R-0</td><td>R/W-U</td><td>R/W-U</td><td>R/W-U</td><td>R/W-U</td><td colspan="8">R/W-U</td></tr>
</table>

R = Read; W = Write; -*n* = Value after reset

Table 14-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions

Bit	Name	Value	Description
31–24	SOMISET[7:0]		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 SOMI0 pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	*Read:* SPISIMOx is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISOMIx is logic 1. *Write:* Logic 1 is placed on SPISOMIx pin if it is in general-purpose output mode.
23-17	Reserved		Reads return zero and writes have no effect.
23–16	SIMOSET[7:0]		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 SOMI0 pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	*Read:* SPISIMIx is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISIMOx is logic 1. *Write:* Logic 1 is placed on SPISIMOx pin if it is in general-purpose output mode.
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions (Continued)

Bit	Name	Value	Description
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 is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISOMI0 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 is logic 0. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* SPISIMO0 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 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 the 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:* SPIENA is logic 0. *Write:* A write to this bit has no effect.
		1	*Read:* SPIENA 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[7:0]		$\overline{\text{SPISCSx}}$ data out set. This bit is only active when the $\overline{\text{SPISCSx}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit sets the corresponding $\overline{\text{SCSDOUT}}$ bit to 1.
		0	*Read:* $\overline{\text{SPISCSx}}$ is logic 0. *Write:* A write to this bit has no effect.
		1	*Read:* $\overline{\text{SPISCSx}}$ is logic 1. *Write:* Logic 1 placed on $\overline{\text{SPISCSx}}$ pin if it is in general-purpose output mode.

14.9.11 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 14-11. SPI Pin Control Register 5 (SPIPC5) [offset = 28h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMICLR[7:0]								SIMOCLR[7:0]							
R/W-U								R/W-U							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIC LR0	SIMOC LR0	CLKCLR	ENACLR	SCSCLR[7:0]							
R-0				R/W-U	R/W-U	R/W-U	R/W-0U	R/W-0U							

R = Read; W = Write; -*n* = Value after reset

Table 14-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions

Bit	Name	Value	Description
31–24	SOMICLR[7:0]		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 clear the pin SOMI0. 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: A write of 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[7:0]		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 clear the pin SOMI0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	*Read:* The current value on SIMODOUTx is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SIMODOUTx is 1. *Write:* Logic 0 is placed on SPISIMOx pin if it is in general-purpose output mode.
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMICLR0		SPISOMI0 data out clear. This bit 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: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SPISOMI0 is 1. *Write:* Logic 0 is placed on SPISOMIx pin if it is in general-purpose output mode.
10	SIMOCLR0		SPISIMO0 data out clear. This bit 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: A write of 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.
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: A write of 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 ENA is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on ENA is 1. *Write:* Logic 0 is placed on $\overline{\text{SPIENA}}$ pin if it is in general-purpose output mode.
7–0	SCSCLR[7:0]		$\overline{\text{SPISCSx}}$ data out clear. This bit is only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose output pins.
		0	*Read:* The current value on SCSDOUTx is 0. Write: A write of 0 to this bit has no effect.
		1	*Read:* The current value on SCSDOUTx is 1. *Write:* Logic 0 is placed on the $\overline{\text{SPISCS}}$ pin if it is in general-purpose output mode.

14.9.12 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 14-12. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIPDR[7:0]								SIMOPDR[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIP DR0	SIMOP DR0	CLK-PDR	ENAPDR	SCSPDR[7:0]							
R-0				R/W-0	R/W-0	R/W-0	R/W-0	R/W-0							

R = Read; W = Write; -*n* = Value after reset

Table 14-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIPDR[7:0]		SPISOMIx open drain enable. This bit enables open drain capability for the SPISOMIx pin if the following conditions are met: • SOMIDIRx = 1 (SPISOMIx pin configured in GIO mode as an output) • SOMIDOUTx = 1 **Bit 11 or bit 24 can both be used to enable open-drain for SOMI0. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	The output value on the SPISOMIx pin is logic 1.
		1	Output pin SPISOMIx is in a high-impedance state.
23-16	SIMOPDR[7:0]		SPISIMOx open drain enable. This bit enables open drain capability for the SPISIMOx pin if the following conditions are met: • SIMODIRx = 1 (SPISIMOx pin configured in GIO mode as an output) • SIMODOUTx = 1 **Bit 10 or bit 16 can both be used to enable open-drain for SIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	The output value on SPISIMOx pin is logic 1.
		1	Output pin SPISIMOx is in a high-impedance state.
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIPDR0		SOMI0 open-drain enable. This bit enables open-drain capability for SOMI0 if the following conditions are met. • SOMI0 pin configured in GIO mode as output pin • Output value on SPISOMI0 pin is logic 1.
		0	Output value 1 of SPISOMI0 pin is logic 1.
		1	Output value 1 of SPISOMI0 is high-impedance.
10	SIMOPDR0		SPISIMO0 open-drain enable. This bit enables open -drain capability for the SPISIMO0 pin if the following conditions are met. • SIMO0 pin configured in GIO mode as output pin • Output value on SPISIMO0 pin is logic 1.
		0	Output value 1 of SPISIMO0 pin is logic 1.
		1	Output value 1 of SPISIMO0 is high-impedance.
9	CLKPDR		CLK open drain enable. This bit enables open drain capability for the pin CLK if the following conditions are met: • SPICLK pin configured in GIO mode as an output pin • SPICLKDOUT = 1
		0	Output value on CLK pin is logic 1.
		1	Output pin CLK is in a high-impedance state.
8	ENAPDR		SPIENA pin open drain enable. This bit enables open drain capability for SPIENA if the following conditions are met: • SPIENA pin configured in GIO mode as an output pin • SPIENADOUT = 1
		0	Output value on SPIENA pin is logic 1.
		1	Output pin SPIENA is in a high-impedance state.
7–0	SCSPDR[7:0]		SPISCSx open drain enable. This bit enables open drain capability for the SPISCSx pin if the following conditions are met: • SPISCS pin configured in GIO mode as an output pin • SPISCSDOUTx = 1
		0	Output value on SCSx pin is logic 1.
		1	Output pin SCSx is in a high-impedance state.

14.9.13 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 14-13. SPI Pin Control Register 7 (SPIPC7) [offset = 30h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIDIS[7:0]								SIMODIS[7:0]							
R/W-n								R/W-n							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIP DIS0	SIMOP DIS0	CLKDIS	ENAPDIS	SCSPDIS[7:0]							
R-0				R/W-n	R/W-n	R/W-n	R/W-nU	R/W-nU							

R = Read; W = Write; -*n* = Value after reset

Table 14-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIDIS[7:0]		SOMIx pull control enable/disable. This bit enables pull control capability for the SOMIx 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 SOMIO. 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[7:0]		SIMOx pull control enable/disable. This bit enables pull control capability for the SIMOx 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 SIMO0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull control on SPISIMOx pin is enabled.
		1	Pull control on SPISIMOx pin is disabled.
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions (Continued)

Bit	Name	Value	Description
11	SOMIPDIS0		SPISOMI0 pull control enable/disable. This bit enables 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 enable/disable. This bit enables 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 SPISIMO0 pin is enabled.
		1	Pull control on SPISIMO0 pin is disabled.
9	CLKPDIS		CLK pull control enable/disable. This bit enables 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 CLK pin is enabled.
		1	Pull control on CLK pin is disabled.
8	ENAPDIS		ENABLE pull control enable/disable. This bit enables 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 ENABLE pin is enabled.
		1	Pull control on ENABLE pin is disabled.
7–0	SCSPDIS[7:0]		SCSx pull control enable/disable. This bit enables pull control capability for the pin $\overline{\text{SPISCSx}}$ pin if it is in input mode regardless of whether it is in functional or GIO mode.
		0	Pull control on SCSx pin is enabled.
		1	Pull control on SCSx pin is disabled.

14.9.14 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 14-14. SPI Pin Control Register 8 (SPIPC8) [offset = 34h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SOMIPSEL[7:0]								SIMOPSEL[7:0]							
R/W-n								R/W-n							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SOMIPSEL	SIMOPSEL	CLKPSEL	ENAPSEL	SCSPSEL[7:0]							
R-0				R/W-n	R/W-n	R/W-n	R/W-	R/W-nU							

R = Read; W = Write; -*n* = Value after reset

Table 14-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

Bit	Name	Value	Description
31–24	SOMIPSEL[7:0]		SPISOMIx pull select. This bit selects the type of pull logic at the SOMIx 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 SOMIx pin
		1	Pull up on SOMIx pin
23–16	SIMOPSEL[7:0]		SPISIMOx pull select. This bit selects the type of pull logic at the SPISIMOx pin. **Note: Bit 10 or bit 16 can be used to set pull-select for SPISOMI0. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull down on SPISIMOx pin
		1	Pull up on SPISIMOx pin
15–12	Reserved		Reads return zero and writes have no effect.

Table 14-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

Bit	Name	Value	Description
11	SOMIPSEL		SOMI pull select. This bit selects the type of pull logic at the SOMI pin.
		0	Pull down on the SPISOMI pin
		1	Pull up on the SOMI pin
10	SIMOPSEL		SPISIMO pull select. This bit selects the type of pull logic at the SPISIMO pin.
		0	Pull down on SPISIMO pin
		1	Pull up on SPISIMO pin
9	CLKPSEL		CLK pull select. This bit selects the type of pull logic at the CLK pin.
		0	Pull down on CLK pin
		1	Pull up on CLK pin
		0	Input buffer for CLK is disabled if PULLDIS = 1
		1	Input buffer for CLK is enabled if PULLDIS = 1
8	ENAPSEL		ENABLE pull select. This bit selects the type of pull logic at the ENABLE pin.
		0	Pull down on ENABLE pin.
		1	Pull up on ENABLE pin.
7–0	SCSPSEL[7:0]		SCSx pull select. This bit selects the type of pull logic at the SCSx pin.
		0	Pull down on SCSx.
		1	Pull up on SCSx pin.

14.9.15 SPI Transmit Data Register 0 (SPIDAT0)

Figure 14-15. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h]

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
TXDATA[15:0]															
R/W-0															

R = Read, W = write, -*n* = Value after reset

Table 14-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	TXDATA(15–0)	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 register 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 SPIDAT1register.** **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.**

14.9.16 SPI Transmit Data Register 1 (SPIDAT1)

Figure 14-16. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CS HOLD	Reserve d	WDEL	DFSEL		CSNR							
R-0			R/W-0	R-0	R/W-0	R/W-0		R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATA[15:0]															
R/W-0															

R = Read, W = write, -*n* = Value after reset

Table 14-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return zero and 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		Reads return zero and 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, $\overline{\text{SPISCS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the SPISCS 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 SPISCS 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 $\overline{\text{SPISCS}}$ pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.

Table 14-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions (Continued)

Bit	Name	Value	Description
25–24	DFSEL		Data word format select
		00	Data word format 0 is selected
		01	Data word format 1 is selected
		10	Data word format 2 is selected
		11	Data word format 3 is selected
23–16	CSNR	0–FFh	Chip select number. CSNR defines the chip-select that will be activated during the data transfer. **Note: Writing to only the control field 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 fields in the control field to select the required phase/polarity combination.**
15–0	TXDATA(15–0)	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. Write to this register ONLY when using the automatic slave chip-select feature (see Section 14.2, *Operating Modes* on page 582 for more information). A write to this register will drive the contents of CSNR[3:0] on the $\overline{\text{SPISCS}}$[3:0] pins, if they are configured as functional pins. 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.**

14.9.17 SPI Receive Buffer Register (SPIBUF)

Figure 14-17. SPI Receive Buffer Register (SPIBUF) [offset = 40h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN-ERR	LCSNR							
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDATA[15:0)]															
R-0															

R = Read, W = write, C = Clear; S = Set; -*n* = Value after reset

Table 14-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions

Bit	Name	Value	Description
31	RXEMPTY		**Receive data buffer empty.** When the host reads the SPIBUF field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into the SPIBUF field.
		1	No data has been received since the last read of SPIBUF.
			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 SPIFLG register.
			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).

Table 14-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

Bit	Name	Value	Description
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 (e.g. CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either 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). This flag is cleared to 0 when the RXDATA 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 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.
28	BITERR		**Bit error.** There was a mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**

Table 14-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

Bit	Name	Value	Description
		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 guaranteed to be for the current buffer.**
		0	No slave desynchronization detected. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		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 portion of the SPIBUF register is read.**
		0	No parity error detected.
		1	A parity error occurred.

Table 14-24. SPI Receive Buffer Register (SPIBUF) Field Descriptions (Continued)

Bit	Name	Value	Description
25	TIMEOUT		**Time-out because of non-activation of ENA 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 SPIFLG register is set. **This bit is valid only in master mode.** **This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.**
		0	No ENA-pin time-out occurred.
		1	An ENA signal time-out occurred.
24	DLENERR		**Data length error flag.** **Note: This flag is cleared to 0 when the RXDATA portion 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[15:0]	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.

14.9.18 SPI Emulation Register (SPIEMU)

Figure 14-18. SPI Emulation Register (SPIEMU) [offset = 44h]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">EMU_RXDATA[15:0]</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

R = Read, -*n* = Value after reset

Table 14-25. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–0	EMU_RXDATA [15:0]	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.

14.9.19 SPI Delay Register (SPIDELAY)

Figure 14-19. SPI Delay Register (SPIDELAY) [offset = 48h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C2TDELAY[7:0]								T2CDELAY[7:0]							
R/W-0								R/W-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2EDELAY[7:0]								C2EDELAY[7:0]							
R/W-0								R/W-0							

R = Read, -*n* = Value after reset

Table 14-26. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Name	Value	Description
31–24	C2TDELAY[74:0]	0-1FF	**Chip-select-active to transmit-start delay.** See Figure 14-20 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 **Note: If C2TDELAY = 0, then $t_{C2TDELAY}$ = 0.** 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 14-26. SPI Delay Register (SPIDELAY) Field Descriptions (Continued)

Bit	Name	Value	Description
23–16	T2CDELAY[74:0]	0–1FF	**Transmit-end-to-chip-select-inactive-delay.** See Figure 14-21 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 **Note: If T2CDELAY = 0, then $t_{T2CDELAY}$ = 0** 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.** **Note: C2TDELAY and T2CDELAY counters do not count when SPIENA stops transmission.** Both C2TDELAY and T2CDELAY counters do not have any dependency on the SPIENA pin value. Even if the 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 SPIENA pin is deasserted by the slave, the master will continue to hold the SPISCS pins active until the T2CDELAY counter overflows. In this way, it is guaranteed that the setup and hold times of the SPISCS 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.

Table 14-26. SPI Delay Register (SPIDELAY) Field Descriptions (Continued)

Bit	Name	Value	Description
15–8	T2EDELAY[7:0]	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 SPIENA signal has to become inactive and after SPISCS 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 isn't disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the SPIENA signal isn't deactivated in time. The DESYNC flag is set to indicate that the slave device did not de-assert its SPIENA pin in time to acknowledge that it received all bits of the sent word. See Figure 14-22 for an example of this condition. **Note: DESYNC is also set if the SPI detects a de-assertion of 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 µs, otherwise DESYNC is set and an interrupt is asserted (if enabled).
7–0	C2EDELAY[7:0]	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 14-23 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 (SPISCS), otherwise the TIMEOUT flag is set and an interrupt is asserted (if enabled).

Figure 14-20. Example: $t_{C2TDELAY}$ = 8 VCLK Cycles

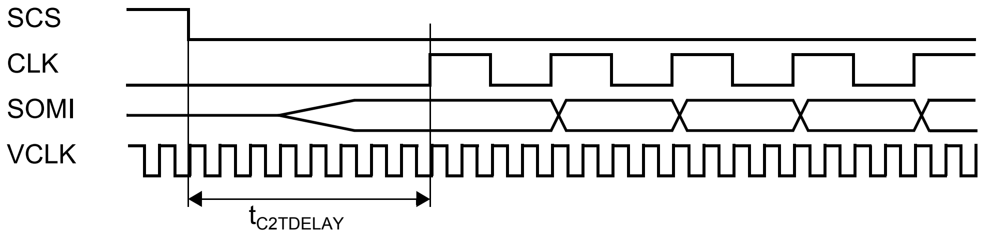


Figure 14-21. Example: $t_{T2CDELAY}$ = 4 VCLK Cycles

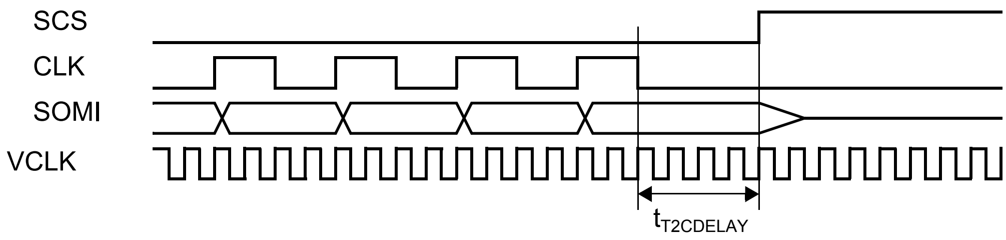


Figure 14-22. Transmit-Data-Finished-to-ENA-Inactive-Timeout

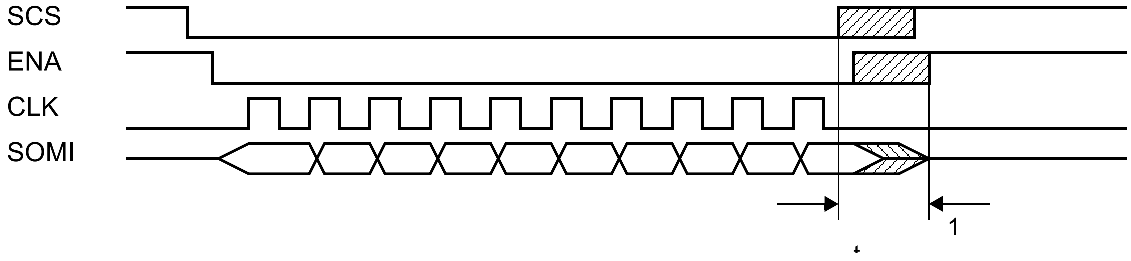
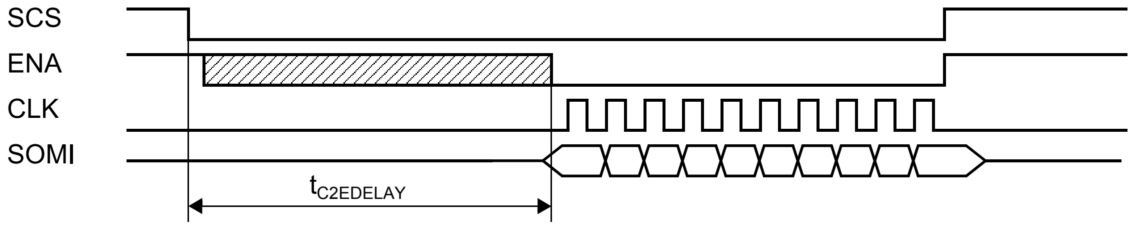


Figure 14-23. Chip-Select-Active-to-ENA-Signal-Active-Timeout



14.9.20 SPI Default Chip Select Register (SPIDEF)

Figure 14-24. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch]

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
Reserved								CSDEF							
R-0								R/W-1							

R = Read, W = Write; -*n* = Value after reset

Table 14-27. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads return zero and writes have no effect.
7–0	CSDEF	0-FFh	**Chip select default pattern.** Master-mode only. The CSDEFx bits are output to the $\overline{\text{SPISCS}}$ pins when no transmission is being performed. It allows the user to set a programmable chip-select pattern that deselects all of the SPI slaves.
		0	$\overline{\text{SPISCS}}$x is set to 0 when no transfer is active.
		1	$\overline{\text{SPISCS}}$x is set to 1 when no transfer is active.

14.9.21 SPI Data Format Registers (SPIFMT[3:0])

Figure 14-25. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch–50h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		WDELAY[5:0]						PAR POL	PARITY ENA	WAIT ENA	SHIFT DIR	Reserved	DIS CS TIMERS	POLARITY	PHASE
R-0		R/WP-0						R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-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[7:0]								Reserved			CHARLEN[4:0]				
R/WP-0								R-0			R/WP-0				

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 14-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–24	WDELAY[5:0]	0–3F	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.
		0	No parity generation/ verification is performed for this data format.
		1	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. **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.**

Table 14-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (Continued)

Bit	Name	Value	Description
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.
19-18	Reserved		Reads return zero and writes have no effect.
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.

Table 14-28. SPI Data Format Registers (SPIFMT[3:0]) Field Descriptions (Continued)

Bit	Name	Value	Description
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: 1. 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. 2. 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 set to 0 the SPI clock signal is low-inactive, i.e., before and after data transfer the clock signal is low.
		1	If POLARITYx is set to 1 the SPI clock signal is high-inactive, i.e., 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 set 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[7:0]		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: $BR_{Formatx} = \frac{VBUSPCLK}{(PRESCALEx + 1)}$ **Note: When PRESCALEx is set to 0, the SPI clock rate defaults to VCLK/2.**
7–5	Reserved		Reads return zero and writes have no effect.
4–0	CHARLEN[4:0]	0–1F	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 0x10 (data word length = 16). Illegal values, such as 0x00 or 0x1F are not allowed; their effect is indeterminate.

14.9.22 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 14-26. ***Interrupt Vector 0*** **(*****INTVECT0*****) [offset = 60h]**

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
Reserved										INTVECT0					SUSP- END0
R-0										R-0					R-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 14-29. ***Transfer Group Interrupt Vector 0*** **(*****INTVECT0*****)]**

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and 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.
		00000	There is no pending interrupt.
		00001 + x	Transfer group x (x=0,..,15) has a pending interrupt. SUSPEND0 reflects the type of interrupt (*suspended* or *finished*).
		10001	Error Interrupt pending. The lower half of SPIFLG contains more details about the type of error.
		10011	The pending interrupt is a “Receive Buffer Overrun” interrupt.
		10010	**SPI mode:** The pending interrupt is a “Receive Buffer Full” interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		10100	**SPI mode:** The pending interrupt is a “Transmit Buffer Empty” interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		all other combinations	**SPI mode:** Reserved. These bit combinations should not occur.

Table 14-29. *Transfer Group Interrupt Vector 0* (*INTVECT0*)] (Continued)

Bit	Name	Value	Description
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 SPIFLG register. 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 SPIFLG register or by reading the RXOVERN_BUF_ADDR register.

14.9.23 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 14-27. *Interrupt Vector 1* (*INTVECT1*) [offset = 64h]

31–16
Reserved
R-0

15–6	5–1	0
Reserved	INTVECT1	SUSP- END1
R-0	R-0	R-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 14-30. *Transfer Group Interrupt Vector 1* (*INTVECT1*)]

Bit	Name	Value	Description
31–6	Reserved		Reads return zero and 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.
		00000	There is no pending interrupt. **SPI mode only.**
		10001	Error Interrupt pending. The lower half of SPIINT1 contains more details about the type of error. **SPI mode only.**
		10011	The pending interrupt is a “Receive Buffer Overrun” interrupt. **SPI mode only.**
		10010	The pending interrupt is a “Receive Buffer Full” interrupt. **SPI mode only.**
		10100	The pending interrupt is a “Transmit Buffer Empty” interrupt. **SPI mode only.**
		all other combinations	Reserved. These bit combinations should not occur. **SPI mode only.**

Table 14-30. *Transfer Group Interrupt Vector 1 (INTVECT1)*] (Continued)

Bit	Name	Value	Description
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 SPIFLG register. 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 SPIFLG register or by reading the RXOVERN_BUF_ADDR register.

14.9.24 Parallel/Modulo Mode Control Register (SPIPMCTRL)

Note:
Do not configure MODCLKPOLx and MMODEx bits since this device does not support modulo mode.

Note:
The bits of this register are used in conjunction with the SPIFMTx registers. Each byte of this register corresponds to one of the SPIFMTx registers.
1. Byte0 (Bits 7:0) are used when SPIFMT0 register is selected by DFSEL[1:0] = 00 in the control field of a buffer.
2. Byte1 (Bits 15:8) are used when SPIFMT1 register is selected by DFSEL[1:0] = 01 in the control field of a buffer.
3. Byte2 (Bits 23:16) are used when SPIFMT2 register is selected by DFSEL[1:0] = 10 in the control field of a buffer.
4. Byte3 (Bits31:24) are used when SPIFMT3 register is selected by DFSEL[1:0] = 11 in the control field of a buffer.

Figure 14-28. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch]

31 30	29	28 27 26	25 24	23 22	21	20 19 18	17 16
Reserved	MOD CLK POL3	MMODE 3(2–0[	PMODE 3(1–0)	Reserved	MOD CLK POL2	MMODE 2(2–0)	PMODE 2(1–0)
R-0	R/WP-0	R/WP-0	R/WP-0	R-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
Reserved	MOD CLK POL1	MMODE 1(2–0)	PMODE 1(1–0)	Reserved	MOD CLK POL0	MMODE 0(2–0)	PMODE 0(1–0)
R-0	R/WP-0	R/WP-0	R/WP-0	R-0	R/WP-0	R/WP-0	R/WP-0

R = Read, W = write, P = Privilege mode, -*n* = Value after reset

Table 14-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zeros and writes have no effect.
29	MOD CLK POL 3		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MODULO MODE[2:0] bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.

Table 14-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
28–26	MMODE 3(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		000	Normal single dataline mode - Default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
25–24	PMODE 3(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	00 = normal operation/1-data line (MMODE should be set to 000)
		01	01 = 2-data line mode (MMODE should be set to 000)
		10	10 = 4-data line mode (MMODE should be set to 000)
		11	11 = 8-data line mode (MMODE should be set to 000)
23–22	Reserved		Reads return zeros and writes have no effect.
21	MOD CLK POL 2		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE[2:0] bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if modulo mode is selected.
20–18	MMODE 2(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if the modulo option is supported by the module).
		000	000 = 1-data line Mode - Default (PMODE should be set to 00)
		001	001 = 2-data line Mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved

Table 14-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
17–16	PMODE 2(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)
15–12	Reserved		Reads return zeros and writes have no effect.
13	MOD CLK POL 1		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE[2:0] bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if modulo mode is selected.
12–10	MMODE 1(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if the modulo option is supported by the module).
		000	1-data line mode - default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
9–8	PMODE 1(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)
7–6	Reserved		Reads return 0 and writes have no effect.

Table 14-31. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
5	MOD CLK POL 0		Modulo mode SPICLK polarity. Determines the polarity of the SPICLK in modulo mode only. If the MMODE[2:0] bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
4–2	MMODE 0(2–0)		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		000	1-data line mode - default (PMODE should be set to 00)
		001	2-data line mode (PMODE should be set to 00)
		010	3-data line mode (PMODE should be set to 00)
		011	4-data line mode (PMODE should be set to 00)
		100	5-data line mode (PMODE should be set to 00)
		101	6-data line mode (PMODE should be set to 01)
		110–111	Reserved
1–0	PMODE 0(1–0)		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4 or 8 data lines.
		00	Normal operation/1-data line (MMODE should be set to 000)
		01	2-data line mode (MMODE should be set to 000)
		10	4-data line mode (MMODE should be set to 000)
		11	8-data line mode (MMODE should be set to 000)

14.9.25 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 14-29. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							Reserved								MSPI- ENA
							R-0								R/WP-0

R = Read, W = write, P = Privilege mode, -*n* = Value after reset

Table 14-32. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved		Reads return zeros and writes have no effect.
16	RXRAMACCESS		**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 set 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		Reads return zeros and 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, i.e., in this mode the MibSPI is fully code-compliant to the standard TMS570 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.

14.9.26 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 14-30 and Table 14-33 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 14-30. TG Interrupt Enable Set Register (TGITENST) [offset = 74h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETINTENRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETINTENSUS[15:0]															
R/W-0															

R = Read; W = Write; -*n* = Value after reset

Table 14-33. TG Interrupt Enable Set Register (TGITENST) Field Descriptions

Bit	Name	Value	Description
31-16	SET INTENRDY[15:0]		TG interrupt set (enable) when transfer finished.
		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	SET INTEN SUS[15:0]		TG interrupt set (enabled) when transfer suspended
		0	*Read:* The TGx-suspended 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-suspended interrupt is enabled. The interrupt gets generated when TGx is suspended. *Write:* Enable the TGx-suspended interrupt. The interrupt gets generated when TGx is suspended.

14.9.27 MibSPI 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 14-31 and Table 14-34 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 14-31. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR INTENRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR INTENSUS[15:0]															
R/W-0															

R = Read; W = Write; *-n* = Value after reset

Table 14-34. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions

Bit	Name	Value	Description
31-16	CLR INTENRDY[15:0]		TG interrupt clear (disabled) when transfer finished.
		0	*Read:* The TGx-completed interrupt is disabled. The 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.
15–0	CLR INTENSUS[15:0]		TG interrupt clear (disabled) when transfer suspended.
		0	*Read:* The TGx-suspended 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-suspended interrupt is enabled. The interrupt gets generated when TGx completes. *Write:* Disables the TGx-suspended interrupt.

14.9.28 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 14-32 and Table 14-35 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 14-32. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SET INTLVLRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET INTLVLSUS[15:0]															
R/W-0															

R = Read; W = Write; -*n* = Value after reset

Table 14-35. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions

Bit	Name	Value	Description
31-16	SET INTLVLRDY[15:0]		Transfer-group completed interrupt level set.
		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	SET INTLVLSUS[15:0]		Transfer-group suspended interrupt level set.
		0	*Read:* 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 INT1.

14.9.29 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 14-33 and Table 14-36 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 14-33. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR INTLVLRDY[15:0]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR INTLVLSUS[15:0]															
R/W-0															

R = Read; W = Write; *-n* = Value after reset

Table 14-36. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

Bit	Name	Value	Description
31-16	CLR INTLVLRDY[15:0]		Transfer-group completed interrupt level clear.
		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:* Sets the TG-x completed interrupt to INT0.
15–0	CLR INTLVLSUS[15:0]		Transfer group suspended interrupt level clear.
		0	*Read:* The TG-x 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:* Sets the TGx-suspended interrupt to INT0.

14.9.30 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 14-34 and Table 14-37 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 14-34. Transfer Group Interrupt Flag Register (TGINTFLAG) [offset = 84h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTFLGRDY[15:0]															
R/WC-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTFLGSUS[15:0]															
R/WC-0															

R = Read; W = Write; C = Clear; -*n* = Value after reset

Table 14-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

Bit	Name	Value	Description
31-16	INTFLGRDY[15:0]		Transfer-group interrupt flag for a transfer-completed interrupt. **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.

Table 14-37. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions (Continued)

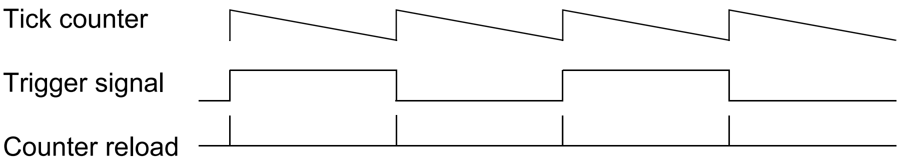
Bit	Name	Value	Description
15–0	INTFLGSUS[15:0]		Transfer-group interrupt flag for a transfer-suspend interrupt. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGSUSx referenced by the vector number given by 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.

14.9.31 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 14-35 as a square wave, illustrates the different trigger event types for the TGs (e.g., rising edge, falling edge, and both edges).

Figure 14-35. Tick Counter Operation



This register is shown in Figure 14-36 and described in Table 14-38.

Figure 14-36. Tick Count Register (TICKCNT) [offset = 90h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TICK ENA	RE LOAD	CLKCTRL(1–0)		Reserved											
R/W-0	R0/S-0	R/W-0		R-0											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TICKVALUE(15–0)															
R/W-0															

R = Read; W = Write; S = Set; C = Clear; *-n* = Value after reset;

Table 14-38. Tick Count Register (TICKCNT) Field Descriptions

Bit	Name	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[1:0]. When the tick counter is enabled it starts down-counting from its current value. When TICK-ENA goes from 0 to 1, the tick counter is automatically loaded with the contents of TICKVALUE.

Table 14-38. Tick Count Register (TICKCNT) Field Descriptions (Continued)

Bit	Name	Value	Description
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(1–0)		**Tick counter clock source control.** CLKCTRL defines the clock source that is used to clock the internal tick counter.
		00	SPICLK of data word format 0 is selected as the clock source of the tick counter
		01	SPICLK of data word format 1 is selected as the clock source of the tick counter
		10	SPICLK of data word format 2 is selected as the clock source of the tick counter
		11	SPICLK of data word format 3 is selected as the clock source of the tick counter
27–16	Reserved		Reads return 0 and writes have no effect.
15–0	TICKVALUE(15–0)	0–FFFF	**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.

14.9.32 Last TG End Pointer (LTGPEND)

Figure 14-37. Last TG End Pointer (LTGPEND) [offset = 94h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			TG IN SERVICE(4–0)					Reserved							
R-0			R-0					R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LPEND(6–0)							Reserved							
R-0	R/W-0							R-0							

R = Read, W = Write, C = Clear, -*n* = Value after reset, x = indeterminate

Table 14-39. Last TG End Pointer (LTGPEND) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return 0 and writes have no effect.
28–24	TG IN SERVICE(4–0)		**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.
		00000	No TG is being serviced by the sequencer.
		00001	TG0 is being serviced by the sequencer.
		. . .	...
		10000	TG15 is being serviced by the sequencer. **Note: The number of transfer groups varies by device.**
		10001–11111	Invalid values
23–15	Reserved		Reads return 0 and writes have no effect.
14–8	LPEND(6–0)	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, i.e., no end address is defined. Therefore LPEND has to be programmed to specify explicitly the end address of the last TG. **Note: LPEND allows SW compatibility for MibSPI variants Because of software compatibility reasons, the last TG end address has to be configurable; otherwise a super-set MibSPI cannot emulate a MibSPI with less TGs without software changes.** **Note: The number of transfer groups varies by device, thus the last TG also varies by device.**
7–0	Reserved		Reads return 0 and writes have no effect.

14.9.33 TGx Control Registers (TGxCTRL)

Each TG can be configured via one dedicated control register. The register description below shows one control register(x) which 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 14-38. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h–D4h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TG ENA[0:15]	ONE SHOT[0:15]	PRST[0:15]	TGTD[0:15]	Reserved				TRGEVT[0:15](3–0)				TRIGSRC[0:15](3–0)			
R/W-0	R/W-0	R/W-0	R-0	R-0				R/W-0				R/W-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PSTART[0:15](6–0)							Reserved	PCURRENT[0:15](6–0)						
R-0	R/W-0							R-0	R-0						

R = Read; W = Write; -*n* = Value after reset;

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions

Bit	Name	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.

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
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, CSHOLD or NOBRK buffer, then only after finishing those transfers, the TG will be restarted. This means that even if the TG is retriggered, the TG will only be restarted after finishing the transfer of the first non-LOCK or non-CSHOLD buffer. In the case of the NOBRK buffer, after completing the ICOUNT number of transfers, the TG will be restarted from its PSTART.** This means that TX control fields such as LOCK and CSHOLD, and DMA control fields such as NOBRK have higher priority over anything else. They have the capability to delay the restart of the TG even if it is retriggered 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		Reads return 0 and writes have no effect.

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description	
23–20	TRIGEVTx(3-0)		**Type of trigger event.** A level-triggered TG can be stopped by de-activating the level trigger. However, the following restrictions apply. 1. Deactivating the level trigger for a TG during a NOBRK transfer does not stop the transfers until all of the ICOUNT number of buffers are transferred for the NOBRK buffer. Once a NOBRK buffer is prefetched, the trigger event loses control over the TG until the NOBRK buffer transfer is completed. 2. 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. 3. 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.	
		0000	never	Never trigger TGx. This is the default value after reset.
		0001	rising edge	A rising edge (0 to 1) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0010	falling edge	A falling edge (1 to 0) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0011	both edges	Rising and falling edges at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		0100	Reserved	
		0101	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.**
		0110	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.**

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description	
		0111	always	A repetitive group transfer will be performed. **Note: By setting the TRIGSRC to 0000b, the TRIGEVT to 0111b (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.**
		1xxx	Reserved	

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description	
19–16	TRIGSRCx(3-0)		**Trigger source.** After reset, the trigger sources of all TGs are disabled.	
		0000	disabled	
		0001	EXT0	External trigger source 0. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0010	EXT1	External trigger source 1. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0011	EXT2	External trigger source 2. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0100	EXT3	External trigger source 3. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0101	EXT4	External trigger source 4. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0110	EXT5	External trigger source 5. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		0111	EXT6	External trigger source 6. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1000	EXT7	External trigger source 7. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1001	EXT8	External trigger source 8. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1010	EXT9	External trigger source 9. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1011	EXT10	External trigger source 10. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1100	EXT11	External trigger source 11. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1101	EXT12	External trigger source 12. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1110	EXT13	External trigger source 13. The actual source varies per device (e.g. HET I/O channel, event pin, etc.).
		1111	TICK	Internal periodic event trigger. The tick counter can initiate periodic group transfers.
15	Reserved		Reads return 0 and writes have no effect.	

Table 14-40. TG Control Registers (TGxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
14–8	PSTARTx(6-0)	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		Reads return 0 and writes have no effect.
6–0	PCURRENTx(6-0)	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; i.e. 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. An entire sequence of words transferred for a NOBRK DMA buffer.
3. Once the last word in a TG is pre-fetched.

14.9.34 DMA Channel Control Register (DMAxCTRL)

Each DMA channel can be configured via one dedicated control register. The register description below shows one exemplary control register that is identical for all DMA channels, e.g., the control register for DMA channel 0 is named DMA0CTRL. The MibSPI supports up to 8 bidirectional DMA channels.

The number of bidirectional DMA channels varies by device. The number of DMA channels and hence the number of DMA channel control registers may vary.

Figure 14-39. DMA Channel Control Register (DMAxCTRL) [offset = D8–114h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONE SHOTx	BUFIDx							RXDMA_MAPx				TXDMA_MAPx			
R/W-0	R/W-0							R/W-0				R/W-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX DMA ENAx	TX DMA ENAx	NO BRKx	ICOUNTx					Reserved	COUNT BIT17x	COUNTx					
R/W-0	R/W-0	R/W-0	R/W-0					x	R-0	R-0					

R = Read, W = Write, C = Clear, -*n* = Value after reset, x = indeterminate

Table 14-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions

Bit	Name	Value	Description
31	ONESHOT		Auto-disable of DMA channel after ICOUNT+1 transfers.
		0	The length of the block transfer is fully controlled by the DMA controller. The enable bits RXDMAENAx and TXDMAENAx are not modified by the MibSPI.
		1	ONESHOT allows a block transfer of defined length (ICOUNTx+1), mainly controlled by the MibSPI and not by the DMA controller. After ICOUNTx +1 transfers, the enable bits RXDMAENAx and TXDMAENAx are automatically cleared by the MibSPI, hence no more DMA requests are generated. In conjunction with NOBRKx, a burst transfer can be initiated without any other transfer through another buffer.
30–24	BUFIDx	0–7Fh	Buffer utilized for DMA transfer. BUFIDx defines the buffer that is utilized for the DMA transfer. In order to synchronize the transfer with the DMA controller with the NOBRK condition the “suspend to wait until...” modes must be used.
23–20	RXDMA_MAPx	0–Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. One request line for receive data and the other for request line for transmit data. RXDMA_MAPx[3:0] defines the number of the physical DMA Request line that is connected to the receive path of the MibSPI DMA channel. If RXDMAENAx and TXDMAENAx are both set to ‘1’, then RXDMA_MAPx[3:0] shall differ from TXDMA_MAPx[3:0] and shall differ from any other used physical DMA Request line. Otherwise unexpected interference may occur.

Table 14-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
19–16	TXDMA_MAPx	0–Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. ne request line for receive data and the other for request line for transmit data. TXDMA_MAPx[3:0] defines the number of the physical DMA Request line that is connected to the transmit path of the MibSPI DMA channel. If RXDMAENAx and TXDMAENAx are both set then TXDMA_MAPx[3:0] shall differ from RXDMA_MAPx[3:0] and shall differ from any other used physical DMA Request line. Otherwise unexpected interference may occur.
15	RXDMAENAx		Receive data DMA channel enable.
		0	No DMA request upon new receive data.
		1	The physical DMA channel for the receive path is enabled. The first DMA request pulse is generated after the first transfer from the referenced buffer (BUFIDx) is finished. The buffer should be configured in as “skip until RXEMPTY is set” or “suspend to wait until RXEMPTY is set” in order to ensure synchronization between the DMA controller and the MibSPI sequencer.
14	TXDMAENAx		Transmit data DMA channel enable.
		0	No DMA request upon new transmit data.
		1	The physical DMA channel for the transmit path is enabled. The first DMA request pulse is generated right after setting TXDMAENAx to load the first transmit data. The buffer should be configured in the as “skip until TXFULL is set” or “suspend to wait until TXFULL is set” in order to ensure synchronization between the DMA controller and the MibSPI sequencer.

Table 14-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
13	NOBRKx		Non-interleaved DMA block transfer. This bit is available in master mode only. **Note: Special Conditions during a NOBRK Buffer Transfer. If a NOBRK DMA buffer is currently being serviced by the sequencer, then it is not allowed to be disabled prematurely.** **During a NOBRK transfer, the following operations are not allowed:** • **Clearing the NOBRKx bit to 0** • **Clearing the RXDMAENAx to 0 (if it is already 1)** • **Clearing the TXDMAENAx to 0 (if it is already 1)** • **Clearing the BUFMODE[2:0] bits to 000** **Any attempts to perform these actions during a NOBRK transfer will produce unpredictable results.**
		0	DMA transfers through the buffer referenced by BUFIDx are interleaved by data transfers from other active buffers or TGs. Every time the sequencer checks the DMA buffer, it performs one transfer and then steps to the next buffer.
		1	NOBRKx ensures that ICOUNTx+1 data transfers are performed from the buffer referenced by BUFIDx without a data transfer from any other buffer. The sequencer remains at the DMA buffer until ICOUNTx+1 transfers have been processed. For example, this can be used to generate a burst transfer to one device without disabling the chip select signal in-between (the concerned buffer has to be configured with CSHOLD=1). Another example would be to have a defined block data transfer in slave mode, synchronous to the master SPI. **Note: Triggering of higher priority TGs or enabling of higher priority DMA channels will not interrupt a NOBRK block transfer.**
12–8	ICOUNTx	0–1Fh	**Initial count of DMA transfers.** ICOUNTx is used to preset the transfer counter COUNTx[4:0]. Every time COUNTx hits zero it is reloaded with ICOUNTx. The real number of transfers equals ICOUNTx plus one. If ONESHOTx is set, ICOUNTx defines the number of DMA transfers that are performed before the MibSPI automatically disables the DMA channels. If NOBRKx is set, ICOUNTx defines the number of DMA transfers that are performed in one sequence without a transfer from any other buffer. **Note: See** Section 14.9.35, *DMAxCOUNT Register (ICOUNT)* **on page 693 and** Section 14.9.36, *DMA Large Count (DMACNTLEN)* **on page 694 about how to increase the ICOUNT to a 16-bit value. With this extended capability, MibSPI can transfer a block of up to 65535 (65K) words without interleaving (if NOBRK is used) or without deasserting the chipselect between the buffers (if CSHOLD is used).**

Table 14-41. DMA Channel Control Register (DMAxCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
7	Reserved		Reads return zeros and writes have no effect.
6	COUNT BIT17x		The 17th bit of the COUNT field of DMAxCOUNT register.
5–0	COUNTx	0–3Fh	Actual number of remaining DMA transfers. This field contains the actual number of DMA transfers that remain, until the DMA channel is disabled, if ONESHOTx is set. **Note: If the TX and RX DMA requests are enabled, the COUNT register will be decremented when the RX has been serviced.**

14.9.35 DMAxCOUNT Register (ICOUNT)

> **Note:**
> These registers are used only if the LARGE COUNT bit in the DMACNTLEN register is set.

> **Note:**
> The number of bidirectional DMA channels varies by device. The number of DMA channels and hence the number of DMA registers varies by device.

Figure 14-40. DMAxCOUNT Register (ICOUNT) [offset = F8–114h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">ICOUNTx[15:0]</td></tr>
<tr><td colspan="16">R/W-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">COUNTx[15:0]</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

R = Read; W = Write; -*n* = Value after reset

Table 14-42. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions

Bit	Name	Value	Description
31–16	ICOUNTx	0–FFFFh	**Initial number of DMA transfers.** ICOUNTx is used to preset the transfer counter COUNTx. Every time COUNTx hits zero, it is reloaded with ICOUNTx. The real number of transfer equals ICOUNTx plus one. If ONESHOTx is set, ICOUNTx defines the number of DMA transfers that are performed before the MibSPI automatically disables the corresponding DMA channel. If NOBRKx is set, ICOUNTx defines the number of DMA transfers that are performed in one sequence without a transfer from any other buffer
15–0	COUNTx	0–FFFFh	**The actual number of remaining DMA transfers.** COUNTx Contains the actual number of DMA transfers that remain, until the DMA channel is disabled, if ONESHOTx is set. Since the real counter value is always ICOUNTx +1, the 17th bit of COUNTx is available on DMACTRLx[6] bit. **Note: Usage Tip for Block Transfer Using a Single DMA Request. It is possible to use the multi-buffer RAM to transfer chunks of data to/from an external SPI. A DMA Controller can be used to handle the data in bursts. Suppose a chunk of 64 bytes of data needs to be transferred and a single DMA request needs to be generated at the end of transferring the 64 bytes. This can be easily achieved by configuring a TG register for the 64 buffer locations and using the DMAxCTRL/DMAxCOUNT registers to configure the last buffer (64th) of the TG as the BUFID and enable RXDMA (NOBRK = 0). At the end of the transfer of the 64th buffer, a DMA request will be generated on the selected DMA request channel. The DMA controller can do a burst read of all 64 bytes from RXRAM and/or then do a burst write to all 64 bytes to the TXRAM for the next chunk.**

14.9.36 DMA Large Count (DMACNTLEN)

Figure 14-41. DMA Large Count Register (DMACNTLEN) [offset = 118h]

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
Reserved															LARGE COUNT
R-0															R/WP-0

R = Read, -*n* = Value after reset

Table 14-43. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and writes have no effect.
0	LARGE COUNT		Select either the 16-bit DMAxCOUNT counters or the smaller counters in DMAxCTRL.
		0	**Select the DMAxCTRL counters.** Writes to the DMAxCTRL register will modify the ICOUNT value. Reading ICOUNT and COUNT can be done from the DMAxCTRL register. The DMAxCOUNT register should not be used since any write to this register will be overwritten by a subsequent write to the DMAxCTRL register to set the TXDMAENA or RXDMAENA bits.
		1	**Select the DMAxCOUNT counters.** Writes to the DMAxCTRL register will not modify the ICOUNT value. The ICOUNT value must be written to in the DMAxCOUNT register before the RXDMAENA or TXDMAENA bits are set in the DMAxCTRL register. The DMAx-COUNT register should be used for reading COUNT or ICOUNT.

14.9.37 Multi-buffer RAM *Uncorrectable Parity Error Control Register (UERRCTRL)*

Figure 14-42. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h]

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
Reserved							PTESTEN	Reserved				EDEN[3:0]			
R-0							R/WP-0	R-0				R/WP-0101			

R = Read, WP = Write in Privilege mode only, -*n* = Value after reset

Table 14-44. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and 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 14.11, *Parity Memory* on page 713 for further details about parity memory testing.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7–4	Reserved		Reads return zero and writes have no effect.
3–0	EDEN[3:0]		Error detection enable. These bits enable parity error detection.
		0101	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 detection.**

14.9.38 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT)

Figure 14-43. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h]

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
Reserved														EDFLG1	EDFLG0
R-0														R/C-0	R/C-0

R = Read; C = Clear; -*n* = Value after reset

Table 14-45. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and 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 zero 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 zero to this bit has no effect.
		1	*Read:* An error was detected and the address is captured in the UERRADDR0 register. *Write:* The bit was cleared.

14.9.39 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1)

Figure 14-44. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h]

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
Reserved						UERRADDR1[9:0]									
R-0						R/C-x									

R = Read, C = Clear; -*n* = Value after power-on reset; -*x* = Interdeterminate

Table 14-46. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9–0	OVERRADDR1 [9:0]	200–3FFh	**Uncorrectable parity error address for RXRAM.** This register holds the address where a parity error is generated while reading RXRAM. Only the CPU or DMA 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 0x200 - 0x3FF. The register does not clear its contents during or after module-level reset, system-level reset or even power-up reset. A read operation to this register clears its contents to the default value 0x200. After a power-up 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.**

14.9.40 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0)

Figure 14-45. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [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	7	6	5	4	3	2	1	0
Reserved							UERRADDR0[8:0]								
R-0							R/C-x								

R = Read; C-Clear; -*n* = Value after power-on reset; -*x* = Indeterminate

Table 14-47. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
8–0	UERRADDR1 [8:0]	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 0x000 - 0x1FF. The register does not clear its contents during or after module-level reset, system-level reset, or even power-up reset. A read operation to this register clears its contents to all 0s. After a power-up 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.**

14.9.41 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 14-46. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h]

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
Reserved						RXOVRN_BUF_ADDR[9:0]									
R-0						R-0x200									

R = Read, -*n* = Value after power-on reset

Table 14-48. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		Reads return zero and writes have no effect.
9–0	RXOVRN_BUF_ADDR[9:0]	200–3FCh	**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 0x200 - 0x3FC. Contents of this register are valid only when any of the TGINTVECT0 or TGINTVECT1 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.**

14.9.42 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 IO LPBK TST ENA field is set to 1010.

Figure 14-47. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							SCS FAIL FLG	Reserved			CTRL BIT ERR	CTRLD ES YNC	CTRL PAR ERR	CTRL TIME OUT	CTRL DLEN ERR
R-0							R/C-0	R-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
Reserved				IOLPBKTSTENA[3:0]				Reserved		ERR SCS PIN			CTRL SCS PIN ERR	LPBK TYPE	RXP ENA
R-0				R/WP-0101				R-0		R/WP-000			R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in Privilege mode only, C = Clear; -*n* = Value after power-on reset

Table 14-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zero and writes have no effect.
24	SCS FAIL FLAG		Bit indicating a failure on SPISCS 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 zero 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 SPISCS[3:0] pins failed. A stuck-at fault is detected on one of the SPISCS[3:0]. 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		Reads return zero and 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	Introduces bit errors by inverting the value of the incoming data during loopback.

Table 14-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (Continued)

Bit	Name	Value	Description
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 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.
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 $\overline{\text{SPISCS}}$ pin (if functional) is forced to 1 when the module starts shifting data.
15–12	Reserved		Reads return zero and writes have no effect.
11–8	IOLPBKTSTENA[3:0]		Module I/O loopback test enable key.
		1010	Enable I/O loopback test mode.
		All other values	Disable I/O loopback test mode.
7–6	Reserved		Reads return zero and writes have no effect.

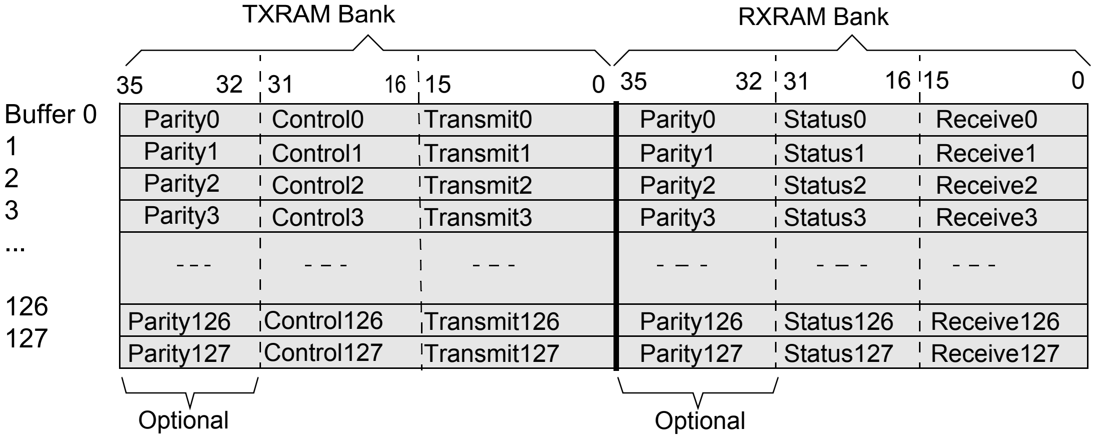
Table 14-49. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (Continued)

Bit	Name	Value	Description
5–3	ERR SCS PIN[2:0]		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 chipselect pin selected by this field is forced to the opposite of its value in the CSNR.
		000	Select $\overline{\text{SPISCS}}$[0] for injecting error
		001	Select $\overline{\text{SPISCS}}$[1] for injecting error
		. . .	. . .
		111	Select $\overline{\text{SPISCS}}$[7] for injecting error.
2	CTRL SCS PIN ERR		Enable/disable the injection of an error on the $\overline{\text{SPISCS}}$[3:0] pins. The individual $\overline{\text{SPISCS}}$[3:0] pins can be chosen using the ERR SCS PIN field.
		0	Disable the $\overline{\text{SPISCS}}$[3:0] error-inducing logic.
		1	Enable the $\overline{\text{SPISCS}}$[3:0] error-inducing logic.
1	LPBK TYPE		**Module I/O loopback type (analog/digital).** See Figure 14-23 for the different types of loopback modes.
		0	Enable Digital loopback when IOLPBKTSTENA = 1010.
		1	Enable Analog loopback when IOLPBKTSTENA = 1010.
0	RXP ENA		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.

14.10 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 14-48. A 4-bit parity field per word is also included in each bank of RAM.

Figure 14-48. Multi-Buffer RAM Configuration



	TXRAM 35–32	TXRAM 31–16	TXRAM 15–0	RXRAM 35–32	RXRAM 31–16	RXRAM 15–0
Buffer 0	Parity0	Control0	Transmit0	Parity0	Status0	Receive0
1	Parity1	Control1	Transmit1	Parity1	Status1	Receive1
2	Parity2	Control2	Transmit2	Parity2	Status2	Receive2
3	Parity3	Control3	Transmit3	Parity3	Status3	Receive3
...	- - -	- - -	- - -	- - -	- - -	- - -
126	Parity126	Control126	Transmit126	Parity126	Status126	Receive126
127	Parity127	Control127	Transmit127	Parity127	Status127	Receive127

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.

Note:
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.

14.10.1 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’ (Multibuffer RAM initialization is complete) before configuring multi-buffer RAM.

Besides the default auto initialization after reset, the auto-initialization sequence can also be done in following ways:

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 MSIENA 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:
The multibuffer related registers might be cleared during the auto initialization. Therefore, do NOT trigger auto Initialization after configuring any Multibuffer mode registers.

14.10.2 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 offset of the multi-buffer RAM is 0xFF0E 0000.

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
Transmit Buffers																
0x00h–1FCh Buffer[0:127]	BUFMODE(2–0)			CS HOLD	LOCK	WDEL	DFSEL(1–0)		CSNR(7–0)							
	TXDATA(15–0)															
Receive Buffers:																
0x200–3FCh Buffer[0:127]	RXEM PTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARIT YERR	TIME OUT	DLEN ERR	LCSNR(7–0)							
	RXDATA(15–0)															

14.10.3 Multi-buffer RAM Transmit Data Register

Each word of TXRAM is a transmit-buffer register.

Figure 14-49. Multi-buffer RAM Transmit Data Register [offset = Base + 000–1FFh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUFMODE			CS HOLD	LOCK	WDEL	DFSEL[1:0]		CSNR[7:0]							
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	7	6	5	4	3	2	1	0
TXDATA[15:0]															
R/W-0															

R = Read, W = write, -*n* = Value after reset

Table 14-50. Multi-buffer RAM Transmit Data Register Field Descriptions

Bit	Name	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.
		000	**disabled.** The buffer is disabled
		001	**skip single-transfer mode.** Skip this buffer until the corresponding TXFULL flag is set (i.e. new transmit data is available).
		010	**skip overwrite-protect mode.** Skip this buffer until the corresponding RXEMPTY flag is set (i.e. new receive data can be stored in RXDATA without data loss).
		011	**skip single-transfer overwrite-protect mode**. Skip this buffer until both of the corresponding TXFULL and RXEMPTY flags are set. (i.e. new transmit data available and previous data received by the host).
		100	**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.
		101	**suspend single-transfer mode.** Suspend-to-wait until the corresponding TXFULL flag is set (i.e. the sequencer stops at the current buffer until new transmit data is written in the TXDATA field).

Table 14-50. Multi-buffer RAM Transmit Data Register Field Descriptions (Continued)

Bit	Name	Value	Description
		110	**suspend overwrite-protect mode.** Suspend-to-wait until the corresponding RXEMPTY flag is set (i.e. the sequencer stops at the current buffer until the previously-received data is read by the host.
		111	**suspend single-transfer overwrite-protect mode.** Suspend-to-wait until the corresponding TXFULL and RXEMPTY flags are set (i.e. 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 in compatibility-mode of MibSPI, (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	LOCK		Lock two consecutive buffer words. Do not allow interruption by TG's 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 will be ignored.**
		0	No delay will be inserted. However, SPISCS pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the SPISCS 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 SPISCS 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 SPISCS pins will be de-activated for at least (WDELAY + 2) * VCLK_Period duration.

Table 14-50. Multi-buffer RAM Transmit Data Register Field Descriptions (Continued)

Bit	Name	Value	Description
25–24	DFSEL[2:0]		Data word format select
		00	Data word format 0 is selected
		01	Data word format 1 is selected
		10	Data word format 2 is selected
		11	Data word format 3 is selected
23–16	CSNR[7:0]	0–FF	Chip select number. CSNR defines the chip-select that will be activated during the data transfer. **Note:** **1) Writing to only the control field (using byte writes) 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 fields in the control field to select the required phase/polarity combination.** **2) Bits [23:20] is not writable in the device due to non availability of Chip select pins CS[4:7].**
15–0	TXDATA[15:0]	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 SPIDAT1 to 0x0000. Write to this register ONLY when using the automatic slave chip-select feature (see Section 14.2, *Operating Modes* on page 582 for more information). A write to this register will drive the contents of CSNR[7:0] on the SPISCS[3:0] pins, if they are configured as functional pins. 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.**

14.10.4 Multi-buffer RAM Receive Buffer Register

Each word of RXRAM is a receive-buffer register.

Figure 14-50. Multi-buffer RAM Receive Buffer Register [offset = RAM Base + 200–3FFh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	DLEN-ERR	LCSNR[7:0]							
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDATA[15:0)]															
R/W-0															

R = Read, W = write, C = Clear; S = Set; -*n* = Value after reset

Table 14-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
31	RXEMPTY		**Receive data buffer empty.** When the host reads the SPIBUF field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into the SPIBUF field.
		1	No data has been received since the last read of SPIBUF.
			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 SPIFLG register. 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).

Table 14-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
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 (e.g. CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either 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). This flag is cleared to 0 when the RXDATA 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 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.
		11	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.

Table 14-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
28	BITERR		**Bit error.** There was a mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		1	A bit error occurred. The MibSPI 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: There is a possible inconsistency of DESYNC in the 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 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.** **This inconsistency in the desync flag is valid only in the compatibility mode of MibSPI. In multi-buffer mode, the desync flag is always guaranteed to be for the current buffer.**
		0	No slave desynchronization detected. **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		1	A slave device is desynchronized.

Table 14-51. Multi-buffer Receive Buffer Register Field Descriptions

Bit	Name	Value	Description
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 portion 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 ENA pin.** The MibSPI 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 SPIFLG register is set. **This bit is valid only in master mode.** **This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.**
		0	No ENA-pin time-out occurred.
		1	An ENA signal time-out occurred.
24	DLENERR		**Data length error flag.** **Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.**
		0	No data-length error has occurred.
		1	A data length error has occurred.
23–16	LCSNR[7:0]	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. The updated after transmission during the write-back of received data.
15–0	RXDATA[15:0]	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.

14.11 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/DMA 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 14-51 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.

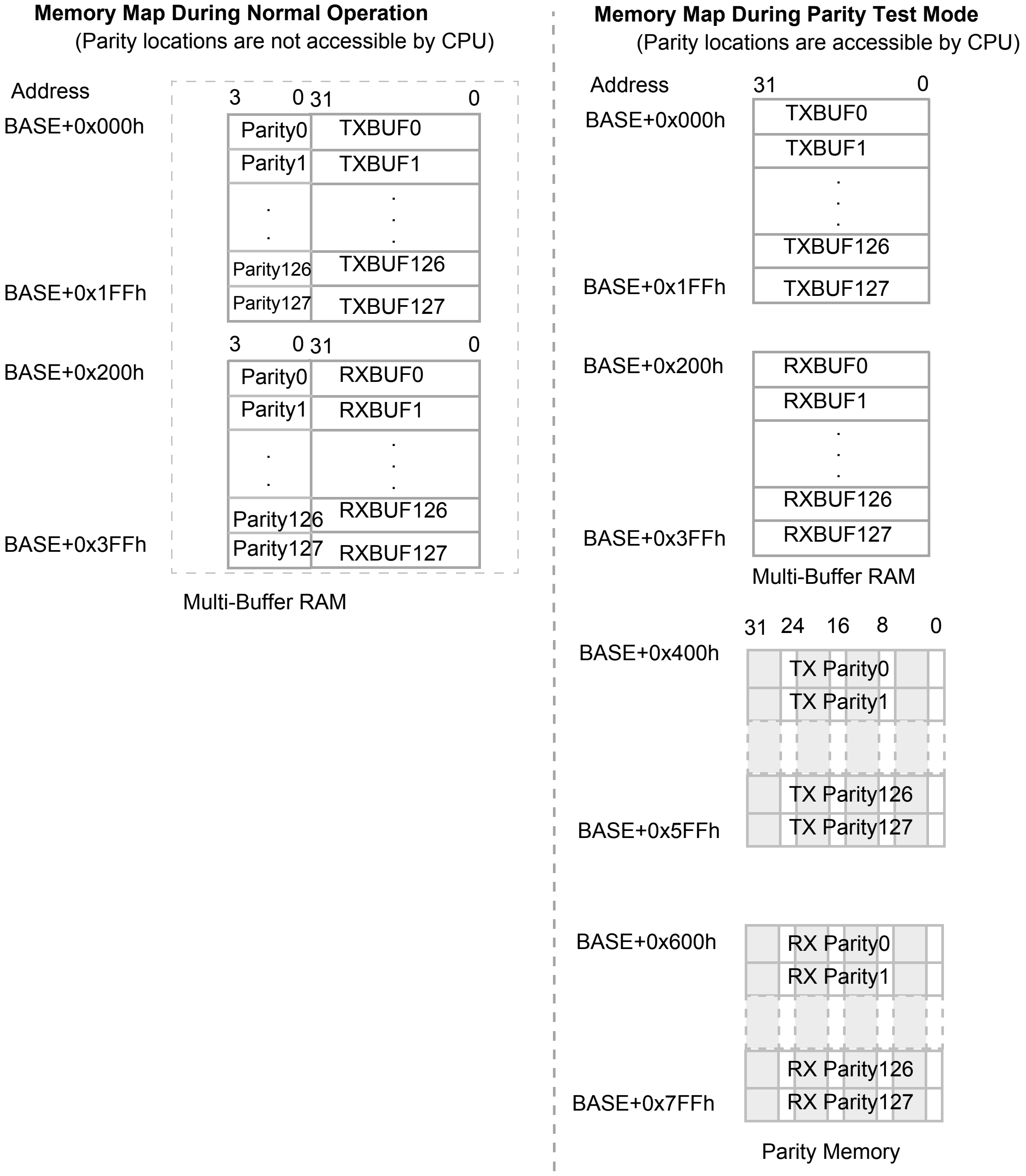
See the device-specific data sheet to get the actual base address of the multi-buffer RAM.

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 14-52 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 14-51. Memory Map for Parity Locations during Normal and Test Mode



Memory Map During Normal Operation
(Parity locations are not accessible by CPU)

Memory Map During Parity Test Mode
(Parity locations are accessible by CPU)

* BASE - Base Address of Multi-Buffer RAM
Refer to specific Device Data sheet
for the actual value of BASE.

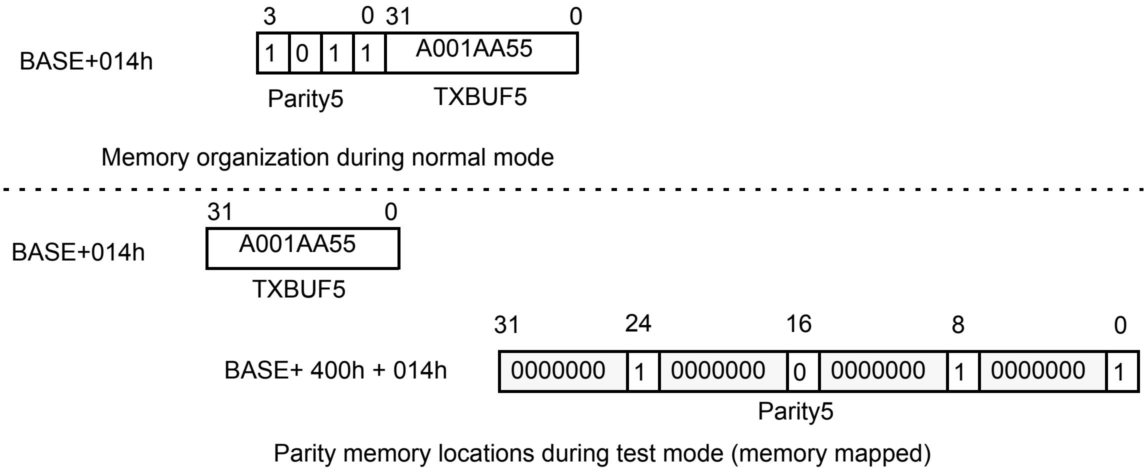
* Shaded areas indicate they are physically not present.

14.11.1 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 14-52 for a diagram.

Figure 14-52. Example of Memory Mapped Parity Locations during Test Mode



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.

14.12 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:

- o SPICLK (SPI Clock) - Output
- o SPISIMO (SPI Slave In Master Out) - Output
- o $\overline{\text{SPISCS}}$[7:0] (SPI Slave Chip Selects) - Output
- o SPISOMI (SPI Slave Out Master In) - Input
- o $\overline{\text{SPIENA}}$ (SPI slave ready Enable) - Input

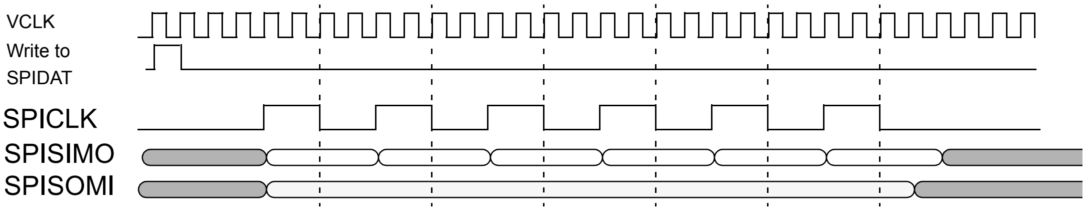
Slave mode SPI:

- o SPICLK - Input
- o SPISIMO - Input
- o $\overline{\text{SPISCS}}$ - Input
- o SPISOMI - Output
- o $\overline{\text{SPIENA}}$ - Output

All the timing diagrams given below are with Phase='0' & Polarity = '0' unless explicitly stated otherwise.

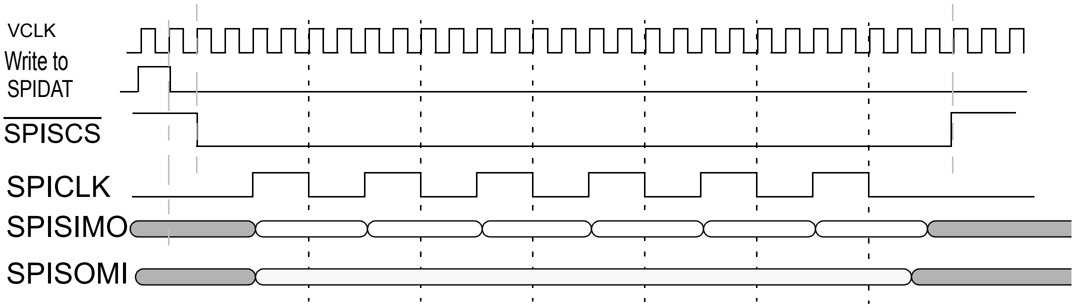
14.12.1 Master Mode Timings for SPI/MibSPI

Figure 14-53. SPI/MibSPI Pins during Master mode 3-pin configuration



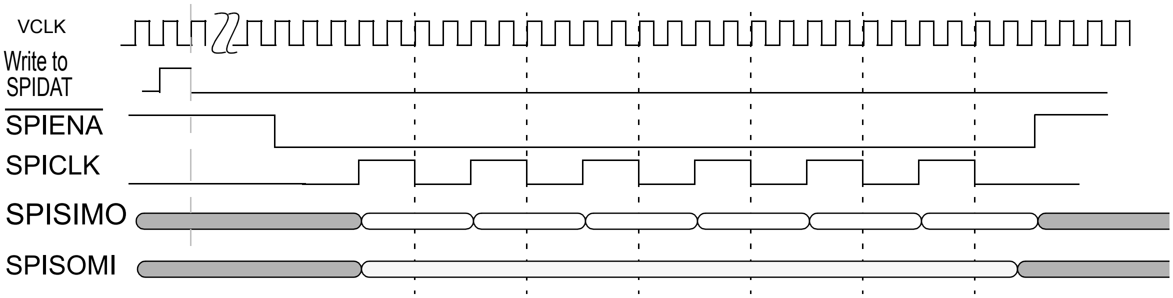
* Dotted vertical lines indicate the receive edges

Figure 14-54. SPI/MibSPI pins during Master mode 4-pin with $\overline{\text{SPISCS}}$ configuration.



* Dotted vertical lines indicate the receive edges

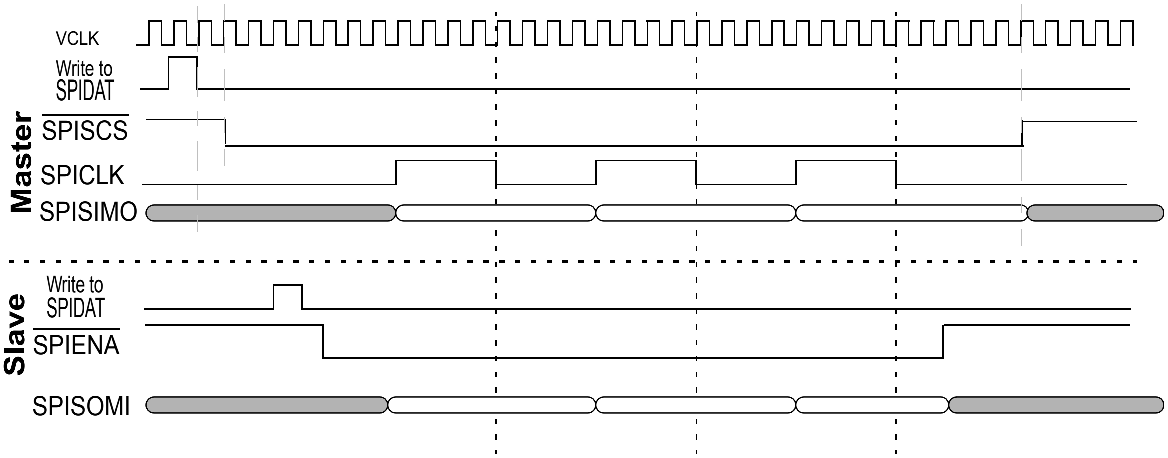
Figure 14-55. SPI/MibSPI pins during Master mode in 4pin with $\overline{\text{SPIENA}}$ configuration



* De-activation of $\overline{\text{SPIENA}}$ pin is controlled by the Slave.

* Dotted vertical lines indicate the receive edges

Figure 14-56. SPI/MibSPI pins during Master/Slave mode with 5-pin configuration

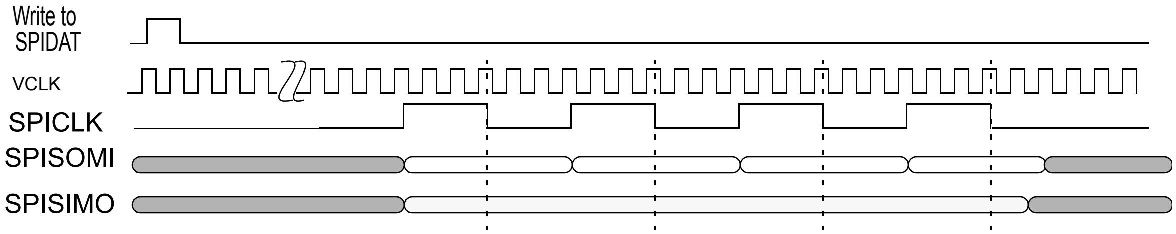


* Dotted vertical lines indicate the receive edges for the Master

* ENABLE_HIGHZ is set to ‘0’ in Slave SPI

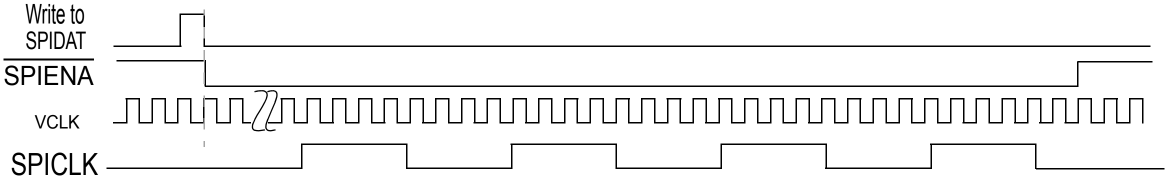
14.12.2 Slave Mode Timings for SPI/MibSPI

Figure 14-57. SPI/MibSPI pins during Slave mode 3-pin configuration



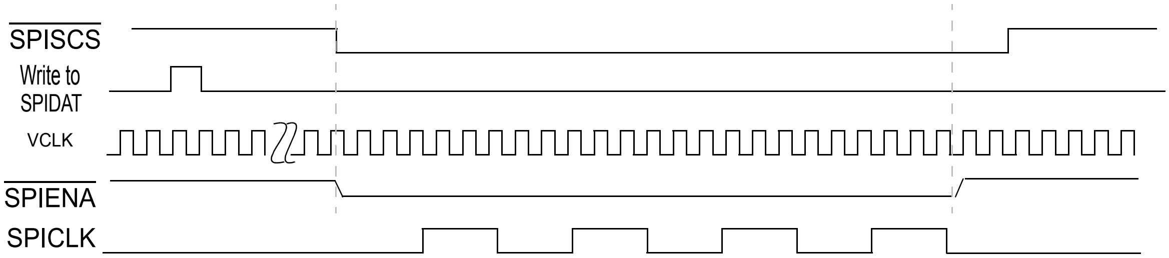
* Dotted vertical lines indicate the receive edges

Figure 14-58. SPI/MibSPI pins during Slave mode in 4pin with $\overline{\text{SPIENA}}$ configuration



* Diagram shows a relationship between the $\overline{\text{SPIENA}}$ from Slave and SPICLK from Master

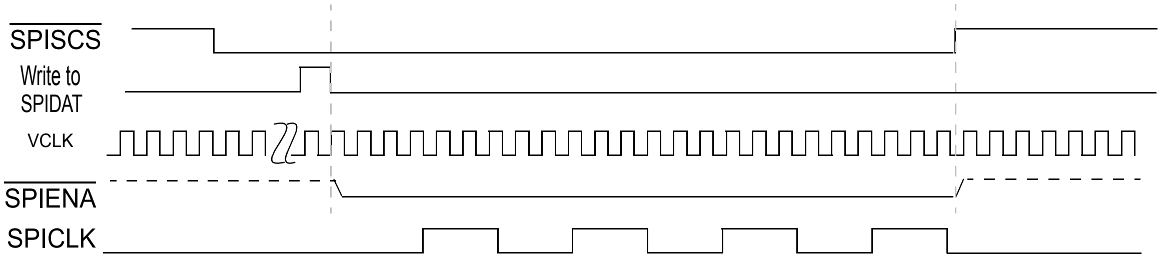
Figure 14-59. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single Slave).



* ENABLE_HIGHZ is set to ‘0’ in Slave SPI

* Diagram shows relationship between the $\overline{\text{SPISCS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in Push-Pull mode

Figure 14-60. SPI/MibSPI pins during Slave mode in 5pin configuration - (Single/Multi Slave).



* ENABLE_HIGHZ is set to ‘1’ in Slave SPI

* Diagram shows relationship between the $\overline{\text{SPISCS}}$ from a Master to $\overline{\text{SPIENA}}$ from Slave SPI when $\overline{\text{SPIENA}}$ is configured in High-Z mode

14.12.3 Timing Parameters of SPI/MibSPI pins in all the modes.

Table 14-52. Timing parameters of SPI/MibSPI pins.

SI No.	Delay Item	Master	Slave	Note
1	SPICLK edge to SPISIMO	0	-	
2	SPICLK edge to SPISOMI	-	0	
3a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = ‘0’ (4pin configuration- Master, CSHOLD = ‘0’)	2VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being ‘0’ for the current and the previous buffer transfer
3b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = ‘0’ (4pin configuration- Master, CSHOLD = ‘1’)	3VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being ‘1’ for the current and the previous buffer transfer
4a	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = ‘1’ (4pin configuration- Master, CSHOLD = ‘0’)	0.5 SPICLK + 2VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being ‘0’ for the current and the previous buffer transfer
4b	$\overline{\text{SPISCS}}$ - SPICLK edge Phase = ‘1’ (4pin configuration- Master, CSHOLD = ‘1’)	0.5 SPICLK + 3VCLK	-	Delay between $\overline{\text{SPISCS}}$ assertion by the Master to the first edge of its SPICLK. This timing is based on CSHOLD being ‘1’ for the current and the previous buffer transfer
5	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = ‘0’ (4pin/5pin configuration - Master)	3VCLK	-	Delay between $\overline{\text{SPIENA}}$ assertion from the Slave to the start of first edge of SPICLK by the Master
6	$\overline{\text{SPIENA}}$ - SPICLK edge Phase = ‘1’ (4pin/5pin configuration - Master)	0.5 SPICLK + 3VCLK	-	
7	SPICLK - $\overline{\text{SPIENA}}$ Phase, Polarity = “00” or “11” (4pin/5pin configuration - Slave)	-	1.5 VCLK to 2.5 VCLK	Delay between the last fall-edge of SPICLK to the deassertion of $\overline{\text{SPIENA}}$ pin by the Slave
8	SPICLK - $\overline{\text{SPIENA}}$ Phase, Polarity = “01” or “10” (4pin/5pin configuration - Slave)	-	1.5 VCLK to 2.5 VCLK	Delay between the last rise-edge of SPICLK to the deassertion of $\overline{\text{SPIENA}}$ pin by the Slave.
9	SPICLK - $\overline{\text{SPISCS}}$ Phase = ‘0’, Master	0.5 SPICLK + 1VCLK	-	Delay between the last edge of SPICLK to de-activation of $\overline{\text{SPISCS}}$ pin by the Master
10	SPICLK - $\overline{\text{SPISCS}}$ Phase = ‘1’, Master	1VCLK	-	
11	$\overline{\text{SPISCS}}$ - $\overline{\text{SPIENA}}$ (5pin), Slave	-	0	Delay between assertion of $\overline{\text{SPISCS}}$ by the Master to the assertion of $\overline{\text{SPIENA}}$ pin by the Slave (assuming Slave is ready with the TX data)

Table 14-52. Timing parameters of SPI/MibSPI pins.

Sl No.	Delay Item	Master	Slave	Note
12	Hold time requirement on SPISOMI Input w.r.t SPICLK	0	-	The incoming data on SPISOMI pin is sampled w.r.t SPICLK itself
13	Hold time requirement on SPISIMO Input w.r.t SPICLK	-	0	The incoming data on SPISIMO pin is sampled w.r.t SPICLK itself
14	Maximum hold time on SPIENA after the final SPICLK to pin supported by Master (Phase = ‘0’)	0.5 SPICLK + 1VCLK	-	After the last edge of SPICLK, the maximum delay within which SPIENA pin should be deasserted so that the Master does not start the next buffer transfer. SPIENA should be deasserted before this time.
15	Maximum hold time on SPIENA after the final SPICLK to pin supported by Master (Phase = ‘1’)	1VCLK	-	After the last edge of SPICLK, the maximum delay within which nENA pin should be deasserted so that the Master does not start the next buffer transfer. SPIENA should be deasserted before this time.
16	Minimum delay between de-selecting the Slave using its SPISCS pin and clocking a different Slave	-	1VCLK	A SPICLK edge within 1VCLK of deassertion of SPISCS will be treated a valid edge in Slave mode.
17	Minimum inactive time of SPISCS pin(s) between two data words	2VCLK	-	This can be increased by using WDELAY counter.
18	Time at which SPIENA pin will be sampled after SPISCS goes active	C2TDELAY	-	If C2TDELAY is not programmed or ‘0’, then the FSM will look for the ENA pin value 1VCLK after the asserting the ChipSelect. The FSM always looks at the registered version of ENA pin value.
The ChipSelect to SPICLK or SPICLK to ChipSelect timings do not include C2T/T2C delay timer.				

All the numbers are logical delays. Actual delays between the pins will depend upon the differences in I/O buffer delays of each of the protocol pins. For example in a MASTER mode SPI, if the O/P buffer on SPICLK pin is of 4mA drive strength and the buffer on SPISIMO is of 2mA strength, then depending upon the loads on these pins, the delay difference between SPICLK & SPISIMO pins could vary between 5ns and 15ns or even more.

Chapter 15

SPNU489C - February 2012

Analog To Digital Converter (ADC) Module

15.1 Overview

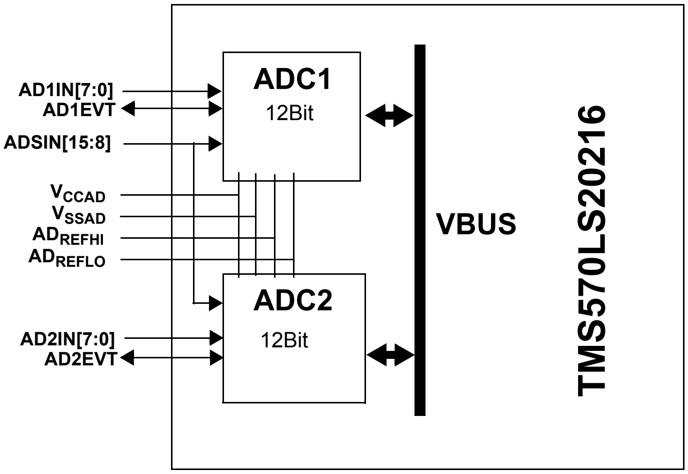
The main features of the 12-bit ADC implemented in this device include:

- 12-bit resolution
- Successive-approximation-register (SAR) architecture
- Three conversion groups: Group1, Group2 and Event Group
- Software and Hardware triggered Event Groups
- 64 word ADC memory/FIFO
  - Size of memory regions for each conversion group is programmable
  - Provides parity protection
- Single and Continuous conversion modes
- Embedded self-test and calibration logic
- External event pin (ADEVT) programmable as general-purpose I/O

Two 12-bit ADC cores and 24 ADC input channels are implemented in the TMS570LS20216 device. Figure 15-1 illustrates the connection of the two A/D converter peripherals on the TMS570LS20216 device.

- Each ADC supports 16 channels.
- Each ADC has 8 dedicated channels.
- Two ADCs core share 8 channels.
- The references are shared between the two cores.

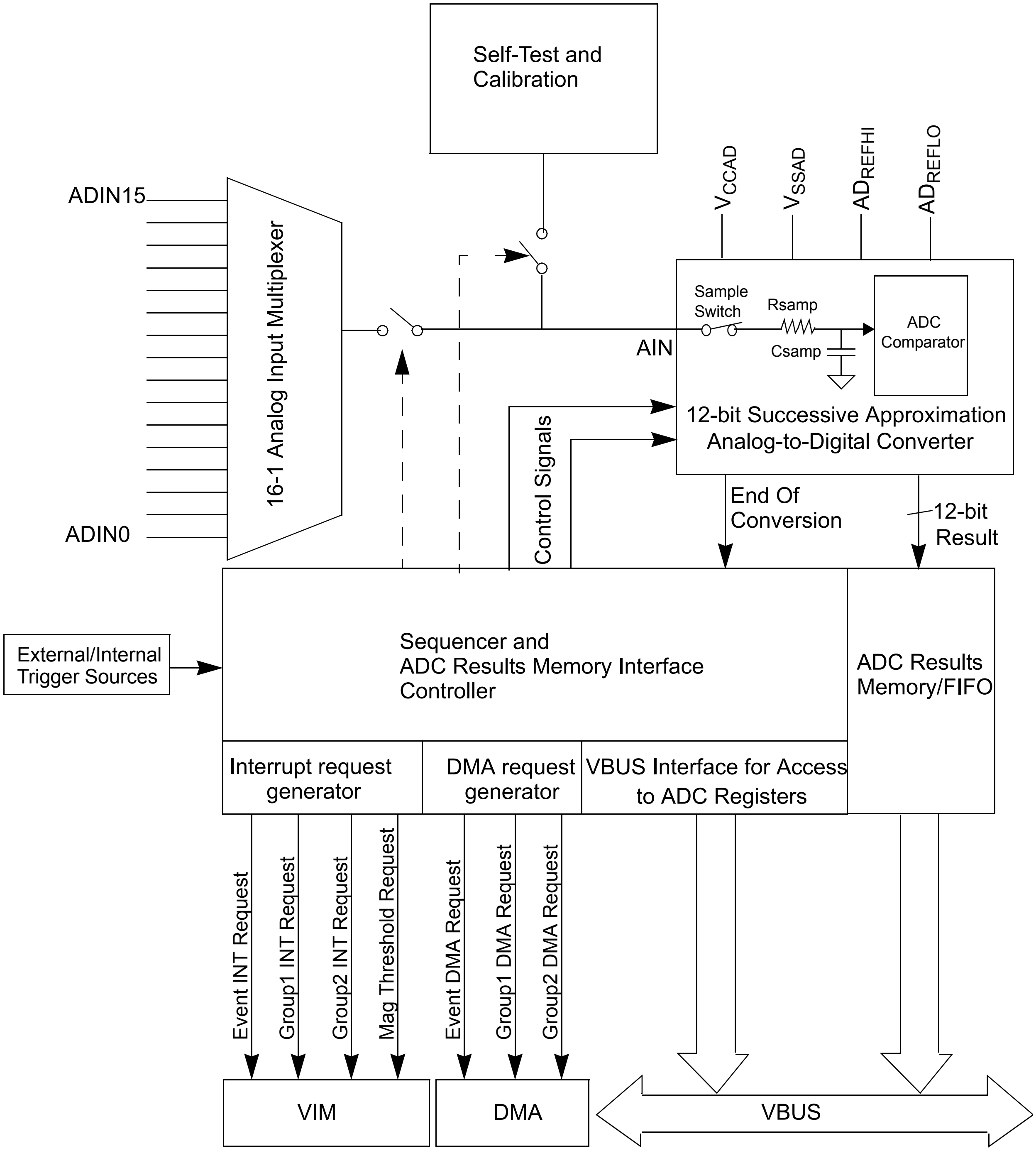
Figure 15-1. Channel Assignments of Two ADC cores



15.2 Introduction

This section presents a brief functional description of the analog-to-digital converter (ADC) module. Figure 15-2 illustrates the components of the ADC module.

Figure 15-2. ADC Block Diagram



15.2.1 Analog Input Multiplexer

The analog input multiplexer (MUX) connects the selected input channel to the AIN input of the ADC core. The ADC module supports up to 16 inputs as shown in Figure 15-2 . The sequencer selects the channel to be converted.

15.2.2 Self-Test and Calibration

The ADC includes specific hardware for detecting open/shorts on an ADC analog input pin. It also allows the application program to calibrate the ADC. The self-test mode and the calibration modes are controlled by the sequencer. For more details, see the Section 15.7 and Section 15.8.2.

15.2.3 Analog-to-Digital Converter Core

The analog conversion range is determined by the reference voltages: AD_{REFHI} and AD_{REFLO}. Both AD_{REFHI} and AD_{REFLO} must be chosen not to exceed the analog power supplies: V_{CCAD} and V_{SSAD}. Input voltages between AD_{REFHI} and AD_{REFLO} produce a conversion result given by (EQ 1):

$$DiDigitalResult\frac{4096x(InputVoltage - ADREFLO)}{(ADREFHI - ADREFLO)} \qquad \text{(EQ 1)}$$

15.2.4 Sequencer

The sequencer coordinates the operations of the ADC, including the analog input multiplexer, 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.

The logic handles CPU read/write operations, interrupts, halts, resets, memory allocation and handling. The logic in this block also supports self-test mode and calibration operations. This logic block generates the internal ADC clock that controls the ADC timing.

All the features of the sequencer are discussed in detail in the following sections of this document.

15.2.5 Conversion groups

The ADC module can service one of three conversion groups at any time: Group1, Group2, and the Event Group. The Group1 and Group2 are software-triggered by default and can be configured as hardware triggered (ADG1MODECR.3, ADG2MODECR.3) while the Event Group is hardware triggered only. Each conversion group has a separated set of registers to:

- Select the input channel,
- Configure in single or continuous conversion mode,
- Set acquisition time,
- Read conversion data,
- Handle the group conversion end interrupt, group memory threshold interrupt and group memory overrun interrupt.

15.3 Basic Features and Usage of the ADC

Figure 15-3 - Figure 15-5 shows how to start AD conversions after reset. Figure 15-3 describes how to initialize the ADC module; Figure 15-4 describes how to start/trigger single conversion; And Figure 15-5 describes how to start/trigger continuous conversion. The function in these flow charts will be described in details in following sub-sections.

Figure 15-3. ADC Initialization Flow Chart

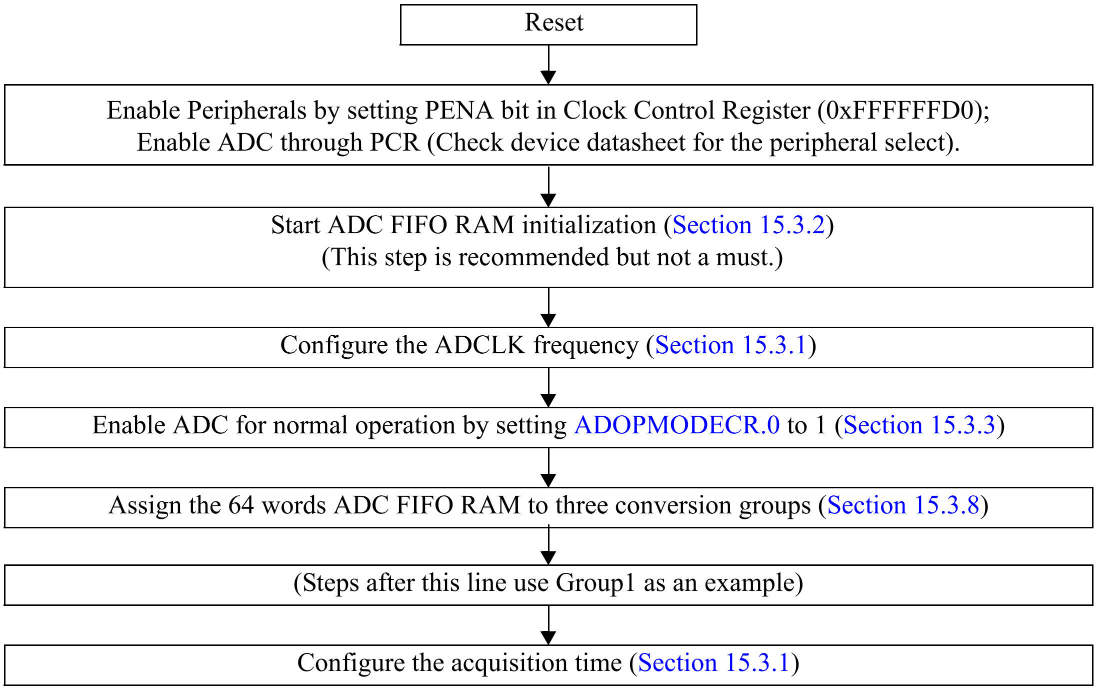
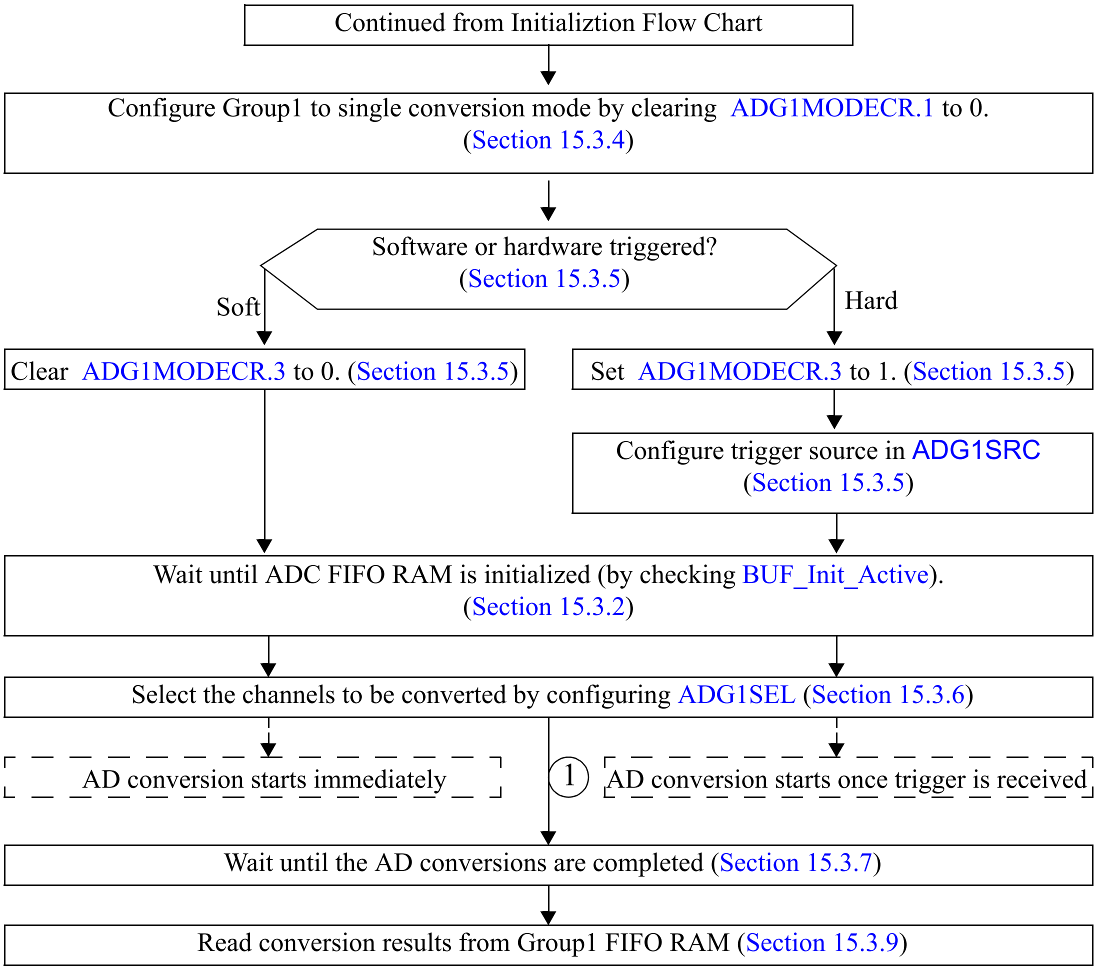
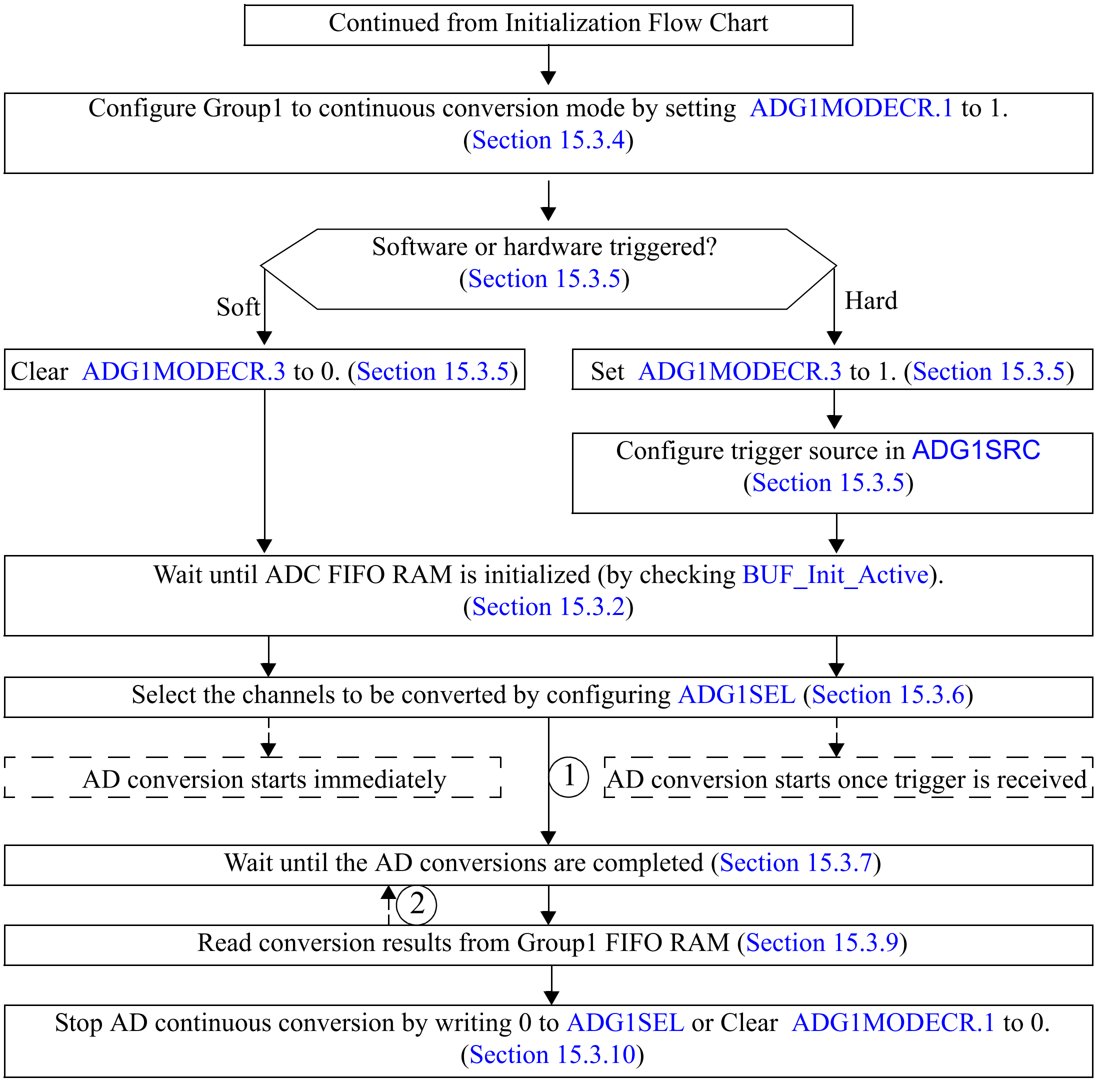

Reset
Enable Peripherals by setting PENA bit in Clock Control Register (0xFFFFFFD0);
Enable ADC through PCR (Check device datasheet for the peripheral select).
Start ADC FIFO RAM initialization (Section 15.3.2)
(This step is recommended but not a must.)
Configure the ADCLK frequency (Section 15.3.1)
Enable ADC for normal operation by setting ADOPMODECR.0 to 1 (Section 15.3.3)
Assign the 64 words ADC FIFO RAM to three conversion groups (Section 15.3.8)
(Steps after this line use Group1 as an example)
Configure the acquisition time (Section 15.3.1)

Figure 15-4. Flow Chart for Single Conversion


Continued from Initializtion Flow Chart
Configure Group1 to single conversion mode by clearing ADG1MODECR.1 to 0. (Section 15.3.4)
Software or hardware triggered? (Section 15.3.5)
Soft
Hard
Clear ADG1MODECR.3 to 0. (Section 15.3.5)
Set ADG1MODECR.3 to 1. (Section 15.3.5)
Configure trigger source in ADG1SRC (Section 15.3.5)
Wait until ADC FIFO RAM is initialized (by checking BUF_Init_Active). (Section 15.3.2)
Select the channels to be converted by configuring ADG1SEL (Section 15.3.6)
AD conversion starts immediately
1
AD conversion starts once trigger is received
Wait until the AD conversions are completed (Section 15.3.7)
Read conversion results from Group1 FIFO RAM (Section 15.3.9)

1. If hardware triggered, ADC core must wait for the trigger signal before conversion starts.

Figure 15-5. Flow Chart for Continuous Conversion


Continued from Initialization Flow Chart
Configure Group1 to continuous conversion mode by setting ADG1MODECR.1 to 1. (Section 15.3.4)
Software or hardware triggered? (Section 15.3.5)
Soft
Hard
Clear ADG1MODECR.3 to 0. (Section 15.3.5)
Set ADG1MODECR.3 to 1. (Section 15.3.5)
Configure trigger source in ADG1SRC (Section 15.3.5)
Wait until ADC FIFO RAM is initialized (by checking BUF_Init_Active). (Section 15.3.2)
Select the channels to be converted by configuring ADG1SEL (Section 15.3.6)
AD conversion starts immediately
1
AD conversion starts once trigger is received
Wait until the AD conversions are completed (Section 15.3.7)
2
Read conversion results from Group1 FIFO RAM (Section 15.3.9)
Stop AD continuous conversion by writing 0 to ADG1SEL or Clear ADG1MODECR.1 to 0. (Section 15.3.10)

1. If hardware triggered, ADC core must wait for the trigger signal before conversion starts.
2. In continuous conversion mode, ADC will keep AD converting, writing FIFO and set ADG1SR.0.

15.3.1 How to setup the ADCLK speed and the acquisition time?

The clock for the ADC core is ADCLK. 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 in the ADCLOCKCR, bits [4:0], is used to divide down the VCLK by 1 up to 32. The ADCLK frequency range is specified in device datasheet.

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 4097 ADCLK cycles.

For example, Group1 acquisition time, t_{ACQG1} = ADG1SAMP[11:0] + 2, in ADCLK cycles.

15.3.2 How to initialize ADC Results FIFO RAM?

The ADC module allows the application to auto-initialize the ADC results FIFO RAM to all zeros and fill in the parity bits. To avoid parity errors when reading ADC RAM location without AD Conversion data, it is recommended to initialize the ADC RAM. 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 1010b to the bits [3:0] of the MINITGCR register of the System module.
2. Set the control bit for the ADC results FIFO RAM in the MSIENA 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 ADC results FIFO 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.
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.

15.3.3 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).

15.3.4 How to configure single or continuous modes?

ADG1MODECR.1, ADG2MODECR.1, and ADEVMODECR.1 are used to select between either single or continuous conversion mode for each of the three groups.

15.3.5 How to configure software or hardware trigger?

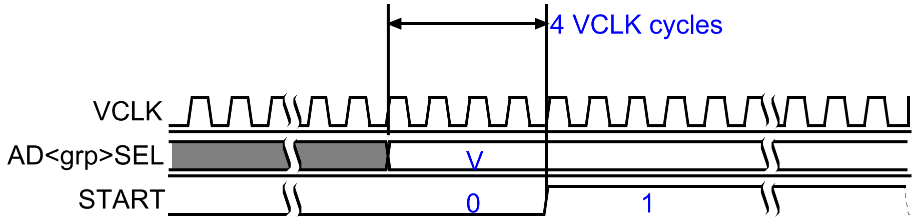
There are two trigger modes, software-triggered and hardware event-triggered. If a conversion group is configured to be software-triggered, writing the desired channels to the respective Channel-Select Registers will start the AD conversion. If a conversion group is configured to be hardware triggered, writing the Channel-Select Registers only select the desired channels to be converted. To start the conversion, the application must provide a trigger event, for example, a rising edge of the ADEVT pin.

The Event Group is hardware triggered only. The Group1 and Group2 operating mode control registers (ADG1MODECR.1 and ADG2MODECR.1) have an extra control bit: HW_TRIG. Setting this bit to one configures the group to be hardware event-triggered while clearing this bit to zero configures the group to be software-triggered, which is the default.

When a group is configured to be hardware triggered, the trigger source is defined for each group in the ADEVSRC, ADG1SRC and the ADG2SRC registers.

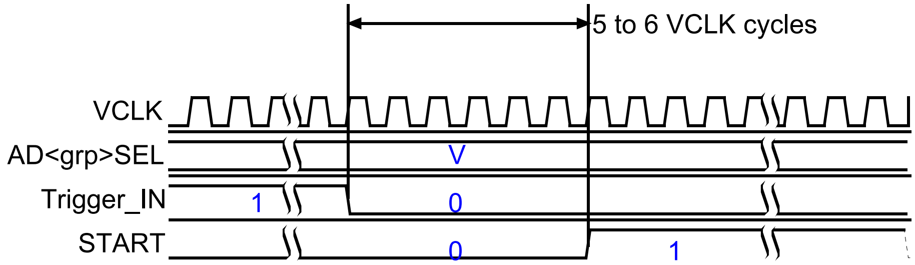
For Software trigger, it takes 4 vclk cycles from trigger to assertion of START pulse (start sampling).

Figure 15-6. Timing Diagram of Software Trigger



For HW trigger, it takes 5-6 vclk cycles from trigger to assertion of START pulse. The delay through synchronizer may introduce another 1 or 2 cycles uncertainty

Figure 15-7. Timing Diagram of Hardware Trigger



15.3.6 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 hardware-triggered only. There are up to eight hardware event trigger sources defined for the ADC module. 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 through ADEVSRC register, 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.

15.3.7 How to know the group conversion is completed?

Each conversion group has a ‘conversions ended’ bit (ADEVSR.0, ADG1SR.0, ADG2SR.0). This bit will be set when all the channels selected in the group has completed. In continuous conversion mode, since the selected channels are served again and again, this bit will be set again and again.

15.3.8 How are results stored in the results memory?

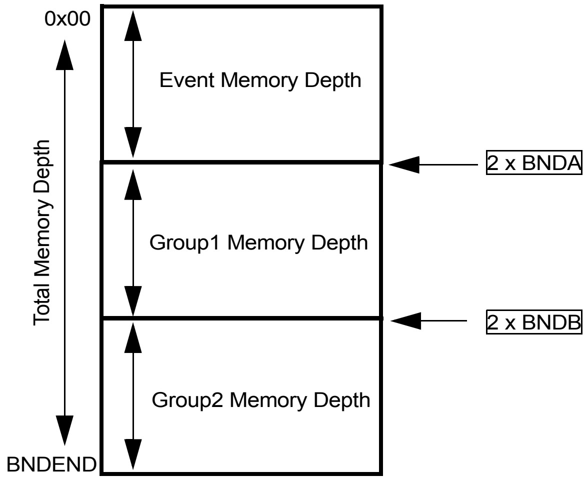
The ADC stores the conversion results in three separate FIFOs in the ADC results FIFO RAM, one FIFO for each group. ADBNDCR contains two 9-bit pointers BNDA and BNDB. They are used to partition the total memory available into three memory regions as shown in Figure 15-8. ADBNDEND contains a 3-bit field called BNDEND that configures the total memory available.

- total FIFO size is 64 32-bit words.

The user must program BNDEND as 010b (64 words) or less. A reason you would do 'less' is for compatibility with another chip that has less ADC RAM.

- three FIFOs are allocated from the 64 words
- size of EVENT group FIFO is 2 x BNDA words
- size of GROUP1 FIFO is 2 x (BNDB - BNDA) words
- size of GROUP2 FIFO is 2^{BNDEND} x 16 - 2 x BNDB words

Figure 15-8. FIFO Memory Partitioning between Conversion Groups



15.3.9 How to read the results from the results memory?

The CPU can read the conversion results in one of two ways:

a) By reading the group’s conversion results from a FIFO queue (ADEVBUFFER, ADG1BUFFER, ADG2BUFFER), or

b) By reading the group’s conversion results directly by accessing the correct ADC RAM location.

15.3.9.1 Reading conversion results from the FIFO

The conversion results for each group can be accessed via a range of addresses provided to facilitate the use of the ARM Load-Multiple (LDM) instruction. A single read performed using the LDR instruction can 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 between offset 0x90 and 0xAF (ADEVBUFFER) pulls out one conversion result from the Event Group memory.

Figure 15-9. Format of conversion result read from FIFO, 12-bit ADC

0x90 to 0xAF[1] ADEVBUFFER Page 813	EV_E MPTY	Reserved	EV_CHID
	Reserved	EV_DR	
0xB0 to 0xCF[2] ADG1BUFFER Page 814	G1_E MPTY	Reserved	G2_CHID
	Reserved	G1_DR	
0xD0 to 0xEF[3] ADG2BUFFER Page 815	G2_E MPTY	Reserved	G2_CHID
	Reserved	G2_DR	

(1) Reading any address within this range accesses the EVENT Group FIFO.
(2) Reading any address within this range accesses the Group1 FIFO.
(3) Reading any address within this range accesses the Group2 FIFO.

Also, for debug purposes, each buffer has an emulation address (ADEVEMUBUFFER, ADG1EMUBUFFER, ADG2EMUBUFFER) that returns the next conversion result from the buffer without removing the result from the buffer itself. For example, reading from offset 0xF0 (ADEVEMUBUFFER) 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.

15.3.9.2 Reading conversion results from the RAM

Figure 15-10 shows the direct mapped view of the ADC result memory. This view may be used instead of the FIFO view if a direct mapped view suits the application better. The table base address is 0xFF3E0000 for ADC1 and 0xFF3A0000 for ADC2. Each ADC has 64 32-bit words result memory.

Figure 15-10. ADC Memory Mapping

ADC1	ADC2	
0xFF3E0000	0xFF3A0000	Conversion word 0
0xFF3E0004	0xFF3A0004	Conversion word 1
0xFF3E0008	0xFF3A0008	Conversion word 2
⋮	⋮	⋮
0xFF3E01F8	0xFF3A01F8	Conversion word 62
0xFF3E00FC	0xFF3A00FC	Conversion word 63

The format of the each conversion word is shown in Figure 15-11. ‘Empty’ flag does not exists in the RAM but it does exist during FIFO read.

Figure 15-11. Format of ADC RAM, 12-bit ADC

<table>
<tr><td rowspan="2">ADC RAM address</td><td colspan="2">Reserved</td><td>chan-nel id [4]</td></tr>
<tr><td>channel id [3:0]</td><td colspan="2">12-bit conversion result</td></tr>
</table>

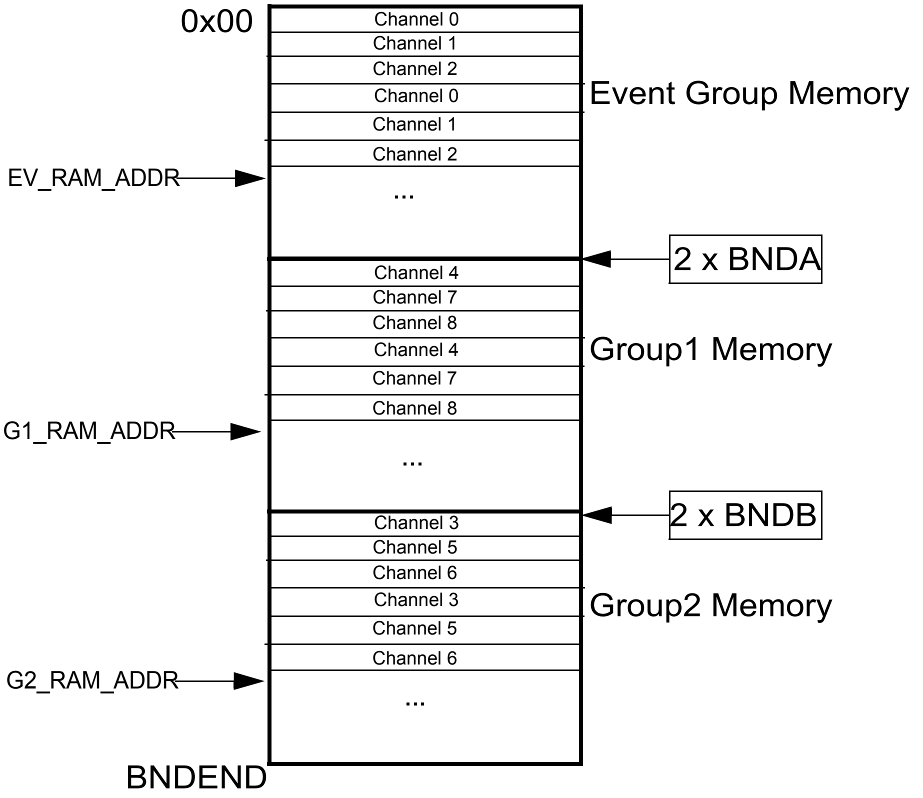
To read conversion results from the RAM directly, the application needs to identify the address ranges for each of the three memory regions for the three conversion groups after performing the segmentation as described in Section 15.3.8. It is up to the application to read the desired results from the three conversion groups. A separate register for each group (ADEVRAMWRADDR, ADG1RAMWRADDR and ADG2RAMWRADDR) holds the ADC buffer index for that group where the ADC will write the next conversion result. For example, if ADG1RAMWRADDR is 0x21, the ADC will write the next conversion result to buffer 33, with the address of (RAM Base Address + 0x21 x 4). The application can therefore calculate how many valid conversion results are available to be read. This mode also allows the application to selectively read the conversion results for any particular input channel of interest without having to read other channels' conversion results first. The format of the result read directly from RAM is slightly different (no EMPTY bit) from that read from the FIFO interface as shown below.

Please refer to Section 15.3.2 for information about initializing the ADC memory. Please refer to Section 15.8.1 for information about parity check of the ADC memory.

15.3.9.3 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 Group1, and channels 3, 5, and 6 are selected for conversion in Group2. The conversion results will get stored in the three memory regions as shown below:

Figure 15-12. Conversion results storage



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 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 Group1 from the RAM directly. The conversion results for the Group1 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.

15.3.10 How to stop a conversion?

A group’s conversion can be stopped by clearing the group’s channel select register. Writing a non-zero value to the group’s channel select register resets the group’s results FIFO pointer.

In continuous conversion mode, clearing EV_MODE, G1_MODE, and G2_MODE bits will force the corresponding conversion group into single conversion mode. The conversion will stop when all the channels selected in the group has completed.

15.3.11 List of ‘Do NOT’

Do not write to the control registers especially the input-select registers (ADEVSEL, ADG1SEL, ADG2SEL) if there are ongoing AD conversions and the conversion result is needed by the application.

Do not trigger off the ADC RAM auto-initialization process if the ADC module is any of the conversion modes.

Do not write a new conversion result to the results RAM if the ADC Results RAM Test Mode is enabled.

Do not modify the SELF_TEST bit (ADCALCR.24) to enable or disable the self-test mode if there are ongoing AD conversions.

Do not modify the HILO bit (ADCALCR.8) to change the test voltage source if there are ongoing AD conversions.

15.4 Advanced Conversion Group Configuration Options

15.4.1 Single or Continuous Conversion Modes

The EV_MODE, G1_MODE, and G2_MODE bits in the group's operating mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) are used to select between either single or continuous conversion mode for each of the three groups.

15.4.1.1 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, Group1 and 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 (ADEVSR, ADG1SR, ADG2SR). 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.

15.4.1.2 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.

15.4.2 Conversion Group Freeze Capability

Each conversion group can be configured to allow its conversions to be frozen whenever there is a request for conversion in another 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 new request will start after the conversion of the current channel of the active group is complete. 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.

15.4.3 Conversion Group Priority

There's an inherent priority between the conversions groups (EV, G1, G2) - Event Group being the highest priority and Group2 being the lowest priority group. Similarly the channel0 has the highest priority and channel 31 has the lowest priority within a group and it's limited to the group. So, if two groups are triggered at the same time, then the higher priority group will start first, complete and then the next priority group will be addressed.

By default (with FREEZE feature disabled), while a group conversion is underway, no other groups can stop/ break the conversions until all the channels in that group are completed even if there are higher priority groups pending.

With "FREEZE" feature enabled for a conversion group, it can be interrupted by a new request on any other conversion groups. By using the FREEZE bit appropriately between the 3 groups, it's possible to maintain priority order of group conversions.

15.4.4 8-bit, 10-bit or 12-bit Result Mode

Some applications can use only the ADC conversion results as 8-bit or 10-bit. The ADC module can automatically shift a group's conversion results right as the result is being read from the FIFO (Section 15.3.9.1). The conversion results read directly from memory is not impacted. This eliminates the need for the application program to shift each conversion result itself, and results in more efficient code execution.

The 10-bit result mode is enabled by setting the G1_DATA_FMT, G2_DATA_FMT, and EV_DATA_FMT in the group's operating mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) control bits to "01". The 8-bit result mode is enabled by setting the G1_DATA_FMT, G2_DATA_FMT, and EV_DATA_FMT control bits to "10".

15.4.5 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 **all** the group's results memory upon an overrun condition. Only then can the ADC continue to write new results to the memory. Otherwise, the latest conversion data will be discarded.

15.4.6 Group Channel Id Storage Option

The channel number is always stored in the FIFO RAM along with the conversion result. But the channel number field can be masked out during reads from the FIFO if CHID bit in the group's operating mode control register (ADEVMODECR, ADG1MODECR, ADG2MODECR) is set.

If the CHID bit is not set, the bits [20:16] are forced to 00000 when the conversion results are read out in FIFO mode (ADEVBUFFER, ADG1BUFFER, ADG2BUFFER) from the group's results memory.

If the CHID bit is set, the bits [20:16] in the group's results memory contain the input channel number will be read out in FIFO mode.

15.4.7 Group Trigger Options

The Group1 and Group2 operating mode control registers (ADG1MODECR, ADG2MODECR) 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.

15.5 ADC Module Interrupts

This section describes the interrupts generated by the ADC module. Each conversion group is capable of three interrupts: Group Conversion End Interrupt, Group Memory Threshold Interrupt and Group Memory Overrun Interrupt. The ADC module also has the capability to generate ADC Magnitude Threshold Interrupts.

15.5.1 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 in the corresponding Group Interrupt Enable Control Register(ADEVINTENA, ADG1INTENA, ADG2INTENA).

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. In the continuous - conversion mode, this interrupt will be triggered again and again since the group's END bit will be set repeatedly.

15.5.2 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 (ADMAGINTxCR) 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 threshold counter is a 16 bit signed integer, with legal values between -256 and +255. Programmer should initialize the threshold counter to the number of results that the FIFO should hold before interrupting the CPU. The threshold register value tracks the level of data within the FIFO. This counter decrements whenever a new result is stored in the FIFO, and increments when a value is read from the FIFO. 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.

15.5.3 Group Memory Overrun Interrupt

Whenever the ADC produces a new result but there is no free storage in the corresponding group memory, an overrun occurs. An overrun interrupt is available for each group. In case of an overrun, the application should follow Section 15.4.5 to configure the ADC.

15.5.4 ADC Magnitude Threshold Interrupts

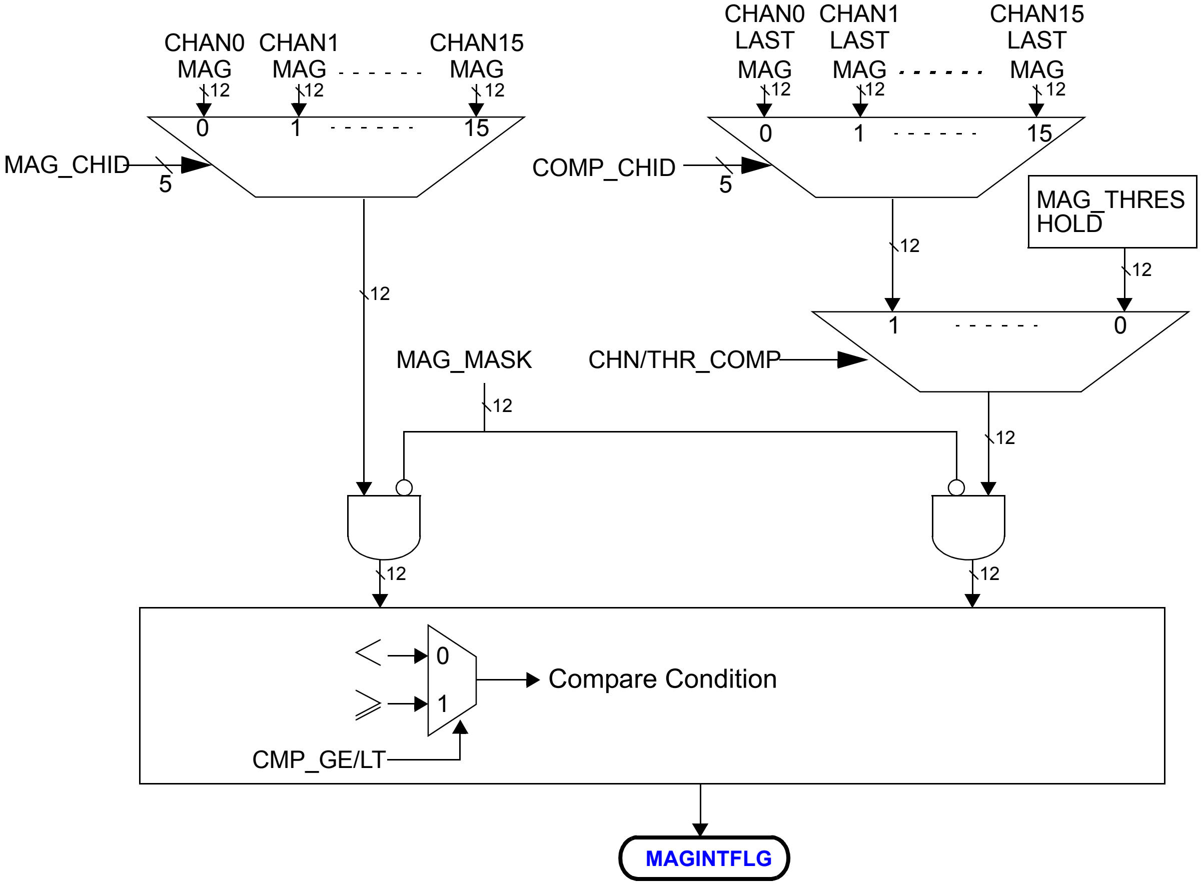
The ADC allows a magnitude threshold interrupt to be generated if the conversion result from a certain channel reaches a predetermined threshold. The predetermined threshold can be a fixed digital value or the last conversion result of channel COMP_CHIDx[4:0]. The interrupt condition can be "greater than or equal to" or "less than". The magnitude comparison can be performed on up to three channels. The comparison parameters are programmed via the Magnitude Threshold Control Register (ADMAGINTxCR). If the conversion result for the selected channel meets the condition programmed, an interrupt is generated.

15.5.4.1 Magnitude Threshold Interrupt Configuration

Figure 15-13 shows how the magnitude threshold interrupts are configured. The following control 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.
2. MAG_CHID: Specifies the channel number from 0 to 15 whose conversion result needs to be monitored.
3. COMP_CHID: Specifies the channel number from 0 to 15 whose last conversion result is used for the comparison with the conversion result of the channel being monitored.
4. MAG_THRESHOLD: 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_THRESHOLD field.
6. MAG_MASK: Specifies the mask for the comparison. The ADC module will mask the corresponding bit in the register ADMAGxMASK for the comparison.
7. MAG_INT_FLG: Specifies whether the magnitude comparison condition is true or false.

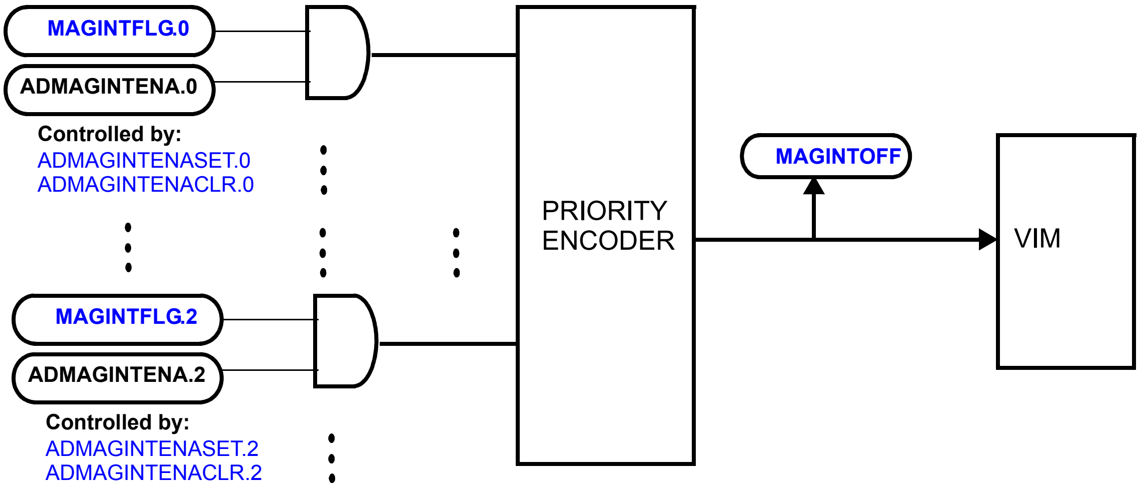
Figure 15-13. Magnitude Threshold Interrupt Configuration



15.5.4.2 *Magnitude Threshold Interrupt Generation*

Figure 15-14 shows how the magnitude threshold interrupt is generated and sent to VIM (Vector Interrupt Module).

Figure 15-14. Magnitude Threshold Interrupt Generation



Each of the three magnitude interrupts also have separate interrupt enable set and clear 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.

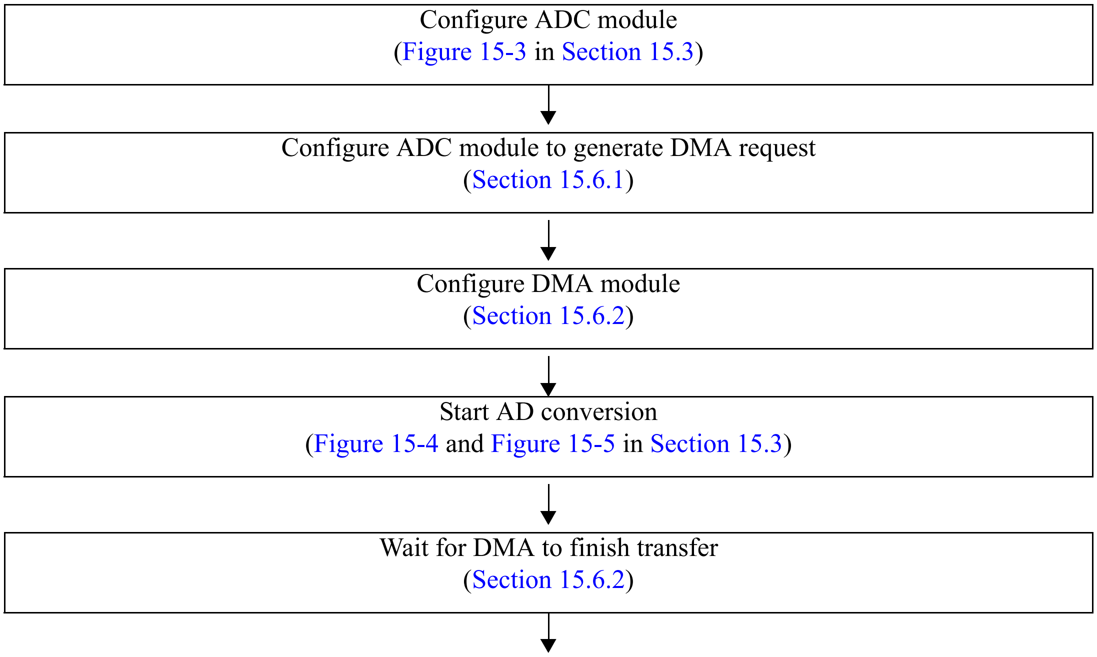
It is possible to have multiple magnitude threshold interrupts pending at the same time. The magnitude threshold interrupt offset register holds the index of the currently pending highest priority magnitude threshold interrupt. The magnitude threshold interrupt 1 has the highest priority while the magnitude 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.

15.6 Servicing the ADC by DMA

The ADC module is capable of requesting DMA transfer. Figure 15-15 describes how to configure the ADC and DMA to transfer the ADC results and the following section explains Figure 15-15 in details.

Figure 15-15. Flow Chart for Servicing the ADC by DMA


Configure ADC module
(Figure 15-3 in Section 15.3)
Configure ADC module to generate DMA request
(Section 15.6.1)
Configure DMA module
(Section 15.6.2)
Start AD conversion
(Figure 15-4 and Figure 15-5 in Section 15.3)
Wait for DMA to finish transfer
(Section 15.6.2)

15.6.1 Configure ADC Module to Generate DMA Request

Two type of DMA requests can be generated for request mode transfers.

- DMA transfer request whenever the ADC module writes the conversion result to ADC RAM.

  The DMA request is generated after the group's memory region leaves the EMPTY state and contains data. Subsequently, when the region contains data, a new DMA request is generated after the DMA reads a data word out of the region, if this read does not cause the memory region to be empty. The DMA request occurs on the very next cycle after the read from the memory region.

  This mode can be enabled by setting the DMA Transfer Enable bit (bit 0) in the respective group's DMA control register. The Gx_BLK_XFER field should be set to 0.

- DMA transfer request when the ADC has written GxBLOCKS number of buffers into that group memory. To enable this mode, the application must:
  - Set Gx_BLK_XFER field in the respective group's DMA control register to 1, AND,
  - Write a non-zero value to GxBLOCKS field in the respective group's DMA control register.

    The GxBLOCKS field specifies the number of conversion results written in the region before a DMA request is generated. Read value of this field shows the current value of the Threshold Counter. This is the down counter shared by the Threshold Counter Interrupt logic. It is expected that the DMA controller is capable of reading out the GxBLOCKS number of data from the FIFO before another Block Count of ADC conversions complete. For example, if the GxBLOCKS count is configured for 10, then ADC module will generate a DMA request at the end of the 10th conversion.

15.6.2 Configure DMA Module

The following sequence can be used to configure the DMA module to respond to the ADC request. The application can choose different ways to read AD conversion results by DMA. Please refer to the DMA module user guide for details.

Note:
All the registers mentioned in this section are DMA registers.

1. Write ‘1’ to bit GCTRL.0 to reset DMA.
2. Initialize DMA control packet RAM.
3. Write ‘1’ to bit GCTRL.16 to enable DMA.
4. Set the parameters for DMA control packet.
   a. Set Initial Source Address in the ISADDR register as the AD Group FIFO address.
   b. Set Initial Destination Address in IDADDR register as the destination address of the DMA transfer.
   c. Define Initial frame transfer count and Initial element transfer count in the ITCOUNT register. Table 15-1 shows an example of setting ITCOUNT register.

Table 15-1. Example of Setting ITCOUNT register

	DMA transfer request whenever the ADC module writes the conversion result to ADC RAM	DMA transfer request when the ADC has written GxBLOCKS number of buffers into that group memory
Initial frame transfer count	The total AD channel numbers in that AD group channel select register	1
Initial element transfer count	1	The total AD channel numbers in that AD group channel select register

   d. Define the read element size, write element size, transfer type, addressing mode read, addressing mode write, Auto initiation and next channel to be triggered in the CHCTRL register. Table 15-2 shows an example of setting CHCTRL register.

Table 15-2. Example of Setting CHCTRL register

	DMA transfer request whenever the ADC module writes the conversion result to ADC RAM	DMA transfer request when the ADC has written GxBLOCKS number of buffers into that group memory
Read element size	32-bit	32-bit
Write element size	32-bit	32-bit
Transfer type	Frame type	Block type
Addressing mode read	Constant	Constant
Addressing mode write	Post-increment	Post-increment
Auto initiation	On	Off
Next channel to be triggered	Non	Non

   e. Set Frame Index Offset Register (FIOFF) and Element Index Offset Register (EIOFF).

      These parameters depend on the Mode Increment of Read and Write.
   f. Set Port Assignment Register (PAR) to assign a port to the selected channel.
5. Set the DREQASI register to assign the selected DMA channel for the ADC DMA request line.

   Please check the device datasheet for the ADC DMA request line number.
6. Set the corresponding hardware enable bit in the HWCHENAS register.

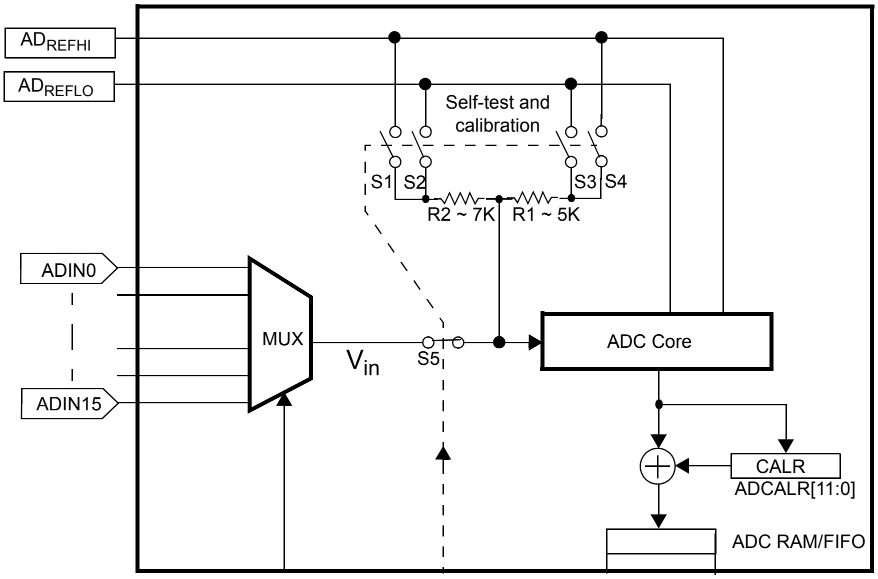
After the DMA transfer is completed, the corresponding Block Transfer Complete flag (in the BTCFLAG register) will be set. The application can poll this bit to check the status of DMA transfer.

15.7 ADC Error Calibration

The ADC Error Calibration is used to calculate the offset error and provide a compensation value for normal ADC conversions. 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 RAM.

Figure 15-16 shows the self-test and calibration logic embedded in this device. In normal conversion mode, S5 is closed while S1-S4 are opened. In calibration mode, S5 is opened and the offset error is calculated based on the conversion of the embedded calibration reference voltage.

Figure 15-16. Self-Test and Calibration Logic



15.7.1 Calibration and Offset Error Correction Procedure

The basic calibration routine is as follows:

1. Enable calibration via CAL_EN (ADCALCR.0).

   User must make sure the self-test mode (SELF_TEST, ADCALCR.24) is disabled and no conversion group is being serviced when the calibration mode is enabled.

2. Select the voltage source via BRIDGE_EN and HILO (ADCALCR[9:8]).

   These two bits control the voltage to the calibration reference device shown in Figure 15-16. The positions of the switches in calibration mode are listed in Table 15-3.

Table 15-3. Calibration Reference Voltages†

CAL_EN	BRIDGE_EN	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	0	1	0	0	$(AD_{REFHI}*R1 + AD_{REFLO}*R2) / (R1 + R2)$
1	0	1	0	1	0	1	0	$(AD_{REFLO}*R1 + AD_{REFHI}*R2) / (R1 + R2)$
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}

† The state of the switches in this table assumes that self-test mode is not enabled.

3. Start the conversion by writing a ‘1’ to CAL_ST.

   When CAL_ST (ADCALCR.16) is set, a calibration conversion is started. The voltage source selected via the bits BRIDGE_EN and HILO is converted once (single conversion mode).

   Before starting the conversion, the user must make sure the calibration conversion meets the minimum sampling time specification for the ADC. Please see Section 15.3.1 for details.
4. Wait for CAL_ST to go to 0.
5. Get the results from ADCALR and save to memory.

   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.
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 in the 2’s complement form with the computed error correction value.

   For example, assume the [D(cal1)+D(cal2)] /2=2048.5 during the Mid-Point Calibration (Section 15.7.2), the ADCALR register should be set to 0xFFF to compensate the offset error.
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.

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.

15.7.2 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%).

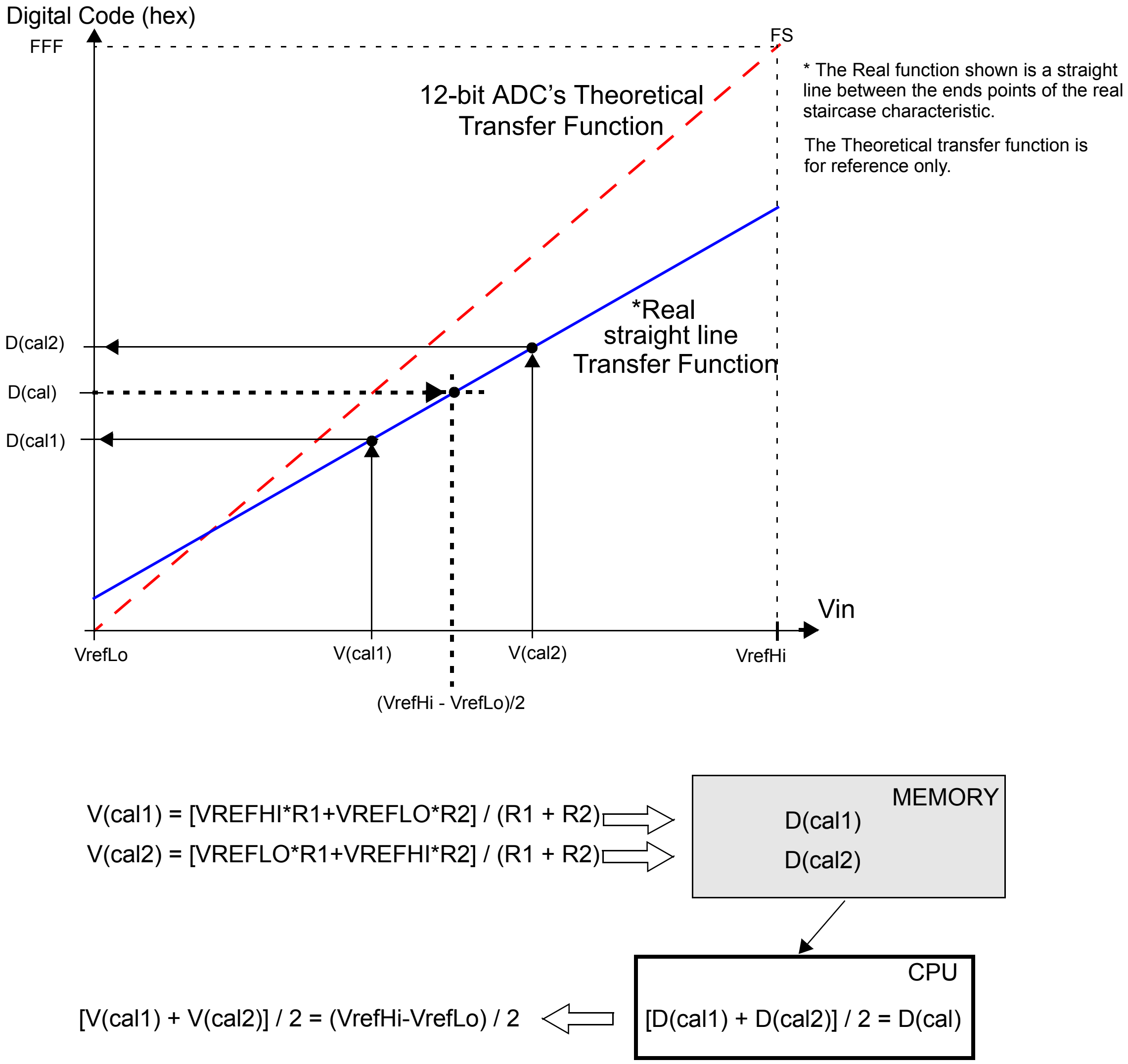
The switched reference voltage device has been specially designed to support a differential measurement of its mid-point voltage. This ensures the accuracy of the mid-point reference, and hence the efficiency of the calibration.

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 15-17 summarizes the mid-point calibration flow.

Figure 15-17. Mid-point value calculation


Digital Code (hex)
FFF
FS
12-bit ADC's Theoretical Transfer Function
* The Real function shown is a straight line between the ends points of the real staircase characteristic.
The Theoretical transfer function is for reference only.
*Real straight line Transfer Function
D(cal2)
D(cal)
D(cal1)
Vin
VrefLo
V(cal1)
V(cal2)
VrefHi
(VrefHi - VrefLo)/2
V(cal1) = [VREFHI*R1+VREFLO*R2] / (R1 + R2)
V(cal2) = [VREFLO*R1+VREFHI*R2] / (R1 + R2)
MEMORY
D(cal1)
D(cal2)
CPU
[D(cal1) + D(cal2)] / 2 = D(cal)
[V(cal1) + V(cal2)] / 2 = (VrefHi-VrefLo) / 2

15.8 ADC Built In Diagnostics and Self Test Logic

15.8.1 ADC RAM Parity and Test

For safety reasons, the ADC RAM has protection by parity to prevent corruption by soft error. The parity scheme is implemented as a continuous background check based on memory access. Section 15.8.1.1 and Section 15.8.1.2 describe how parity works and how to check the scheme of parity.

15.8.1.1 ADC results FIFO RAM Parity

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 ADC RAM parity is controlled by the DEVCR1 register in the system module (address 0xFFFFFFDC). The parity enable is controlled by the ADPARCR register. After reset, the parity is disabled. The application should enable parity check if parity protection is needed.

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 and signals it to the ESM module. 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.

15.8.1.2 Testing the Parity Checking Mechanism

To test the parity checking mechanism itself, the parity RAM has to be made accessible in order to allow manually inserting parity errors. In the defined conversion modes of the ADC, only the ADC module is allowed to write to the results FIFO RAM. A special test mode, ADC results FIFO RAM Test Mode, is defined to allow the application to also write into the ADC results FIFO RAM. Only 32-bit reads and writes are allowed to the ADC results FIFO RAM in this test mode.

Note: Contention on access to ADC results FIFO RAM
The application must ensure that the ADC is not likely to write a new conversion result to the results FIFO RAM when the ADC results FIFO RAM Test Mode is enabled.

The ADC results FIFO RAM Test Mode is enabled by setting the RAM_TEST_EN bit in the ADOPMODECR register.

To manually insert an error into the parity bits themselves, the parity bits have to be mapped to an different address. Once the TEST bit in the ADPARCR register 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 as shown in Figure 15-18.

Note:
Every conversion word has one parity bit. This parity bit is calculated based on bit 15:0. Bit 31:17 are reserved. Bit 16 is not used since each ADC core only supports 16 channels.

Figure 15-18. Parity Bit Map

ADC1	ADC2	31 16	0	
0xFF3E0000	0xFF3A0000	See Note	word 0	
0xFF3E0004	0xFF3A0004	See Note	word 1	
0xFF3E0008	0xFF3A0008	See Note	word 2	ADC Conversion Data
⋮	⋮	⋮		
0xFF3E00FC	0xFF3A00FC	See Note	word 63	

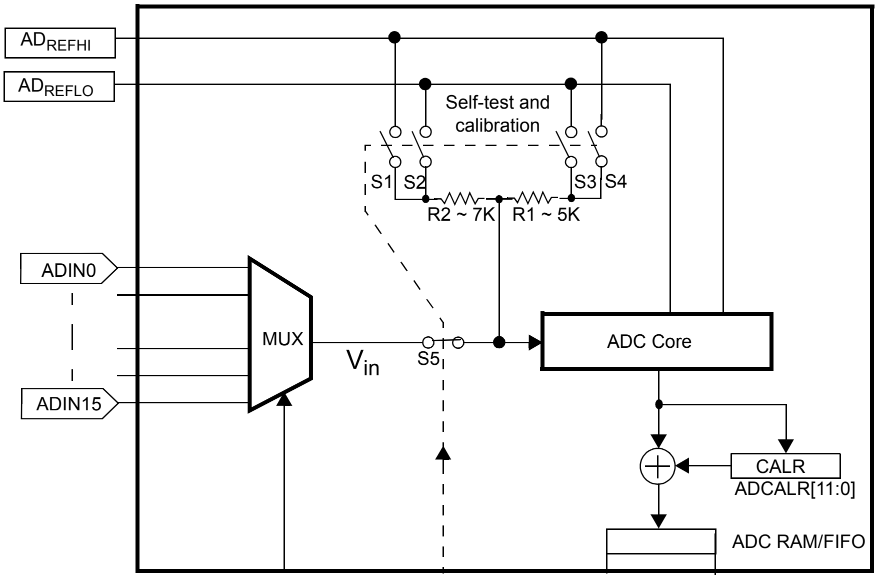
ADC1	ADC2	31 1	0	
0xFF3E1000	0xFF3A1000	Read 0	P0	
0xFF3E1004	0xFF3A1004	Read 0	P1	
0xFF3E1008	0xFF3A1008	Read 0	P2	ADC RAM Parity
⋮	⋮	⋮		
0xFF3E10FC	0xFF3A10FC	Read 0	P63	

Note: Bit 31:17 is reserved. Bit16 is the CHID[4]. Since this device only support 16 channels per ADC core, this bit is not used. Therefore, no parity protection is implemented for bit 31:16.

15.8.2 ADC Self-Test Mode

The self-test mode is used to detect an **open** or a **short** on the ADC input channels. The logic associated with both self-test and calibration is shown in Figure 15-19. Section 15.8.2.1 introduces how to configure the ADC module to do the self-test and Section 15.8.2.2 shows how to use the self-test to detect an open and short on the ADC input channels.

Figure 15-19. Self-Test and Calibration Logic



15.8.2.1 ADC Self-Test Conversions

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. Before a self-test conversion starts, the application must

1. enable the self-test mode by setting the SELF_TEST bit (ADCALCR.24).
2. define the test voltage source by the HILO bit (ADCALCR.8).

Table 15-4 shows how to set the registers to define the test voltage input to the ADC core.

Table 15-4. Self-Test Reference Voltages†

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}

† Switches refer to Figure 15-19.

The conversion group register setting, conversion data storage in self-test mode is the same as a normal conversion. After the test voltage source is defined, the application must follow the instructions in Section 15.3 to start the AD conversion and read the conversion data.

When an AD conversion (either a self-test conversion or a normal AD conversion) is ongoing, the application should NOT:

1. modify the SELF_TEST bit (ADCALCR.24) to enable or disable the self-test mode;
2. modify the HILO bit (ADCALCR.8) to change the test voltage source.

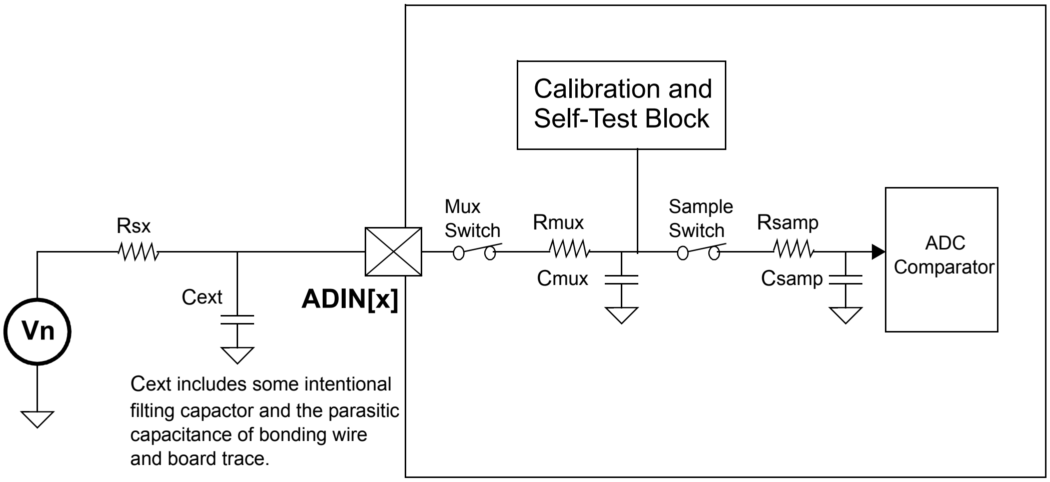
The acquisition time for each conversion is extended to twice the normal configured acquisition time by hardware. The selected reference voltage and the input voltage from the ADINx input channel are both connected to the ADC internal sampling capacitor throughout this extended acquisition period.

Section 15.8.2.2 shows an example to check open and short using ADC self-test conversions.

15.8.2.2 Use of Self-Test Mode to Determine Open/Short on ADC Input Channels

The external circuit shown in Figure 15-20 can be used to check the open and short.

Figure 15-20. Board Level Circuit for Self-Test Mode


Calibration and Self-Test Block
Rsx
Mux Switch
Rmux
Sample Switch
Rsamp
ADC Comparator
Cmux
Csamp
Cext
ADIN[x]
Vn
Cext includes some intentional filting capacitor and the parasitic capacitance of bonding wire and board trace.

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 the following table.

Table 15-5. 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}
AD_{REFLO} < Vn < AD_{REFHI}	AD_{REFHI}	AD_{REFLO}	Open

15.9 ADC Special Modes

The ADC module supports some special modes for power saving and discharging sampling capacitor purposes.

15.9.1 ADC Powerdown Mode

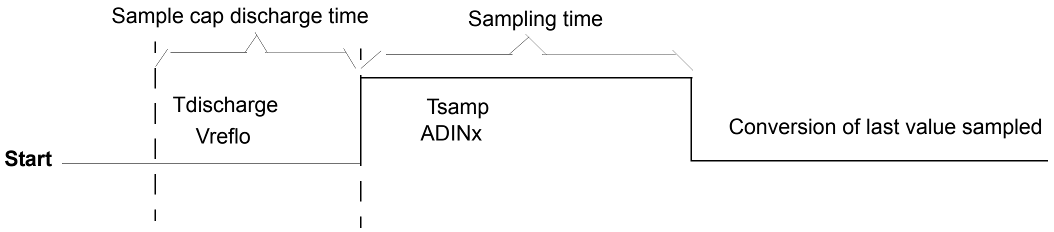
In the power-down mode, the clocks to the ADC module are stopped and the module is in a static state. This results in the lowest possible ADC power consumption. Please refer the device datasheet to identify the power consumption in low power mode.

The ADC enters power-down mode when the appropriate control bit in the power down control register in the peripheral central resource (PCR) registers frame is set. Please refer to the device datasheet to identify the appropriate control bit for powering down the ADC module.

15.9.2 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. This mode can be used if the application requires it.

Figure 15-21. Timing for Sample Capacitor Discharge Mode



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 15-21. 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.

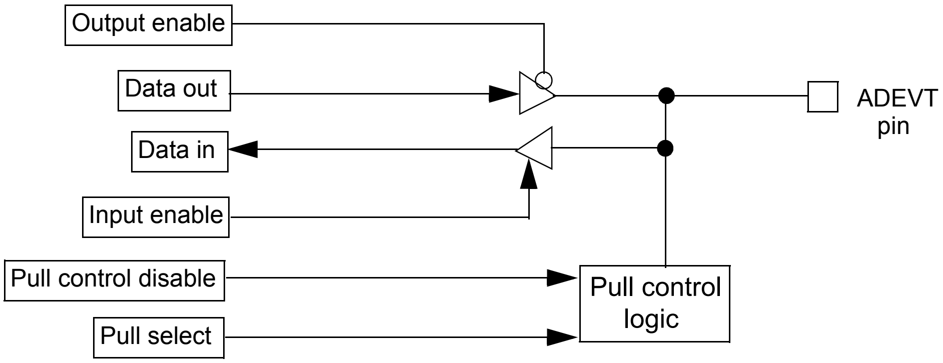
15.10 ADEVT Pin General Purpose I/O Functionality

The ADEVT pin (AD1EVT and AD2EVT pins of TMS570LS20216) can be configured to be a general-purpose I/O pin. The following sections describe the different ways in which the application can configure the ADEVT pin.

15.10.1 GPIO Functionality

Figure 15-22 illustrates the GPIO functionality.

Figure 15-22. GPIO Functionality



The following apply if the device is out of reset:

- Pull control. The pull control is enabled after reset. 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. The pull control can be disabled by setting the PDIS (pull control disable) bit in the ADEVTPDIS register.

> **Note:**
> The pull function is only available when the pin is configured as input. It is always disabled if the pin is configured as output.

- Input buffer. The input buffer is disabled only if the pin direction is set as input in the ADEVTDIR register AND the pull control is disabled AND pull down is selected. In all other cases, the input buffer is enabled.
- 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. The open-drain feature can be enabled through ADEVTPDR. This function only applies to output mode.
  - The output buffer is enabled if ADEVTOUT is 0.
  - The output buffer is disabled if ADEVTOUT is 1.

15.10.2 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 15-6.

Table 15-6. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR)	Pull Disable (PDIS)	Pull Select (PSEL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Undefined	Disabled	Undefined
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
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

15.11 ADC Control Registers

All registers in the ADC module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. The following table 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 0xFFF7C000 for ADC1 and 0xFFF7C200 for ADC2.

Table 15-7. ADC Registers Summary

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
0x00 ADRSTCR Page 762	Reserved															
	Reserved															Reset
0x04 ADOPMODECR Page 763	Reserved							COS	Reserved			Reserved				RAM_TEST_EN
	Reserved							Res	Reserved							ADC_EN
0x08 ADCLOCKCR Page 765	Reserved															
	Reserved											PS[4:0]				
0x0C ADCALCR Page 766	Reserved							SELF_TEST	Reserved							CAL_ST
	Reserved						BRIDGE_EN	HILO	Reserved							CAL_EN
0x10 ADEVMODECR Page 768	Reserved															
	Reserved						EV_DATA_FMT		Reserved		EV_CHID	OVR_EV_RAM_IGN	Reserved		EV_MODE	FRZ_EV
0x14 ADG1MODECR Page 770	Reserved															
	Reserved						G1_DATA_FMT		Reserved		G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	Res	G1_MODE	FRZ_G1
0x18 ADG2MODECR Page 773	Reserved															
	Reserved						G2_DATA_FMT		Reserved		G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	Res	G2_MODE	FRZ_G2

Table 15-7. ADC Registers Summary (Continued)

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
0x1C ADEVSRC Page 776	Reserved															
	Reserved												EV_EDG_SEL	EVSRC[2:0]		
0x20 ADG1SRC Page 776	Reserved															
	Reserved												G1_EDG_SEL	G1SRC[2:0]		
0x24 ADG2SRC Page 776	Reserved															
	Reserved												G2_EDG_SEL	G2SRC[2:0]		
0x28 ADEVINTENA Page 778	Reserved															
	Reserved												EV_END_INT_EN	Res	EV_OVR_INT_EN	EV_THR_INT_EN
0x2C ADG1INTENA Page 779	Reserved															
	Reserved												G1_END_INT_EN	Res	G1_OVR_INT_EN	G1_THR_INT_EN
0x30 ADG2INTENA Page 780	Reserved															
	Reserved												G2_END_INT_EN	Res	G2_OVR_INT_EN	G2_THR_INT_EN
0x34 ADEVINTFLG Page 781	Reserved															
	Reserved												EV_END	EV_MEM_EMPTY	EV_MEM_OVERRUN	EV_THR_INT_FLG
0x38 ADG1INTFLG Page 783	Reserved															
	Reserved												G1_END	G1_MEM_EMPTY	G1_MEM_OVERRUN	G1_THR_INT_FLG

Table 15-7. ADC Registers Summary (Continued)

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
0x3C ADG2INTFLG Page 785	Reserved															
	Reserved												G2_END	G2_MEM_EMPTY	G2_MEM_OVERRUN	G2_THR_INT_FLG
0x40 ADEVINTCR Page 787	Reserved															
	Sign Extension								EV_THR[8:0]							
0x44 ADG1INTCR Page 788	Reserved															
	Sign Extension								G1_THR[8:0]							
0x48 ADG2INTCR Page 789	Reserved															
	Sign Extension								G2_THR[8:0]							
0x4C ADEVDMACR Page 790	Reserved							EVBLOCKS[8:0]								
	Reserved													EV_BLK_XFER	Res	EV_DMA_EN
0x50 ADG1DMACR Page 792	Reserved							G1BLOCKS[8:0]								
	Reserved													G1_BLK_XFER	Res	G1_DMA_EN
0x54 ADG2DMACR Page 794	Reserved							G2BLOCKS[8:0]								
	Reserved													G2_BLK_XFER	Res	G2_DMA_EN
0x58 ADBNDCR Page 796	Reserved							BNDA[8:0]								
	Reserved							BNDB[8:0]								

Table 15-7. ADC Registers Summary (Continued)

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
0x5C ADBNDEND Page 797	Reserved															BUF_Init_Active
	Reserved													BNDEND[2:0]		
0x60 ADEVSAMP Page 799	Reserved															
	Reserved				EV_ACQ[11:0]											
0x64 ADG1SAMP Page 800	Reserved															
	Reserved				G1_ACQ[11:0]											
0x68 ADG2SAMP Page 801	Reserved															
	Reserved				G2_ACQ[11:0]											
0x6C ADEVSR Page 802	Reserved															
	Reserved												EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
0x70 ADG1SR Page 804	Reserved															
	Reserved												G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
0x74 ADG2SR Page 806	Reserved															
	Reserved												G2_MEM_EMPTY	G2_BUSY	G2_STOP	G2_END
0x78 ADEVSEL Page 808	Reserved															
	EV_SEL[15:0]															

Table 15-7. ADC Registers Summary (Continued)

Offset Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7C ADG1SEL Page 809	Reserved															
	G1_SEL[15:0]															
0x80 ADG2SEL Page 810	Reserved															
	G2_SEL[15:0]															
0x84 ADCALR Page 811	Reserved															
	Reserved				ADCALR[11:0]											
0x88	Reserved for ADC logic state machine debug purposes															
	Reserved for ADC logic state machine debug purposes															
0x8C ADLASTCONV Page 812	Reserved															
	LAST_CONV[15:0]															
0x90 to 0xAF ADEVBUFFER Page 813	EV_EMPTY	Reserved										EV_CHID				
	Reserved				EV_DR											
0xB0 to 0xCF ADG1BUFFER Page 814	G1_EMPTY	Reserved										G2_CHID				
	Reserved				G1_DR											
0xD0 to 0xEF ADG2BUFFER Page 815	G2_EMPTY	Reserved										G2_CHID				
	Reserved				G2_DR											

Table 15-7. ADC Registers Summary (Continued)

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
0xF0 ADEVEMU BUFFER Page 816	EV_EMPTY	Reserved										EV_CHID				
	Reserved				EV_DR											
0xF4 ADG1EMU BUFFER Page 816	G1_EMPTY	Reserved										G2_CHID				
	Reserved				G1_DR											
0xF8 ADG2EMU BUFFER Page 816	G2_EMPTY	Reserved										G2_CHID				
	Reserved				G2_DR											
0xFC ADEVTDIR Page 817	Reserved															
	Reserved															ADEVT_DIR
0x100 ADEVTOUT Page 818	Reserved															
	Reserved															ADEVT_OUT
0x104 ADEVTIN Page 819	Reserved															
	Reserved															ADEVT_IN
0x108 ADEVTSET Page 820	Reserved															
	Reserved															ADEVT_SET
0x10C ADEVTCLR Page 821	Reserved															
	Reserved															ADEVT_CLR

Table 15-7. ADC Registers Summary (Continued)

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
0x110 ADEVTPDR Page 822	Reserved															
	Reserved															ADEVT_PDR
0x114 ADEVTPDIS Page 823	Reserved															
	Reserved															ADEVT_PDIS
0x118 ADEVTPSEL Page 824	Reserved															
	Reserved															ADEVT_PSEL
0x11C ADEVSAMPDISEN Page 825	Reserved															
	EV_SAMP_DIS_CYC[7:0]								Reserved							EV_SAMP_DIS_EN
0x120 ADG1SAMPDISEN Page 826	Reserved															
	G1_SAMP_DIS_CYC[7:0]								Reserved							G1_SAMP_DIS_EN
0x124 ADG2SAMPDISEN Page 827	Reserved															
	G2_SAMP_DIS_CYC[7:0]								Reserved							G2_SAMP_DIS_EN
0x128 ADMAGINTCR1 Page 828	Reserved				MAG_THR1[11:0]											
	CHN/THR_COMP1	CMP_GE/LT1	Res	COMP_CHID1[4:0]					Reserved			MAG_CHID1[4:0]				
0x12C ADMAG1MASK Page 830	Reserved															
	Reserved				MAG1_MASK[11:0]											

Table 15-7. ADC Registers Summary (Continued)

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
0x130 ADMAGINTCR2 Page 828	Reserved				MAG_THR2[11:0]											
	CHN/THR_COMP2	CMP_GE/LT2	Res	COMP_CHID2[4:0]					Reserved			MAG_CHID2[4:0]				
0x134 ADMAG2MASK Page 830	Reserved															
	Reserved				MAG2_MASK[11:0]											
0x138 ADMAGINTCR3 Page 828	Reserved				MAG_THR3[11:0]											
	CHN/THR_COMP3	CMP_GE/LT3	Res	COMP_CHID3[4:0]					Reserved			MAG_CHID3[4:0]				
0x13C ADMAG3MASK Page 830	Reserved															
	Reserved				MAG3_MASK[11:0]											
0x140 to 0x154 Reserved	Reserved															
	Reserved															
0x158 ADMAGINTENA-SET Page 831	Reserved															
	Reserved													MAG_INT_ENA_SET[2:0]		
0x15C ADMAGINTENA-CLR Page 832	Reserved															
	Reserved													MAG_INT_ENA_CLR[2:0]		
0x160 ADMAGTHR INTFLG Page 833	Reserved															
	Reserved													MAG_THR_INT_FLG[2:0]		

Table 15-7. ADC Registers Summary (Continued)

Offset Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x164 ADMAGTHRINTOFFSET Page 834	Reserved															
	Reserved												MAG_INT_OFF[3:0]			
0x168 ADEVFIFORESETCR Page 835	Reserved															
	Reserved															EV_FIFO_RESET
0x16C ADG1FIFORESETCR Page 836	Reserved															
	Reserved															G1_FIFO_RESET
0x170 ADG2FIFORESETCR Page 837	Reserved															
	Reserved															G2_FIFO_RESET
0x174 ADEVRAMADDR Page 838	Reserved															
	Reserved							EV_RAM_ADDR[8:0]								
0x178 ADG1RAMADDR Page 839	Reserved															
	Reserved							G1_RAM_ADDR[8:0]								
0x17C ADG2RAMADDR Page 840	Reserved															
	Reserved							G2_RAM_ADDR[8:0]								
0x180 ADPARCR Page 841	Reserved															
	Reserved							TEST	Reserved				PARITY_ENA[3:0]			

Table 15-7. ADC Registers Summary (Continued)

Offset Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184 ADPARADDR Page 842	ERROR_ADDRESS[31:16]															
	ERROR_ADDRESS[15:0]															

15.11.1 ADC Reset Control Register (ADRSTCR)

Figure 15-23 and Table 15-8 describe the ADRSTCR register.

Figure 15-23. ADC Reset Control Register (ADRSTCR) [offset = 0x0]

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
Reserved															RESET
R-0															RWP-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-8. ADC Reset Control Register (ADRSTCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, 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.

15.11.2 ADC Operating Mode Control Register (ADOPMODECR)

Figure 15-24 and Table 15-9 describe the ADOPMODECR register.

Figure 15-24. ADC Operating Mode Control Register (ADOPMODECR) [offset = 0x4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							COS	Reserved			Reserved				RAM_TEST_EN
R-0							RW-0	R-0			RW-1010				RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							Res	Reserved							ADC_EN
R-0							RW-0	R-0							RW-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset

Table 15-9. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, 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 debug 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. User or privileged 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 zeros, writes have no effect.
20-17	Reserved	1010	Reserved for TI internal use only. This field shouldn't not be changed to anything else than 1010, otherwise the result of conversion can be unexpected.
16	RAM_TEST_EN		Enable the ADC results FIFO RAM Test Mode. Please refer to Section 15.8.1 for more details. User or privileged mode read/write:
		0	ADC RAM Test Mode is disabled. The application cannot write to the ADC RAM by the CPU or any other master.
		1	ADC RAM Test Mode is enabled. The application can directly write to the ADC RAM by the CPU or any other master.

Table 15-9. ADC Operating Mode Control Register (ADOPMODECR) Field Descriptions (Continued)

Bit	Name	Value	Description
15-9	Reserved	0	Reads return zeros, writes have no effect.
8	Reserved	0	Reads return zeros. **Note: Do NOT write a ‘1’ to this bit.**
7-1	Reserved	0	Reads return zeros, 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. User or privileged 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.

15.11.3 ADC Clock Control Register (ADCLOCKCR)

Figure 15-25 and Table 15-10 describe the ADCLOCKCR register.

Figure 15-25. ADC Clock Control Register (ADCLOCKCR) [offset = 0x8]

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
Reserved											PS[4:0]				
R-0											RW-0				

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-10. ADC Clock Control Register (ADCLOCKCR) Field Descriptions

Bit	Name	Value	Description
31–5	Reserved	0	Reads return zeros, writes have no effect.
4-0	PS[4:0]	00000 to 11111	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. User or privileged mode read/write: $t_{C(ADCLK)} = t_{C(VCLK)} * (PS[4:0] + 1)$, where $t_{C(ADCLK)}$ is the period of the ADCLK, and $t_{C(VCLK)}$ is the period of the VCLK.

15.11.4 ADC Calibration Mode Control Register (ADCALCR)

Figure 15-26 and Table 15-11 describe the ADCALCR register.

Figure 15-26. ADC Calibration Mode Control Register (ADCALCR) [offset = 0xC]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BRIDGE_EN	HILO	Reserved							CAL_EN
R-0						RW-0	RW-0	R--0							RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-11. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, 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 HILO bit determines the voltage. The desired conversion mode is configured in the group mode control registers. For more details on the ADC Self Test Mode, please refer to Section 15.8.2. User or privileged mode read/write:
		0	ADC Self Test mode is disabled.
		1	ADC Self Test mode is enabled.
23–17	Reserved	0	Reads return zeros, writes have no effect.
16	CAL_ST		ADC Calibration Conversion Start. Setting the CAL_ST bit while the CAL_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 read/write:
		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 zeros, writes have no effect.

Table 15-11. ADC Calibration Mode Control Register (ADCALCR) Field Descriptions (Continued)

Bit	Name	Value	Description
9	BRIDGE_EN		BRIDGE_ENable.
		0 - 1	In the ADC Calibration Mode, when set with the HILO bit, BRIDGE_EN allows a reference voltage to be converted. The Table 15-3, “Calibration Reference Voltages†,” on page 743 defines the four different reference voltages that can be selected.
			In other modes, this bit has no effect.
8	HILO		ADC self test mode and Calibration Mode Reference Source Selection.
		0 - 1	In the ADC self test mode, this bit defines the test voltage to be combined through a resistor with the selected input pin voltage. The Table 15-4, “Self-Test Reference Voltages†,” on page 747 defines the two different test voltages that can be selected.
			In the ADC Calibration Mode, when set with the BRIDGE_EN bit, this bit defines a reference voltage to be converted. The Table 15-3, “Calibration Reference Voltages†,” on page 743 defines the four different reference voltages that can be selected.
			In the ADC module’s normal operating mode, this bit has no effect.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	CAL_EN		ADC Calibration Enable. When this bit is set, the analog input channel multiplexer 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.
			Please refer to Section 15.7 for more details on the ADC calibration mode.
			User or privileged mode read/write:
		0	Calibration mode is disabled.
		1	Calibration mode is enabled.

15.11.5 ADC Event Group Operating Mode Control Register (*ADEVMODECR)*

Figure 15-27 and Table 15-12 describe the ADEVMODECR register.

Figure 15-27. ADC Event Group Operating Mode Control Register (ADEVMODECR) [offset = 0x10]

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
Reserved						EV_DATA_FMT		Reserved		EV_CHID	OVR_EV_RAM_IGN	Reserved		EV_MODE	FRZ_EV
R-0						RW-0		R-0		RW-0	RW-0	R-0		RW-0	RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-12. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved	0	Reads return zeros, writes have no effect.
9–8	EV_DATA_FMT		Event Group (Read) Data Format. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Event Group results memory, that is, when reading from ADEVBUFFER or ADEVEMUBUFFER.
		00	Conversion Data is read out of the Event Group memory in full 12-bit format.
		01	Conversion Data is read out of the Event Group memory in 10-bit format (RESULT Bits [11:0] >> 2).
		10	Conversion Data is read out of the Event Group memory in 8-bit format (RESULT Bits [11:0] >> 4).
		11	Reserved. - Defaults to 12-bit format if programmed.
7–6	Reserved	0	Reads return zeros, writes have no effect.
5	EV_CHID		Channel ID Mode for the Event Group. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 12-bit conversion result. User or privileged mode read/write:
		0	Data is read out of Event Group memory with the CHID field forced to 00000.
		1	Data is read out of Event Group memory with the CHID field containing the ID of the channel to which the digital result belongs.

Table 15-12. ADC Event Group Operating Mode Control Register (ADEVMODECR) Field Descriptions

Bit	Name	Value	Description
4	OVR_EV_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Event Group results memory under an overrun condition. User or privileged 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.
3–2	Reserved	0	Reads return zeros, writes have no effect.
1	EV_MODE		This bit selects whether the Event Group, as defined by the ADEVSEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Event Group.
		1	Enable continuous conversion mode of Event Group.
0	FRZ_EV		Event Group Freeze Enable. This bit allows an Event Group conversion sequence to be preempted if a Group1 or a Group2 conversion is requested. After these conversion groups complete their operation, the Event Group resumes its operation from the point where it was preempted. While the Event Group conversion is preempted, 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. User or privileged mode read/write:
		0	Event Group conversions cannot be preempted. 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 preempted whenever there is a request for conversion from Group1 or Group2.

15.11.6 ADC Group1 Operating Mode Control Register (*ADG1MODECR)*

Figure 15-28 and Table 15-13 describe the ADG1MODECR register.

Figure 15-28. ADC Group1 Operating Mode Control Register (ADG1MODECR) [offset = 0x14]

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
Reserved						G1_DATA_FMT		Reserved		G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	Res	G1_MODE	FRZ_G1
R-0						RW-0		R-0		RW-0	RW-0	RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-13. ADC Event Group Operating Mode Control Register (ADG1MODECR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved	0	Reads return zeros, writes have no effect.
9–8	G1_DATA_FMT		Group1 (Read) Data Format. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Group1 results memory, that is, when reading from ADG1BUFFER or ADG1EMUBUFFER.
		00	Conversion Data is read out of the Group1 memory in full 12-bit format.
		01	Conversion Data is read out of the Group1 memory in 10-bit format (RESULT Bits [11:0] >> 2).
		10	Conversion Data is read out of the Group1 memory in 8-bit format (RESULT Bits [11:0] >> 4).
		11	Reserved. - Defaults to 12-bit format if programmed.
7–6	Reserved	0	Reads return zeros, writes have no effect.
5	G1_CHID		Channel ID Mode for the Group1. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 12-bit conversion result. User or privileged mode read/write:
		0	Data is read out of Group1 memory with the CHID field forced to 00000.
		1	Data is read out of Group1 memory with the CHID field containing the ID of the channel to which the digital result belongs.

Table 15-13. ADC Event Group Operating Mode Control Register (ADG1MODECR) Field Descriptions

Bit	Name	Value	Description
4	OVR_G1_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group1 results memory under an overrun condition. User or privileged 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.
3	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, please refer to Section 15.3.6. User or privileged 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).
2	Reserved	0	Reads return zeros, writes have no effect.
1	G1_MODE		This bit selects whether the Group1, as defined by the ADG1SEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Group1.
		1	Enable continuous conversion mode of Group1.

Table 15-13. ADC Event Group Operating Mode Control Register (ADG1MODECR) Field Descriptions

Bit	Name	Value	Description
0	FRZ_G1		Group1 Freeze Enable. This bit allows a Group1 conversion sequence to be preempted if an Event Group or a Group2 conversion is requested. After these conversion groups complete their operation, the Group1 resumes its operation from the point where it was preempted. While the Group1 conversion is preempted, 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. User or privileged mode read/write:
		0	Group1 conversions cannot be preempted. 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 preempted whenever there is a request for conversion from Event Group or Group2.

15.11.7 ADC Group2 Operating Mode Control Register (*ADG2MODECR)*

Figure 15-29 and Table 15-14 describe the ADG2MODECR register.

Figure 15-29. ADC Group2 Operating Mode Control Register (ADG2MODECR) [offset = 0x18]

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
Reserved						G2_DATA_FMT		Reserved		G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	Res	G2_MODE	FRZ_G2
R-0						RW-0		R-0		RW-0	RW-0	RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, RS = Read/Set, -*n* = value after reset

Table 15-14. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
31–10	Reserved	0	Reads return zeros, writes have no effect.
9–8	G2_DATA_FMT		Group2 (Read) Data Format. It determines the format in which the Conversion Result is read out when using the FIFO interface of the Group2 results memory, that is, when reading from ADG2BUFFER or ADG2EMUBUFFER.
		00	Conversion Data is read out of the Group2 memory in full 12-bit format.
		01	Conversion Data is read out of the Group2 memory in 10-bit format (RESULT Bits [11:0] >> 2).
		10	Conversion Data is read out of the Group2 memory in 8-bit format (RESULT Bits [11:0] >> 4).
		11	Reserved. - Defaults to 12-bit format if programmed.
7–6	Reserved		Reads return zeros, writes have no effect.
5	G2_CHID		Channel ID Mode for the Group2. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results FIFO RAM. Any 32-bit read performed in the “read from RAM” mode will return the 5-bit channel id along with the 12-bit conversion result. User or privileged mode read/write:
		0	Data is read out of Group2 memory with the CHID field forced to 00000.
		1	Data is read out of Group2 memory with the CHID field containing the ID of the channel to which the digital result belongs.

Table 15-14. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
4	OVR_G2_RAM_IGN		This bit allows the ADC module to overwrite the contents of the Group2 results memory under an overrun condition. User or privileged 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.
3	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, please refer to Section 15.3.6. User or privileged 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).
2	Reserved		Reads return zeros, writes have no effect.
1	G2_MODE		This bit selects whether the Group2, as defined by the ADG2SEL register, is converted in single or continuous conversion mode. User or privileged mode read/write:
		0	Enable single conversion mode of Group2.
		1	Enable continuous conversion mode of Group2.

Table 15-14. ADC Group2 Operating Mode Control Register (ADG2MODECR) Field Descriptions

Bit	Name	Value	Description
0	FRZ_G2		Group2 Freeze Enable. This bit allows a Group2 conversion sequence to be preempted if an Event Group or a Group1 conversion is requested. After these conversion groups complete their operation, the Group2 resumes its operation from the point where it was preempted. While the Group2 conversion is preempted, 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. User or privileged mode read/write:
		0	Group2 conversions cannot be preempted. 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 preempted whenever there is a request for conversion from Event Group or Group1.

15.11.8 ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC)

Figure 15-30, Figure 15-31, Figure 15-32and Table 15-15 describe the ADEVSRC, ADG1SRC and ADG2SRC register.

Figure 15-30. ADC Event Group Trigger Source Select Register (ADEVSRC) [offset = 0x1C]

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
Reserved												EV_EDG_SEL	EV_SRC		
R-0												RW-0	RW-0		

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Figure 15-31. ADC Group1 Trigger Source Select Register (ADG1SRC) [offset = 0x20]

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
Reserved												G1_EDG_SEL	G1_SRC		
R-0												RW-0	RW-0		

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Figure 15-32. ADC Group2 Trigger Source Select Register (ADG2SRC) [offset = 0x24]

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
Reserved												G2_EDG_SEL	G2_SRC		
R-0												RW-0	RW-0		

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-15. ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.

Table 15-15. ADC Trigger Source Select Register (ADEVSRC, ADG1SRC and ADG2SRC) Field Descriptions

Bit	Name	Value	Description
3	EV_EDG_SEL G1_EDG_SEL G2_EDG_SEL		EV Group / Group1/ Group2 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group conversion. User or privileged mode read/write:
		0	A falling edge on the selected source will trigger the Group conversion.
		1	A rising edge on the selected source will trigger the Group conversion.
2–0	EV_SRC G1_SRC G2_SRC		Event Group / Group1/ Group2 Trigger Source. The application can select different trigger source from ADEVT pin, NHET, RTI and GIOB. User or privileged mode read/write:
		000	AD1EVT is the trigger source (ADC core 1); AD2EVT is the trigger source (ADC core 2).
		001	NHET[8] is the trigger source.
		010	NHET[10] is the trigger source.
		011	RTI compare 0 is the trigger source.
		100	NHET[17] is the trigger source.
		101	NHET[19] is the trigger source.
		110	GIOB[0] is the trigger source.
		111	GIOB[1] is the trigger source.
			Note: the trigger sources listed above are specific to the TMS570LS20216.

15.11.9 ADC Event Interrupt Enable Control Register (ADEVINTENA)

Figure 15-33 and Table 15-16 describe the ADEVINTENA register.

Figure 15-33. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) [offset = 0x28]

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
Reserved												EV_END_INT_EN	Res	EV_OVR_INT_EN	EV_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-16. ADC Event Group Interrupt Enable Control Register (ADEVINTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	EV_END_INT_EN		Event Group Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts.
			User or privileged mode read/write:
		0	Event Group Conversion End Interrupt is disabled.
		1	Event Group Conversion End Interrupt is Enabled.
2	Reserved	0	Reads return zeros, 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 which is already full. For more details on the overrun interrupts please refer to Section 15.5.3
			User or privileged mode read/write:
		0	Event Group memory overrun interrupt is disabled.
		1	Event Group memory overrun interrupt is enabled.
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. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts.
			User or privileged mode read/write:
		0	Event Group threshold interrupt is disabled.
		1	Event Group threshold interrupt is enabled.

15.11.10 ADC Group1 Interrupt Enable Control Register (ADG1INTENA)

Figure 15-34 and Table 15-17 describe the ADG1INTENA register.

Figure 15-34. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) [offset = 0x2C]

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
Reserved												G1_END_INT_EN	Res	G1_OVR_INT_EN	G1_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-17. ADC Group1 Interrupt Enable Control Register (ADG1INTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G1_END_INT_EN		Group1 Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts. User or privileged mode read/write:
		0	Group1 Conversion End Interrupt is disabled.
		1	Group1 Conversion End Interrupt is enabled.
2	Reserved	0	Reads return zeros, 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 which is already full. For more details on the overrun interrupts please refer to Section 15.5.3 User or privileged mode read/write:
		0	Group1 memory overrun interrupt is disabled.
		1	Group1 memory overrun interrupt is enabled.
0	G1_THR_INT_EN		Group1 Threshold Interrupt Enable. A Group1 threshold interrupt occurs when the programmed Group1 threshold counter counts down to zero. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts. User or privileged mode read/write:
		0	Group1 threshold interrupt is disabled.
		1	Group1 threshold interrupt is enabled.

15.11.11 ADC Group2 Interrupt Enable Control Register (ADG2INTENA)

Figure 15-35 and Table 15-18 describe the ADG1INTENA register.

Figure 15-35. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) [offset = 0x30]

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
Reserved												G2_END_INT_EN	Res	G2_OVR_INT_EN	G2_THR_INT_EN
R-0												RW-0	R-0	RW-0	RW-0

RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-18. ADC Group2 Interrupt Enable Control Register (ADG2INTENA) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G2_END_INT_EN		Group2 Conversion End Interrupt Enable. Please refer to Section 15.5.1 for more details on the conversion end interrupts. User or privileged mode read/write:
		0	Group2 Conversion End Interrupt is disabled.
		1	Group2 Conversion End Interrupt is enabled.
2	Reserved	0	Reads return zeros, 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 which is already full. For more details on the overrun interrupts please refer to Section 15.5.3 User or privileged mode read/write:
		0	Group2 memory overrun interrupt is disabled.
		1	Group2 memory overrun interrupt is enabled.
0	G2_THR_INT_EN		Group2 Threshold Interrupt Enable. A Group2 threshold interrupt occurs when the programmed Group2 threshold counter counts down to zero. 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. Please refer to Section 15.5.2 for more details on the threshold interrupts. User or privileged mode read/write:
		0	Group2 threshold interrupt is disabled.
		1	Group2 threshold interrupt is enabled.

15.11.12 ADC Event Group Interrupt Flag Register (ADEVINTFLG)

Figure 15-33 and Table 15-19 describe the ADEVINTFLG register.

Figure 15-36. ADC Event Group Interrupt Flag Register (ADEVINTFLG) [offset = 0x34]

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
Reserved												EV_END	EV_MEM_EMPTY	EV_MEM_OVERRUN	EV_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-19. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	EV_END		Event Group Conversion End. This bit and the Event Group status register bit 0 (ADEVSR.0) are identical. User or privileged 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 (ADEVSR.0). - By reading one conversion result from ADEVBUFFER - any Event Group FIFO read - By writing a new set of channels to the Event Group channel select register (ADEVSEL).
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. This bit and the Event Group status register bit 3 (ADEVSR.3) are identical. User or privileged mode read:
		0	The Event Group results memory is not empty.
		1	The Event Group results memory is empty.

Table 15-19. ADC Event Group Interrupt Flag Register (ADEVINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	EV_MEM_OVERRUN		Event Group Memory Overrun. This is a read-only bit; writes have no effect.
			User or privileged 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.
			User or privileged mode read:
		0	Event Group buffer has no pending interrupt.
		1	Event Group buffer has pending interrupt due to threshold being reached.
			This bit can be cleared by writing a ‘1’ ; writing a ‘0’ has no effect.

15.11.13 ADC Group1 Interrupt Flag Register (ADG1INTFLG)

Figure 15-37 and Table 15-20 describe the ADG1INTFLG register.

Figure 15-37. ADC Group1 Interrupt Flag Register (ADG1INTFLG) [offset = 0x38]

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
Reserved												G1_END	G1_MEM_EMPTY	G1_MEM_OVERRUN	G1_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-20. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G1_END		Group1 Conversion End. This bit and the Group1 status register bit 0 (ADG1SR.0) are identical. User or privileged 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 (ADG1SR.0) - By reading one conversion result from ADG1BUFFER - any Group1 FIFO read - By writing a new set of channels to the Group1 channel select register (ADG1SEL).
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. This bit and the Group1 status register bit 3 (ADG1SR.3) are identical. User or privileged mode read:
		0	The Group1 results memory is not empty.
		1	The Group1 results memory is empty.

Table 15-20. ADC Group1 Interrupt Flag Register (ADG1INTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	G1_MEM_OVERRUN		Group1 Memory Overrun. This is a read-only bit; writes have no effect.
			User or privileged mode read:
		0	Group1 results memory has not overrun.
		1	Group1 results memory has overrun.
0	G1_THR_INT_FLG		Group1 Threshold Interrupt Flag.
			User or privileged mode read:
		0	Group1 buffer has no pending interrupt.
		1	Group1 buffer has pending interrupt due to threshold being reached.
			This bit can be cleared only by writing a ‘1’; writing a ‘0’ has no effect.

15.11.14 ADC Group2 Interrupt Flag Register (ADG2INTFLG)

Figure 15-38 and Table 15-21 describe the ADG2INTFLG register.

Figure 15-38. ADC Group2 Interrupt Flag Register (ADG2INTFLG) [offset = 0x3C]

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
Reserved												G2_END	G2_MEM_EMPTY	G2_MEM_OVERRUN	G2_THR_INT_FLG
R-0												RC-0	R-1	R-0	RC-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-21. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3	G2_END		Group2 Conversion End. This bit and the Group2 status register bit 0 (ADG2SR.0) are identical.
			User or privileged 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 (ADG2SR.0) - By reading one conversion result from ADG2BUFFER - any Group2 FIFO read - By writing a new set of channels to the Group2 channel select register (ADG2SEL).
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. This bit and the Group2 status register bit 3 (ADG2SR.3) are identical.
			User or privileged mode read:
		0	The Group2 results memory is not empty.
		1	The Group2 results memory is empty.

Table 15-21. ADC Group2 Interrupt Flag Register (ADG2INTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
1	G2_MEM_OVERRUN		Group2 Memory Overrun. This is a read-only bit; writes have no effect.
			User or privileged mode read:
		0	Group2 results memory has not overrun.
		1	Group2 results memory has overrun.
0	G2_THR_INT_FLG		Group2 Threshold Interrupt Flag.
			User or privileged mode read:
		0	Group2 buffer has no pending interrupt.
		1	Group2 buffer has pending interrupt due to threshold being reached.
			This bit can be cleared only by writing a ‘1’; writing a ‘0’ has no effect.

15.11.15 ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR)

Figure 15-39 and Table 15-22 describe the ADEVTHRINTCR register.

Figure 15-39. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) [offset = 0x40]

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
Sign Extension							EV_THR[8:0]								
R-0							RW-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-22. ADC Event Group Threshold Interrupt Control Register (ADEVTHRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–9	Sign Extension		These bits always read the same as the bit 8 of this register, writes have no effect.
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. After conversions begin, this register tracks the number of ADC sample results currently held in Event Group memory; by decrementing after each new result is placed in the Event Group FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU or DMA. When this counter decrements from 1 to 0, it causes the EV_THR_INT_FLG to be set, and an interrupt request is generated if the EV_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.11.16 ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR)

Figure 15-40 and Table 15-23 describe the ADG1THRINTCR register.

Figure 15-40. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) [offset = 0x44]

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
Sign Extension							G1_THR[8:0]								
R-0							RW-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-23. ADC Group1 Threshold Interrupt Control Register (ADG1THRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, 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. After conversions begin, this register tracks the number of ADC sample results currently held in Group1 memory; by decrementing after each new result is placed in the Group1 FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU or DMA. When this counter decrements from 1 to 0, it causes the G1_THR_INT_FLG to be set, and an interrupt request is generated if the G1_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.11.17 ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR)

Figure 15-42 and Table 15-25 describe the ADG2THRINTCR register.

Figure 15-41. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) [offset = 0x48]

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
Sign Extension							G2_THR[8:0]								
R-0							RW-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-24. ADC Group2 Threshold Interrupt Control Register (ADG2THRINTCR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, 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. After conversions begin, this register tracks the number of ADC sample results currently held in Group2 memory; by decrementing after each new result is placed in the Group2 FIFO by the AD and incrementing as each value is read out of the FIFO by the CPU or DMA. When this counter decrements from 1 to 0, it causes the G2_THR_INT_FLG to be set, and an interrupt request is generated if the G2_THR_INT_EN bit is set. Please refer to Section 15.5.2 for more details on the threshold interrupts.

15.11.18 ADC Event Group DMA Control Register (ADEVDMACR)

Figure 15-42 and Table 15-25 describe the ADEVDMACR register.

Figure 15-42. ADC Event Group DMA Control Register (ADEVDMACR) [offset = 0x4C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">EVBLOCKS[8:0]</td></tr>
<tr><td colspan="7">R-0</td><td colspan="9">RW-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">Reserved</td><td>EV_BLK_XFER</td><td>Res</td><td>EV_DMA_EN</td></tr>
<tr><td colspan="13">R-0</td><td>RW-0</td><td>R-0</td><td>RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-25. ADC Event Group DMA Control Register (ADEVDMACR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24–16	EVBLOCKS		Number of Event Group memory buffers to be transferred using DMA if the ADC module generates a DMA request. If the EV_BLK_XFER bit is set to ‘1’, a DMA request will be generated after the EV FIFO collects EVBLOCKS number of conversion results. This feature is to be used in place of the threshold interrupt for the Event Group. Any value written to the EVBLOCKS field can also be read from the EVTHR field of the ADEVTHRINTCR register.
		0	No DMA transfer occurs even if EV_BLK_XFER is set to ‘1’.
		Other	Read value of this field shows the current value of the Threshold Counter. This is the down counter shared by the Threshold Counter Interrupt logic. It is expected that DMA controller is capable of reading out the Block Count number of data from the FIFO before another Block Count of ADC conversions complete. For example, if the BLOCK count is configured for 10, then ADC module will generate a DMA request at the end of the 10th conversion. DMA controller should complete reading out 10 data before next set of 10 conversions complete. Please refer to Section 15.6 for more details.
15–3	Reserved	0	Reads return zeros, writes have no effect.
2	EV_BLK_XFER		Event Group Block DMA Transfer Enable.
		0	ADC module generates a DMA request for each write to the Event Group memory if EV_DMA_EN is set.
		1	ADC module generates a DMA request when the ADC has written EVBLOCKS number of buffers into the Event Group memory. If EV_BLK_XFER bit is set to ‘1’, EV_DMA_EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches ‘0’ from a count value of 1.

Table 15-25. ADC Event Group DMA Control Register (ADEVDMACR) Field Descriptions

Bit	Name	Value	Description
1	Reserved	0	Reads return zeros, writes have no effect.
0	EV_DMA_EN		Event Group DMA Transfer Enable.
		0	ADC module does not generate a DMA request when it writes the conversion result to the Event Group memory.
		1	ADC module generates a DMA transfer when the ADC has written to the Event Group memory. Please refer to Section 15.6 for more details. EV_BLK_XFER bit should be set to ‘0’, for this normal DMA request to be generated.

15.11.19 ADC Group1 DMA Control Register (ADG1DMACR)

Figure 15-42 and Table 15-25 describe the ADG1DMACR register.

Figure 15-43. ADC Group1 DMA Control Register (ADG1DMACR) [offset = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							G1BLOCKS[8:0]								
R-0							RW-0								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													G1_BLK_XFER	Res	G1_DMA_EN
R-0													RW-0	R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-26. ADC Group1 DMA Control Register (ADG1DMACR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24–16	G1BLOCKS		Number of Group1 memory buffers to be transferred using DMA if the ADC module generates a DMA request. If the G1_BLK_XFER bit is set to '1', a DMA request will be generated after the G1 FIFO collects G1BLOCKS number of conversion results. This feature is to be used in place of the threshold interrupt for the Group1. Any value written to the G1BLOCKS field can also be read from the G1THR field of the ADG1THRINTCR register.
		0	No DMA transfer occurs even if G1_BLK_XFER is set to '1'.
		Other	Read value of this field shows the current value of the Threshold Counter. This is the down counter shared by the Threshold Counter Interrupt logic. It is expected that DMA controller is capable of reading out the Block Count number of data from the FIFO before another Block Count of ADC conversions complete. For example, if the BLOCK count is configured for 10, then ADC module will generate a DMA request at the end of the 10th conversion. DMA controller should complete reading out 10 data before next set of 10 conversions complete. Please refer to Section 15.6 for more details.
15–3	Reserved	0	Reads return zeros, writes have no effect.
2	G1_BLK_XFER		Group1 Block DMA Transfer Enable.
		0	ADC module generates a DMA request for each write to the Group1 memory if G1_DMA_EN is set.
		1	ADC module generates a DMA request when the ADC has written G1BLOCKS number of buffers into the Group1 memory. If G1_BLK_XFER bit is set to '1', G1_DMA_EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches '0' from a count value of 1.

Table 15-26. ADC Group1 DMA Control Register (ADG1DMACR) Field Descriptions

Bit	Name	Value	Description
1	Reserved	0	Reads return zeros, writes have no effect.
0	G1_DMA_EN		Group1 DMA Transfer Enable.
		0	ADC module does not generate a DMA request when it writes the conversion result to the Group1 memory.
		1	ADC module generates a DMA transfer when the ADC has written to the Group1 memory. Please refer to Section 15.6 for more details. G1_BLK_XFER bit should be set to ‘0’, for this normal DMA request to be generated.

15.11.20 ADC Group2 DMA Control Register (ADG2DMACR)

Figure 15-42 and Table 15-25 describe the ADG2DMACR register.

Figure 15-44. ADC Group2 DMA Control Register (ADG2DMACR) [offset = 0x54]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">G2BLOCKS[8:0]</td></tr>
<tr><td colspan="7">R-0</td><td colspan="9">RW-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">Reserved</td><td>G2_BLK_XFER</td><td>Res</td><td>G2_DMA_EN</td></tr>
<tr><td colspan="13">R-0</td><td>RW-0</td><td>R-0</td><td>RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-27. ADC Group2 DMA Control Register (ADG2DMACR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, writes have no effect.
24–16	G2BLOCKS		Number of Group2 memory buffers to be transferred using DMA if the ADC module generates a DMA request. If the G2_BLK_XFER bit is set to ‘1’, a DMA request will be generated after the G2 FIFO collects G2BLOCKS number of conversion results. This feature is to be used in place of the threshold interrupt for the Group2. Any value written to the G2BLOCKS field can also be read from the G2THR field of the ADG2THRINTCR register.
		0	No DMA transfer occurs even if G2_BLK_XFER is set to ‘1’.
		Other	Read value of this field shows the current value of the Threshold Counter. This is the down counter shared by the Threshold Counter Interrupt logic. It is expected that DMA controller is capable of reading out the Block Count number of data from the FIFO before another Block Count of ADC conversions complete. For example, if the BLOCK count is configured for 10, then ADC module will generate a DMA request at the end of the 10th conversion. DMA controller should complete reading out 10 data before next set of 10 conversions complete. Please refer to Section 15.6 for more details.
15–3	Reserved	0	Reads return zeros, writes have no effect.
2	G2_BLK_XFER		Group2 Block DMA Transfer Enable.
		0	ADC module generates a DMA request for each write to the Group2 memory if G2_DMA_EN is set.
		1	ADC module generates a DMA request when the ADC has written G2BLOCKS number of buffers into the Group2 memory. If G2_BLK_XFER bit is set to ‘1’, G2_DMA_EN bit is ignored and DMA requests will be generated every time the Threshold Counter reaches ‘0’ from a count value of 1.

Table 15-27. ADC Group2 DMA Control Register (ADG2DMACR) Field Descriptions

Bit	Name	Value	Description
1	Reserved	0	Reads return zeros, writes have no effect.
0	G2_DMA_EN		Group2 DMA Transfer Enable.
		0	ADC module does not generate a DMA request when it writes the conversion result to the Group2 memory.
		1	ADC module generates a DMA transfer when the ADC has written to the Group2 memory. G2_BLK_XFER bit should be set to ‘0’, for this normal DMA request to be generated. Please refer to Section 15.6 for more details.

15.11.21 ADC Results Memory Configuration Register (ADBNDCR)

Figure 15-45 and Table 15-28 describe the ADBNDCR register.

Figure 15-45. ADC Results Memory Configuration Register (ADBNDCR) [offset = 0x58]

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

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-28. ADC Results Memory Configuration Register (ADBNDCR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved	0	Reads return zeros, 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. User or privileged 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.
		0x001 to 0x020	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 zeros, 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. User or privileged mode read:
		0	Event Group as well as Group1 conversions are not required.
		0x001 to 0x020	A total of 2 * (BNDB - BNDA) buffers are available in the ADC results memory for storing Group1 conversion results.

Please refer to Section 15.3.8 for further details on how the conversion results are stored in the ADC results FIFO RAM.

15.11.22 ADC Results Memory Size Configuration Register (ADBNDEND)

Figure 15-46 and Table 15-29 describe the ADBNDCR register.

Figure 15-46. ADC Results Memory Size Configuration Register (ADBNDEND) [offset = 0x5C]

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

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-29. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved	0	Reads return zeros, writes have no effect.
16	BUF_Init_Active		ADC Results Memory Auto-initialization Status. User or privileged mode read only:
		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. Please refer to the Architecture User Guide for more details.
		1	ADC results memory is being initialized, and the ADC is not available for conversion.
15–3	Reserved	0	Reads return zeros, writes have no effect.

Table 15-29. ADC Results Memory Size Configuration Register (ADBNDEND) Field Descriptions (Continued)

Bit	Name	Value	Description
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 size of the ADC results memory available on the specific device (64 words available on TMS570LS20216). Please refer to the specific device datasheet for information on the memory available for storing the ADC conversion results. For device TMS570LS20216, the user must program BNDEND as 010b (64 words) or less. The reason you would do 'less' is for compatibility with another chip that has less ADC RAM. User or privileged mode read/write:
		000b	16 words available for storing ADC conversion results.
		001b	32 words available for storing ADC conversion results.
		010b	64 words available for storing ADC conversion results.
		011b	128 words available for storing ADC conversion results.
		100b	192 words available for storing ADC conversion results.
		101b	256 words available for storing ADC conversion results.
		110b	512 words available for storing ADC conversion results.
		111b	1024 words available for storing ADC conversion results.

15.11.23 ADC Event Group Sampling Time Configuration Register (ADEVSAMP)

Figure 15-47 and Table 15-30 describe the ADEVSAMP register.

Figure 15-47. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) [offset = 0x60]

31–16
Reserved
R-0

15–12	11–0
Reserved	EV_ACQ[11:0]
R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-30. ADC Event Group Sampling Time Configuration Register (ADEVSAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	EV_ACQ		Event Group Acquisition Time. These bits define the sampling window (SW) for the Event Group conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$, where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. 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. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.11.24 ADC Group1 Sampling Time Configuration Register (ADG1SAMP)

Figure 15-48 and Table 15-31 describe the ADG1SAMP register.

Figure 15-48. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) [offset = 0x64]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="12">G1_ACQ[11:0]</td></tr>
<tr><td colspan="4">R-0</td><td colspan="12">RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-31. ADC Group1 Sampling Time Configuration Register (ADG1SAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G1_ACQ		Group1 Acquisition Time. These bits define the sampling window (SW) for the Group1 conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. 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. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.11.25 ADC Group2 Sampling Time Configuration Register (ADG2SAMP)

Figure 15-49 and Table 15-32 describe the ADG2SAMP register.

Figure 15-49. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) [offset = 0x68]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="12">G2_ACQ[11:0]</td></tr>
<tr><td colspan="4">R-0</td><td colspan="12">RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-32. ADC Group2 Sampling Time Configuration Register (ADG2SAMP) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G2_ACQ		Group2 Acquisition Time. These bits define the sampling window (SW) for the Group2 conversions. Sample Window = (EV_ACQ + 2) x $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns. 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. Please refer to the device datasheet to determine the minimum sampling time for this device.

15.11.26 ADC Event Group Status Register (ADEVSR)

Figure 15-50 and Table 15-33 describe the ADEVSR register.

Figure 15-50. ADC Event Group Status Register (ADEVSR) [offset = 0x6C]

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
Reserved												EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
R-0												R-1	R-0	R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-33. ADC Event Group Status Register (ADEVSR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, 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. This bit and the Event Group Interrupt Flag Register bit 2 (ADEVINTFLG.2) are identical. User or privileged mode read:
		0	The Event Group results memory has unread conversion results.
		1	The Event Group results memory is empty, or all conversion results in the memory have already been read.
2	EV_BUSY		Event Group Conversion Busy. User or privileged mode read:
		0	Event Group conversions are neither in progress nor preempted.
		1	Event Group conversions are either in progress, or are preempted 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. User or privileged mode read:
		0	Event Group conversions are not currently preempted.
		1	Event Group conversions are currently preempted.

Table 15-33. ADC Event Group Status Register (ADEVSR) Field Descriptions (Continued)

Bit	Name	Value	Description
0	EV_END		Event Group Conversions Ended. This bit and the Event Group Interrupt Flag Register bit 3 (ADEVINTFLG.3) are identical. User or privileged 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

15.11.27 ADC Group1 Status Register (ADG1SR)

Figure 15-51 and Table 15-34 describe the ADG1SR register.

Figure 15-51. ADC Group1 Status Register (ADG1SR) [offset = 0x70]

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
Reserved												G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
R-0												R-1	R-0	R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-34. ADC Group1 Status Register (ADG1SR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, 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. This bit and the Group1 Interrupt Flag Register bit 2 (ADG1INTFLG.2) are identical. User or privileged mode read:
		0	The Group1 results memory has unread conversion results.
		1	The Group1 results memory is empty, or all conversion results in the memory have already been read.
2	G1_BUSY		Group1 Conversion Busy. User or privileged mode read:
		0	Group1 conversions are neither in progress nor preempted.
		1	Group1 conversions are either in progress, or are preempted 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. User or privileged mode read:
		0	Group1 conversions are not currently preempted.
		1	Group1 conversions are currently preempted.

Bit	Name	Value	Description
0	G1_END		Group1 Conversions Ended. This bit and the Group1 Interrupt Flag Register bit 3 (ADG1INTFLG.3) are identical. User or privileged 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

15.11.28 ADC Group2 Status Register (ADG2SR)

Figure 15-52 and Table 15-35 describe the ADG2SR register.

Figure 15-52. ADC Group2 Status Register (ADG2SR) [offset = 0x74]

31–16
Reserved
R-0

15–4	3	2	1	0
Reserved	G2_MEM_EMPTY	G2_BUSY	G2_STOP	G2_END
R-0	R-1	R-0	R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-35. ADC Group2 Status Register (ADG2SR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, 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. This bit and the Group2 Interrupt Flag Register bit 2 (ADG2INTFLG.2) are identical. User or privileged mode read:
		0	The Group2 results memory has unread conversion results.
		1	The Group2 results memory is empty, or all conversion results in the memory have already been read.
2	G2_BUSY		Group2 Conversion Busy. User or privileged mode read:
		0	Group2 conversions are neither in progress nor preempted.
		1	Group2 conversions are either in progress, or are preempted 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. User or privileged mode read:
		0	Group2 conversions are not currently preempted.
		1	Group2 conversions are currently preempted.

Table 15-35. ADC Group2 Status Register (ADG2SR) Field Descriptions (Continued)

Bit	Name	Value	Description
0	G2_END		Group2 Conversions Ended. This bit and the Group2 Interrupt Flag Register bit 3 (ADG2INTFLG.3) are identical. User or privileged 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

15.11.29 ADC Event Group Channel Select Register (ADEVSEL)

Figure 15-53 and Table 15-36 describe the ADEVSEL register.

Figure 15-53. ADC Event Group Channel Select Register (ADEVSEL) [offset = 0x78]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EV_SEL[15:0]															
RW-0															

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-36. ADC Event Group Channel Select Register (ADEVSEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	EV_SEL		Event Group channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Event Group.
		Non-zero	Setting bit EV_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Event Group is triggered.

Note: Clearing ADEVSEL During a Conversion

Writing 0x0000 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 or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

15.11.30 ADC Group1 Channel Select Register (ADG1SEL)

Figure 15-54 and Table 15-37 describe the ADG1SEL register.

Figure 15-54. ADC Group1 Channel Select Register (ADG1SEL) [offset = 0x7C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">RW-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">G1_SEL[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-37. ADC Group1 Channel Select Register (ADG1SEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	G1_SEL		Group1 channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Group1.
		Non-zero	Setting bit G1_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Group1 is triggered.

Note: Clearing ADG1SEL During a Conversion

Writing 0x0000 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 or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

15.11.31 ADC Group2 Channel Select Register (ADG2SEL)

Figure 15-55 and Table 15-38 describe the ADG2SEL register.

Figure 15-55. ADC Group2 Channel Select Register (ADG2SEL) [offset = 0x80]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G2_SEL[15:0]															
RW-0															

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-38. ADC Group2 Channel Select Register (ADG2SEL) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–0	G2_SEL		Group2 channels selected. User or privileged mode read/write:
		0	No ADC input channel is selected for conversion in the Group2.
		Non-zero	Setting bit G2_SEL[x] means that pin ADIN[x] (x=0~7) or ADSIN[x] (x=8~15) will be converted as part of the Event Group. All the selected channels will be converted in ascending order when the Group2 is triggered.

Note: Clearing ADG2SEL During a Conversion

Writing 0x0000 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 or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

15.11.32 ADC Calibration and Error Offset Correction Register (ADCALR)

Figure 15-56 and Table 15-39 describe the ADCALR register.

Figure 15-56. ADC Calibration and Error Offset Correction Register (ADCALR) [offset = 0x84]

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
Reserved				ADCALR[11:0]											
R-0				RW-0											

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-39. ADC Calibration and Error Offset Correction Register (ADCALR) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	ADCALR		ADC Calibration Result and Offset Error Correction Value. 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 12-bit 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, please refer to the Section 15.7.

15.11.33 ADC Channel Last Conversion Value Register (ADLASTCONV)

Figure 15-57 and Table 15-40 describe the ADLASTCONV register.

Figure 15-57. ADC Channel Last Conversion Value Register (ADLASTCONV) [offset = 0x8C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAST_CONV[15:0]															
R-U															

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-40. ADC Channel Last Conversion Value Register (ADLASTCONV) Field Descriptions

Bit	Name	Value	Description
31–0	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. Any conversion result from 0 to 0x7FF resoves as 0, otherwise resolves as 1. 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 the midpoint reference voltage was measured at the last conversion for this channel.

15.11.34 ADC Event Group Results FIFO (ADEVBUFFER)

Figure 15-58 and Table 15-41 describe the ADEVBUFFER register.

Figure 15-58. ADC Event Group Results FIFO (ADEVBUFFER) [offset = 0x90 - 0xAF]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EV_EMPTY	Reserved										EV_CHID				
R-1	R-0										RW-U				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				EV_DR											
R-0				R-U											

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Event Group results FIFO location is aliased eight times, so that any 32-bit 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.

Table 15-41. ADC Event Group Results FIFO (ADEVBUFFER) Field Descriptions

Bit	Name	Value	Description
31	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. User or privileged 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.
30–21	Reserved	0	Reads return zeros, writes have no effect.
20–16	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. User or privileged 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).
		00001b to 11111b	The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
15–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	EV_DR		Event Group Digital Conversion Result.

15.11.35 ADC Group1 Results FIFO (ADG1BUFFER)

Figure 15-59 and Table 15-42 describe the ADG1BUFFER register.

Figure 15-59. ADC Group1 Results FIFO (ADG1BUFFER) [offset = 0xB0 - 0xCF]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G1_EMPTY	Reserved										G1_CHID				
R-1	R-0										RW-U				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				G1_DR											
R-0				R-U											

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Group1 results FIFO location is aliased eight times, so that any 32-bit 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.

Table 15-42. ADC Group1 Results FIFO (ADG1BUFFER) Field Descriptions

Bit	Name	Value	Description
31	G1_EMPTY		Group1 FIFO Empty. This bit is applicable only when the “read from FIFO” mode is used for reading the Group1 conversion results. User or privileged 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.
30–21	Reserved	0	Reads return zeros, writes have no effect.
20–16	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. User or privileged 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).
		00001b to 11111b	The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
15–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G1_DR		Group1 Digital Conversion Result.

15.11.36 ADC Group2 Results FIFO (ADG2BUFFER)

Figure 15-59 and Table 15-42 describe the ADG2BUFFER register.

Figure 15-60. ADC Group2 Results FIFO (ADG2BUFFER) [offset = 0xD0 - 0xEF]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G2_EMPTY	Reserved										G2_CHID				
R-1	R-0										RW-U				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				G2_DR											
R-0				R-U											

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

The Group2 results FIFO location is aliased eight times, so that any 32-bit 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.

Table 15-43. ADC Group2 Results FIFO (ADG2BUFFER) Field Descriptions

Bit	Name	Value	Description
31	G2_EMPTY		Group2 FIFO Empty. This bit is applicable only when the “read from FIFO” mode is used for reading the Group2 conversion results. User or privileged 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 Group1 results memory.
30–21	Reserved	0	Reads return zeros, writes have no effect.
20–16	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. User or privileged 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).
		00001b to 11111b	The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field (e.g. 00001b means the conversion result is from the ADC input channel 1).
15–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	G2_DR		Group2 Digital Conversion Result.

15.11.37 ADC Event Group Results Emulation FIFO (ADEVEMUBUFFER) [offset = 0xF0]

Reading this register returns an identical result as reading the ADEVBUFFER but that it doesn't affect the status flags in the Event Group interrupt flag register or the Event Group status register, and the FIFO threshold counter. Please refer to the ADEVBUFFER register for bit field information.

15.11.38 ADC Group1 Results Emulation FIFO (ADG1EMUBUFFER) [offset = 0xF4]

Reading this register returns an identical result as reading the ADG1BUFFER but that it doesn't affect the status flags in the Group1 interrupt flag register or the Group1 status register, and the FIFO threshold counter. Please refer to the ADG1BUFFER register for bit field information.

15.11.39 ADC Group2 Results Emulation FIFO (ADG2EMUBUFFER) [offset = 0xF8]

Reading this register returns an identical result as reading the ADG2BUFFER but that it doesn't affect the status flags in the Group2 interrupt flag register or the Group2 status register, and the FIFO threshold counter. Please refer to the ADG2BUFFER register for bit field information.

15.11.40 ADC ADEVT Pin Direction Control Register (ADEVTDIR)

Figure 15-61 and Table 15-44 describe the ADEVTDIR register.

Figure 15-61. ADC ADEVT Pin Direction Control Register (ADEVTDIR) [offset = 0xFC]

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
Reserved															ADEVT_DIR
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-44. ADC ADEVT Pin Direction Control Register (ADEVTDIR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_DIR		ADEVT Pin Direction. Any operating mode read/write:
		0	The ADEVT pin is an input. **Note: If the pin direction is set as an input , the output buffer is tristated.**
		1	The ADEVT pin is an output. **Note: The input buffer is always enabled except for the situation stated in Section 15.10.2.**

15.11.41 ADC ADEVT Pin Output Value Control Register (ADEVTOUT)

Figure 15-62 and Table 15-45 describe the ADEVTOUT register.

Figure 15-62. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) [offset = 0x100]

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
Reserved															ADEVT_ OUT
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-45. ADC ADEVT Pin Output Value Control Register (ADEVTOUT) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, 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	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 ADEVTPDR bit = 1 and ADEVTOUT bit = 1. **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.11.42 ADC ADEVT Pin Input Value Register (ADEVTIN)

Figure 15-63 and Table 15-46 describe the ADEVTOUT register.

Figure 15-63. ADC ADEVT Pin Input Value Register (ADEVTIN) [offset = 0x104]

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
Reserved															ADEVT_IN
R-0															R-U

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-46. ADC ADEVT Pin Input Value Register (ADEVTIN) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, 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	The pin is at logic low (0).
		1	The pin is at logic high (1).

15.11.43 ADC ADEVT Pin Set Register (ADEVTSET)

Figure 15-64 and Table 15-47 describe the ADEVTSET register.

Figure 15-64. ADC ADEVT Pin Set Register (ADEVTSET) [offset = 0x108]

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
Reserved															ADEVT_ SET
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-47. ADC ADEVT Pin Set Register (ADEVTSET) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_SET		ADEVT Pin Set. This bit drives the output of the ADEVT pin high. Any operating mode read/write:
		0	*Write:* Writing a zero has no effect.
		1	*Write:* ADEVTOUT[0] is driven to logic high. **Note: The current logic state of the ADEVTOUT[0] bit will also be displayed by this bit.** **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.11.44 ADC ADEVT Pin Clear Register (ADEVTCLR)

Figure 15-65 and Table 15-48 describe the ADEVTCLR register.

Figure 15-65. ADC ADEVT Pin Clear Register (ADEVTCLR) [offset = 0x10C]

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
Reserved															ADEVT_CLR
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-48. ADC ADEVT Pin Clear Register (ADEVTCLR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_CLR		ADEVT Pin Clear. This bit drives the output of the ADEVT pin low. Any operating mode read/write:
		0	*Write:* Writing a zero has no effect.
		1	*Write:* ADEVTOUT[0] is driven to logic low. **Note: The current logic state of the ADEVTOUT[0] bit will also be displayed by this bit.** **Note: ADEVT pin is placed in output mode by setting the ADEVTDIR bit to 1.**

15.11.45 ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR)

Figure 15-66 and Table 15-49 describe the ADEVTPDR register.

Figure 15-66. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) [offset = 0x110]

31–16
Reserved
R-0

15–1	0
Reserved	ADEVT_PDR
R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-49. ADC ADEVT Pin Open Drain Enable Register (ADEVTPDR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_PDR		ADEVT Pin Open Drain Enable. Any operating mode read/write:
		0	The ADEVT pin is configured in push/pull (normal GIO) mode. The output voltage is driven to V_{OL} or lower if ADEVTOUT[0] =0 and V_{OH} or higher if ADEVTOUT[0] bit =1.
		1	The ADEVT pin is configured in open drain mode. The ADEVTOUT[0] bit controls the state of the ADEVT output buffer: ADEVTOUT[0] = 0 The ADEVT output buffer is driven low; ADEVTOUT[0] = 1 The ADEVT output buffer is tristated.

15.11.46 ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS)

Figure 15-67 and Table 15-50 describe the ADEVTPDIS register.

Figure 15-67. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) [offset = 0x114]

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
Reserved															ADEVT_PDIS
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-50. ADC ADEVT Pin Pull Control Disable Register (ADEVTPDIS) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, 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	The pull functionality is enabled.
		1	The pull functionality is disabled.

15.11.47 ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL)

Figure 15-68 and Table 15-51 describe the ADEVTPSEL register.

Figure 15-68. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) [offset = 0x118]

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
Reserved															ADEVT_PSEL
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-51. ADC ADEVT Pin Pull Control Select Register (ADEVTPSEL) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	ADEVT_PSEL		ADEVT Pin Pull Control Select.
			Any operating mode read/write:
		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 ADEVTPDIS to 0.

15.11.48 ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN)

Figure 15-69 and Table 15-52 describe the ADEVSAMPDISEN register.

Figure 15-69. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) [offset = 0x11C]

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
EV_SAMP_DIS_CYC								Reserved							EV_SAMP_DIS_EN
RW-0								R-0							RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-52. ADC Event Group Sample Cap Discharge Control Register (ADEVSAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	EV_SAMP_DIS_CYC		Event Group sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (EV_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	EV_SAMP_DIS_EN		Event Group sample cap discharge enable. User or privileged 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.

15.11.49 ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN)

Figure 15-70 and Table 15-53 describe the ADG1SAMPDISEN register.

Figure 15-70. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) [offset = 0x120]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">G1_SAMP_DIS_CYC</td><td colspan="7">Reserved</td><td>G1_SAMP_DIS_EN</td></tr>
<tr><td colspan="8">RW-0</td><td colspan="7">R-0</td><td>RW-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-53. ADC Group1 Sample Cap Discharge Control Register (ADG1SAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	G1_SAMP_DIS_CYC		Group1 sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (G1_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	G1_SAMP_DIS_EN		Group1 sample cap discharge enable. User or privileged 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.

15.11.50 ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN)

Figure 15-71 and Table 15-54 describe the ADG2SAMPDISEN register.

Figure 15-71. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) [offset = 0x124]

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
G2_SAMP_DIS_CYC								Reserved							G2_SAMP_DIS_EN
RW-0								R-0							RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-54. ADC Group2 Sample Cap Discharge Control Register (ADG2SAMPDISEN) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15–8	G2_SAMP_DIS_CYC		Group2 sample cap discharge cycles. These bits specify the ADC internal sampling capacitor discharge duration before sampling the input channel voltage: (G2_SAMP_DIS_CYC) * $t_{C(ADCLK)}$. where $t_{C(ADCLK)}$ is the ADCLK clock period in ns.
7–1	Reserved	0	Reads return zeros, writes have no effect.
0	G2_SAMP_DIS_EN		Group2 sample cap discharge enable.
			User or privileged 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.

15.11.51 ADC Magnitude Compare Interrupt Control Registers (ADMAGINTxCR)

Figure 15-72 and Table 15-55 describe the ADMAGINTxCR registers. This device supports three magnitude compare interrupts.

Figure 15-72. ADC Magnitude Compare Interruptx Control Registers (ADMAGINTxCR) [offset = 0x128, 0x130 and 0x138]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MAG_THRx[11:0]											
R-0				RW-0											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHN/THR_COMPx	CMP_GE/LTx	Res	COMP_CHIDx[4:0]					Reserved			MAG_CHIDx[4:0]				
RW-0	RW-0	R-0	RW-0					R-0			RW-0				

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-55. ADC Magnitude Compare Interruptx Control Register (ADMAGINTxCR) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved	0	Reads return zeros, writes have no effect.
27–16	MAG_THRx[11:0]		These bits specify the 12-bit compare value which the ADC will use for the comparison with the MAG_CHIDx channel's conversion result.
15	CHN/ THR_COMPx		Channel OR Threshold comparison. User or privileged 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.
14	CMP_GE/LTx		"Greater than or equal to" OR "Less than" comparison operator. User or privileged 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).
13	Reserved	0	Reads return zeros, writes have no effect.
12–8	COMP_CHIDx[4:0]		These bits specify the channel number from 0 to 15 whose last conversion result is compared with the MAG_CHIDx channel's conversion result.

Table 15-55. ADC Magnitude Compare Interruptx Control Register (ADMAGINTxCR) Field Descriptions

Bit	Name	Value	Description
7–5	Reserved	0	Reads return zeros, writes have no effect.
4–0	MAG_CHIDx[4:0]		These bits specify the channel number from 0 to 15 for which the conversion result needs to be monitored by the ADC.

15.11.52 ADC Magnitude Compare Mask (ADMAGxMASK)

Figure 15-73 and Table 15-56 describe the ADMAGxMASK registers. There are three such registers for the three magnitude compare interrupts.

Figure 15-73. ADC Magnitude Compare Interrupt Mask (ADMAGxMASK) [offset = 0x12C, 0x134 and 0x13C]

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
Reserved				MAGx_MASK[11:0]											
R-0				RW-0											

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-56. ADC Magnitude Compare Interrupt Mask (ADMAGINTxMASK) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–0	MAGx_MASK[11:0]		These bits specify the mask for the comparison in order to generate the magnitude compare interrupt # x.
			User or privileged 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.

15.11.53 ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET)

Figure 15-74 and Table 15-57 describe the ADMAGINTENASET register.

Figure 15-74. ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET) [offset = 0x158]

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
Reserved													MAG_INT_ENA_SET[2:0]		
R-0													RW-0		

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-57. ADC Magnitude Compare Interrupt Enable Set (ADMAGINTENASET) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
2–0	MAG_INT_ENA_SET[2:0]		Each of these three bits, when set, enable the corresponding magnitude compare interrupt. Bit0 controls magnitude compare interrupt 1. Bit1 controls magnitude compare interrupt 2. Bit2 controls magnitude compare interrupt 3. 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.

15.11.54 ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR)

Figure 15-75 and Table 15-58 describe the ADMAGINTENACLR register.

Figure 15-75. ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR) [offset = 0x15C]

31–16
Reserved
R-0

15–3	2–0
Reserved	MAG_INT_ENA_CLR[2:0]
R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-58. ADC Magnitude Compare Interrupt Enable Clear (ADMAGINTENACLR) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
2–0	MAG_INT_ENA_ CLR[2:0]		Each of these three bits, when set, disable the corresponding magnitude compare interrupt. Bit0 controls magnitude compare interrupt 1. Bit1 controls magnitude compare interrupt 2. Bit2 controls magnitude compare interrupt 3. 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.

15.11.55 ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG)

Figure 15-76 and Table 15-59 describe the ADMAGINTFLG register.

Figure 15-76. ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG) [offset = 0x160]

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
Reserved													MAG_INT_FLG[2:0]		
R-0													RW-0		

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-59. ADC Magnitude Compare Interrupt Flag (ADMAGINTFLG) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved	0	Reads return zeros, writes have no effect.
2–0	MAG_INT_FLG [2:0]		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. Bit0 is the flag of magnitude compare interrupt 1. Bit1 is the flag of magnitude compare interrupt 2. Bit2 is the flag of magnitude compare interrupt 3. Any operation mode, for each bit:
		0	Read: The condition for the corresponding magnitude threshold interrupt has NOT been met. Write: The corresponding flag is left unchanged.
		1	Read: The condition for the corresponding magnitude threshold interrupt has been met. Write: The corresponding flag is cleared. The flag can also be cleared by reading from the magnitude compare interrupt offset register.

15.11.56 ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF)

Figure 15-77 and Table 15-60 describe the ADMAGINTOFF register.

Figure 15-77. ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF) [offset = 0x164]

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
Reserved												MAG_INT_OFF[3:0]			
R-0												RC-0			

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-60. ADC Magnitude Compare Interrupt Offset (ADMAGINTOFF) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros, writes have no effect.
3–0	MAG_INT_OFF [3:0]		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. User or privileged mode read:
		0000b	No magnitude compare interrupt is pending.
		0001b	Magnitude compare interrupt # 1 is pending.
		0010b	Magnitude compare interrupt # 2 is pending.
		0011b	Magnitude compare interrupt # 3 is pending.
		0100b	Invalid
		0101b	Invalid
		0110b	Invalid
		0111b	Invalid

15.11.57 ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR)

Figure 15-78 and Table 15-61 describe the ADEVFIFORESETCR register.

Figure 15-78. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) [offset = 0x168]

31–16
Reserved
R-0

15–1	0
Reserved	EV_FIFO_RESET
R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-61. ADC Event Group FIFO Reset Control Register (ADEVFIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, 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’.

15.11.58 ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR)

Figure 15-79 and Table 15-62 describe the ADG1FIFORESETCR register.

Figure 15-79. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) [offset = 0x16C]

31–16
Reserved
R-0

15–1	0
Reserved	G1_FIFO_RESET
R-0	RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-62. ADC Group1 FIFO Reset Control Register (ADG1FIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	G1_FIFO_RESET		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’.

15.11.59 ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR)

Figure 15-80 and Table 15-63 describe the ADG2FIFORESETCR register.

Figure 15-80. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) [offset = 0x170]

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
Reserved															G2_FIFO_RESET
R-0															RW-0

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-63. ADC Group2 FIFO Reset Control Register (ADG2FIFORESETCR) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved	0	Reads return zeros, writes have no effect.
0	G2_FIFO_RESET		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’.

15.11.60 ADC Event Group RAM Write Address (ADEVRAMWRADDR)

Figure 15-81 and Table 15-64 describe the ADEVRAMWRADDR register.

Figure 15-81. ADC Event Group RAM Write Address (ADEVRAMWRADDR) [offset = 0x174]

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
Reserved							EV_RAM_ADDR[8:0]								
R-0							R-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-64. ADC Event Group RAM Write Address (ADEVRAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	EV_RAM_ADDR [8:0]		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 till that time.

15.11.61 ADC Group1 RAM Write Address (ADG1RAMWRADDR)

Figure 15-82 and Table 15-65 describe the ADG1RAMWRADDR register.

Figure 15-82. ADC Group1 RAM Write Address (ADG1RAMWRADDR) [offset = 0x178]

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
Reserved							G1_RAM_ADDR[8:0]								
R-0							R-0								

R = Read Only, RW = Read/Write, RC = Read/Clear, *-n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-65. ADC Group1 RAM Write Address (ADG1RAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	G1_RAM_ADDR [8:0]		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 till that time.

15.11.62 ADC Group2 RAM Write Address (ADG2RAMWRADDR)

Figure 15-83 and Table 15-66 describe the ADG2RAMWRADDR register.

Figure 15-83. ADC Group2 RAM Write Address (ADG2RAMWRADDR) [offset = 0x17C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">G2_RAM_ADDR[8:0]</td></tr>
<tr><td colspan="7">R-0</td><td colspan="9">R-0</td></tr>
</table>

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-66. ADC Group2 RAM Write Address (ADG2RAMWRADDR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8–0	G2_RAM_ADDR [8:0]		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 till that time.

15.11.63 ADC Parity Control Register (ADPARCR)

Figure 15-84 and Table 15-67 describe the ADPARCR register.

Figure 15-84. ADC Parity Control Register (ADPARCR) [offset = 0x180]

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
Reserved							TEST	Reserved				PARITY_ENA [3:0]			
R-0							RWP-0	R-0				RWP-0101			

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-67. ADC Parity Control Register (ADPARCR) Field Descriptions

Bit	Name	Value	Description
31–9	Reserved	0	Reads return zeros, writes have no effect.
8	TEST		This bit maps the parity bits into the ADC results FIFO 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 zeros, writes have no effect.
3–0	PARITY_ENA [3:0]		Enable/disable parity checking. These bits enable/disable 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 ESM module. Any operation mode read, privileged mode write:
		0101b	Parity check is disabled.
		Any other	Parity check is enabled. **Note: It is recommended to write 1010b to enable parity, to guard against soft error from flipping this field to a disabled state.**

15.11.64 ADC Parity Error Address (ADPARADDR)

Figure 15-85 and Table 15-68 describe the ADPARCR register.

Figure 15-85. ADC Parity Error Address (ADPARADDR) [offset = 0x184]

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
Reserved				ERROR_ADDRESS [9:0]										Reserved	
R-0				R-U										R-0	

R = Read Only, RW = Read/Write, RC = Read/Clear, -*n* = value after reset, RWP = Read in all modes, write in privileged mode only

Table 15-68. ADC Parity Error Address (ADPARADDR) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved	0	Reads return zeros, writes have no effect.
11–2	ERROR_ADDRESS [9:0]		These bits hold the address of the first parity error generated in the ADC results FIFO RAM. This error address is frozen from being updated until it is read by the application. During debugger memory reads, this address is maintained frozen even when read.
1–0	Reserved	0	Reads return zeros, writes have no effect. Reading [11:0] provides the 32-bit aligned address.

Chapter 16

SPNU489C-February 2012

Controller Area Network (DCAN) Module

16.1 Overview

This reference guide contains a general description of the DCAN Controller Area Network module. The Controller Area Network is a high integrity serial communications protocol for distributed real-time applications.

The DCAN module supports bit rates up to 1 Mbit/s and is compliant to the CAN 2.0B protocol specification.

This device has 3 DAN modules , referred as DCAN1,DCAN2 and DCAN3 in this document.

16.1.1 Features

The DCAN implements the following features:

- Supports CAN protocol version 2.0 part A, B
- Bit rates up to 1 MBit/s
- Dual clock source
- 32 and 64 message objects (Refer section 16.1.3.3)
- Individual identifier masks for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- Message RAM parity check mechanism
- Direct access to Message RAM during test mode
- CAN Rx / Tx pins configurable as general purpose IO pins
- Supports Two interrupt lines - Level 0 and Level 1.
- Global power down and wakeup support
- Local power down and wakeup support
- Automatic RAM initialization
- Supports DMA access.

16.1.2 Functional Description

The DCAN performs CAN protocol communication according to ISO 11898-1. The bit rate can be programmed to values up to 1 MBit/s. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

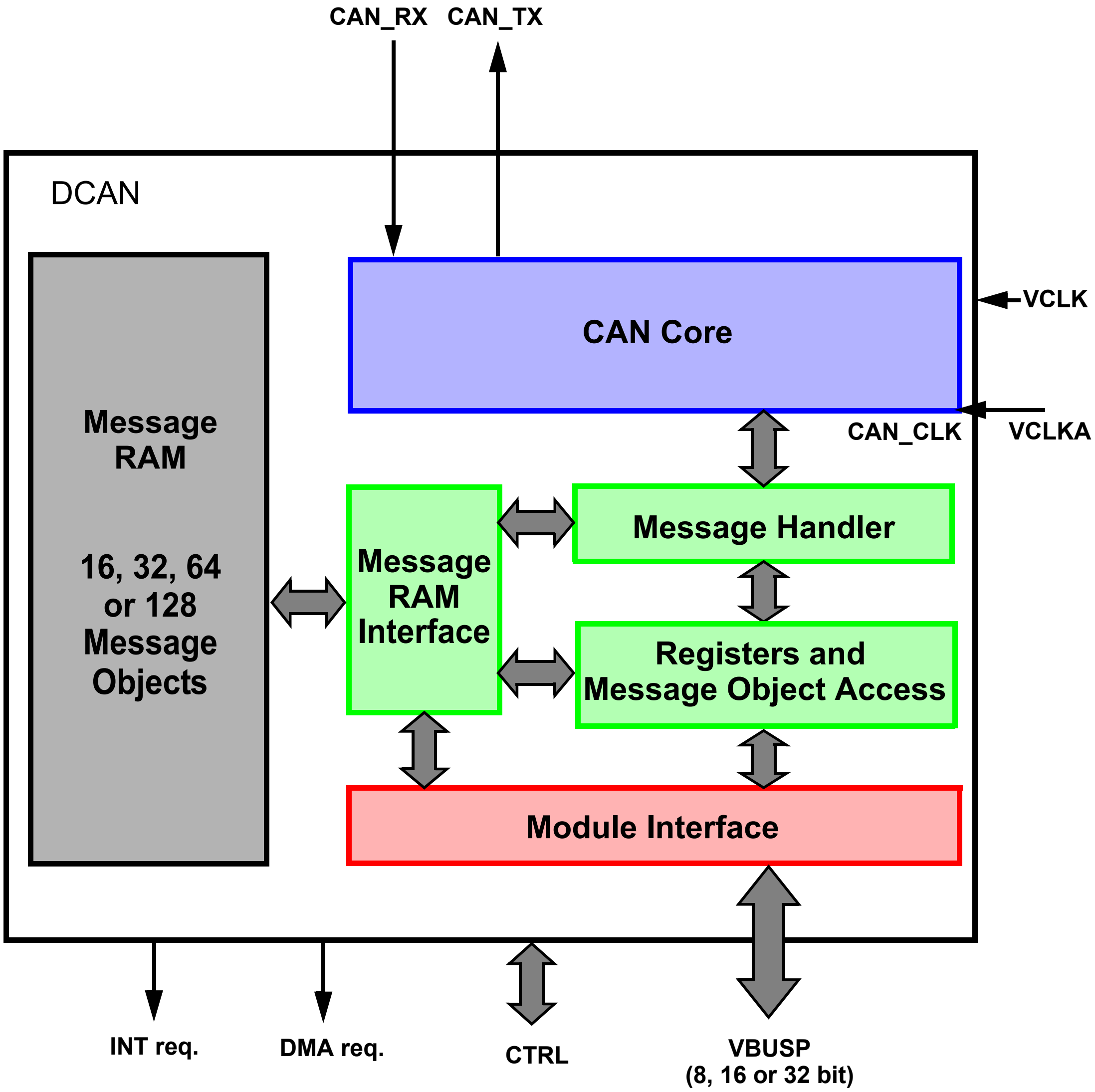
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, the handling of transmission requests as well as the generation of interrupts or DMA requests.

The register set of the DCAN can be accessed directly by the CPU via the module interface. These registers are used to control and configure the CAN Core and the Message Handler, and to access the Message RAM.

16.1.3 DCAN Block Diagram

Figure 16-1. Block Diagram



16.1.3.1 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.

16.1.3.2 Message Handler

The Message Handler is a state machine which 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/DMA request generation as programmed in the control registers.

16.1.3.3 Message RAM

The DCAN1 and DCAN2 Message RAM enables storage of 64 CAN messages and DCAN3 Message RAM enables storage of 32 CAN messages.

16.1.3.4 Message RAM Interface

Three Interface Register sets control the CPU read and write accesses to the Message RAM. There are two Interface Registers sets for read and write access (IF1 and IF2) and one Interface Register set for read access only (IF3). Additional information can be found in section 16.12.

The Interface Registers have the same word-length as the Message RAM.

16.1.3.5 Registers and Message Object access

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the Message RAM are done through Interface Registers. In a dedicated test mode, the Message RAM is memory mapped and can be directly accessed by either CPU or DMA.

16.1.3.6 Module Interface

The DCAN module registers are accessed by the CPU or user software through a 32bit peripheral bus interface.

16.1.3.7 Dual clock source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (VCLK) and the peripheral asynchronous clock source domain (VCLKA) for CAN_CLK.

If a frequency modulated clock output from FMzPLL is used as the VCLK source, then VCLKA should be derived from an unmodulated clock source (e.g. OSCIN source). Additional information can be found in section 16.3.

The clock source for VCLKA is selected by the Peripheral Asynchronous Clock Source Register in the system module (Additional information can be found in Architecture Chapter section 16.12).

16.2 CAN Operation

After hardware reset, the Init bit in the CAN Control Register is set and all CAN protocol functions are disabled.

The CAN module must be initialized before operating it.

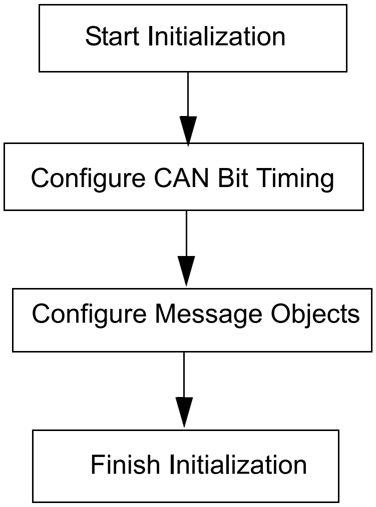
16.2.1 CAN Module Initialization

A general CAN module initialization would mean the following two critical steps:

1. Configuration of the CAN Bit Timing
2. Configuration of Message objects.

To initialize the CAN Controller, the CPU has to set up the CAN Bit timing and those message objects which have to be used for CAN communication. Message objects that are not needed, can be deactivated .

Figure 16-2. CAN module general initialization flow



16.2.1.1 Configuration of CAN bit Timing

The CAN module must be in initialization mode to configure the CAN bit timing.

Refer Figure 16-3 for CAN bit timing software configuration flow.

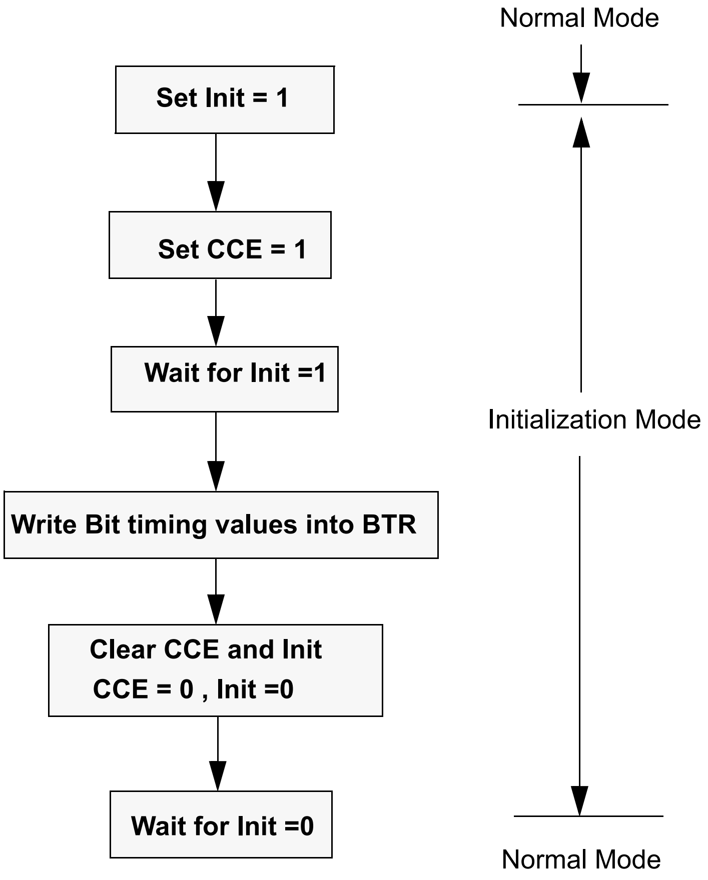
Step 1: Enter "initialization mode" by setting the Init (Initialization) bit in the CAN Control Register.

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.

Figure 16-3. CAN Bit-timing configuration



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 section 16.11.2 for BTR value calculation for a given bit-timing.

Step 5: Clear the CCE and Init bit.

Step 6: Wait for the Init bit to clear. This would make sure that the module has come out of “initialization mode”.

Following this, the module comes to operation by synchronizing itself to the CAN bus (provided the BTR is configured as per the CAN bus baudrate). Although the Message Objects has to be configured before carrying out any communication.

Note:
The module would not come out of the “initialization mode” if any incorrect BTR values are written in step 4.

Note:
The required message objects should be configured as transmit or receive objects before start of data transfer as explained in section 16.2.1.

16.2.1.2 Configuration of Message Objects

The message objects can be configured only through the Interface registers and the cpu does not have direct access to the message object (Message RAM) . One has to know about the interface register set (IFx) usage (section 16.12)and the message object structure (section 16.13)before configuring the message objects.

Refer to section 16.9 to know about the procedure to configure the message objects. All the message objects should be configured to particular identifiers or set to not valid before the message transfer is started.

It is possible to change the configuration of message objects during normal operation (that is between data transfers).

Note:
The message objects initialization is independent of the bit-timing configuration.

16.2.1.3 DCAN RAM Hardware Initialization

If the memory hardware initialization for the DCAN module is enabled in the device system module control registers, the complete RAM can be initialized with zeros and the parity bits are set accordingly.

Initialization speed of DCAN RAM = (Number of message objects + 5) x (VBUS cycle time).

For more details on RAM hardware initialization , refer to the system module reference guide.

16.2.2 CAN Message Transfer (Normal Operation)

Once the DCAN is initialized and Init bit is reset to zero, 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 / IE1 are zero).

The CPU may poll all MessageObject’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 e.g. the identifier mask is used, the arbitration bits which 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.

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. 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.

16.2.2.1 Automatic Retransmission

According to the CAN Specification (ISO11898), the DCAN provides a mechanism to automatically retransmit frames which 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 16.10.3.

16.2.2.2 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 which 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 which 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.

16.2.3 Test Modes

The DCAN provides several test modes which are mainly intended for production tests or self test.

For all test modes, Test bit in the CAN Control Register needs to be set to one. This enables write access to the Test Register.

16.2.3.1 Silent Mode

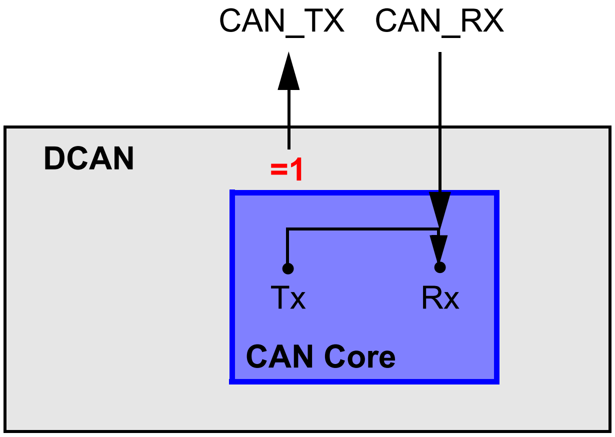
The Silent Mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (e.g. acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN Core.

Figure 16-4 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 one.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

Figure 16-4. *CAN Core in Silent Mode*



16.2.3.2 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.

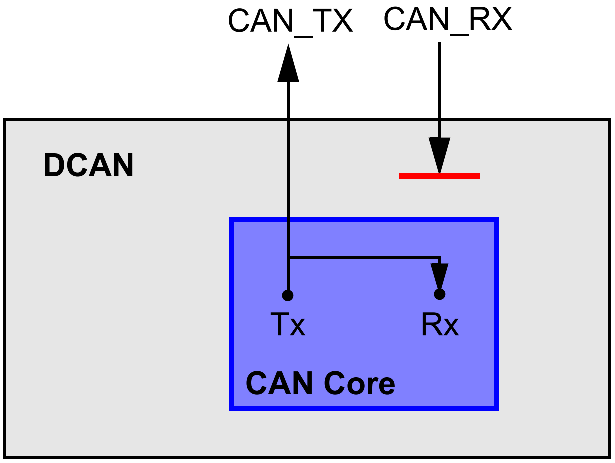
In order to be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode.

Figure 16-5 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 bit LBack in Test Register to one.

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 16.2.3.3).

Figure 16-5. *CAN Core in Loop Back Mode*



16.2.3.3 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.

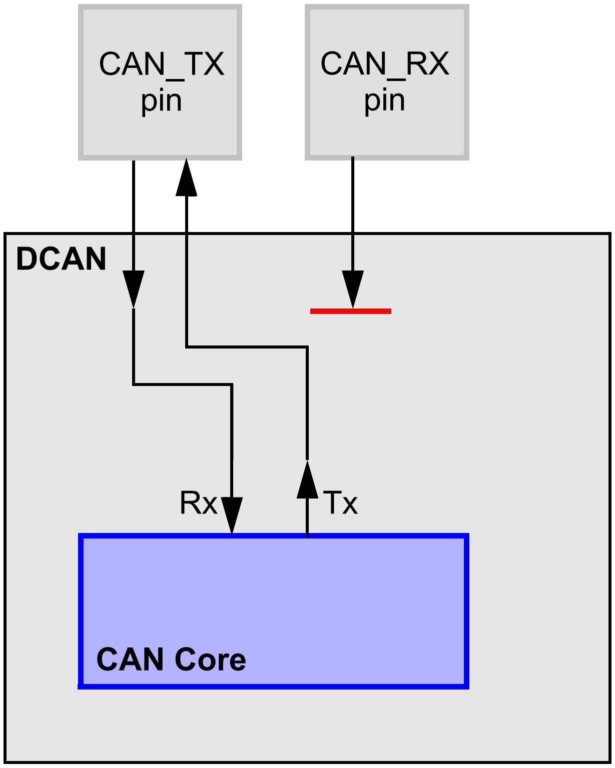
With this configuration, the Tx pin IO circuit can be tested.

External Loop Back Mode can be activated by setting bit **ExL** in Test Register to one.

Figure 16-6 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 set), the ExL bit will be ignored.

Figure 16-6. ***CAN Core in External Loop Back Mode***

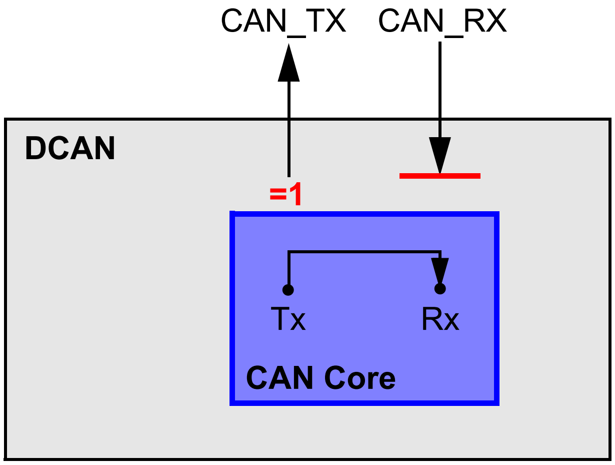


16.2.3.4 Loop Back combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by setting bits LBack and Silent at the same time. This mode can be used for a “Hot Selftest”, i.e. 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 16-7 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 16-7. *CAN Core in Loop Back combined with Silent Mode*



16.2.3.5 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 bits Tx[1:0] as described in section 16.15.6.

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.

16.3 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (VCLK) as the general module clock source, and the peripheral asynchronous clock source domain (VCLKA) provided to the CAN core (as clock source CAN_CLK) for generating the CAN Bit Timing.

Both clock domains can be derived from the same clock source (so that VCLK = VCLKA). However, if frequency modulation in the FMzPLL is enabled (spread spectrum clock), then due to the high precision clocking requirements of the CAN Core, the FMzPLL clock source should not be used for VCLKA. Alternatively, a separate clock without any modulation (e.g. derived directly from the OSCIN clock) should be used for VCLKA.

Please refer 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:

$$f_{0,\,VCLK} \pm \Delta f_{FM,\,VCLK} \geq f_{CANCLK}$$

Note:
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.

16.4 Interrupt functionality

Interrupts can be generated on two interrupt lines:

1. DCAN0INT line

2. DCAN1INT line.

These lines can be enabled by setting IE0 and IE1 bits respectively in CAN Control Register.

The DCAN provides three groups of interrupt sources: Message Object Interrupts, Status Change Interrupts and Error Interrupts.(see Figure 16-8 and Figure 16-9)

The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt Register (see section 16.15.5). 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 which 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.

16.4.1 Message Object interrupts

Message Object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE which are described in section 16.13.1.

Message Object interrupts can be routed to either DCAN0INT or DCAN1INT line, controlled by the Interrupt Multiplexer Register, see section 16.15.17.

16.4.2 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 which 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 16.5.2).

16.4.3 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 which has to be enabled by setting IE0 in the CAN Control Register.

Figure 16-8. *CAN Interrupt Topology 1*

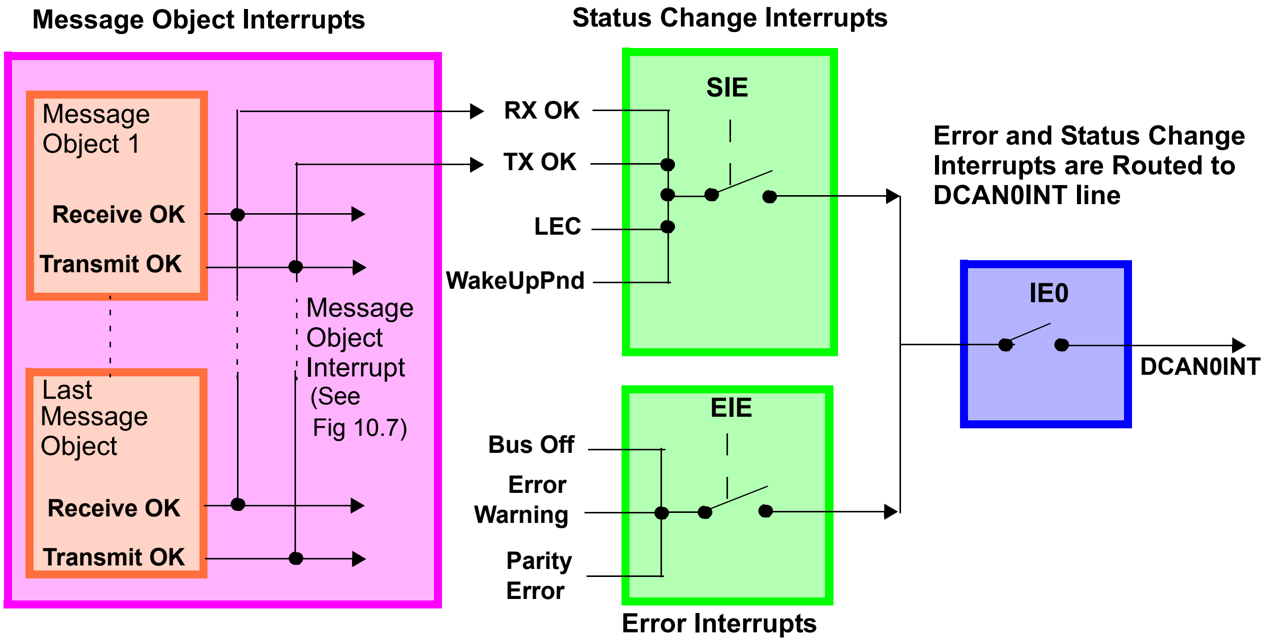
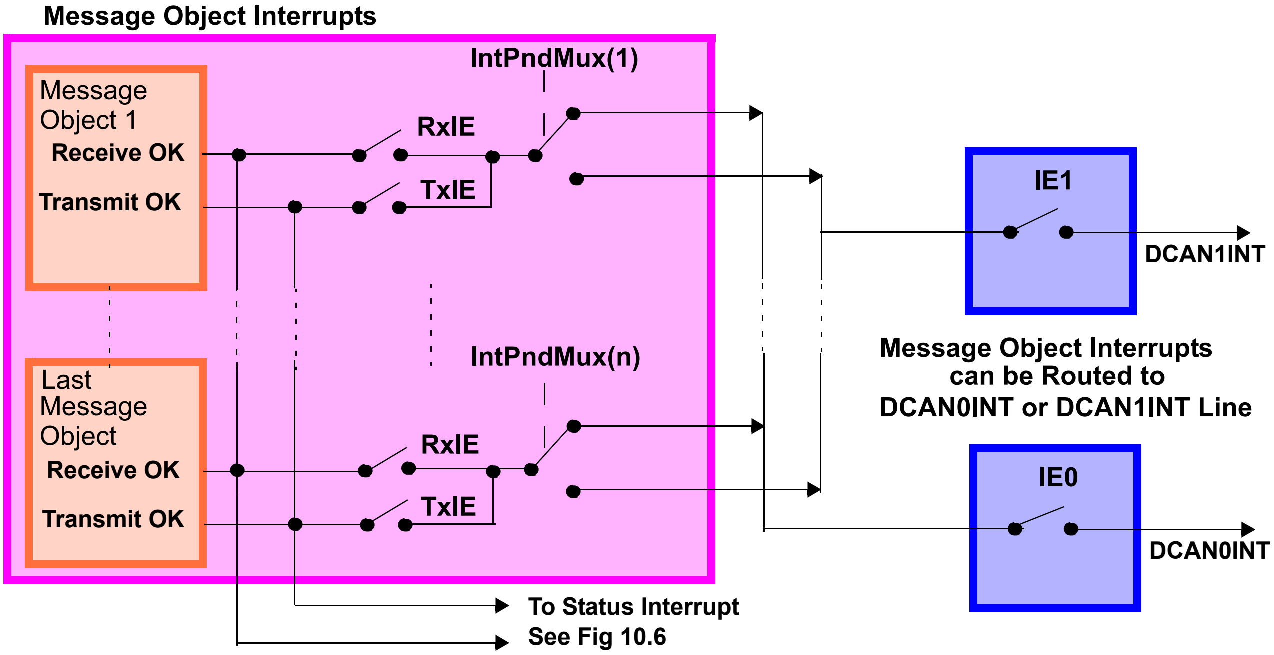


Figure 16-9. *CAN Interrupt Topology 2*



16.5 Global Power Down Mode

The TMSx70 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).

16.5.1 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.

16.5.2 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.

16.6 Local Power Down Mode

Besides from the centralized power down mechanism controlled by the PCR module (global power down, see section 16.5), the DCAN supports a local power down mode which can be controlled within the DCAN control registers.

16.6.1 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 16.6.2) which 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.

16.6.2 Wakeup from local power down

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.

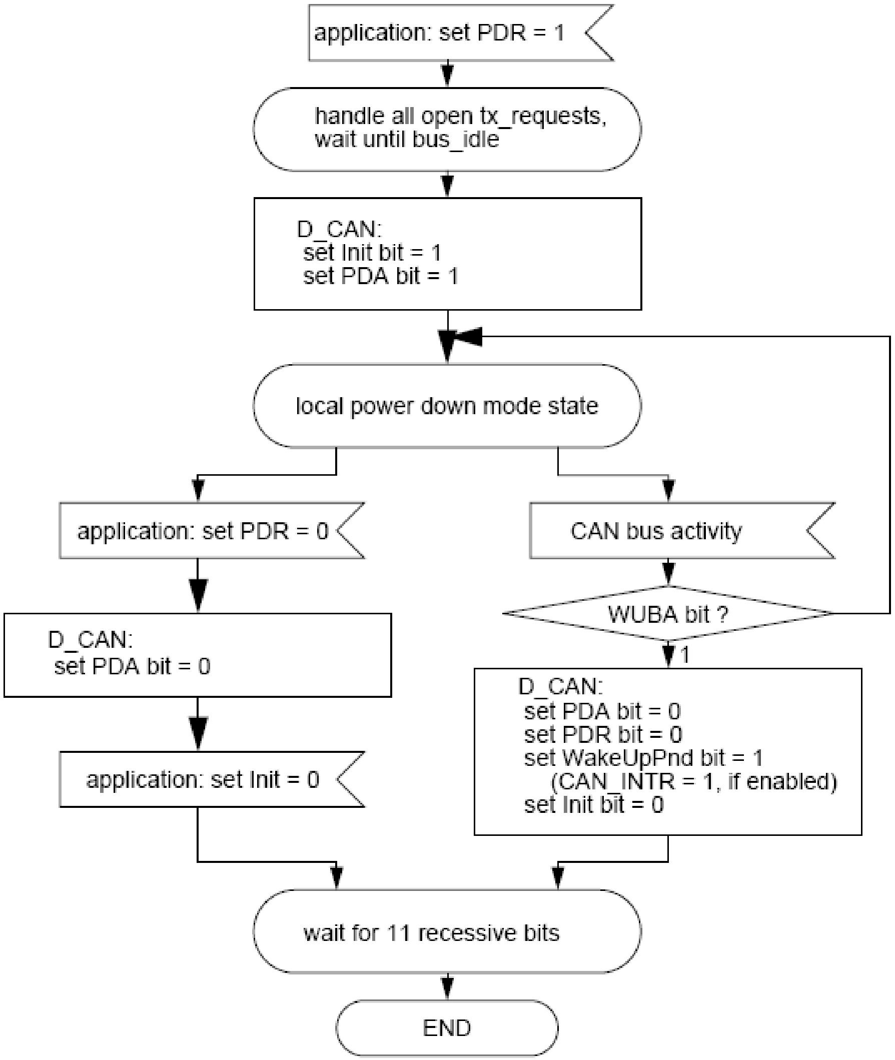
2. Alternatively, a CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control Register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wake up sequence. It will clear the PDR bit in 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 16-10 shows a flow diagram about entering and leaving local power down mode.

Figure 16-10. Local power down mode flow diagram


application: set PDR = 1
handle all open tx_requests, wait until bus_idle
D_CAN: set Init bit = 1 set PDA bit = 1
local power down mode state
application: set PDR = 0
CAN bus activity
WUBA bit ?
1
D_CAN: set PDA bit = 0
D_CAN: set PDA bit = 0 set PDR bit = 0 set WakeUpPnd bit = 1 (CAN_INTR = 1, if enabled) set Init bit = 0
application: set Init = 0
wait for 11 recessive bits
END

16.7 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 16.13.4.

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 16.13.3) or in RDA mode (see section 16.13.4). 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.

16.7.1 Behavior on parity error

On any read access to Message RAM, e.g. 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, e.g. by immediately checking the Parity Error Code register on parity error interrupt.

Note:
During RAM initialization, no parity check will be done.

16.7.2 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.

16.8 Debug/Suspend Mode

The module supports the usage of an external debug unit by providing functions like pausing DCAN activities and making Message RAM content accessible via VBUSP interface.

Before entering debug/suspend mode, the circuit will either wait until a started transmission or reception will be finished and Bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit IDS in CAN Control Register.

Afterwards, the DCAN enters debug/suspend mode, indicated by InitDbg flag in CAN Control Register.

During Debug/Suspend mode, all DCAN registers can be accessed. Reading reserved bits will return ‘0’. Writing to reserved bits will have no effect.

Also, the Message RAM will be memory mapped. This allows the external debug unit to read the Message RAM. For the memory organization, see section 16.13.3).

Note:
During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Note:
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 (clear of DMAActive flag by r/w)

16.9 Configuration of Message Objects

The whole Message RAM should be configured before the end of the initialization, however it is also possible to change the configuration of message objects during CAN communication.

The CAN software driver library should offer subroutines that:

- Transfer a complete message structure into a message object. (Configuration)
- Transfer the data bytes of a message into a message object and set TxRqst and NewDat. (Start a new transmission)
- Get the data bytes of a message from a message object and clear NewDat (and IntPnd). (Read received data)
- Get the complete message from a message object and clear NewDat (and IntPnd).
  (Read a received message, including identifier, from a message object with UMask = ‘1’)

Parameters of the subroutines are the Message Number and a pointer to a complete message structure or to the data bytes of a message structure.

Two examples of assigning the IFx Interface Register sets to these subroutines are shown here:

In the first method, the tasks of the application program that may access the module are divided in two groups. Each group is restricted to the use of one of the Interface Register sets. The tasks of one group may interrupt tasks of the other group, but not of the same group.

In the second method, which may be a special case of the first method, there are only two tasks in the application program that access the module. A Read_Message task that uses IF2 or IF3 to get received messages from the Message RAM and a Write_Message task that uses IF1 to write messages to be transmitted (or to be configured) into the Message RAM. Both tasks may interrupt each other.

16.9.1 Configuration of a Transmit Object for Data Frames

Figure 16-11 shows how a Transmit Object can be initialized.

Figure 16-11. 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 16.10.8. Identifier masking must be disabled (UMask = ‘0’) if no Remote Frames are allowed to set the TxRqst bit (RmtEn = ‘0’).

16.9.2 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.

16.9.3 Configuration of a Single Receive Object for Data Frames

Figure 16-12 shows how a Receive Object for Data Frames can be initialized.

Figure 16-12. 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.

16.9.4 Configuration of a Single Receive Object for Remote Frames

Figure 16-13 shows how a Receive Object for Remote Frames can be initialized.

Figure 16-13. 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 16.9.5.

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 16.10.8.

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.

16.9.5 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 one, configuring it as the end of the block.

16.10 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 to enable and request their transmission. The transmission is requested automatically when a matching Remote Frame is received.

The application may read messages which 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 interrupt-driven or after polling of NewDat.

16.10.1 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 above 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 e.g. about all pending transmission requests.

All Message Handler registers are read-only.

16.10.2 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 e.g. 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.

16.10.3 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.

16.10.4 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 16.10.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

16.10.5 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. Neither d nor TxRqst have to be reset before this operation.

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.

16.10.6 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.

16.10.7 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.

16.10.8 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.

16.10.9 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.

16.10.10 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.

16.10.11Storing 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.

16.10.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects which 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.

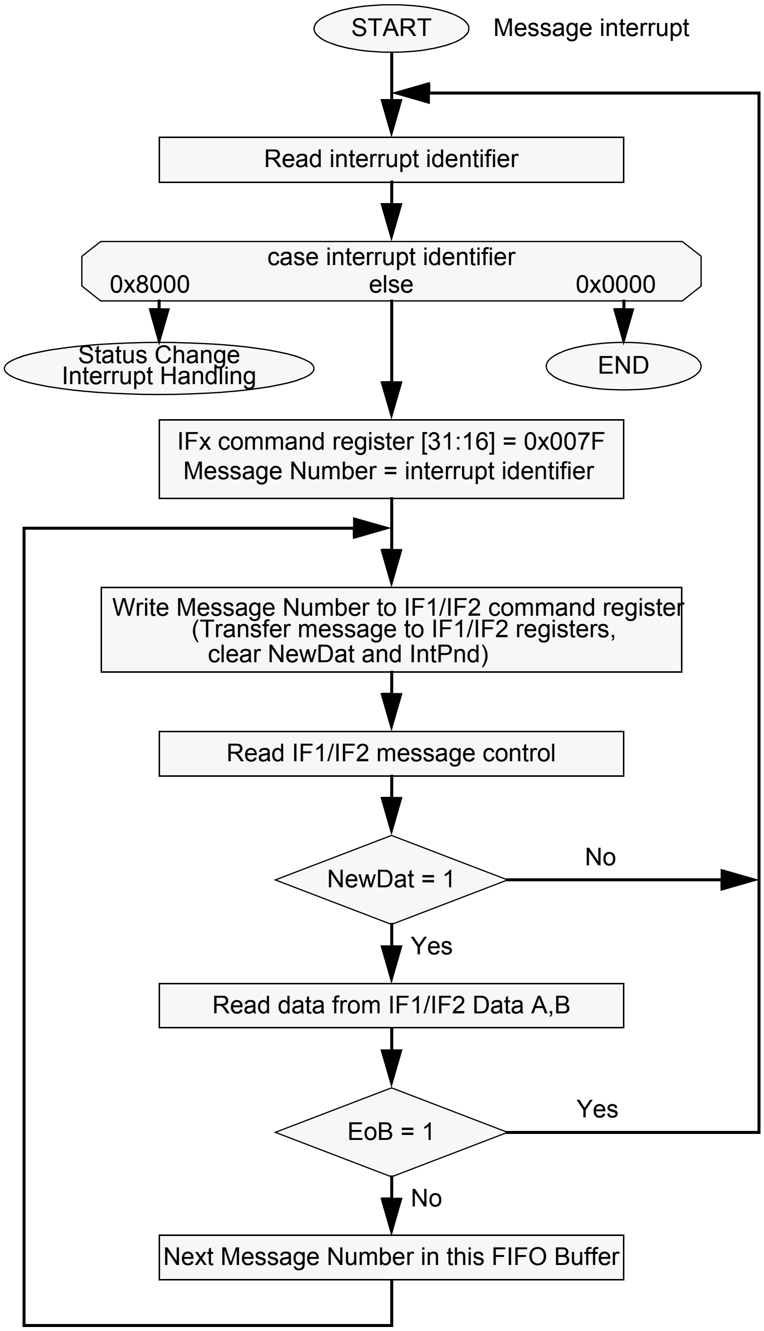
A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 16-14.

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 16-14. CPU Handling of a FIFO Buffer (Interrupt Driven)


START
Message interrupt
Read interrupt identifier
case interrupt identifier
0x8000
else
0x0000
Status Change Interrupt Handling
END
IFx command register [31:16] = 0x007F
Message Number = interrupt identifier
Write Message Number to IF1/IF2 command register (Transfer message to IF1/IF2 registers, clear NewDat and IntPnd)
Read IF1/IF2 message control
NewDat = 1
No
Yes
Read data from IF1/IF2 Data A,B
EoB = 1
Yes
No
Next Message Number in this FIFO Buffer

16.11 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.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

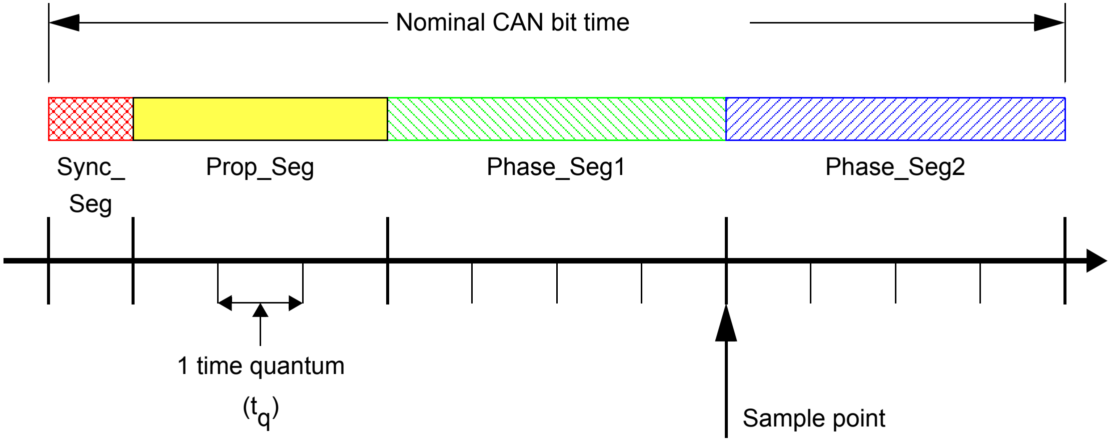
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

16.11.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 16-15):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 16-15. 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 16-1 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different Bit time configurations.

Table 16-1. 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

Note:
For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

16.11.1.1 Synchronization Segment

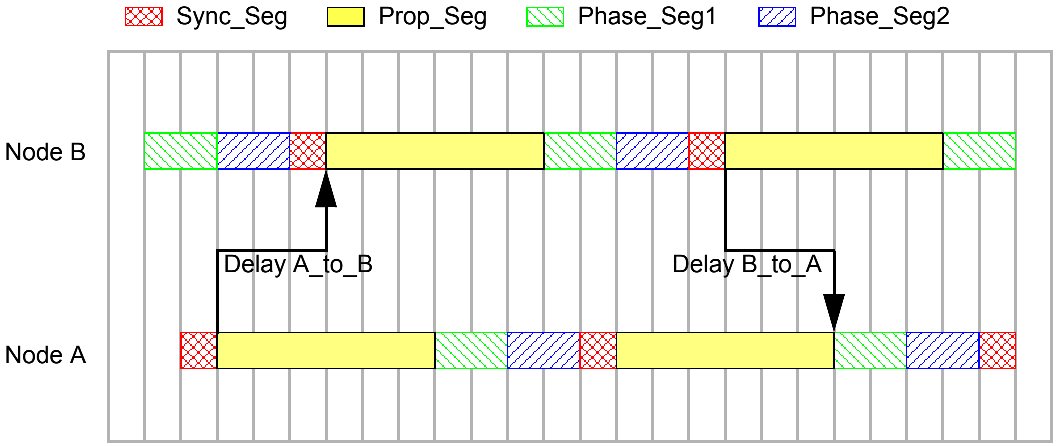
The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

16.11.1.2 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.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in figure 16-16 shows the phase shift and propagation times between two CAN nodes.

Figure 16-16. The Propagation Time Segment



Delay A_to_B >= node output delay(A) + bus line delay(AÆB) + node input delay(B)

Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 • [max(node output delay+ bus line delay + node input delay)]

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the Bit timing configuration (Prop_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The DCAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

16.11.1.3 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.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame only resynchronization is possible.

- Hard Synchronization
  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.
- Bit Resynchronizations
  Resynchronization leads to a shortening or lengthening of the Bit time such that the position of the sample point is shifted with regard to the edge.
  When the phase error of the edge which causes resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.
  When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

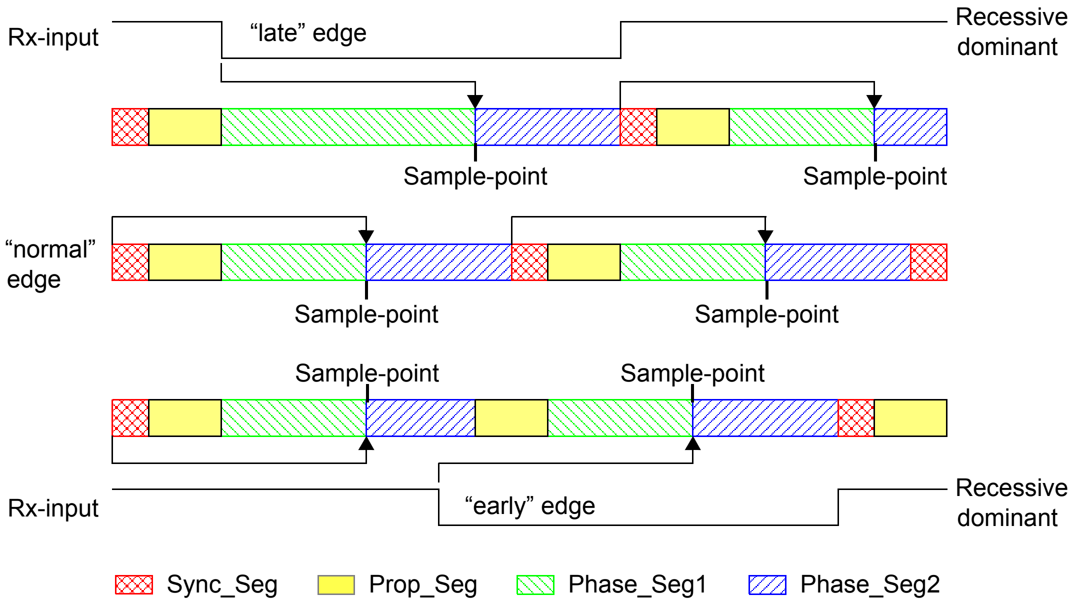
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

The examples in figure 16-17 show how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

Figure 16-17. Synchronization on Late and Early Edges



In the first example, an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync_Seg to the sample point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example, an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

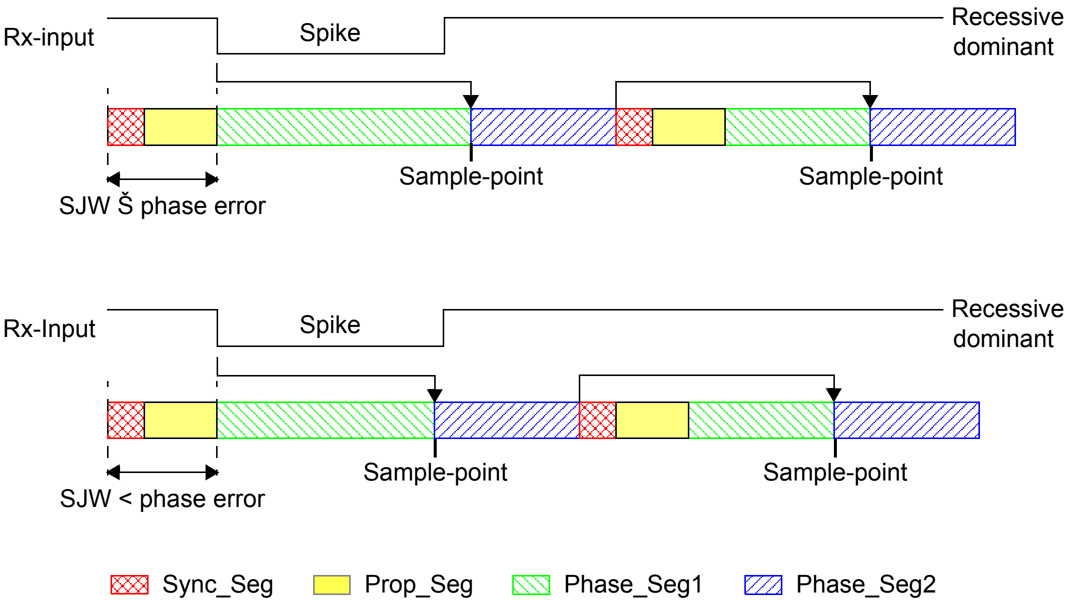
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync_Seg when synchronizing on an "*early*" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 16-18 show how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

Figure 16-18. Filtering of Short Dominant Spikes



16.11.1.4 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

$$(1 - df) \bullet f_{nom} \leq f_{osc} \leq (1 + df) \bullet f_{nom}$$

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):

$$\text{I:} \quad df \leq \frac{\min(TSeg1,\ TSeg2)}{2x(13x(bit\_time - TSeg2))}$$

$$\text{II:} \quad df \leq \frac{SJW}{20xbit\_time}$$

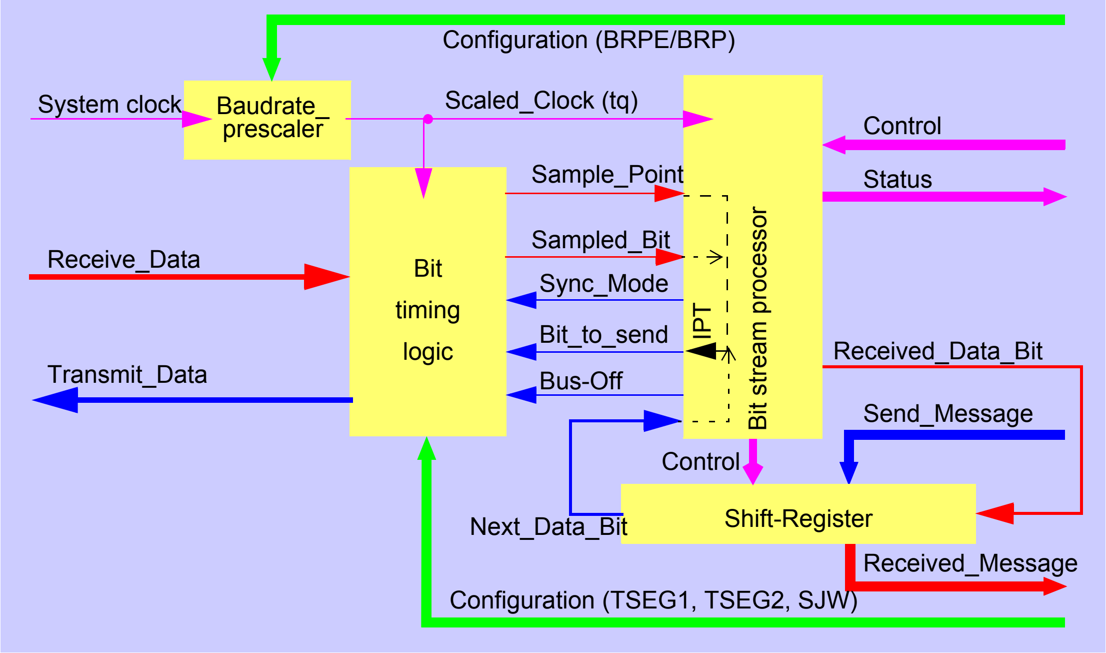
It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

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.

16.11.2 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 (see Figure 16-19).

Figure 16-19. Structure of the CAN Core's CAN Protocol Controller



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, e.g. 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.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the Bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine, is evaluated once each bit time, at the Sample Point.

The Shift Register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (e.g. data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT), which is 0 t_q for the DCAN.

Generally, the IPT is CAN controller specific, but may not be longer than 2 t_q. The IPC length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

16.11.2.1 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.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in section 16.11.2.3

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which 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

16.11.2.2 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	0.43	%	=	$\frac{\min(TSeg1,\ TSeg2)}{2x(13x(bit\_time - TSeg2))}$
			=	$\frac{0.1\mu s}{2x(13x(1\mu s - 0.1\mu s))}$

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= 0 x 00000700.

16.11.2.3 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	1.58	%	=	$\frac{\min(TSeg1,\ TSeg2)}{2x(13x(bit\_time - TSeg2))}$
			=	$\frac{3\mu s}{2x(13x(9\mu s - 3\mu s))}$

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= 0x000024C1.

16.12 Message Interface Register Sets

The Interface Register sets control the CPU read and write accesses to the Message RAM. There are two Interface Registers Sets for read / write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore the data transfer from the IF1/IF2 Registers to the Message RAM requires the Message Handler to perform a read-modify-write cycle: First those parts of the message object that are not to be changed are read from the Message RAM into the Interface Register set, and after the update the whole content of the Interface Register set is written into the message object.

After the partial write of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be set to the actual contents of the selected message object.

After the partial read of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be left unchanged.

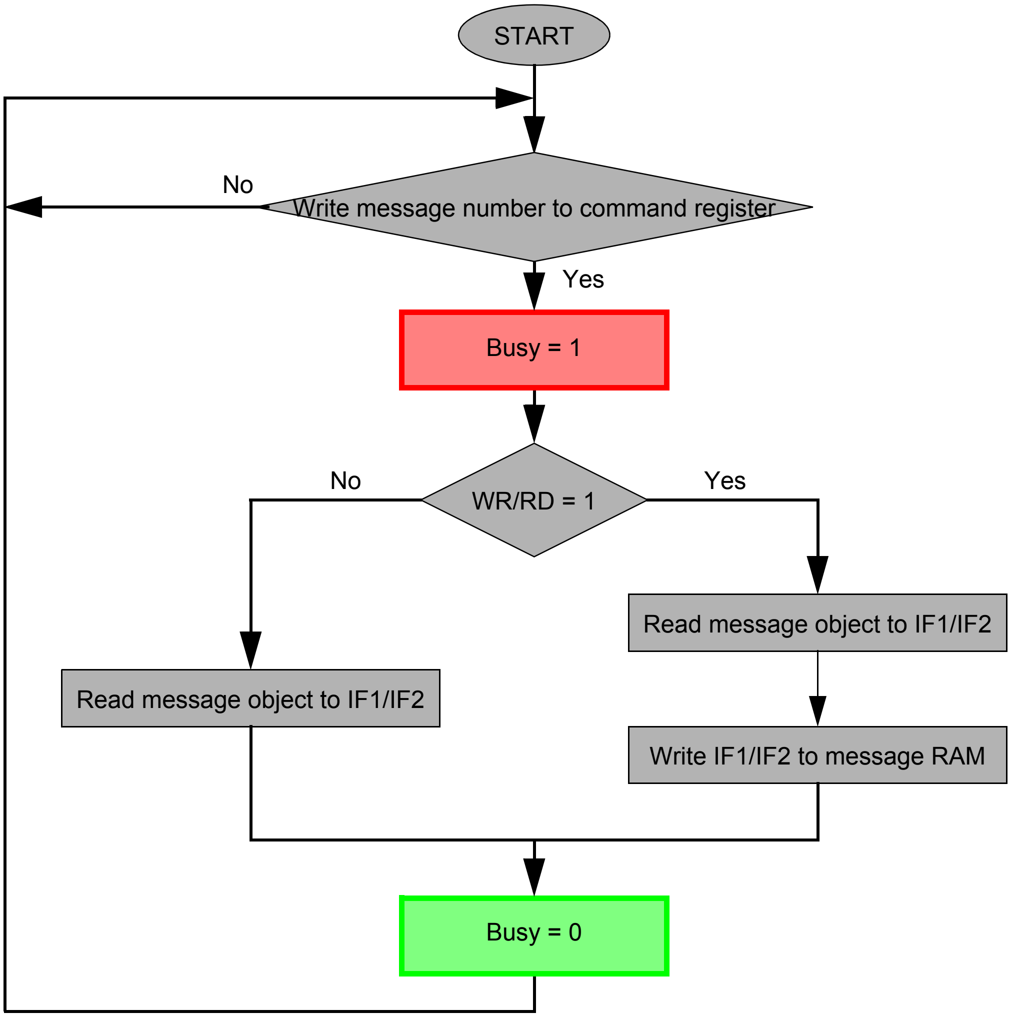
By buffering the data to be transferred, the Interface Register sets avoid conflicts between concurrent CPU accesses to the Message RAM and CAN message reception and transmission. A complete message object (see section 16.13.1) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set (see section 16.15.19) in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

16.12.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 Registers Sets control the data transfer to and from the message object. 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 16-20).

Figure 16-20. Data Transfer Between IF1 / IF2 Registers and Message RAM


START
No
Write message number to command register
Yes
Busy = 1
No
WR/RD = 1
Yes
Read message object to IF1/IF2
Read message object to IF1/IF2
Write IF1/IF2 to message RAM
Busy = 0

16.12.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The intention of this feature of IF3 is to provide an interface for the DMA to read packets efficiently. The automatic update functionality can be programmed for each message object (see IF3 Update Enable register, section 16.15.28).

All valid message objects in Message RAM which 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 (if no previous DMA transfers are ongoing), 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.

If DCAN internal IF3 update is complete, a DMA request is generated. The DMA request stays active until first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in CAN Control register. Please refer to the device datasheet to find out if this DMA source is available.

Note:
The IF3 register set can not be used for transferring data into message objects.

16.13 Message RAM

The DCAN Message RAM contains message objects and parity bits for the message objects. There are up to 128 message objects in the Message RAM.

During normal operation, accesses to the Message RAM are performed via the Interface Register sets, and the CPU cannot directly access the Message RAM.

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.

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 16.13.3)

2. In RAM Direct Access (RDA) mode (see section 16.13.4)

For the Message RAM Base address, please refer to the device datasheet.

16.13.1 Structure of Message Objects

Figure 16-21 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Figure 16-21. 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 16-2. 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: This bit may be kept at level ‘1’ even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the Data Length Code DLC[3:0] are changed. It should be reset if the Messages Object is no longer required.

Table 16-2. Message Object Field Descriptions

Name	Value	Description
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 one, the message object’s mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
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 = one).

Table 16-2. Message Object Field Descriptions

Name	Value	Description
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.
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 one.
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.

Table 16-2. Message Object Field Descriptions

Name	Value	Description
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 bits.
	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.

16.13.2 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, etc.

Note:
‘0’ is not a valid message object number.

Note:
Do not write to mailboxes beyond the number described in the device datasheet. A write to a mailbox address beyond the implemented number of mailboxes might corrupt an implemented mailbox.

Note:
The base address for DCAN1 RAM is 0xFF1E 0000, DCAN2 RAM is 0xFF1C 0000 and DCAN3 RAM is 0xFF1A 0000

Message Object number 1 has the highest priority.

Table 16-3. Message RAM addressing in Debug/Suspend and RDA mode

Message Object number	offset from base address	word number	Debug/Suspend mode, see section 16.13.3	RDA mode, see section 16.13.4
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	--

Table 16-3. Message RAM addressing in Debug/Suspend and RDA mode

last implemented (32(DCAN3) or 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	--

16.13.3 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 16-22. 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
	reserved															
MsgAddr + 0x00	reserved											Parity[4:0]				
	MXtd	MDir	reserved	Msk[28:16]												
MsgAddr + 0x04	Msk[15:0]															
	reserved	Xtd	Dir	ID[28:16]												
MsgAddr + 0x08	ID[15:0]															
	reserved															
MsgAddr + 0x0C	reserved	MsgLst	reserved	UMask	TxIE	RxTE	RmtEn	reserved	EOB	reserved			DLC[3:0]			
	Data 3								Data 2							
MsgAddr + 0x10	Data 1								Data 0							
	Data 7								Data 6							
MsgAddr + 0x14	Data 5								Data 4							

16.13.4 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 splitted 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.

Note:
Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit not set) will be ignored.

Figure 16-23. 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
	Data 4								Data 5							
MsgAddr + 0x00	Data 6								Data 7							
	Data 0								Data 1							
MsgAddr + 0x04	Data 2								Data 3							
	ID[27:12]															
MsgAddr + 0x08	ID[11:0]												DLC[3:0]			
	Msk[28:13]															
MsgAddr + 0x0C	Msk[12:0]													Xtd	Dir	ID[28]
	reserved															Parity[4]
MsgAddr + 0x10	Parity[3:0]				unused				MsgLst	UMask	TxIE	RxTE	RmtEn	EOB	MXtd	MDir

Note:
Write to unused bits has no effect.

16.14 GIO support

The CAN_RX and CAN_TX pins of the 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 16.15.29) and the CAN RX IO Control register (see section 16.15.30).

16.15 DCAN Control Registers

The base address for the

DCAN1 registers is 0xFFF7 DC00.

DCAN2 registers is 0xFFF7 DE00.

DCAN3 registers is 0xFFF7 E000.

Figure 16-24. DCAN Control Registers

<table>
<tr><th>Offset Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x00
DCAN CTL</td><td colspan="6">Reserved</td><td>WUBA</td><td>PDR</td><td colspan="3">Reserved</td><td>DE3</td><td>DE2</td><td>DE1</td><td>IE1</td><td>InitDbg</td></tr>
<tr><td>SWR</td><td>Reserved</td><td colspan="4">PMD</td><td>ABO</td><td>IDS</td><td>Test</td><td>CCE</td><td>DAR</td><td>Reserved</td><td>EIE</td><td>SIE</td><td>IE0</td><td>Init</td></tr>
<tr><td rowspan="2">0x04
DCAN ES</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="5">Reserved</td><td>PDA</td><td>Wake UpPnd</td><td>PER</td><td>BOff</td><td>EWarn</td><td>EPass</td><td>RxOK</td><td>TxOK</td><td colspan="3">LEC</td></tr>
<tr><td rowspan="2">0x08
DCAN ERRC</td><td colspan="16">Reserved</td></tr>
<tr><td>RP</td><td colspan="7">REC[6:0]</td><td colspan="8">TEC[7:0]</td></tr>
<tr><td rowspan="2">0x0C
DCAN BTR</td><td colspan="12">Reserved</td><td colspan="4">BRPE</td></tr>
<tr><td>Reserved</td><td colspan="3">TSeg2</td><td colspan="4">TSeg1</td><td colspan="2">SJW</td><td colspan="6">BRP</td></tr>
<tr><td rowspan="2">0x10
DCAN INT</td><td colspan="8">Reserved</td><td colspan="8">Int1ID[7:0]</td></tr>
<tr><td colspan="16">Int0ID[15:0]</td></tr>
<tr><td rowspan="2">0x14
DCAN TEST</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="6">Reserved</td><td>RDA</td><td>EXL</td><td>Rx</td><td colspan="2">Tx[1:0]</td><td>LBack</td><td>Silent</td><td colspan="3">Reserved</td></tr>
<tr><td rowspan="2">0x1C
DCAN PERR</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="3">Word Number</td><td colspan="8">Message Number</td></tr>
</table>

Offset Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80 DCAN ABOTR	ABO Time[31:16]															
	ABO Time[15:0]															
0x84 DCAN TXRQ X	Reserved															
	TxRqstReg8		TxRqstReg7		TxRqstReg6		TxRqstReg5		TxRqstReg4		TxRqstReg3		TxRqstReg2		TxRqstReg1	
0x88 DCAN TXRQ12	TxRqst[32:17]															
	TxRqst[16:1]															
0x8C DCAN TXRQ34	TxRqst[64:49]															
	TxRqst[48:33]															
0x90 DCAN TXRQ56	TxRqst[96:81]															
	TxRqst[80:65]															
0x94 DCAN TXRQ78	TxRqst[128:113]															
	TxRqst[112:97]															
0x98 DCAN NWDAT X	Reserved															
	NewDatReg8		NewDatReg7		NewDatReg6		NewDatReg5		NewDatReg4		NewDatReg3		NewDatReg2		NewDatReg1	
0x9C DCAN NWDAT12	NewDat[32:17]															
	NewDat[16:1]															
0xA0 DCAN NWDAT34	NewDat[64:49]															
	NewDat[48:33]															

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
0xA4 DCAN NWDAT56	NewDat[96:81]															
	NewDat[80:65]															
0xA8 DCAN NWDAT78	NewDat[128:113]															
	NewDat[112:97]															
0xAC DCAN INTPND X	Reserved															
	IntPndReg8		IntPndReg7		IntPndReg6		IntPndReg5		IntPndReg4		IntPndReg3		IntPndReg2		IntPndReg1	
0xB0 DCAN INTPND12	IntPnd[32:17]															
	IntPnd[16:1]															
0xB4 DCAN INTPND34	IntPnd[64:49]															
	IntPnd[48:33]															
0xB8 DCAN INTPND56	IntPnd[96:81]															
	IntPnd[80:65]															
0xBC DCAN INTPND78	IntPnd[128:113]															
	IntPnd[112:97]															
0xC0 DCAN MSGVAL X	Reserved															
	MsgValReg8		MsgValReg7		MsgValReg6		MsgValReg5		MsgValReg4		MsgValReg3		MsgValReg2		MsgValReg1	

Offset Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC4 DCAN MSGVAL12	MsgVal[32:17]															
	MsgVal[16:1]															
0xC8 DCAN MSGVAL34	MsgVal[64:49]															
	MsgVal[48:33]															
0xCC DCAN MSGVAL56	MsgVal[96:81]															
	MsgVal[80:65]															
0xD0 DCAN MSGVAL78	MsgVal[128:113]															
	MsgVal[112:97]															
0xD8 DCAN INTMUX12	IntMux[32:17]															
	IntMux[16:1]															
0xDC DCAN INTMUX34	IntMux[64:49]															
	IntMux[48:33]															
0xE0 DCAN INTMUX56	IntMux[96:81]															
	IntMux[80:65]															
0xE4 DCAN INTMUX78	IntMux[128:113]															
	IntMux[112:97]															

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
0x100 DCAN IF1CMD	Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ New Dat	Data A	Data B
	Busy	DMA active	Reserved						Message Number							
0x104 DCAN IF1MSK	MXtd	MDir	Reserved	Msk[28:16]												
	Msk[15:0]															
0x108 DCAN IF1ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															
0x10C DCAN IF1MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
0x110 DCAN IF1DATA	Data 3								Data 2							
	Data 1								Data 0							
0x114 DCAN IF1DATB	Data 7								Data 6							
	Data 5								Data 4							
0x120 DCAN IF2CMD	Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/ New- Dat	Data A	Data B
	Busy	DMA active	Reserved						Message Number							
0x124 DCAN IF2MSK	MXtd	MDir	Reserved	Msk[28:16]												
	Msk[15:0]															
0x128 DCAN IF2ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															

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
0x12C DCAN IF2MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
0x130 DCAN IF2DATA	Data 3								Data 2							
	Data 1								Data 0							
0x134 DCAN IF2DATB	Data 7								Data 6							
	Data 5								Data 4							
0x140 DCAN IF3OBS	Reserved															
	IF3 Upd	Reserved		IF3 SDB	IF3 SDA	IF3 SC	IF3 SA	IF3 SM	Reserved			Data B	DataA	Ctrl	Arb	Mask
0x144 DCAN IF3MSK	MXtd	MDir	Reserved	Msk[28:16]												
	Msk[15:0]															
0x148 DCAN IF3ARB	MsgVal	Xtd	Dir	ID[28:16]												
	ID[15:0]															
0x14C DCAN IF3MCTL	Reserved															
	New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
0x150 DCAN IF3DATA	Data 3								Data 2							
	Data 1								Data 0							
0x154 DCAN IF3DATB	Data 7								Data 6							
	Data 5								Data 4							

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
0x160 DCAN IF3UPD12	IF3UpdEn[32:17]															
	IF3UpdEn[16:1]															
0x164 DCAN IF3UPD34	IF3UpdEn[64:49]															
	IF3UpdEn[48:33]															
0x168 DCAN IF3UPD56	IF3UpdEn[96:81]															
	IF3UpdEn[80:65]															
0x16C DCAN IF3UPD78	IF3UpdEn[128:113]															
	IF3UpdEn[112:97]															
0x1E0 DCAN TIOC	Reserved												SR	PU	PD	OD
	Reserved												Func	Dir	Out	In
0x1E4 DCAN RIOC	Reserved												SR	PU	PD	OD
	Reserved												Func	Dir	Out	In

After hardware reset, the registers of the DCAN hold the values shown in the register descriptions.

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’.

16.15.1 CAN Control Register (DCAN CTL)

Figure 16-25. CAN Control Register (DCAN CTL) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						WUBA	PDR	Reserved			DE3	DE2	DE1	IE1	InitDbg
R-0						R/W-0	R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWR	Reserved	PMD				ABO	IDS	Test	CCE	DAR	Reserved	EIE	SIE	IE0	Init
R/WP-0	R-0	R/W-0x5				R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1

R = Read, W = Write, WP = Write protected by init bit, -*n* = Value after reset

Table 16-4. CAN Control Register Field Descriptions

Bit	Name	Value	Description
31-26	Reserved		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 16.6). 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 16.6).
23-21	Reserved		These bits are always read as 0. Writes have no effect.

Table 16-4. CAN Control Register Field Descriptions

Bit	Name	Value	Description
20	DE3		Enable DMA request line for IF3
		0	Disabled
		1	Enabled
			Note: A pending DMA request for IF3 remains active until first access to one of the IF3 registers.
19	DE2		Enable DMA request line for IF2
		0	Disabled
		1	Enabled
			Note: A pending DMA request for IF2 remains active until first access to one of the IF2 registers.
18	DE1		Enable DMA request line for IF1
		0	Disabled
		1	Enabled
			Note: A pending DMA request for IF1 remains active until first access to one of the IF1 registers.
17	IE1		Interrupt line 1 Enable
		0	Disabled - Module Interrupt DCAN1INT is always low.
		1	Enabled - Interrupts will assert line DCAN1INT to one; 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 not entered.
		1	Debug mode requested and 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		This bit is always read as 0. Writes have no effect.

Table 16-4. CAN Control Register Field Descriptions

Bit	Name	Value	Description
13-10	PMD		Parity on/off
		0x5 Others	Parity function disabled Parity function 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 configuration registers.
		1	The CPU has write access to the configuration registers (when Init bit is set).
5	DAR		Disable Automatic Retransmission
		0	Automatic Retransmission of not successful messages enabled.
		1	Automatic Retransmission disabled.
4	Reserved		This bit is always read as 0. Writes have no effect.
3	EIE		Error Interrupt Enable
		0	Disabled - PER, BOff and EWarn bits can not generate an interrupt.
		1	Enabled - PER, BOff and EWarn bits can generate an interrupt at DCAN0INT line and affect the Interrupt Register.

Table 16-4. CAN Control Register Field Descriptions

Bit	Name	Value	Description
2	SIE		Status Change Interrupt Enable
		0	Disabled - WakeUpPnd, RxOk, TxOk and LEC bits can not 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 one; 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 (refer 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.

16.15.2 Error and Status Register (DCAN ES)

Figure 16-26. Error and Status Register (DCAN ES) [offset = 0x04]

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
Reserved					PDA	Wake UpPnd	PER	BOff	EWarn	EPass	RxOK	TxOK	LEC		
R-0					R-0	R/C-0	R/C-0	R-0	R-0	R-0	R/C-0	R/C-0	R/S-111		

R = Read, S = Set by Read, C = Clear by Read, -*n* = Value after reset

.

Table 16-5. Error and Status Register Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		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	WakeUp Pnd		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.

Table 16-5. Error and Status Register Field Descriptions

Bit	Name	Value	Description
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 16-5. Error and Status Register Field Descriptions

Bit	Name	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
		1	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.
		2	Form Error**:** A fixed format part of a received frame has the wrong format.
		3	Ack Error**:** The message this CAN Core transmitted was not acknowledged by another node.
		4	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.
		5	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).
		6	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).
		7	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 '7'.

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 set the LEC to value '7'. Additionally, the Status Interrupt value (0x8000) in the Interrupt Register will be replaced by the next lower priority interrupt value.

Note:

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.

16.15.3 Error Counter Register (DCAN ERRC)

Figure 16-27. Error Counter Register (DCAN ERRC) [offset = 0x08]

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[6:0]							TEC[7:0]							
R-0	R-0							R-0							

R = Read, -*n* = Value after reset

.

Table 16-6. Error Counter Register Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		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[6:0]		Receive Error Counter. Actual state of the Receive Error Counter. (values from 0 to 255).
7-0	TEC[7:0]		Transmit Error Counter. Actual state of the Transmit Error Counter. (values from 0 to 255).

16.15.4 Bit Timing Register (DCAN BTR)

Figure 16-28. Bit Timing Register (DCAN BTR) [offset = 0x0C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TSeg2			TSeg1				SJW		BRP					
R-0	R/WP-0x2			R/WP-0x3				R/WP-0		R/WP-0x1					

R = Read, WP = Write Protected by CCE bit, -*n* = Value after reset

.

Table 16-7. Bit Timing Register Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		These bits are always read as 0. Writes have no effect.
19-16	BRPE	0x00-0x0F	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		This bit is always read as 0. Writes have no effect.
14-12	TSeg2	0x0-0x7	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value which is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	0x01-0x0F	Time segment before the sample point Valid programmed values are 1 to15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0x0-0x3	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	0x00-0x3F	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.

Note:
The CAN bit time may be programmed in the range of 8 to 25 time quanta.

Note:
The CAN time quantum may be programmed in the range of 1 to1024 CAN_CLK periods.

With a CAN_CLK of 8 MHz and BRPE = 0x00, the reset value of 0x00002301 configures the DCAN for a bit rate of 500kBit/s.

For details see section 16.11.

16.15.5 Interrupt Register (DCAN INT)

Figure 16-29. Interrupt Register (DCAN INT) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								Int1ID[7:0]							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Int0ID[15:0]															
R-0															

R = Read, -*n* = Value after reset

Table 16-8. Interrupt Register Field Descriptions

Bit	Name	Value	Description
31-24	Reserved		These bits are always read as 0. Writes have no effect.
23-16	Int1ID[23:16]		Interrupt 1 Identifier (indicates the message object with the highest pending interrupt)
		0x00	No interrupt is pending
		0x01-0x80	Number of message object which caused the interrupt.
		0x81-0xFF	Unused
			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.

Table 16-8. Interrupt Register Field Descriptions (Continued)

Bit	Name	Value	Description
15-0	Int0ID[15:0]		Interrupt Identifier (the number here indicates the source of the interrupt)
		0x0000	No interrupt is pending
		0x0001-0x0080	Number of message object which caused the interrupt.
		0x0081-0x7FFF	Unused
		0x8000	Error and Status Register value is not 0x07.
		0x8001-0xFFFF	Unused
			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.

16.15.6 Test Register (DCAN TEST)

Figure 16-30. Test Register (DCAN TEST) [offset = 0x14]

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
Reserved						RDA	EXL	Rx	Tx[1:0]		LBack	Silent	Reserved		
R-0						R/WP-0	R/WP-0	R-U	R/WP-0		R/WP-0	R/WP-0	R-0		

R = Read, WP = Write Protected by Test bit, -*n* = Value after reset, -U = Undefined

.

Table 16-9. Test Register Field Descriptions

Bit	Name	Value	Description
31–10	Reserved		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
		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[1:0]		Control of CAN_TX pin
		00	Normal operation, CAN_TX is controlled by the CAN Core.
		01	Sample Point can be monitored at CAN_TX pin.
		10	CAN_TX pin drives a dominant value.
		11	CAN_TX pin drives a recessive value.
4	LBack		Loop Back Mode
		0	Disabled
		1	Enabled

Table 16-9. Test Register Field Descriptions

Bit	Name	Value	Description
3	Silent		Silent Mode
		0	Disabled
		1	Enabled
2-0	Reserved		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 one. 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 Test bit is cleared.

Note:
The Test Register is only writable if Test bit in CAN Control Register is set.

Note:
Setting Tx[1:0] other than ‘00’ will disturb message transfer.

Note:
When the internal loop back mode is active (bit LBack is set), bit EXL will be ignored.

16.15.7 Parity Error Code Register (DCAN PERR)

Figure 16-31. Parity Error Code Register (DCAN PERR) [offset = 0x1C]

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
Reserved					Word Number			Message Number							
R-0					R-U			R-U							

R = Read, -*n* = Value after reset, -U = Undefined

.

Table 16-10. Parity Error Code Register Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		These bits are always read as 0. Writes have no effect.
10-8	Word Number		Word number where parity error has been detected
		0x01-0x05	RDA word number (1 to 5) of the message object (according to the Message RAM representation in RDA mode, see section 16.13.4).
7–0	Message Number	0x01-0x80	Message object number where parity error has been detected

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.

16.15.8 Auto-Bus-On Time Register (DCAN ABOTR)

Figure 16-32. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABO Time[31:16]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABO Time[15:0]															
R/W-0															

R = Read, W = Write, -*n* = Value after reset

.

Table 16-11. Auto-Bus-On Time Register Field Descriptions

Bit	Name	Value	Description
31–0	ABO Time		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 bit ABO in CAN Control Register. The Auto-Bus-On timer is realized by a 32-bit counter which starts to count down to zero 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.

Note:
During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

16.15.9 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 16-33. Transmission Request X Register (DCAN TXRQ X) [offset = 0x84]

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	

R = Read, -*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.

16.15.10 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 16-34. Transmission Request Registers [offset = 0x88 to 0x94]

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
0x88 CAN TXRQ12	TxRqst[32:17]															
	R-0															
	TxRqst[16:1]															
	R-0															
0x8C CAN TXRQ34	TxRqst[64:49]															
	R-0															
	TxRqst[48:33]															
	R-0															
0x90 CAN TXRQ56	TxRqst[96:81]															
	R-0															
	TxRqst[80:65]															
	R-0															
0x94 CAN TXRQ78	TxRqst[128:113]															
	R-0															
	TxRqst[112:97]															
	R-0															

R = Read, -*n* = Value after reset

Table 16-12. 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.

16.15.11 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 16-35. New Data X Register (DCAN NWDAT X) [offset = 0x98]

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	

R = Read, -*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.

16.15.12 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 16-36. New Data Registers [offset = 0x9C to 0xA8]

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
0x9C CAN NWDAT12	NewDat[32:17]															
	R-0															
	NewDat[16:1]															
	R-0															
0xA0 CAN NWDAT34	NewDat[64:49]															
	R-0															
	NewDat[48:33]															
	R-0															
0xA4 CAN NWDAT56	NewDat[96:81]															
	R-0															
	NewDat[80:65]															
	R-0															
0xA8 CAN NWDAT78	NewDat[128:113]															
	R-0															
	NewDat[112:97]															
	R-0															

R = Read, -*n* = Value after reset

Table 16-13. 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.

16.15.13 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 16-37. Interrupt Pending X Register (DCAN INTPND X) [offset = 0xAC]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">IntPndReg8</td><td colspan="2">IntPndReg7</td><td colspan="2">IntPndReg6</td><td colspan="2">IntPndReg5</td><td colspan="2">IntPndReg4</td><td colspan="2">IntPndReg3</td><td colspan="2">IntPndReg2</td><td colspan="2">IntPndReg1</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-0</td></tr>
</table>

R = Read, -*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.

16.15.14 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 16-38. Interrupt Pending Registers [offset = 0xB0 to 0xBC]

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Address† Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB0 CAN INTPND12	IntPnd[32:17]															
	R-0															
	IntPnd[16:1]															
	R-0															
0xB4 CAN INTPND34	IntPnd[64:49]															
	R-0															
	IntPnd[48:33]															
	R-0															
0xB8 CAN INTPND56	IntPnd[96:81]															
	R-0															
	IntPnd[80:65]															
	R-0															
0xBC CAN INTPND78	IntPnd[128:113]															
	R-0															
	IntPnd[112:97]															
	R-0															

R = Read, *-n* = Value after reset

Table 16-14. 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.

16.15.15 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 16-39. Message Valid X Register (DCAN MSGVAL X) [offset = 0xC0]

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	

R = Read, -*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.

16.15.16 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 16-40. Message Valid Registers [offset = 0xC4 to 0xD0]

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
0xC4 CAN MSGVAL12	MsgVal[32:17]															
	R-0															
	MsgVal[16:1]															
	R-0															
0xC8 CAN MSGVAL34	MsgVal[64:49]															
	R-0															
	MsgVal[48:33]															
	R-0															
0xCC CAN MSGVAL56	MsgVal[96:81]															
	R-0															
	MsgVal[80:65]															
	R-0															
0xD0 CAN MSGVAL78	MsgVal[128:113]															
	R-0															
	MsgVal[112:97]															
	R-0															

R = Read, -*n* = Value after reset

Table 16-15. 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.

16.15.17 Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)

The IntMux flag determine 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 16-41. Interrupt Multiplexer Registers [offset = 0xD8 to 0xE4]

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
0xD8 CAN INTMUX12	IntMux[31:16]															
	R/W-0															
	IntMux[15:0]															
	R/W-0															
0xDC CAN INTMUX34	IntMux[63:48]															
	R/W-0															
	IntMux[47:32]															
	R/W-0															
0xE0 CAN INTMUX56	IntMux[95:80]															
	R/W-0															
	IntMux[79:64]															
	R/W-0															
0xE4 CAN INTMUX78	IntMux[127:112]															
	R/W-0															
	IntMux[111:96]															
	R/W-0															

R = Read, W = Write, -*n* = Value after reset

Table 16-16. Interrupt Multiplexer Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntMux[127:0]		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

Table 16-16. Interrupt Multiplexer Registers Field Descriptions (Continued)

Bit	Name	Value	Description
		0	DCAN0INT line is active if corresponding IntPnd flag is one.
		1	DCAN1INT line is active if corresponding IntPnd flag is one.

16.15.18 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 Busy bit is one, IF1/IF2 Register sets are write protected.

Note:
For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by r/w) is disabled during Debug/Suspend mode.

Note:
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 16-42. IF1 Command Registers (DCAN IF1CMD) [offset = 0x100]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst /New-Dat	Data A	Data B
R-0								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
Busy	DMA active	Reserved						Message Number							
R-0	R/WP/C-0	R-0						R/WP-0x1							

R = Read, WP = Protected Write (protected by Busy bit), C = Clear by IF1 access, -*n* = Value after reset

Figure 16-43. IF2 Command Registers (CAN IF2CMD) [offset = 0x120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								WR/RD	Mask	Arb	Control	Clr IntPnd	TxRqst/NewDat	Data A	Data B
R-0								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
Busy	DMA active	Reserved						Message Number							
R-0	R/WP/C-0	R-0						R/WP-0x1							

R = Read, WP = Protected Write (protected by Busy bit), C = Clear by IF2 access, -*n* = Value after reset

Table 16-17. IF1/IF2 Command Register Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		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]).

Table 16-17. IF1/IF2 Command Register Field Descriptions (Continued)

Bit	Name	Value	Description
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 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 only be controlled by 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 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 one independent of the values in IF1/IF2 Message Control Register. Note: 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.

Table 16-17. IF1/IF2 Command Register Field Descriptions (Continued)

Bit	Name	Value	Description
17	Data A		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.
16	Data B		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 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 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 one 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 been finished.
14	DMAactive		Activation of DMA feature for subsequent internal IF1/IF2 update
		0	DMA request line is independent of IF1/IF2 activities.
		1	DMA is requested after completed transfer between IF1/IF2 Register Set and Message RAM. The DMA request remains active until the first read or write to one of the IF1/IF2 registers; an exception is a write to Message Number (Bits [7:0]) when DMAactive is one. Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.
13-8	Reserved		These bits are always read as 0. Writes have no effect.

Table 16-17. IF1/IF2 Command Register Field Descriptions (Continued)

Bit	Name	Value	Description
7–0	Message Number		Number of message object in Message RAM which is used for data transfer
		0x00	Invalid message number
		0x01-0x80	Valid message numbers (if 128 message objects are implemented)
		0x81-0xFF	Invalid message numbers (if 128 message objects are implemented)

16.15.19 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 16.13.1.

> **Note:**
> While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 16-44. IF1 Mask Register (DCAN IF1MSK) [offset = 0x104]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd	MDir	Reserved	Msk[28:16]												
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk[15:0]															
R/WP-0xFFFF															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-45. IF2 Mask Register (DCAN IF2MSK) [offset = 0x124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd	MDir	Reserved	Msk[28:16]												
R/WP-1	R/WP-1	R-1	R/WP-0x1FFF												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk[15:0]															
R/WP-0xFFFF															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

.

Table 16-18. IF1/IF2 Mask Registers Field Descriptions

Bit	Name	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 together with mask bits Msk[28:18] are considered.
30	MDir		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.
			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		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.

16.15.20 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 16.13.1

Note:
While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 16-46. IF1 Arbitration Register (DCAN IF1ARB) [offset = 0x108]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
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
ID[15:0]															
R/WP-0															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-47. IF2 Arbitration Register (DCAN IF2ARB) [offset = 0x128]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
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
ID[15:0]															
R/WP-0															

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Table 16-19. IF1/IF2 Arbitration Registers Field Descriptions

Bit	Name	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.
			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. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir, DLC[3:0], RxIE, TxIE, RmtEn, EoB, UMask, the mask bits Msk28:0, MXtd, and MDir are modified, or if the messages object is no longer required.
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 reception of a Data Frame with 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 reception of a Remote Frame with 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 = one, standard frames in message objects with Xtd = zero.

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.

16.15.21 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 16.13.1.

> **Note:**
> While Busy bit of IF1/IF2 Command Register is one, IF1/IF2 Register Set is write protected.

Figure 16-48. IF1 Message Control Register (DCAN IF1MCTL) [offset = 0x10C]

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
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-49. IF2 Message Control Register (DCAN IF2MCTL) [offset = 0x12C]

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
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0			R/WP-0			

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

.

Table 16-20. IF1 / IF2 Message Control Registers Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		These bits are always read as 0. Writes have no effect.

Table 16-20. IF1 / IF2 Message Control Registers Field Descriptions (Continued)

Bit	Name	Value	Description
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 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.
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 ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
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.

Table 16-20. IF1 / IF2 Message Control Registers Field Descriptions (Continued)

Bit	Name	Value	Description
8	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.
7	EoB		Data Frame has 0-8 data bits.
		0	Data Frame has 8 data bytes.
		1	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.
			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 one.
6-5	Reserved		These bits are always read as 0. Writes have no effect.
3-0	DLC[3:0]		Data Length Code
		0-8	Data Frame has 0-8 data bits.
		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.

16.15.22 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 16-50. IF1 Data A Register (DCAN IF1DATA) [offset = 0x110]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Data 3</td><td colspan="8">Data 2</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Data 1</td><td colspan="8">Data 0</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
</table>

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-51. IF1 Data B Register (DCAN IF1DATB) [offset = 0x114]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Data 7</td><td colspan="8">Data 6</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Data 5</td><td colspan="8">Data 4</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
</table>

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-52. IF2 Data A Register (DCAN IF2DATA) [offset = 0x130]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Data 7</td><td colspan="8">Data 6</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Data 5</td><td colspan="8">Data 4</td></tr>
<tr><td colspan="8">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
</table>

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

Figure 16-53. IF2 Data B Register (DCAN IF2DATB) [offset = 0x134]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data 7								Data 6							
R/WP-0								R/WP-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data 5								Data 4							
R/WP-0								R/WP-0							

R = Read, WP = Protected Write (protected by Busy bit), -*n* = Value after reset

16.15.23 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 16.13.1).

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 in order to complete a DMA read cycle. 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 without activating the DMA request line and without waiting for DMA read accesses.

A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register.

The status of the current read-cycle can be observed via status flags (Bits [12:8]).

If an interrupt line is available for IF3, an interrupt will be generated by IF3Upd flag. Please refer to the device datasheet to find out if this interrupt source is available

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 or interrupt mode.

Figure 16-54. IF3 Observation Register (DCAN IF3OBS) [offset = 0x140]

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
IF3 Upd	Reserved		IF3 SDB	IF3 SDA	IF3 SC	IF3 SA	IF3 SM	Reserved			Data B	DataA	Ctrl	Arb	Mask
R-0	R-0		R-0	R-0	R-0	R-0	R-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, WP = Protected Write, S = Set, C = Clear by Read, U = Undefined, -*n* = Value after reset

Table 16-21. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		These bits are always read as 0. Writes have no effect.
15	IF3 Upd		IF3 Update Data
		0	No new data has been loaded since last IF3 read.
		1	New data has been loaded since last IF3 read.

Table 16-21. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
14–13	Reserved		These bits are always read as 0. Writes have no effect.
12	IF3 SDB		IF3 Status of Data B read access
		0	All Data B bytes are already read out, or are not marked to be read.
		1	Data B section has still data to be read out.
11	IF3 SDA		IF3 Status of Data A read access
		0	All Data A bytes are already read out, or are not marked to be read.
		1	Data A section has still data to be read out.
10	IF3 SC		IF3 Status of Control bits read access
		0	All Control section bytes are already read out, or are not marked to be read.
		1	Control section has still data to be read out.
9	IF3 SA		IF3 Status of Arbitration data read access
		0	All Arbitration data bytes are already read out, or are not marked to be read.
		1	Arbitration section has still data to be read out.
8	IF3 SM		IF3 Status of Mask data read access
		0	All Mask data bytes are already read out, or are not marked to be read.
		1	Mask section has still data to be read out.
7–5	Reserved		These bits are always read as 0. Writes have no effect.
4	DataB		Data B read observation
		0	Data B section has not 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 has not to be read.
		1	Data A section has to be read to enable next IF3 update.
2	Ctrl		Ctrl read observation
		0	Ctrl section has not to be read.
		1	Ctrl section has to be read to enable next IF3 update.

Table 16-21. IF3 Observation register Field Descriptions

Bit	Name	Value	Description
1	Arb		Arbitration data read observation
		0	Arbitration data has not to be read.
		1	Arbitration data has to be read to enable next IF3 update.
0	Mask		Mask data read observation
		0	Mask data has not to be read.
		1	Mask data has to be read to enable next IF3 update.

16.15.24 IF3 Mask Register (DCAN IF3MSK)

Figure 16-55. IF3 Mask Register (DCAN IF3MSK) [offset = 0x144]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td>MXtd</td><td>MDir</td><td>Reserved</td><td colspan="13">Msk[28:16]</td></tr>
<tr><td>R-1</td><td>R-1</td><td>R-1</td><td colspan="13">R-0x1FFF</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Msk[15:0]</td></tr>
<tr><td colspan="16">R-0xFFFF</td></tr>
</table>

R = Read, -*n* = Value after reset

Table 16-22. IF3 Mask Register Field Descriptions

Bit	Name	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.
			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 Massage 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		These bits are always read as 0. 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.

16.15.25 IF3 Arbitration Register (DCAN IF3ARB)

Figure 16-56. IF3 Arbitration Register (DCAN IF3ARB) [offset = 0x148]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal	Xtd	Dir	ID[28:16]												
R-0	R-0	R-0	R-0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
R-0															

R = Read, *-n* = Value after reset

Table 16-23. IF3 Arbitration Register Field Descriptions

Bit	Name	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.
			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. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.
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 reception of a Data Frame with matching identifier, that message is stored in this message object.
		1	Direction = transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier (“Extended Frame”)
		ID[28:18]	11-bit Identifier (“Standard Frame”)

16.15.26 IF3 Message Control Register (DCAN IF3MCTL)

Figure 16-57. IF3 Message Control Register (DCAN IF3MCTL) [offset = 0x14C]

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
New Dat	Msg Lst	Int Pnd	UMask	TxIE	RxIE	Rmt En	Tx Rqst	EoB	Reserved			DLC[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0			R-0			

R = Read, -*n* = Value after reset

Table 16-24. IF3 Message Control Register Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		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 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.
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.

Table 16-24. IF3 Message Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering
			If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.
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 is 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 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 one.
6-4	Reserved		These bits are always read as 0. Writes have no effect.

Table 16-24. IF3 Message Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
3-0	DLC[3:0]		Data Length Code
		0-8	Data Frame has 0-8 data bits.
		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.

16.15.27 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 16-58. IF3 Data A Register (DCAN IF3DATA) [offset = 0x150]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data 3								Data 2							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data 1								Data 0							
R-0								R-0							

R = Read, -*n* = Value after reset

Figure 16-59. IF3 Data A Register (DCAN IF3DATB) [offset = 0x154]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data 7								Data 6							
R-0								R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data 5								Data 4							
R-0								R-0							

R = Read, -*n* = Value after reset

16.15.28 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 (e.g. 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 16-60. IF3 Update Enable Registers [offset = 0x160 to 0x16C]

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
0x160 CAN IF3UPD12	IF3UpdEn[32:17]															
	R/W-0															
	IF3UpdEn[16:1]															
	R/W-0															
0x164 CAN IF3UPD34	IF3UpdEn[64:49]															
	R/W-0															
	IF3UpdEn[48:33]															
	R/W-0															
0x168 CAN IF3UPD56	IF3UpdEn[96:81]															
	R/W-0															
	IF3UpdEn[80:65]															
	R/W-0															
0x16C CAN IF3UPD78	IF3UpdEn[128:113]															
	R/W-0															
	IF3UpdEn[112:97]															
	R/W-0															

R = Read, W = Write, *-n* = Value after reset

Table 16-25. 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.

16.15.29 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.

Note:
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 16-61. CAN TX IO Control Register (DCAN TIOC) [offset = 0x1E0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													PU	PD	OD
R-0												R/W-0	R/W-D	R/W-D	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												Func	Dir	Out	In
R-0												R/WP-0	R/WP-0	R/WP-0	R-U

R = Read, W = Write, WP = Protected Write (protected by Init bit), -*n* = Value after reset; D = device dependent

.

Table 16-26. CAN TX IO Control Register Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		These bits are always read as 0. Writes have no effect.
18	PU		CAN_TX pull up/pull down select. This bit is only active when CAN_TX is configured to be an input
		0	CAN_TX pull down is selected, when pull logic is active (PD = 0).
		1	CAN_TX pull up 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

Table 16-26. CAN TX IO Control Register Field Descriptions (Continued)

Bit	Name	Value	Description
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		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 driven to logic high (1)
			Forced to Tx output of the CAN Core, if Init bit of CAN Control register is reset.
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 (e.g. while reset of the DCAN module).

16.15.30 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.

Note:
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 16-62. CAN RX IO Control Register (DCAN RIOC) [offset = 0x1E4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													PU	PD	OD
R-0												R/W-0	R/W-D	R/W-D	R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												Func	Dir	Out	In
R-0												R/WP-0	R/WP-0	R/W-0	R-U

R = Read, W = Write, WP = Protected Write (protected by Init bit), -*n* = Value after reset; D = device dependent

Table 16-27. CAN RX IO Control Register Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		These bits are always read as 0. Writes have no effect.
18	PU		CAN_RX pull up/pull down select. This bit is only active when CAN_RX is configured to be an input
		0	CAN_TX pull down is selected, when pull logic is active (PD = 0).
		1	CAN_TX pull up 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

Table 16-27. CAN RX IO Control Register Field Descriptions

Bit	Name	Value	Description
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		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.
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 ‘1’ 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 driven to 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 (e.g. while reset of the DCAN module).

FlexRay and FlexRay Transfer Unit Modules

This reference guide describes Texas Instruments TMS570 FlexRay communication module and FlexRay Transfer Unit (FTU) Module. The FTU is similar to a DMA (Direct Memory Access) module, but it is specialized to transfer Flexray data to or from the microcontroller RAM.

Topic ... Page

17.1 Overview

The FlexRay module performs communication according to the FlexRay protocol specification v2.1. The sample clock bitrate can be programmed to values up to 10 MBit per second. Additional bus driver (BD) hardware is required for connection to the physical layer.

For communication on a FlexRay network, individual message buffers with up to 254 data bytes are configurable. The message storage consists of a single-ported message RAM that holds up to 128 message buffers. All functions concerning the handling of messages are implemented in the message handler. Those functions are the acceptance filtering, the transfer of messages between the two FlexRay Channel Protocol Controllers and the message RAM, maintaining the transmission schedule as well as providing message status information.

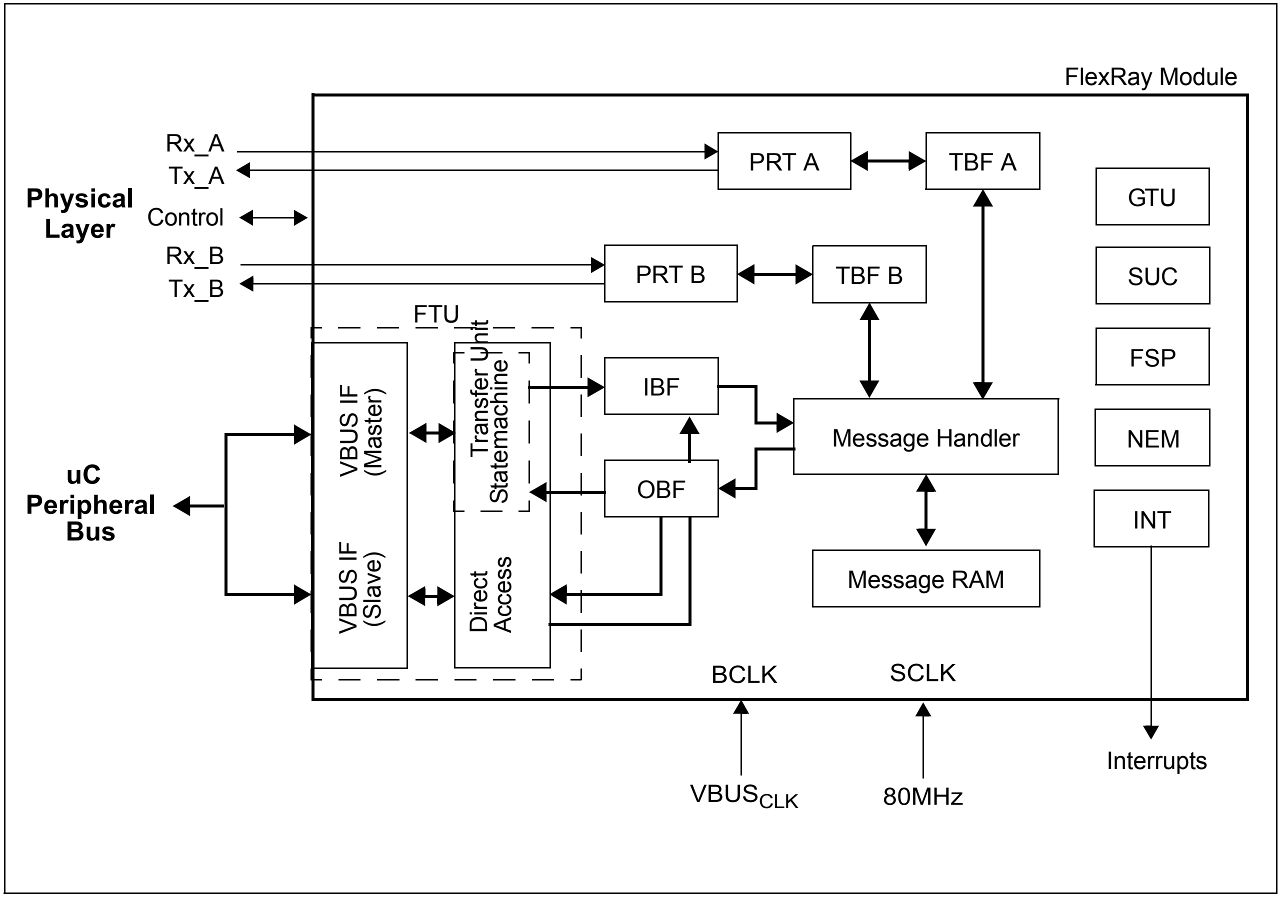
The register set of the FlexRay module can be accessed directly by the CPU via the VBUS interface. These registers are used to control, configure and monitor the FlexRay channel protocol controllers, message handler, global time unit, system universal control, frame/symbol processing, network management, interrupt control, and to access the message RAM via the input / output buffer.

17.1.1 Feature List

- Conformance with FlexRay protocol specification v2.1.
- Data rates of up to 10 Mbit/s on each channel
- Up to 128 message buffers
- 8 Kbyte of message RAM for storage of e.g. 128 message buffers with max. 48 byte data section or up to 30 message buffers with 254 byte data section
- Configuration of message buffers with different payload lengths
- One configurable receive FIFO
- Each message buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO
- CPU access to message buffers via input and output buffer
- Specialized DMA like FlexRay Transfer Unit (FTU) for automatic data transfer between data memory and message buffers without CPU interaction
- Filtering for slot counter, cycle counter, and channel
- Maskable module interrupts
- Supports Network Management

17.2 FlexRay Module Block Diagram

Figure 17-1. Block Diagram



The TI FlexRay module contains the following blocks:

- Peripheral Interface (VBUS IF)

  Interface to the Peripheral Bus of the TMS570 microcontroller architecture. The FlexRay module can either act as a VBUS master or VBUS slave
- FlexRay Transfer Unit (FTU)

  The internal intelligent state-machine (Transfer Unit State Machine) is able to transfer data between the input buffer (IBF) and output buffer (OBF) of the communication controller and the system memory without CPU interaction.

Note:
Since the FlexRay module is accessed through the FTU, the FTU must be powered up by the according bit in the Peripheral Power Down Registers of the System Module before accessing any FlexRay module register. For details please refer to the Architecture Documentation and the device specific Data Sheet.

- Input Buffer (IBF)

  For write access to the message buffers configured in the message RAM, the CPU or the FTU can write the header and data section for a specific message buffer to the input buffer. The message handler then transfers the data from the input buffer to the selected message buffer in the message RAM.

- Output Buffer (OBF)

  For read access to a message buffer configured in the message RAM the message handler transfers the selected message buffer to the output buffer. After the transfer has completed, the CPU or the FTU can read the header and data section of the transferred message buffer from the output buffer.

- Message Handler (MHD)

  The message handler controls data transfers between the following components:

  - Input / output buffer and message RAM
  - Transient buffer RAMs of the two FlexRay protocol controllers and message RAM

- Message RAM

  The message RAM stores up to 128 FlexRay message buffers together with the related configuration data (header and data partition).

- The Transient Buffer RAM (TBF A/B):

  Stores the data section of two complete messages

- FlexRay Channel Protocol Controller (PRT A/B)

  The FlexRay channel protocol controllers consist of a shift register and the FlexRay protocol FSM (Finite State Machine). They are connected to the transient buffer RAMs for intermediate message storage and to the physical layer via bus drivers (BD).

  They perform the following functionality:

  - Control and check of bit timing
  - Reception / transmission of FlexRay frames and symbols
  - Check of header CRC
  - Generation / check of frame CRC
  - Interfacing to bus driver

  The FlexRay channel protocol controllers have interfaces to:

  - Physical layer (bus driver)
  - Transient buffer RAM
  - Message handler
  - Global Time Unit
  - System universal control
  - Frame and symbol processing
  - Network management
  - Interrupt control

- Global time unit (GTU)

  The GTU performs the following functions:

  - Generation of microtick
  - Generation of macrotick
  - Fault tolerant clock synchronization by FTM algorithm
    - rate and offset correction
    - offset correction
  - Cycle counter
  - Timing control of static segment
  - Timing control of dynamic segment (minislotting)
  - Support of external clock correction

- System Universal Control (SUC)

  The SUC controls the following functions:
  - Configuration
  - Wakeup
  - Startup
  - Normal Operation
  - Passive Operation
  - Monitor Mode
- Frame and Symbol Processing (FSP)

  The frame and symbol processing controls the following functions:
  - Checks the correct timing of frames and symbols
  - Tests the syntactical and semantic correctness of received frames
  - Sets the slot status flags
- Network Management (NEM)

  Handles the network management vector.
- Interrupt Control (INT)

  The interrupt controller performs the following functions:
  - Provides error and status interrupt flags
  - Enable / disable interrupt sources
  - Assignment of interrupt sources to the two module interrupt lines
  - Enable / disable module interrupt lines
  - Manages the two interrupt timers
  - Stop watch time capturing
- 80MHz Clock Signal

  Clock signal for the sample clock (SCLK) of the FlexRay module

Note:
The FPLL (PLL2 / Clock Source 6) is used to provide the 80MHz clock to the Flexray Module. More information about the FPLL can be found in the Phase-Locked Loop (PLL) document.

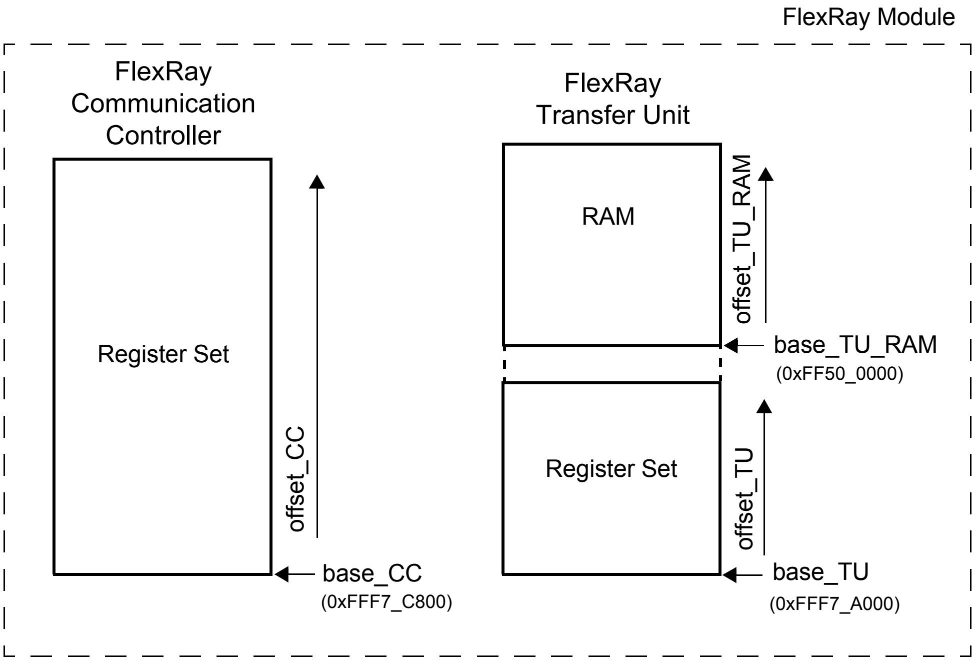
- Module Clock ($VBUS_{CLK}$)

  The FlexRay module clock (BCLK) is derived from the Peripheral Clock $VBUS_{CLK}$ of the microcontroller

17.3 FlexRay Module Block Mapping

The following Figure 17-2 shows the different module blocks of the FlexRay module. It is split into 3 sections, the communication controller, the transfer unit and the transfer unit RAM. The RAM of the communication controller is only memory mapped in test mode, where it is mapped to the register set address range.

Figure 17-2. FlexRay Module Blocks



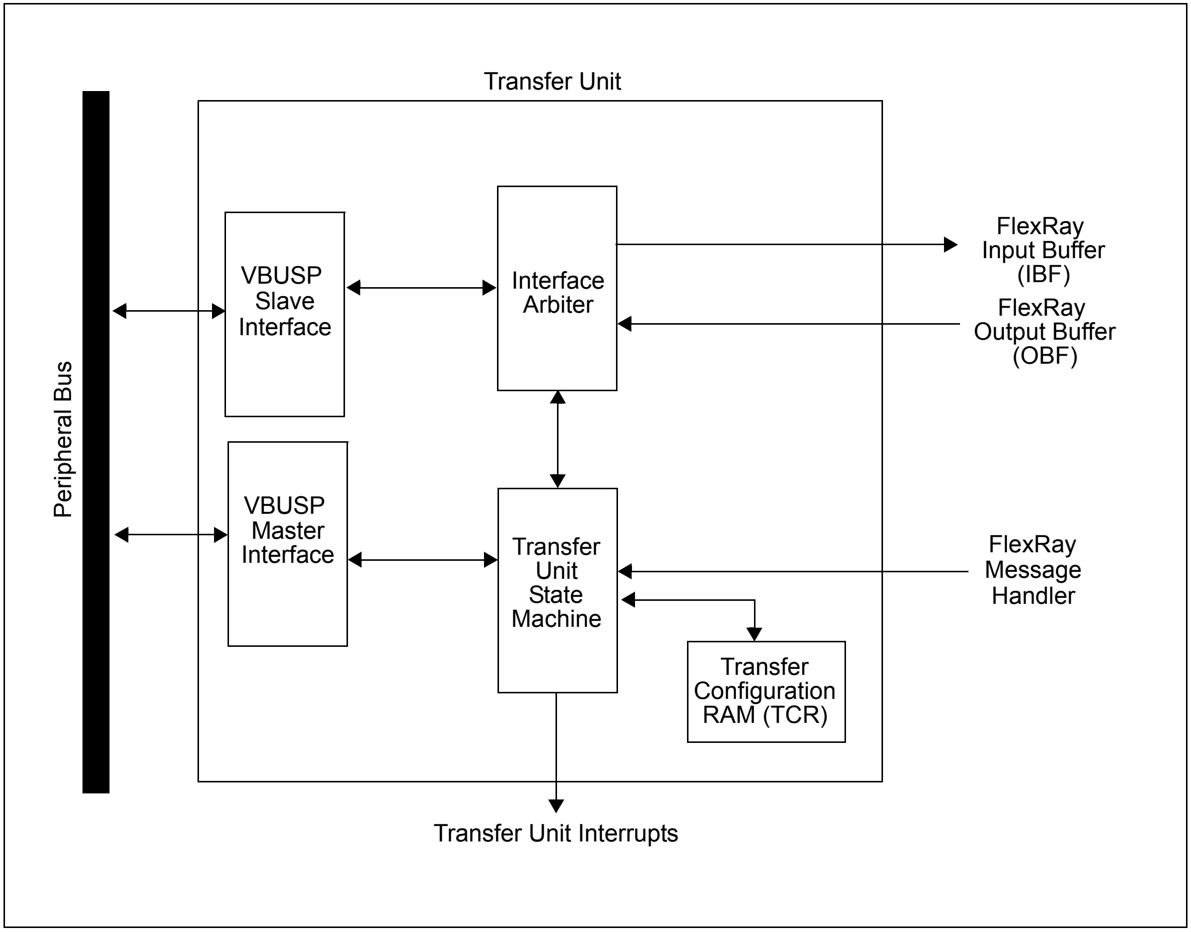
The address ranges of the 3 FlexRay blocks are shown in the table below.

Table 17-1. FlexRay Address Range Table

Module	Address Range
FlexRay Communication Controller	0xFFF7_C800 - 0xFFF7_CFFF
FlexRay TU	0xFFF7_A000 - 0xFFF7_A1FF
Flexray TU RAM	0xFF50_0000 - 0xFF51_FFFF

17.4 Transfer Unit Block Diagram

Figure 17-3. Transfer Unit



The FlexRay Transfer Unit (FTU) has an internal intelligent state-machine (Transfer Unit State Machine) to transfer data between the Input and Output Buffer Interfaces of the FlexRay core module and the system memory of the microcontroller without CPU interaction. It operates in a similar manner to a DMA (Direct Memory Access) module.

The FlexRay Input Buffer (IBF) and FlexRay Output Buffer (OBF) can also be accessed directly by the CPU. In this case the IBF and OBF are 8-, 16-, and 32-bit accessible.
For transfers using the Transfer Unit State Machine only 4x32-bit data packages (4 word bursts) are supported.

The Interface Arbiter controls the access to the IBF and OBF. Direct CPU accesses to IBF and OBF are not possible, if the Transfer Unit State Machine is switched on. Accesses will be ignored and the associated error interrupt will be generated.

The Transfer Unit State Machine is the head of all manual, event driven and automatic message transfer activities. It controls the Transfer Unit interrupt generation related to transfer protocol correctness, status and violations of the message transfers.

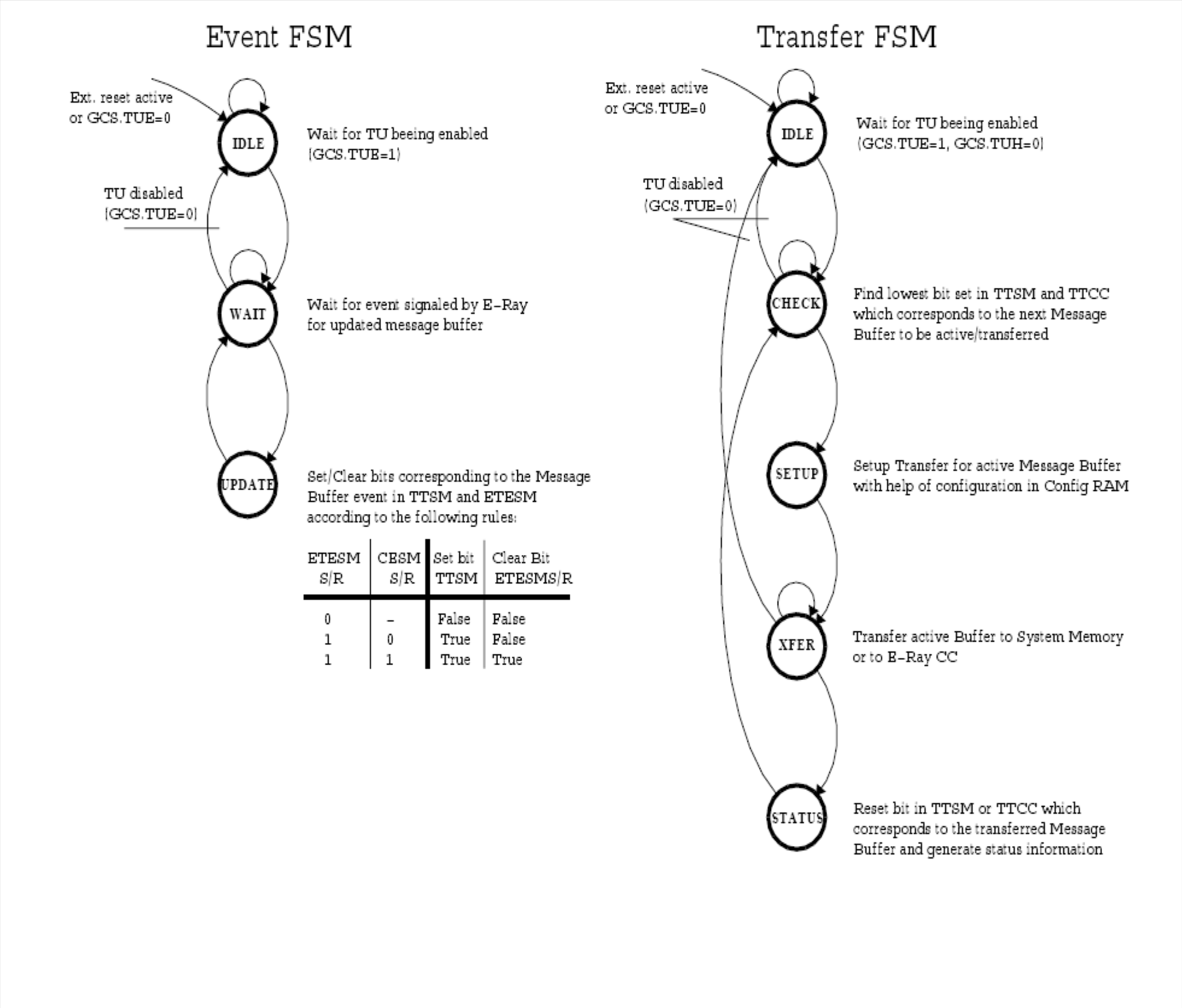
With the Transfer Configuration RAM (TCR) the transfer sequence, executed by the Transfer Unit State Machine, can be configured.

The usage of the Transfer Unit allows the user to setup a mirror of the FlexRay message RAM in the fast accessible data RAM of the microcontroller. The Transfer Unit can handle the data transfers between the data RAM and the FlexRay message RAM in the ‘background’ without CPU interaction.

17.5 Transfer Unit Functional Description

The following diagram shows the principle of the Transfer Unit operation including Event State Machine (left) and Transfer State Machine (right).

Figure 17-4. FlexRay Transfer Unit Operation Principle (simplified)


Event FSM
Ext. reset active
or GCS.TUE=0
IDLE
Wait for TU beeing enabled
(GCS.TUE=1)
TU disabled
(GCS.TUE=0)
WAIT
Wait for event signaled by E-Ray
for updated message buffer
UPDATE
Set/Clear bits corresponding to the Message
Buffer event in TTSM and ETESM
according to the following rules:
Transfer FSM
Ext. reset active
or GCS.TUE=0
IDLE
Wait for TU beeing enabled
(GCS.TUE=1, GCS.TUH=0)
TU disabled
(GCS.TUE=0)
CHECK
Find lowest bit set in TTSM and TTCC
which corresponds to the next Message
Buffer to be active/transferred
SETUP
Setup Transfer for active Message Buffer
with help of configuration in Config RAM
XFER
Transfer active Buffer to System Memory
or to E-Ray CC
STATUS
Reset bit in TTSM or TTCC which
corresponds to the transferred Message
Buffer and generate status information

ETESM S/R	CESM S/R	Set bit TTSM	Clear Bit ETESMS/R
0	-	False	False
1	0	True	False
1	1	True	True

17.5.1 Transfer Control

17.5.1.1 Transfer Start and Halt

The Transfer Unit State Machine can be halted, effectively stopping the Transfer Unit transfer sequence (after completion of the current transfer cycle). After releasing from halt state, the Transfer Unit resumes exactly, where it was halted without data loss.

Note:
It is the software’s responsibility to ensure data coherency, when the FlexRay module continues to receive data, but the Transfer Unit doesn't transfer it

17.5.1.2 Transfer Abort

A Transfer Unit transfer will be aborted and the Transfer Unit will be disabled automatically in case of

- a parity error while accessing the Transfer Configuration RAM (TCR)
- a memory protection error while accessing the data RAM of the microcontroller. In this case, the ongoing transfer is aborted but the TUE bit in GCS/R may not get reset. User shall clear the TUE bit manually by software.

17.5.1.3 Transfer Reset

The Transfer Unit State Machine can be reset by the Transfer Unit Enable bit in the Global Control (GC) Register. Switching off resets the Transfer Unit State Machine, but not the module register contents and the Transfer Configuration RAM (TCR). So, after reenabling the Transfer Unit no reconfiguration of the Transfer Unit is required.

17.5.1.4 Transfer Modes

Possible transfer sequence modes are

- Manual by triggering the desired transfer by setting the according bit in the Trigger Transfer to System Memory Register (TTSM) or the Trigger Transfer to Communication Controller Register (TTCC)
- Event-Driven (transfers from FlexRay Communication Controller to the System Memory only) using the Enable Transfer on Event to System Memory Register (ETESM). The transfer trigger occurs upon completion of reception or transmission of a frame via the FlexRay bus

Note:
By setting the corresponding bit in the Enable Transfer on Event to System Memory Register (ETESM) prior to a on-demand transfer to the Communication Controller via the Trigger Transfer to Communication Controller Register (TTCC), an event triggered transmission back to the System Memory can be initiated, once the buffer has been sent out on the FlexRay bus. This mechanism can be used i.e. to automatically read back the header status information to the system memory after a transmission occurred.

- Continuous by using the Clear on Event to System Memory (CESM)

17.5.1.5 Transfer Size and Types

The data transferred by the Transfer Unit can be selected as

- data and header section
- header section only
- data section only

The amount of transferred payload words is derived from the *Payload Length Configured* (PLC) information configured by the Header Write Header Section 2 Register (WRHS2).

Note:
For transfers by the Transfer Unit only multiple of 4x32-bit data packages are supported.

Note:
The Transfer Unit always reads the number of data specified with PLC from the source, but fills up the transfer data with 0 to next 4 word boundary, due to the 4 word burst transfer. Therefore the memory buffer the TU stores the data (destination) has to be setup always to the next 4 word boundary.

17.5.1.6 Transfer Status Indication

There are 3 registers indicating the transfer status

- Transfer Status Current Buffer (TSCB) shows the current transfer buffer status
- Last Transferred Buffer to Communication Controller (LTBCC) indicates the last completed buffer transfer to the communication controller
- Last Transferred Buffer to System Memory (LTBSM) shows the last completed buffer transfer to system memory

17.5.1.7 *Transfer Mirror Function*

The following Registers can be mirrored to the system memory starting at the base address defined in the Base Address of Mirrored Status (BAMS) register:

- Transfer Status Current Buffer (TSCB)
- Last Transferred Buffer to Communication Controller (LTBCC)
- Last Transferred Buffer to System Memory (LTBSM)
- Transfer to System Memory Occurred 1/2/3/4 (TSMO1-4)
- Transfer to Communication Controller Occurred 1/2/3/4 (TCCO1-4)
- Transfer Occurred OFFset (TOOFF)

The mirrored values are updated after completion of a buffer transfer.

The mirroring of these registers can be disabled if not needed.

Figure 17-5. Mirroring Address Mapping

Address	Register
BAMS+0x00	TSCB
BAMS+0x04	LTBCC
BAMS+0x08	LTBSM
BAMS+0x0C	TSMO1
BAMS+0x10	TSMO2
BAMS+0x14	TSMO3
BAMS+0x18	TSMO4
BAMS+0x1C	TCCO1
BAMS+0x20	TCCO2
BAMS+0x24	TCCO3
BAMS+0x28	TCCO4
BAMS+0x2C	TOOFF

17.5.1.8 *Endianess Correction*

For the data transfer via the Transfer Unit an Endianess correction mechanism can be used to switch big Endianess data to little Endianess data and vice versa.

For maximum flexibility 6 bits are available in the Global Control Register to control

- Header Data byte-order
- Payload Data byte-order
- Byte-order of the FlexRay Core registers and the Transfer Configuration RAM data of the Transfer Unit

independently and in both directions.

17.5.1.9 *Transfer Data Package*

The following Figure 17-6 shows the data of a transfer data package. Regardless whether the header gets transferred or not, the buffer address always points to element “Header1”.

Figure 17-6. Transfer Data Package

0x000	Header1
0x004	Header2
0x008	Header3
0x00C	Buffer Status#
0x010	Payload1
0x014	Payload2
.....	.....
0x10C	Payload64

# transferred only from Communication Controller to System Memory

17.5.1.10 Transfer Start Address to Message Buffer Number Assignment

The assignment of a FlexRay message buffer number to the transfer location in system memory is done by the combination of

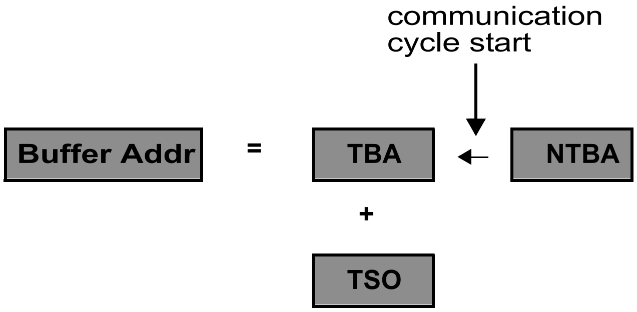
- the Transfer Start Offset (TSO) field in a Transfer Configuration RAM (TCR) entry
- the Transfer Base Address Register (TBA)

Each entry of the TCR holds a 14 bit offset value (TSO). The TSO offset will be added to the content of the TBA register. The TBA register holds the 32bit base address-pointer to a location of the data RAM.

A value written to Next Transfer Base Address (NTBA) will be loaded in the TBA at the next communication cycle start. This allows efficient multi-buffering of the message buffers in the system memory. The Transfer Not Ready (NTR) flag in the Transfer Error Interrupt (TEIR) register can be used to determine, if NTBA can be reloaded by the CPU.

Note:
If a value is written to TBA, NTBA is set to the same value.

Figure 17-7. Transfer Start Address to Message Buffer Number Assignment



17.5.1.11 Transfer Priority

The Transfer Unit will transfer the message buffers from low to high message buffer numbers.

In case the same buffer is pending in both the Trigger Transfer to Communication Register (TTCC) and the Trigger Transfer to System Memory Register (TTSM), the priority between TTCC and TTSM is determined by the Transfer Priority bit (GC.PRIO) in the Transfer Unit Global Control Register.

17.5.1.12 Read Transfers

A read transfer is the data transfer from FlexRay message buffer RAM to the system memory of the microcontroller.

For read transfers the registers Trigger Transfer to System Memory (TTSM), Enable Transfer on Event to System Memory (ETESM) and Clear on Event to System Memory (CESM) has to be setup.

The amount and type of data to be transferred can be selected as

- data and header section
- header section only
- data section only

which can be configured on the Transmit Configuration RAM (TCR).

The amount of 32bit data section words per buffer to be transferred is read from the Payload Length Configured (RDHS2.PLC(6..0)) configuration information. This information is part of the header section stored in the message RAM of the Communication Controller.

17.5.1.13 Write Transfers

A write transfer is the data transfer from the system memory of the microcontroller to the FlexRay message buffer RAM.

For write transfers the Trigger Transfer to Communication Register (TTCC) has to be setup.

The amount and type of data transferred can be selected as

- data and header section
- header section only
- data section only

which can be configured on the Transmit Configuration RAM (TCR).

It can be configured in the TCR, if Set Transmission Request Host (STXRH) bit in the Input Buffer Command Mask (IBCM) of the Communication Controller will be set. This would trigger the transfer to the FlexRay bus.

If a data and header section transfer is selected, the amount of 32bit data section words to be transferred is read from the Payload Length Configured (PLC) configuration information stored in Header2 word in the system memory.

If a data section only transfer is selected, the amount of 32bit data section words to be transferred is read from the Payload Length Configured (RDHS2.PLC(6..0)) configuration information. This information is part of the header section stored in the message RAM of the Communication Controller.

17.5.1.14 Transfer Unit Event Interface

The Transfer Unit Event Control generates transfer trigger signals for transfers in the following cases:

- For transmit (TX) message buffers, a write transfer trigger is generated, if a transmit event occurs. The configured TX message buffers generate a transfer trigger, except when a Nullframe in static segment or no frame in the dynamic segment is sent.
- For receive (RX) message buffers, a read transfer trigger is generated, if a receive event occurs in the static segment.
- For receive (RX) message buffers, a read transfer trigger is generated if a receive event occurs in the dynamic segment, updated in the actual cycle and no Nullframe.

If a buffer is part of the FIFO, no transfer trigger is generated!

17.5.2 Transfer Configuration RAM

The Transfer Configuration RAM (TCR) consists of 128 entries, one entry for each possible FlexRay buffer. Entry 1 is assigned to FlexRay buffer 1, entry 2 to FlexRay buffer 2,..., and entry 128 is assigned to FlexRay buffer 128.

Each TCR entry defines:

- the direction of FTU transfer (SM to CC or CC to SM)
- data transfer size (Header + Data, Header only or Data only)
- if transmit request, for data transferred to the CC, should be set at the FlexRay Communication Controller to send out the data to the FlexRay bus.
- the 14-bit buffer address offset, which, in combination with the Transfer Base Address defined in TBA, specifies the start of the according FlexRay message buffer in the system memory RAM.

Note:
It is recommended to clear the whole TCR before configuration, in order to avoid unexpected transfer behavior due to old configuration contents or random TCR RAM contents after power on reset.

17.5.3 Parity Protection

The Transfer Configuration RAM (TCR) is parity protected. The parity protection is switched off by default and can be switched on/off by writing a 4 bit key to dedicated parity lock bits in the Global Control Register of the Transfer Unit. The parity protection supports even and odd parity.

After a parity error in the Transfer Configuration RAM occurred, the affected address is stored in the Parity Error Address register (PEADR).

17.5.4 Memory Protection Mechanism

This feature allows to restrict accesses to certain areas in memory in order to protect critical application data from unintentionally being accessed by the *Transfer Unit State Machine*.

One memory section (start- and end address) can be defined, which allows read and write accesses for the *Transfer Unit State Machine*.

If the end address is smaller or equal to the start address, data transfers will be blocked.
For accesses outside this memory area any access performed by the *Transfer Unit State Machine* leads the State Machine not to perform the transfer. In case of a protection violation a flag will be set and the Memory Protection Violation interrupt will be activated. The *Transfer Unit State Machine* will be disabled in this case.

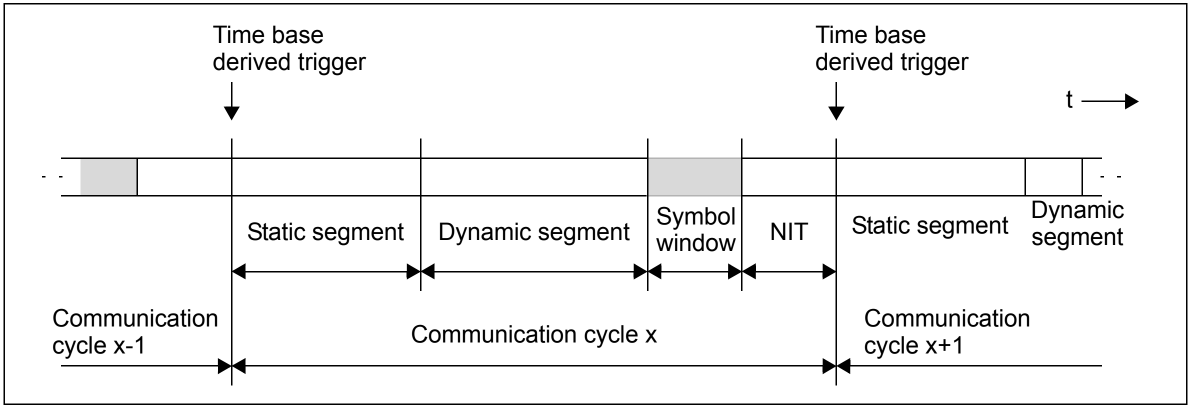
The default setting of the *Transfer Unit State Machine* memory protection address range setup is

- 0x00000000 for start address
- 0x00000000 for end address

This means a valid address range must be setup, before the Transfer Unit can be used.

17.6 Communication Cycle

Figure 17-8. Structure of Communication Cycle



A communication cycle in FlexRay consists of the following elements:

- Static segment
- Dynamic segment
- Symbol window
- Network idle time (NIT)

Static segment, dynamic segment, and symbol window form the network communication time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized macrotick.

17.6.1 Static Segment

The Static Segment is characterized by the following features:

- Time slots of fixed length (optionally protected by bus guardian)
- Start of frame transmission at action point of the respective static slot
- Payload length same for all frames on both channels

Parameters: number of static slots GTUC7.NSS[9:0], static slot length GTUC7.SSL[9:0], Payload Length Static MHDC.SFDL[6:0], action point offset GTUC9.APO[5:0]

17.6.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

- All controllers have bus access (no bus guardian protection possible)
- Variable payload length and duration of slots, different for both channels
- Start of transmission at minislot action point

Parameters: number of minislots GTUC8.NMS[12:0], minislot length GTUC8.MSL[5:0], minislot action point offset GTUC9.MAPO[4:0], start of latest transmit (last minislot) MHDC.SLT[12:0]

17.6.3 Symbol Window

During the symbol window only **one** media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in NORMAL_ACTIVE state to test the bus guardian.

The symbol window is characterized by the following features:

- Send single symbol
- Transmission of the MTS symbol starts at the symbol windows action point

Parameters: Symbol Window Action Point Offset GTUC9.APO[4:0] (same as for static slots), Network Idle Time Start GTUC4.NIT[13:0]

17.6.4 Network Idle Time (NIT)

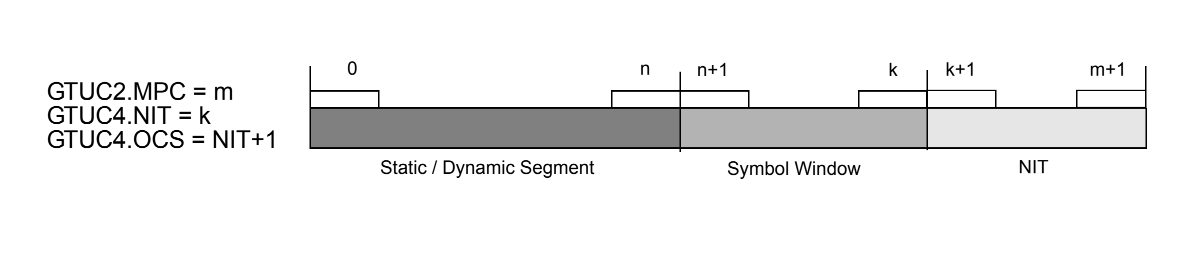
During network idle time the communication controller has to perform the following tasks:

- Calculate clock correction terms (offset and rate)
- Distribute offset correction over multiple macroticks after offset correction start
- Perform cluster cycle related tasks

Parameters: network idle time start GTUC4.NIT[13:0], offset correction start GTUC4.OCS[13:0]

17.6.5 Configuration of NIT Start and Offset Correction Start

Figure 17-9. Configuration of NIT Start and Offset Correction Start



The number of macroticks per cycle is assumed to be m. It is configured by programming GTUC2.MPC = m in the GTU Configuration Register 2.

The static / dynamic segment starts with macrotick 0 and ends with macrotick n:
n = static segment length + dynamic segment offset + dynamic segment length - 1MT

The static segment length is configured by GTUC7.SSL and GTUC7.NSS. The dynamic segment length is configured by GTUC8.MSL and GTUC8.NMS.

The dynamic segment offset is ActionPointOffset - MinislotActionPointOffset or 0 MT if the result is negative. For details please refer to the FlexRay Communications System Protocol Specification from the FlexRay Consortium.

The NIT starts with macrotick k+1 and ends with the last macrotick of cycle m-1. It has to be configured by setting GTUC4.NIT = k.

For this FlexRay module the offset correction start is required to be GTUC4.OCS >= GTUC4.NIT + 1 = k+1.

The length of symbol window results from the number of macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by k - n.

17.7 Communication Modes

The FlexRay Protocol Specification v2.1 defines the Time-Triggered Distributed (TT-D) mode.

17.7.1 Time-Triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

- Pure static: minimum 2 static slots + symbol window (optional)
- Mixed static/dynamic: minimum 2 static slots + dynamic segment + symbol window (optional)

A minimum of two coldstart nodes need to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each startup frame must be a sync frame, therefore all coldstart nodes are sync nodes.

17.8 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received sync frames from other nodes.

17.8.1 Global Time

Activities in a FlexRay node, including communication, are based on the concept of a global time, even though each individual node maintains its own view of it. It is the clock synchronization mechanism that differentiates the FlexRay cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (macrotick counter).

Cluster specific:

- Macrotick (MT) = basic unit of time measurement in a FlexRay network,
  a macrotick consists of an integer number of microticks (μT)
- Cycle length = duration of a communication cycle in units of macroticks (MT)

17.8.2 Local Time

Internally, nodes time their behavior with microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore microticks are controller-specific units. They may have different duration in different controllers. The precision of a nodes local time difference measurements is a microtick (μT).

Node specific:

- Sample clock -> prescaler -> microtick (μT); typically 25ns, see section 17.24.5.BRP(1-0) for details.
- Oscillator clock -> prescaler -> microtick (μT)
- μT = basic unit of time measurement in a communication controller, clock correction is done in units of μTs
- Cycle counter + macrotick counter = nodes local view of the global time

17.8.3 Synchronization Process

Clock synchronization is performed by means of sync frames. Only preconfigured nodes (sync nodes) are allowed to send sync frames. In a two-channel cluster a sync node has to send its sync frame on both channels.

For synchronization in FlexRay the following constraints have to be considered:

- Max. one sync frame per node in one communication cycle
- Max. 15 sync frames per cluster in one communication cycle
- Every node has to use a preconfigured number of sync frames (GTUC2.SNM[3:0]) for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of sync frames received during the static segment is measured. In a two channel cluster the sync node has to be configured to send sync frames on both channels.The calculation of correction terms is done during NIT (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay Protocol Specification v2.1, Chapter 8.

17.8.3.1 Offset (Phase) Correction

- Only deviation values measured and stored in the current cycle used
- For a two channel node the smaller value will be taken
- Calculation during NIT of **every** communication cycle
- Offset correction value calculated in even cycles used for error checking only
- Checked against limit values

- Correction value is a signed integer number of μTs
- Correction done in **odd** numbered cycles, distributed over the macroticks beginning at offset correction start up to cycle end (end of NIT) to shift nodes next start of cycle (MTs lengthened / shortened)

17.8.3.2 Rate (Frequency) Correction

- Pairs of deviation values measured and stored in even / odd cycle pair used
- For a two channel node the average of the differences from the two channels is used
- Calculated during NIT of **odd** numbered cycles
- Cluster drift damping is performed using global damping value
- Checked against limit values
- Correction value is a signed integer number of μTs
- Distributed over macroticks comprising the next **even/odd** cycle pair (MTs lengthened / shortened)

17.8.4 Sync Frame Transmission

Sync frame transmission is only possible from buffer 0 and 1. Message buffer 1 may be used for sync frame transmission in case that sync frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to ’1’.

Message buffers used for sync frame transmission have to be configured with the key slot ID and can be (re)configured in DEFAULT_CONFIG or CONFIG state only. For nodes transmitting sync frames SUCC1.TXSY must be set to ’1’.

17.8.5 External Clock Synchronization

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

- External offset / rate correction value is a signed integer
- External offset / rate correction value is added to calculated offset / rate correction value
- Aggregated offset / rate correction term (external + internal) is **not** checked against configured limits

17.9 Error Handling

The implemented error handling concept is intended to ensure that in the presence of a lower layer protocol error in a single node communication between non-affected nodes can be maintained. In some cases, higher layer program command activity is required for the communication controller to resume normal operation. A change of the error handling state will set bit EIR.PEMC and may trigger an interrupt to the host if enabled. The actual error mode is signalled by CCEV.ERRM(1–0).

Table 17-2. Error Modes of the POC (Degradation Model)

Error Mode	Activity
ACTIVE	**Full operation**, State: NORMAL_ACTIVE The communication controller is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status flags from registers EIR and SIR.
PASSIVE	**Reduced operation**, State: NORMAL_PASSIVE, communication controller self rescue allowed The communication controller stops transmitting frames and symbols, but received frames are still processed. Clock synchronization mechanisms are continued based on received frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status flags from registers EIR and SIR.
COMM_HALT	**Operation halted**, State: HALT, communication controller self rescue not allowed The communication controller stops frame and symbol processing, clock synchronization processing, and the macrotick generation. The host has still access to error and status information by reading the error and status flags from registers EIR and SIR. The bus drivers are disabled.

17.9.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the “maximum without clock correction passive” limit defined by SUCC3.WCP(3–0), the POC transits from NORMAL_ACTIVE to NORMAL_PASSIVE state. When it reaches the “maximum without clock correction fatal” limit defined by SUCC3.WCP(3–0), it transits NORMAL_ACTIVE or NORMAL_PASSIVE to the HALT state.

The Clock Correction Failed Counter CCEV.CCFC(3–0) allows the host to monitor the duration of the inability of a node to compute clock correction terms after the communication controller passed protocol startup phase. It will be incremented by one at the end of any odd numbered communication cycle where either the Missing Offset Correction flag SFS.MOCS or the Missing Rate Correction flag SFS.MRCS is set.

The clock correction failed counter is reset to zero at the end of an odd communication cycle if neither the Missing Offset Correction flag SFS.MOCS nor the Missing Rate Correction flag SFS.MRCS is set.

The Clock Correction Failed Counter stops incrementing when the “maximum without clock correction fatal” value SUCC3.WCP(3–0) is reached (i.e. incrementing the counter at its maximum value will not cause it to wraparound back to zero). The clock correction failed counter will be initialized to zero when the communication controller enters READY state or when NORMAL_ACTIVE state is entered.

Note:
The transition to HALT state is prevented if SUCC1.HCSE is not set.

17.9.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from NORMAL_PASSIVE to NORMAL_ACTIVE state. SUCC1.PTA(4–0) defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the communication controller is allowed to transit from

NORMAL_PASSIVE to NORMAL_ACTIVE state. If SUCC1.PTA(4–0) is set to zero the communication controller is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state.

17.9.3 HALT Command

In case the host wants to stop FlexRay communication of the local node it can bring the communication controller into HALT state by asserting the HALT command. This can be done by writing SUCC1.CMD(3–0) = 0110. In order to shut down communication on an entire FlexRay network, a higher layer protocol is required to assure that all nodes apply the HALT command at the same time.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL[5:0].

When called in NORMAL_ACTIVE or NORMAL_PASSIVE state the POC transits to HALT state at the end of the current cycle. When called in any other state SUCC1.CMD(3–0) will be reset to 0000 = “command_not_accepted” and bit EIR.CNA in the error interrupt register is set to 1. If enabled an interrupt to the host is generated.

17.9.4 FREEZE Command

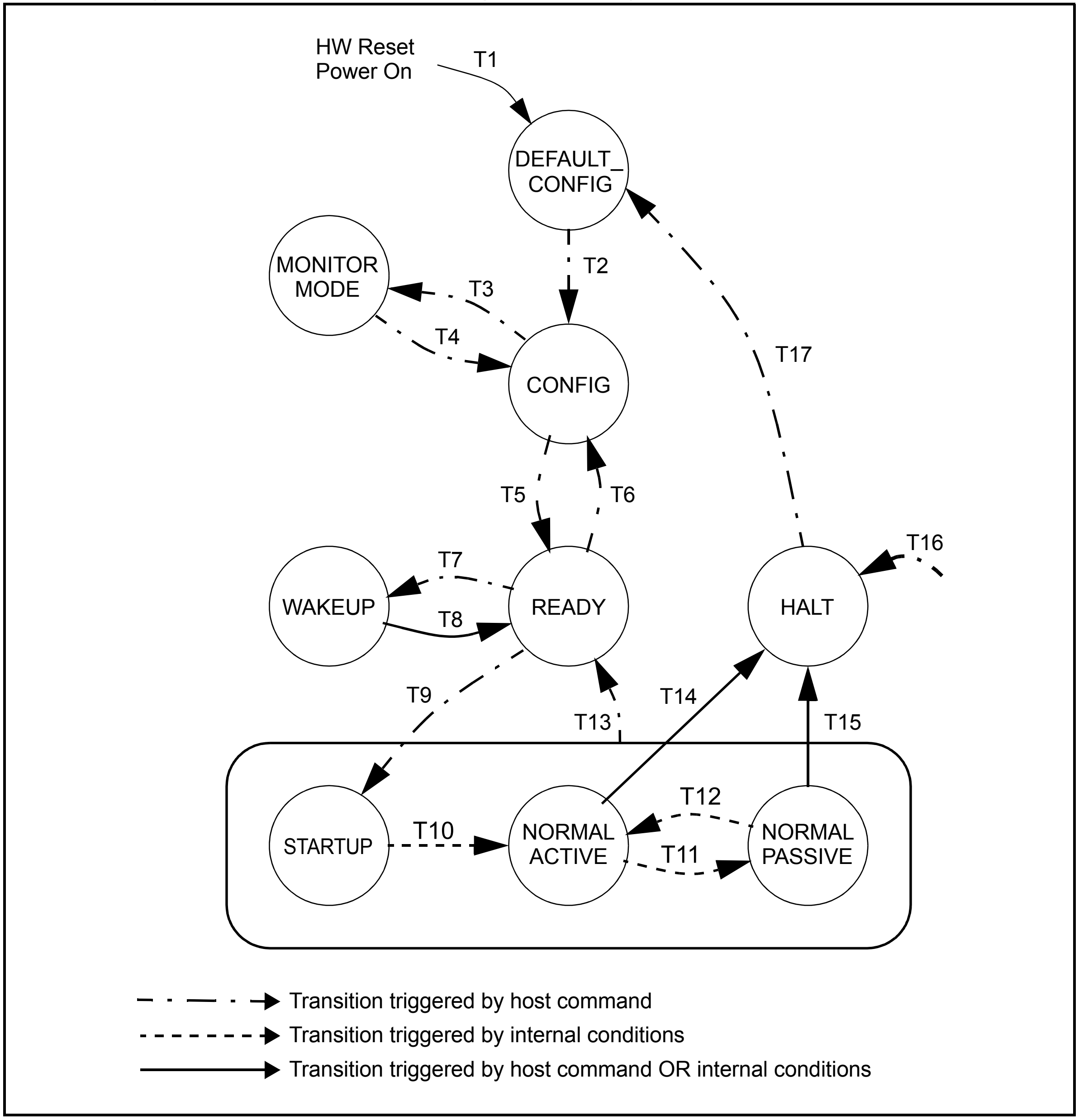
In case the host detects a severe error condition it can bring the communication controller into HALT state by asserting the FREEZE command. This can be done by writing SUCC1.CMD(3–0) = 0111. The FREEZE command triggers the entry of the HALT state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL[5:0].

17.10 Communication Controller States

17.10.1 Communication Controller State Diagram

Figure 17-10. Overall State Diagram of E-Ray Communication Controller


HW Reset
Power On
T1
DEFAULT_
CONFIG
T2
MONITOR
MODE
T3
T4
CONFIG
T17
T5
T6
WAKEUP
T7
T8
READY
HALT
T16
T9
T13
T14
T15
STARTUP
T10
NORMAL
ACTIVE
T12
T11
NORMAL
PASSIVE
Transition triggered by host command
Transition triggered by internal conditions
Transition triggered by host command OR internal conditions

State transitions are controlled by the reset and FlexRay receive (rxd1,2) pins, the POC state machine, and by the CHI command vector SUCC1.CMD(3–0).

The Communication Controller exits from all states to HALT state after application of the FREEZE command (SUCC1.CMD(3–0) = "0111").

Table 17-3. State Transitions of Communication Controller Overall State Machine

T#	Condition	From	To
1	Hardware reset	All States	DEFAULT_CONFIG
2	Command CONFIG, SUCC1.CMD(3–0) = 0001	DEFAULT_CONFIG	CONFIG
3	Unlock sequence followed by command MONITOR_MODE, SUCC1.CMD(3–0) = 1011	CONFIG	MONITOR_MODE
4	Command CONFIG, SUCC1.CMD(3–0) = 0001	MONITOR_MODE	CONFIG
5	Unlock sequence followed by command READY, SUCC1.CMD(3–0) = 0010	CONFIG	READY
6	Command CONFIG, SUCC1.CMD(3–0) = 0001	READY	CONFIG
7	Command WAKEUP, SUCC1.CMD(3–0) = 0011	READY	WAKEUP
8	Complete, non-aborted transmission of wakeup pattern OR received WUP OR received frame header OR command READY, SUCC1.CMD(3–0) = 0010	WAKEUP	READY
9	Command RUN, SUCC1.CMD(3–0) = 0100	READY	STARTUP
10	Successful startup	STARTUP	NORMAL_ACTIVE
11	Clock correction failed counter reached “maximum without clock correction passive” limit configured by SUCC3.WCP(3–0)	NORMAL_ACTIVE	NORMAL_PASSIVE
12	Number of valid correction terms reached the Passive to Active limit configured by SUCC1.PTA(4–0)	NORMAL_PASSIVE	NORMAL_ACTIVE
13	Command READY, SUCC1.CMD(3–0) = 0010	STARTUP, NORMAL_ACTIVE, NORMAL_PASSIVE	READY
14	Clock Correction Failed counter reached “maximum without clock correction fatal” limit configured by SUCC3.WCF(3–0) AND bit SUCC1.HCSE set to 1 OR command HALT, SUCC1.CMD(3–0) = 0110	NORMAL_ACTIVE	HALT
15	Clock Correction Failed counter reached “maximum without clock correction fatal” limit configured by SUCC3.WCF(3–0) AND bit SUCC1.HCSE set to 1 OR command HALT, SUCC1.CMD(3–0) = 0110	NORMAL_PASSIVE	HALT
16	Command FREEZE, SUCC1.CMD(3–0) = 0111	All States	HALT
17	Command CONFIG, SUCC1.CMD(3–0) = 0001	HALT	DEFAULT_CONFIG

17.10.2 DEFAULT_CONFIG State

In DEFAULT_CONFIG state, the communication controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The communication controller enters this state:

- When leaving hardware reset
- When exiting from HALT state

To leave DEFAULT_CONFIG state the host has to write SUCC1.CMD(3–0) = 0001. The communication controller then transits to CONFIG state.

17.10.3 CONFIG State

In CONFIG state, the communication controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the communication controller configuration.

The communication controller enters this state

- When exiting from DEFAULT_CONFIG state
- When exiting from MONITOR_MODE or READY state

When the state has been entered via HALT and DEFAULT_CONFIG state, the host can analyses status information and configuration. Before leaving CONFIG state the host has to assure that the configuration is fault-free.

To leave CONFIG state, the host has to perform the unlock sequence. Directly after unlocking the CONFIG state the host has to write SUCC1.CMD(3–0) to enter the next state.

Note:
The message buffer status registers (MHDS, TXRQ1/2/3/4, NDAT1/2/3/4, MBSC1/2/3/4) and status data stored in the message RAM and are not affected by the transition of the POC from CONFIG to READY state.

When the communication controller is in CONFIG state it is also possible to bring the communication controller into a power saving mode by halting the module clocks. To do this the host has to assure that all Message RAM transfers have finished before turning off the clocks.

17.10.4 MONITOR_MODE

After unlocking CONFIG state and writing SUCC1.CMD(3–0) = 1011 the communication controller enters MONITOR_MODE. In this mode the communication controller is able to receive FlexRay frames and to detect wakeup pattern. The temporal integrity of received frames is not checked, and therefore cycle counter filtering is not supported. This mode can be used for debugging purposes in case e.g. that startup of a FlexRay network fails. After writing SUCC1.CMD(3–0) = 0001 the communication controller transits back to CONFIG state.

In MONITOR_MODE the pick first valid mechanism is disabled. This means that a receive message buffer may only be configured to receive on one channel. Received frames are stored into message buffers according to frame ID and receive channel. Null frames are handled like data frames. After frame reception only status bits MBS.VFRA, MBS.VFRB, MBS.MLST, MBS.RCIS, MBS.SFIS, MBS.SYNS, MBS.NFIS, MBS.PPIS, MBS.RESS have valid values.

In MONITOR_MODE the communication controller is not able to distinguish between CAS and MTS symbols. In case one of these symbols is received on one or both of the two channels, the flags SIR.MTSA resp. SIR.MTSB are set. SIR.CAS has no function in MONITOR_MODE.

17.10.5 READY State

After unlocking CONFIG state and writing SUCC1.CMD(3–0) = 0010 the communication controller enters READY state. From this state the communication controller can transit to WAKEUP state and perform a cluster wakeup or to STARTUP state to perform a coldstart or to integrate into a running cluster.

The communication controller enters this state

- When exiting from CONFIG, WAKEUP, STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state by writing SUCC1.CMD(3–0) = 0010 (READY command).

The communication controller exits from this state

- To CONFIG state by writing SUCC1.CMD(3–0) = 0001 (CONFIG command)
- To WAKEUP state by writing SUCC1.CMD(3–0) = 0011 (WAKEUP command)
- To STARTUP state by writing SUCC1.CMD(3–0) = 0100 (RUN command)

Internal counters and the communication controller status flags are reset when the communication controller enters STARTUP state.

Note:
Status bits MHDS[14:0], registers TXRQ1/2/3/4, and status data stored in the Message RAM are not affected by the transition of the POC from READY to STARTUP state.

17.10.6 WAKEUP State

The description below is intended to help configuring wakeup for the FlexRay module. A detailed description of the wakeup procedure can be found in the FlexRay Protocol Specification v2.1, Section 7.1.

The communication controller enters this state

- When exiting from READY state by writing SUCC1.CMD(3–0) = 0011 (WAKEUP command).

The communication controller exits from this state to READY state

- After complete non-aborted transmission of wakeup pattern
- After WUP reception
- After detecting a WUP collision
- After reception of a frame header
- By writing SUCC1.CMD(3–0) = 0010 (READY command)

The communication controller exits from this state to HALT state

- By writing SUCC1.CMD(3–0) = 0111 (FREEZE command)

The cluster wakeup must precede the communication startup in order to ensure that all nodes in a cluster are awake. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an **external** wakeup source.

The host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the communication controller and configures bus guardian (if available) and communication controller to perform the cluster wakeup. The communication controller provides to the host the ability to transmit a special wakeup pattern on each of its available channels separately. The communication controller needs to recognize the wakeup pattern only during wakeup and startup phase.

Wakeup may be performed on only one channel at a time. The host has to configure the wakeup channel while the communication controller is in CONFIG state by writing bit WUCS in the SUC Configuration Register 1. The communication controller ensures that ongoing communication on this channel is not disturbed. The communication controller cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the communication controller returns to READY state and signals the change of the wakeup status to the host by setting bit SIR.WST in the status interrupt register. The wakeup status vector CCSV.WSV(2–0) can be read from the communication controller status vector register. If a valid wakeup pattern was received also either bit SIR.WUPA or bit SIR.WUPB in the status interrupt register is set.

Figure 17-11. Structure of POC State WAKEUP

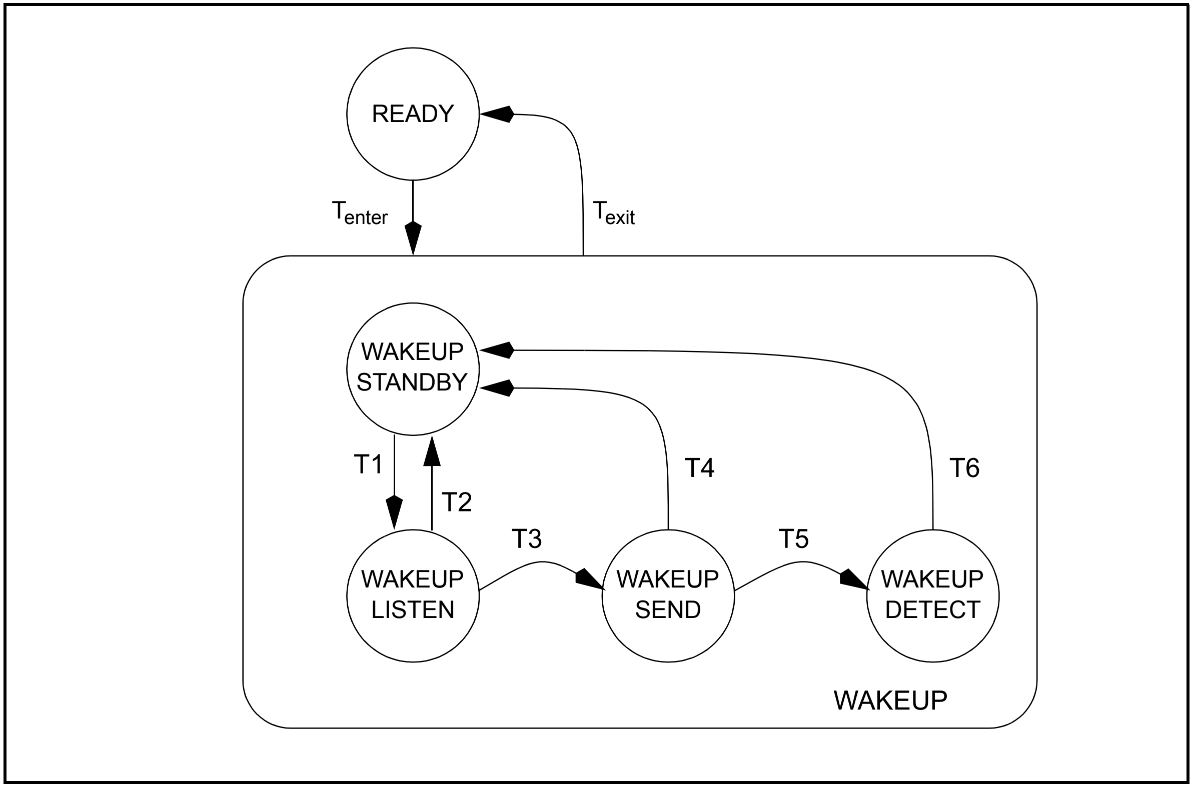


Table 17-4. State Transitions WAKEUP

T	Condition	From	To
enter	Host commands change to WAKEUP state by writing SUCC1.CMD(3–0) = 0011 (WAKEUP command)	READY	WAKEUP
1	CHI command WAKEUP triggers wakeup FSM to transit to WAKEUP_LISTEN state	WAKEUP_STANDBY	WAKEUP_LISTEN
2	Received WUP on wakeup channel selected by bit SUCC1.WUCS OR frame header on either available channel	WAKEUP_LISTEN	WAKEUP_STANDBY
3	Timer event	WAKEUP_LISTEN	WAKEUP_SEND
4	Complete, non-aborted transmission of wakeup pattern	WAKEUP_SEND	WAKEUP_STANDBY
5	Collision detected	WAKEUP_SEND	WAKEUP_DETECT
6	Wakeup timer expired OR WUP detected on wakeup channel selected by bit SUCC1.WUCS OR frame header received on either available channel	WAKEUP_DETECT	WAKEUP_STANDBY
exit	Wakeup completed (after T2 or T4 or T6) OR host commands change to READY state by writing SUCC1.CMD(3–0) = 0010 (READY command). This command also resets the wakeup FSM to WAKEUP_STANDBY state.	WAKEUP	READY

The WAKEUP_LISTEN state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters Listen Timeout SUCC2.LT[20:0] and Listen Timeout Noise SUCC2.LTN(3–

0). Listen timeout enables a fast cluster wakeup in case of a noise free environment, while listen timeout noise enables wakeup under more difficult conditions regarding noise interference.

In WAKEUP_SEND state the communication controller transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the host has to bring the communication controller into STARTUP state by CHI command RUN.

In WAKEUP_DETECT state the communication controller attempts to identify the reason for the wakeup collision detected in WAKEUP_SEND state. The monitoring is bounded by the expiration of listen timeout as configured by SUCC2.LT[20:0] in the SUC Configuration Register 2. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a frame header indication existing communication, causes the direct transition to READY state. Otherwise WAKEUP_DETECT is left after expiration of listen timeout; in this case the reason for wakeup collision is unknown.

The host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay Protocol Specification recommends that two different communication controllers shall awake the two channels.

17.10.6.1 Host Activities

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the host. The wakeup pattern is detected by the remote BDs and signalled to their local hosts.

Wakeup procedure controlled by host (single-channel wakeup):

- Configure the communication controller in CONFIG state
  - Select wakeup channel by programming bit SUCC1.WUCS
- Check local BDs whether a WUP was received
- Activate BD of selected wakeup channel
- Command communication controller to enter READY state
- Command communication controller to start wakeup on the configured channel by writing SUCC1.CMD(3–0) = 0011
  - communication controller enters WAKEUP_LISTEN
  - communication controller returns to READY state and signals status of wakeup attempt to host
- Wait predefined time to allow the other nodes to wakeup and configure themselves
- Coldstart node:
  - in dual channel cluster wait for WUP on the other channel
  - Reset Coldstart Inhibit flag CCSV.CSI by writing SUCC1.CMD(3–0) = 1001 (ALLOW_COLDSTART command)
- Command communication controller to enter startup by writing SUCC1.CMD(3–0) = 0100 (RUN command)

Wakeup procedure triggered by the bus driver:

- Wakeup recognized by bus driver
- bus driver triggers power-up of host (if required)
- bus driver signals wakeup event to host
- Host configures its local communication controller
- If necessary host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves

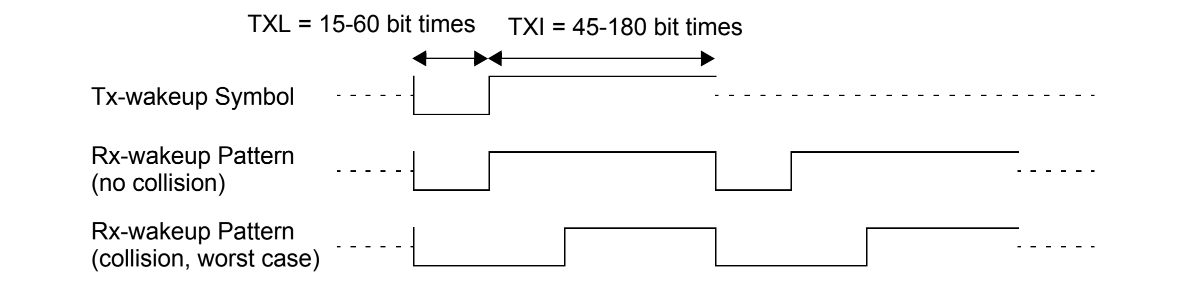
- Host commands communication controller to enter STARTUP state by writing SUCC1.CMD(3–0) = 0100 (RUN command)

17.10.6.2 Wake Up Pattern (WUP)

The wake up pattern (WUP) is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by the PRT Configuration Registers 1 and 2.

- Single channel wakeup, wake up symbol may not be sent on both channels at the same time
- Wake up symbol collision resilient for up to two sending nodes
  (two overlapping wakeup symbols still recognizable)
- Wake up symbol must be configured identical in all nodes of a cluster
- Wakeup symbol transmit low time configured by PRTC2.TXL(5–0), see section 17.24.6 for details.
- Wakeup symbol idle time configured by PRTC2.TXI(7–0), used to listen for activity on the bus, see section 17.24.6 for details.
- A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup
- Number of repetitions configurable by PRTC1.RWP(5–0) (2 to 63 repetitions), see section 17.24.5 for details.
- Wakeup symbol receive window length configured by PRTC1.RXW(8–0), see section 17.24.5 for details.
- Wakeup symbol receive low time configured by PRTC2.RXL(5–0), see section 17.24.6 for details.
- Wakeup symbol receive idle time configured by PRTC2.RXI(5–0), see section 17.24.6 for details.

Figure 17-12. Timing of Wake Up Pattern



17.10.7 STARTUP State

The description below is intended to help configuring startup for the FlexRay module. A detailed description of the startup procedure can be found in the FlexRay Protocol Specification v2.1, Section 7.2.

Any node entering STARTUP state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup frames. Startup frames are both sync frames and null frames during startup.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter NORMAL_ACTIVE state via (see Figure 17-13):

- Coldstart path initiating the schedule synchronization (leading coldstart node)
- Coldstart path joining other coldstart nodes (following coldstart node)
- Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has bits SUCC1.TXST and SUCC1.TXSY set to 1. Message buffer 0 holds the key slot ID which defines the slot number where the startup frame is send. In the frame header of the startup frame the startup frame indicator bit is set.

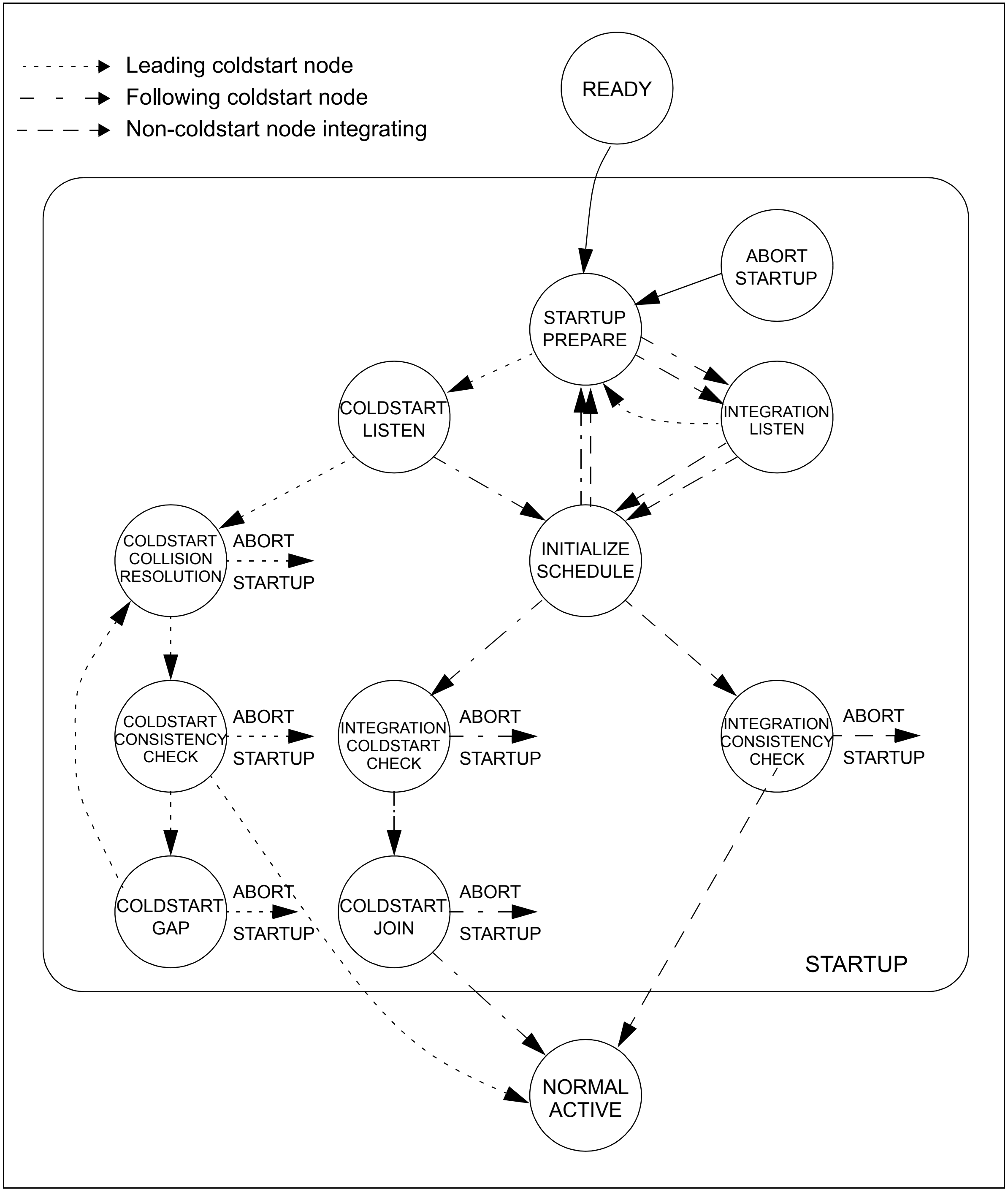
In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each startup frame must also be a sync frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by SUCC1.CSA(4–0) in the SUC Configuration Register 1.

A non-coldstart node requires at least two startup frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive sync frames from which to derive the TDMA schedule information. During integration the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

Figure 17-13. State Diagram Time-Triggered Startup



17.10.7.1 Coldstart Inhibit Mode

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit CCSV.CSI in the Communication Controller Status Vector Register is set, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit CCSV.CSI is set whenever the POC enters READY state. The bit has to be cleared under control of the host by CHI command ALLOW_COLDSTART (SUCC1.CMD(3–0) = 1001)

17.10.7.2 Startup Timeouts

The communication controller supplies two different µT timers supporting two timeout values, startup timeout and startup noise timeout. The two timers are started when the communication controller enters the COLDSTART_LISTEN state. The expiration of either of these timers causes the node to leave the initial sensing phase (COLDSTART_LISTEN state) with the intention of starting up communication.

Note:
The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values SUCC2.LT[20:0] and SUCC2.LTN(3–0) from the SUC Configuration Register 2.

Startup Timeout

The startup timeout limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others. The startup timer is configured by programming SUCC2.LT[20:0] in the SUC Configuration Register 2.

The startup timeout time can be calculated from the contents of SUCC2.LT[20:0] (Reference to the FlexRay Protocol Specification: pdListenTimeout)

The startup timer is restarted upon:

- Entering the COLDSTART_LISTEN state
- Both channels reaching idle state while in COLDSTART_LISTEN state

The startup timer is stopped:

- If communication channel activity is detected on one of the configured channels while the node is in the COLDSTART_LISTEN state
- When the COLDSTART_LISTEN state is left

Once the startup timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

Startup Noise Timeout

At the same time the startup timer is started for the first time (transition from STARTUP_PREPARE state to COLDSTART_LISTEN state), the startup noise timer is started. This additional timeout is used to improve reliability of the startup procedure in the presence of noise. The startup noise timer is configured by programming SUCC2.LTN(3–0) in the SUC Configuration Register 2.

The startup noise timeout time can be calculated as the product of SUCC2.LT[20:0] * SUCC2.LTN(3–0) (Reference to the FlexRay Protocol Specification: pdListenTimeout • gListenNoise)

The startup noise timer is restarted upon:

- Entering the COLDSTART_LISTEN state
- Reception of correctly decoded headers or CAS symbols while the node is in COLDSTART_LISTEN state

The startup noise timer is stopped when the COLDSTART_LISTEN state is left.

Once the startup noise timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won't be restarted when random channel activity is sensed, this timeout defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

17.10.7.3 Path of Leading Coldstart Node (Initiating Coldstart)

When a coldstart node enters COLDSTART_LISTEN, it listens to its attached channels.

If no communication is detected, the node enters the COLDSTART_COLLISION_RESOLUTION state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup frame. Since each coldstart node is allowed to perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a frame header during these four cycles, it re-enters the COLDSTART_LISTEN state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup frames.

After four cycles in COLDSTART_COLLISION_RESOLUTION state, the node that initiated the coldstart enters the COLDSTART_CONSISTENCY_CHECK state. It collects all startup frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid startup frame pair, the node leaves COLDSTART_CONSISTENCY_CHECK and enters NORMAL_ACTIVE state.

The number of coldstart attempts that a node is allowed to perform is configured by SUCC1.CSA(4–0) in the SUC configuration Register 1. The number of remaining coldstarts attempts can be read from CCSV.RCA(4–0) of communication controller status vector register. The number of remaining attempts is reduced by one for each attempted coldstart. A node may enter the COLDSTART_LISTEN state only if this value is larger than *one* and it may enter the COLDSTART_COLLISION_RESOLUTION state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

17.10.7.4 Path of Following Coldstart Node (Responding to Leading Coldstart Node)

When a coldstart node enters the COLDSTART_LISTEN state, it tries to receive a valid pair of startup frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid startup frame has been received, the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_COLDSTART_CHECK state is entered.

In INTEGRATION_COLDSTART_CHECK state, it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all sync frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient frames from the same node it has integrated on, the COLDSTART_JOIN state is entered.

In COLDSTART_JOIN state, following coldstart nodes begin to transmit their own startup frames and continue to do so in subsequent cycles. Thereby, the leading coldstart node and the nodes joining it can check if their schedules agree with each other. If the clock correction signals any error, the node aborts the integration attempt. If a node in this state sees at least one valid startup frame during all even cycles in this state and at least one valid startup frame pair during all double cycles in this state, the node leaves COLDSTART_JOIN state and enters NORMAL_ACTIVE state. Thereby it leaves STARTUP at least one cycle after the node that initiated the coldstart.

17.10.7.5 Path of Non-coldstart Node

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels.

As soon as a valid startup frame has been received the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_CHECK state is entered.

In INTEGRATION_CONSISTENCY_CHECK state it is verified that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) send startup frames that agree with the nodes own schedule. Clock correction is activated, and if any errors are signalled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup frames or the startup frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid startup frame pairs or the startup frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup frames are received within an even cycle, or less than two valid startup frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid startup frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_OPERATION. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

17.10.8 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering STARTUP via coldstart path) and one additional node have entered the NORMAL_ACTIVE state, the startup phase for the cluster has finished. In the NORMAL_ACTIVE state, all configured messages are scheduled for transmission. This includes all data frames as well as the sync frames. Rate and offset measurement is started in all even cycles (even/odd cycle pairs required).

In NORMAL_ACTIVE state the communication controller supports regular communication functions

- The communication controller performs transmissions and reception on the FlexRay bus as configured
- Clock synchronization is running
- The host interface is operational

The communication controller exits from that state to

- HALT state by writing SUCC1.CMD(3–0) = 0110 (HALT command, at the end of the current cycle)
- HALT state by writing SUCC1.CMD(3–0) = 0111 (FREEZE command, immediately)
- HALT state due to change of the error state from ACTIVE to COMM_HALT
- NORMAL_PASSIVE state due to change of the error state from ACTIVE to PASSIVE
- READY state by writing SUCC1.CMD(3–0) = 0010 (READY command)

17.10.9 NORMAL_PASSIVE State

NORMAL_PASSIVE state is entered from NORMAL_ACTIVE state when the error state changes from ACTIVE to PASSIVE.

In NORMAL_PASSIVE state, the node is able to receive all frames (node is fully synchronized and performs clock synchronization). Contrary to the NORMAL_ACTIVE state, the node does not actively participate in communication, i.e. neither symbols nor frames are transmitted.

In NORMAL_PASSIVE state:

- The communication controller performs reception on the FlexRay bus
- The communication controller does not transmit any frames or symbols on the FlexRay bus
- Clock synchronization is running
- The host interface is operational

The communication controller exits from this state to

- HALT state by writing SUCC1.CMD(3–0) = 0110 (HALT command, at the end of the current cycle)
- HALT state by writing SUCC1.CMD(3–0) = 0111 (FREEZE command, immediately)
- HALT state due to change of the error state from PASSIVE to COMM_HALT
- NORMAL_ACTIVE state due to change of the error state from PASSIVE to ACTIVE. The transition takes place when CCEV.PTAC(4–0) equals SUCC1.PTA(4–0) - 1.
- To READY state by writing SUCC1.CMD(3–0) = 0010 (READY command)

17.10.10HALT State

In this state all communication (reception and transmission) is stopped.

The communication controller enters this state

- By writing SUCC1.CMD(3–0) = 0110 (HALT command) while the communication controller is in NORMAL_ACTIVE or NORMAL_PASSIVE state
- By writing SUCC1.CMD(3–0) = 0111 (FREEZE command) from all states
- When exiting from NORMAL_ACTIVE state because the clock correction failed counter reached the “maximum without clock correction fatal” limit and SUCC1.HCSE is set
- When exiting from NORMAL_PASSIVE state because the clock correction failed counter reached the “maximum without clock correction fatal” limit and SUCC1.HCSE is set

The communication controller exits from this state to DEFAULT_CONFIG state

- By writing SUCC1.CMD(3–0) = 0001 (CONFIG command)

When the communication controller transits from HALT state to DEFAULT_CONFIG state all configuration and status data is maintained for analyzing purposes.

When the host writes SUCC1.CMD(3–0) = 0110 (HALT command), the communication controller sets bit CCSV.HRQ and enters HALT state at next end of cycle.

When the host writes SUCC1.CMD(3–0) = 0111 (FREEZE command), the communication controller enters HALT state immediately and sets the CCSV.FSI bit in the communication controller status vector register.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL[5:0].

17.11 Network Management

The accrued network management (NM) vector is located in the Network Management Register (NMV) 1…3. The communication controller performs a logical OR operation over all NM vectors out of all received valid NM frames with the Payload Preamble Indicator (PPI) bit set. Only a static frame may be configured to hold NM information. The communication controller updates the NM vector at the end of each cycle.

The length of the NM vector can be configured from 0 to 12 bytes by NEMC.NML(3–0). The NM vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay frames with the PPI bit set, the PPIT bit in the header section of the respective transmit buffer has to be set via WRHS1.PPIT. In addition the host has to write the NM information to the data section of the respective transmit buffer.

The evaluation of the NM vector has to be done by the application running on the host.

Note:
In case a message buffer is configured for transmission / reception of network management frames, the payload length configured in header 2 of that message buffer should be equal or greater than the length of the NM vector configured by NEMC.NML[3:0]. When the Communication Controller transits to HALT state, the cycle count is not incremented and therefore the NM vector is not updated. In this case NMV1..3 holds the value from the cycle before.

17.12 Filtering and Masking

Filtering is done by comparison of the configuration of assigned message buffers against actual slot and cycle counter values and channel ID (channel A, B). A message buffer is only updated / transmitted if the required matches occur.

Filtering is done on:

- Slot counter
- Cycle counter
- Channel ID

The following filter combinations for acceptance / transmit filtering are allowed:

- Slot counter + Channel ID
- Slot counter + Cycle counter + Channel ID

All configured filters must match in order to store a received message in a message buffer.

> **Note:**
> For the FIFO the acceptance filter is configured by the FIFO Rejection Filter and the FIFO Rejection Filter mask.

A message will be transmitted in the time slot corresponding to the configured frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

17.12.1 Slot Counter Filtering

Every transmit and receive buffer contains a frame ID stored in the header section. This frame ID is compared against the actual slot counter value in order to assign receive and transmit buffers to the corresponding slot.

If two or more message buffers are configured with the same frame ID, and if they have a matching cycle counter filter value for the same slot, then the message buffer with the **lowest** message buffer number is used.

17.12.2 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in the header section 1 of each message buffer.

If message buffer 0 resp. 1 is configured to hold the startup / sync frame or the single slot frame by bits SUCC1.TXST, SUCC1.TXSY, and SUCC1.TSM, cycle counter filtering for message buffer 0 resp. 1 must be disabled.

> **Note:**
> Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay network is **not** allowed.

The set of cycle numbers belonging to a cycle set is determined as described in Table 17-5.

Table 17-5. Definition of Cycle Set

Cycle Code	Matching Cycle Counter Values		
0b000000x	All cycles		
0b000001***c***	Every second cycle	at (cycle count)mod2	= ***c***
0b00001***cc***	Every fourth cycle	at (cycle count)mod4	= ***cc***
0b0001***ccc***	Every eighth cycle	at (cycle count)mod8	= ***ccc***

Table 17-5. Definition of Cycle Set

Cycle Code	Matching Cycle Counter Values		
0b001**1*cccc***	Every sixteenth cycle	at (cycle count)mod16	= ***cccc***
0b01**1*ccccc***	Every thirty-second cycle	at (cycle count)mod32	= ***ccccc***
0b**1*cccccc***	Every sixty-fourth cycle	at (cycle count)mod64	= ***cccccc***

Table 17-6 below gives some examples for valid cycle sets to be used for cycle counter filtering:

Table 17-6. Examples for Valid Cycle Sets

Cycle Code	Matching Cycle Counter Values
0b00000**1*1***	1-3-5-7- .... -63 ↲
0b0000**1*00***	0-4-8-12- .... -60 ↲
0b000**1*110***	6-14-22-30- .... -62 ↲
0b00**1*1000***	8-24-40-56 ↲
0b0**1*00011***	3-35 ↲
0b**1*001001***	9 ↲

The received message is stored only if the cycle counter value of the cycle during which the message is received matches an element of the receive buffer’s cycle set. Other filter criteria must also be met.

The content of a transmit buffer is transmitted on the configured channel(s) when an element of the cycle set matches the current cycle counter value. Other filter criteria must also be met.

17.12.3 Channel ID Filtering

There is a 2-bit channel filtering field (CHA, CHB) located in the header section of each message buffer in the message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers (see table 17-7).

Table 17-7. Channel Filtering Configuration

CHA	CHB	Transmit Buffer transmit frame	Receive Buffer store valid receive frame
1	1	On both channels (static segment only)	Received on channel A or B (store first semantically valid frame, static segment only)
1	0	On channel A	Received on channel A
0	1	On channel B	Received on channel B
0	0	No transmission	Ignore frame

The contents of a transmit buffer is transmitted on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be set up for transmission on both channels (**CHA** and **CHB** set).

Valid received frames are stored if they are received on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a receive buffer may be setup for reception on both channels (**CHA** and **CHB** set).

Note:
If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no frames are transmitted resp. received frames are ignored (same function as CHA = CHB = 0)

17.12.4 FIFO Filtering

For FIFO filtering there is one rejection filter and one rejection filter mask available. The FIFO filter consists of channel filter FRF.CH[1:0], frame ID filter FRF.FID[10:0], and cycle counter filter FRF.CYF[6:0]. Registers FRF and FRFM can be configured in DEFAULT_CONFIG or CONFIG state only. The filter configuration in the header section of message buffers belonging to the FIFO is ignored.

The 7-bit cycle counter filter determines the cycle set to which frame ID and channel rejection filter are applied. In cycles not belonging to the cycle set specified by FRF.CYF[6:0], all frames are rejected.

A valid received frame is stored in the FIFO if channel ID, frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

17.13 Transmit Process

17.13.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the frame ID corresponding to the next sending slot is selected for transmission.

The data section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the input buffer has to be started by writing to the Input Buffer Command Request Register latest at this time.

17.13.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest frame ID) is selected next. In the dynamic segment different slot counter sequences on channel A and channel B are possible (concurrent sending of different frame IDs on both channels).

The data section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the input buffer has to be started by writing to the Input Buffer Command Request Register latest at this time.

The start of latest transmit configured by MHDC.SLT(12–0) in the MHD configuration register 1 defines the maximum minislot value allowed before inhibiting new frame transmission in the dynamic segment of the current cycle.

17.13.3 Transmit Buffers

E-Ray message buffers can be configured as transmit buffers by programming bit **CFG** in the header section of the respective message buffer to '1' via WRHS1.

There exist the following possibilities to assign a transmit buffer to the communication controller channels:

- Static segment:channel A **or** channel B,
  channel A **and** channel B
- Dynamic segment:channel A **or** channel B

Message buffer 0 resp. 1 is dedicated to hold the startup frame, the sync frame, or the designated single slot frame as configured by SUCC1.TXST, SUCC1.TXSY, and SUCC1.TSM in the SUC Configuration Register 1. In this case it can be reconfigured in DEFAULT_CONFIG or CONFIG state only. This ensures that any node transmits at most one startup / sync frame per communication cycle. Transmission of startup / sync frames from other message buffers is not possible.

All other message buffers configured for transmission in static or dynamic segment are reconfigurable during runtime depending on the configuration of MRC.SEC[1:0]. Due to the organization of the data partition in the message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the header section of a message buffer may lead to erroneous configurations.

If a message buffer is reconfigured (header section updated) during runtime, it may happen that this message buffer is not send out in the respective communication cycle.

The communication controller does not have the capability to calculate the header CRC. The host is supposed to provide the header CRCs for all transmit buffers. If network management is required the host has to set the PPIT bit in the header section of the respective message buffer to 1 and write the network management information to the data section of the message buffer (see section 17.11).

The payload length field configures the data payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by MHDC.SFDL(6–0) in the Message Handler Configuration Register 1, the communication controller generates padding bytes to ensure that frames have proper physical length. The padding pattern is logical zero.

Note:
In case of an odd payload length (PLC=1,3,5,...) the application will write zero to the last 16 bit of the message buffers data section to ensure that the padding pattern is all zero.

Each transmit buffer provides a transmission mode flag TXM that allows the host to configure the transmission mode for the transmit buffer. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode.

In single-shot mode the Communication Controller resets the respective TXR flag after transmission has completed. Now the Host may update the transmit buffer.

In continuous mode, the Communication Controller does not reset the respective transmission request flag TXR after successful transmission. In this case a frame is sent out each time the filter criteria match. The TXR flag can be reset by the Host by writing the respective message buffer number to the IBCR register while bit IBCM.STXRH is set to ’0’.

If two or more transmit buffers meet the filter criteria simultaneously, the transmit buffer with the lowest message buffer number will be transmitted in the respective slot.

17.13.4 Frame Transmission

The following steps are required to prepare a message buffer for transmission:

- Configure the transmit buffer in the Message RAM via WRHS1, WRHS2, and WRHS3
- Write the data section of the transmit buffer via WRDSn
- Transfer the configuration and message data from Input Buffer to the Message RAM by writing the number of the target message buffer to register IBCR
- If configured in the Input Buffer Command Mask (IBCM) register the Transmission request flag (TXR) for the respective message buffer will be set as soon as the transfer has completed, and the message buffer is ready for transmission.
- Check whether the message buffer has been transmitted by checking the TXR bits (TXR = 0) in the Transmission request 1,2,3,4 registers (single-shot mode only).

After transmission has completed, the respective TXR flag in the Transmission request 1,2,3,4 register is reset (single- shot mode), and, if bit MBI in the header section of the message buffer is set, flag SIR.TXI in the Status Interrupt Register is set to ’1’. If enabled, an interrupt is generated.

17.13.5 Null Frame Transmission

If in static segment the host does not set the transmission request flag before transmit time, and if there is no other transmit buffer with matching filter criteria, the communication controller transmits a null frame with the null frame indication bit set and the payload data set to zero.

In the following cases the communication controller transmits a null frame:

- If the message buffer with the lowest message buffer number matching the filter criteria does not have its transmission request flag set (TXR = ’0’).
- No transmit buffer configured for the slot has a cycle counter filter that matches the current cycle. In this case, no message buffer status MBS is updated.

Null frames are not transmitted in the dynamic segment.

17.14 Receive Process

17.14.1 Dedicated Receive Buffers

A portion of the E-Ray message buffers can be configured as dedicated receive buffers by programming bit CFG in the header section of the respective message buffer to 0. This can be done via the Write Header Section 1 register.

The following possibilities exist to assign a receive buffer to the Communication Controller channels:

- Static segment:channel A **or** channel B,
  channel A **and** channel B (the communication controller stores the first semantically valid frame)
- Dynamic segment: channel A **or** channel B

The communication controller transfers payload data of valid received messages from the shift registers of the FlexRay protocol controller (channel A or B) to the receive buffer with the matching filter configuration. A receive buffer is able to store all frame elements except the frame CRC.

All message buffers configured for reception in static or dynamic segment are reconfigurable during runtime depending on the configuration of MRC.SEC[1:0] of the Message RAM Configuration register. If a message buffer is reconfigured (header section updated) during runtime it may happen that in the respective communication cycle a received message is lost.

If two or more receive buffers meet the filter criteria simultaneously, the receive buffer with the lowest message buffer number is updated with the received message.

17.14.2 Frame Reception

The following steps are required to prepare a dedicated message buffer for reception:

- Configure the receive buffer in the Message RAM via WRHS1, WRHS2, and WRHS3
- Transfer the configuration from input buffer to the message RAM by writing the number of the target message buffer to the Input Buffer Command Request (IBCR) register.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal acceptance filtering process, which takes place every time the communication controller receives a message. The first matching receive buffer is updated from the received message.

If a valid payload segment was stored in the data section of a message buffer, the respective ND flag in the NDAT1,2,3,4 registers is set, and, if bit MBI in the header section of that message buffer is set, flag SIR.RXI in the Status Interrupt Register is set to ’1’. If enabled, an interrupt is generated.

In case that bit ND was already set when the Message Handler updates the message buffer, bit MBS.MLST of the respective message buffer is set and the unprocessed message data is lost.

If no frame, a null frame, or a corrupted frame is received in a slot, the data section of the message buffer configured for this slot is not updated. In this case only the flags in the respective message buffer status (MBS) is updated.

When the Message Handler changed the message buffer status MBS in the header section of a message buffer, the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set, and if bit MBI in the header section of that message buffer is set, flag SIR.MBSI in the Status Interrupt Register is set to ’1’. If enabled an interrupt is generated.

If the payload length of a received frame PLR[6:0] is longer than the value programmed by PLC[6:0] in the header section of the respective message buffer, the data field stored in the message buffer is truncated to that length.

To read a receive buffer from the message RAM via the output buffer proceed as described in section 17.16.2.2.

Note:
The ND and MBS flags are automatically cleared by the message handler when the payload data and the header of a received message have been transferred to the output buffer, respectively.

17.14.3 Null Frame Reception

The payload segment of a received null frame is **not** copied into the matching dedicated receive buffer. If a null frame has been received, only the message buffer status MBS of the matching message buffer is updated from the received null frame. All bits in header 2 and 3 of the matching message buffer remain unchanged. They are updated from received data frames only.

When the Message Handler changed the message buffer status MBS in the header section of a message buffer, the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 register is set, and if bit MBI in the header section of that message buffer is set, flag SIR.MBSI in the Status Interrupt Register is set to ’1’. If enabled, an interrupt is generated.

17.15 FIFO Function

17.15.1 Description

A group of the message buffers can be configured as a cyclic First-In-First-Out (FIFO) buffer. The group of message buffers belonging to the FIFO is contiguous in the register map starting with the message buffer referenced by MRC.FFB(7–0) and ending with the message buffer referenced by MRC.LCB(7–0) in the message RAM configuration register. Up to 128 message buffers can be assigned to the FIFO.

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case frame ID, payload length, receive cycle count, and the message buffer status MBS of the addressed FIFO message buffer are overwritten with frame ID, payload length, receive cycle count, and the status from the received frame. Bit SIR.RFNE in the status interrupt register shows that the FIFO is not empty, bit SIR.RFCL is set when the receive FIFO fill level FSR.RFFL[7:0] is equal or greater than the critical level as configured by FCL.CL[7:0], bit EIR.RFO shows that a FIFO overrun has been detected. If enabled, interrupts are generated.

If null frames are not rejected by the FIFO rejection filter, the null frames will be treated like data frames when they are stored into the FIFO.

There are two index registers associated with the FIFO. The PUT Index register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the message buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available message buffer. If the PIDX register is incremented past the highest numbered message buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) message buffer in the FIFO chain. The GET Index register (GIDX) is used to address the next message buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a message buffer belonging to the FIFO to the output buffer. The PUT Index register and the GET Index register are not accessible by the host CPU.

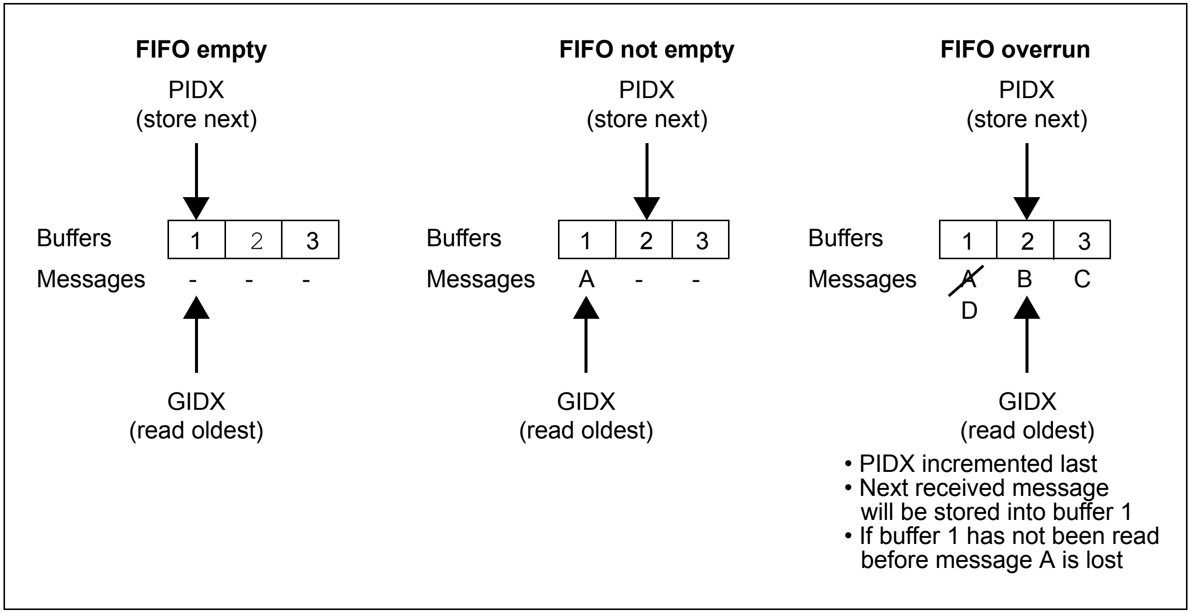
The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message overwrites the oldest message in the FIFO. This will set FIFO overrun flag EIR.RFO in the error interrupt register.

A FIFO not empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag SIR.RFNE is set. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in Figure 17-14 for a three message buffer FIFO.

The programmable FIFO Rejection Filter register (FRF) defines a filter pattern for messages to be rejected. The FIFO rejection filter consists of channel filter, frame ID filter, and cycle counter filter. If bit FRF.RSS is set to 1 (default), all messages received in the static segment are rejected by the FIFO. If bit FRF.RNF is set to 1 (default), received null frames are not stored in the FIFO.

The FIFO Rejection Filter mask register (FRFM) specifies which bits of the frame ID filter in the FIFO Rejection Filter register are marked don’t care for rejection filtering.

Figure 17-14. FIFO Status: Empty, Not Empty, and Overrun



17.15.2 Configuration of the FIFO

(Re)configuration of message buffers belonging to the FIFO is only possible when the Communication Controller is in DEFAULT_CONFIG or CONFIG state. While the Communication Controller is in DEFAULT_CONFIG or CONFIG state, the FIFO function is not available.

For all message buffers belonging to the FIFO should have the same the payload length configured to the same value by WRHS2.PLC[6:0] of the Write Header Section 2 register. The data pointer to the first 32-bit word of the data section of the respective message buffer in the Message RAM has to be configured by WRHS3.DP[10:0].

All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask. The values configured in the header sections of the message buffers belonging to the FIFO are, with exception of DP and PLC, irrelevant.

Note:
It is recommended to program the MBI bits of the message buffers belonging to the FIFO to ’0’ by WRHS1.MBI to avoid RX interrupts to be generated.

If the payload length of a received frame is longer than the value programmed by WRHS2.PLC(6–0) in the header section of the respective message buffer, the data field stored in a message buffer of the FIFO is truncated to that length.

17.15.3 Access to the FIFO

For FIFO access outside DEFAULT_CONFIG and CONFIG state, the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first message buffer of the FIFO (referenced by MRC.FFB(7–0)) to the Output Buffer Command Request (OBCR) register. The message handler then transfers the message buffer addressed by the GET Index register (GIDX) to the output buffer. After this transfer the GET Index register (GIDX) is incremented.

17.16 Message Handling

The message handler controls data transfers between the input / output buffer and the message RAM and between the message RAM and the two transient buffer RAMs. All accesses to the internal RAMs are 32+1 bit accesses. The additional bit is used for parity checking.

Access to the message buffers stored in the message RAM is done under control of the message handler state machine. This avoids conflicts between accesses of the two protocol controllers and the host CPU to the message RAM.

Frame IDs of message buffers assigned to the static segment have to be in the range from 1 to GTU7.NSS(9–0) as configured in the GTU configuration register 7. Frame IDs of message buffers assigned to the dynamic segment have to be in the range from GTU7.NSS(9–0) + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

17.16.1 Reconfiguration of Message Buffers

In case that an application needs to operate with more than 128 different messages, static and dynamic message buffers may be reconfigured during FlexRay operation. This is done by updating the header section of the respective message buffer via Input Buffer registers WRHS1,2,3.

Reconfiguration has to be enabled via control bits MRC.SEC[1:0] in the Message RAM Configuration Register.

If a message buffer has not been transmitted / updated from a received frame before reconfiguration starts, the respective message is lost.

The point in time when a reconfigured message buffer is ready for transmission / reception according to the reconfigured frame ID depends on the actual state of the slot counter when the update of the header section has completed. Therefore it may happen that a reconfigured message buffer is not transmitted / updated from a received frame in the cycle where it was reconfigured.

The Message RAM is scanned according to Table 17-8 below:

Table 17-8. Scan of Message RAM

Start of Scan in Slot	Scan for Slots
1	2...15, 1 (next cycle)
8	16...23, 1 (next cycle)
16	24...31, 1 (next cycle)
24	32...39, 1 (next cycle)
....	...

A Message RAM scan is terminated with the start of NIT regardless whether it has completed or not. The scan of the Message RAM for slots 2 to 15 starts at the beginning of slot 1 of the actual cycle. The scan of the Message RAM for slot 1 is done in the cycle before by checking in parallel to each scan of the Message RAM whether there is a message buffer configured for slot 1 of the next cycle.

The number of the first dynamic message buffer is configured by MRC.FDB[7:0] in the Message RAM Configuration register. In case a Message RAM scan starts while the Communication Controller is in dynamic segment, the scan starts with the message buffer number configured by MRC.FDB[7:0].

In case a message buffer should be reconfigured to be used in slot 1 of the next cycle, the following has to be considered:

- If the message buffer to be reconfigured for slot 1 is part of the "Static Buffers", it will only be found if it is reconfigured before the last Message RAM scan in the static segment of the actual cycle evaluates this message buffer.
- If the message buffer to be reconfigured for slot 1 is part of the "Static Buffers", it will only be found if it is reconfigured before the last Message RAM scan in the static segment of the actual cycle evaluates this message buffer.
- If the message buffer to be reconfigured for slot 1 is part of the "Static + Dynamic Buffers", it will be found if it is reconfigured before the last Message RAM scan in the actual cycle evaluates this message buffer.
- The start of NIT terminates the Message RAM scan. In case the Message RAM scan has not evaluated the reconfigured message buffer until this point in time, the message buffer will not be considered for the next cycle.

Note:
Reconfiguration of message buffers may lead to the loss of messages and therefore has to be used very carefully. In worst case (reconfiguration in consecutive cycles) it may happen that a message buffer is never transmitted / updated from a received frame.

17.16.2 Host Access to Message RAM

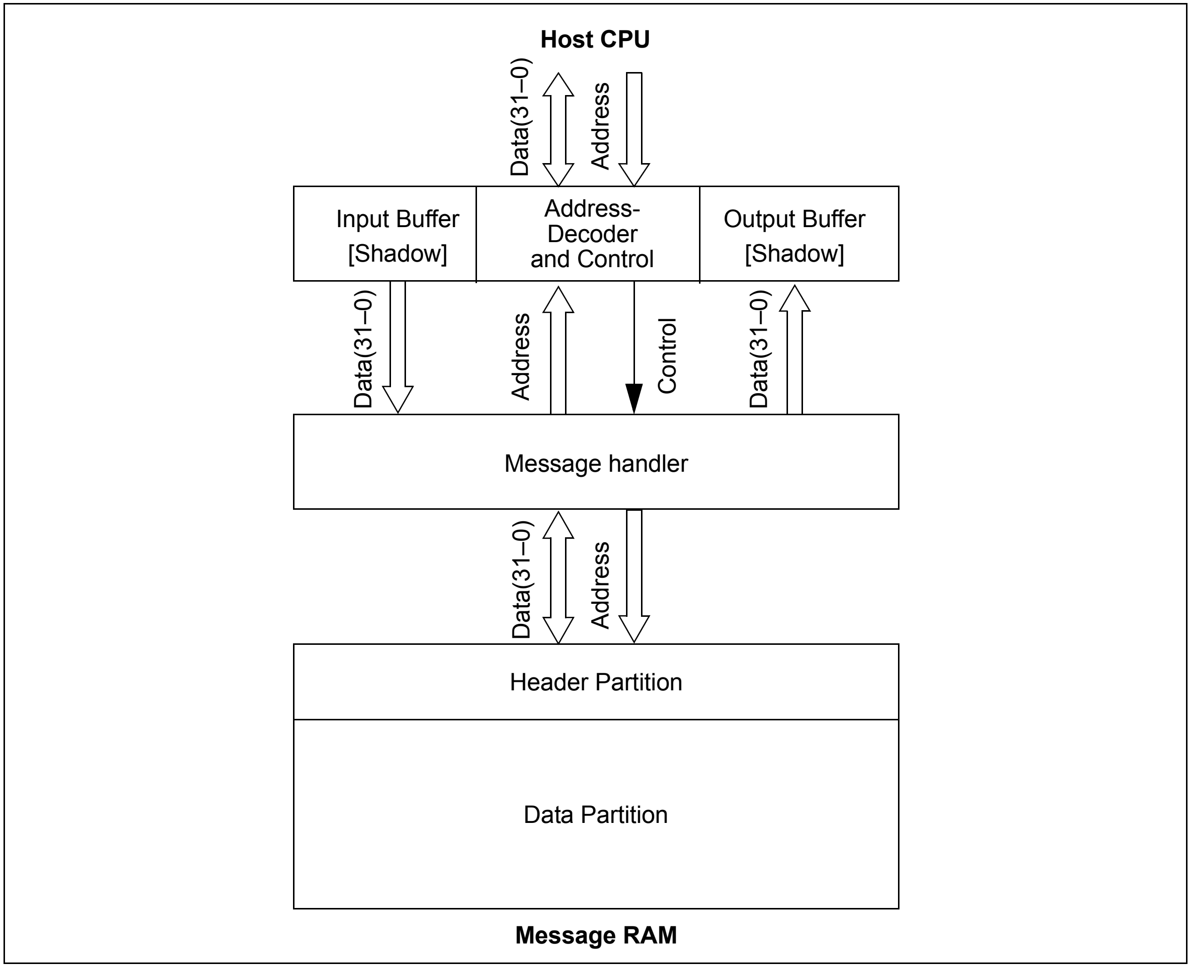
The message transfer between input buffer and message RAM as well as between message RAM and output buffer is triggered by the host CPU by writing the number of the target / source message buffer to be accessed to the input or output buffer command request register (IBCR/OBCR).

The input / output buffer command mask registers can be used to write / read header and data section of the selected message buffer separately.

If bit IBCM.STXR in the input buffer command mask register is set (STXR = 1), the transmission request flag TXR of the selected message buffer is automatically set after the message buffer has been updated.
If bit IBCM.STXR in the input buffer command mask register is reset (STXR = 0), the transmission request flag TXR of the selected message buffer is reset. This can be used to stop transmission from message buffers operated in continuous mode.

Input buffer (IBF) and the output buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the host CPU (IBF host / OBF host), while the other half (IBF shadow / OBF shadow) is accessed by the message handler for data transfers between IBF / OBF and message RAM.

Figure 17-15. Host Access to Message RAM

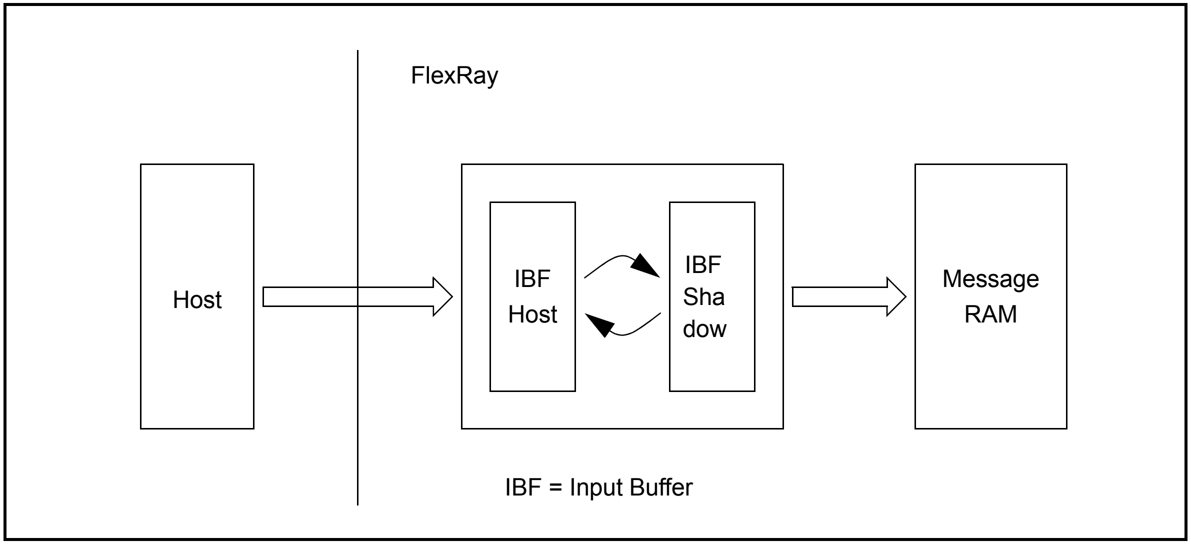


17.16.2.1 Data Transfer from Input Buffer to Message RAM

To configure / update a message buffer in the message RAM, the host has to write the data to WRDSn and the header to WRHS1…3. The specific action is selected by configuring the input buffer command mask IBCM.

When the host writes the number of the target message buffer in the message RAM to IBCR.IBRH(6–0) in the input buffer command request register IBCR, IBF host and IBF shadow are swapped (see Figure 17-16).

Figure 17-16. Double Buffer Structure Input Buffer



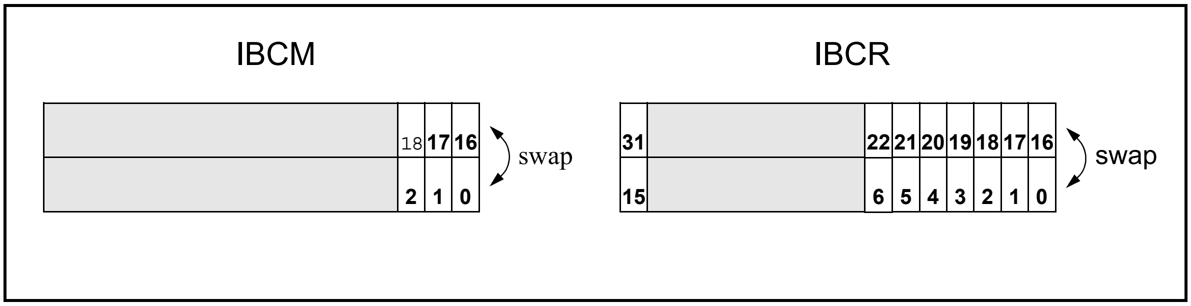


Figure 17-17. Swapping of IBCM and IBCR Bits

With this write operation the IBCR.IBSYS bit in the input buffer command request register is set to 1. The message handler then starts to transfer the contents of IBF shadow to the message buffer in the message RAM selected by IBCR.IBRS(6–0).

While the message handler transfers the data from IBF shadow to the target message buffer in the message RAM, the host may write the next message to IBF host. After the transfer between IBF shadow and the message RAM has completed, the IBCR.IBSYS bit is set back to 0 and the next transfer to the message RAM may be started by the host by writing the respective target message buffer number to IBCR.IBRH(6–0) in the input buffer command request register.

If a write access to IBCR.IBRH(6–0) occurs while IBCR.IBSYS is 1, IBCR.IBSYH is set to 1. After completion of the ongoing data transfer from IBF shadow to the message RAM, IBF host and IBF shadow are swapped, IBCR.IBSYH is reset to 0, IBCR.IBSYS remains set to 1, and the next transfer to the message RAM is started. In addition the message buffer numbers stored under IBCR.IBRH(6–0) and IBCR.IBRS(6–0) and the command mask flags are also swapped.

Example of a 8/16/32-bit host access sequence:

- Configure / update n-st message buffer via IBF
- Wait until IBCR.IBSYH is reset
- Write data section to WRDSn
- Write header section to WRHS1,2,3
- Write command mask: write IBCM.STXRH, IBCM.LHSH, IBCM.LDSH
- Demand data transfer to target message buffer: write IBCR.IBRH(6–0)

Configure / update (n+1)-nd message buffer via IBF

- Wait until IBCR.IBSYH is reset
- Write data section to WRDSn
- Write header section to WRHS1,2,3
- Write command mask: write IBCM.STXRH, IBCM.LHSH, IBCM.LDSH
- Demand data transfer to target message buffer: write IBCR.IBRH(6–0)

Configure / update further message buffer via IBF in the same way.

Note:
Any write access to IBF while IBCR.IBSYH is '1' will set error flag EIR.IIBA in the Error Interrupt Register to '1'. In this case the write access has no effect.

Table 17-9. Assignment of Input Buffer Command Mask Bits

Position	Access	Bit	Function
18	r	STXRS	Set Transmission Request shadow ongoing or finished
17	r	LDSS	Load Data Section shadow ongoing or finished
16	r	LHSS	Load Header Section shadow ongoing or finished
2	r/w	STXRH	Set Transmission Request Host
1	r/w	LDSH	Load Data Section Host
0	r/w	LHSH	Load Header Section Host

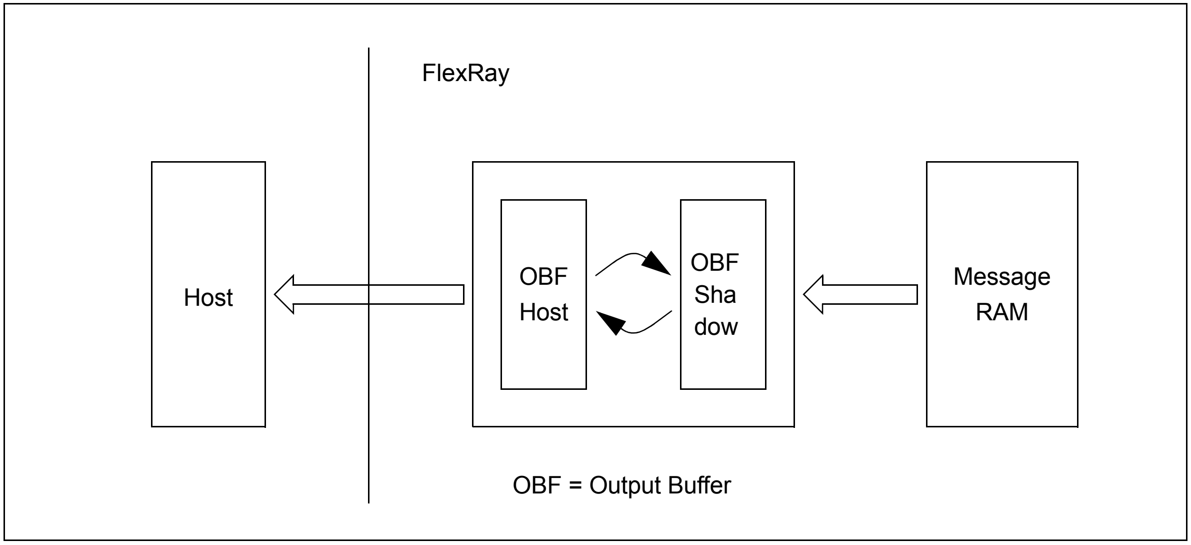
Table 17-10. Assignment of Input Buffer Command Request bits

Pos.	Access	Bit	Function
31	r	IBSYS	IBF Busy Shadow, signals ongoing transfer from IBF shadow to message RAM
22…16	r	IBRS(6–0)	IBF Request Shadow, number of message buffer currently / last updated
15	r	IBSYH	IBF Busy Host, transfer request pending for message buffer referenced by IBRH(6–0)
6…0	r/w	IBRH(6–0)	IBF Request Host, number of message buffer to be updated next

17.16.2.2 Data Transfer from Message RAM to Output Buffer

To read out a message buffer from the message RAM, the host has to write to the output buffer command mask and command request register to trigger the data transfer. After a transfer has completed the host can read the transferred data from the RDDSn, RDHS1,2,3, and MBS.

Figure 17-18. Double Buffer Structure Output Buffer

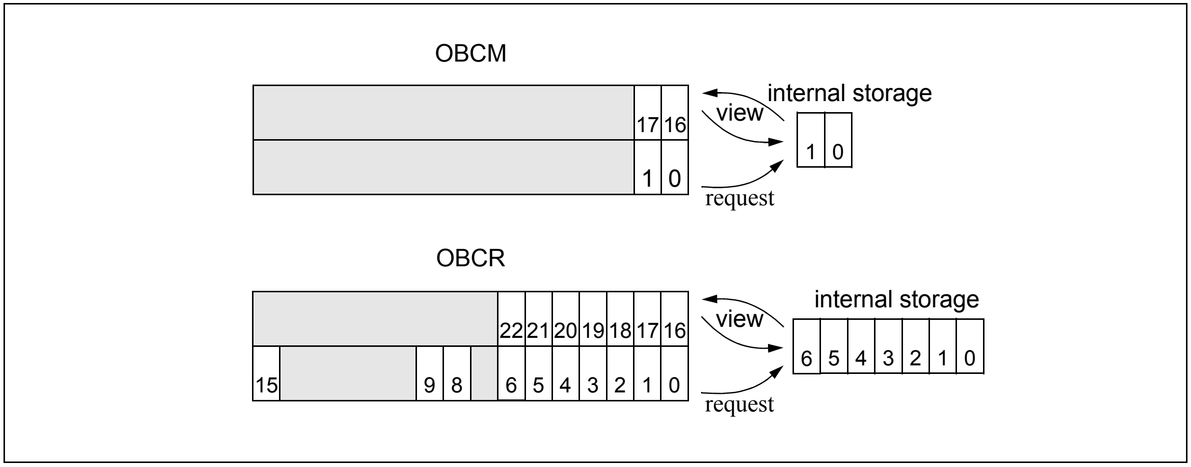


OBF host and OBF shadow as well as bits OBCM.RHSS, OBCM.RDSS, OBCM.RHSH, OBCM.RDSH from the output buffer command mask register and bits OBCM.OBRS(6–0), OBCM.OBRH(6–0) from the output buffer command request register are swapped under control of bits OBCR.VIEW and OBCR.REQ from the output buffer command request register.

Writing bit OBCR.REQ in the output buffer command request register to 1 copies bits OBCM.RHSS, OBCM.RDSS from the output buffer command mask register and bits OBCR.OBRS(6–0) from the output buffer command request register to an internal storage (see Figure 17-19).

After setting OBCR.REQ to 1, OBCR.OBSYS is set to 1, and the transfer of the message buffer selected by OBCR.OBRS(6–0) from the message RAM to OBF shadow is started. After the transfer between the message RAM and OBF shadow has completed, the OBCR.OBSYS bit is set back to 0. Bits OBCR.REQ and OBCR.VIEW can only be set to 1 while OBCR.OBSYS is 0.

Figure 17-19. Swapping of OBCM and OBCR Bits



OBF host and OBF shadow are swapped by setting bit OBCR.VIEW in the output buffer command request register to 1 while bit OBCR.OBSYS is 0 (see Figure 17-18).

In addition bits OBCR.OBRH(6–0) are swapped with the output buffer command request registers internal storage and bits OBCM.RHSH, OBCM.RDSH are swapped with the output buffer command mask registers internal storage thus assuring that the message buffer number stored in OBCR.OBRH(6–0) and the mask configuration stored in OBCM.RHSH, OBCM.RDSH matches the transferred data stored in OBF host (see Figure 17-19).

Now the host can read the transferred message buffer from OBF host while the message handler may transfer the next message from the message RAM to OBF shadow.

> **Note:**
> If bits REQ and VIEW are set to 1 with the same write access while OBSYS is 0, OBSYS is automatically set to 1 and OBF shadow and OBF host are swapped. Additionally mask bits OBCM.RDSH and OBCM.RHSH are swapped with the registers internal storage to keep them attached to the respective Output Buffer transfer. Afterwards OBRS (6-0) is copied to the register internal storage, mask bits OBCM.RDSS and OBCM.RHSS are copied to register OBCM internal storage, and the transfer of the selected message buffer from the Message RAM to OBF shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF Host. When the current transfer between Message RAM and OBF shadow has completed, this is signalled by setting OBSYS back to 0.

Example of an 8/16/32-bit host access to a single message buffer:

If a single message buffer has to be read out, two separate write accesses to OBCR.REQ and OBCR.VIEW are necessary:

- Wait until OBCR.OBSYS is reset
- Write Output Buffer Command Mask OBCM.RHSS, OBCM.RDSS
- Request transfer of message buffer to OBF Shadow by writing OBCR.OBRS(6-0) and OBCR.REQ (in case of and 8-bit Host interface, OBCR.OBRS[6:0] has to be written before OBCR.REQ).
- Wait until OBCR.OBSYS is reset
- Toggle OBF Shadow and OBF Host by writing OBCR.VIEW = ’1’
- Read out transferred message buffer by reading RDDSn, RDHS1,2,3, and MBS

Example of an 8/16/32-bit host access sequence:

Request transfer of 1st message buffer to OBF shadow

- Wait until OBCR.OBSYS is reset
- Write Output Buffer Command Mask OBCM.RHSS, OBCM.RDSS for 1st message buffer
- Request transfer of 1st message buffer to OBF Shadow by writing OBCR.OBRS(6-0) and OBCR.REQ (in case of an 8-bit Host interface, OBCR.OBRS(6-0) has to be written before OBCR.REQ).

Toggle OBF Shadow and OBF Host to read out 1st transferred message buffer and request transfer of 2nd message buffer:

Request transfer of 2nd message buffer to OBF shadow, read out 1st message buffer from OBF host

- Wait until OBCR.OBSYS is reset
- Write Output Buffer Command Mask OBCM.RHSS, OBCM.RDSS for 2nd message buffer
- Toggle OBF Shadow and OBF Host and start transfer of 2nd message buffer to OBF Shadow simultaneously by writing OBCR.OBRS(6-0) of 2nd message buffer, OBCR.REQ, and OBCR.VIEW (in case of and 8-bit Host interface, OBCR.OBRS(6-0) has to be written before OBCR.REQ and OBCR.VIEW).
- Read out 1st transferred message buffer by reading RDDSn, RDHS13, and MBS

For further transfers continue the same way.

Demand access to last requested message buffer without request of another message buffer:

- Wait until OBCR.OBSYS is reset
- Demand access to last transferred message buffer by writing OBCR.VIEW
- Read out last transferred message buffer by reading RDDSn, RDHS1,2,3, and MBS

Table 17-11. Assignment of Output Buffer Command Mask Bits

Position	Access	Bit	Function
17	r	RDSH	Read Data Section Host access
16	r	RHSH	Read Header Section Host access
1	r/w	RDSS	Read Data Section Shadow
0	r/w	RHSS	Read Header Section Shadow

Table 17-12. Assignment of Output Buffer Command Request Bits

Position	Access	Bit	Function
22...16	r	OBRH(6–0)	OBF Request Host, number of message buffer available for host access
15	r	OBSYS	OBF Busy Shadow, signals ongoing transfer from message RAM to OBF Shadow
9	r/w	REQ	Request Transfer from message RAM to OBF Shadow
8	r/w	VIEW	View OBF Shadow, swap OBF Shadow and OBF Host
6...0	r/w	OBRS(6–0)	OBF Request Shadow, number of message buffer for next request

17.16.3 FlexRay Protocol Controller Access to Message RAM

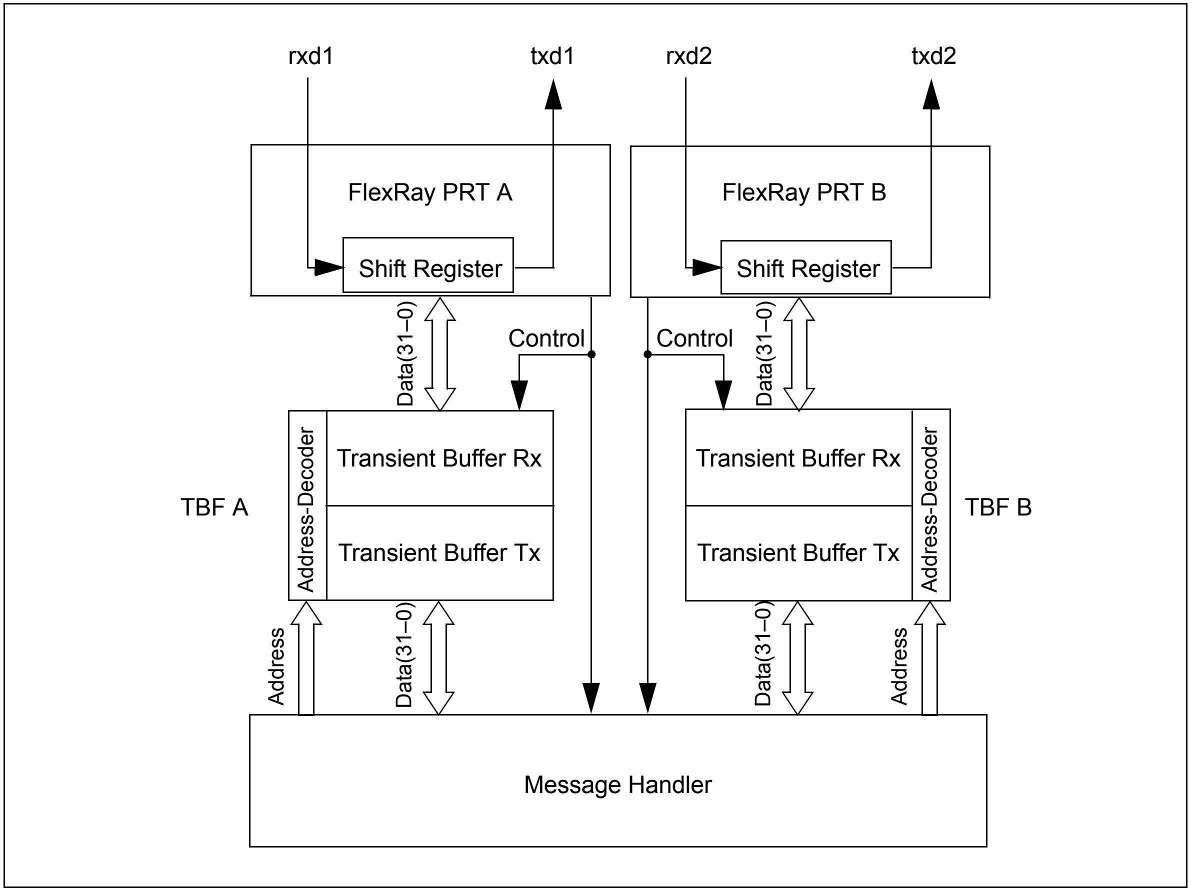
The two transient buffer RAMs (TBF A,B) are used to buffer the data for transfer between the two FlexRay channel protocol controllers and the message RAM.

Each transient buffer RAM is build up as a double buffer, able to store two complete FlexRay messages. There is always one buffer assigned to the corresponding protocol controller while the other one is accessible by the message handler.

If e.g. the message handler writes the next message to be send to transient buffer Tx, the FlexRay Channel protocol controller can access transient buffer Rx to store the message it is actually receiving. During transmission of the message stored in transient buffer Tx, the message handler transfers the last received message stored in transient buffer Rx to the message RAM (if it passes acceptance filtering) and updates the respective message buffer.

Data transfers between the transient buffer RAMs and the shift registers of the FlexRay channel protocol controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay messages.

Figure 17-20. Access to Transient Buffer RAMs



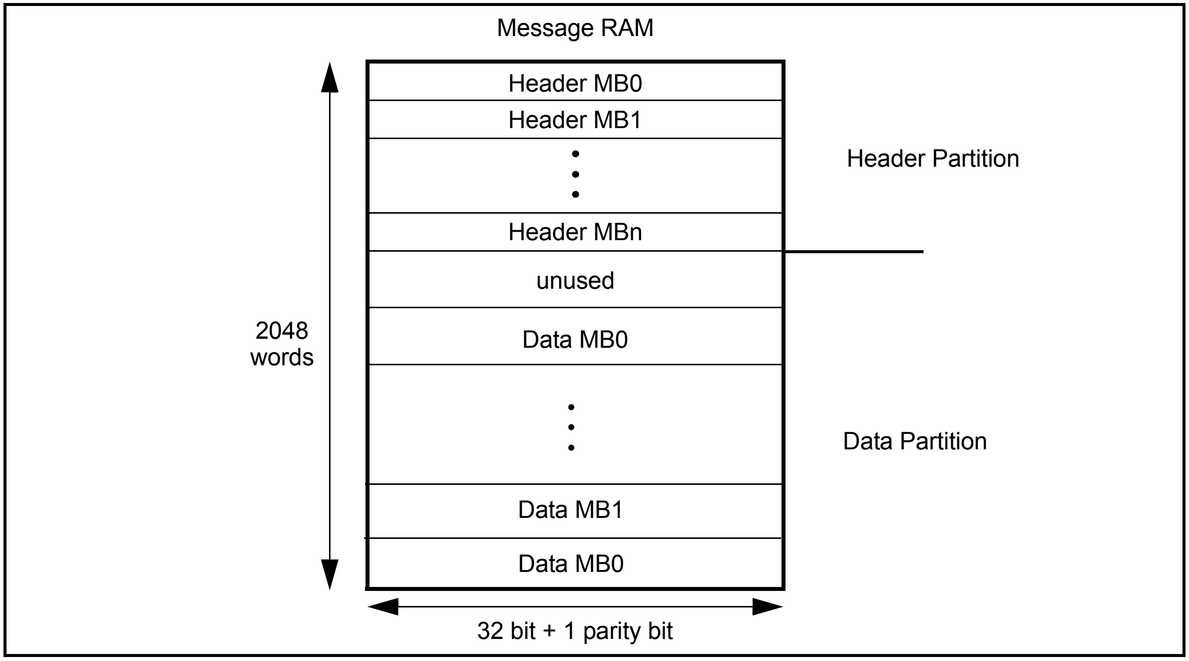
17.17 Message RAM

To avoid conflicts between host access to the message RAM and FlexRay message reception / transmission, the host CPU cannot directly access the message buffers in the message RAM. These accesses are handled via the input and output buffers. The message RAM is able to store up to 128 message buffers depending on the configured payload length.

The message RAM is organized 2048 x 33 = 67,584 bit. Each 32-bit word is protected by a parity bit. To achieve the required flexibility with respect to different numbers of data bytes per FlexRay frame (0…254), the message RAM has a structure as shown in Figure 17-21.

The data partition is allowed to start at Message RAM word number: (MRC.LCB + 1) • 4

Figure 17-21. Configuration Example of Message Buffers in the Message RAM



Header Partition

Stores header segments of FlexRay frames:

- Supports a maximum of 128 message buffers
- Each message buffer has a header of four 32+1 bit words
- Header 3 of each message buffer holds the 11-bit data pointer to the respective data section in the data partition

Data Partition

Flexible storage of data sections with different length. Some maximum values are:

- 30 message buffers with 254 byte data section each
- Or 56 message buffers with 128 byte data section each
- Or 128 message buffers with 48 byte data section each

Restriction: header partition + data partition may not occupy more than 2048 32+1 bit words.

17.17.1 Header Partition

The elements used for configuration of a message buffer as well as the actual message buffer status are stored in the header partition of the message RAM as listed in Figure 17-22 below. Configuration of the header sections of the message buffers is done via IBF (Write Header Section 1,2,3). Read access to the header sections is done via OBF (read header section 1,2,3 + message buffer status). The data pointer has to be calculated by the programmer to define the starting point of the data section for the respective message buffer in the data partition of the message RAM. The data pointer should not be modified during runtime. For message buffers belonging to the receive FIFO (re)configuration should be done in DEFAULT_CONFIG or CONFIG state only.

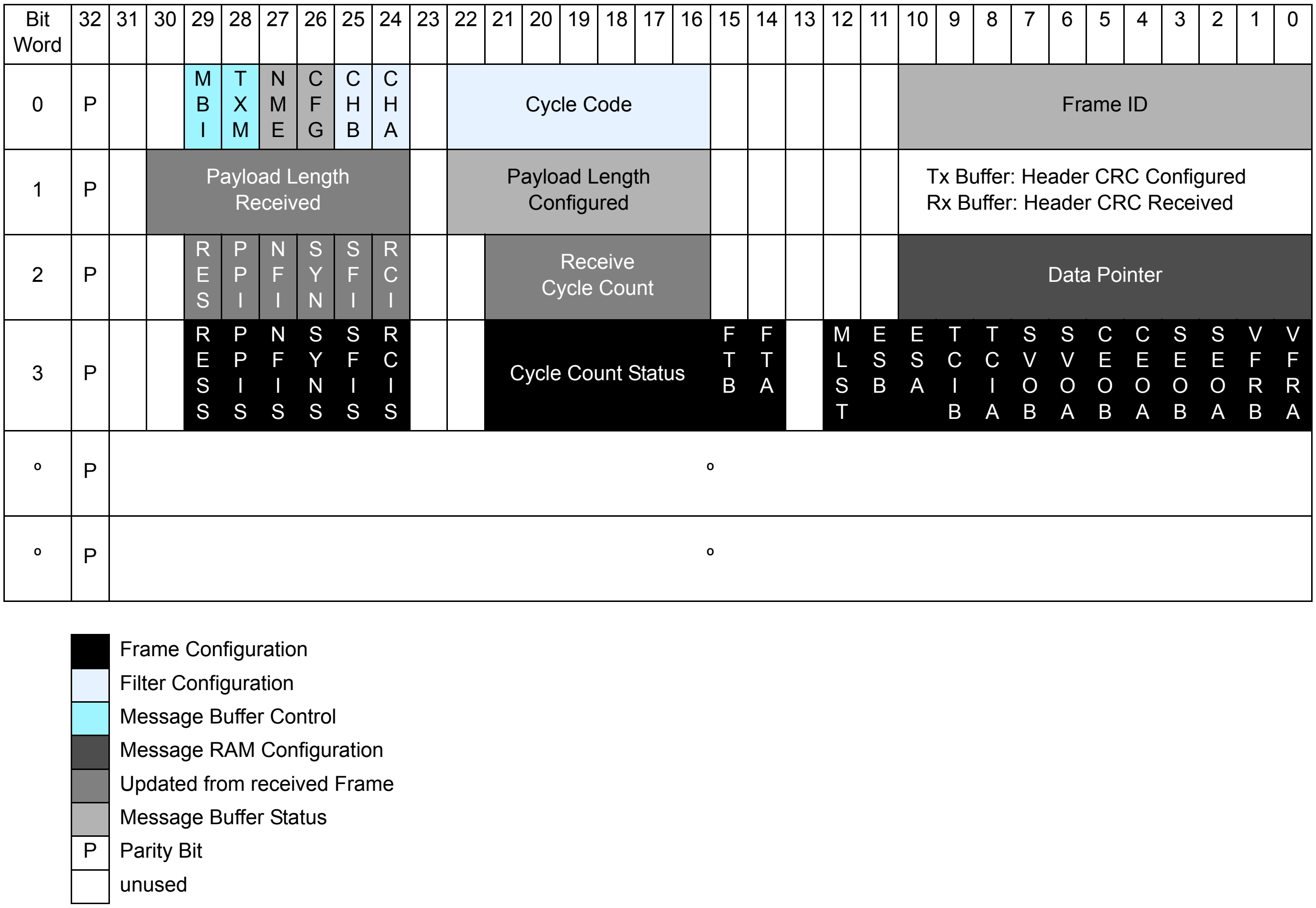
The header section of each message buffer occupies four 32+1 bit words in the header partition of the message RAM. The header of message buffer 0 starts with the first word in the message RAM.

For transmit buffers the Header CRC has to be calculated by the host CPU.

Payload length received (PLR(6-0)), receive cycle count (RCC(5-0)), received on channel indication (RCI), startup frame indication bit (SFI), sync bit (SYN), null frame indication bit (NFI), payload preamble indication bit (PPI), and reserved bit (RES) are only updated from received valid data frames only.

Header word 3 of each configured message buffer holds the respective message buffer status MBS.

Figure 17-22. Header Section of Message Buffer in Message RAM



Bit Word	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P			MBI	TXM	NME	CFG	CHB	CHA		Cycle Code												Frame ID										
1	P		Payload Length Received								Payload Length Configured												Tx Buffer: Header CRC Configured Rx Buffer: Header CRC Received										
2	P			RES	PPI	NFI	SYN	SFI	RCI			Receive Cycle Count											Data Pointer										
3	P			RESS	PPIS	NFIS	SYNS	SFIS	RCIS			Cycle Count Status						FTB	FTA		MLST	ESB	ESA	TCIB	TCIA	SVOB	SVOA	CEOB	CEOA	SEOB	SEOA	VFRB	VFRA
°	P	°																															
°	P	°																															

■ Frame Configuration
□ Filter Configuration
▨ Message Buffer Control
▩ Message RAM Configuration
▦ Updated from received Frame
▤ Message Buffer Status
P Parity Bit
□ unused

Header 1 (word 0)

Write access via WRHS1, read access via RDHS1:

- Frame ID- Slot counter filtering configuration
- Cycle Code- Cycle counter filtering configuration
- CHA, CHB- Channel filtering configuration

- CFG- Message buffer configuration: receive / transmit
- PPIT- Payload Preamble Indicator Transmit
- TXM- Transmit mode configuration: single-shot / continuous
- MBI- message buffer receive / transmit interrupt enable

Header 2 (word 1)

Write access via WRHS2, read access via RDHS2:

- Header CRC- Transmit Buffer:Configured by the host (calculated from frame header)
  - Receive Buffer:Updated from received frame
- Payload Length Configured- Length of data section (2-byte words) as configured by the host
- Payload Length Received- Length of payload segment (2-byte words) stored from received frame

Header 3 (word 2)

Write access via WRHS3, read access via RDHS3:

- Data Pointer- Pointer to the beginning of the corresponding data section in the data partition

Read access via RDHS3, valid for receive buffers only, updated from received frames:

- Receive Cycle Count - Cycle count from received frame
- RCI- Received on Channel Indicator
- SFI- Startup Frame Indicator
- SYN- Sync Frame Indicator
- NFI- Null Frame Indicator
- PPI- Payload Preamble Indicator
- RES- REServed bit

Message Buffer Status MBS (word 3)

Read access via MBS, updated by the communication controller at the end of the configured slot.

- VFRA- Valid Frame received on channel A
- VFRB- Valid Frame received on channel B
- SEOA- Syntax Error Observed on channel A
- SEOB- Syntax Error Observed on channel B
- CEOA- Content Error Observed on channel A
- CEOB- Content Error Observed on channel B
- SVOA- Slot Boundary Violation Observed on channel A
- SVOB- Slot Boundary Violation Observed on channel B
- TCIA- Transmission Conflict Indication channel A
- TCIB- Transmission Conflict Indication channel B
- ESA- Empty Slot Channel A
- ESB- Empty Slot Channel B
- MLST- Message Lost
- FTA- Frame Transmitted on Channel A
- FTB- Frame Transmitted on Channel B
- Cycle Count Status- Actual cycle count when status was updated
- RCIS- Received on Channel Indicator Status

- SFIS- Startup Frame Indication Status
- SYNS- Sync Frame Indicator Status
- NFIS- Null Frame Indicator Status
- PPIS- Payload Preamble Indicator Status
- RESS- Reserved Bit Status

17.17.2 Data Partition

The data partition of the message RAM stores the data sections of the message buffers configured for reception / transmission as defined in the header partition. The number of data bytes for each message buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay protocol controllers and the message RAM as well as between the host interface and the message RAM, the physical width of the message RAM is set to 4 bytes plus one parity bit.

The data partition starts after the last word of the header partition. When configuring the message buffers in the message RAM the programmer has to assure that the data pointers point to addresses within the data partition. Figure 17-23 below shows an example how the data sections of the configured message buffers can be stored in the data partition of the message RAM.

The beginning of a message buffers data section is determined by the data pointer and the payload length configured in the message buffers header section, respectively. This enables a flexible usage of the available RAM space for storage of message buffers with different data length.

If the size of the data section is an odd number of 2-byte words, the remaining 16 bits in the last 32-bit word are unused (see Figure 17-23).

Figure 17-23. Example Structure of Data Partition in Message RAM

Bit Word	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
°	P	unused								unused								unused								unused							
°	P	unused								unused								unused								unused							
°	P	MBn Data3								MBn Data2								MBn Data1								MBn Data0							
°	P	°								°								°								°							
°	P	°								°								°								°							
°	P	MBn Data(m)								MBn Data(m-1)								MBn Data(m-2)								MBn Data(m-3)							
°	P	°								°								°								°							
°	P	MB1 Data3								MB1 Data2								MB1 Data1								MB1 Data0							
°	P	°								°								°								°							
°	P	MB1 Data(k)								MB1 Data(k-1)								MB1 Data(k-2)								MB1 Data(k-3)							
2046	P	MB0 Data3								MB0 Data2								MB0 Data1								MB0 Data0							
2047	P	unused								unused								MB0 Data5								MB0 Data4							

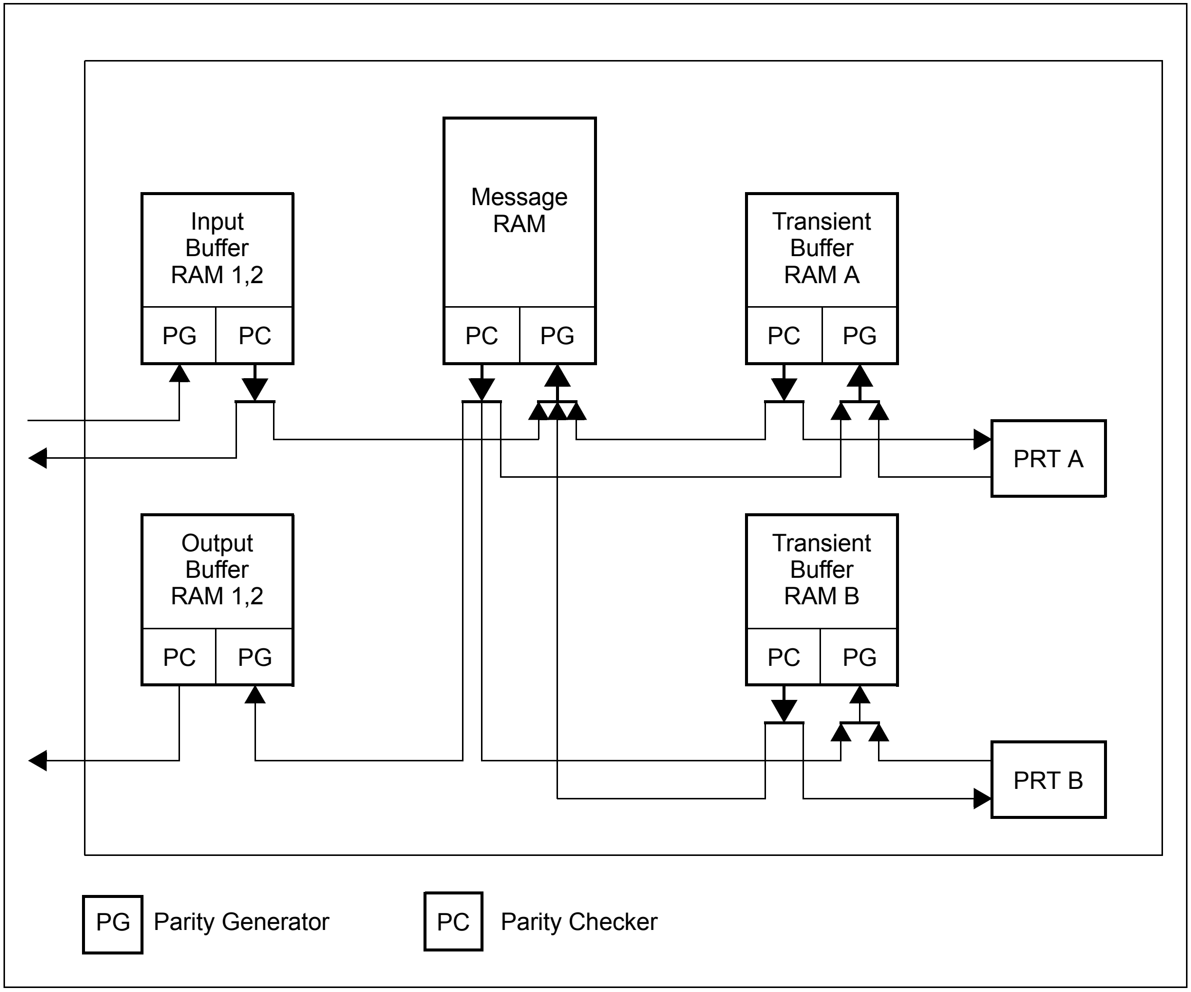
17.17.3 Parity Check

There is a parity checking mechanism implemented in the FlexRay module to assure the integrity of the data stored in the seven RAM blocks of the module. The RAM blocks have a parity generator / checker attached as shown in Figure 17-24. When data is written to a RAM block, the local parity generator generates the parity bit. The FlexRay module uses an even parity (with an even number of ones in the 32-bit data word a zero parity bit is generated). The parity bit is stored together with the respective data word. The parity is checked each time a data word is read from any of the RAM blocks. The module internal data buses have a width of 32 bits.

If a parity error is detected, the respective error flag is set in the message handler status (MHDS) register is set. These single error flags control the error interrupt flag EIR.PERR in the error interrupt register

Figure 17-24 shows the data paths between the RAM blocks and the parity generators / checkers.

Figure 17-24. Parity Generation and Check



Note:
The parity generator and checker are not part of the RAM blocks, but of the RAM access logic which is part of the FlexRay core.

When a parity error has been detected the following actions will be performed:

In all cases

- The respective parity error flag in the message handler status (MHDS) register is set
- The parity error flag EIR.PERR in the error interrupt register is set, and if enabled, a module interrupt to the CPU will be generated.

Additionally in specific cases

1. Parity error during data transfer from input buffer RAM 1,2 ⇒ message RAM
   a. Transfer of header and/or data section:
   – MHDS.PIBF bit is set
   – MHDS.FMBD bit is set to indicate that MHDS.FMB(6–0) points to a faulty message buffer
   – MHDS.FMB(6–0) indicates the number of the faulty message buffer
   – Transmit buffer: Transmission request for the respective message buffer is not set

   b. Transfer of data section only: Parity error when reading header section of respective message buffer from Message RAM.
   - MHDS.PMR bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB[6:0] points to a faulty message buffer
   - MHDS.FMB[6:0] indicates the number of the faulty message buffer
   - The data section of the respective message buffer is not updated
   - Transmit buffer: Transmission request for the respective message buffer is not set
2. Parity error during host CPU reading input buffer RAM 1,2
   - MHDS.PIBF bit is set
3. Parity error during scan of header sections in message RAM
   - MHDS.PMR bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB(6–0) points to a faulty message buffer
   - MHDS.FMB(6–0) indicates the number of the faulty message buffer
   - Ignore message buffer (message buffer is skipped)
4. Parity error during data transfer from message RAM to transient buffer RAM 1,2
   - MHDS.PMR bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB(6–0) points to the faulty message buffer
   - MHDS.FMB(6–0) indicates the number of the faulty message buffer
   - Frame not transmitted, frames already in transmission are invalidated by setting the frame CRC to zero
5. Parity error during data transfer from transient buffer RAM 1,2 to protocol controller 1, 2
   - MHDS.PTBF1,2 bit is set
   - Frames already in transmission are invalidated by setting the frame CRC to zero
6. Parity error during data transfer from transient buffer RAM 1,2 to message RAM
   a. Parity error when reading header section of respective message buffer from message RAM
   - MHDS.PMR bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB(6–0) points to a faulty message buffer
   - MHDS.FMB(6–0) indicates the number of the faulty message buffer
   - The data section of the respective message buffer is not updated

   b. Parity error when reading transient buffer RAM 1,2:
   - MHDS.PTBF1,2 bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB[6:0] points to a faulty message buffer
   - MHDS.FMB[6:0] indicates the number of the faulty message buffer
7. Parity error during data transfer from message RAM to output buffer RAM
   - MHDS.PMR bit is set
   - MHDS.FMBD bit is set to indicate that MHDS.FMB(6–0) points to faulty message buffer
   - MHDS.FMB(6–0) indicates the number of the faulty message buffer
8. Parity error during host CPU reading output buffer RAM 1,2
   - MHDS.POBF bit is set
9. Parity error during data read of transient buffer RAM 1,2
   When a parity error occurs during the Message Handler reads a frame with network management

information (PPI = 1) from the transient buffer RAM 1,2 the corresponding network management vector register NMV1,2,3 is not updated from that frame.

17.17.4 Host Handling of Parity Errors

Parity error caused by transient bit flips can be fixed by:

17.17.4.1 Self-healing

Parity errors located in

- Input Buffer RAM 1,2
- Output Buffer RAM 1,2
- Data Section of Message RAM
- Transient Buffer RAM A
- Transient Buffer RAM B

are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

17.17.4.2 CLEAR_RAMS Command

When called in DEFAULT_CONFIG or CONFIG state POC command CLEAR_RAMS initializes all module-internal RAMs to zero.

17.17.4.3 Temporary Unlocking of Header Section

A parity error in the header section of a locked message buffer can be fixed by a transfer from the input buffer to the locked buffer header section. For this transfer, the write-access to the IBCR (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see 1.24.5. Lock Register (LCK). For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by MRC.SEC[1:0], and will be updated with new data.

17.18 Transfer Unit Interrupts

Host accesses to communication controller via the IBF and the OBF (0x400-0x7FF) are forbidden, if the Transfer Unit State Machine is enabled. Accesses will be ignored and an associated error interrupt will be generated.

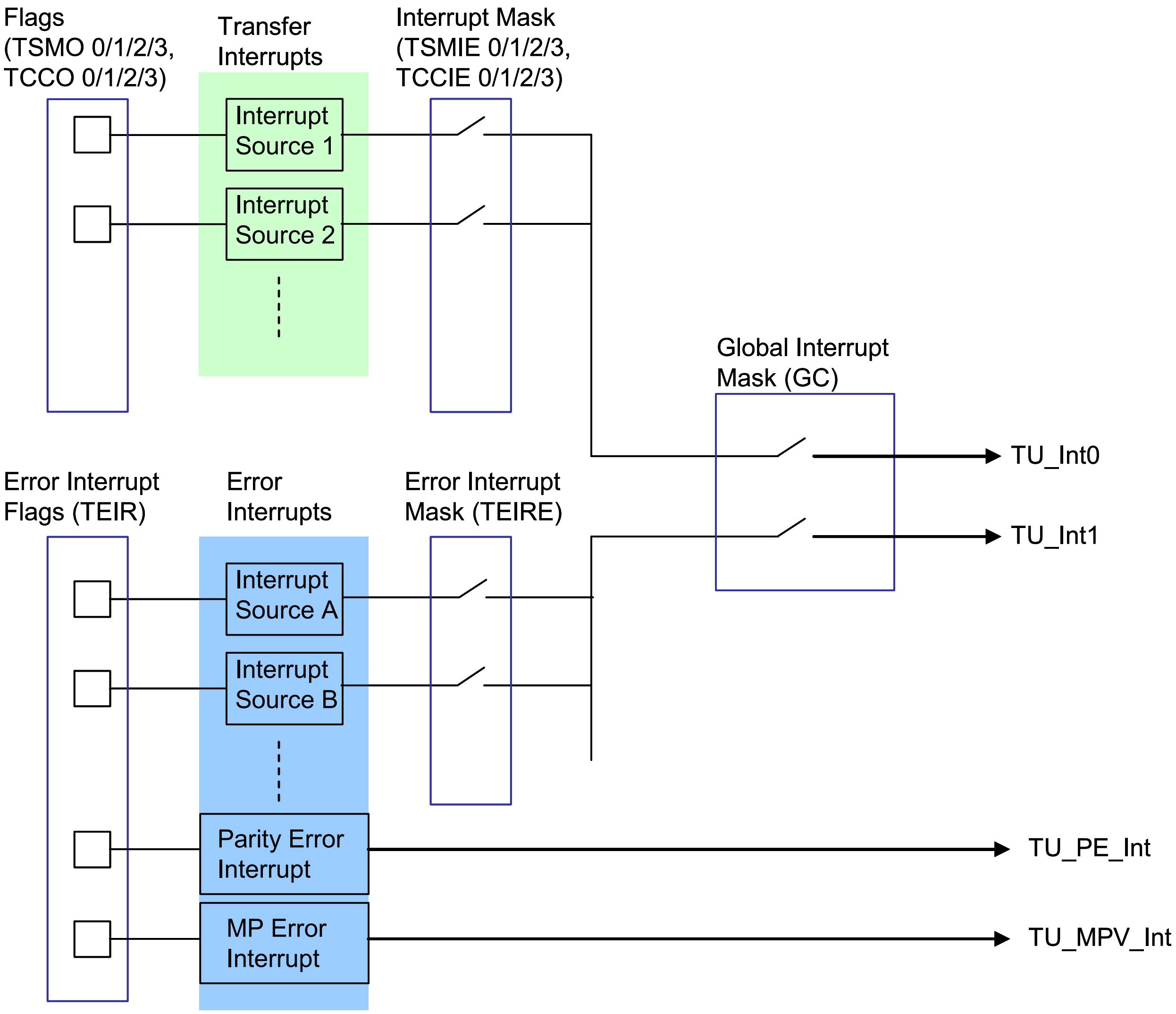
17.18.1 Interrupt Structure

One interrupt enable bit is provided for each bit in the transfer occurred status registers. Maskable error interrupts are possible for all error conditions except parity error and memory protection error.

The parity error and the memory protection error have separate non-maskable lines. Both turn off the Transfer Unit after finishing the current word access cycle.

The following Figure 17-25 shows the interrupt structure of the FlexRay Transfer Unit.

Figure 17-25. Transfer Unit Interrupt Structure



17.18.2 Interrupt Enables

TSMIE and TCCIE control the buffer transfer interrupts for every buffer in both directions. The TEIRE registers controls the maskable error interrupt sources which are:

- VBUSP master transaction errors
- Access of host to communication controller IBF or OBF during Transfer Unit State Machine enabled
- Transfer ongoing/pending during base address reload (only occurs if NTBA != TBA)

The transfer interrupts use a separate interrupt line (TU_int0).

17.18.3 Interrupt Flags

The TSMO and TCCO flags indicate buffer transfer status interrupts whereas the TEIR flags indicate interrupt sources for maskable and non-mask able error interrupts.

17.18.4 Error Interrupts

Memory protection violation interrupt and parity error interrupt have their own private lines and are non-maskable:

- If a memory protection violation occurs the Memory Protection Violation Interrupt (TU_MPV_int) line will be activated
- If a parity error occurs while accessing the TCR, the Parity Error Interrupt (TU_PE_int) line will be activated

The maskable error interrupts are individually maskable and use a separate interrupt line (TU_int1).

The error interrupt flags are set by the *Transfer Unit State Machine* and can be cleared by the CPU by writing a ‘1’. If the CPU clears the flag, while the *Transfer Unit State Machine* set it at the same time, the flag remains set.

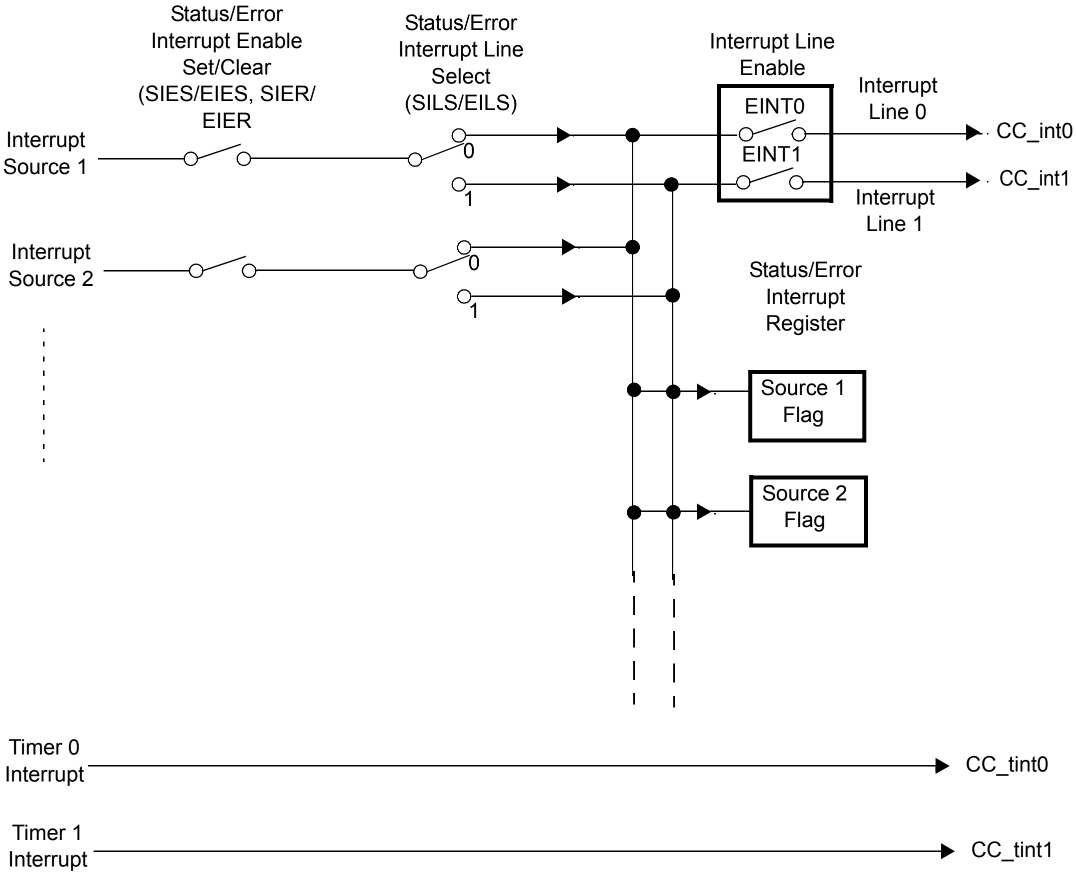
17.19 Communication Controller Interrupts

In general, interrupts provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the controller, a frame is received or transmitted, a configured timer interrupt is activated, or a stop watch event occurred. This enables the host CPU to react very quickly on specific error conditions, status changes, or timer events. On the other hand too many interrupts can cause the host to miss deadlines required for the application. Therefore the communication controller supports disable / enable controls for each individual interrupt source separately.

An interrupt may be triggered when

- An error was detected
- A status flag is set
- A timer reaches a preconfigured value
- A message transfer from input buffer to message RAM or from message RAM to output buffer has completed
- A stop watch event occurred

Figure 17-26. Interrupt Structure



Tracking status and generating interrupts when a status change or an error occurs are two independent tasks. Regardless of whether an interrupt is enabled or not, the corresponding status is tracked and indicated by the Communication Controller. The host has access to the actual status and error information by reading the error interrupt register and the status interrupt register.

Table 17-13. Module Interrupt Flags and Interrupt Line Enable

register	Bit	Function
EIR	PEMC	Protocol error Mode Changed
	CNA	Command Not Valid
	SFBM	Sync Frames Below Minimum
	SFO	Sync Frame Overflow
	CCF	Clock Correction Failure
	CCL	CHI Command Locked
	PERR	Parity error
	RFO	Receive FIFO Overrun
	EFA	Empty FIFO Access
	IIBA	Illegal Input Buffer Access
	IOBA	Illegal Output Buffer Access
	MHF	Message Handler Constraints Flag
	EDA	Error Detected on Channel A
	LTVA	Latest Transmit Violation Channel A
	TABA	Transmission Across Boundary Channel A
	EDB	Error Detected on Channel B
	LTVB	Latest Transmit Violation Channel B
	TABB	Transmission Across Boundary Channel B
SIR	WST	Wakeup Status
	CAS	Collision Avoidance Symbol
	CYCS	Cycle Start Interrupt
	TXI	Transmit Interrupt
	RXI	Receive Interrupt
	RFNE	Receive FIFO not Empty
	RFCL	Receive FIFO Critical Level
	NMVC	Network Management Vector Changed
	TI0	Timer Interrupt 0
	TI1	Timer Interrupt 1
	TIBC	Transfer Input Buffer Completed
	TOBC	Transfer Output Buffer Completed
	SWE	Stop Watch Event
	SUCS	Startup Completed Successfully
	MBSI	Message Buffer Status Interrupt
	SDS	Start of Dynamic Segment
	WUPA	Wakeup Pattern Channel A
	MTSA	MTS Received on Channel A
	WUPB	Wakeup Pattern Channel B
	MTSB	MTS Received on Channel B
ILE	EINT0	Enable Interrupt Line 0
	EINT1	Enable Interrupt Line 1

The interrupt lines to the host, CC_int0 and CC_int1, are controlled by the enabled interrupts. In addition each of the two interrupt lines to the host CPU can be enabled / disabled separately by programming bit EINT0 and EINT1 in the interrupt Line Enable register.

The two timer interrupts generated by interrupt timer 0 and 1 are available on pins CC_tint0 and CC_tint1. They can be configured via the timer 0 and timer 1 configuration register.

Note:
The timer interrupts might be lost if an interrupt service routine is in execution while a FlexRay timer interrupt occurs. **Root cause**: When the FlexRay timer interrupt occurs, the interrupt signal feeding to VIM is only valid for one macro tick. Since VIM is level triggered, if VIM is busy in this one macro tick, the interrupt would be lost. **Work around**: The timer interrupts are available also as status flags which can trigger status interrupts. Status interrupts can be used to avoid this problem. In case more status interrupts are used (up to 19 status interrupt sources are possible), the software must handle the identification of the interrupt source.

When a transfer between IBF / OBF and the Message RAM has completed bit SIR.TIBC or SIR.TOBC is set.

17.20 Register Map

All registers are organized as 32-bit registers. 32/16/8-bit accesses are supported. For FlexRayTU transfers only 4x32-bit data packages are supported.

The Transfer Unit State Machine registers use the offset address range 0x000 to 0x1FC.

Transfer Configuration RAM uses the offset address range 0x000 to 0x01FC in normal mode and 0x000 to 0x3FC in parity test mode.

An address range overview of the FlexRayTU relevant registers and memory gives Table 17-14:

Table 17-14. Transfer Unit Register Set

Offset_TU	Symbol	Name	Reset	Acc
Transfer Unit Register				
0x000	GSN0	Global Static Number 0	5432 ABCD	r
0x004	GSN1	Global Static Number 1	ABCD 5432	r
0x008 - 0x00C		reserved (2)	0000 0000	r
0x010	GCS	Global Control Set	0005 0000	r/w
0x014	GCR	Global Control Reset	0005 0000	r/w
0x018	TSCB	Transfer Status	0000 0100[a]	r
0x01C	LTBCC	Last Transferred Buffer to Communication Controller	0000 0000	r
0x020	LTBSM	Last Transferred Buffer to System Memory	0000 0000	r
0x024	TBA	Transfer Base Address	0000 0000	r/w
0x028	NTBA	Next Transfer Base Address	0000 0000	r/w
0x02C	BAMS	Base Address of Mirrored Status	0000 0000	r/w
0x030	SAMP	Start Address of Memory Protection	0000 0000	r/w
0x034	EAMP	End Address of Memory Protection	0000 0000	r/w
0x038 - 0x03C		reserved (2)	0000 0000	r
0x040	TSMO1	Transfer to System Memory Occurred (Message Buffer No. 0-31)	0000 0000	r/w
0x044	TSMO2	Transfer to System Memory Occurred (Message Buffer 32-63)	0000 0000	r/w
0x048	TSMO3	Transfer to System Memory Occurred (Message Buffer 64-95)	0000 0000	r/w
0x04C	TSMO4	Transfer to System Memory Occurred (Message Buffer 96-127)	0000 0000	r/w
0x050	TCCO1	Transfer to Communication Controller Occurred (Message Buffer No. 0-31)	0000 0000	r/w
0x054	TCCO2	Transfer to Communication Controller Occurred (Message Buffer No. 32-63)	0000 0000	r/w
0x058	TCCO3	Transfer to Communication Controller Occurred (Message Buffer No. 64-95)	0000 0000	r/w

Table 17-14. Transfer Unit Register Set (Continued)

Offset_TU	Symbol	Name	Reset	Acc
0x05C	TCCO4	Transfer to Communication Controller Occurred (Message Buffer No. 96-127)	0000 0000	r/w
0x060	TOOFF	Transfer Occurred Offset	0000 0000	r
0x064 - 0x06C		reserved (3)	0000 0000	r
0x070	PEADR	Parity Error Address	0000 0xuu[b]	r
0x074	TEIR	Transfer Error InterRupt	0000 0000	r/w
0x078	TEIRES	Transfer Error InterRupt Enable Set	0000 0000	r/w
0x07C	TEIRER	Transfer Error InterRupt Enable Reset	0000 0000	r/w
0x080	TTSMS1	Trigger Transfer to System Memory Set (Message Buffer No. 0-31)	0000 0000	r/w
0x084	TTSMR1	Trigger Transfer to System Memory Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x088	TTSMS2	Trigger Transfer to System Memory Set (Message Buffer No. 32-63)	0000 0000	r/w
0x08C	TTSMR2	Trigger Transfer to System Memory Reset (Message Buffer No. 32-63)	0000 0000	r/w
0x090	TTSMS3	Trigger Transfer to System Memory Set (Message Buffer No. 64-95)	0000 0000	r/w
0x094	TTSMR3	Trigger Transfer to System Memory Reset (Message Buffer No. 64-95)	0000 0000	r/w
0x098	TTSMS4	Trigger Transfer to System Memory Set (Message Buffer No. 96-127)	0000 0000	r/w
0x09C	TTSMR4	Trigger Transfer to System Memory Reset (Message Buffer No. 96-127)	0000 0000	r/w
0x0A0	TTCCS1	Trigger Transfer to Communication Controller Set (Message Buffer No. 0-31)	0000 0000	r/w
0x0A4	TTCCR1	Trigger Transfer to Communication Controller Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x0A8	TTCCS2	Trigger Transfer to Communication Controller Set (Message Buffer 32-63)	0000 0000	r/w
0x0AC	TTCCR2	Trigger Transfer to Communication Controller Reset (Message Buffer 32-63)	0000 0000	r/w
0x0B0	TTCCS3	Trigger Transfer to Communication Controller Set (Message Buffer 64-95)	0000 0000	r/w
0x0B4	TTCCR3	Trigger Transfer to Communication Controller Reset (Message Buffer 64-95)	0000 0000	r/w

Table 17-14. Transfer Unit Register Set (Continued)

Offset_TU	Symbol	Name	Reset	Acc
0x0B8	TTCCS4	Trigger Transfer to Communication Controller Set (Message Buffer 96-127)	0000 0000	r/w
0x0BC	TTCCR4	Trigger Transfer to Communication Controller Reset (Message Buffer 96-127)	0000 0000	r/w
0x0C0	ETESMS1	Enable Transfer on Event to System Memory Set (Message Buffer No. 0-31)	0000 0000	r/w
0x0C4	ETESMR1	Enable Transfer on Event to System Memory Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x0C8	ETESMS2	Enable Transfer on Event to System Memory Set (Message Buffer No. 32-63)	0000 0000	r/w
0x0CC	ETESMR2	Enable Transfer on Event to System Memory Reset (Message Buffer No. 32-63)	0000 0000	r/w
0x0D0	ETESMS3	Enable Transfer on Event to System Memory Set (Message Buffer No. 64-95)	0000 0000	r/w
0x0D4	ETESMR3	Enable Transfer on Event to System Memory Reset (Message Buffer No. 64-95)	0000 0000	r/w
0x0D8	ETESMS4	Enable Transfer on Event to System Memory Set (Message Buffer No. 96-127)	0000 0000	r/w
0x0DC	ETESMR4	Enable Transfer on Event to System Memory Reset (Message Buffer No. 96-127)	0000 0000	r/w
0x0E0	CESMS1	Clear on Event to System Memory Set (Message Buffer No. 0-31)	0000 0000	r/w
0x0E4	CESMR1	Clear on Event to System Memory Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x0E8	CESMS2	Clear on Event to System Memory Set (Message Buffer No. 32-63)	0000 0000	r/w
0x0EC	CESMR2	Clear on Event to System Memory Reset (Message Buffer No. 32-63)	0000 0000	r/w
0x0F0	CESMS3	Clear on Event to System Memory Set (Message Buffer No. 64-95)	0000 0000	r/w
0x0F4	CESMR3	Clear on Event to System Memory Reset (Message Buffer No. 64-95)	0000 0000	r/w
0x0F8	CESMS4	Clear on Event to System Memory Set (Message Buffer No. 96-127)	0000 0000	r/w
0x0FC	CESMR4	Clear on Event to System Memory Reset (Message Buffer No. 96-127)	0000 0000	r/w

Table 17-14. Transfer Unit Register Set (Continued)

Offset_TU	Symbol	Name	Reset	Acc
0x100	TSMIES1	Transfer to System Memory Interrupt Enable Set (Message Buffer No. 0-31)	0000 0000	r/w
0x104	TSMIER1	Transfer to System Memory Interrupt Enable Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x108	TSMIES2	Transfer to System Memory Interrupt Enable Set (Message Buffer No. 32-63)	0000 0000	r/w
0x10C	TSMIER2	Transfer to System Memory Interrupt Enable Reset (Message Buffer No. 32-63)	0000 0000	r/w
0x110	TSMIES3	Transfer to System Memory Interrupt Enable Set (Message Buffer No. 64-95)	0000 0000	r/w
0x114	TSMIER3	Transfer to System Memory Interrupt Enable Reset (Message Buffer No. 64-95)	0000 0000	r/w
0x118	TSMIES4	Transfer to System Memory Interrupt Enable Set (Message Buffer No. 96-127)	0000 0000	r/w
0x11C	TSMIER4	Transfer to System Memory Interrupt Enable Reset (Message Buffer No. 96-127)	0000 0000	r/w
0x120	TCCIES1	Transfer to Communication Controller Interrupt Enable Set (Message Buffer No. 0-31)	0000 0000	r/w
0x124	TCCIER1	Transfer to Communication Controller Interrupt Enable Reset (Message Buffer No. 0-31)	0000 0000	r/w
0x128	TCCIES2	Transfer to Communication Controller Interrupt Enable Set (Message Buffer No. 32-63)	0000 0000	r/w
0x12C	TCCIER2	Transfer to Communication Controller Interrupt Enable Reset (Message Buffer No. 32-63)	0000 0000	r/w
0x130	TCCIES3	Transfer to Communication Controller Interrupt Enable Set (Message Buffer No. 64-95)	0000 0000	r/w
0x134	TCCIER3	Transfer to Communication Controller Interrupt Enable Reset (Message Buffer No. 64-95)	0000 0000	r/w
0x138	TCCIES4	Transfer to Communication Controller Interrupt Enable Set (Message Buffer No. 96-127)	0000 0000	r/w

Table 17-14. Transfer Unit Register Set (Continued)

Offset_TU	Symbol	Name	Reset	Acc
0x13C	TCCIER4	Transfer to Communication Controller Interrupt Enable Reset (Message Buffer No. 96-127)	0000 0000	r/w
0x140- 0x1FC		reserved (48)	0000 0000	r

a. 00010100 in Debug Mode
b. x=“000u“; u = undefined

Table 17-15. Transfer Unit RAM

Offset_TU_ RAM	Symbol	Name	Reset	Acc
Transfer Configuration RAM				
0x000- 0x1FC	TCRx	Transfer Configuration Ram 128 x 22 (Message Buffer No. 0-127) in normal mode	undefined	r/w
0x200- 0x3FC	TCRPx	Transfer Configuration Ram 128 x 22 (Message Buffer No. 0-127) in parity test mode	undefined	r/w

17.21 Transfer Unit Registers

17.21.1 Global Static Number 0 (GSN0)

This register contains a constant to check correctness of data transfers.

Figure 17-27. *Global Static Number 0 (GSN0)* [offset_TU = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
R-0	R-1	R-0	R-1	R-0	R-1	R-0	R-0	R-0	R-0	R-1	R-1	R-0	R-0	R-1	R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R-1	R-0	R-1	R-0	R-1	R-0	R-1	R-1	R-1	R-1	R-0	R-0	R-1	R-1	R-0	R-1

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-16. Global Static Number 0 (GSN0) Field Descriptions

Bit	Name	Value	Description
31–16	Dx	0x5432	
15-0	Dx	0xABCD	(complement of D[31:16])

17.21.2 Global Static Number 1 (GSN1)

This register contains a constant to check correctness of data transfers.

Figure 17-28. *Global Static Number 1 (GSN1)* [offset_TU = 0x04]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
R-1	R-0	R-1	R-0	R-1	R-0	R-1	R-1	R-1	R-1	R-0	R-0	R-1	R-1	R-0	R-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R-0	R-1	R-0	R-1	R-0	R-1	R-0	R-0	R-0	R-0	R-1	R-1	R-0	R-0	R-1	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-17. Global Static Number 1 (GSN1) Field Descriptions

Bit	Name	Value	Description
31–16	D[31:16]	0xABCD	
15-0	D[15:0]	0x5432	(complement of D[31:16])

17.21.3 Global Control Set/Reset (GCS/R)

The GC Register reflects the configuration mode and allows to configure the basic Transfer Unit behavior.

For safety reasons the GC register has 2 addresses. The bits are set by writing ‘1’ to GCS and reset by writing ‘1’ to GCR, writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-29. Global Control Set/Reset (GCS) [offset_TU = 0x10]

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x010	END-VBM	END-VBS	ENDR 1	ENDR 0	ENDH 1	ENDH 0	ENDP 1	ENDP 0	reserved		PRIO	PFT	PAL3	PAL2	PAL1	PAL0
	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	R-0		RS-0	RS-0	RS-0	RS-1	RS-0	RS-1
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	reserved	CET-ESM	CTTCC	CTTSM	reserved			ETSM	reserved		SILE	EILE	reserved		TUH	TUE
	R-0	RS-0	RS-0	RS-0	R-0			RS-0	R-0		RS-0	RS-0	R-0		RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-18. Global Control Set/Reset (GCS)

Bit	Name	Value	Description
31	ENDVBM		Endianess Correction on VBusp Master
		0	Endianess correction switched off (Endianess is default: Little Endianess equal to Big Endian word invariant (ARM:BE-32), same as all other peripherals) (Example 32 Bit Word = ABCD)
		1	Endianess correction switched on (E-Ray Register, Header and Payload Endianess is according the configuration of bits ENDR0/1 ENDH0/1, ENDP0/1)
30	ENDVBS		ENDianess correction on VBusp Slave
		0	Endianess correction switched off (Endianess is default: Little Endianess equal to Big Endian word invariant (ARM:BE-32), same as all other peripherals) Example 32 Bit Word = ABCD
		1	Endianess correction switched on (E-Ray Register, Header and Payload Endianess is according the configuration of bits ENDR0/1, ENDH0/1, ENDP0/1)
29-28	ENDRx		ENDianess Correction for No (header or payload) Data Sink Access Byte-order control of CPU access to E-Ray register, Transfer Unit register and Transfer Unit ram data. Data transferred between CPU and data sink will be corrected.
		00	Remapped to 0xABCD
		01	Remapped to 0xBADC
		10	Remapped to 0xCDAB
		11	Remapped to 0xDCBA

Table 17-18. Global Control Set/Reset (GCS) (Continued)

Bit	Name	Value	Description
27-26	ENDHx		ENDianess Correction for Header
		00	Remapped to 0xABCD
		01	Remapped to 0xBADC
		10	Remapped to 0xCDAB
		11	Remapped to 0xDCBA
25-24	ENDPx		ENDianess Correction for Payload
		00	Remapped to 0xABCD
		01	Remapped to 0xBADC
		10	Remapped to 0xCDAB
		11	Remapped to 0xDCBA
23-22	Reserved		Reads return zero and writes have no effect.
21	PRIO		Transfer Priority
		0	TTSM gets higher priority than TTCC
		1	TTCC gets higher priority than TTSM
20	PFT		Parity for Test
		0	Do not use parity test feature. TCRP not readable and writable.
		1	Use test feature for testing parity mechanism. TCRP is readable and writable.
19-16	PALx		Parity Lock
		0101	Parity protection for TCR is switched off.
		others	Parity protection for TCR is switched on.
15	Reserved		Reads return zero and writes have no effect.
14	CETESM		Clear ETESM Register Clear all bits of Enable Transfer on Event to System Memory register.
		0	Do not clear the register.
		1	Clear the register.
13	CTTCC		Clear TTCC Register
		0	Do not clear the register.
		1	Clear the register.
12	CTTSM		Clear TTSM Register
		0	Do not clear the register.
		1	Clear the register.

Table 17-18. Global Control Set/Reset (GCS) (Continued)

Bit	Name	Value	Description
11-9	Reserved		Reads return zero and writes have no effect.
8	ETSM		Enable Transfer Status Mirrored Mirror technic must be adjustable.
		0	Disable mirror function for TSCB, LTBCC, LTBSM, TSMO1-4, TCCO1-4 and TOOFF.
		1	Enable mirror function for TSCB, LTBCC, LTBSM, TSMO1-4, TCCO1-4 and TOOFF.
7-6	Reserved		Reads return zero and writes have no effect.
5	SILE		Status Interrupt Line Enable Enable/Disable status line interrupt.
		0	TU_int0 is disabled.
		1	TU_int0 is enabled.
4	EILE		Error Interrupt Line Enable Enable/Disable error interrupt line.
		0	TU_int1 is disabled.
		1	TU_int1 is enabled.
3-2	Reserved		Reads return zero and writes have no effect.
1	TUH		Transfer Unit Halted When halted, the Transfer Unit State Machine finishes the ongoing VBUSM access before it stops working. After deassertion, the Transfer Unit State Machine continues at the point it was halted before. No reconfiguration is required.
		0	Transfer Unit not halted
		1	Transfer Unit halted Note: If the Transfer Unit State Machine halts, all mirroring registers contained the last transfer not the current transfer information.
0	TUE		Transfer Unit Enabled Enable/Disable transfer unit.
		0	Transfer Unit disabled, reset Transfer Unit State Machine, completion of the current VBUS transfer cycle but data could be corrupt.
		1	Transfer Unit enabled Note: Before switching on the Transfer Unit, the registers must be set up. After reenabling of the Transfer Unit State Machine the contents of the module registers and the TCR is still valid (assuming it was continuously powered).

Figure 17-30. Global Control Set/Reset (GCR) [offset_TU = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
END-VBM	END-VBS	ENDR1	ENDR0	ENDH1	ENDH0	ENDP1	ENDP0	reserved		PRIO	PFT	PAL3	PAL2	PAL1	PAL0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	R-0		RS-0	RS-0	RS-0	RS-1	RS-0	RS-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	CET-ESM	CTTCC	CTTSM	reserved			ETSM	reserved		SILE	EILE	reserved		TUH	TUE
R-0	RS-0	RS-0	RS-0	R-0			RS-0	R-0		RS-0	RS-0	R-0		RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Note:
For bit description see GCS register.

17.21.4 Transfer Status Current Buffer (TSCB)

The Transfer Status Current Buffer displays the current buffer in progress and indicates if the Transfer Unit State Machine is idle and is halt.

Figure 17-31. Transfer Status Current Buffer (TSCB) [offset_TU = 0x18]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											TSMS4	TSMS3	TSMS2	TSMS1	TSMS0
R-0											R-0	R-0	R-0	R-0	R-0[a]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			STUH	Reserved			IDLE	Reserved	BN6	BN5	BN4	BN3	BN2	BN1	BN0
R-0			R-0	R-0			RC-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

a. This bit is read as 1 after reset when in Debug Mode

Table 17-19. Transfer Status Current Buffer (TSCB) Field Descriptions

Bit	Name	Value	Description
31–21	Reserved		Reads return zero and writes have no effect.
20-16	TSMS[4:0]		Transfer State Machine Status Reflects the current status of the transfer state machine for debug purpose (only available in debug mode in combination with a debugger). In Normal Operation Mode the value of TSMS[4-0] is always read as 0. Transfer Trigger to System Memory:
		00001	Start state (TTSM_START)
		00010	Output Buffer Command Mask access state (TTSM_OBCM)
		00011	Request state (TTSM_REQ)
		00100	View state (TTSM_VIEW)
		00101	Check state (TTSM_CHECK)
		00110	Read Header Section access state (TTSM_RDHS)
		00111	Read Data Section access state (TTSM_RDDS)
			Transfer Trigger to Communication Controller:
		01000	Start state (TTCC_START)
		01001	Busy state (TTCC_IBUSY)
		01010	Check state (TTCC_CHECK)
		01011	Write Header Section access state (TTCC_WRHS)
		01100	Payload Read state (TTCC_PLC_READ)
		01101	Payload Calculation state (TTCC_PLC_CALC)

Table 17-19. Transfer Status Current Buffer (TSCB) Field Descriptions (Continued)

Bit	Name	Value	Description
		01110	Write Data Section access state (TTCC_WRDS)
		01111	Input Buffer Command Mask access state (TTCC_IBCM)
		10000	Input Buffer Command Request access state (TTCC_IBCR)
		10001	Mirror state (TTCC_MIRROR)
		10010	End state (TTSM_END)
		11111	Undefined state
15-13	Reserved		Reads return zero and writes have no effect.
12	STUH		Status of Transfer Unit State Machine for Halt Detection
		0	TUSM is not in HALT status.
		1	TUSM is in HALT status.
11-9	Reserved		Reads return zero and writes have no effect.
8	IDLE		Detects Transfer State Machine State IDLE Will be set if the transfer unit state machine is in IDLE state and ready to start the next transfer, but nothing is requested.
		0	Detection of no TUSM IDLE to IDLE transition after last clear.
		1	Detection of a TUSM IDLE to IDLE transition.
7	Reserved		Reads return zero and writes have no effect.
6-0	BN[6:0]		Buffer Number 7-bit value for buffer number, updated after transfer state machine leaves state IDLE.

17.21.5 Last Transferred Buffer to Communication Controller (LTBCC)

Shows the number of the last completely transferred message buffer from system memory to the communication controller.

Figure 17-32. Last Transferred Buffer to Communication Controller (LTBCC) [offset_TU = 0x1C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									BN6	BN5	BN4	BN3	BN2	BN1	BN0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -n = Value after reset

Table 17-20. Last Transferred Buffer to Communication Controller (LTBCC) Field Descriptions

Bit	Name	Value	Description
31–7	Reserved		Reads return zero and writes have no effect.
6-0	BN[6:0]		Buffer Number 7-bit value for buffer number.

17.21.6 Last Transferred Buffer to System Memory (LTBSM)

Shows the number of the last completely transferred message buffer from communication controller to the system memory.

Figure 17-33. Last Transferred Buffer to System Memory (LTBSM) [offset_TU = 0x20]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									BN6	BN5	BN4	BN3	BN2	BN1	BN0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -n = Value after reset

Table 17-21. Last Transferred Buffer to System Memory (LTBSM) Field Descriptions

Bit	Name	Value	Description
31–7	Reserved		Reads return zero and writes have no effect.
6-0	BN[6:0]		Buffer Number 7-bit value for buffer number.

17.21.7 Transfer Base Address (TBA)

The Transfer Base Address register holds a 32bit aligned 32bit base-pointer, which defines the base address for the data to be transferred.

Note:
A write to this register modifies also the NTBA register.

Figure 17-34. Transfer Base Address (TBA) [offset_TU = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBA31	TBA30	TBA29	TBA28	TBA27	TBA26	TBA25	TBA24	TBA23	TBA22	TBA21	TBA20	TBA19	TBA18	TBA17	TBA16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBA15	TBA14	TBA13	TBA12	TBA11	TBA10	TBA9	TBA8	TBA7	TBA6	TBA5	TBA4	TBA3	TBA2	TBA1	TBA0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-22. Transfer Base Address (TBA) Field Descriptions

Bit	Name	Value	Description
31-0	TBA[31:0]		Transfer Base Address 32-bit base pointer, 2 LSB are not significant (32-bit accesses only) will be ‘0’ for read.

17.21.8 Next Transfer Base Address (NTBA)

The Next Transfer Base Address hold a 32bit aligned 32bit base-pointer to be loaded into TBA during next cycle start.

Note:
A write on TBA register modifies also the NTBA register.

Figure 17-35. Next Transfer Base Address (NTBA) [offset_TU = 0x28]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTBA 15	NTBA 14	NTBA 13	NTBA 12	NTBA 11	NTBA 10	NTBA 9	NTBA 8	NTBA 7	NTBA 6	NTBA 5	NTBA 4	NTBA 3	NTBA 2	NTBA 1	NTBA 0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-23. Next Transfer Base Address (NTBA) Field Descriptions

Bit	Name	Value	Description
31-0	NTBA[31:0]		Next Transfer Base Address 32-bit base pointer, 2 LSB are not significant (32-bit accesses only) will be ‘0’ for read.

17.21.9 Base Address of Mirrored Status (BAMS)

The Base Address of Mirrored Status hold a 32bit aligned 32bit base-pointer to be use for mirror transactions.

Figure 17-36. Base Address of Mirrored Status (BAMS) [offset_TU = 0x2C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAMS 15	BAMS 14	BAMS 13	BAMS 12	BAMS 11	BAMS 10	BAMS 9	BAMS 8	BAMS 7	BAMS 6	BAMS 5	BAMS 4	BAMS 3	BAMS 2	BAMS 1	BAMS 0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-24. Base Address of Mirrored Status (BAMS) Field Descriptions

Bit	Name	Value	Description
31-0	BAMS[31:0]		Base Address of Mirrored Status 32-bit base pointer, 2 LSB are not significant (32-bit accesses only) will be ‘0’ for read.

17.21.10Start Address of Memory Protection (SAMP)

The Start Address of Memory Protection hold a 32bit address.

Figure 17-37. Start Address of Memory Protection (SAMP) [offset_TU = 0x30]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAMP 15	SAMP 14	SAMP 13	SAMP 12	SAMP 11	SAMP 10	SAMP 9	SAMP 8	SAMP 7	SAMP 6	SAMP 5	SAMP 4	SAMP 3	SAMP 2	SAMP 1	SAMP 0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-25. Start Address of Memory Protection (SAMP) Field Descriptions

Bit	Name	Value	Description
31-0	SAMP[31:0]		Start Address Memory Protection Start address of the memory area, which allows read and write accesses for the Transfer Unit State Machine. 32-bit base pointer, 2 LSB are not significant (32-bit accesses only) will be ‘0’ for read.

17.21.11End Address of Memory Protection (EAMP)

The End Address of Memory Protection hold a 32bit address.

Figure 17-38. End Address of Memory Protection (EAMP) [offset_TU = 0x30]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EAMP 15	EAMP 14	EAMP 13	EAMP 12	EAMP 11	EAMP 10	EAMP 9	EAMP 8	EAMP 7	EAMP 6	EAMP 5	EAMP 4	EAMP 3	EAMP 2	EAMP 1	EAMP 0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-26. End Address of Memory Protection (EAMP) Field Descriptions

Bit	Name	Value	Description
31-0	EAMP[31:0]		End Address Memory Protection End address of the memory area, which allows read and write accesses for the Transfer Unit State Machine. 32-bit address, 2 LSB are not significant (32-bit accesses only) will be ‘0’ for read.

17.21.12Transfer to System Memory Occurred 1/2/3/4 (TSMO1-4)

The Transfer to System Memory Occurred register reflects the message buffer transfer status for a VBUSP master transfer transaction to the system memory. Four 32-bit registers reflect all possible 128 message buffers.

Note:
Writing ’1’ will clear a bit. Writing ’0’ will leave a bit unchanged.

Figure 17-39. Transfer to System Memory Occurred 1 (TSMO1) [offset_TU = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMO 1-31	TSMO 1-30	TSMO 1-29	TSMO 1-28	TSMO 1-27	TSMO 1-26	TSMO 1-25	TSMO 1-24	TSMO 1-23	TSMO 1-22	TSMO 1-21	TSMO 1-20	TSMO 1-19	TSMO 1-18	TSMO 1-17	TSMO 1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMO 1-15	TSMO 1-14	TSMO 1-13	TSMO 1-12	TSMO 1-11	TSMO 1-10	TSMO 1-9	TSMO 1-8	TSMO 1-7	TSMO 1-6	TSMO 1-5	TSMO 1-4	TSMO 1-3	TSMO 1-2	TSMO 1-1	TSMO 1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-27. Transfer to System Memory Occurred 1 (TSMO1) Field Descriptions

Bit	Name	Value	Description
31-0	TSMO1[31:0]		Transfer to System Memory Occurred Register 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register reflects a finished message buffer transfer to the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-40. Transfer to System Memory Occurred 2 (TSMO2) [offset_TU = 0x44]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMO 2-31	TSMO 2-30	TSMO 2-29	TSMO 2-28	TSMO 2-27	TSMO 2-26	TSMO 2-25	TSMO 2-24	TSMO 2-23	TSMO 2-22	TSMO 2-21	TSMO 2-20	TSMO 2-19	TSMO 2-18	TSMO 2-17	TSMO 2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMO 2-15	TSMO 2-14	TSMO 2-13	TSMO 2-12	TSMO 2-11	TSMO 2-10	TSMO 2-9	TSMO 2-8	TSMO 2-7	TSMO 2-6	TSMO 2-5	TSMO 2-4	TSMO 2-3	TSMO 2-2	TSMO 2-1	TSMO 2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-28. Transfer to System Memory Occurred 2 (TSMO2) Field Descriptions

Bit	Name	Value	Description
31-0	TSMO2[31:0]		Transfer to System Memory Occurred Register 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register reflects a finished message buffer transfer to the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-41. Transfer to System Memory Occurred 3 (TSMO3) [offset_TU = 0x48]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMO 3-31	TSMO 3-30	TSMO 3-29	TSMO 3-28	TSMO 3-27	TSMO 3-26	TSMO 3-25	TSMO 3-24	TSMO 3-23	TSMO 3-22	TSMO 3-21	TSMO 3-20	TSMO 3-19	TSMO 3-18	TSMO 3-17	TSMO 3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMO 3-15	TSMO 3-14	TSMO 3-13	TSMO 3-12	TSMO 3-11	TSMO 3-10	TSMO 3-9	TSMO 3-8	TSMO 3-7	TSMO 3-6	TSMO 3-5	TSMO 3-4	TSMO 3-3	TSMO 3-2	TSMO 3-1	TSMO 3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-29. Transfer to System Memory Occurred 3 (TSMO3) Field Descriptions

Bit	Name	Value	Description
31-0	TSMO3[31:0]		Transfer to System Memory Occurred Register 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register reflects a finished message buffer transfer to the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-42. Transfer to System Memory Occurred 4 (TSMO4) [offset_TU = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMO 4-31	TSMO 4-30	TSMO 4-29	TSMO 4-28	TSMO 4-27	TSMO 4-26	TSMO 4-25	TSMO 4-24	TSMO 4-23	TSMO 4-22	TSMO 4-21	TSMO 4-20	TSMO 4-19	TSMO 4-18	TSMO 4-17	TSMO 4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMO 4-15	TSMO 4-14	TSMO 4-13	TSMO 4-12	TSMO 4-11	TSMO 4-10	TSMO 4-9	TSMO 4-8	TSMO 4-7	TSMO 4-6	TSMO 4-5	TSMO 4-4	TSMO 4-3	TSMO 4-2	TSMO 4-1	TSMO 4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-30. Transfer to System Memory Occurred 4 (TSMO4) Field Descriptions

Bit	Name	Value	Description
31-0	TSMO4[31:0]		Transfer to System Memory Occurred Register 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register reflects a finished message buffer transfer to the system memory.
		0	No transfer occurred
		1	Transfer Occurred

17.21.13Transfer to Communication Controller Occurred 1/2/3/4 (TCCO1-4)

The Transfer to Communication Controller Occurred reflects the message buffer transfer status for a VBUSP master transfer transaction from the system memory. Four 32-bit registers reflect all possible 128 message buffers.

Note:
Writing ’1’ will clear a bit. Writing ’0’ will leave a bit unchanged.

Figure 17-43. Transfer to Communication Controller Occurred 1 (TCCO1) [offset_TU = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCO 1-31	TCCO 1-30	TCCO 1-29	TCCO 1-28	TCCO 1-27	TCCO 1-26	TCCO 1-25	TCCO 1-24	TCCO 1-23	TCCO 1-22	TCCO 1-21	TCCO 1-20	TCCO 1-19	TCCO 1-18	TCCO 1-17	TCCO 1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCO 1-15	TCCO 1-14	TCCO 1-13	TCCO 1-12	TCCO 1-11	TCCO 1-10	TCCO 1-9	TCCO 1-8	TCCO 1-7	TCCO 1-6	TCCO 1-5	TCCO 1-4	TCCO 1-3	TCCO 1-2	TCCO 1-1	TCCO 1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-31. (Transfer *to Communication Controller Occurred* 1 (TCCO1) Field Descriptions

Bit	Name	Value	Description
31-0	TCCO1[31:0]		Transfer to Communication Controller Occurred Register 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register reflects a finished message buffer transfer from the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-44. Transfer to Communication Controller Occurred 2 (TCCO2) [offset_TU = 0x54]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCO 2-31	TCCO 2-30	TCCO 2-29	TCCO 2-28	TCCO 2-27	TCCO 2-26	TCCO 2-25	TCCO 2-24	TCCO 2-23	TCCO 2-22	TCCO 2-21	TCCO 2-20	TCCO 2-19	TCCO 2-18	TCCO 2-17	TCCO 2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCO 2-15	TCCO 2-14	TCCO 2-13	TCCO 2-12	TCCO 2-11	TCCO 2-10	TCCO 2-9	TCCO 2-8	TCCO 2-7	TCCO 2-6	TCCO 2-5	TCCO 2-4	TCCO 2-3	TCCO 2-2	TCCO 2-1	TCCO 2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-32. (Transfer *to Communication Controller Occurred* 2 (TCCO2) Field Descriptions

Bit	Name	Value	Description
31-0	TCCO2[31:0]		Transfer to Communication Controller Occurred Register 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register reflects a finished message buffer transfer from the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-45. Transfer to Communication Controller Occurred 3 (TCCO3) [offset_TU = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCO 3-31	TCCO 3-30	TCCO 3-29	TCCO 3-28	TCCO 3-27	TCCO 3-26	TCCO 3-25	TCCO 3-24	TCCO 3-23	TCCO 3-22	TCCO 3-21	TCCO 3-20	TCCO 3-19	TCCO 3-18	TCCO 3-17	TCCO 3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCO 3-15	TCCO 3-14	TCCO 3-13	TCCO 3-12	TCCO 3-11	TCCO 3-10	TCCO 3-9	TCCO 3-8	TCCO 3-7	TCCO 3-6	TCCO 3-5	TCCO 3-4	TCCO 3-3	TCCO 3-2	TCCO 3-1	TCCO 3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-33. (Transfer *to Communication Controller Occurred* 3 (TCCO3) Field Descriptions

Bit	Name	Value	Description
31-0	TCCO3[31:0]		Transfer to Communication Controller Occurred Register 3 The register bits 0 to 31 are corresponding to message buffers 64 to 65. Each bit of the register reflects a finished message buffer transfer from the system memory.
		0	No transfer occurred
		1	Transfer Occurred

Figure 17-46. Transfer to Communication Controller Occurred 4 (TCCO4) [offset_TU = 0x5C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCO 4-31	TCCO 4-30	TCCO 4-29	TCCO 4-28	TCCO 4-27	TCCO 4-26	TCCO 4-25	TCCO 4-24	TCCO 4-23	TCCO 4-22	TCCO 4-21	TCCO 4-20	TCCO 4-19	TCCO 4-18	TCCO 4-17	TCCO 4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCO 4-15	TCCO 4-14	TCCO 4-13	TCCO 4-12	TCCO 4-11	TCCO 4-10	TCCO 4-9	TCCO 4-8	TCCO 4-7	TCCO 4-6	TCCO 4-5	TCCO 4-4	TCCO 4-3	TCCO 4-2	TCCO 4-1	TCCO 4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-34. (Transfer *to Communication Controller Occurred* 4 (TCCO4) Field Descriptions

Bit	Name	Value	Description
31-0	TCCO4[31:0]		Transfer to Communication Controller Occurred Register 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register reflects a finished message buffer transfer from the system memory.
		0	No transfer occurred
		1	Transfer Occurred

17.21.14Transfer Occurred Offset (TOOFF)

The Transfer Occurred Offset register contains the offset vector to the highest prior pending transfer occurred interrupt and the transfer direction.

After a read access the transfer occurred flag is cleared and the register contents will be updated automatically.

Figure 17-47. Transfer Occurred Offset (TOOFF) [offset_TU = 0x60]

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
Reserved							TDIR	OFF7	OFF6	OFF5	OFF4	OFF3	OFF2	OFF1	OFF0
R-0							R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-35. (Transfer Occurred Offset (TOOFF) Field Descriptions

Bit	Name	Value	Description
31-9	Reserved		Reads return zero and writes have no effect.
8	TDIR		Transfer Direction In case the same interrupt occurs for communication controller and Transfer Unit state machine transfers the PRIO bit in the Global Control register decides about the higher priority.
		0	A transfer to System Memory occurred.
		1	A transfer to the Communication Controller occurred.
7-0	OFF[7:0]		Offset Vector
		00000000	Offset not valid (no transfer occurred, interrupt pending).
		00000001	Interrupt pending for buffer 0.
		00000010	Interrupt pending for buffer 1.
		00000011	Interrupt pending for buffer 2.
		...	
		10000000	Interrupt pending for buffer 127.
		10000001	Reserved
		......	
		11111111	Reserved

17.21.15Parity Error Address (PEADR)

After a parity error in the Transfer Configuration RAM occurred, the affected address is stored in this non resetable register.

The contents of the Parity Error Address register as well as the PE bit in the Transfer Error InterRupt register (TEIR) is cleared automatically when reading the Parity Error Address register.

Figure 17-48. Parity Error Address (PEADR) [offset_TU = 0x70]

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
Reserved							ADR8	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
R-0							RC-U	RC-U	RC-U	RC-U	RC-U	RC-U	RC-U	RC-U	RC-U

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-36. Parity Error Address (PEADR) Field Descriptions

Bit	Name	Value	Description
31-9	Reserved		Reads return zero and writes have no effect.
8-0	ADR[8:0]		Address of failing TCR location ADR[8:2] is the TCR word address were the parity error occurred. ADR[1:0] can be used to find the failing byte in the TCR word the parity error occurred.

Table 17-37. Coding of Parity Error Address

Parity Error in Byte 2	Parity Error in Byte 1	Parity Error in Byte 0 (LSB)	ADR1	ADR0
-	-	1	0	1
-	1	0	1	0
1	0	0	1	1
0	0	0	0	0

Note:
No Byte 3 is defined in TCR

17.21.16Transfer Error Interrupt (TEIR)

The Transfer Error Interrupt register includes the Transfer Unit error flags. The bits in the TEIR are cleared by writing a ‘1’.

Figure 17-49. Transfer Error InterRupt (TEIR) [offset_TU = 0x74]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RSTAT 2	RSTAT 1	RSTAT 0	Reserved	SSTAT 2	SSTAT 1	SSTAT 0	Reserved		TNR	FAC
R-0					RC-0	RC-0	RC-0	R-0	RC-0	RC-0	RC-0	R-0		RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-38. Transfer Error InterRupt (TEIR) Field Descriptions

Bit	Name	Value	Description
31-18	Reserved		Reads return zero and writes have no effect.
17	MPV		Memory Protection Violation
		0	No MPV occurred.
		1	MPV Occurred.
16	PE		Parity Error
		0	No PE occurred.
		1	PE Occurred.
15-11	Reserved		Reads return zero and writes have no effect.
10-8	RSTAT[3:0]		Status of Transfer Unit State Machine Read
		000	Success
		0001	Addressing error
		010	Protection error
		011	Timeout error
		100	Data error
		101	Unsupported addressing mode error
		110	Reserved
		111	Exclusive read failure
7	Reserved		Reads return zero and writes have no effect.

Table 17-38. Transfer Error InterRupt (TEIR) Field Descriptions (Continued)

Bit	Name	Value	Description
6-4	SSTAT[3:0]		Status of Transfer Unit State Machine Write
		000	Success
		0001	Addressing error
		010	Protection error
		011	Timeout error
		100	Reserved
		101	Unsupported addressing mode error
		110	Reserved
		111	Exclusive write failure
3-2	Reserved		Reads return zero and writes have no effect.
1	TNR		Transfer Not Ready
		0	Transfer started and NTBA is loaded to TBA.
		1	Transfer is not ready and NTBA is not loaded to TBA.
0	FAC		Forbidden Access
		0	No forbidden access occurred.
		1	A forbidden access of the CPU to IBF or OBF occurred when the Transfer Unit State Machine was switched on.

17.21.17Transfer Error Interrupt Enable Set/Reset (TEIRES/R)

The Transfer Error Interrupt Enable Set controls the interrupt activation of interrupt line TU_Int1. An interrupt is generated if both the interrupt flag in TEIR and the corresponding bit in TEIRES are set.

A Transfer Error Interrupt is enabled by writing ‘1’ to TEIRES register and disabled by writing ‘1’ to TIERER register. Writing of ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-50. Transfer Error Interrupt Enable Set (TEIRES) [offset_TU = 0x78]

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
Reserved					RSTATE2	RSTATE1	RSTATE0	Reserved	SSTATE2	SSTATE1	SSTATE0	reserved		TNRE	FACE
R-0					RS-0	RS-0	RS-0	R-0	RS-0	RS-0	RS-0	R-0	R-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-39. Transfer Error Interrupt Enable Set (TEIRES)

Bit	Name	Value	Description
31-11	Reserved		Reads return zero and writes have no effect.
10-8	RSTATE[2:0]		Read Transfer Interrupt Generation
		000	Success
		001	Addressing error
		010	Protection error
		011	Timeout error
		100	Data error
		101	Unsupported addressing mode error
		110	Reserved
		111	Exclusive write failure
7	Reserved		Reads return zero and writes have no effect.
6-4	SSTATE[2:0]		Write Transfer Interrupt Generation
		000	Success
		001	Addressing error
3-2	Reserved		Reads return zero and writes have no effect.
1	TNRE		Transfer Not Ready Enable
		0	TNR interrupt disabled
		1	TNR interrupt enabled

Table 17-39. Transfer Error Interrupt Enable Set (TEIRES) (Continued)

Bit	Name	Value	Description
1	FACE		Forbidden Access Enable
		0	FAC interrupt disabled
		1	FAC interrupt enabled

Figure 17-51. Transfer Error Interrupt Enable Reset (TEIRER) [offset_TU = 0x7C]

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
Reserved					RSTATE2	RSTATE1	RSTATE0	Reserved	SSTATE2	SSTATE1	SSTATE0	reserved		TNRE	FACE
R-0					RC-0	RC-0	RC-0	R-0	RC-0	RC-0	RC-0	R-0	R-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-40. Transfer Error Interrupt Enable Reset (TEIRER)

Bit	Name	Value	Description
31-11	Reserved		Reads return zero and writes have no effect.
10-8	RSTATE[2:0]		Read Transfer Interrupt Generation
		000	Success
		001	Addressing error
		010	Protection error
		011	Timeout error
		100	Data error
		101	Unsupported addressing mode error
		110	Reserved
		111	Exclusive write failure
7	Reserved		Reads return zero and writes have no effect.
6-4	SSTATE[2:0]		Write Transfer Interrupt Generation
		000	Success
		001	Addressing error
3-2	Reserved		Reads return zero and writes have no effect.

Table 17-40. Transfer Error Interrupt Enable Reset (TEIRER) (Continued)

Bit	Name	Value	Description
1	TNRE		Transfer Not Ready Enable
		0	TNR interrupt disabled
		1	TNR interrupt enabled
1	FACE		Forbidden Access Enable
		0	FAC interrupt disabled
		1	FAC interrupt enabled

17.21.18Trigger Transfer to System Memory Set/Reset 1/2/3/4 (TTSMS/R1-4)

The Trigger Transfer to System Memory register selects the actual message buffer for a Transfer Unit state machine transfer transaction to system memory. Four 32-bit registers reflect all possible 128 message buffers.

The bits are set by writing ‘1’ to TTSMSx and reset by writing ‘1’ to TTSMRx. Writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-52. Trigger Transfer to System Memory Set 1 (TTSMS1) [offset_TU = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM S1-31	TTSM S1-30	TTSM S1-29	TTSM S1-28	TTSM S1-27	TTSM S1-26	TTSM S1-25	TTSM S1-24	TTSM S1-23	TTSM S1-22	TTSM S1-21	TTSM S1-20	TTSM S1-19	TTSM S1-18	TTSM S1-17	TTSM S1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM S1-15	TTSM S1-14	TTSM S1-13	TTSM S1-12	TTSM S1-11	TTSM S1-10	TTSM S1-9	TTSM S1-8	TTSM S1-7	TTSM S1-6	TTSM S1-5	TTSM S1-4	TTSM S1-3	TTSM S1-2	TTSM S1-1	TTSM S1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-41. Trigger Transfer to System Memory Set 1 (TTSMS1) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMS1[31:0]		Trigger Transfer to System Memory Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register controls the message buffer transfer to the system memory in the following manner (not that only the least significant bit of all four combined TTSM registers will actually scheduled for transmission).
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-53. Trigger Transfer to System Memory Reset 1 (TTSMR1) [offset_TU = 0x84]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM R1-31	TTSM R1-30	TTSM R1-29	TTSM R1-28	TTSM R1-27	TTSM R1-26	TTSM R1-25	TTSM R1-24	TTSM R1-23	TTSM R1-22	TTSM R1-21	TTSM R1-20	TTSM R1-19	TTSM R1-18	TTSM R1-17	TTSM R1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM R1-15	TTSM R1-14	TTSM R1-13	TTSM R1-12	TTSM R1-11	TTSM R1-10	TTSM R1-9	TTSM R1-8	TTSM R1-7	TTSM R1-6	TTSM R1-5	TTSM R1-4	TTSM R1-3	TTSM R1-2	TTSM R1-1	TTSM R1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-42. Trigger Transfer to System Memory Reset 1 (TTSMR1) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMR1[31:0]		Trigger Transfer to System Memory Reset 1 The TTSMR1 register shows the identical values to TTSMS1 if read.

Figure 17-54. Trigger Transfer to System Memory Set 2 (TTSMS2) [offset_TU = 0x88]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM S2-31	TTSM S2-30	TTSM S2-29	TTSM S2-28	TTSM S2-27	TTSM S2-26	TTSM S2-25	TTSM S2-24	TTSM S2-23	TTSM S2-22	TTSM S2-21	TTSM S2-20	TTSM S2-19	TTSM S2-18	TTSM S2-17	TTSM S2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM S2-15	TTSM S2-14	TTSM S2-13	TTSM S2-12	TTSM S2-11	TTSM S2-10	TTSM S2-9	TTSM S2-8	TTSM S2-7	TTSM S2-6	TTSM S2-5	TTSM S2-4	TTSM S2-3	TTSM S2-2	TTSM S2-1	TTSM S2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-43. Trigger Transfer to System Memory Set 2 (TTSMS2) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMS2[31:0]		Trigger Transfer to System Memory Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63.
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-55. Trigger Transfer to System Memory Reset 2 (TTSMR2) [offset_TU = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM R2-31	TTSM R2-30	TTSM R2-29	TTSM R2-28	TTSM R2-27	TTSM R2-26	TTSM R2-25	TTSM R2-24	TTSM R2-23	TTSM R2-22	TTSM R2-21	TTSM R2-20	TTSM R2-19	TTSM R2-18	TTSM R2-17	TTSM R2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM R2-15	TTSM R2-14	TTSM R2-13	TTSM R2-12	TTSM R2-11	TTSM R2-10	TTSM R2-9	TTSM R2-8	TTSM R2-7	TTSM R2-6	TTSM R2-5	TTSM R2-4	TTSM R2-3	TTSM R2-2	TTSM R2-1	TTSM R2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-44. Trigger Transfer to System Memory Reset 2 (TTSMR2) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMR2[31:0]		Trigger Transfer to System Memory Reset 2 The TTSMR2 register shows the identical values to TTSMS2 if read.

The TTSMR2 register shows the identical values to TTSMS2 if read.

Figure 17-56. Trigger Transfer to System Memory Set 3 (TTSMS3) [offset_TU = 0x90]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM S3-31	TTSM S3-30	TTSM S3-29	TTSM S3-28	TTSM S3-27	TTSM S3-26	TTSM S3-25	TTSM S3-24	TTSM S3-23	TTSM S3-22	TTSM S3-21	TTSM S3-20	TTSM S3-19	TTSM S3-18	TTSM S3-17	TTSM S3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM S3-15	TTSM S3-14	TTSM S3-13	TTSM S3-12	TTSM S3-11	TTSM S3-10	TTSM S3-9	TTSM S3-8	TTSM S3-7	TTSM S3-6	TTSM S3-5	TTSM S3-4	TTSM S3-3	TTSM S3-2	TTSM S3-1	TTSM S3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-45. Trigger Transfer to System Memory Set 3 (TTSMS3) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMS3[31:0]		Trigger Transfer to System Memory Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95.
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-57. Trigger Transfer to System Memory Reset 3 (TTSMR3) [offset_TU = 0x94]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM R3-31	TTSM R3-30	TTSM R3-29	TTSM R3-28	TTSM R3-27	TTSM R3-26	TTSM R3-25	TTSM R3-24	TTSM R3-23	TTSM R3-22	TTSM R3-21	TTSM R3-20	TTSM R3-19	TTSM R3-18	TTSM R3-17	TTSM R3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM R3-15	TTSM R3-14	TTSM R3-13	TTSM R3-12	TTSM R3-11	TTSM R3-10	TTSM R3-9	TTSM R3-8	TTSM R3-7	TTSM R3-6	TTSM R3-5	TTSM R3-4	TTSM R3-3	TTSM R3-2	TTSM R3-1	TTSM R3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-46. Trigger Transfer to System Memory Reset 3 (TTSMR3) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMR3[31:0]		Trigger Transfer to System Memory Reset 3 The TTSMR3 register shows the identical values to TTSMS3 if read.

Figure 17-58. Trigger Transfer to System Memory Set 4 (TTSMS4) [offset_TU = 0x98]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM S4-31	TTSM S4-30	TTSM S4-29	TTSM S4-28	TTSM S4-27	TTSM S4-26	TTSM S4-25	TTSM S4-24	TTSM S4-23	TTSM S4-22	TTSM S4-21	TTSM S4-20	TTSM S4-19	TTSM S4-18	TTSM S4-17	TTSM S4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM S4-15	TTSM S4-14	TTSM S4-13	TTSM S4-12	TTSM S4-11	TTSM S4-10	TTSM S4-9	TTSM S4-8	TTSM S4-7	TTSM S4-6	TTSM S4-5	TTSM S4-4	TTSM S4-3	TTSM S4-2	TTSM S4-1	TTSM S4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-47. Trigger Transfer to System Memory Set 4 (TTSMS4) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMS4[31:0]		Trigger Transfer to System Memory Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127.
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-59. Trigger Transfer to System Memory Reset 4 (TTSMR4) [offset_TU = 0x9C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTSM R4-31	TTSM R4-30	TTSM R4-29	TTSM R4-28	TTSM R4-27	TTSM R4-26	TTSM R4-25	TTSM R4-24	TTSM R4-23	TTSM R4-22	TTSM R4-21	TTSM R4-20	TTSM R4-19	TTSM R4-18	TTSM R4-17	TTSM R4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSM R4-15	TTSM R4-14	TTSM R4-13	TTSM R4-12	TTSM R4-11	TTSM R4-10	TTSM R4-9	TTSM R4-8	TTSM R4-7	TTSM R4-6	TTSM R4-5	TTSM R4-4	TTSM R4-3	TTSM R4-2	TTSM R4-1	TTSM R4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-48. Trigger Transfer to System Memory Reset 4 (TTSMR4) Field Descriptions

Bit	Name	Value	Description
31-0	TTSMR4[31:0]		Trigger Transfer to System Memory Reset 4 The TTSMR4 register shows the identical values to TTSMS4 if read.

17.21.19Trigger Transfer to Communication Controller Set/Reset 1/2/3/4 (TTCCS/R1-4)

The Trigger Transfer to Communication Controller registers select the actual message buffer for a Transfer Unit state machine transfer transaction from system memory. Four 32-bit registers reflect all possible 128 message buffers.

The bits are set by writing ‘1’ to TTCCSx and reset by writing ‘1’ to TTCCRx. Writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-60. Trigger Transfer to Communication Controller Set 1 (TTCCS1) [offset_TU = 0xA0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCCS 1-31	TTCCS 1-30	TTCCS 1-29	TTCCS 1-28	TTCCS 1-27	TTCCS 1-26	TTCCS 1-25	TTCCS 1-24	TTCCS 1-23	TTCCS 1-22	TTCCS 1-21	TTCCS 1-20	TTCCS 1-19	TTCCS 1-18	TTCCS 1-17	TTCCS 1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCCS 1-15	TTCCS 1-14	TTCCS 1-13	TTCCS 1-12	TTCCS 1-11	TTCCS 1-10	TTCCS 1-9	TTCCS 1-8	TTCCS 1-7	TTCCS 1-6	TTCCS 1-5	TTCCS 1-4	TTCCS 1-3	TTCCS 1-2	TTCCS 1-1	TTCCS 1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-49. Trigger Transfer to Communication Controller Set 1 (TTCCS1) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCS1[31:0]		Trigger Transfer to Communication Controller Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register controls the message buffer transfer to the communication controller in the following manner:
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-61. Trigger Transfer to Communication Controller Reset 1 (TTCCR1) [offset_TU = 0xA4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCCR1-31	TTCCR1-30	TTCCR1-29	TTCCR1-28	TTCCR1-27	TTCCR1-26	TTCCR1-25	TTCCR1-24	TTCCR1-23	TTCCR1-22	TTCCR1-21	TTCCR1-20	TTCCR1-19	TTCCR1-18	TTCCR1-17	TTCCR1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCCR1-15	TTCCR1-14	TTCCR1-13	TTCCR1-12	TTCCR1-11	TTCCR1-10	TTCCR1-9	TTCCR1-8	TTCCR1-7	TTCCR1-6	TTCCR1-5	TTCCR1-4	TTCCR1-3	TTCCR1-2	TTCCR1-1	TTCCR1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-50. Trigger Transfer to Communication Controller Reset 1 (TTCCR1) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCR1[31:0]		Trigger Transfer to Communication Controller Reset 1 The TTCCR1 register shows the identical values to TTCCS1 if read.

Figure 17-62. Trigger Transfer to Communication Controller Set 2 (TTCCS2) [offset_TU = 0xA8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCCS2-31	TTCCS2-30	TTCCS2-29	TTCCS2-28	TTCCS2-27	TTCCS2-26	TTCCS2-25	TTCCS2-24	TTCCS2-23	TTCCS2-22	TTCCS2-21	TTCCS2-20	TTCCS2-19	TTCCS2-18	TTCCS2-17	TTCCS2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCCS2-15	TTCCS2-14	TTCCS2-13	TTCCS2-12	TTCCS2-11	TTCCS2-10	TTCCS2-9	TTCCS2-8	TTCCS2-7	TTCCS2-6	TTCCS2-5	TTCCS2-4	TTCCS2-3	TTCCS2-2	TTCCS2-1	TTCCS2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-51. Trigger Transfer to Communication Controller Set 2 (TTCCS2) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCS2[31:0]		Trigger Transfer to Communication Controller Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63.
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-63. Trigger Transfer to Communication Controller Reset 2 (TTCCR2) [offset_TU = 0xAC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCC R2-31	TTCC R2-30	TTCC R2-29	TTCC R2-28	TTCC R2-27	TTCC R2-26	TTCC R2-25	TTCC R2-24	TTCC R2-23	TTCC R2-22	TTCC R2-21	TTCC R2-20	TTCC R2-19	TTCC R2-18	TTCC R2-17	TTCC R2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCC R2-15	TTCC R2-14	TTCC R2-13	TTCC R2-12	TTCC R2-11	TTCC R2-10	TTCC R2-9	TTCC R2-8	TTCC R2-7	TTCC R2-6	TTCC R2-5	TTCC R2-4	TTCC R2-3	TTCC R2-2	TTCC R2-1	TTCC R2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-52. Trigger Transfer to Communication Controller Reset 2 (TTCCR2) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCR2[31:0]		Trigger Transfer to Communication Controller Reset 2 The TTCCR2 register shows the identical values to TTCCS2 if read.

Figure 17-64. Trigger Transfer to Communication Controller Set 3 (TTCCS3) [offset_TU = 0xB0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCCS 3-31	TTCCS 3-30	TTCCS 3-29	TTCCS 3-28	TTCCS 3-27	TTCCS 3-26	TTCCS 3-25	TTCCS 3-24	TTCCS 3-23	TTCCS 3-22	TTCCS 3-21	TTCCS 3-20	TTCCS 3-19	TTCCS 3-18	TTCCS 3-17	TTCCS 3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCCS 3-15	TTCCS 3-14	TTCCS 3-13	TTCCS 3-12	TTCCS 3-11	TTCCS 3-10	TTCCS 3-9	TTCCS 3-8	TTCCS 3-7	TTCCS 3-6	TTCCS 3-5	TTCCS 3-4	TTCCS 3-3	TTCCS 3-2	TTCCS 3-1	TTCCS 3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-53. Trigger Transfer to Communication Controller Set 3 (TTCCS3) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCS3[31:0]		Trigger Transfer to Communication Controller Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register controls the message buffer transfer to the communication controller in the following manner:
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-65. Trigger Transfer to Communication Controller Reset 3 (TTCCR3) [offset_TU = 0xB4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCC R3-31	TTCC R3-30	TTCC R3-29	TTCC R3-28	TTCC R3-27	TTCC R3-26	TTCC R3-25	TTCC R3-24	TTCC R3-23	TTCC R3-22	TTCC R3-21	TTCC R3-20	TTCC R3-19	TTCC R3-18	TTCC R3-17	TTCC R3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCC R3-15	TTCC R3-14	TTCC R3-13	TTCC R3-12	TTCC R3-11	TTCC R3-10	TTCC R3-9	TTCC R3-8	TTCC R3-7	TTCC R3-6	TTCC R3-5	TTCC R3-4	TTCC R3-3	TTCC R3-2	TTCC R3-1	TTCC R3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-54. Trigger Transfer to Communication Controller Reset 3 (TTCCR3) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCR3[31:0]		Trigger Transfer to Communication Controller Reset 3 The TTCCR3 register shows the identical values to TTCCS3 if read.

Figure 17-66. Trigger Transfer to Communication Controller Set 4 (TTCCS4) [offset_TU = 0xB8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCCS 4-31	TTCCS 4-30	TTCCS 4-29	TTCCS 4-28	TTCCS 4-27	TTCCS 4-26	TTCCS 4-25	TTCCS 4-24	TTCCS 4-23	TTCCS 4-22	TTCCS 4-21	TTCCS 4-20	TTCCS 4-19	TTCCS 4-18	TTCCS 4-17	TTCCS 4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCCS 4-15	TTCCS 4-14	TTCCS 4-13	TTCCS 4-12	TTCCS 4-11	TTCCS 4-10	TTCCS 4-9	TTCCS 4-8	TTCCS 4-7	TTCCS 4-6	TTCCS 4-5	TTCCS 4-4	TTCCS 4-3	TTCCS 4-2	TTCCS 4-1	TTCCS 4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-55. Trigger Transfer to Communication Controller Set 4 (TTCCS4) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCS4[31:0]		Trigger Transfer to Communication Controller Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register controls the message buffer transfer to the communication controller in the following manner:
		0	No transfer request
		1	Transfer based on address defined in TBA

Figure 17-67. Trigger Transfer to Communication Controller Reset 4 (TTCCR4) [offset_TU = 0xBC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TTCC R4-31	TTCC R4-30	TTCC R4-29	TTCC R4-28	TTCC R4-27	TTCC R4-26	TTCC R4-25	TTCC R4-24	TTCC R4-23	TTCC R4-22	TTCC R4-21	TTCC R4-20	TTCC R4-19	TTCC R4-18	TTCC R4-17	TTCC R4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTCC R4-15	TTCC R4-14	TTCC R4-13	TTCC R4-12	TTCC R4-11	TTCC R4-10	TTCC R4-9	TTCC R4-8	TTCC R4-7	TTCC R4-6	TTCC R4-5	TTCC R4-4	TTCC R4-3	TTCC R4-2	TTCC R4-1	TTCC R4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-56. Trigger Transfer to Communication Controller Reset 4 (TTCCR4) Field Descriptions

Bit	Name	Value	Description
31-0	TTCCR4[31:0]		Trigger Transfer to Communication Controller Reset 4 The TTCCR4 register shows the identical values to TTCCS4 if read.

17.21.20Enable Transfer on Event to System Memory Set/Reset 1-4 (ETESMS/R1-4)

The Enable Transfer on Event to System Memory Set registers enable a message buffer transfer to the system memory after a receive or transmit event. Four 32-bit registers reflect all possible 128 message buffers.

The bits are set by writing ‘1’ to ETESMSx and reset by writing ‘1’ to ETESMRx. Writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-68. Enable Transfer on Event to System Memory Set 1 (ETESMS1) [offset_TU = 0xC0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMS1-31	ETESMS1-30	ETESMS1-29	ETESMS1-28	ETESMS1-27	ETESMS1-26	ETESMS1-25	ETESMS1-24	ETESMS1-23	ETESMS1-22	ETESMS1-21	ETESMS1-20	ETESMS1-19	ETESMS1-18	ETESMS1-17	ETESMS1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMS1-15	ETESMS1-14	ETESMS1-13	ETESMS1-12	ETESMS1-11	ETESMS1-10	ETESMS1-9	ETESMS1-8	ETESMS1-7	ETESMS1-6	ETESMS1-5	ETESMS1-4	ETESMS1-3	ETESMS1-2	ETESMS1-1	ETESMS1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-57. Enable Transfer on Event to System Memory Set 1 Field Descriptions

Bit	Name	Value	Description
31-0	ETESMS1[31:0]		Enable Transfer on Event to System Memory Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register enables a message buffer transfer on event to the system memory:
		0	Transfer on event is disabled
		1	Transfer on event is enabled

Figure 17-69. Enable Transfer on Event to System Memory Reset 1 (ETESMR1) [offset_TU = 0xC4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMR1-31	ETESMR1-30	ETESMR1-29	ETESMR1-28	ETESMR1-27	ETESMR1-26	ETESMR1-25	ETESMR1-24	ETESMR1-23	ETESMR1-22	ETESMR1-21	ETESMR1-20	ETESMR1-19	ETESMR1-18	ETESMR1-17	ETESMR1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMR1-15	ETESMR1-14	ETESMR1-13	ETESMR1-12	ETESMR1-11	ETESMR1-10	ETESMR1-9	ETESMR1-8	ETESMR1-7	ETESMR1-6	ETESMR1-5	ETESMR1-4	ETESMR1-3	ETESMR1-2	ETESMR1-1	ETESMR1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-58. Enable Transfer on Event to System Memory Reset 1 (ETESMR1) Field Descriptions

Bit	Name	Value	Description
31-0	ETESMR1[31:0]		Enable Transfer on Event to System Memory Reset 1 The ETESMR1 register shows the identical values to ETESMS1 if read.

Figure 17-70. Enable Transfer on Event to System Memory Set 2 (ETESMS2) [offset_TU = 0xC8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMS2-31	ETESMS2-30	ETESMS2-29	ETESMS2-28	ETESMS2-27	ETESMS2-26	ETESMS2-25	ETESMS2-24	ETESMS2-23	ETESMS2-22	ETESMS2-21	ETESMS2-20	ETESMS2-19	ETESMS2-18	ETESMS2-17	ETESMS2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMS2-15	ETESMS2-14	ETESMS2-13	ETESMS2-12	ETESMS2-11	ETESMS2-10	ETESMS2-9	ETESMS2-8	ETESMS2-7	ETESMS2-6	ETESMS2-5	ETESMS2-4	ETESMS2-3	ETESMS2-2	ETESMS2-1	ETESMS2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-59. Enable Transfer on Event to System Memory Set 2 Field Descriptions

Bit	Name	Value	Description
31-0	ETESMS2[31:0]		Enable Transfer on Event to System Memory Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register enables a message buffer transfer on event to the system memory:
		0	Transfer on event is disabled
		1	Transfer on event is enabled

Figure 17-71. Enable Transfer on Event to System Memory Reset 2 (ETESMR2) [offset_TU = 0xCC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMR2-31	ETESMR2-30	ETESMR2-29	ETESMR2-28	ETESMR2-27	ETESMR2-26	ETESMR2-25	ETESMR2-24	ETESMR2-23	ETESMR2-22	ETESMR2-21	ETESMR2-20	ETESMR2-19	ETESMR2-18	ETESMR2-17	ETESMR2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMR2-15	ETESMR2-14	ETESMR2-13	ETESMR2-12	ETESMR2-11	ETESMR2-10	ETESMR2-9	ETESMR2-8	ETESMR2-7	ETESMR2-6	ETESMR2-5	ETESMR2-4	ETESMR2-3	ETESMR2-2	ETESMR2-1	ETESMR2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-60. Enable Transfer on Event to System Memory Reset 2 (ETESMR2) Field Descriptions

Bit	Name	Value	Description
31-0	ETESMR2[31:0]		Enable Transfer on Event to System Memory Reset 2 The ETESMR2 register shows the identical values to ETESMS2 if read.

Figure 17-72. Enable Transfer on Event to System Memory Set 3 (ETESMS3) [offset_TU = 0xD0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMS3-31	ETESMS3-30	ETESMS3-29	ETESMS3-28	ETESMS3-27	ETESMS3-26	ETESMS3-25	ETESMS3-24	ETESMS3-23	ETESMS3-22	ETESMS3-21	ETESMS3-20	ETESMS3-19	ETESMS3-18	ETESMS3-17	ETESMS3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMS3-15	ETESMS3-14	ETESMS3-13	ETESMS3-12	ETESMS3-11	ETESMS3-10	ETESMS3-9	ETESMS3-8	ETESMS3-7	ETESMS3-6	ETESMS3-5	ETESMS3-4	ETESMS3-3	ETESMS3-2	ETESMS3-1	ETESMS3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-61. Enable Transfer on Event to System Memory Set 3 Field Descriptions

Bit	Name	Value	Description
31-0	ETESMS3[31:0]		Enable Transfer on Event to System Memory Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register enables a message buffer transfer on event to the system memory:
		0	Transfer on event is disabled
		1	Transfer on event is enabled

Figure 17-73. Enable Transfer on Event to System Memory Reset 3 (ETESMR3) [offset_TU = 0xD4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMR3-31	ETESMR3-30	ETESMR3-29	ETESMR3-28	ETESMR3-27	ETESMR3-26	ETESMR3-25	ETESMR3-24	ETESMR3-23	ETESMR3-22	ETESMR3-21	ETESMR3-20	ETESMR3-19	ETESMR3-18	ETESMR3-17	ETESMR3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMR3-15	ETESMR3-14	ETESMR3-13	ETESMR3-12	ETESMR3-11	ETESMR3-10	ETESMR3-9	ETESMR3-8	ETESMR3-7	ETESMR3-6	ETESMR3-5	ETESMR3-4	ETESMR3-3	ETESMR3-2	ETESMR3-1	ETESMR3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-62. Enable Transfer on Event to System Memory Reset 3 (ETESMR3) Field Descriptions

Bit	Name	Value	Description
31-0	ETESMR3[31:0]		Enable Transfer on Event to System Memory Reset 3 The ETESMR3 register shows the identical values to ETESMS3 if read.

Figure 17-74. Enable Transfer on Event to System Memory Set 4 (ETESMS4) [offset_TU = 0xD8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMS4-31	ETESMS4-30	ETESMS4-29	ETESMS4-28	ETESMS4-27	ETESMS4-26	ETESMS4-25	ETESMS4-24	ETESMS4-23	ETESMS4-22	ETESMS4-21	ETESMS4-20	ETESMS4-19	ETESMS4-18	ETESMS4-17	ETESMS4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMS4-15	ETESMS4-14	ETESMS4-13	ETESMS4-12	ETESMS4-11	ETESMS4-10	ETESMS4-9	ETESMS4-8	ETESMS4-7	ETESMS4-6	ETESMS4-5	ETESMS4-4	ETESMS4-3	ETESMS4-2	ETESMS4-1	ETESMS4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-63. Enable Transfer on Event to System Memory Set 4 Field Descriptions

Bit	Name	Value	Description
31-0	ETESMS4[31:0]		Enable Transfer on Event to System Memory Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register enables a message buffer transfer on event to the system memory:
		0	Transfer on event is disabled
		1	Transfer on event is enabled

Figure 17-75. Enable Transfer on Event to System Memory Reset 4 (ETESMR4) [offset_TU = 0xDC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETESMR4-31	ETESMR4-30	ETESMR4-29	ETESMR4-28	ETESMR4-27	ETESMR4-26	ETESMR4-25	ETESMR4-24	ETESMR4-23	ETESMR4-22	ETESMR4-21	ETESMR4-20	ETESMR4-19	ETESMR4-18	ETESMR4-17	ETESMR4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETESMR4-15	ETESMR4-14	ETESMR4-13	ETESMR4-12	ETESMR4-11	ETESMR4-10	ETESMR4-9	ETESMR4-8	ETESMR4-7	ETESMR4-6	ETESMR4-5	ETESMR4-4	ETESMR4-3	ETESMR4-2	ETESMR4-1	ETESMR4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-64. Enable Transfer on Event to System Memory Reset 4 (ETESMR4) Field Descriptions

Bit	Name	Value	Description
31-0	ETESMR4[31:0]		Enable Transfer on Event to System Memory Reset 4 The ETESMR4 register shows the identical values to ETESMS4 if read.

17.21.21Clear on Event to System Memory Set/Reset 1/2/3/4 (CESMS/R1-4)

The Clear on Event to System Memory registers disables an enabled transfer on event (enabled in ETESM) after a receive or transmit event. Four 32-bit registers reflect all possible 128 message buffers.

The bits are set by writing ‘1’ to CESMSx and reset by writing ‘1’ to CESMRx. Writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-76. Clear on Event to System Memory Set 1 (CESMS1) [offset_TU = 0xE0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM S1-31	CESM S1-30	CESM S1-29	CESM S1-28	CESM S1-27	CESM S1-26	CESM S1-25	CESM S1-24	CESM S1-23	CESM S1-22	CESM S1-21	CESM S1-20	CESM S1-19	CESM S1-18	CESM S1-17	CESM S1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM S1-15	CESM S1-14	CESM S1-13	CESM S1-12	CESM S1-11	CESM S1-10	CESM S1-9	CESM S1-8	CESM S1-7	CESM S1-6	CESM S1-5	CESM S1-4	CESM S1-3	CESM S1-2	CESM S1-1	CESM S1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-65. Clear on Event to System Memory Set 1 (CESMS1) Field Descriptions

Bit	Name	Value	Description
31-0	CESMS1[31:0]		Clear on Event to System Memory Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register enables an automatic clear of the corresponding ETESM1 bit after a receive or transmit event:
		0	No clear
		1	Activate clear

Figure 17-77. Clear on Event to System Memory Reset 1 (CESMR1) [offset_TU = 0xE4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM R1-31	CESM R1-30	CESM R1-29	CESM R1-28	CESM R1-27	CESM R1-26	CESM R1-25	CESM R1-24	CESM R1-23	CESM R1-22	CESM R1-21	CESM R1-20	CESM R1-19	CESM R1-18	CESM R1-17	CESM R1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM R1-15	CESM R1-14	CESM R1-13	CESM R1-12	CESM R1-11	CESM R1-10	CESM R1-9	CESM R1-8	CESM R1-7	CESM R1-6	CESM R1-5	CESM R1-4	CESM R1-3	CESM R1-2	CESM R1-1	CESM R1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-66. Clear on Event to System Memory Reset 1 (CESMR1) Field Descriptions

Bit	Name	Value	Description
31-0	CESMR1[31:0]		Clear on Event to System Memory Reset 1 The CESMR1 register shows the identical values to CESMS1 if read.

Figure 17-78. Clear on Event to System Memory Set 2 (CESMS2) [offset_TU = 0xE8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM S2-31	CESM S2-30	CESM S2-29	CESM S2-28	CESM S2-27	CESM S2-26	CESM S2-25	CESM S2-24	CESM S2-23	CESM S2-22	CESM S2-21	CESM S2-20	CESM S2-19	CESM S2-18	CESM S2-17	CESM S2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM S2-15	CESM S2-14	CESM S2-13	CESM S2-12	CESM S2-11	CESM S2-10	CESM S2-9	CESM S2-8	CESM S2-7	CESM S2-6	CESM S2-5	CESM S2-4	CESM S2-3	CESM S2-2	CESM S2-1	CESM S2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-67. Clear on Event to System Memory Set 2 (CESMS2) Field Descriptions

Bit	Name	Value	Description
31-0	CESMS2[31:0]		Clear on Event to System Memory Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register enables an automatic clear of the corresponding ETESM2 bit after a receive or transmit event:
		0	No clear
		1	Activate clear

Figure 17-79. Clear on Event to System Memory Reset 2 (CESMR2) [offset_TU = 0xEC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM R2-31	CESM R2-30	CESM R2-29	CESM R2-28	CESM R2-27	CESM R2-26	CESM R2-25	CESM R2-24	CESM R2-23	CESM R2-22	CESM R2-21	CESM R2-20	CESM R2-19	CESM R2-18	CESM R2-17	CESM R2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM R2-15	CESM R2-14	CESM R2-13	CESM R2-12	CESM R2-11	CESM R2-10	CESM R2-9	CESM R2-8	CESM R2-7	CESM R2-6	CESM R2-5	CESM R2-4	CESM R2-3	CESM R2-2	CESM R2-1	CESM R2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-68. Clear on Event to System Memory Reset 2 (CESMR2) Field Descriptions

Bit	Name	Value	Description
31-0	CESMR2[31:0]		Clear on Event to System Memory Reset 2 The CESMR2 register shows the identical values to CESMS2 if read.

Figure 17-80. Clear on Event to System Memory Set 3 (CESMS3) [offset_TU = 0xF0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM S3-31	CESM S3-30	CESM S3-29	CESM S3-28	CESM S3-27	CESM S3-26	CESM S3-25	CESM S3-24	CESM S3-23	CESM S3-22	CESM S3-21	CESM S3-20	CESM S3-19	CESM S3-18	CESM S3-17	CESM S3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM S3-15	CESM S3-14	CESM S3-13	CESM S3-12	CESM S3-11	CESM S3-10	CESM S3-9	CESM S3-8	CESM S3-7	CESM S3-6	CESM S3-5	CESM S3-4	CESM S3-3	CESM S3-2	CESM S3-1	CESM S3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-69. Clear on Event to System Memory Set 3 (CESMS3) Field Descriptions

Bit	Name	Value	Description
31-0	CESMS3[31:0]		Clear on Event to System Memory Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register enables an automatic clear of the corresponding ETESM3 bit after a receive or transmit event:
		0	No clear
		1	Activate clear

Figure 17-81. Clear on Event to System Memory Reset 3 (CESMR3) [offset_TU = 0xF4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM R3-31	CESM R3-30	CESM R3-29	CESM R3-28	CESM R3-27	CESM R3-26	CESM R3-25	CESM R3-24	CESM R3-23	CESM R3-22	CESM R3-21	CESM R3-20	CESM R3-19	CESM R3-18	CESM R3-17	CESM R3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM R3-15	CESM R3-14	CESM R3-13	CESM R3-12	CESM R3-11	CESM R3-10	CESM R3-9	CESM R3-8	CESM R3-7	CESM R3-6	CESM R3-5	CESM R3-4	CESM R3-3	CESM R3-2	CESM R3-1	CESM R3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-70. Clear on Event to System Memory Reset 3 (CESMR3) Field Descriptions

Bit	Name	Value	Description
31-0	CESMR3[31:0]		Clear on Event to System Memory Reset 3 The CESMR3 register shows the identical values to CESMS3 if read.

Figure 17-82. Clear on Event to System Memory Set 4 (CESMS4) [offset_TU = 0xF8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM S4-31	CESM S4-30	CESM S4-29	CESM S4-28	CESM S4-27	CESM S4-26	CESM S4-25	CESM S4-24	CESM S4-23	CESM S4-22	CESM S4-21	CESM S4-20	CESM S4-19	CESM S4-18	CESM S4-17	CESM S4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM S4-15	CESM S4-14	CESM S4-13	CESM S4-12	CESM S4-11	CESM S4-10	CESM S4-9	CESM S4-8	CESM S4-7	CESM S4-6	CESM S4-5	CESM S4-4	CESM S4-3	CESM S4-2	CESM S4-1	CESM S4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-71. Clear on Event to System Memory Set 4 (CESMS4) Field Descriptions

Bit	Name	Value	Description
31-0	CESMS4[31:0]		Clear on Event to System Memory Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register enables an automatic clear of the corresponding ETESM4 bit after a receive or transmit event:
		0	No clear
		1	Activate clear

Figure 17-83. Clear on Event to System Memory Reset 4 (CESMR4) [offset_TU = 0xFC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CESM R4-31	CESM R4-30	CESM R4-29	CESM R4-28	CESM R4-27	CESM R4-26	CESM R4-25	CESM R4-24	CESM R4-23	CESM R4-22	CESM R4-21	CESM R4-20	CESM R4-19	CESM R4-18	CESM R4-17	CESM R4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CESM R4-15	CESM R4-14	CESM R4-13	CESM R4-12	CESM R4-11	CESM R4-10	CESM R4-9	CESM R4-8	CESM R4-7	CESM R4-6	CESM R4-5	CESM R4-4	CESM R4-3	CESM R4-2	CESM R4-1	CESM R4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-72. Clear on Event to System Memory Reset 4 (CESMR4) Field Descriptions

Bit	Name	Value	Description
31-0	CESMR4[31:0]		Clear on Event to System Memory Reset 4 The CESMR4 register shows the identical values to CESMS4 if read.

17.21.22Transfer to System Memory Interrupt Enable Set/Reset 1/2/3/4 (TSMIES/R1-4)

The Transfer to System Memory Interrupt Enable registers enable the interrupt generation on interrupt line TU_Int0, after a transfer to the system memory occurred (flagged in TSMO). Four 32-bit Registers reflect all 128 MB’s.

The bits are set by writing ‘1’ to TSMIESx and reset by writing ‘1’ to TSMIERx. Writing a ‘0’ has no effect. Reading from both addresses will result in the same value.

Figure 17-84. Transfer to System Memory Interrupt Enable Set 1 (TSMIES1) [offset_TU = 0x100]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE S1-31	TSMIE S1-30	TSMIE S1-29	TSMIE S1-28	TSMIE S1-27	TSMIE S1-26	TSMIE S1-25	TSMIE S1-24	TSMIE S1-23	TSMIE S1-22	TSMIE S1-21	TSMIE S1-20	TSMIE S1-19	TSMIE S1-18	TSMIE S1-17	TSMIE S1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE S1-15	TSMIE S1-14	TSMIE S1-13	TSMIE S1-12	TSMIE S1-11	TSMIE S1-10	TSMIE S1-9	TSMIE S1-8	TSMIE S1-7	TSMIE S1-6	TSMIE S1-5	TSMIE S1-4	TSMIE S1-3	TSMIE S1-2	TSMIE S1-1	TSMIE S1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-73. Transfer to System Memory Interrupt Enable Set 1 (TSMIES1) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIES1[31:0]		Transfer to System Memory Interrupt Enable Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register enables a potential interrupt, which occurs if the corresponding TSMO1 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-85. Transfer to System Memory Interrupt Enable Reset 1 (TSMIER1) [offset_TU = 0x104]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE R1-31	TSMIE R1-30	TSMIE R1-29	TSMIE R1-28	TSMIE R1-27	TSMIE R1-26	TSMIE R1-25	TSMIE R1-24	TSMIE R1-23	TSMIE R1-22	TSMIE R1-21	TSMIE R1-20	TSMIE R1-19	TSMIE R1-18	TSMIE R1-17	TSMIE R1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE R1-15	TSMIE R1-14	TSMIE R1-13	TSMIE R1-12	TSMIE R1-11	TSMIE R1-10	TSMIE R1-9	TSMIE R1-8	TSMIE R1-7	TSMIE R1-6	TSMIE R1-5	TSMIE R1-4	TSMIE R1-3	TSMIE R1-2	TSMIE R1-1	TSMIE R1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-74. Transfer to System Memory Interrupt Enable Reset 1 (TSMIER1) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIER1[31:0]		Transfer to System Memory Interrupt Enable Reset 1 The TSMIER1 register shows the identical values to TSMIES1 if read.

Figure 17-86. Transfer to System Memory Interrupt Enable Set 2 (TSMIES2) [offset_TU = 0x108]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE S2-31	TSMIE S2-30	TSMIE S2-29	TSMIE S2-28	TSMIE S2-27	TSMIE S2-26	TSMIE S2-25	TSMIE S2-24	TSMIE S2-23	TSMIE S2-22	TSMIE S2-21	TSMIE S2-20	TSMIE S2-19	TSMIE S2-18	TSMIE S2-17	TSMIE S2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE S2-15	TSMIE S2-14	TSMIE S2-13	TSMIE S2-12	TSMIE S2-11	TSMIE S2-10	TSMIE S2-9	TSMIE S2-8	TSMIE S2-7	TSMIE S2-6	TSMIE S2-5	TSMIE S2-4	TSMIE S2-3	TSMIE S2-2	TSMIE S2-1	TSMIE S2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-75. Transfer to System Memory Interrupt Enable Set 2 (TSMIES2) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIES2[31:0]		Transfer to System Memory Interrupt Enable Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register enables a potential interrupt, which occurs if the corresponding TSMO2 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-87. Transfer to System Memory Interrupt Enable Reset 2 (TSMIER2) [offset_TU = 0x10C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE R2-31	TSMIE R2-30	TSMIE R2-29	TSMIE R2-28	TSMIE R2-27	TSMIE R2-26	TSMIE R2-25	TSMIE R2-24	TSMIE R2-23	TSMIE R2-22	TSMIE R2-21	TSMIE R2-20	TSMIE R2-19	TSMIE R2-18	TSMIE R2-17	TSMIE R2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE R2-15	TSMIE R2-14	TSMIE R2-13	TSMIE R2-12	TSMIE R2-11	TSMIE R2-10	TSMIE R2-9	TSMIE R2-8	TSMIE R2-7	TSMIE R2-6	TSMIE R2-5	TSMIE R2-4	TSMIE R2-3	TSMIE R2-2	TSMIE R2-1	TSMIE R2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-76. Transfer to System Memory Interrupt Enable Reset 2 (TSMIER2) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIER2[31:0]		Transfer to System Memory Interrupt Enable Reset 2 The TSMIER2 register shows the identical values to TSMIES2 if read.

Figure 17-88. Transfer to System Memory Interrupt Enable Set 3 (TSMIES3) [offset_TU = 0x110]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE S3-31	TSMIE S3-30	TSMIE S3-29	TSMIE S3-28	TSMIE S3-27	TSMIE S3-26	TSMIE S3-25	TSMIE S3-24	TSMIE S3-23	TSMIE S3-22	TSMIE S3-21	TSMIE S3-20	TSMIE S3-19	TSMIE S3-18	TSMIE S3-17	TSMIE S3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE S3-15	TSMIE S3-14	TSMIE S3-13	TSMIE S3-12	TSMIE S3-11	TSMIE S3-10	TSMIE S3-9	TSMIE S3-8	TSMIE S3-7	TSMIE S3-6	TSMIE S3-5	TSMIE S3-4	TSMIE S3-3	TSMIE S3-2	TSMIE S3-1	TSMIE S3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-77. Transfer to System Memory Interrupt Enable Set 3 (TSMIES3) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIES3[31:0]		Transfer to System Memory Interrupt Enable Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register enables a potential interrupt, which occurs if the corresponding TSMO3 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-89. Transfer to System Memory Interrupt Enable Reset 3 (TSMIER3) [offset_TU = 0x114]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE R3-31	TSMIE R3-30	TSMIE R3-29	TSMIE R3-28	TSMIE R3-27	TSMIE R3-26	TSMIE R3-25	TSMIE R3-24	TSMIE R3-23	TSMIE R3-22	TSMIE R3-21	TSMIE R3-20	TSMIE R3-19	TSMIE R3-18	TSMIE R3-17	TSMIE R3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE R3-15	TSMIE R3-14	TSMIE R3-13	TSMIE R3-12	TSMIE R3-11	TSMIE R3-10	TSMIE R3-9	TSMIE R3-8	TSMIE R3-7	TSMIE R3-6	TSMIE R3-5	TSMIE R3-4	TSMIE R3-3	TSMIE R3-2	TSMIE R3-1	TSMIE R3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-78. Transfer to System Memory Interrupt Enable Reset 3 (TSMIER3) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIER3[31:0]		Transfer to System Memory Interrupt Enable Reset 3 The TSMIER3 register shows the identical values to TSMIES3 if read.

Figure 17-90. Transfer to System Memory Interrupt Enable Set 4 (TSMIES4) [offset_TU = 0x118]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE S4-31	TSMIE S4-30	TSMIE S4-29	TSMIE S4-28	TSMIE S4-27	TSMIE S4-26	TSMIE S4-25	TSMIE S4-24	TSMIE S4-23	TSMIE S4-22	TSMIE S4-21	TSMIE S4-20	TSMIE S4-19	TSMIE S4-18	TSMIE S4-17	TSMIE S4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE S4-15	TSMIE S4-14	TSMIE S4-13	TSMIE S4-12	TSMIE S4-11	TSMIE S4-10	TSMIE S4-9	TSMIE S4-8	TSMIE S4-7	TSMIE S4-6	TSMIE S4-5	TSMIE S4-4	TSMIE S4-3	TSMIE S4-2	TSMIE S4-1	TSMIE S4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-79. Transfer to System Memory Interrupt Enable Set 4 (TSMIES4) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIES4[31:0]		Transfer to System Memory Interrupt Enable Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register enables a potential interrupt, which occurs if the corresponding TSMO4 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-91. Transfer to System Memory Interrupt Enable Reset 4 (TSMIER4) [offset_TU = 0x11C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSMIE R4-31	TSMIE R4-30	TSMIE R4-29	TSMIE R4-28	TSMIE R4-27	TSMIE R4-26	TSMIE R4-25	TSMIE R4-24	TSMIE R4-23	TSMIE R4-22	TSMIE R4-21	TSMIE R4-20	TSMIE R4-19	TSMIE R4-18	TSMIE R4-17	TSMIE R4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMIE R4-15	TSMIE R4-14	TSMIE R4-13	TSMIE R4-12	TSMIE R4-11	TSMIE R4-10	TSMIE R4-9	TSMIE R4-8	TSMIE R4-7	TSMIE R4-6	TSMIE R4-5	TSMIE R4-4	TSMIE R4-3	TSMIE R4-2	TSMIE R4-1	TSMIE R4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-80. Transfer to System Memory Interrupt Enable Reset 4 (TSMIER4) Field Descriptions

Bit	Name	Value	Description
31-0	TSMIER4[31:0]		Transfer to System Memory Interrupt Enable Reset 4 The TSMIER4 register shows the identical values to TSMIES4 if read.

17.21.23Transfer to Communication Controller Interrupt Enable Set/Reset 1/2/3/4 (TCCIES/R1-4)

The Transfer to Communication Controller Interrupt Enable registers enables the interrupt generation on interrupt line TU_Int0, after a transfer to the communication controller occurred (flagged in TCCO). Four 32-bit Registers reflect all 128 MB's.

The bits are set by writing '1' to TCCIESx and reset by writing '1' to TCCIERx. Writing a '0' has no effect. Reading from both addresses will result in the same value.

Figure 17-92. Transfer to Communication Controller Interrupt Enable Set 1 (TCCIES1) [offset_TU = 0x120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIES1-31	TCCIES1-30	TCCIES1-29	TCCIES1-28	TCCIES1-27	TCCIES1-26	TCCIES1-25	TCCIES1-24	TCCIES1-23	TCCIES1-22	TCCIES1-21	TCCIES1-20	TCCIES1-19	TCCIES1-18	TCCIES1-17	TCCIES1-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIES1-15	TCCIES1-14	TCCIES1-13	TCCIES1-12	TCCIES1-11	TCCIES1-10	TCCIES1-9	TCCIES1-8	TCCIES1-7	TCCIES1-6	TCCIES1-5	TCCIES1-4	TCCIES1-3	TCCIES1-2	TCCIES1-1	TCCIES1-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-81. Transfer to Communication Controller Interrupt Enable Set 1 (TCCIES1) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIES1[31:0]		Transfer to Communication Controller Interrupt Enable Set 1 The register bits 0 to 31 are corresponding to message buffers 0 to 31. Each bit of the register enables a potential interrupt, which occurs if the corresponding TCCO1 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-93. Transfer to Communication Controller Interrupt Enable Reset 1 (TCCIER1) [offset_TU = 0x124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE R1-31	TCCIE R1-30	TCCIE R1-29	TCCIE R1-28	TCCIE R1-27	TCCIE R1-26	TCCIE R1-25	TCCIE R1-24	TCCIE R1-23	TCCIE R1-22	TCCIE R1-21	TCCIE R1-20	TCCIE R1-19	TCCIE R1-18	TCCIE R1-17	TCCIE R1-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE R1-15	TCCIE R1-14	TCCIE R1-13	TCCIE R1-12	TCCIE R1-11	TCCIE R1-10	TCCIE R1-9	TCCIE R1-8	TCCIE R1-7	TCCIE R1-6	TCCIE R1-5	TCCIE R1-4	TCCIE R1-3	TCCIE R1-2	TCCIE R1-1	TCCIE R1-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-82. Transfer to Communication Controller Interrupt Enable Reset 1 (TCCIER1) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIER1[31:0]		Transfer to Communication Controller Interrupt Enable Reset 1 The TCCIER1 register shows the identical values to TCCIES1 if read.

Figure 17-94. Transfer to Communication Controller Interrupt Enable Set 2 (TCCIES2) [offset_TU = 0x128]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE S2-31	TCCIE S2-30	TCCIE S2-29	TCCIE S2-28	TCCIE S2-27	TCCIE S2-26	TCCIE S2-25	TCCIE S2-24	TCCIE S2-23	TCCIE S2-22	TCCIE S2-21	TCCIE S2-20	TCCIE S2-19	TCCIE S2-18	TCCIE S2-17	TCCIE S2-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE S2-15	TCCIE S2-14	TCCIE S2-13	TCCIE S2-12	TCCIE S2-11	TCCIE S2-10	TCCIE S2-9	TCCIE S2-8	TCCIE S2-7	TCCIE S2-6	TCCIE S2-5	TCCIE S2-4	TCCIE S2-3	TCCIE S2-2	TCCIE S2-1	TCCIE S2-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-83. Transfer to Communication Controller Interrupt Enable Set 2 (TCCIES2) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIES2[31:0]		Transfer to Communication Controller Interrupt Enable Set 2 The register bits 0 to 31 are corresponding to message buffers 32 to 63. Each bit of the register enables a potential interrupt, which occurs if the corresponding TCCO2 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-95. Transfer to Communication Controller Interrupt Enable Reset 2 (TCCIER2) [offset_TU = 0x12C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE R2-31	TCCIE R2-30	TCCIE R2-29	TCCIE R2-28	TCCIE R2-27	TCCIE R2-26	TCCIE R2-25	TCCIE R2-24	TCCIE R2-23	TCCIE R2-22	TCCIE R2-21	TCCIE R2-20	TCCIE R2-19	TCCIE R2-18	TCCIE R2-17	TCCIE R2-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE R2-15	TCCIE R2-14	TCCIE R2-13	TCCIE R2-12	TCCIE R2-11	TCCIE R2-10	TCCIE R2-9	TCCIE R2-8	TCCIE R2-7	TCCIE R2-6	TCCIE R2-5	TCCIE R2-4	TCCIE R2-3	TCCIE R2-2	TCCIE R2-1	TCCIE R2-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-84. Transfer to Communication Controller Interrupt Enable Reset 2 (TCCIER2) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIER2[31:0]		Transfer to Communication Controller Interrupt Enable Reset 2 The TCCIER2 register shows the identical values to TCCIES2 if read.

Figure 17-96. Transfer to Communication Controller Interrupt Enable Set 3 (TCCIES3) [offset_TU = 0x130]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE S3-31	TCCIE S3-30	TCCIE S3-29	TCCIE S3-28	TCCIE S3-27	TCCIE S3-26	TCCIE S3-25	TCCIE S3-24	TCCIE S3-23	TCCIE S3-22	TCCIE S3-21	TCCIE S3-20	TCCIE S3-19	TCCIE S3-18	TCCIE S3-17	TCCIE S3-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE S3-15	TCCIE S3-14	TCCIE S3-13	TCCIE S3-12	TCCIE S3-11	TCCIE S3-10	TCCIE S3-9	TCCIE S3-8	TCCIE S3-7	TCCIE S3-6	TCCIE S3-5	TCCIE S3-4	TCCIE S3-3	TCCIE S3-2	TCCIE S3-1	TCCIE S3-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-85. Transfer to Communication Controller Interrupt Enable Set 3 (TCCIES3) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIES3[31:0]		Transfer to Communication Controller Interrupt Enable Set 3 The register bits 0 to 31 are corresponding to message buffers 64 to 95. Each bit of the register enables a potential interrupt, which occurs if the corresponding TCCO3 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-97. Transfer to Communication Controller Interrupt Enable Reset 3 (TCCIER3) [offset_TU = 0x134]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE R3-31	TCCIE R3-30	TCCIE R3-29	TCCIE R3-28	TCCIE R3-27	TCCIE R3-26	TCCIE R3-25	TCCIE R3-24	TCCIE R3-23	TCCIE R3-22	TCCIE R3-21	TCCIE R3-20	TCCIE R3-19	TCCIE R3-18	TCCIE R3-17	TCCIE R3-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE R3-15	TCCIE R3-14	TCCIE R3-13	TCCIE R3-12	TCCIE R3-11	TCCIE R3-10	TCCIE R3-9	TCCIE R3-8	TCCIE R3-7	TCCIE R3-6	TCCIE R3-5	TCCIE R3-4	TCCIE R3-3	TCCIE R3-2	TCCIE R3-1	TCCIE R3-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-86. Transfer to Communication Controller Interrupt Enable Reset 3 (TCCIER3) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIER3[31:0]		Transfer to Communication Controller Interrupt Enable Reset 3 The TCCIER3 register shows the identical values to TCCIES3 if read.

Figure 17-98. Transfer to Communication Controller Interrupt Enable Set 4 (TCCIES4) [offset_TU = 0x138]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE S4-31	TCCIE S4-30	TCCIE S4-29	TCCIE S4-28	TCCIE S4-27	TCCIE S4-26	TCCIE S4-25	TCCIE S4-24	TCCIE S4-23	TCCIE S4-22	TCCIE S4-21	TCCIE S4-20	TCCIE S4-19	TCCIE S4-18	TCCIE S4-17	TCCIE S4-16
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE S4-15	TCCIE S4-14	TCCIE S4-13	TCCIE S4-12	TCCIE S4-11	TCCIE S4-10	TCCIE S4-9	TCCIE S4-8	TCCIE S4-7	TCCIE S4-6	TCCIE S4-5	TCCIE S4-4	TCCIE S4-3	TCCIE S4-2	TCCIE S4-1	TCCIE S4-0
RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0	RS-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-87. Transfer to Communication Controller Interrupt Enable Set 4 (TCCIES4) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIES4[31:0]		Transfer to Communication Controller Interrupt Enable Set 4 The register bits 0 to 31 are corresponding to message buffers 96 to 127. Each bit of the register enables a potential interrupt, which occurs if the corresponding TCCO4 bit is set:
		0	No interrupt
		1	Interrupt is generated

Figure 17-99. Transfer to Communication Controller Interrupt Enable Reset 4 (TCCIER4) [offset_TU = 0x13C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCIE R4-31	TCCIE R4-30	TCCIE R4-29	TCCIE R4-28	TCCIE R4-27	TCCIE R4-26	TCCIE R4-25	TCCIE R4-24	TCCIE R4-23	TCCIE R4-22	TCCIE R4-21	TCCIE R4-20	TCCIE R4-19	TCCIE R4-18	TCCIE R4-17	TCCIE R4-16
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCCIE R4-15	TCCIE R4-14	TCCIE R4-13	TCCIE R4-12	TCCIE R4-11	TCCIE R4-10	TCCIE R4-9	TCCIE R4-8	TCCIE R4-7	TCCIE R4-6	TCCIE R4-5	TCCIE R4-4	TCCIE R4-3	TCCIE R4-2	TCCIE R4-1	TCCIE R4-0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after reset

Table 17-88. Transfer to Communication Controller Interrupt Enable Reset 4 (TCCIER4) Field Descriptions

Bit	Name	Value	Description
31-0	TCCIER4[31:0]		Transfer to Communication Controller Interrupt Enable Reset 4 The TCCIER4 register shows the identical values to TCCIES4 if read.

17.21.24Transfer Configuration RAM (TCR)

The TCR consists of 128 entries, each 19 bit wide. The TCR is parity protected. The parity protection can be switched on/off by the 4-bit key (PAL[3:0]) in the Global Control (GC) register.

Figure 17-100. Transfer Configuration RAM (TCR) [offset_TU_RAM = 0x0000 - 0x01FF]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved													STXR	THTSM	TPTSM
R-0													RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THTCC	TPTCC	TSO13	TSO12	TSO11	TSO10	TSO9	TSO8	TSO7	TSO6	TSO5	TSO4	TSO3	TSO2	TSO1	TSO0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after memory initialization

Table 17-89. Transfer Configuration RAM (TCR) Field Descriptions

Bit	Name	Value	Description
31-19	Reserved		Reads return zero and writes have no effect.
18	STXR		Set Transmit Request Control set/reset of buffer transmit requests in the communication controller.
		0	Transfer Unit State Machine will set IBCM.STXRH to ’0’ during a transfer to the communication controller.
		1	Transfer Unit State Machine will set IBCM.STXRH to ’1’ during a transfer to the communication controller.
17	THTSM		Transfer Header to System Memory
		0	Transfer Unit State Machine will not transfer buffer header to system memory.
		1	Transfer Unit State Machine will transfer buffer header to system memory.
16	TPTSM		Transfer Payload to System Memory
		0	Transfer Unit State Machine will not transfer buffer payload to system memory.
		1	Transfer Unit State Machine will transfer buffer payload to system memory.
15	THTCC		Transfer Header to Communication Controller
		0	Transfer Unit State Machine will not transfer buffer header to the communication controller.
		1	Transfer Unit State Machine will transfer buffer header to the communication controller.

Table 17-89. Transfer Configuration RAM (TCR) Field Descriptions (Continued)

Bit	Name	Value	Description
14	TPTCC		Transfer Payload to Communication Controller
		0	Transfer Unit State Machine will not transfer buffer payload to the communication controller.
		1	Transfer Unit State Machine will transfer buffer payload to the communication controller.
13-0	TSO[13:0]		Transfer Start Offset 14-bit buffer address offset in system memory. The resulting address in system memory is computed by adding the 32-bit aligned buffer address offset (TSO[13:0] = buffer address offset bits 15:2) to the base address defined in the TBA register.

17.21.25TCR Parity Test Mode

In parity test mode (PFT bit is set in Global Control Register (GC)) the parity information are visible and can be read or written. The corresponding TCR entry can be found by subtracting 0x200 from the TCR offset.

Figure 17-101. Parity Information in TCR Parity Test Mode [offset_TU_RAM = 0x0200 - 0x03FF]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															PAB2
R-0															RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved							PAB1	reserved							PAB0
R-0							RW-0	R-0							RW-0

R = Read; W = Write; S = Set; C = Clear; U = Undefined; -n = Value after memory initialization

Table 17-90. Transfer Configuration RAM (TCR) Field Descriptions

Bit	Name	Value	Description
31-17	Reserved		Reads return zero and writes have no effect.
16	PAB2		Parity Bit for TCRx Byte 2 Parity information for byte 2 of TCRx[18:16]. Reserved TCR bits are ignored for parity calculation.
15-9	Reserved		Reads return zero and writes have no effect.
8	PAB1		Parity Bit for TCRx Byte 1 Parity information for byte 1 of TCRx[15:8].
7-1	Reserved		Reads return zero and writes have no effect.
0	PAB0		Parity Bit for Byte 0

17.22 Communication Controller Register Map

The FlexRay module allocates an address space of 2 Kbytes (0x0000 to 0x07FF). The registers are organized as 32-bit registers. 8/16-bit accesses are also supported. CPU access to the message RAM is done via the input and output buffers. They buffer data to be transferred to and from the message RAM under control of the message handler, avoiding conflicts between CPU accesses and message reception / transmission.

The test registers located on address 0x0010 and 0x0014 are writable only under the conditions described in section 17.23.1.

The assignment of the message buffers is done according to the scheme shown in Table 17-91 below. The number N of available message buffers depends on the payload length of the configured message buffers. The maximum number of message buffers is 128. The maximum payload length supported is 254 bytes.

The message buffers are separated into three consecutive groups; see Figure 17-102:

- Static buffers- Transmit / receive buffers assigned to static segment
- Static + Dynamic buffers- Transmit / receive buffers assigned to static or dynamic segment
- FIFO- Receive FIFO

The message buffer separation configuration can be changed in DEFAULT_CONFIG or CONFIG state only by programming register MRC (see Chapter 17.26.1, Message RAM Configuration).

The first group starts with message buffer 0 and consists of static message buffers only. message buffer 0 is dedicated to hold the startup / sync frame or the single slot frame, if the node transmits one, as configured by SUCC1.TXST, SUCC1.TXSY, and SUCC1.TSM. In addition, message buffer 1 may be used for sync frame transmission in case that sync frames or single-slot frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to '1' and message buffers 0 and 1 have to be configured with the key slot ID and can be (re)configured in DEFAULT_CONFIG or CONFIG state only.

The second group consists of message buffers assigned to the static or to the dynamic segment. Message buffers belonging to this group may be reconfigured during run time from dynamic to static or vice versa depending on the state of MRC.SEC[1:0].

The message buffers belonging to the third group are concatenated to a single receive FIFO.

Figure 17-102. Message Buffer Assignment

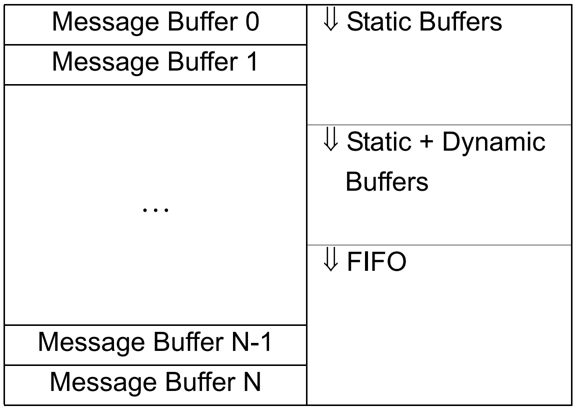


Table 17-91. Communication Controller Register Map

Address	Symbol	Name	Reset	Acc
0x0000		reserved	0000 0000	r
0x0004		reserved	0000 0000	r
0x0008		reserved	0000 0000	r
0x000C		reserved	0000 0000	r
Special registers (see section 17.23.1)				
0x0010	TEST1	Test register 1	0000 0300	r/w
0x0014	TEST2	Test register 2	0000 0000	r/w
0x0018		reserved (1)	0000 0000	r
0x001C	LCK	Lock register	0000 0000	r/w
Interrupt registers (see section 17.23.2)				
0x0020	EIR	Error interrupt register	0000 0000	r/w
0x0024	SIR	Status interrupt register	0000 0000	r/w
0x0028	EILS	Error interrupt line select	0000 0000	r/w
0x002C	SILS	Status interrupt line select	0303 FFFF	r/w
0x0030	EIES	Error interrupt enable set	0000 0000	r/w
0x0034	EIER	Error interrupt enable reset	0000 0000	r/w
0x0038	SIES	Status interrupt enable set	0000 0000	r/w
0x003C	SIER	Status interrupt enable reset	0000 0000	r/w
0x0040	ILE	Interrupt line enable	0000 0000	r/w
0x0044	T0C	Timer 0 configuration	0000 0000	r/w
0x0048	T1C	Timer 1 configuration	0002 0000	r/w
0x004C	STPW1	Stop watch register1	0000 0000	r/w
0x0050	STPW2	Stop watch register2	0000 0000	r/w
0x0054 - 0x007C		reserved (11)	0000 0000	r
communication controller control registers (see section 17.24)				
0x0080	SUCC1	SUC configuration register 1	0C40 1080[a] 0C40 1000	r/w
0x0084	SUCC2	SUC configuration register 2	0100 0504	r/w
0x0088	SUCC3	SUC configuration register 3	0000 0011	r/w
0x008C	NEMC	NEM configuration register	0000 0000	r/w
0x0090	PRTC1	PRT configuration register 1	084C 0633	r/w
0x0094	PRTC2	PRT configuration register 2	0F2D 0A0E	r/w
0x0098	MHDC	MHD configuration register	0000 0000	r/w
0x009C		reserved (1)	0000 0000	r
0x00A0	GTUC1	GTU configuration register 1	0000 0280	r/w
0x00A4	GTUC2	GTU configuration register 2	0002 000A	r/w

Table 17-91. Communication Controller Register Map (Continued)

Address	Symbol	Name	Reset	Acc
0x00A8	GTUC3	GTU configuration register 3	0202 0000	r/w
0x00AC	GTUC4	GTU configuration register 4	0008 0007	r/w
0x00B0	GTUC5	GTU configuration register 5	0E00 0000	r/w
0x00B4	GTUC6	GTU configuration register 6	0002 0000	r/w
0x00B8	GTUC7	GTU configuration register 7	0002 0004	r/w
0x00BC	GTUC8	GTU configuration register 8	0000 0002	r/w
0x00C0	GTUC9	GTU configuration register 9	0000 0101	r/w
0x00C4	GTUC10	GTU configuration register 10	0002 0005	r/w
0x00C8	GTUC11	GTU configuration register 11	0000 0000	r/w
0x00CC -0x00FC		reserved (13)	0000 0000	r
communication controller status registers (see section 17.25)				
0x0100	CCSV	communication controller status vector	0010 4000	r
0x0104	CCEV	communication controller error vector	0000 0000	r
0x0108 - 0x010C		reserved (2)	0000 0000	r
0x0110	SCV	Slot counter value	0000 0000	r
0x0114	MTCCV	Macrotick and cycle counter value	0000 0000	r
0x0118	RCV	Rate correction value	0000 0000	r
0x011C	OCV	Offset correction value	0000 0000	r
0x0120	SFS	Sync frame status	0000 0000	r
0x0124	SWNIT	Symbol window and NIT status	0000 0000	r
0x0128	ACS	Aggregated channel status	0000 0000	r/w
0x012C		reserved (1)	0000 0000	r
0x0130 - 0x0168	ESIDn	Even sync ID [1...15]	0000 0000	r
0x016C		reserved (1)	0000 0000	r
0x0170 - 0x01A8	OSIDn	Odd sync ID [1...15]	0000 0000	r
0x01AC		reserved (1)	0000 0000	r
0x01B0 - 0x01B8	NMVn	Network management vector [1...3]	0000 0000	r
0x01BC - 0x02FC		reserved (81)	0000 0000	r
Message buffer control registers (see section 17.26)				
0x0300	MRC	Message RAM configuration	0180 0000	r/w
0x0304	FRF	FIFO rejection filter	0180 0000	r/w
0x0308	FRFM	FIFO rejection filter mask	0000 0000	r/w
0x030C	FCL	FIFO Critical Level	0000 0080	r/w

Table 17-91. Communication Controller Register Map (Continued)

Address	Symbol	Name	Reset	Acc
Message buffer status registers (see section 17.27)				
0x0310	MHDS	Message handler status	0000 0080[a] 0000 0000	r/w
0x0314	LDTS	Last dynamic transmit slot	0000 0000	r
0x0318	FSR	FIFO status register	0000 0000	r
0x031C	MHDF	Message handler constraint flags	0000 0000	r/w
0x0320	TXRQ1	Transmission request 1	0000 0000	r
0x0324	TXRQ2	Transmission request 2	0000 0000	r
0x0328	TXRQ3	Transmission request 3	0000 0000	r
0x032C	TXRQ4	Transmission request 4	0000 0000	r
0x0330	NDAT1	New data 1	0000 0000	r
0x0334	NDAT2	New data 2	0000 0000	r
0x0338	NDAT3	New data 3	0000 0000	r
0x033C	NDAT4	New data 4	0000 0000	r
0x0340	MBSC1	Message buffer status changed 1	0000 0000	r
0x0344	MBSC2	Message buffer status changed 2	0000 0000	r
0x0348	MBSC3	Message buffer status changed 3	0000 0000	r
0x034C	MBSC4	Message buffer status changed 4	0000 0000	r
0x0350 - 0x03EC		reserved (40)	0000 0000	r
Identification register (see section 17.28)				
0x03F0	CREL	Core release register	[release info]	r
0x03F4	ENDN	Endian register	8765 4321	r
0x03F8 - 0x03FC		reserved (2)	0000 0000	r
Input buffer (see section 17.29)				
0x0400 - 0x04FC	WRDSn	Write data section [1...64]	0000 0000	r/w
0x0500	WRHS1	Write header section 1	0000 0000	r/w
0x0504	WRHS2	Write header section 2	0000 0000	r/w
0x0508	WRHS3	Write header section 3	0000 0000	r/w
0x050C		reserved (1)	0000 0000	r/w
0x0510	IBCM	Input buffer command mask	0000 0000	r/w
0x0514	IBCR	Input buffer command request	0000 0000	r/w
0x0518 - 0x05FC		reserved (58)	0000 0000	r
Output buffer (see section 17.30)				
0x0600 - 0x06FC	RDDSn	Read data section [1...64]	0000 0000	r

Table 17-91. Communication Controller Register Map (Continued)

Address	Symbol	Name	Reset	Acc
0x0700	RDHS1	Read header section 1	0000 0000	r
0x0704	RDHS2	Read header section 2	0000 0000	r
0x0708	RDHS3	Read header section 3	0000 0000	r
0x070C	MBS	Message buffer status	0000 0000	r
0x0710	OBCM	Output buffer command mask	0000 0000	r/w
0x0714	OBCR	Output buffer command request	0000 0000	r/w
0x0718 – 0x07FC		reserved (58)	0000 0000	r

a. during initialization of internal RAM blocks

Table 17-92 shows a summary of the communication controller registers.

Table 17-92. Communication Controller Register Summary

<table>
<tr><th>Offset
Address[1]
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td colspan="15"></td><td colspan="2"></td></tr>
<tr><td rowspan="2">0x10
TEST1
Page 1106</td><td colspan="4">CERB(3-0)</td><td colspan="4">CERA(3-0)</td><td colspan="2">Reserved</td><td>TXENB</td><td>TXENA</td><td>TXB</td><td>TXA</td><td>RXB</td><td>RXA</td></tr>
<tr><td colspan="6">Reserved</td><td>TMC1</td><td>TMC0</td><td colspan="2">Reserved</td><td colspan="2">TMC(1–0)</td><td colspan="2">Reserved</td><td>ELBE</td><td>WRTEN</td></tr>
<tr><td rowspan="2">0x14
TEST2
Page 1111</td><td colspan="16">Reserved</td></tr>
<tr><td>RDPB</td><td>WRPB</td><td colspan="7">Reserved</td><td colspan="3">SSEL2(3–0)</td><td>Reserved</td><td colspan="3">RS(2–0)</td></tr>
<tr><td rowspan="2">0x1C
LCK
Page 1114</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="8">TMK(7–0)</td><td colspan="8">CLK(7–0)</td></tr>
<tr><td rowspan="2">0x20
EIR
Page 1115</td><td colspan="5">Reserved</td><td>TABB</td><td>LTVB</td><td>EDB</td><td colspan="5">Reserved</td><td>TABA</td><td>LTVA</td><td>EDA</td></tr>
<tr><td colspan="4">Reserved</td><td>MHF</td><td>IOBA</td><td>IIBA</td><td>EFA</td><td>RFO</td><td>PERR</td><td>CCL</td><td>CCF</td><td>SFO</td><td>SFBM</td><td>CNA</td><td>PEMC</td></tr>
<tr><td rowspan="2">0x24
SIR
Page 1119</td><td colspan="6">Reserved</td><td>MTSB</td><td>WUPB</td><td colspan="6">Reserved</td><td>MTSA</td><td>WUPA</td></tr>
<tr><td>SDS</td><td>MBSI</td><td>SUCS</td><td>SWE</td><td>TOBC</td><td>TIBC</td><td>TI1</td><td>TI0</td><td>NMVC</td><td>RFCL</td><td>RFNE</td><td>RXI</td><td>TXI</td><td>CYCS</td><td>CAS</td><td>WST</td></tr>
</table>

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x28 EILS Page 1123	Reserved					TABBL	LTVBL	EDBL	Reserved					TABAL	LTVAL	EDAL
	Reserved				MHFL	IOBAL	IIBAL	EFAL	RFOL	PERRL		CCFL	SFOL	SFBML	CNAL	PEMCL
0x2C SILS Page 1126	Reserved						MTSBL	WUPBL	Reserved						MTSAL	WUPAL
	SDSL	MBSIL	SUCSL	SWEL	TOBCL	TIBCL	TI1L	TI0L	NMVCL	RFFL	RFNEL	RXIL	TXIL	CYCSL	CASL	WSTL
0x30 EIES 34h EIER Page 1129	Reserved					TABBE	LTVBE	EDBE	Reserved					TABAE	LTVAE	EDAE
	Reserved				MHFE	IOBAE	IIBAE	EFAE	RFOE	PERRE	CCLE	CCFE	SFOE	SFBMB	CNAE	PEMCE
0x38 SIES 3C SIER Page 1132	Reserved							MTSBE	WUPBE	Reserved					MTSAE	WUPAE
	SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCB	RFFE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTF
0x40 ILE Page 1135	Reserved															
	Reserved														EINT(1–0)	
0x44 T0C Page 1136	Reserved		T0MO(13–0)													
	Reserved	T0CC(6–0)							Reserved						T0MS	T0RC
0x48 T1C Page 1137	Reserved		T1MO(13–0)													
	Reserved														T1MS	T1RC
0x4C STPW1 Page 1138	Reserved		SMTV(13–0)													
	Reserved		SCCV(5–0)						Reserved	EINT1	EINT0	EETP	SSWT	EDGE	SWMS	ESWT

Table 17-92. Communication Controller Register Summary (Continued)

<table>
<tr><th>Offset
Address[1]
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x50
STPW2
Page 1138</td><td colspan="5">Reserved</td><td colspan="11">SSCVB(10–0)</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="11">SSCVA(10:0)</td></tr>
<tr><td rowspan="2">0x80
SUCC1
Page 1141</td><td colspan="4"></td><td>CCHB*</td><td>CCHA*</td><td>MTSB*</td><td>MTSA*</td><td>HCSE*</td><td>TSM*</td><td>WUCS
</td><td colspan="5">PTA(4–0)</td></tr>
<tr><td colspan="5">CSA(4–0)*</td><td></td><td>TXSY*</td><td>TXST*</td><td>PBSY</td><td colspan="3"></td><td colspan="4">CMD(3–0)</td></tr>
<tr><td rowspan="2">0x84
SUCC2
Page 1148</td><td colspan="4">Reserved</td><td colspan="4">LTN(3–0)*</td><td colspan="3">Reserved</td><td colspan="5">LT(20–16)*</td></tr>
<tr><td colspan="16">LT1(5–0)*</td></tr>
<tr><td rowspan="2">0x88
SUCC3
Page 1149</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">WCF(3–0)*</td><td colspan="4">WCP(3–0)*</td></tr>
<tr><td rowspan="2">0x8C
NEMC
Page 1150</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">NML(3–0)*</td></tr>
<tr><td rowspan="2">0x90
PRTC1
Page 1151</td><td colspan="6">RWP(5–0)*</td><td>Reserv
ed</td><td colspan="9">RXW(8–0)*</td></tr>
<tr><td colspan="2">BRP(1–0)*</td><td colspan="2">SPP(1:0)</td><td>Reserv
ed</td><td>CASM
6</td><td colspan="6">CASM(5:0)</td><td colspan="4">TSST(3–0)*</td></tr>
<tr><td rowspan="2">0x94
PRTC2
Page 1153</td><td colspan="2">Reserved</td><td colspan="6">TXL(5–0)*</td><td colspan="8">TX(17–10)*</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="6">RXL(5–0)*</td><td colspan="2">Reserved</td><td colspan="6">RX(15–10)*</td></tr>
<tr><td rowspan="2">0x98
MHDC
Page 1154</td><td colspan="3">Reserved</td><td colspan="13">SLT(12–0)*</td></tr>
<tr><td colspan="9">Reserved</td><td colspan="7">SFDL(6–0)*</td></tr>
</table>

*All bits marked with an asterisk can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-92. Communication Controller Register Summary (Continued)

<table>
<tr><th>Offset
Address(1)
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0xA0
GTUC1
Page 1155</td><td colspan="12">Reserved</td><td colspan="4">UT(19–16)*</td></tr>
<tr><td colspan="16">UT(15–0)*</td></tr>
<tr><td rowspan="2">0xA4
GTUC2
Page 1156</td><td colspan="12">Reserved</td><td colspan="4">SNM(3–0)*</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">MPC(13–0)*</td></tr>
<tr><td rowspan="2">0xA8
GTUC3
Page 1157</td><td>Reserved</td><td colspan="7">MIOB(6:0)</td><td>Reserved</td><td colspan="7">MTIO(6–0)*</td></tr>
<tr><td colspan="8">UIOB(7–0)*</td><td colspan="8">UIOA(7–0)*</td></tr>
<tr><td rowspan="2">0xAC
GTUC4
Page 1158</td><td colspan="2">Reserved</td><td colspan="14">OCS(13–0)*</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">NIT(13–0)*</td></tr>
<tr><td rowspan="2">0xB0
GTUC5
Page 1159</td><td colspan="8">DEC(7–0)*</td><td colspan="3">Reserved</td><td colspan="5">CDD(4–0)*</td></tr>
<tr><td colspan="8">DCB(7–0)*</td><td colspan="8">DCA(7–0)*</td></tr>
<tr><td rowspan="2">0xB4
GTUC6
Page 1160</td><td colspan="5">Reserved</td><td colspan="11">MOD(10–0)*</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="11">ASR(10–0)*</td></tr>
<tr><td rowspan="2">0xB8
GTUC7
Page 1161</td><td colspan="6">Reserved</td><td colspan="10">NSS(9–0)*</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">SSL(9:0)</td></tr>
<tr><td rowspan="2">0xBC
GTUC8
Page 1162</td><td colspan="3">Reserved</td><td colspan="13">NMS(12–0)*</td></tr>
<tr><td colspan="10">Reserved</td><td colspan="6">MSL(5–0)*</td></tr>
</table>

Table 17-92. Communication Controller Register Summary (Continued)

<table>
<tr><th>Offset
Address[1]
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0xC0
GTUC9
Page 1163</td><td colspan="14">Reserved</td><td colspan="2">DSI(1–0)*</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="5">MAP0(4–0)*</td><td colspan="2">Reserved</td><td colspan="6">APO(5–0)*</td></tr>
<tr><td rowspan="2">0xC4
GTUC10
Page 1164</td><td colspan="5">Reserved</td><td colspan="11">MRC(10–0)*</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">MOC(13–0)*</td></tr>
<tr><td rowspan="2">0xC8
GTUC11
Page 1165</td><td colspan="5">Reserved</td><td colspan="3">ERC(2–0)*</td><td colspan="5">Reserved</td><td colspan="3">EOC(2–0)*</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="2">ERCC(1:0)</td><td colspan="6">Reserved</td><td colspan="2">ECC(1–0)</td></tr>
<tr><td rowspan="2">0x100
CCSV
Page 1167</td><td colspan="2">Reserved</td><td colspan="6">PSL(5:0)</td><td colspan="5">RCA(4–0)</td><td colspan="3">WSV(2–0)</td></tr>
<tr><td>Reserv ed</td><td>CSI</td><td>CSAI</td><td>CSNI</td><td colspan="2">Reserved</td><td colspan="2">SLM(1–0)</td><td>HRQ</td><td>FSI</td><td colspan="6">POCS(5–0)</td></tr>
<tr><td rowspan="2">0x104
CCEV
Page 1171</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="5">PTAC(4–0)</td><td colspan="2">ERRM(1–0)</td><td colspan="2">Reserved</td><td colspan="4">CCFC(3–0)</td></tr>
<tr><td rowspan="2">0x110
SCV
Page 1172</td><td colspan="5">Reserved</td><td colspan="11">SCCB1(10–0)</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="11">SCCA1(10–0)</td></tr>
<tr><td rowspan="2">0x114
MTCC
Page 1173</td><td colspan="10">Reserved</td><td colspan="6">CCV(5–0)</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">MTV(13–0)</td></tr>
<tr><td rowspan="2">0x118
RCV
Page 1174</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="11">RCV(11–0)</td></tr>
</table>

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x11C OCV Page 1175	Reserved												OCV(19–16)			
	OCV(15–0)															
0x120 SFS Page 1176	Reserved												RCLR	MRCS	OCLR	MOCS
	VSB0(3–0)				VSBE(3–0)				VSA0(3–0)				VSAE(3–0)			
0x124 SWNIT Page 1178	Reserved															
	Reserved				SPNB	SENB	SBNA	SENA	MTSB	MTSA	TCSB	SBSB	SESB	TCSA	SBSA	SESA
0x128 ACS Page 1180	Reserved															
	Reserved			SBVB	CIB	CEDB	SEDB	VFRB	Reserved			SBVA	CIA	CEDA	SEDA	VFRA
0x130–0x168 ESIDn Page 1183	Reserved															
	RXEB	RXEA	Reserved				EID(9–0)									
0x170–0x1A8 OSIDn Page 1184	Reserved															
	RXOB	RXOA	Reserved				OID(9–0)									
0x1B0–0x1B8 NMVn Page 1185	NM(31–16)															
	NM(15–0)															
0x300 MRC Page 1186	Reserved					SPLM	SEC(1:0)		LCB(7–0)							
	FFB(7–0)								FDB(7–0)							

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address(1) 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
0x304 FRF Page 1189	Reserved							RNF*	RSS*	CYF(6–0)						
	Reserved			FID(10–0)											CH(1–0)	
0x308 FRFM Page 1191	Reserved															
	Reserved			MFID(10–0)												
0x30C FCL Page 1192	Reserved															
	Reserved									CL(7–0)						
0x310 MHDS Page 1193	Reserved	MBU(6–0)							Reserved	MBT(6–0)						
	Reserved	FMB(6–0)							CRAM	MFMB	FMBD	PTBF2	PTBF1	PMR	POBF	PIBF
0x314 LDTS Page 1195	Reserved					LDTB(10:0)										
	Reserved					LDTA(10:0)										
0x318 FSR Page 1196	Reserved															
	RFFL(7:0)								Reserved					RFO	RFCL	RFNE
0x31C MHDF Page 1198	Reserved															
								WAHP	Reserved		TBFB	TBFA	FNFB	FNFA	SNUB	SNUA
0x320 TX4RQ1 Page 1201	TXR[31:16]															
	TXR[15:0]															

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x324 TX4RQ2 Page 1201	TXR[63:48]															
	TXR[47:32]															
0x328 TX4RQ3 Page 1201	TXR[95:80]															
	TXR[79:64]															
0x32C TX4RQ4 Page 1201	TXR[127:112]															
	TXR[111:96]															
0x330 NDAT1 Page 1203	ND(31–16)															
	ND[15:0]															
0x334 NDAT2 Page 1203	ND[63:48]															
	ND[47:32]															
0x338 NDAT3 Page 1203	ND[95:80]															
	ND[79:64]															
0x33C NDAT4 Page 1203	ND[127:112]															
	ND[111:96]															
0x340 MBSC1 Page 1205	MBS[31:16]															
	MBS[15:0]															

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x344 MBSC2 Page 1205	MBS[63:48]															
	MBS[47:32]															
0x348 MBSC3 Page 1205	MBS[95:80]															
	MBS[79:64]															
0x34C MBSC4 Page 1205	MBS[127:112]															
	MBS[111:96]															
0x3F0 CREL Page 1207	REL(3:0)				STEP(7:0)								YEAR(3:0)			
	MON(7:0)								DAY(7:0)							
0x3F4 ENDN Page 1208	ETV(31:16)															
	ETV(15:0)															
0x400–0x4FC WRDSn Page 1209	MD[31:16]															
	MD[15:0]															
0x500 WRHS Page 1210	Reserved		MBI	TXM	PPIT	CFG	CHB	CHA	Reserved	CYC(6–0)						
	Reserved					FID(10–0)										
0x504 WRHS2 Page 1212	Reserved									PLC(6–0)						
	Reserved					CRC(10–0)										

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x508 WRHS3 Page 1213	Reserved															
	Reserved					DP(10–0)*										
0x510 IBCM Page 1214	Reserved													STXRS	LDSS	LHSS
	Reserved													STXRH	LDSH	LHSH
0x514 IBCR Page 1216	IBSYS	Reserved								IBRS(6–0)						
	IBSYH	Reserved								IBRH(6–0)						
0x600–0x6FC RDDSn Page 1218	MD(31–16)															
	MD(15–0)															
0x700 RDHS1 Page 1219	Reserved		MBI	TXM	PPIT	CFG	CHB	CHA	Reserved	CYC(6–0)						
	Reserved					FID(10–0)										
0x704 RDHS2 Page 1221	Reserved	PLR(6–0)							Reserved	PLC(6–0)						
	Reserved					CRC(10–0)										
0x708 RDHS3 Page 1223	Reserved		RES	PPI	NFI	SYN	SFI	RCI	Reserved		RCC(5–0)					
	Reserved					DP(10–0)										
0x70C MBS Page 1225	Reserved		RESS	PPIS	NFIS	SYNS	SFIS	RCTS	Reserved		CCS(5:0)					
	FTB	FTA	Reserved	MLST	ESB	ESA	TCIB	TCIA	SVOB	SVOA	CEOB	CEOA	SEOB	SEOA	VFRB	VFRA

Table 17-92. Communication Controller Register Summary (Continued)

Offset Address[1] 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
0x710 OBCM Page 1229	Reserved														RDSH	RHSH
	Reserved														RDSS	RHSS
0z714 OBCR Page 1231	Reserved									OBRH(6–0)						
	OBSYS	Reserved					REQ	VIEW	Reserved	OBRS(6–0)						

17.23 Communication Controller Registers

17.23.1 Special Registers

17.23.1.1 Test Register 1 (TEST1)

Test register 1 holds the control bits to configure the test modes of the FlexRay module. Write access to these bits is only possible if the WRTEN bit is set.

Figure 17-103. Test Register1 (TEST1) [offset_CC = 0x10]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="4">CERB(3:0)</td><td colspan="4">CERA(3:0)</td><td colspan="2">Reserved</td><td>TXENB</td><td>TXENA</td><td>TXB</td><td>TXA</td><td>RXB</td><td>RXA</td></tr>
<tr><td colspan="4">R-0</td><td colspan="4">R-0</td><td>R-0</td><td>R-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R-0</td><td>R-0</td></tr>
</table>

<table>
<tr><td>15</td><td></td><td></td><td></td><td></td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td>AOB</td><td>AOA</td><td colspan="2">Reserved</td><td colspan="2">TMC(1–0)</td><td colspan="2">Reserved</td><td>ELBE</td><td>WRT EN</td></tr>
<tr><td colspan="6">R-0</td><td colspan="2">R/W-1</td><td colspan="2">R-0</td><td colspan="2">R/W-0</td><td colspan="2">R-0</td><td>R/W-0</td><td>R/W-0</td></tr>
</table>

R = Read, W = Write; -*n* = Value after reset

Figure 17-104. Test Register1 (TEST1) Field Descriptions

Bit	Name	Value	Description
31-28	CERB(3:0)		Coding Error Report Channel B. Set when a coding error is detected on channel B. Reset to zero when register TEST1 is read or written. Once the CERB[3:0] is set it will remain unchanged until the Host accesses the TEST1 register
		0	No coding error detected.
		1	Header CRC error detected.
		2	Frame CRC error detected.
		3	Frame Start Sequence FSS too long.
		4	First bit of Byte Start Sequence BSS seen LOW.
		5	Second bit of Byte Start Sequence BSS seen HIGH.
		6	First bit of Frame End Sequence FES seen HIGH.
		7	Second bit of Frame End Sequence FES seen LOW.
		8	CAS / MTS symbol seen too short.
		9	CAS / MTS symbol seen too long.
		10 - 15	reserved.

Figure 17-104. Test Register1 (TEST1) Field Descriptions (Continued)

Bit	Name	Value	Description
27-24	CERA(3:0)		Coding Error Report Channel A. Set when a coding error is detected on channel A. Reset to zero when register TEST1 is read or written. Once the CERA[3:0] is set it will remain unchanged until the Host accesses the TEST1 register
		0	No coding error detected.
		1	Header CRC error detected.
		2	Frame CRC error detected.
		3	Frame Start Sequence FSS too long.
		4	First bit of Byte Start Sequence BSS seen LOW.
		5	Second bit of Byte Start Sequence BSS seen HIGH.
		6	First bit of Frame End Sequence FES seen HIGH.
		7	Second bit of Frame End Sequence FES seen LOW.
		8	CAS / MTS symbol too short.
		9	CAS / MTS symbol too long.
		10- 15	reserved.
			Note: Coding errors are also signalled when the communication controller is in MONITOR_MODE. The error codes regarding CAS / MTS symbols concern only the monitored bit pattern, irrelevant whether those bit patterns occurred in the symbol window or elsewhere.
23–22	Reserved		Reads return zero and writes have no effect.
21	TXENB		Control of channel B transmit enable pin.
		0	txen2 pin drives a 0.
		1	txen2 pin drives a 1.
20	TXENA		Control of channel A transmit enable pin.
		0	txen1 pin drives a 0.
		1	txen1 pin drives a 1.
19	TXB		Control of channel B transmit pin.
		0	txd2 pin drives a 0.
		1	txd2 pin drives a 1.
18	TXA		Control of channel A transmit pin.
		0	txd1 pin drives a 0.
		1	txd1 pin drives a 1

Figure 17-104. Test Register1 (TEST1) Field Descriptions (Continued)

Bit	Name	Value	Description
17	RXB		Monitor channel B receive pin.
		0	rxd2 = 0
		1	rxd2 = 1
16	RXA		Monitor channel A receive pin.
		0	rxd1 = 0
		1	rxd1 = 1
15–10	Reserved		Reads return zero and writes have no effect.
9	AOB		Activity on B. The channel idle condition is specified in the FlexRay protocol spec v2.1, chapter 3, BITSTRB process
		0	No activity detected, channel B idle
		1	Activity detected, channel B not idle
8	AOA		Activity on A.The channel idle condition is specified in the FlexRay protocol spec v2.1, chapter 3, BITSTRB process
		0	No activity detected, channel A idle
		1	Activity detected, channel A not idle
7–6	Reserved		Reads return zero and writes have no effect.
5–4	TMC(1–0)		Test mode control.
		0	Normal operation mode, default
		1h	RAM test mode - All RAM blocks of the FlexRay module are directly accessible by the host. This mode is intended to enable testing of the embedded RAM blocks during production testing.
		2h	I/O test mode - The output pins txd1, txd2, txen1, txen2 are driven to the values defined by bits TXA, TXB, TXENA, TXENB. The values applied to the input pins rxd1, rxd2 can be read from register bits RXA, RXB
		3h	unused - mapped to normal operation mode.
3–2	Reserved		Reads return zero and writes have no effect.

Figure 17-104. Test Register1 (TEST1) Field Descriptions (Continued)

Bit	Name	Value	Description
1	ELBE		External Loop Back Enable. There are two possibilities to perform a loop back test. External loop back via physical layer or internal loop back for in-system self-test (default). In case of an internal loop back pins txen1,2 are in their inactive state, pins txd1,2 are set to HIGH, pins rxd1,2 are not evaluated. Bit ELBE is evaluated only when POC is in loop back mode and test mode control is in normal operation mode TMC[1:0] = "00".
		0	Internal loop back (default)
		1h	External loop back
0	WRTEN		Write test register enable. Enables write access to the test registers. To set the bit from 0 to 1 the test mode key has to be written as defined in section 17.23.1.4. The unlock sequence is not required when WRTEN is kept at 1 while other bits of the register are changed. The bit can be reset to 0 at any time.
		0	Write access to the test register is disabled.
		1	Write access to the test register is enabled.

17.23.1.1.1Asynchronous Transmit Mode (ATM)

The asynchronous transmit mode is entered by writing 1110 to the controller host interface command vector CMD(3–0) in the SUC configuration register 1 (controller host interface command: ATM) while the communication controller is in CONFIG state and bit WRTEN in the test register 1 is set to 1. When called in any other state or when bit WRTEN is not set, CMD(3–0) will be reset to 0000 = command_not_accepted. POCS(5–0) in the communication controller status vector will show 00 1000 while the FlexRay module is in ATM mode.

Asynchronous transmit mode can be left by writing 0001 (controller host interface command: CONFIG) to the controller host interface command vector CMD(3–0) in the SUC configuration register 1.

In ATM mode transmission of a FlexRay frame is triggered by writing the number of the respective message buffer to the input buffer command request register while bit STXR in the input buffer command mask register is set to 1. In this mode wakeup, startup, and clock synchronization are bypassed, the controller host interface command SEND_MTS results in the immediate transmission of a MTS symbol.

17.23.1.2 Loop Back Mode

The loop back mode is entered by writing 1111 to the controller host interface command vector CMD(3–0) in the SUC configuration register 1 (controller host interface command: LOOP_BACK) while the communication controller is in CONFIG state and bit WRTEN in the test register 1 is set to 1. This write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). When called in any other state or when bit WRTEN is not set, CMD(3–0) will be reset to 0000 = command_not_accepted. POCS(5–0) in the communication controller status vector will show 00 1101 while the FlexRay module is in loop back mode.

Loop back mode can be left by writing 0001 (controller host interface command: CONFIG) to the controller host interface command vector CMD(3–0) in the SUC configuration register 1.

The loop back test mode is intended to check the modules internal data paths. Normal, time triggered operation is not possible in loop back mode.

There are two possibilities to perform a loop back test. External loop back via the physical layer (TEST1.ELBE = ’1’) or internal loop back for in-system self-test (TEST1.ELBE = ’0’). In case of an internal loop back pins txen1,2_n are in their inactive state, pins txd1,2 are set, pins rxd1,2 are not evaluated.

A loop back test is started by the host configuring the FlexRay module and then writing a message to the input buffer and requesting the transmission by writing to the input buffer command request register. The message handler will transfer the message into the message RAM and then into the transient buffer of the selected channel. The channel protocol controller (PRT) will read (in 32-bit words) the message from the transmit part of the transient buffer and load it into its Rx / Tx shift register. The serial transmission is looped back into the shift register; its content is written into the receive part of the channels transient buffer before the next word is loaded.

The PRT and the message handler will then treat this transmitted message like a received message, perform an acceptance filtering, and store the message into the message RAM (assuming the message passed the acceptance filter, thus testing the acceptance filter logic). The loop back test ends with the host requesting this received message from the message RAM and then checking the contents of the output buffer.

Each FlexRay channel is tested separately. The FlexRay module cannot receive messages from the FlexRay bus while it is in the loop back mode.

The cycle counter value of frames used in loop back mode can be programmed by writing to the CCV(5:0) bits of the MTCCV register (writable in ATM and loop back mode only).

Note:
In case of an odd payload the last two bytes of the looped-back payload will be shifted by 16 bits to the right inside the last 32-bit data word.

17.23.1.3 Test Register 2 (TEST2)

Test register 2 holds all bits required for RAM test of the seven embedded RAM blocks of the communication controller. Write access to this register is only possible when bit WRTEN in the test register 1 is set.

Figure 17-105 and Table 17-102 illustrate this register.

Figure 17-105. Test Register2 (TEST2) [offset_CC = 0x14]

31	16
Reserved	
R-0	

15	14	13	7	6	5	4	3	2	0
RDPB	WRPB	Reserved		SSEL2(2–0)			Reserve d	RS(2–0)	
R-0	R/W-0	R-0		R/W-0			R-0	R/W-0	

R = Read, W = Write; -*n* = Value after reset

Figure 17-106. Test Register2 (TEST2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15	RDPB	0–1	Read parity bit. Value of parity bit read from the selected RAM location.
14	WRPB	0–1	Write parity bit. Value of parity bit to be written to the selected RAM location
13–7	Reserved		Reads return zero and writes have no effect.
6–4	SSEL(2–0)		Segment select. To enable access to the complete message RAM (8192 byte addresses) the message RAM is segmented.
		0	Access to RAM bytes 0000h to 03FFh enabled
		1h	Access to RAM bytes 0400h to 07FFh enabled
		2h	Access to RAM bytes 0800h to 0BFFh enabled
		3h	Access to RAM bytes 0C00h to 0FFFh enabled
		4h	Access to RAM bytes 1000h to 13FFh enabled
		5h	Access to RAM bytes 1400h to 17FFh enabled
		6h	Access to RAM bytes 1800h to 1BFFh enabled
		7h	Access to RAM bytes 1C00h to 1FFFh enabled
3	Reserved		Reads return zero and writes have no effect.

Figure 17-106. Test Register2 (TEST2) Field Descriptions (Continued)

Bit	Name	Value	Description
2–0	RS(2–0)		RAM select. In RAM test mode the RAM blocks selected by RS(2–0) are mapped to module address 0x400 to 7FF (1024 byte addresses).
		0	Input buffer RAM 1
		1h	Input buffer RAM 2
		2h	Output buffer RAM 1
		3h	Output buffer RAM 2
		4h	Transient buffer RAM A
		5h	Transient buffer RAM B
		6h	Message RAM
		7h	Unused

17.23.1.3.2RAM test mode

In RAM test mode [TMC(1–0) = 01], one of the seven RAM blocks can be selected for direct R/W access by programming RS(2–0) to the respective value; see Figure 17-107.

For external access the selected RAM block is mapped to address space 400h to 7FF (1024 byte addresses or 256 word addresses).

Since the length of the Message RAM exceeds the available address space, the Message RAM is segmented into segments of 1024 bytes. The segments can be selected by programming the bits SSEL(2:0) of test register 2.

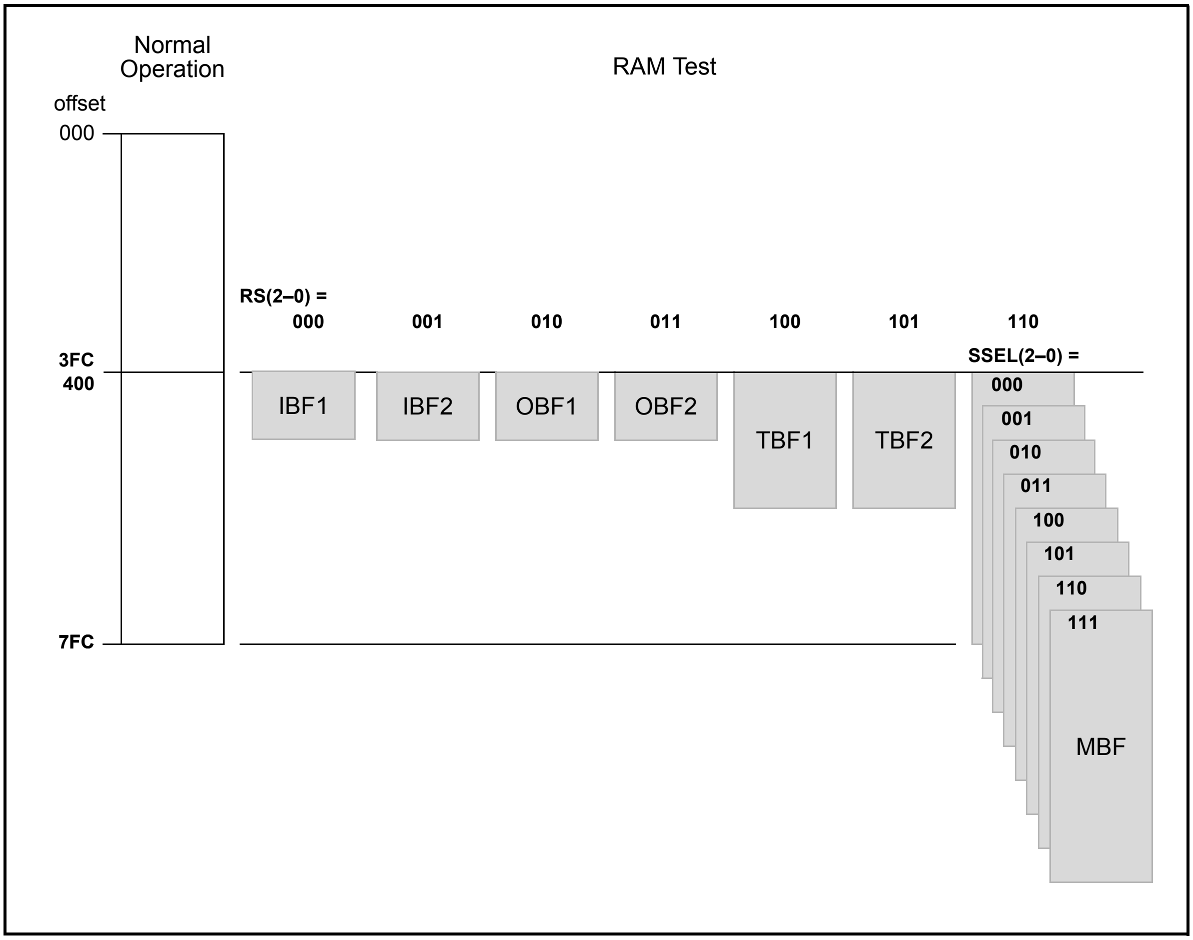


Figure 17-107. Test Mode Access to Communication Controller RAM Blocks

17.23.1.4 Lock Register (LCK)

The lock register is write-only. Reading the register will return 0x0000.

Figure 17-108 and Table 17-93 illustrate this register.

Figure 17-108. Lock Register (LCK) [offset_CC = 0x1C]

31	16
Reserved	
R-0	

15	8	7	0
TMK(7–0)		CLK(7–0)	
R/W-0		R/W-0	

R = Read, W = Write; -*n* = Value after reset

I

Table 17-93. Lock Register1 (LCK) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zero and writes have no effect.
15–8	TMK(8–0)	0– 1FF	Test mode key. To write bit WRTEN in the test register to 1, the write operation has to be directly preceded by two consecutive write accesses to the test mode key (unlock sequence). If this write sequence is interrupted by other write accesses, bit WRTEN is not set to 1 and the sequence has to be repeated. First write (LCK.TMK[7:0]):0x75 = 0b0111 0101 Second write (LCK.TMK[7:0]):0x8A = 0b1000 1010 Third write: TEST1.WRTEN = 1
7–0	CLK(7–0)	0–FF	Configuration lock key. To leave CONFIG state by writing to CMD(3–0) in the SUC configuration register 1 (commands READY; MONITOR_MODE; ATM; LOOP_BACK), the write operation has to be directly preceded by two consecutive write accesses to the configuration lock key (unlock sequence). If this write sequence is interrupted by other write accesses, the communication controller remains in CONFIG state and the sequence has to be repeated. First write (LCK.CLK[7:0]): 0xCE= 0b1100 1110 Second write (LCK.CLK[7:0]): 0x31= 0b0011 0001 Third write (SUCC.CMD[3:0])

Note:
In case that the Host uses 8/16-bit accesses to write the listed bit fields, the programmer has to ensure that no "dummy accesses" e.g. to the remaining register bytes / words are inserted by the compiler.

17.23.2 Interrupt Registers

17.23.2.1 Error Interrupt Register (EIR)

The flags are set when the communication controller detects one of the listed error conditions. They remain set until the host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. A reset will also clear the register.

Figure 17-109 and Table 17-94 illustrate this register.

Figure 17-109. Error Interrupt Register (EIR) [offset_CC = 0x20]

31–27	26	25	24	23–19	18	17	16
Reserved	TABB	LTVB	EDB	Reserved	TABA	LTVA	EDA
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

15–12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MHF	IOBA	IIBA	EFA	RFO	PERR	CCL	CCF	SFO	SFBM	CNA	PEMC
R-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	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

.

Table 17-94. Error Interrupt Register (EIR) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26	TABB		Transmission Across Boundary Channel B. The flag signals to the Host that a transmission across a slot boundary occurred for channel B.
		0	No transmission across slot boundary detected on channel B
		1	Transmission across slot boundary detected on channel B
25	LTVB		Latest transmit violation channel B. The flag signals a latest transmit violation on channel B to the host.
		0	No latest transmit violation detected on channel B
		1	Latest transmit violation detected on channel B
24	EDB		Error detected on channel B. This bit is set whenever one of the flags SEDB, CEDB, CIB, SBVB in the Aggregated channel status register is set.
		0	No error detected on channel B
		1	Error detected on channel B
23–19	Reserved		Reads return zero and writes have no effect.
18	TABA		Transmission Across Boundary Channel A. The flag signals to the Host that a transmission across a slot boundary occurred for channel A.
		0	No transmission across slot boundary detected on channel A

Table 17-94. Error Interrupt Register (EIR) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	Transmission across slot boundary detected on channel A
17	LTVA		Latest transmit violation channel A. The flag signals a latest transmit violation on channel A to the host.
		0	No latest transmit violation detected on channel A
		1	Latest transmit violation detected on channel A
16	EDA		Error detected on channel A. This bit is set whenever one of the flags SEDA, CEDA, CIA, SBVA in the Aggregated channel status register is set.
		0	No error detected on channel A
		1	Error detected on channel A
15–12			Reads return zero and writes have no effect.
11	MHF		Message Handler Constraints Flag. The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags MHDF.SNUA, MHDF.SNUB, MHDF.FNFA, MHDF.FNFB, MHDF.TBFA, MHDF.TBFB, MHDF.WAHP changes from ’0’ to ’1’.
		0	No Message Handler failure detected
		1	Message Handler failure detected
10	IOBA		Illegal Output buffer Access. This flag is set by the communication controller when the Host requests the transfer of a message buffer from the Message RAM to the Output Buffer while OBCR.OBSYS is set to ’1’.
		0	No illegal Host access to Output Buffer occurred
		1	Illegal Host access to Output Buffer occurred
9	IIBA		Illegal Input Buffer Access. This flag is set by the communication controller when the Host wants to modify a message buffer via Input Buffer and one of the following conditions applies: 1. The communication controller is not in CONFIG or DEFAULT_CONFIG state and the Host writes to the Input Buffer Command Request register to modify the following: • the Header section of message buffer 0, 1 if configured for transmission in key slot • the Header section of static message buffers with buffer number < MRC.FDB[7:0] while MRC.SEC[1:0] = "01" • the Header section of any static or dynamic message buffer while MRC.SEC[1:0] = "1x" • Header and / or data section of any message buffer belonging to the receive FIFO 2. The Host writes to any register of the Input Buffer while IBCR.IBSYH is set to ’1’.
		0	No illegal Host access to Input Buffer occurred
		1	Illegal Host access to Input Buffer occurred

Table 17-94. Error Interrupt Register (EIR) Field Descriptions (Continued)

Bit	Name	Value	Description
8	EFA		Empty FIFO Access. This flag is set by the communication controller when the Host requests the transfer of a message from the receive FIFO via Output Buffer while the receive FIFO is empty.
		0	No Host access to empty FIFO occurred
		1	Host access to empty FIFO occurred
7	RFO		Receive FIFO overrun. This flag is set by the communication controller when a receive FIFO overrun was detected. The flag is cleared by the next FIFO read access of the host. After this read access one position in the FIFO is empty again.
		0	No receive FIFO overrun detected
		1	A receive FIFO overrun has been detected
6	PERR		Parity error. The flag signals a parity error to the host. The flag is set by the parity logic of the communication controller when it detects a parity error while reading from one of the FlexRay RAM blocks. See section 17.27.1 for more information.
		0	No parity error detected
		1	Parity error detected
5	CCL		CHI Command Locked. The flag signals that the write access to the CHI command vector SUCC1.CMD[3:0] was not successful because it coincided with a POC state change triggered by protocol functions. In this case bit CNA is also set to ’1’.
		0	CHI command accepted
		1	CHI command not accepted
4	CCF		Clock correction failure This flag is set at the end of the cycle whenever one of the following errors occurred: • Missing rate correction signal • Missing offset correction signal • Clock correction Failed counter stopped at 15 • Clock correction Limit Reached The clock correction status is monitored in the communication controller error vector and sync frame status register.
		0	No clock correction error
		1	Clock correction failed
3	SFO		Sync frame overflow Set when either the number of sync frames received during the last communication cycle or the total number of different sync frame IDs received during the last double cycle exceeds the maximum number of sync frames as defined by SNM(3:0) in the GTU configuration register 2.
		0	Number of received sync frames in the configured range
		1	More sync frames received than configured by SNM(3–0)

Table 17-94. Error Interrupt Register (EIR) Field Descriptions (Continued)

Bit	Name	Value	Description
2	SFBM		Sync frames below minimum. This flag signals that the number of sync frames received during the last communication cycle was below the limit required by the FlexRay protocol. The minimum number of sync frames per communication cycle is 2.
		0	Two or more sync frames received during last communication cycle
		1	Less than two sync frame received during last communication cycle
1	CNA		Command not accepted. The flag signals that the controller host interface command vector CMD(3–0) in the SUC configuration register 1 was reset to 0000 due to an unaccepted controller host interface command.
		0	controller host interface command accepted
		1	controller host interface command not accepted
0	PEMC		POC error mode changed. This flag is set whenever the error mode signalled by ERRM(1–0) in the communication controller error vector register has changed.
		0	Error mode has not changed
		1	Error mode has changed

17.23.2.2 Status Interrupt Register (SIR)

The flags are set by the communication controller when a corresponding event occurs. They remain set until the host clears them. If enabled, an interrupt is pending while one of the bits is set. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. A hardware reset will also clear the register.

Figure 17-110 and Table 17-95 illustrate this register.

Figure 17-110. Status Interrupt Register (SIR) [offset_CC = 0x24]

31 26	25	24	23 18	17	16
Reserved	MTSB	WUPB	Reserved	MTSA	WUPA
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDS	MBSI	SUCS	SWE	TOBC	TIBC	TI1	TI0	NMVC	RFCL	RFNE	RXI	TXI	CYCS	CAS	WST
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

Table 17-95. Status Interrupt Register (SIR) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25	MTSB		MTS received on channel B. Media access test symbol received on channel B during the last symbol window. Updated by the communication controller for each channel at the end of the symbol window.
		0	No MTS symbol received
		1	MTS symbol received
24	WUPB		Wakeup pattern channel B. This flag is set by the communication controller when a wakeup pattern was received on channel B.
		0	No wakeup pattern on channel B
		1	Wakeup pattern on channel B
23–18	Reserved		Reads return zero and writes have no effect.
17	MTSA		MTS received on channel A. Media access test symbol received on channel A during the last symbol window. Updated by the communication controller for each channel at the end of the symbol window.
		0	No MTS symbol received
		1	MTS symbol received

Table 17-95. Status Interrupt Register (SIR) Field Descriptions (Continued)

Bit	Name	Value	Description
16	WUPA		Wakeup pattern channel A. This flag is set by the communication controller when a wakeup pattern was received on channel A.
		0	No wakeup pattern on channel A
		1	Wakeup pattern on channel A
15	SDS		Start of Dynamic Segment. This flag is set by the communication controller when the dynamic segment starts.
		0	Dynamic segment not yet started
		1	Dynamic segment started
14	MBSI		Message buffer status interrupt. This flag is set by the communication controller if bit MBI of a dedicated receive buffer is set to 1 and when the status of that message buffer has been updated due to reception of a: • valid frame with payload • valid frame with payload zero • null frame • corrupted frame or an empty slot
		0	No message buffer status has been updated.
		1	Message buffer status of at least one receive buffer has been updated
13	SUCS		Startup completed successfully. This flag is set whenever a startup completed successfully and the communication controller entered NORMAL_ACTIVE state.
		0	No startup completed successfully
		1	Startup completed successfully
12	SWE		Stop watch event. If enabled by the respective control bits located in the Stop watch register, a detected edge on external stop watch pin or a software trigger event will generate a stop watch event. **Note: Please refer to the device data sheet, if the external stop watch pin is available**
		0	No stop watch event
		1	Stop watch event occurred
11	TOBC		Transfer output buffer completed. This flag is set whenever a transfer from the message RAM to the output buffer has completed and bit OBSYS in the output buffer command request register has been reset by the message handler.
		0	No transfer completed since bit was reset
		1	Transfer between message RAM and output buffer completed
10	TIBC		Transfer input buffer completed. This flag is set whenever a transfer from input buffer to the message RAM has completed and bit IBSYS in the input buffer command request register has been reset by the message handler.

Table 17-95. Status Interrupt Register (SIR) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	No transfer completed since bit was reset
		1	Transfer between input buffer and message RAM completed
9	TI1		Timer interrupt 1. This flag is set whenever the conditions programmed in the timer interrupt 1 configuration register are met. A timer interrupt 1 is also signalled on pin CC_tint1.
		0	No timer interrupt 1
		1	Timer interrupt 1 occurred
8	TI0		Timer interrupt 0. This flag is set whenever the conditions programmed in the timer interrupt 0 configuration register are met. A timer interrupt 0 is also signalled on pin CC_tint0.
		0	No timer interrupt 0
		1	Timer interrupt 0 occurred
7	NMVC		Network management vector changed. This interrupt flag signals a change in the Network management vector visible to the host.
		0	No change in the network management vector
		1	Network management vector changed
6	RFCL		Receive FIFO critical level. This flag is set when the receive FIFO fill level FSR.RFFL[7:0] is equal or greater than the critical level as configured by FCL.CL[7:0].
		0	Receive FIFO below critical level
		1	Receive FIFO critical level reached
5	RFNE		Receive FIFO not empty. This flag is set by the communication controller when a received valid frame was stored into the empty receive FIFO. The actual state of the receive FIFO is monitored in register FSR.
		0	Receive FIFO is empty
		1	Receive FIFO is not empty
4	RXI		Receive interrupt. This flag is set by the communication controller when the payload segment of a received valid frame was stored into the data section of a matching dedicated receive buffer and if bit MBI of that message buffer is set to 1.
		0	No data section has been updated
		1	At least one data section has been updated
3	TXI		Transmit interrupt. This flag is set by the communication controller after successful frame transmission if bit MBI in the respective message buffer is set to 1.
		0	No frame transmitted
		1	At least one frame was transmitted successfully

Table 17-95. Status Interrupt Register (SIR) Field Descriptions (Continued)

Bit	Name	Value	Description
2	CYCS		Cycle start interrupt. This flag is set by the communication controller when a communication cycle starts.
		0	No communication cycle started
		1	Communication cycle started
1	CAS		Collision avoidance symbol. This flag is set by the communication controller when a CAS was received.
		0	No CAS symbol received
		1	CAS symbol received
0	WST		This flag is set when WSV(2:0) in the communication controller status vector register changes to a value other than UNDEFINED.
		0	Wakeup status unchanged
		1	Wakeup status changed

17.23.2.3 Error Interrupt Line Select (EILS)

The settings in the error interrupt line select register assigns an interrupt generated by a specific error interrupt flag to one of the two module interrupt lines (CC_int0 or CC_int1).

Figure 17-111 and Table 17-96 illustrate this register.

Figure 17-111. Error Interrupt Line Select Register (EILS) [offset_CC = 0x28]

31–27	26	25	24	23–19	18	17	16
Reserved	TABBL	LTVBL	EDBL	Reserved	TABAL	LTVAL	EDAL
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

15–11	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MHFL	IOBAL	IIBAL	EFAL	RFOL	PERRL	CCLL	CCFL	SFOL	SFBML	CNAL	PEMCL
R-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	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

Table 17-96. Error Interrupt Line Select Register (EILS) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26	TABBL		Transmission Across Boundary Channel B Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
25	LTVBL		Latest transmit violation channel B interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
24	EDBL		Error detected on channel B interrupt line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
23–19	Reserved		Reads return zero and writes have no effect.
18	TABAL		Transmission Across Boundary Channel A Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
17	LTVAL		Latest transmit violation channel A interrupt line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1

Table 17-96. Error Interrupt Line Select Register (EILS) Field Descriptions (Continued)

Bit	Name	Value	Description
16	EDAL		Error detected on channel A interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
15–12	Reserved		Reads return zero and writes have no effect.
11	MHFL		Message Handler Constraints Flag Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
10	IOBAL		Illegal Output Buffer Access Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
9	IIBAL		Illegal Output Buffer Access Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
8	EVAL		Empty FIFO Access Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
7	RFOL		Receive FIFO overrun interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
6	PERRL		Parity error interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
5	CCLL		CHI Command Locked Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
4	CCFL		Clock correction failure interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
3	SFOL		Sync frame overflow interrupt line.

Table 17-96. Error Interrupt Line Select Register (EILS) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
2	SFBML		Sync frames below minimum interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
1	CNAL		Command not Accepted interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
0	PEMCL		POC error mode changed interrupt line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1

17.23.2.4 Status Interrupt Line Select (SILS)

The settings in the status interrupt line select register assign an interrupt generated by a specific status interrupt flag to one of the two module interrupt lines (CC_int0 or CC_int1).

Figure 17-112 and Table 17-97 illustrate this register.

Figure 17-112. Status Interrupt Line Select Register (SILS) [offset_CC = 0x2C]

31 26	25	24	23 18	17	16
Reserved	MTSBL	WUPBL	Reserved	MTSAL	WUPAL
R-0	R/W-1	R/W-1	R-0	R/W-1	R/W-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSL	MBSIL	SUCSL	SWEL	TOBCL	TIBCL	TI1L	TI0L	NMVCL	RFFL	RFNEL	RXIL	TXIL	CYCSL	CASL	WSTL
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

R = Read, W = Write; -*n* = Value after reset

Table 17-97. Status Interrupt Line Select Register (SILS) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25	MTSBL		Media access test symbol channel B interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
24	WUPBL		Wakeup pattern channel B interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
23–18	Reserved		Reads return zero and writes have no effect.
17	MTSAL		Media access test symbol channel A interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
16	WUPAL		Wakeup pattern channel A interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
15	SDSL		Start of Dynamic Segment Interrupt Line
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1

Table 17-97. Status Interrupt Line Select Register (SILS) Field Descriptions (Continued)

Bit	Name	Value	Description
14	MBSIL		Message buffer status interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
13	SUCSL		Startup completed Successfully interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
12	SWEL		Stop watch event interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
11	TOBCL		Transfer output buffer completed interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
10	TIBCL		Transfer input buffer completed interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
9	TI1L		Timer interrupt 1 line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
8	TI0L		Timer interrupt 0 line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
7	NMVCL		Network management vector changed interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
6	RFFL		Receive FIFO full interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
5	RFNEL		Receive FIFO not empty interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1

Table 17-97. Status Interrupt Line Select Register (SILS) Field Descriptions (Continued)

Bit	Name	Value	Description
4	RXIL		Receive interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
3	TXIL		Transmit interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
2	CYCSL		Cycle start interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
1	CASL		Collision Avoidance symbol interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1
0	WSTL		Wakeup status interrupt line.
		0	Interrupt assigned to interrupt line CC_int0
		1	Interrupt assigned to interrupt line CC_int1

17.23.2.5 Error Interrupt Enable Set / Reset (EIES, EIER)

The settings in the error interrupt enable register determine which status changes in the error interrupt register will result in an interrupt. The enable bits are set by writing to address 0x0030 and reset by writing to address 0x0034. Writing a 1 sets / resets the specific enable bit, a 0 has no effect.

Figure 17-113 and Table 17-98 illustrate this register.

Figure 17-113. Error Interrupt Enable Set/Reset Register (EIES, EIER) [offset_CC = 0x30–0x34]

31 27	26	25	24	23 19	18	17	16
Reserved	TABBE	LTVBE	EDBE	Reserved	TABAE	LTVAE	EDAE
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

15 12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MHFE	IOBAE	IIBAE	EFAE	RFOE	PERRE	CCLE	CCFE	SFOE	SFBME	CNAE	PEMCE
R-0	R/W-	R/W-	R/W-	R/W-	R/W-0	R/W-0	R/W-	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

Table 17-98. Error Interrupt Set/Reset Register (EIES, EIER) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26	TABBE		Transmission across boundary channel B interrupt enable.
		0	Interrupt disabled
		1	Transmission across boundary channel B interrupt enabled
25	LTVBE		Latest transmit violation channel B interrupt enable.
		0	Interrupt disabled
		1	Latest transmit violation channel B interrupt enabled
24	EDBE		Error detected on channel B interrupt enable.
		0	Interrupt disabled
		1	Error detected on channel B interrupt enabled
23–19	Reserved		Reads return zero and writes have no effect.
18	TABAE		Transmission across boundary channel A interrupt enable.
		0	Interrupt disabled
		1	Transmission across boundary channel A interrupt enabled
17	LTVAE		Latest transmit violation channel A interrupt enable.
		0	Interrupt disabled

Table 17-98. Error Interrupt Set/Reset Register (EIES, EIER) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	Latest transmit violation channel A interrupt enabled
16	EDAE		Error detected on channel A interrupt enable.
		0	Interrupt disabled
		1	Error detected on channel A interrupt enabled
15–12	Reserved		Reads return zero and writes have no effect.
11	MHFE		Message handler constraints flag interrupt enable.
		0	Interrupt disabled
		1	Message handler constraints flag interrupt enabled
10	IOBAE		Illegal output buffer access interrupt enable.
		0	Interrupt disabled
		1	Illegal output buffer access interrupt enabled
9	IIBAE		Illegal input buffer access interrupt enable.
		0	Interrupt disabled
		1	Illegal input buffer access interrupt enabled
8	EFAE		Empty FIFO access interrupt enable.
		0	Interrupt disabled
		1	Empty FIFO access interrupt enabled
7	RFOE		Receive FIFO overrun interrupt enable.
		0	Interrupt disabled
		1	Receive FIFO overrun interrupt enabled
6	PERRE		Parity error interrupt enable.
		0	Interrupt disabled
		1	Parity error interrupt enabled
5	CCLE		CHI command locked interrupt enable.
		0	Interrupt disabled
		1	CHI command locked interrupt enabled
4	CCFE		Clock correction failure interrupt enable.
		0	Interrupt disabled
		1	Clock correction failure interrupt enabled
3	SFOE		Sync frame overflow interrupt enable.

Table 17-98. Error Interrupt Set/Reset Register (EIES, EIER) Field Descriptions (Continued)

Bit	Name	Value	Description
		0	Interrupt disabled
		1	Sync frame overflow interrupt enabled
2	SFBME		Sync frames below minimum interrupt enable.
		0	Interrupt disabled
		1	Sync frames below minimum interrupt enabled
1	CNAE		Command not Accepted interrupt enable.
		0	Interrupt disabled
		1	Command not valid interrupt enabled
0	PEMCE		POC error mode changed interrupt enable.
		0	Interrupt disabled
		1	Protocol error mode changed interrupt enabled

17.23.2.6 Status Interrupt Enable Set / Reset Register (SIES, SIER)

The settings in the status interrupt enable register determine which status changes in the status interrupt register will result in an interrupt. The enable bits are set by writing to address 0x0038 and reset by writing to address 0x003C. Writing a 1 sets / resets the specific enable bit, a 0 has no effect.

Figure 17-114 and Table 17-99 illustrate this register.

Figure 17-114. Status Interrupt Enable Set/Reset Register (SIES, SIER) [offset_CC = 0x38/0x3C]

31	26	25	24	23	18	17	16
Reserved		MTSBE	WUPBE	Reserved		MTSAE	WUPAE
R-0		R/W-0	R/W-0	R-0		R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCE	RFFE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTE
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

Table 17-99. Status Interrupt Enable Set/Reset Register (SIES, SIER) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25	MTSBE		MTS received on channel B interrupt enable.
		0	Interrupt disabled
		1	MTS received on channel B interrupt enabled
24	WUPBE		Wakeup pattern channel B interrupt enable.
		0	Interrupt disabled
		1	Wakeup pattern channel B interrupt enabled
23–18	Reserved		
17	MTSAE		MTS received on channel A interrupt enable
		0	Interrupt disabled
		1	MTS received on channel A interrupt enabled
16	WUPAE		Wakeup pattern channel A interrupt enable.
		0	Interrupt disabled
		1	Wakeup pattern channel A interrupt enabled
15	SDSE		Start of dynamic segment interrupt enable.
		0	Interrupt disabled
		1	Start of dynamic segment interrupt enabled

Table 17-99. Status Interrupt Enable Set/Reset Register (SIES, SIER) Field Descriptions (Continued)

Bit	Name	Value	Description
14	MBSIE		Message buffer status interrupt enable.
		0	Interrupt disabled
		1	Message buffer status interrupt enabled
13	SUCSE		Startup completed successfully interrupt enable.
		0	Interrupt disabled
		1	Startup completed successfully interrupt enabled
12	SWEE		Stop watch event interrupt enable.
		0	Interrupt disabled
		1	Stop watch event interrupt enabled
11	TOBCE		Transfer output buffer completed interrupt enable.
		0	Interrupt disabled
		1	Transfer output buffer completed interrupt enabled
10	TIBCE		Transfer input buffer completed interrupt enable.
		0	Interrupt disabled
		1	Transfer input buffer completed interrupt enabled
9	TI1E		Timer interrupt 1 enable.
		0	Interrupt disabled
		1	Timer interrupt 1 enabled
8	TI0E		Timer interrupt 0 enable.
		0	Interrupt disabled
		1	Timer interrupt 0 enabled
7	NMVCE		Network management vector changed interrupt enable.
		0	Interrupt disabled
		1	Network management vector changed interrupt enabled
6	RFFE		Receive FIFO full interrupt enable.
		0	Interrupt disabled
		1	Receive FIFO overrun interrupt enabled
5	RFNEE		Receive FIFO not empty interrupt enable.
		0	Interrupt disabled
		1	Receive FIFO not empty interrupt enabled
4	RXIE		Receive interrupt enable.
		0	Interrupt disabled

Table 17-99. Status Interrupt Enable Set/Reset Register (SIES, SIER) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	Receive interrupt enabled
3	TXIE		Transmit interrupt enable.
		0	Interrupt disabled
		1	transmit interrupt enabled
2	CYCSE		Cycle start interrupt enable.
		0	Interrupt disabled
		1	Cycle start interrupt enabled
1	CASE		Collision avoidance symbol interrupt enable.
		0	Interrupt disabled
		1	Collision Avoidance symbol interrupt enabled
0	WSTE		Wakeup status interrupt enable.
		0	Interrupt disabled
		1	Wakeup status interrupt enabled

17.23.2.7 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines (CC_int0, CC_int1) can be enabled / disabled separately by programming bit EINT0 and EINT1.

Figure 17-115 and Table 17-100 illustrate this register.

Figure 17-115. Interrupt Line Enable Register (ILE) [offset_CC = 0x40]

31–16
Reserved
RW-0

15–2	1–0
Reserved	EINT(1–0)
R-0	R/W-0

R = Read, W = Write; -*n* = Value after reset

Table 17-100. Interrupt Line Enable Register (ILE) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return zero and writes have no effect.
1–0	EINT(1–0)	0 - 3h	Enable interrupt line (1–0).
		0	Interrupt line CC_int1 and CC_int0 disabled
		1	Interrupt line CC_int1 disabled and CC_int0 enabled
		2	Interrupt line CC_int1 enabled and CC_int0 disabled
		3	Interrupt line CC_int1 and CC_int0 enabled

17.23.2.8 Timer 0 Configuration Register (T0C)

This absolute timer specifies, in terms of cycle count and macrotick, the point in time when the timer 0 interrupt occurs. The timer 0 interrupt generates a non maskable interrupt signal on CC_tint0.

Timer 0 can be activated as long as the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state. Timer 0 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by writing bit T0RC to 0.

Figure 17-116 and Table 17-101 illustrate this register.

Figure 17-116. Timer 0 Configuration Register (T0C) [offset_CC = 0x44]

31 30	29 … 16
Reserved	TOMO(13–0)
R-0	R/W-0

15	14 … 8	7 … 2	1	0
Reserve d	TOCC(7–0)	Reserved	TOMS	TORC
R-0	R/W-0	R-0	R/W-0	

R = Read, W = Write; -*n* = Value after reset

Table 17-101. Timer 0 Configuration Register (T0C) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–16	TOMO(13–0)	0–3FFFh	Timer 0 macrotick offset. Configures the macrotick offset from the beginning of the cycle where the interrupt is to occur. The Timer 0 interrupt occurs at this offset for each cycle in the cycle set.
15	Reserved		Reads return zero and writes have no effect.
14–8	TOCC(7–0)	0–FFh	Timer 0 cycle code. The 7-bit timer 0 cycle code determines the cycle set used for generation of the timer 0 interrupt.
7–2	Reserved		Reads return zero and writes have no effect.
1	T0MS		Timer 0 mode select.
		0	Single-shot mode
		1	Continuous mode
0	T0RC		Timer 0 Run control
		0	Timer 0 halted
		1	Timer 0 running

17.23.2.9 Timer 1 Configuration Register (T1C)

This relative timer generates an interrupt on the non maskable interrupt signal CC_tint1 after the specified number of macroticks has expired.

Timer 1 can be activated as long as the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state. Timer 1 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by writing bit T1RC to 0.

Figure 17-117 and Table 17-102 illustrate this register.

Figure 17-117. Timer 1 Configuration Register (T1C) [offset_CC = 0x48]

31–30	29–16
Reserved	TIMC(13–0)
R-0	R/W-2

15–2	1	0
Reserved	T1MS	T1RC
R-0	R/W-0	

R = Read, W = Write; -*n* = Value after reset

Table 17-102. Timer 1 Configuration Register (T1C) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–16	TIMC(13–0)		Timer 1 macrotick count. When the configured macrotick count is reached the timer 1 interrupt is generated. In case the configured macrotick count is not within the valid range, timer 1 will not start. Valid values: 2 to 16383 macroticks in continuous mode 1 to 16383 macroticks in single-shot mode
		2–3FFFh	Continuous mode
		1–3FFFh	Single-shot mode
15–2	Reserved		Reads return zero and writes have no effect.
1	T1MS		Timer 1 mode select.
		0	Single-shot mode
		1	Continuous mode
0	T1RC		Timer 1 run control.
		0	Timer 1 halted
		1	Timer 1 running

17.23.2.10Stop Watch Register 1 Register (STPW1)

The stop watch is activated by an interrupt event (CC_int0 or CC_int1), by writing bit SSWT to 1, or by an external event.

With the macrotick counter increment following next to the stop watch activation the actual cycle counter and macrotick value is stored in the stop watch register 1 (stop watch event) and the slot counter values for channel A and channel B are stored in stop watch register 2.

Figure 17-118 and Table 17-103 illustrate the stop watch register1, Figure 17-119 and Table 17-104 illustrate stop watch register 2.

Figure 17-118. Stop Watch Register 1 (STPW1) [offset_CC = 0x4C]

31 30	29 16
Reserved	SMTV(13–0)
R-0	R-0

15 14	13 8	7	6	5	4	3	2	1	0
Reserved	SCCV(5–0)	Reserved	EINT1	EINT0	EETP	SSWT	EDGE	SWMS	ESWT
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read; -*n* = Value after reset

Table 17-103. Stop Watch Register 1 (STPW1) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–16	SMTV(13–0)	0–3E80h	Stopped macrotick value. State of the macrotick counter when the stop watch event occurred.
15–14	Reserved		
13–8	SCCV(5–0)	0–3Fh	Stopped cycle counter value. State of the cycle counter when the stop watch event occurred.
7	Reserved		
6	EINT1		Enable interrupt 1 trigger. Enables stop watch trigger by CC_int1 event if ESWT = ’1’.
		0	Stop watch trigger by CC_int1 disabled
		1	CC_int1 event triggers stop watch
5	EINT0		Enable interrupt 0 trigger. Enables stop watch trigger by CC_int0 event if ESWT = ’1’.
		0	Stop watch trigger by CC_int0 disabled
		1	CC_int0 event triggers stop watch

Table 17-103. Stop Watch Register 1 (STPW1) Field Descriptions (Continued)

Bit	Name	Value	Description
4	EETP		Enable external trigger pin. Enables stop watch trigger event from external pin, if ESWT = ’1’.
		0	External trigger disabled
		1	Stop watch activated by external trigger
3	SSWT		Software stop watch trigger. When the host writes this bit to 1 the stop watch is activated. After the actual cycle counter and macrotick value are stored in the stop watch register this bit is reset to 0. The bit is only writable while ESWT = 0.
		0	Software trigger reset
		1	Stop watch activated by software trigger
2	EDGE		Stop watch trigger edge select.
		0	Falling edge
		1	Rising edge
1	SWMS		Stop watch mode select.
		0	Single-shot mode
		1	Continuous mode
0	ESWT		External stop watch trigger. If enabled an external event activates the stop watch. In single-shot mode this bit is reset to 0 after the stop watch event occurred.
		0	External stop watch trigger disabled
		1	External stop watch trigger enabled

Note:
Bits ESWT and SSWT cannot be set to ’1’ simultaneously. In this case the write access to the register is ignored, and both bits keep their previous values. Either the external stop watch trigger or the software stop watch trigger may be used.

Note:
The availability of an external stop watch pin is device dependant. Please refer to the device data sheet for details.

17.23.2.11 Stop Watch Register 2 Register (STPW2)

Figure 17-119 and Table 17-104 illustrate this stop watch register 2.

Figure 17-119. Stop Watch Register 2 (STPW2) [offset_CC = 0x50]

<table>
<tr><td>31</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="11">SSCVB(10-0)</td></tr>
<tr><td colspan="2">R-0</td><td colspan="11">R-0</td></tr>
<tr><td>15</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="11">SSCVA(10-0)</td></tr>
<tr><td colspan="2">R-0</td><td colspan="11">R-0</td></tr>
</table>

R = Read; -*n* = Value after reset

Table 17-104. Stop Watch Register 2 (STPW2) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26–16	SSCVB(10–0)	0–7FFh	Stop watch captured slot counter value channel B. State of the slot counter for channel B when the stop watch event occurred.
15–11	Reserved		Reads return zero and writes have no effect.
10–0	SSCVA(10–0)	0–7FFh	Stop watch captured slot counter value channel A. State of the slot counter for channel A when the stop watch event occurred.

17.24 Control Registers

This section describes the registers provided by the communication controller to allow the host to control the operation of the communication controller. The FlexRay protocol specification requires the host to write application configuration data in CONFIG state only.

Note:
Please be aware that the configuration registers are not locked for writing in DEFAULT_CONFIG state.

The configuration data is reset when DEFAULT_CONFIG state is entered from hardware reset. To change POC state from DEFAULT_CONFIG to CONFIG state the host has to apply the controller host interface command CONFIG. If the host wants the communication controller to leave CONFIG state, the host has to proceed as described in section 17.23.1.4.

Note:
All bits marked with an asterisk (*) can be updated in DEFAULT_CONFIG or CONFIG state only.

17.24.1 SUC Configuration Register 1 (SUCC1)

Figure 17-120 and Table 17-105 illustrate this register.

Figure 17-120. SUC Configuration Register 1 (SUCC1) [offset_CC = 0x80]

31 28	27	26	25	24	23	22	21	20	19 16
Reserved	CCHB*	CCHA*	MTSB*	MTSA*	HCSE*	TSM*	WUCS*	PTA4*	PTA(3–0)*
R-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0

15 11	10	9	8	7	6 4	3 0
CSA(4–0)*	Reserved	TXSY*	TXST*	PBSY	Reserved	CMD(3–0)
R/W-00010	R-0	R/W-0	R/W-0	R-1	R-0	R/W-0

R = Read; W = Write; *-n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-105. SUC Configuration Register 1 (SUCC1) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
27	CCHB*		Connected to channel B. Configures whether the node is connected to channel B.
		0	Not connected to channel B
		1	Node connected to channel B (default by hardware reset)

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-105. SUC Configuration Register 1 (SUCC1) Field Descriptions (Continued)

Bit	Name	Value	Description
26	CCHA*		Connected to channel A. Configures whether the node is connected to channel A.
		0	Not connected to channel A
		1	Node connected to channel A (default by hardware reset)
25	MTSB*		Select channel B for MTS Transmission. The bit selects channel B for MTS symbol transmission if requested by writing CMD(3–0) = 1000. The flag is reset by default and may be modified only in DEFAULT_CONFIG or CONFIG state.
		0	Channel B not selected for MTS transmission
		1	Channel B selected for MTS transmission
			Note: MTSB may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to SUC Configuration Register 1 (SUCC1) is directly preceded by the unlock sequence for the Configuration Lock Key as described in 1.24.5 Lock Register (LCK). This may be combined with CHI command SEND_MTS. If both bits MTSA and MTSB are set to ’1’ an MTS symbol will be transmitted on both channels when requested by writing CMD[3:0] = "1000".
24	MTSA*		Select channel A for MTS Transmission. The bit selects channel A for MTS symbol transmission if requested by writing CMD(3–0) = 1000. The flag is reset by default and may be modified only in DEFAULT_CONFIG or CONFIG state.
		0	Channel A not selected for MTS transmission
		1	Channel A selected for MTS transmission
			Note: MTSA may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to SUC Configuration Register 1 (SUCC1) is directly preceded by the unlock sequence for the Configuration Lock Key as described in 1.24.5 Lock Register (LCK). This may be combined with CHI command SEND_MTS. If both bits MTSA and MTSB are set to ’1’ an MTS symbol will be transmitted on both channels when requested by writing CMD[3:0] = "1000".
23	HCSE*		Halt due to clock sync error. Controls reaction of the communication controller to a clock synchronization error. The bit can be modified in DEFAULT_CONFIG or CONFIG state only.
		0	communication controller will enter/remain in NORMAL_PASSIVE
		1	communication controller will enter HALT state

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-105. SUC Configuration Register 1 (SUCC1) Field Descriptions (Continued)

Bit	Name	Value	Description
22	TSM*		Transmission slot mode. Selects the initial transmission slot mode. In SINGLE slot mode the communication controller may only transmit in the preconfigured key slot. This slot is defined by the key slot ID, which is configured in the header section of message buffer 0. In all slot mode the communication controller may transmit in all slots. The bit can be written in DEFAULT_CONFIG or CONFIG state only. The communication controller changes to all slot mode when the host successfully applied the ALL_SLOTS command by writing CMD(3–0) = 0101 in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. The actual slot mode is monitored by SLM(1–0) in register CCSV.
		0	All slot mode
		1	Single slot mode (default by hardware reset)
21	WUCS*		Wakeup channel select. With this bit the host selects the channel on which the communication controller sends the Wakeup pattern. The communication controller ignores any attempt to change the status of this bit when not in DEFAULT_CONFIG or CONFIG state.
		0	Send wakeup pattern on channel A
		1	Send wakeup pattern on channel B
20–16	PTA(4–0)*	0–1Fh even/odd cycle pairs	Passive to active. Defines the number of consecutive even/odd cycle pairs that must have valid clock correction terms before the communication controller is allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state. If set to 00000 the communication controller is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state. It can be modified in DEFAULT_CONFIG or CONFIG state only.
15–11	CSA(4–0)*	2–1Fh	Cold start attempts. Configures the maximum number of attempts that a cold starting node is permitted to try to start up the network without receiving any valid response from another node. It can be modified in DEFAULT_CONFIG or CONFIG state only. Must be identical in all nodes of a cluster.
10	Reserved		Reads return zero and writes have no effect.
9	TXSY*		Transmit sync frame in key slot. Defines whether the key slot is used to transmit a sync frame. The bit can be modified in DEFAULT_CONFIG or CONFIG state only. **Note: The protocol requires that both bits TXST and TXSY are set for coldstart nodes.**
		0	No sync frame transmission in key slot, node is neither sync nor coldstart node
		1	Key slot used to transmit sync frame, node is sync node

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-105. SUC Configuration Register 1 (SUCC1) Field Descriptions (Continued)

Bit	Name	Value	Description
8	TXST*		Transmit startup frame in key slot. Defines whether the key slot is used to transmit a startup frame. The bit can be modified in DEFAULT_CONFIG or CONFIG state only. Note: The protocol requires that both bits TXST and TXSY are set for coldstart nodes.
		0	No startup frame transmitted in key slot, node is non-coldstarter
		1	Key slot used to transmit startup frame, node is leading or following coldstarter
7	PBSY*		POC busy. Signals that the POC is busy and cannot accept a command from the host. CMD(3–0) is locked against write accesses.
		0	POC not busy, CMD(3–0) writable
		1	POC is busy, CMD(3–0) locked
6–4	Reserved		Reads return zero and writes have no effect.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-105. SUC Configuration Register 1 (SUCC1) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	CMD(3–0)*		The controller host interface command vector. The host may write any controller host interface command at any time, but certain commands are enabled only in certain POC states. If a command is not enabled, it will not be executed, the controller host interface command vector CMD(3–0) will be reset to 0000 = command_not_accepted, and flag CNA in the error interrupt register will be set to 1. In case the previous controller host interface (CHI) command has not yet completed, EIR.CCL is set to ’1’ together with EIR.CNA; the CHI command needs to be repeated. Except for HALT state, a POC state change command applied while the communication controller is already in the requested POC state neither causes a state change nor will EIR.CNA be set.
		0h	command_not_accepted
		1h	CONFIG
		2h	READY
		3h	WAKEUP
		4h	RUN
		5h	ALL_SLOTS
		6h	HALT
		7h	FREEZE
		8h	SEND_MTS
		9h	ALLOW_COLDSTART
		Ah	RESET_STATUS_INDICATORS
		Bh	MONITOR_MODE
		Ch	CLEAR_RAMS
		D–Eh	reserved
		Fh	LOOPBACK MODE

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Controller Host Interface Command Vector:

The following gives more information about the controller host interface commands.

- Reading CMD(3–0) shows whether the last controller host interface command was accepted.
- The actual POC state is monitored by POCS(5–0) in the communication controller status vector
- In most cases the Host must check SUCC1.PBSY before writing a new CHI command.

command_not_accepted

CMD(3–0) is reset to 0000 due to one of the following conditions:

- Illegal command applied by the host
- Host applied command to leave CONFIG state without preceding configuration lock key
- Host applied new command while execution of the previous host command has not completed

- Host writes command_not_accepted

When CMD(3–0) is reset to 0000 due to an unaccepted command, bit CNA in the error interrupt register is set, and - if enabled - an interrupt is generated. Commands which are not accepted are not executed.

CONFIG

Go to POC state CONFIG when called in POC states DEFAULT_CONFIG, READY, or in MONITOR_MODE. When called in HALT state the communication controller transits to POC state DEFAULT_CONFIG. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

READY

Go to POC state READY when called in POC states CONFIG, NORMAL_ACTIVE, NORMAL_PASSIVE, STARTUP, or WAKEUP. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

WAKEUP

Go to POC state WAKEUP when called in POC state READY. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

RUN

Go to POC state STARTUP when called in POC state READY. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

ALL_SLOTS

Leave single slot mode after successful startup / integration at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

HALT

Set the Halt request HRQ bit in the communication controller status vector register and go to POC state HALT at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

FREEZE

Go to POC state HALT immediately and set the Freeze status Indicator FSI bit in the communication controller status vector register. Can be called from any state.

SEND_MTS

Send single MTS symbol during the symbol window of the following cycle on the channel configured by MTSA, MTSB, when called in POC state NORMAL_ACTIVE. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

ALLOW_COLDSTART

The command resets bit CSI in the CCSV register to enable coldstarting of the node when called in any POC state except DEFAULT_CONFIG, CONFIG or HALT. When called in these states, CMD(3–0) will be reset to 0000 = command_not_accepted.

RESET_STATUS_INDICATORS

Reset status flags FSI, HRQ, CSNI, and CSAI in the communication controller status vector register.

CLEAR_RAMS

Sets bit CRAM in the message handler status register when called in DEFAULT_CONFIG or CONFIG state. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted. Bit CRAM is also set when the communication controller leaves hardware reset. By setting bit CRAM all internal RAM blocks are initialized to zero. During the initialization of the RAMs, PBSY will show POC busy. Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS.

The initialization of the Communication Controller internal RAM blocks takes 2048 VBUS clock cycles. There should be no host access to IBF or OBF during initialization of the internal RAM blocks after hardware reset or after assertion of controller host interface command CLEAR_RAMS. Before asserting controller host interface command CLEAR_RAMS the host should be aware that no transfer between message RAM and IBF / OBF or the transient buffer RAMs is ongoing. This command also resets the message buffer status registers (MHDS, TXRQ1/2/3/4, NDAT1/2/3/4, MBSC1/2/3/4).

Note:
All accepted commands with exception of CLEAR_RAMS and SEND_MTS will cause a change of the POC state in the VBUS clock domain after at most 8 cycles of the slower of the two clocks VBUS clock and 80MHz sample clock coming from the PLL. It is assumed that POC was not busy when the command was applied and that no POC state change was forced by bus activity in that time frame. Reading register Communication Controller Status Vector (CCSV) will show data that is additionally delayed by synchronization from sample clock to VBUS clock domain and by the CPU interface. The maximum additional delay is 12 cycles of the slower of the two clocks VBUS clock and sample clock.

MONITOR_MODE

Enter MONITOR_MODE when called in POC state CONFIG. In this mode the communication controller is able to receive FlexRay frames and CAS / MTS symbols. It is also able to detect coding errors. The temporal integrity of received frames is not checked. This mode can be used for debugging purposes, e.g. in case that the startup of a FlexRay network fails. When called in any other state, CMD(3–0) will be reset to 0000 = command_not_accepted.

17.24.2 SUC Configuration Register 2 (SUCC2)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-121 and Table 17-106 illustrate this register.

Figure 17-121. SUC Configuration Register2 (SUCC2) [offset_CC = 0x84]

<table>
<tr><td>31</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>21</td><td>20</td><td>19</td><td>18</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="4">LTN(3–0)*</td><td colspan="2">Reserved</td><td colspan="4">LT(20–16)*</td></tr>
<tr><td colspan="2">R-0</td><td colspan="4">R/W-1h</td><td colspan="2">R-0</td><td colspan="4">R/W-0</td></tr>
</table>

<table>
<tr><td>15</td><td>0</td></tr>
<tr><td colspan="2">LT(15–0)*</td></tr>
<tr><td colspan="2">R/W-504h</td></tr>
</table>

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

Table 17-106. SUC Configuration Register2 (SUCC2) Field Descriptions

Bit	Name	Value	Description
31–28	Reserved		Reads return zero and writes have no effect.
29–16	LTN(3–0)*	2–Fh	Listen timeout noise. Configures the upper limit for the startup and wakeup listen timeout in the presence of noise. Must be identical in all nodes of a cluster. The wakeup / startup noise timeout is calculated as follows: LT[20:0] • (LTN[3:0] + 1)
23–21	Reserved		Reads return zero and writes have no effect.
20–0	LT(20–0)*	504–139706h µT	Listen timeout. Configures the upper limit of the startup and wakeup listen timeout.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

17.24.3 SUC Configuration Register 3 (SUCC3)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-122 and Table 17-107 illustrate this register.

Figure 17-122. SUC Configuration Register 3 (SUCC3) [offset_CC = 0x88]

31–16
Reserved
R-0

15–8	7–4	3–0
Reserved	WCF(3–0)*	WCP(3–0)*
R-0	R/W-1h	R/W-1h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-107. SUC Configuration Register 3 (SUCC3) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are zeros and writes have no effect.
7–4	WCF(3–0)*	1–Fh	Maximum without clock correction fatal. These bits define the number of consecutive even/odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE or NORMAL_PASSIVE state. These must be identical in all nodes of a cluster. Note: The transition to HALT state is prevented if SUCC1.HCSE is not set.
3–0	WCP(3–0)*	1–Fh	Maximum without clock correction passive.These bits define the number of consecutive even/odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE to NORMAL_PASSIVE to HALT state. These must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only.

17.24.4 NEM Configuration Register (NEMC)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-123 and Table 17-108 illustrate this register.

Figure 17-123. NEM Configuration Register (NEMC) [offset_CC = 0x8C]

31–16
Reserved
R-0

15–4	3–0
Reserved	NML(3–0)*
R-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-108. NEM Configuration Register (NEMC) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads are undefined and writes have no effect.
3–0	NML(3–0)*	0–Ch bytes	Network management vector length (in bytes). These bits configure the length of the NM vector. The configured length must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.5 PRT Configuration Register 1 (PRTC1)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-124 and Table 17-109 illustrate this register.

Figure 17-124. PRT Configuration Register 1 (PRTC1) [offset_CC = 0x90]

31 26	25	24 16
RPW(5–0)*	Reserved	RXW(8–0)*
R/W-2h	R-0	R/W-4Ch

15 14	13 12	11	10 4	3 0
BRP(1–0)*	SPP(1-0)	Reserved	CASM(6–0)*	TSST(3–0)*
R/W-0	R/W-0	R-0	R-1 R/W-23h	R/W-3h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-109. PRT Configuration Register 1 (PRTC1) Field Descriptions

Bit	Name	Value	Description
31–26	RWP(5–0)*	0x2–0x3F	Repetition of transmission wakeup pattern. These bits configure the number of repetitions (sequences) of the transmission wakeup symbol.
25	Reserved		Reads return zero and writes have no effect.
24–16	RXW(8–0)*	0x4C–0x12D	Wakeup symbol receive window length. Configures the number of bit times used by the node to test the duration of the received wakeup pattern. Must be identical in all nodes of a cluster.
15–14	BRP(1–0)*		Baud rate prescaler. These bits configure the baud rate on the FlexRay bus. The baud rates listed below are valid with a sample clock of 80 MHz. One bit time always consists of 8 samples independent of the configured baud rate.
		0x0	10 MBit/s (Sample Clock Period = 12.5ns; 1 µT = 25ns; Samples per µT = 2)
		0x1	5 MBit/s (Sample Clock Period = 25ns; 1 µT = 25ns; Samples per µT = 1)
		0x2, 0x3	2.5 MBit/s (Sample Clock Period = 50ns; 1 µT = 50ns; Samples per µT = 1)
13–12	SPP(1–0)*		Strobe Point Position. Defines the sample count value for strobing. The strobed bit value is set to the voted value when the sample count is incremented to the value configured by SPP[1:0]. Note: The current revision 2.1 of the FlexRay protocol requires that SPP[1:0] = "00". The alternate strobe point positions could be used to compensate for asymmetries in the physical layer.
		0x0, 0x3	Sample 5 (default)
		0x1	Sample 4
		0x2	Sample 6

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-109. PRT Configuration Register 1 (PRTC1) Field Descriptions (Continued)

Bit	Name	Value	Description
11	Reserved		Reads return zero and writes have no effect.
10–4	CASM(6–0)*	0x43–0x63 bit times	Collision avoidance symbol max (in bit times). These bits configure the upper limit of the acceptance window for a collision avoidance symbol (CAS). CASM6 is always '1'.
3–0	TSST(3–0)*	0x3–0xF bit times	Transmission start sequence transmitter (in bit times). These bits configure the duration of the transmission start sequence (TSS) in terms of bit times (1 bit time = 4 µT = 100ns @ 10Mbps). Must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.6 PRT Configuration Register 2 (PRTC2)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-125 and Table 17-110 illustrate this register.

Figure 17-125. PRT Configuration Register 2 (PRTC2) [offset_CC = 0x94]

31	30	29	24	23	16
Reserved		TXL(5–0)*		TXI(7–0)*	
R-0	R-0	R/W-Fh		R/W-2Dh	

15	14	13	8	7	6	5	0
Reserved		RXL(5–0)*		Reserved		RXI(5–0)*	
R-0	R-0	R/W-Ah		R-0	R-0	R/W-Eh	

R = Read; W = Write; *-n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-110. PRT Configuration Register 2 (PRTC2) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–24	TXL(5–0)*	0xF–0x3C bit times	Wakeup symbol transmit low (in bit times). These bits configure the active low phase of the wakeup symbol. The duration must be identical in all nodes of a cluster.
23–16	TXI(7–0)*	0x2D–0xB4 bit times	Wakeup symbol transmit idle (in bit times). These bits configure the number of bittimes used by the node to transmit the idle phase of the wakeup symbol. Durations must be identical in all nodes of a cluster.
15–14	Reserved		Reads return zero and writes have no effect.
13–8	RXL(5–0)*	0xA–0x37 bit times	Wakeup symbol receive low (in bit times). These bits configure the number of bit times used by the node to test the duration of the low phase of the received wakeup symbol. Must be identical in all nodes of a cluster.
7–6	Reserved		Reads return zero and writes have no effect.
5–0	RXI(5–0)*	0xE–0x3B bit times	Wakeup symbol receive idle (in bit times). These bits configure the number of bit times used by the node to test the duration of the idle phase of the received wakeup symbol. Must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.7 MHD Configuration Register (MHDC)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-126 and Table 17-111 illustrate this register.

Figure 17-126. MHD Configuration Register (MHDC) [offset_CC = 0x98]

31–29	28–16
Reserved	SLT(12–0)*
R-0	R/W-0

15–7	6–0
Reserved	SFDL(6–0)*
R-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-111. MHD Configuration Register (MHDC) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return zero and writes have no effect.
28–16	SLT(12–0)*	0x0– 0x1F2D minislots	Start of latest transmit (in minislots). These bits configure the maximum minislot value allowed before inhibiting new frame transmissions in the Dynamic Segment of the cycle. There is no transmission in dynamic segment if SLT[12:0] is set to zero.
15–8	Reserved		Reads return zero and writes have no effect.
7–0	SFDL(6–0)*	0x0– 0x7F	Static frame data length. These bits configure the cluster-wide payload length for all frames sent in the static segment in double bytes. The payload length must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.8 GTU Configuration Register 1 (GTUC1)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-127 and Table 17-112 illustrate this register.

Figure 17-127. GTU Configuration Register 1 (GTUC1) [offset_CC = 0xA0]

31–20	19–16
Reserved	UT(19–16)*
R-0	R/W-0

15–0
UT(15–0)*
R/W-0280h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-112. GTU Configuration Register 1 (GTUC1) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads return zero and writes have no effect.
19–0	UT(19–0)*	0x280– 0x9C400 µT	Microtick per cycle (in microticks). These bits configure the duration of the communication cycle in microticks.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.9 GTU Configuration Register 2 (GTUC2)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-128 and Table 17-113 illustrate this register.

Figure 17-128. GTU Configuration Register 2 (GTUC2) [offset_CC = 0xA4]

31	20	19	16
Reserved		SNM(3–0)*	
R-0		R/W-2h	

15	14	13	0
Reserved		MPC(13–0)*	
R-0	R-0	R/W-000Ah	

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-113. GTU Configuration Register 2 (GTUC2) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads return zero and writes have no effect.
19–16	SNM(3–0)*	0x2–0xF frames	Sync node max (in frames). These bits configure the maximum number of frames within a cluster with sync frame indicator bit SYN set. The number of frames must be identical in all nodes of a cluster.
15–14	Reserved		Reads return zero and writes have no effect.
13–0	MPC(13–0)*	0xA–0x3E80 MT	Macrotick per cycle (in macroticks). These bits configure the duration of one communication cycle in macroticks. The cycle length must be identical in all nodes of a cluster.
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only			

17.24.10GTU Configuration Register 3 (GTUC3)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-129 and Table 17-114 illustrate this register.

Figure 17-129. GTU Configuration Register 3 (GTUC3) [offset_CC = 0xA8]

31	30 – 24	23	22 – 16
Reserved	MIOB(6–0)*	Reserved	MIOA(6–0)*
R-0	R/W-02h	R-0	R/W-02h

15 – 8	7 – 0
UIOB(7–0)*	UIOA(7–0)*
R/W-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-114. GTU Configuration Register 3 (GTUC3) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return zero and writes have no effect.
30–24	MIOB(6–0)*	0x2–0x48 MT	Macrotick initial offset channel B (in macroticks). These bits configure the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration. Must be identical in all nodes of a cluster.
23	Reserved		Reads return zero and writes have no effect.
22–16	MIOA(6–0)*	0x2–0x48 MT	Macrotick initial offset channel A (in macroticks). These bits configure the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration. Must be identical in all nodes of a cluster.
15–8	UIOB(7–0)*	0x0–0xF0 µT	Microtick initial offset channel B (in microticks). These bits configure the number of microticks between the actual time reference point on channel B and the subsequent macrotick boundary of the secondary time reference point. The parameter has to be set for each channel independently.
7–0	UIOA(7–0)*	0x0–0xF0 µT	Microtick initial offset channel A (in microticks). These bits configure the number of microticks between the actual time reference point on channel A and the subsequent macrotick boundary of the secondary time reference point. The parameter has to be set for each channel independently.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.11GTU Configuration Register 4 (GTUC4)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-130 and Table 17-115 illustrate this register.

Figure 17-130. GTU Configuration Register 4 (GTUC4) [offset_CC = 0xAC]

31	30	29	28	27	26	25	24	23	22	21	16
Reserved		OCS(13–0)*									
R-0	R-0	R/W-Ah									

15	14	13	0
Reserved		NIT(13–0)*	
R-0	R-0	R/W-9	

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-115. GTU Configuration Register 4 (GTUC4) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads return zero and writes have no effect.
29–16	OCS(13–0)*	0x8– 0x3E7E MT	Offset correction start (in macroticks). These bits determine the start of the offset correction within the NIT phase, calculated from start of cycle. Must be identical in all nodes of a cluster.
15–14	Reserved		Reads return zero and writes have no effect.
13–0	NIT(13–0)*	0x7– 0x3E7D MT	Network idle time start (in macroticks). These bits configure the starting point of the network idle time (NIT) at the end of the communication cycle expressed in terms of macroticks from the beginning of the cycle. The number must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.12GTU Configuration Register 5 (GTUC5)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-131 and Table 17-116 illustrate this register.

Figure 17-131. GTU Configuration Register 5 (GTUC5) [offset_CC = 0xB0]

31	30 24	23 21	20 16
DEC(7–0)*		Reserved	CDD(4–0)*
R-0	R/W-Eh	R-0	R/W-0

15 8	7 0
DCB(7–0)*	DCA(7–0)*
R/W-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-116. GTU Configuration Register 5 (GTUC5) Field Descriptions

Bit	Name	Value	Description
31–24	DEC(7–0)*	0xE–0x8F μT	Decoding correction (in microticks). These bits configure the decoding correction value used to determine the primary time reference point.
23–21	Reserved		Reads return zero and writes have no effect.
20–16	CDD(4–0)*	0x0–0x14 μT	Cluster drift damping (in microticks). These bits configure the cluster drift damping value used in clock synchronization to minimize accumulation of rounding errors.
15–8	DCB(7–0)*	0x0–0xC8 μT	Delay compensation channel B (in microticks). These bits are used to compensate for reception delays on the indicated channel. This compensates propagation delays for microticks in the range of 0.0125 to 0.05s. In practice, the minimum propagation delay of all sync nodes should be applied.
7–0	DCA(7–0)*	0x0-0xC8 μT	Delay compensation channel A (in microticks). These bits are used to compensate for reception delays on the indicated channel. This compensates propagation delays for microticks in the range of 0.0125 to 0.05s. In practice, the minimum propagation delay of all sync nodes should be applied.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.13GTU Configuration Register 6 (GTUC6)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-132 and Table 17-117 illustrate this register.

Figure 17-132. GTU Configuration Register 6 (GTUC6) [offset_CC = 0xB4]

31 … 27	26 … 16
Reserved	MOD(10–0)*
R-0	R/W-2h

15 … 11	10 … 0
Reserved	ASR(10–0)*
R-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-117. GTU Configuration Register 6 (GTUC6) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26–16	MOD(10–0)*	0x2–0x783 µT	Maximum oscillator drift (in microticks). Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle in µT.
15–11	Reserved		Reads return zero and writes have no effect.
10–0	ASR(10–0)*	0x0–0x753 µT	Accepted startup range (in microticks). Number of microticks constituting the expanded range of measured deviation for startup frames during integration.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.14GTU Configuration Register 7 (GTUC7)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-133 and Table 17-118 illustrate this register.

Figure 17-133. GTU Configuration Register 7 (GTUC7) [offset_CC = 0xB8]

31 26	25 16
Reserved	NSS(9–0)*
R-0	R/W-2h*

15 11	10 0
Reserved	SSL(9–0)*
R-0	R/W-4h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-118. GTU Configuration Register 7 (GTUC7) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Reads return zero and writes have no effect.
25–16	NSS(9–0)*	0x2– 0x3FF MT	Number of static slots (in macroticks). These bits configure the number of static slots in a cycle. At least two coldstart nodes must be configured to startup a FlexRay network. The number of static slots must be identical in all nodes of a cluster.
15–11	Reserved		Reads return zero and writes have no effect.
10–0	SSL(9–0)*	0x4– 0x293 MT	Static slot length (in macroticks). These bits configure the duration of a static slot. The static slot length must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.15GTU Configuration Register 8 (GTUC8)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-134 and Table 17-119 illustrate this register.

Figure 17-134. GTU Configuration Register 8 (GTUC8) [offset_CC = 0xBC]

31–29	28–16
Reserved	NMS(12–0)*
R-0	R/W-0

15–6	5–0
Reserved	MSL(5–0)*
R-0	R/W-2h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-119. GTU Configuration Register 8 (GTUC8) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return zero and writes have no effect.
28–16	NMS(12–0)	0x0–0x1F32	Number of minislots. These bits configure the number of minislots in the dynamic segment of a cycle. The number of minislots must be identical in all nodes of a cluster.
15–7	Reserved		Reads return zero and writes have no effect.
5–0	MSL(5–0)	0x2–0x3F MT	Minislot length (in macroticks). These bits configure the duration of a minislot. The minislot length must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.16GTU Configuration Register 9 (GTUC9)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-135 and Table 17-120 illustrate this register.

Figure 17-135. GTU Configuration Register 9 (GTUC9) [offset_CC = 0xC0]

31–18	17–16
Reserved	DSI(1–0)*
R-0	R/W-0

15–13	12–8	7–6	5–0
Reserved	MAPO(4–0)*	Reserved	APO(5–0)
R-0	R/W-1h	R-0	R/W-1h

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-120. GTU Configuration Register 9 (GTUC9) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return zero and writes have no effect.
17–16	DSI(1–0)*	0x0–0x2	Dynamic slot idle phase (in minislots). The duration of the dynamic slot idle phase has to be greater or equal than the idle detection time. Must be identical in all nodes of a cluster.
15–13	Reserved		Reads return zero and writes have no effect.
12–8	MAPO(4–0)*	0x1–0x1F MT	Minislot action point offset (in macroticks). These bits configure the minislot action point offset within the minislots of the dynamic segment. The minislot action point offset must be identical in all nodes of a cluster.
7–6	Reserved		Reads return zero and writes have no effect.
5–0	APO(5–0)*	0x1–0x3F MT	Action point offset (in macroticks). These bits configure the action point offset within static slots and symbol window. The action point offset must be identical in all nodes of a cluster.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.17GTU Configuration Register 10 (GTUC10)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-136 and Table 17-121 illustrate this register.

Figure 17-136. GTU Configuration Register 10 (GTUC10) [offset_CC = 0xC4]

31 27	26 16
Reserved	MRC(10–0)*
R-0	R/W-2h

15	13 0
Reserved	MOC(13–0)
R-0	R/W-5

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-121. GTU Configuration Register 10 (GTUC10) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26–16	MRC(10–0)	0x2–0x783 µT	Maximum rate correction (in microticks). Holds the maximum permitted rate correction value to be applied by the internal clock synchronization algorithm. The communication controller checks the internal rate correction value against the maximum rate correction value (absolute value).
15–14	Reserved		Reads return zero and writes have no effect.
13–0	MOC(13–0)	0x5–0x3BA2 µT	Maximum offset correction (in microticks). Holds the maximum permitted offset correction value to be applied by the internal clock synchronization algorithm (absolute value). The communication controller checks the internal offset correction value against the maximum offset correction value.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.24.18GTU Configuration Register 11 (GTUC11)

Figure 17-137 and Table 17-122 illustrate this register.

Figure 17-137. GTU Configuration Register 11 (GTUC11) [offset_CC = 0xC8]

31 27	26 24	23 19	18 16
Reserved	ERC(2–0)*	Reserved	EOC(2–0)*
R-0	R/W-0	R-0	R/W-0

15 10	9 8	7 2	1 0
Reserved	ERCC(1-0)	Reserved	EOCC(1–0)*
R-0	R/W-0	R-0	R/W-0

R = Read; W = Write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-122. GTU Configuration Register 11 (GTUC11) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads return zero and writes have no effect.
26–24	ERC(2–0)*	0x0–0x7 µT	External rate correction (in microticks). Holds the external clock rate correction value to be applied by the internal clock synchronization algorithm. The value is subtracted/added from/to the calculated rate correction value. The value is applied during NIT. May be modified in DEFAULT_CONFIG or CONFIG state only.
23–19	Reserved		Reads return zero and writes have no effect.
18–16	EOC(2–0)*	0x0–0x7 µT	External offset correction (in microticks). Holds the external clock offset correction value to be applied by the internal clock synchronization algorithm. The value is subtracted/added from/to the calculated offset correction value. The value is applied during NIT. May be modified in DEFAULT_CONFIG or CONFIG state only.
15–10	Reserved		Reads return zero and writes have no effect.
9–8	ERCC(1–0)*		External rate correction control. By writing to ERCC(1–0), the external rate correction is enabled as specified below. Should be modified only outside NIT.
		0x0, 0x1	No external rate correction
		0x2	External rate correction value subtracted from calculated rate correction value
		0x3	External rate correction value added to calculated rate correction value
7–2	Reserved		Reads return zero and writes have no effect.
1–0	EOCC(1–0)*		External offset correction control. By writing to EOCC(1–0), the external offset correction is enabled as specified below. Should be modified only outside NIT.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-122. GTU Configuration Register 11 (GTUC11) Field Descriptions (Continued)

Bit	Name	Value	Description
		0x0, 0x1	No external offset correction
		0x2	External offset correction value subtracted from calculated offset correction value
		0x3	External offset correction value added to calculated offset correction value

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.25 Status Registers

During 8/16-bit accesses to status variables coded with more than 8/16-bit, the variable might be updated by the communication controller between two accesses (non-atomic read accesses). All internal counters and the communication controller status flags are reset when the communication controller transits from CONFIG to READY state.

17.25.1 Communication Controller Status Vector (CCSV)

Figure 17-138 and Table 17-123 illustrate this register.

Figure 17-138. Communication Controller Status Vector Register (CCSV) [offset_CC = 0x100]

31 30	29 24	23 19	18 16
Reserved	PSL(5-0)	RCA(4–0)	WSV(2–0)
R-0	R-0	R-2	R-0

15	14	13	12	11 10	9 8	7	6	5 0
Reserve d	CSI	CSAI	CSNI	Reserved	SLM(1–0)	HQR	FSI	POCS(5–0)
R-0	R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; W = Write; -*n* = Value after reset

Table 17-123. Communication Controller Status Vector Register (CCSV) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads are zeros and writes have no effect
29–24	PSL(5–0)		POC Status Log. Status of POCS[5:0] immediately before entering HALT state. Set when entering HALT state. Set to HALT when FREEZE command is applied during HALT state and FSI is not already set i.e. the HALT state was not reached by FREEZE command. Reset to "00 0000" when leaving HALT state.
23–19	RCA(4–0)	0x0–0x1F	Remaining coldstart attempts. Indicates the number of remaining coldstart attempts. The maximum number of coldstart attempts is configured by CSA(4–0) in the SUC configuration register 1.

Table 17-123. Communication Controller Status Vector Register (CCSV) Field Descriptions (Continued)

Bit	Name	Value	Description
18–16	WSV(2–0)		Wakeup status. Indicates the status of the current wakeup attempt. Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to EFAULT_CONFIG state.
		0x0	UNDEFINED. No wakeup attempt since CONFIG state was left.
		0x1	RECEIVED_HEADER. Set when the communication controller finishes wakeup due to the reception of a frame header without coding violation on either channel in WAKEUP_LISTEN or WAKEUP_DETECT state.
		0x2	RECEIVED_WUP. Set when the communication controller finishes wakeup due to the reception of a valid wakeup pattern on the configured wakeup channel in WAKEUP_LISTEN or WAKEUP_DETECT state.
		0x3	COLLISION_HEADER. Set when the communication controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid header on either channel.
		0x4	COLLISION_WUP. Set when the communication controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid wakeup pattern on the configured wakeup channel.
		0x5	COLLISION_UNKNOWN. Set when the communication controller stops wakeup by leaving WAKEUP_DETECT state after expiration of the wakeup timer without receiving a valid wakeup pattern or a valid frame header.
		0x6	TRANSMITTED. Set when the communication controller has successfully completed the transmission of the wakeup pattern.
		0x7	Reserved
14	CSI		Cold start inhibit. Indicates that the node is disabled from cold starting. The flag is set whenever the POC enters READY state. The flag has to be reset under control of the host by the controller host interface command ALLOW_COLDSTART (CMD(3–0) = 1001).
		0	Cold starting of node enabled
		1	Cold starting of node disabled
13	CSAI		Coldstart abort indicator. Coldstart aborted. Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.
12	CSNI		Coldstart noise indicator. Indicates that the cold start procedure occurred under noisy conditions. Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.
11–10	Reserved		Reads are zeros and writes have no effect

Table 17-123. Communication Controller Status Vector Register (CCSV) Field Descriptions (Continued)

Bit	Name	Value	Description
9–8	SLM(1–0)		Slot mode. Indicates the actual slot mode of the POC in states READY, WAKEUP, STARTUP, NORMAL_ACTIVE, and NORMAL_PASSIVE. Default is SINGLE. Changes to ALL, depending on SUCC1.TSM. In NORMAL_ACTIVE or NORMAL_PASSIVE state the CHI command ALL_SLOTS will change the slot mode from SINGLE over ALL_PENDING to ALL. Set to SINGLE in all other states.
		0x0	SINGLE
		0x1	Reserved
		0x2	ALL_PENDING
		0x3	ALL
7	HRQ	0–1	Halt request. Indicates that a request from the Host has been received to halt the POC at the end of the communication cycle. Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or when entering READY state.
6	FSI	0–1	Freeze status indicator. Indicates that the POC has entered the HALT state due to CHI command FREEZE or due to an error condition requiring an immediate POC halt. Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state.
5–0	POCS(5–0)		Protocol operation control status. **Indicates the actual state of operation of the Communication Controller Protocol Operation Control:**
		0x0	DEFAULT_CONFIG state
		0x1	READY state
		0x2	NORMAL_ACTIVE state
		0x3	NORMAL_PASSIVE state
		0x4	HALT state
		0x5	MONITOR_MODE state
		0x6–0xC	Reserved
		0xD	LOOPBACK MODE state
		0xE	Reserved
		0xF	CONFIG state
			Indicates the actual state of operation of the POC in the wakeup path:
		0x10	WAKEUP_STANDBY state
		0x11	WAKEUP_LISTEN state
		0x12	WAKEUP_SEND state
		0x13	WAKEUP_DETECT state

Table 17-123. Communication Controller Status Vector Register (CCSV) Field Descriptions (Continued)

Bit	Name	Value	Description
		0x14–0x1F	Reserved
			Indicates the actual state of operation of the POC in the startup path:
		0x20	STARTUP_PREPARE state
		0x21	COLDSTART_LISTEN state
		0x22	COLDSTART_COLLISION_RESOLUTION state
		0x23	COLDSTART_CONSISTENCY_CHECK state
		0x24	COLDSTART_GAP state
		0x25	COLDSTART_JOIN state
		0x26	INTEGRATION_COLDSTART_CHECK state
		0x27	INTEGRATION_LISTEN state
		0x28	INTEGRATION_CONSISTENCY_CHECK state
		0x29	INITIALIZE_SCHEDULE state
		0x2A	ABORT_STARTUP state
		0x2B–0x3F	Reserved

Note:
CHI command RESET_STATUS_INDICATORS (SUCC1.CMD[3:0] = "1010") resets flags FSI, HRQ, CSNI, CSAI, the slot mode SLM[1:0], and the wakeup status WSV[2:0]

17.25.2 .Communication Controller Error Vector (CCEV)

Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

Figure 17-139 and Table 17-124 illustrate this register.

Figure 17-139. Communication Controller Error Vector Register (CCEV) [offset_CC = 0x104]

31–16
Reserved
R-0

15–13	12–8	7–6	5–4	3–0
Reserved	PTAC(4–0)	ERRM(1–0)	Reserved	CCFC(3–0)
R-0	R-0	R-0	R-0	R-0

R = Read; -*n* = Value after reset

Table 17-124. Communication Controller Error Vector Register (CCEV) Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads are zeros and writes have no effect
12–8	PTAC(4–0)	0x0–0x1F	Passive to active count. Indicates the number of consecutive even / odd cycle pairs that have passed with valid rate and offset correction terms, while the node is waiting to transit from NORMAL_PASSIVE state to NORMAL_ACTIVE state. The transition takes place when PTAC(4–0) equals PTA(4–0)-1 as defined in the SUC configuration register 1
7–6	ERRM(1–0)		Error mode. Indicates the actual error mode of the POC.
		0x0	ACTIVE
		0x1	PASSIVE
		0x2	COMM_HALT
		0x3	Reserved
5–4	Reserved		Reads are zeros and writes have no effect
3–0	CCFC(3–0)	0x0–0x15	Clock correction failed counter. The clock correction failed counter is incremented by one at the end of any odd communication cycle where either the missing offset correction error or missing rate correction error are active. The clock correction failed counter is reset to 0 at the end of an odd communication cycle if neither the offset correction failed nor the rate correction failed errors are active. The clock correction failed counter stops at 15.

17.25.3 Slot Counter Value (SCV)

This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state. Figure 17-140 and Table 17-125 illustrate this register.

Figure 17-140. Slot Counter Vector Register (SCV) [offset_CC = 0x110]

31 27	26 16
Reserved	SCCB(10–0)
R-0	R-0

15 11	10 0
Reserved	SCCA(10–0)
R-0	R-0

R = Read; W = Write; -*n* = Value after reset

Table 17-125. Slot Counter Vector Register (SCV) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads are zeros and writes have no effect
26–16	SCCB(10–0)	0x1–0x7FF	Slot counter channel B. Current slot counter value channel B. The value is incremented by the communication controller and reset at the start of a communication cycle.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	SCCA(10–0)	0x1–0x7FF	Slot counter channel A. Current slot counter value channel A. The value is incremented by the communication controller and reset at the start of a communication cycle.

17.25.4 Macrotick and Cycle Counter Value (MTCCV)

Figure 17-141 and Table 17-126 illustrate this register.

Figure 17-141. Macrotick and Cycle Counter Register (MTCCV) [offset_CC = 0x114]

31–22	21–16
Reserved	CCV(5–0)
R-0	R-0

15–14	13–0
Reserved	MTV(13–0)
R-0	R-0

R = Read; -*n* = Value after reset

Table 17-126. Macrotick and Cycle Counter Register (MTCCV) Field Descriptions

Bit	Name	Value	Description
31–22	Reserved		Reads are zeros and writes have no effect
21–16	CCV(5–0)	0x0–0x3F	Cycle counter value. Current cycle counter value. The value is incremented by the communication controller at the start of a communication cycle.
15–14	Reserved		Reads are zeros and writes have no effect
13–0	MTV(13–0)	0x0–0x3E80	Macrotick value. Current macrotick value. The value is incremented by the communication controller and reset at the start of a communication cycle.

17.25.5 Rate Correction Value (RCV)

This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state. Figure 17-142 and Table 17-127 illustrate this register.

Figure 17-142. Rate Correction Value Register (RCV) [offset_CC = 0x118]

31	16
Reserved	
R-0	

15	12	11	0
Reserved		RCV(11–0)	
R-0		R-0	

R = Read; *-n* = Value after reset

Table 17-127. Rate Correction Value Register (RCV) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads are zeros and writes have no effect
11–0	RCV(11–0)		Rate correction value (in microticks). Rate correction value (two's complement). Calculated internal rate correction value before limitation. If the RCV value exceeds the limits defined by GTUC10.MRC[10:0], flag SFS.RCLR is set to '1'.

Note:
The external rate correction value is added to the limited rate correction value.

17.25.6 Offset Correction Value (OCV)

Figure 17-143 and Table 17-128 illustrate this register.

Figure 17-143. Offset Correction Value Register (OCV) [offset_CC = 0x11C]

31 20	19 16
Reserved	OVC[19:16]
R-0	R-0

15 0
OCV(15–0)
R-0

R = Read; -*n* = Value after reset

Table 17-128. Offset Correction Value Register (OCV) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads are zeros and writes have no effect
19–0	OCV(19–0)		Offset correction value (in microticks). Offset correction value (two's complement). Calculated internal offset correction value before limitation. If the OCV value exceeds the limits defined by GTUC10.MOC[13:0], flag SFS.OCLR is set to '1'

Note:
The external offset correction value is added to the limited offset correction value.

17.25.7 Sync Frame Status (SFS)

This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state. Figure 17-144 and Table 17-129 illustrate this register.

Figure 17-144. Sync Frame Status Register (SFS) [offset_CC = 0x120]

31 20	19	18	17	16
	RCLR	MRCS	OCLR	MOCS
R-0	R-0	R-0	R-0	R-0

15 12	11 8	7 4	3 0
VSBO(3–0)	VSBE(3–0)	VSAO(3–0)	VSAE(3–0)
R-0	R-0	R-0	R-0

R = Read; -*n* = Value after reset

Table 17-129. Sync Frame Status Register (SFS) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Reads are zeros and writes have no effect
19	RCLR		Rate correction limit reached. The Rate Correction Limit Reached flag signals to the Host, that the rate correction value has exceeded its limit as defined by GTUC10.MRC[10:0]. The flag is updated by the communication controller at start of offset correction phase.
		0	Rate correction below limit
		1	Rate correction limit reached
18	MRCS		Missing rate correction signal. The missing rate correction signal signals to the host that no rate correction can be performed because no pairs of even/odd sync frames were received. The flag is updated by the communication controller at start of offset correction phase.
		0	Rate correction signal valid
		1	Missing rate correction signal
17	OCLR		Offset correction limit reached. The offset correction limit reached flag signals to the host that the offset correction value has reached its limit as defined by GTUC10.MOC[13:0]. The flag is updated by the communication controller at start of offset correction phase.
		0	Offset correction below limit
		1	Offset correction limit reached
16	MOCS		Missing offset correction signal. The missing offset correction signal signals to the host that no rate correction can be performed because no pairs of even / odd sync frames were received. The flag is updated by the communication controller at start of offset correction phase.
		0	Offset correction signal valid
		1	Missing offset correction signal

Table 17-129. Sync Frame Status Register (SFS) Field Descriptions (Continued)

Bit	Name	Value	Description
15–12	VSBO(3–0)	0x0–0xF	Valid sync frames channel B, odd communication cycle. Holds the number of valid sync frames received on channel B in the odd communication cycle. If transmission of sync frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each odd communication cycle.
11–8	VSBE(3–0)	0x0–0xF	Valid synch frames channel B, even communication cycle. Holds the number of valid sync frames received and transmitted on channel B in the even communication cycle. If transmission of sync frames is enabled by SUCC1.TXSY the value is incremented by one.The value is updated during the NIT of each even communication cycle.
7–4	VSAO(3–0)	0x0–0xF	Valid synch frames channel A, odd communication cycle. Holds the number of valid sync frames received and transmitted on channel A in the odd communication cycle. If transmission of sync frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each odd communication cycle.
3–0	VSAE(3–0)	0x0–0xF	Valid synch frames channel A, even communication cycle. Holds the number of valid sync frames received and transmitted on channel A in the even communication cycle. If transmission of sync frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the NIT of each even communication cycle.

Note:
The bit fields VSBO(3–0), VSBE(3–0), VSAO(3–0), VSAE(3–0) are only valid if the respective channel is assigned to the communication controller bySUCC1.CCHA or SUCC1.CCHB.

17.25.8 Symbol Window and NIT Status (SWNIT)

Symbol window related status information. Updated by the communication controller at the end of the symbol window for each channel. During startup the status data is not updated. This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state.

Figure 17-145 and Table 17-130 illustrate this register.

Figure 17-145. Symbol Window and NIT Status Register (SWNIT) [offset_CC = 0x124]

31–16
Reserved
R-0

15–12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SBNB	SENB	SBNA	SENA	MTSB	MTSA	TCSB	SBSB	SESB	TCSA	SBSA	SESA
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; -*n* = Value after reset

Table 17-130. Symbol Window and NIT Status Register (SWNIT) Field Descriptions

Bit	Name	Value	Description
31–12	Reserved		Reads are zeros and writes have no effect
11	SBNB		Slot boundary violation during NIT channel B.
		0	No slot boundary violation detected
		1	Slot boundary violation during NIT detected on channel B
10	SENB		Syntax error during NIT channel B.
		0	No syntax error detected
		1	Syntax error during NIT detected on channel B
9	SBNA		Slot boundary violation during NIT channel A.
		0	No slot boundary violation detected
		1	Slot boundary violation during NIT detected on channel A
8	SENA		Syntax error during NIT channel A.
		0	No syntax error detected
		1	Syntax error during NIT detected on channel A
7	MTSB		MTS Received on Channel B. Media Access Test symbol received on channel B during the last symbol window. Updated by the communication controller for each channel at the end of the symbol window. When this bit is set to ’1’, also interrupt flag SIR.MTSB is set to ’1’.
		0	No MTS symbol received on channel B
		1	MTS symbol received on channel B

Table 17-130. Symbol Window and NIT Status Register (SWNIT) Field Descriptions (Continued)

Bit	Name	Value	Description
6	MTSA		MTS Received on Channel A. Media Access Test symbol received on channel A during the last symbol window. Updated by the communication controller for each channel at the end of the symbol window. When this bit is set to ’1’, also interrupt flag SIR.MTSA is set to ’1’.
		0	No MTS symbol received on channel A
		1	MTS symbol received on channel A
5	TCSB		Transmission conflict in symbol window channel B.
		0	No transmission conflict detected
		1	Transmission conflict in symbol window detected on channel B
4	SBSB		Slot boundary violation in symbol window channel B.
		0	No slot boundary violation detected
		1	Slot boundary violation during symbol window detected on channel B
3	SESB		Syntax error in symbol window channel B.
		0	No syntax error detected
		1	Syntax error during symbol window detected on channel B
2	TCSA		Transmission conflict in symbol window channel A.
		0	No transmission conflict detected
		1	Transmission conflict in symbol window detected on channel A
1	SBSA		Slot boundary violation in symbol window channel A.
		0	No slot boundary violation detected
		1	Slot boundary violation during symbol window detected on channel A
0	SESA		Syntax error in symbol window channel A.
		0	No syntax error detected
		1	Syntax error during symbol window detected on channel A

17.25.9 Aggregated Channel Status (ACS)

The aggregated channel status provides the host with an accrued status of channel activity for all communication slots regardless of whether they are assigned for transmission or subscribed for reception. The aggregated channel status also includes status data from the symbol phase and the network idle time. The status data is updated (set) after each slot and aggregated until it is reset by the host. During startup the status data is not updated. A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect on the flag. This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state.

Figure 17-146 and Table 17-131 illustrate this register.

Figure 17-146. Aggregated Channel Status Register (ACS) [offset_CC = 0x128]

31–16
Reserved
R-0

15–13	12	11	10	9	8	7–5	4	3	2	1	0
Reserved	SBVB	CIB	CEDB	SEDB	VFRB	Reserved	SBVA	CIA	CEDA	SEDA	VFRA
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read; W = Write; -*n* = Value after reset

Table 17-131. Aggregated Channel Status Register (ACS) Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads are zeros and writes have no effect
12	SBVB		Slot boundary violation on channel B. One or more slot boundary violations were observed on channel B at any time during the observation period (static or dynamic slots including symbol window and NIT).
		0	No slot boundary violation observed
		1	Slot boundary violation(s) observed on channel B
11	CIB		Communication indicator channel B. One or more valid frames were received on channel B in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.
		0	No valid frame(s) received in slots containing any additional communication
		1	Valid frame(s) received on channel B in slots containing any additional communication
10	CEDB		Content error detected on channel B. One or more frames with a content error were received on channel B in any static or dynamic slot during the observation period.
		0	No frame with content error received
		1	Frame(s) with content error received on channel B

Table 17-131. Aggregated Channel Status Register (ACS) Field Descriptions (Continued)

Bit	Name	Value	Description
9	SEDB		Syntax error detected on channel B. One or more syntax errors in static or dynamic slots including symbol window and NIT were observed on channel B.
		0	No syntax error observed
		1	Syntax error(s) observed on channel B
8	VFRB		Valid frame received on channel B. One or more valid frames were received on channel B in any static or dynamic slot during the observation period. Reset is under control of the host.
		0	No valid frame received
		1	Valid frame(s) received on channel B
7–5		0	
4	SBVA		Slot boundary violation on channel A. One or more slot boundary violations were observed on channel A at any time during the observation period (static or dynamic slots including symbol window and NIT).
		0	No slot boundary violation observed
		1	Slot boundary violation(s) observed on channel A
3	CIA		Communication indicator channel A. One or more valid frames were received on channel A in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.
		0	No valid frame(s) received in slots containing any additional communication
		1	Valid frame(s) received on channel A in slots containing any additional communication
2	CEDA		Content error detected on channel A. One or more frames with a content error were received on channel A in any static or dynamic slot during the observation period.
		0	No frame with content error received
		1	Frame(s) with content error received on channel A
1	SEDA		Syntax error detected on channel A. One or more syntax errors in static or dynamic slots including symbol window and NIT were observed on channel A.
		0	No syntax error observed
		1	Syntax error(s) observed on channel A

Table 17-131. Aggregated Channel Status Register (ACS) Field Descriptions (Continued)

Bit	Name	Value	Description
0	VFRA		Valid frame received on channel A. One or more valid frames were received on channel A in any static or dynamic slot during the observation period.
		0	No valid frame received
		1	Valid frame(s) received on channel A

17.25.10Even Sync ID [1. . .15] (ESIDn)

Registers ESID1 to ESID15 hold the frame IDs of the sync frames received in **even** communication cycles, assorted in ascending order, with register ESID1 holding the lowest received sync frame ID. If the node transmits a sync frame in an even communication cycle by itself, register ESID1 holds the respective sync frame ID as configured in message buffer 0. The value is updated during the NIT of each even communication cycle. This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state.

Figure 17-147 and Table 17-132 illustrate these registers.

Figure 17-147. Even Sync ID Register [1. . .15] (ESIDn) [offset_CC = 0x130–0x168]

31–16
Reserved
R-0

15	14	13–10	9–0
RXEB	RXEA	Reserved	EID(9–0)
R-0	R-0	R-0	R-0

R = Read; -*n* = Value after reset

Table 17-132. Even Sync ID Register [1...15] (ESIDn) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads are zeros and writes have no effect
15	RXEB		Received even sync ID on channel B. A sync frame corresponding to the stored even sync ID was received on channel B.
		0	Sync frame not received on channel B
		1	Sync frame received on channel B
14	RXEA		Received even sync ID on channel A. A sync frame corresponding to the stored even sync ID was received on channel A.
		0	Sync frame not received on channel A
		1	Sync frame received on channel A
13–10	Reserved		Reads are zeros and writes have no effect
9–0	EID(9–0)		Even Sync ID. Sync frame ID even communication cycle.

17.25.11Odd Sync ID [1 . . .15] (OSIDn)

Registers OSID1 to OSID15 hold the frame IDs of the sync frames received in **odd** communication cycles, assorted in ascending order, with register OSID1 holding the lowest received sync frame ID. If the node transmits a sync frame in an odd communication cycle by itself, register OSID1 holds the respective sync frame ID as configured in message buffer 0. The value is updated during the NIT of each odd communication cycle. This register is reset when the Communication Controller leaves CONFIG state or enters STARTUP state.

Figure 17-148 and Table 17-133 illustrate these registers.

Figure 17-148. Odd Sync ID Register [1. . .15] (OSIDn) [offset_CC = 0x170–0x1A8]

31		16
	R-0	R-0

15	14	13	10	9	0
RXOB	RXOA	Reserved		OID(9–0)	
R-0	R-0	R-0		R-0	

R = Read; -*n* = Value after reset

Table 17-133. Odd Sync ID Register [1. . .15] (OSIDn) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads are zeros and writes have no effect
15	RXOB		Received odd sync ID on channel B. A sync frame corresponding to the stored even sync ID was received on channel B.
		0	Sync frame not received on channel B
		1	Sync frame received on channel B
14	RXOA		Received odd sync ID on channel A. A sync frame corresponding to the stored even sync ID was received on channel A.
		0	Sync frame not received on channel A
		1	Sync frame received on channel A
13–10	Reserved		Reads are zeros and writes have no effect
9–0	OID(9–0)		Odd sync ID. Sync frame ID odd communication cycle.

17.25.12Network Management Vector [1. . .3] (NMVn)

The three network management registers hold the accrued NM vector (configurable 0–12 bytes). The accrued NM vector is generated by the communication controller by bit-wise ORing each NM vector received (valid frames with PPI = 1) on each channel.

The communication controller updates the NM vector at the end of each communication cycle as long as the communication controller is either in NORMAL_ACTIVE or NORMAL_PASSIVE state.

NMVn-bytes exceeding the configured NM vector length are not valid.

Figure 17-149 illustrates these registers and Table 17-134 shows the assignment of the data bytes to the network management vector.

Figure 17-149. Network Management Register [1. . .3] (NMVn) [offset_CC = 0x1B0–0x1B8]

31	16
NMI[31:16]	
R-0	

15	0
NMI(15–0)	
R-0	

R = Read; -*n* = Value after reset

Table 17-134. Assignment of Data Bytes to Network Management Vector

Bit **Word**	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMV1	Data3								Data2								Data1								Data0							
NMV2	Data7								Data6								Data5								Data4							
NMV3	Data11								Data10								Data9								Data8							

17.26 Message Buffer Control Registers

17.26.1 Message RAM Configuration (MRC)

The message RAM Configuration register defines the number of message buffers assigned to the static segment, dynamic segment, and FIFO. The register can be written during DEFAULT_CONFIG or CONFIG state only.

Figure 17-150 and Table 17-135 illustrate this register.

Figure 17-150. Message RAM Configuration Register (MRC) [offset_CC = 0x300]

31 27	26	24	23 16
Reserved	SPLM	SEC(1-0)	LCB(7–0)*
R-0	R-1	R-0	R/W-80h

15 8	7 0
FFB(7–0)*	FDB(7–0)*
R/W-0	R/W-0

R = Read; W = write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-135. Message RAM Configuration Register (MRC) Field Descriptions

Bit	Name	Value	Description
31–27	Reserved		Reads are zeros and writes have no effect
26	SPLM*		Sync Frame Payload Multiplex. This bit is only evaluated if the node is configured as sync node (SUCC1.TXSY = ’1’) or for single slot mode operation (SUCC1.TSM = ’1’). When this bit is set to ’1’ message buffers 0 and 1 are dedicated for sync frame transmission with different payload data on channel A and B. When this bit is set to ’0’, sync frames are transmitted from message buffer 0 with the same payload data on both channels. Note that the channel filter configuration for message buffer 0 resp. message buffer 1 has to be chosen accordingly.
		0	Only message buffer 0 locked against reconfiguration
		1	Both message buffers 0 and 1 are locked against reconfiguration
25–24	SEC(1–0)*		Secure Buffers. Not evaluated when the communication controller is in DEFAULT_CONFIG or CONFIG state.
		0x00	Reconfiguration of message buffers enabled with numbers < 0xFFB enabled Exception: In nodes configured for sync frame transmission or for single slot mode operation message buffer 0 (and if SPLM = ’1’, also message buffer 1) is always locked
		0x01	Reconfiguration of message buffers with numbers < FDB and with numbers FFB locked and transmission of message buffers for static segment with numbers FDB disabled
		0x10	Reconfiguration of all message buffers locked

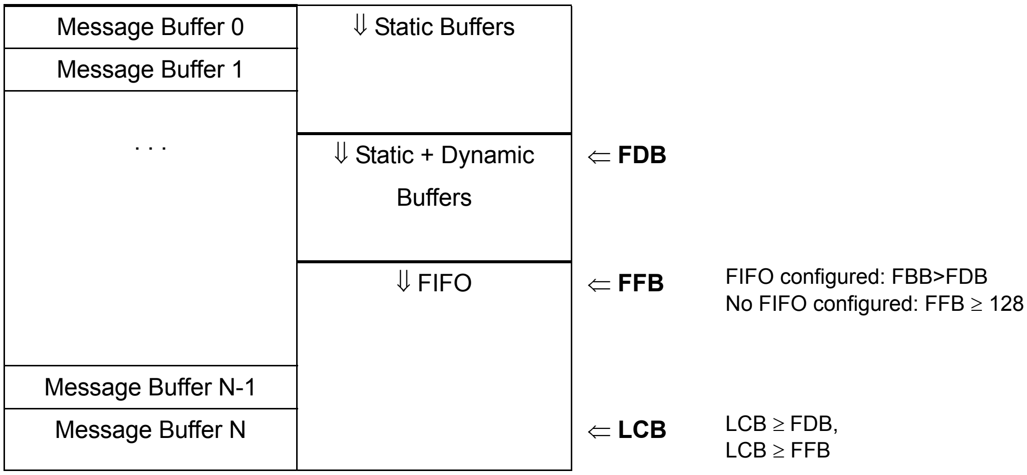
Table 17-135. Message RAM Configuration Register (MRC) Field Descriptions (Continued)

Bit	Name	Value	Description
		0x11h	Reconfiguration of all message buffers locked and transmission of message buffers for static segment with numbers FDB disabled
23–16	LCB(7–0)*		Last configured buffer.
		0x0...x07F	Number of message buffers is LCB + 1
		> 0x80	No message buffer configured
15–8	FFB(7–0)*		First buffer of FIFO.
		0x0	All message buffers assigned to the FIFO
		0x0...0x7F	Message buffers from FFB to LCB assigned to the FIFO
		≥ 0x80	No message buffer assigned to the FIFO
7–0	FDB(7–0)*		First dynamic buffer.
		0x0	No group of pure static buffers configured
		0x0...0x7F	Message buffers 0 to FDB - 1 reserved for static segment
		≥ 0x80	No dynamic buffers configured

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Note:
In case the node is configured as sync node (SUCC1.TXSY = ’1’) or for single slot mode operation(SUCC1.TSM = ’1’), message buffer 0 resp. 1 is reserved for sync frames or single slot frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation message buffer 0 resp. 1 is treated like all other message buffers.

Figure 17-151. Buffer Configuration



The programmer must ensure that the configuration defined by FDB(7–0), FFB(7–0), and LCB(7–0) is valid.

Note:
The communication controller does not check for erroneous configurations.

Note: Maximum Number of Header Sections
The maximum number of header sections is 128. This means a maximum of 128 message buffers can be configured. The maximum length of the data sections is 254 bytes. The length of the data section may be configured different for each message buffer.

In case two or more message buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the "Static Buffers" or at the beginning of the "Static + Dynamic Buffers" section. The FlexRay protocol specification requires that each node has to send a frame in its key slot. Therefore at least message buffer 0 is reserved for transmission in the key slot. Due to this requirement a maximum number of 127 message buffers can be assigned to the FIFO. Nevertheless, a non protocol conform configuration without a transmission slot in the static segment would still be operational.

The payload length configured and the length of the data sections need to be configured identical for all message buffers belonging to the FIFO via WRHS2.PLC[6:0] and WRHS3.DP[10:0].

When the communication controller is not in DEFAULT_CONFIG or CONFIG state reconfiguration of message buffers belonging to the FIFO is locked.

17.26.2 FIFO Rejection Filter (FRF)

The FIFO rejection filter defines a user specified sequence of bits with which channel, frame ID, and cycle count of the incoming frames are compared. Together with the FIFO rejection filter mask (FRFM), this register determines whether a message is rejected by the FIFO. The FRF register can be written during DEFAULT_CONFIG or CONFIG state only.

Figure 17-152 and Table 17-136 illustrate this register.

Figure 17-152. FIFO Rejection Filter Register (FRF) [offset_CC = 0x304]

31 25	24	23	22 16
Reserved	RNF	RSS	CYF(6–0)
R-0	R-1	R-1	R/W-0

15 13	12 2	1 0
Reserved	FID(10–0)	CH(1–0)
R-0	R/W-0	R/W-0

R = Read; W = write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-136. FIFO Rejection Filter Register (FRF) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads are zeros and writes have no effect
24	RNF*		Reject null frames. If this bit is set, received null frames are not stored in the FIFO.
		0	Null frames are stored in the FIFO
		1	Reject all null frames
23	RSS		Reject in static segment. If this bit is set, the FIFO is used only for the dynamic segment.
		0	FIFO also used in static segment
		1	Reject messages in static segment
22–16	CYF(6–0)*		Cycle counter filter. The 7-bit cycle counter filter determines the cycle set to which the frame ID FIFO rejection filter and the channel FIFO rejection filter are applied. In cycles not belonging to the cycle set specified by CYF[6:0], all frames are rejected. For details about the configuration of the cycle counter filter.
15–13	Reserved		Reads are zeros and writes have no effect
12–2	FID(10–0)*	0x0– 0x7FF	Frame ID filter. A frame ID filter value of zero means that no frame ID is rejected.

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-136. FIFO Rejection Filter Register (FRF) Field Descriptions (Continued)

Bit	Name	Value	Description
1–0	CH(1–0)*		Channel filter. **Note: If reception on both channels is configured, also in the static segment both frames (from channel A and B) are always stored in the FIFO, even if they are identical.**
		0x0	Receive on both channels
		0x2	Receive only on channel B
		0x3	Receive only on channel A
		0x4	No reception

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.26.3 FIFO Rejection Filter Mask (FRFM)

The FIFO rejection filter mask specifies which of the corresponding frame ID filter bits are relevant for rejection filtering. If a bit is set, it indicates that the state of the corresponding bit in the FRF register will not be considered for rejection filtering. The FRFM register can be written during DEFAULT_CONFIG or CONFIG state only.

Figure 17-153 and Table 17-137 illustrate this register.

Figure 17-153. FIFO Rejection Filter Mask Register (FRFM) [offset_CC = 0x308]

31–16
Reserved
R-0

15–13	12–2	1	0
Reserved	MFID(10–0)*	Reserved	
R-0	R/W-0	R-0	R-0

R = Read; W = write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-137. FIFO Rejection Filter Mask Register (FRFM) Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads are zeros and writes have no effect
12–2	MFID(10–0)*		Mask Frame ID Filter
		0	Corresponding frame ID filter bit is used for rejection filtering.
		1	Ignore corresponding frame ID filter bit.
1–0	Reserved		Reads are zeros and writes have no effect

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.26.4 FIFO Critical Level (FCL)

The communication controller accepts modifications of the register in DEFAULT_CONFIG or CONFIG state only.

Figure 17-153 and Table 17-137 illustrate this register.

Figure 17-154. FIFO Critical Level Register (FCL) [offset_CC = 0x30C]

31 – 16
Reserved
R-0

15 – 8	7 – 0
Reserved	CL(7-0)
R-0	R/W-80h

R = Read; W = write; -*n* = Value after reset
*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

Table 17-138. FIFO Critical Level Register (FCL) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are zeros and writes have no effect
7–0	CL(7–0)*		Critical Level. When the receive FIFO fill level FSR.RFFL[7:0] is equal or greater than the critical level configured by CL[7:0], the receive FIFO critical level flag FSR.RFCL is set. If CL[7:0] is programmed to values > 128, bit FSR.RFCL is never set. When FSR.RFCL changes from ’0’ to’1’ bit SIR.RFCL is set to ’1’, and if enabled, an interrupt is generated

*These bits can be updated in DEFAULT_CONFIG or CONFIG state only

17.27 Message Buffer Status Registers

17.27.1 Message Handler Status (MHDS)

A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. A hardware reset will also clear the register.

Figure 17-155 and Table 17-139 illustrate this register.

Figure 17-155. Message Handler Status (MHDS) [offset_CC = 0x310]

31	30–24	23	22–16
Reserved	MBU(6–0)	Reserved	MBT(6–0)
R-0	R-0	R-0	R-0

15	14–8	7	6	5	4	3	2	1	0
Reserved	FMB(6–0)	CRAM	MFMB	FMBD	PTFB2	PTFB1	PMR	POBF	PIBF
R-0	R-0	R-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-139. Message Handler Status (MHDS) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads are zeros and writes have no effect
30–24	MBU(6–0)	0x0–0x7F	Message buffer updated. Number of the message buffer that was updated last by the communication controller. For this message buffer, the respective ND and/or MBC flag in the new data 1...4 (NDAT1...4) and the message buffer status changed 1...4 (MBSC1...4) registers are also set. Note: MBU[6-0] are reset when the communication controller leaves CONFIG state or enters STARTUP state.
23	Reserved		Reads are zeros and writes have no effect
22–16	MBT(6–0)	0x0–0x7F	Message buffer transmitted. Number of the last successfully transmitted message buffer. If the message buffer is configured for single-shot mode, the respective TXR flag in the Transmission request register 1...4 (TXRQ1..4) was reset. Note: MBT[6-0] are reset when the communication controller leaves CONFIG state or enters STARTUP state.
15	Reserved		Reads are zeros and writes have no effect
14–8	FMB(6–0)	0x0–0x7F	Faulty message buffer. A parity error occurred when reading from a message buffer or when transferring data from Input Buffer or Transient Buffer 1,2 to the message buffer referenced by FMB(6–0). This value is only valid when one of the flags PIBF, PMR, PTBF1, PTBF2, and flag FMBD is set. Is not updated while flag FMBD is set.

Table 17-139. Message Handler Status (MHDS) Field Descriptions (Continued)

Bit	Name	Value	Description
7	CRAM		Clear all internal RAMs. Signals that execution of the controller host interface command CLEAR_RAMS is ongoing (all bits of all internal RAM blocks are written to 0). The bit is set by hardware reset or by the controller host interface command CLEAR_RAMS.
		0	No execution of the controller host interface command CLEAR_RAMS
		1	Execution of the controller host interface command CLEAR_RAMS ongoing
6	MFMB		Multiple faulty message buffers detected.
		0	No additional faulty message buffer
		1	Another faulty message buffer was detected while flag FMBD is set
5	FMBD		Faulty message buffer detected.
		0	No faulty message buffer
		1	Message buffer referenced by FMB(6–0) holds faulty data due to a parity error
4	PTBF2		Parity error transient buffer RAM B.
		0	No parity error
		1	Parity error occurred when reading transient buffer RAM B
3	PTBF1		Parity error transient buffer RAM A.
		0	No parity error
		1	Parity error occurred when reading transient buffer RAM A
2	PMR		Parity error message RAM.
		0	No parity error
		1	Parity error occurred when reading message RAM
1	POBF		Parity error output buffer RAM 1, 2.
		0	No parity error
		1	Parity error occurred when message handler read output buffer RAM 1,2
0	PIBF		Parity error input buffer RAM 1, 2.
		0	No parity error
		1	Parity error occurred when message handler read input buffer RAM 1,2

Note:
When one of the flags PIBF, POBF, PMR, PTBF1, PTBF2 changes from ’0’ to ’1’ the PERR flag in the Error Interrupt Register (EIR) is set to ’1’.

17.27.2 Last Dynamic Transmit Slot (LDTS)

The register is reset when the communication controller leaves CONFIG state or enters STARTUP state Figure 17-155 and Table 17-139 illustrate this register.

Figure 17-156. Last Dynamic Transmit Slot (LDTS) [offset_CC = 0x314]

31 27	26 16
Reserved	LDTB(10–0)
R-0	R-0

15 11	10 0
Reserved	LDTA(10-0)
R-0	R-0

R = Read; W = write; -*n* = Value after reset

Table 17-140. Last Dynamic Transmit Slot (LDTS) Field Descriptions

Bit	Name	Value	Description
31-27	Reserved		Reads are zeros and writes have no effect
26–16	LDTB(10–0)		Last Dynamic Transmission Channel B. Value of Slot Counter B at the time of the last frame transmission on channel A in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment.
15-11	Reserved		Reads are zeros and writes have no effect
10–0	LDTA(10–0)		Last Dynamic Transmission Channel A. Value of Slot Counter A at the time of the last frame transmission on channel A in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment.

17.27.3 FIFO Status Register (FSR)

The register is reset when the communication controller leaves CONFIG state or enters STARTUP state. Figure 17-155 and Table 17-139 illustrate this register.

Figure 17-157. FIFO Status Register (FSR) [offset_CC = 0x318]

31–16
Reserved
R-0

15–8	7–3	2	1	0
RFFL(7-0)	Reserved	RFO	RFCL	RFNE
R-0	R-0	R-0	R-0	R-0

R = Read; W = write; -*n* = Value after reset

Table 17-141. FIFO Status Register (FSR) Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads are zeros and writes have no effect
15–8	RFFL(7–0)	0x0–0x7F	Receive FIFO Fill Level. Number of FIFO buffers filled with new data not yet read by the Host.
7-3	Reserved		Reads are zeros and writes have no effect
2	RFO		Receive FIFO Overrun. The flag is set by the communication controller when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. In addition, interrupt flag RFO in the Error Interrupt Register (EIR) is set.The flag is cleared by the next FIFO read access issued by the Host.
		0	No receive FIFO overrun detected
		1	A receive FIFO overrun has been detected
1	RFCL		Receive FIFO Critical Level. This flag is set when the receive FIFO fill level RFFL[7:0] is equal or greater than the critical level as configured by CL[7:0] in the FIFO Critical Level register (FCL). The flag is cleared by the communication controller as soon as RFFL[7:0] drops below FCL.CL[7:0]. When RFCL changes from ’0’ to ’1’ the RFCL flag in the Status Interrupt register (SIR) is set to ’1’, and if enabled, an interrupt is generated.
		0	Receive FIFO below critical level
		1	Receive FIFO critical level reached

Table 17-141. FIFO Status Register (FSR) Field Descriptions (Continued)

Bit	Name	Value	Description
0	RFNE		Receive FIFO Not Empty. This flag is set by the communication controller when a received valid frame (data or null frame depending on rejection mask) was stored in the FIFO. In addition, interrupt flag RFNE in the Status Interrupt register (SR) is set. The bit is reset after the Host has read all message from the FIFO.
		0	Receive FIFO is empty
		1	Receive FIFO is not empty

17.27.4 Message Handler Constraints Flags (MHDF)

Some constraints exist for the Message Handler regarding VBUSclk frequency, Message RAM configuration, and FlexRay bus traffic. In order to simplify software development, constraints violations are reported by setting flags in the MHDF

A flag is cleared by writing a ’1’ to the corresponding bit position. Writing a ’0’ has no effect on the flag. A hardware reset will also clear the register. The register is reset when the communication controller leaves CONFIG state or enters STARTUP state.

Figure 17-155 and Table 17-139 illustrate this register.

Figure 17-158. Message Handler Constraints Flags (MHDF) [offset_CC = 0x31C]

31–16
Reserved
R-0

15–9	8	7	6	5	4	3	2	1	0
Reserved	WAHP	TNSA	TNSB	TBFB	TBFA	FNFB	FNFA	SNUB	SNUA
R-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	R/W-0

R = Read; W = write; *-n* = Value after reset

Table 17-142. Message Handler Constraint Flags (MHDF) Field Descriptions

Bit	Name	Value	Description
31-9	Reserved		Reads are zeros and writes have no effect
8	WAHP		Write attempt to header partition. This flag is set by the communication controller when the message handler tries to write message data into the header partition of the Message RAM due to faulty configuration of a message buffer. The write attempt is not executed, to protect the header partition from unintended write accesses.
		0	No write attempt to header partition
		1	Write attempt to header partition
7	TNSA		Transmission Not Started Channel A This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel A at the action point of the configured slot.
		0	No transmission not started on channel A
		1	Transmission not started on channel A

Table 17-142. Message Handler Constraint Flags (MHDF) Field Descriptions (Continued)

Bit	Name	Value	Description
6	TNSB		Transmission Not Started Channel B This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel B at the action point of the configured slot.
		0	No transmission not started on channel B
		1	Transmission not started on channel B
5	TBFB		Transient buffer access failure B. This flag is set by the communication controller when a read or write access to TBF B requested by PRT B could not complete within the available time.
		0	No TBF B access failure
		1	TBF B access failure
4	TBFA		Transient buffer access failure A. This flag is set by the communication controller when a read or write access to TBF A requested by PRT A could not complete within the available time.
		0	No TBF A access failure
		1	TBF A access failure
3	FNFB		Find sequence not finished channel B. This flag is set by the communication controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer) with respect to channel B.
		0	No find sequence not finished for channel B
		1	Find sequence not finished for channel B
2	FNFA		Find sequence not finished channel A. This flag is set by the communication controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer) with respect to channel A.
		0	No find sequence not finished for channel A
		1	Find sequence not finished for channel A
1	SNUB		Status not updated channel B. This flag is set by the communication controller when the Message Handler, due to overload condition, was not able to update a message buffer’s status MBS with respect to channel B.
		0	No overload condition occurred when updating MBS for channel B
		1	MBS for channel B not updated
0	SNUA		Status not updated channel A. This flag is set by the communication controller when the Message Handler, due to overload condition, was not able to update a message buffer’s status MBS with respect to channel A.
		0	No overload condition occurred when updating MBS for channel A
		1	MBS for channel A not updated

Note:

When one of the flags SNUA, SNUB, FNFA, FNFB, TBFA, TBFB, WAHP changes from'0' to '1', interrupt flag MHF in the Error Interrupt register (EIR) is set to '1'.

17.27.5 Transmission Request 1/2/3/4 (TXRQ1/2/3/4)

These four registers reflect the state of the TXR flags of all configured message buffers. The flags are evaluated for transmit buffers only. If the number of configured message buffers is less than 128, the remaining TXR flags have no meaning.

Figure 17-159 through Figure 17-162 and Table 17-143 illustrate these registers.

Figure 17-159. Transmission Request Register 4 (TXRQ4) [offset_CC = 0x32C]

31	16
TXR[127:112]	
R-0	

15	0
TXR[111:96]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-160. Transmission Request Register 3 (TXRQ3) [offset_CC = 0x328]

31	16
TXR[95:80]	
R-0	

15	0
TXR[79:64]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-161. Transmission Request Register 2 (TXRQ2) [offset_CC = 0x324]

31	16
TXR[63:48]	
R-0	

15	0
TXR[47:32]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-162. Transmission Request Register 1 (TXRQ1) [offset_CC = 0x320]

31	16
TXR[31:16]	
R-0	R-0

15	0
TXR[15:0]	
R-0	R-0

R = Read; -*n* = Value after reset

Table 17-143. Transmission Request Register (TXRQ1/2/3/4) Field Description

Bit	Name	Value	Description
127–0	TXR(127-0)]		Transmission request.
		0	The respective message buffer is not ready for transmission.
		1	If the flag is set, the respective message buffer is ready for transmission. Respectively, transmission of this message buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

17.27.6 New Data 1/2/3/4 (NDAT1/2/3/4)

The four registers reflect the state of the ND flags of all configured message buffers. **ND** flags corresponding to transmit buffers have no meaning. If the number of configured message buffers is less than 128, the remaining ND flags have no meaning. The registers are reset when the communication controller leaves CONFIG state or enters STARTUP state.

Figure 17-163 through Figure 17-166 and Table 17-144 illustrate these registers.

Figure 17-163. New Data Register 4 (NDAT4) [offset_CC = 0x33C]

31	16
ND[127:112]	
R-0	
15	0
ND[111:96]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-164. New Data Register 3 (NDAT3) [offset_CC = 0x338]

31	16
ND[95:80]	
R-0	
15	0
ND[79:64]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-165. New Data Register 2 (NDAT2) [offset_CC = 0x334]

31	16
ND[63:48]	
R-0	
15	0
ND[47:32]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-166. New Data Register 1 (NDAT1) [offset_CC = 0x330]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND[31:16]															
R-0															

15	0
ND[15:0]	
R-0	

R = Read; -*n* = Value after reset

Table 17-144. New Data Register (NDAT1/2/3/4) Field Descriptions

Bit	Name	Value	Description
127–0	ND(127:0)		New data.
		0	The flags are reset when the header section of the corresponding message buffer is reconfigured or when the data section has been transferred to the output buffer.
		1	The flags are set when a valid received data frame matches the message buffer’s filter configuration, independent of the payload length received or the payload length configured for that message buffer. The flags are not set after reception of null frames except for message buffers belonging to the receive FIFO.

17.27.7 Message Buffer Status Changed 1/2/3/4 (MBSC1/2/3/4)

The four registers reflect the state of the MBC flags of all configured message buffers. If the number of configured message buffers is less than 128, the remaining MBC flags have no meaning.

Figure 17-167 through Figure 17-170 and Table 17-145 illustrate these registers.

Figure 17-167. Message Buffer Status Changed Register 4 (MBSC4) [offset_CC = 0x34C]

31	16
MBS[127:112]	
R-0	

15	0
MBS[111:96]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-168. Message Buffer Status Changed Register 3 (MBSC3) [offset_CC = 0x348]

31	16
MBS[95:80]	
R-0	

15	0
MBS[79:64]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-169. Message Buffer Status Changed Register 2 (MBSC2) [offset_CC = 0x344]

31	16
MBS[63:48]	
R-0	

15	0
MBS[47:32]	
R-0	

R = Read; -*n* = Value after reset

Figure 17-170. Message Buffer Status Changed Register 1 (MBSC1) [offset_CC = 0x340]

31	16
MBS[31:16]	
R-0	

15	0
MBS[15:0]	
R-0	

R = Read; -*n* = Value after reset

Table 17-145. Message Buffer Status Changed Register (MBSC1/2/3/4) Field Descriptions

Bit	Name	Value	Description
127–0	MBS(127-0)		Message buffer status changed.
		0	A flag is reset when the header section of the corresponding message buffer is reconfigured or when it has been transferred to the Output Buffer.
		1	The flag is set whenever the Message Handler changes one of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the header section (see Chapter 17.30.5, Message Buffer Status (MBS)) of the respective message buffer.

17.28 Identification Registers

17.28.1 Core Release Register (CREL)

Figure 17-171 and Table 17-146 illustrate this register.

Figure 17-171. Core Release Register (CREL) [offset_CC = 0x3F0]

31 – 28	27 – 20	19 – 16
REL(3-0)	STEP(7-0)	YEAR(3-0)
R-release info	R-release info	R-release info

15 – 8	7 – 0
MON(7-0)	DAY(7-0)
R-release info	R-release info

R = Read; W = write; -*n* = Value after reset

Table 17-146. Core Release Register (CREL) Field Descriptions

Bit	Name	Value	Description
31-28	REL(3-0)		Core Release. One digit, BCD-coded.
27–20	STEP(7–0)		Step of Core Release. Two digits, BCD-coded.
19-16	YEAR(3-0)		Design Time Stamp, Year. One digit, BCD-coded.
15-8	MON(7-0)		Design Time Stamp, Month. Two digits, BCD-coded.
7-0	DAY(7-0)		Design Time Stamp, Day. Two digits, BCD-coded.

Below Table 17-147 shows the release coding in register CREL.

Table 17-147. Release Coding

			Core Release Register	
Release	Step	Sub-Step	Contents	Name
1	0	0	1006 0519	Revision 1.0.0
1	0	1	1016 1211	Revision 1.0.1
1	0	2	10271031	Revision 1.0.2

17.28.2 Endian Register (ENDN)

Figure 17-172 and Table 17-147 illustrate this register.

Figure 17-172. Endian Register (ENDN) [offset_CC = 0x3F4]

31	16
ETV(31-16)	
R-8765h	

15	8	7	0
ETV(15-0)			
R-4321h			

R = Read; W = write; -*n* = Value after reset

Table 17-148. Endian Register (ENDN) Field Descriptions

Bit	Name	Value	Description
31-0	ETV(31-0)		Endianess Test Value. The Endianess test value is 0x87654321.

17.29 Input Buffer

Double buffer structure consisting of input buffer host and input buffer shadow. While the host can write to input buffer host, the transfer to the message RAM is done from input buffer shadow. The input buffer holds the header and data sections to be transferred to the selected message buffer in the message RAM. It is used to configure the message buffers in the message RAM and to update the data sections of transmit buffers.

When updating the header section of a message buffer in the Message RAM from the Input Buffer, the Message Buffer Status as described in Chapter 17, Message Buffer Status (MBS) is automatically reset to zero.

The header sections of message buffers belonging to the receive FIFO can only be (re)configured when the communication controller is in DEFAULT_CONFIG or CONFIG state. For those message buffers only the payload length configured and the data pointer need to be configured by bits PLC[6.0] of the Write Header Section 2 (WRHS2) and by bits DP[10:0] of Write Header Section 3 (WRHS3). All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask.

17.29.1 Write Data Section [1. . .64] (WRDSn)

Holds the data words to be transferred to the data section of the addressed message buffer. The data words (DW_n) are written to the message RAM in transmission order from DW_1 (byte0, byte1) to DW_{PL} (DW_{PL} = number of data words as defined by the payload length configured in PLC(6-0) of the Write Header Section 2 (WRHS2).

Figure 17-173 and Table 17-149 illustrate these registers.

Figure 17-173. Write Data Section [1. . .64] Register (WRDSn) [offset_CC = 0x400–0x4FC]

31 ... 16
MD(31–16)
R/W-0

15 ... 0
MD(15–0)
R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-149. Write Data Section Register [1. . .64] (WRDSn) Field Descriptions

Bit	Name	Value	Description
31–0	MD(31–0)	0x0-0xFFFFFFFF	Message data. Note: DW127 is located on WRDS64.MD[15:0]. In this case WRDS64.MD[31:16] is unused (no valid data). The input buffer RAMs are initialized to 0 when leaving hardware reset or by the controller host interface command CLEAR_RAMS. MD(31–24) = DW_{2n}, $byte_{4n-1}$ MD(23–16) = DW_{2n}, $byte_{4n-2}$ MD(15–8) = DW_{2n-1}, $byte_{4n-3}$ MD(7–0) = DW_{2n-1}, $byte_{4n-4}$

17.29.2 Write Header Section 1 (WRHS1)

Figure 17-174 and Table 17-150 illustrate this register.

Figure 17-174. Write Header Section Register 1 (WRHS1) [offset_CC = 0x500]

31–30	29	28	27	26	25	24	23	22–16
Reserved	MBI	TXM	PPIT	CFG	CHB	CHA	Reserved	CYC(6–0)
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

15–11	10–0
Reserved	FID(10–0)
R-0	R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-150. Write Header Section Register 1 (WRHS1) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads are zeros and writes have no effect
29	MBI		Message buffer interrupt. This bit enables the receive/transmit interrupt for the corresponding message buffer. After a dedicated receive buffer has been updated by the message handler, flag RXI and/or MBSI in the status interrupt register are set. After successful transmission the TXI flag in the status interrupt register is set.
		0	The corresponding message buffer interrupt is enabled
		1	The corresponding message buffer interrupt is disabled
28	TXM		Transmission mode. This bit is used to select the transmission mode.
		0	Continuous mode
		1	Single-shot mode
27	PPIT		Payload preamble indicator transmit. This bit is used to control the state of the Payload Preamble Indicator in transmit frames. If the bit is set in a static message buffer, the respective message buffer holds network management information. If the bit is set in a dynamic message buffer, the first two bytes of the payload segment may be used for message ID filtering by the receiver. Message ID filtering of received FlexRay frames is not supported by the FlexRay module, but can be done by the host CPU.
		0	Payload Preamble Indicator not set
		1	Payload Preamble Indicator set

Table 17-150. Write Header Section Register 1 (WRHS1) Field Descriptions (Continued)

Bit	Name	Value	Description
26	CFG		Message buffer configuration bit. This bit is used to configure the corresponding buffer as Transmit buffer or as receive buffer. For message buffers belonging to the receive FIFO the bit is not evaluated.
		0	The corresponding buffer is configured as receive buffer
		1	The corresponding buffer is configured as Transmit buffer
25–24	CHB, CHA	0–4h	Channel filter control. The 2-bit channel filtering field associated with each buffer serves as a filter for receive buffers and as a control field for transmit buffers. See Table 17-151 for bit descriptions. **Note: If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no frames are transmitted resp. received frames are ignored (same function as CHA = CHB = 0)**
23	Reserved		Reads are zeros and writes have no effect
22–16	CYC(6–0)	0–7Fh	Cycle code. The 7-bit cycle code determines the cycle set used for cycle counter filtering.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	FID(10–0)	0–7FFh	Frame ID. Frame ID of the selected message buffer. The frame ID defines the slot number for transmission / reception of the respective message. **Note: Message buffers with frame ID = 0 are considered not valid.**

Table 17-151. Channel Filter Control Bit Descriptions

CHA	CHB	**Transmit Buffer** transmit frame on	**Receive Buffer** store frame received from
1	1	both channels (static segment only)	channel A or B (store first semantically valid frame, static segment only)
1	0	channel A	channel A
0	1	channel B	channel B
0	0	no transmission	ignore frame

17.29.3 Write Header Section 2 (WRHS2)

Figure 17-175 and Table 17-152 illustrate this register.

Figure 17-175. Write Header Section Register 2 (WRHS2) [offset_CC = 0x504]

31–23	22–16
Reserved	PLC(6–0)
R-0	R/W-0

15–11	10–0
Reserved	CRC(10–0)
R-0	R/W-0

R = Read; W = write; *-n* = Value after reset

Table 17-152. Write Header Section Register 2 (WRHS2) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads are zeros and writes have no effect
22–16	PLC(6–0)	0x0–0x3F	Payload length configured. Length of data section (number of 2-byte words) as configured by the host. During static segment the static frame data length as configured by SFDL(6–0) in the MHD configuration register defines the payload length for all static frames. If the payload length configured by PLC(6–0) is shorter than this value padding bytes are inserted to ensure that frames have proper physical length. The padding pattern is logical *zero*.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	CRC(10–0)	0x0–0x7FF	Header CRC. Receive Buffer: configuration not required Transmit buffer: Header CRC calculated and configured by the host. For calculation of the header CRC the payload length of the frame send on the bus has to be considered. In static segment the payload length of all frames is configured by MHDC.SFDL[6:0].

17.29.4 Write Header Section 3 (WRHS3)

Figure 17-176 and Table 17-153 illustrate this register.

Figure 17-176. Write Header Section Register 3 (WRHS3) [offset_CC = 0x508]

31   16
Reserved
R-0

15 11	10 0
Reserved	DP(10–0)
R-0	R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-153. Write Header Section Register 3 (WRHS3) Field Descriptions

Bit	Name	Value	Description
31–11	Reserved		Reads are zeros and writes have no effect
10–0	DP(10–0)	0x1– 0x7FF	Data pointer. Pointer to the first 32-bit word of the data section of the addressed message buffer in the message RAM.

17.29.5 Input Buffer Command Mask (IBCM)

Configures how the message buffer in the message RAM selected by the input buffer command request register is updated. When IBF host and IBF shadow are swapped, also mask bits LHSH, LDSH, and STXRH are swapped with bits LHSS, LDSS, and STXRS to keep them attached to the respective input buffer transfer.

Figure 17-177 and Table 17-154 illustrate this register.

Figure 17-177. Input Buffer Command Mask Register (IBCM) [offset_CC = 0x510]

31–19	18	17	16
Reserved	STXRS	LDSS	LHSS
R-0	R-0	R-0	R-0

15–3	2	1	0
Reserved	STXRH	LDSH	LHSH
R-0	R/W-0	R/W-0	R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-154. Input Buffer Command Mask Register (IBCM) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads are zeros and writes have no effect
18	STXRS		Set transmission request shadow.
		0	Reset TXR flag
		1	Set TXR flag, transmit buffer released for transmission (operation ongoing or finished)
17	LDSS		Load data section shadow.
		0	Data section is not updated
		1	Data section selected for transfer from input buffer to the message RAM (transfer ongoing or finished)
16	LHSS		Load header section shadow.
		0	Header section is not updated
		1	Header section selected for transfer from input buffer to the message RAM (transfer ongoing or finished)
15–3	Reserved		Reads are zeros and writes have no effect
2	STXRH		Set transmission request host. If this bit is set to 1, the transmission request flag TXR for the selected message buffer is set in the transmission request registers to release the message buffer for transmission. In single-shot mode the flag is cleared by the communication controller after transmission has completed. The flags is evaluated for transmit buffers only.
		0	Reset transmission request flag
		1	Set transmission request flag; transmit buffer released for transmission

Table 17-154. Input Buffer Command Mask Register (IBCM) Field Descriptions (Continued)

Bit	Name	Value	Description
1	LDSH		Load data section host.
		0	Data section is not updated
		1	Data section selected for transfer from input buffer to the message RAM
0	LHSH		Load header section host.
		0	Header section is not updated
		1	Header section selected for transfer from input buffer to the message RAM

17.29.6 Input Buffer Command Request (IBCR)

When the host writes the number of a target message buffer in the message RAM to IBRH(6–0) in the input buffer command request register, IBF host and IBF shadow are swapped. In addition the message buffer numbers stored under IBRH(6–0) and IBRS(6–0) are also swapped.

With this write operation the IBSYS bit in the input buffer command request register is set to 1. The message handler then starts to transfer the contents of IBF shadow to the message buffer in the message RAM selected by IBRS(6–0).

While the message handler transfers the data from IBF shadow to the target message buffer in the message RAM, the host may configure the next message in the IBF host. After the transfer between IBF shadow and the message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the message RAM may be started by the host by writing the respective target message buffer number to IBRH(6–0).

If a write access to IBRH(6–0) occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF shadow to the message RAM, IBF host and IBF shadow are swapped, IBSYH is reset to 0. IBSYS remains set to 1, and the next transfer to the message RAM is started. In addition the message buffer numbers stored under IBRH(6–0) and IBRS(6–0) are also swapped.

Any write access to an Input Buffer Register while both IBSYS and IBSYH are set will cause the error flag IIBA in the Error Interrupt Register (EIR) to be set. In this case the Input Buffer will not be changed.

Figure 17-178 and Table 17-155 illustrate this register.

Figure 17-178. Input Buffer Command Request Register (IBCR) [offset_CC = 0x514]

31	30 — 23	22 — 16
IBSYS	Reserved	IBRS(6–0)
R-0	R-0	R-0

15	14 — 7	6 — 0
IBSYH	Reserved	IBRH(6–0)
R-0	R-0	R/W-0

R = Read; W = write; *-n* = Value after reset

Table 17-155. Input Buffer Command Request Register (IBCR) Field Descriptions

Bit	Name	Value	Description
31	IBSYS		Input buffer busy shadow. Set to 1 after writing IBRH(6–0). When the transfer between IBF shadow and the message RAM has completed, IBSYS is set back to 0.
		0	Transfer between IBF shadow and message RAM completed
		1	Transfer between IBF shadow and message RAM in progress
30–23	Reserved		Reads are zeros and writes have no effect
22–16	IBRS(6–0)	0x0–0x7F	Input buffer request shadow. Number of the target message buffer actually updated / lately updated.

Table 17-155. Input Buffer Command Request Register (IBCR) Field Descriptions (Continued)

Bit	Name	Value	Description
15	IBSYH		Input buffer busy host. Set to 1 by writing IBRH(6–0) while IBSYS is still 1. After the ongoing transfer between IBF shadow and the message RAM has completed, the IBSYH is set back to 0.
		0	No request pending
		1	Request while transfer between IBF shadow and message RAM in progress
14–7	Reserved		Reads are zeros and writes have no effect
6–0	IBRH(6–0)	0x0–0x7F	Input buffer request host. Selects the target message buffer in the Message RAM for data transfer from Input Buffer.

17.30 Output Buffer

Double buffer structure consisting of output buffer host and output buffer shadow. While the host can read from output buffer host, the transfer from the message RAM is done to output buffer shadow. The output buffer holds the header and data sections of requested message buffers transferred from the message RAM. Used to read out message buffers from the message RAM.

17.30.1 Read Data Section [1…64] (RDDSn)

Holds the data words read from the data section of the addressed message buffer. The data words (DW_n) are read from the message RAM in reception order from DW_1 (byte0, byte1) to DW_{PL} (DW_{PL} = number of data words as defined by the payload length configured in bits PLC(6-0) of the Read Header Section 2 (RDHS2)).

Figure 17-179 and Table 17-156 illustrate these registers.

Figure 17-179. Read Data Section Register (RDDSn) [offset_CC = 0x600–0x6FC]

31		16
	MD(31–16)	
	R/W-0	

15		0
	MD(15–0)	
	R/W-0	

R = Read; W = Write; -*n* = Value after reset

Table 17-156. Read Data Section Register (RDDSn) Field Descriptions

Bit	Name	Value	Description
31–0	MD(31–0)	0x0– 0xFFFF FFFF	Message data. Note: DW127 is located on RDDS64.MD[15:0]. In this case RDDS64.MD[31:16] is unused (no valid data). The output buffer RAMs are initialized to 0 when leaving hardware reset or by the controller host interface command CLEAR_RAMS. MD(31–24) = DW_{2n}, $byte_{4n-1}$ MD(23–16) = DW_{2n}, $byte_{4n-2}$ MD(15–8) = DW_{2n-1}, $byte_{4n-3}$ MD(7–0) = DW_{2n-1}, $byte_{4n-4}$

17.30.2 Read Header Section 1 (RDHS1)

Figure 17-180 and Table 17-157 illustrate this register.

Figure 17-180. Read Header Section Register 1 (RDHS1) [offset_CC = 0x700]

31	30	29	28	27	26	25	24	23	22	16
Reserved		MBI	TXM	PPIT	CFG	CHB	CHA	Reserved	CYC(6–0)	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	

15	11	10	0
Reserved		FID(10–0)	
R-0		R-0	

R = Read; W = Write; -*n* = Value after reset

Table 17-157. Read Header Section Register 1 (RDHS1) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads are zeros and writes have no effect
29	MBI		Message buffer interrupt.
		0	The corresponding message buffer interrupt is enabled
		1	The corresponding message buffer interrupt is disabled
28	TXM		Transmission mode. This bit is used to select the transmission mode.
		0	Continuous mode
		1	Single-shot mode
27	PPIT		Payload preamble indicator transmit.
		0	Payload Preamble Indicator not set
		1	Payload Preamble Indicator set
26	CFG		Message buffer configuration bit.
		0	The corresponding buffer is configured as receive buffer
		1	The corresponding buffer is configured as transmit buffer
25–24	CHB, CHA		Channel filter control. See Table 17-151 for bit descriptions.
23	Reserved		Reads are zeros and writes have no effect
22–16	CYC(6–0)	0x0–0x7F	Cycle code. The 7-bit cycle code determines the cycle set used for cycle counter filtering.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	FID(10–0)	0x0–0x7FF	Frame ID. Frame ID of the selected message buffer. **Note: Message buffers with frame ID = 1 are considered not valid.**

Note:
In case the message buffer read from the message RAM belongs to the receive FIFO, FID(10–0) and CHA, CHB were updated from the received frame while CYC(6–0), CFG, PPIT, TXM, and MBI are reset to zero.

Note:
For bit description see also section 17.29.2

17.30.3 Read Header Section 2 (RDHS2)

Figure 17-175 and Table 17-152 illustrate this register.

Figure 17-181. Read Header Section Register 2 (RDHS2) [offset_CC = 0x704]

31	30 24	23	22 16
Reserved	PLR(6–0)	Reserved	PLC(6–0)
R-0	R-0	R-0	R-0

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10 0</th></tr>
<tr><td colspan="5">Reserved</td><td>CRC(10–0)</td></tr>
<tr><td>R-0</td><td>R-0</td><td>R-0</td><td>R-0</td><td>R-0</td><td>R-0</td></tr>
</table>

R = Read; W = write; -*n* = Value after reset

Table 17-158. Read Header Section Register 2 (RDHS2) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads are zeros and writes have no effect
30–24	PLR(6–0)	0x0–0x7F	Payload length received. Payload length value updated from received frame (exception: if message buffer belongs to the receive FIFO PLR[6:0] is also updated from received null frames). When a message is stored into a message buffer the following behavior with respect to payload length received and payload length configured is implemented: **PLR[6:0] > PLC[6:0]:** The payload data stored in the message buffer is truncated to the payload length configured if PLC[6:0] even or else truncated to PLC[6:0] + 1. **PLR[6:0]<= PLC[6:0]:** The received payload data is stored into the message buffers data section. The remaining data bytes of the data section as configured by PLC[6:0] are filled with undefined data **PLR[6:0] = zero:** The message buffers data section is filled with undefined data **PLC[6:0] = zero:** Message buffer has no data section configured. No data is stored into the message buffers data section.
23	Reserved		Reads are zeros and writes have no effect
22–16	PLC(6–0)	0x0–0x7F	Payload length configured. Length of data section (number of 2-byte words) as configured by the host.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	CRC(10–0)	0x0–0x7FF	Header CRC. Receive buffer: Header CRC updated from receive frame Transmit buffer: Header CRC calculated and configured by the host

Note:

The Message RAM is organized in 4-byte words. When received data is stored into a message buffer’s data section, the number of 2-byte data words written into the message buffer is PLC[6:0] rounded to the next even value. PLC[6:0] should be configured identical for all message buffers belonging to the receive FIFO. Header 2 is updated from data frames only.

17.30.4 Read Header Section 3 (RDHS3)

Figure 17-182 and Table 17-159 illustrate this register.

Figure 17-182. Read Header Section Register 3 (RDHS3) [offset_CC = 0x708]

31	30	29	28	27	26	25	24	23	22	21	16
Reserved		RES	PPI	NFI	SYN	SFI	RCI	Reserved		RCC(5–0)	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		R-0	

15	11	10	0
Reserved		DP(10–0)	
R-0		R/W-0	

R = Read; W = write; *-n* = Value after reset

Table 17-159. Read Header Section Register 3 (RDHS3) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads are zeros and writes have no effect
29	RES	0–1	Reserved bit. Reflects the state of the received reserved bit. The reserved bit is transmitted as 0.
28	PPI		Payload preamble indicator. The payload preamble indicator defines whether a network management vector or message ID is contained within the payload segment of the received frame.
		0	The payload segment of the received frame does not contain a network management vector or a message ID.
		1	*Static segment:* Network management vector at the beginning of the payload. *Dynamic segment:* Message ID at the beginning of the payload.
27	NFI		Null frame indicator. Is set to ’1’ after storage of the first received data frame.
		0	Up to now no data frame has been stored into the respective message buffer
		1	At least one data frame has been stored into the respective message buffer
26	SYN		Sync frame indicator. A sync frame is marked by the sync frame indicator.
		0	The received frame is not a sync frame
		1	The received frame is a sync frame.
25	SFI		Startup frame indicator. A startup frame is marked by the startup frame indicator.
		0	The received frame is not a startup frame
		1	The received frame is a startup frame.

Table 17-159. Read Header Section Register 3 (RDHS3) Field Descriptions (Continued)

Bit	Name	Value	Description
24	RCI		Received on channel indicator. Indicates the channel from which the received data frame was taken to update the respective receive buffer.
		0	Frame received on channel B
		1	Frame received on channel A
23–16	RCC(5–0)		Receive cycle count. Cycle counter value updated from received frame.
15–11	Reserved		Reads are zeros and writes have no effect
10–0	DP(10–0)		Data pointer. Pointer to the first 32-bit word of the data section of the addressed message buffer in the message RAM.

Note:
Header 3 is updated from data frames only.

17.30.5 Message Buffer Status (MBS)

The message buffer status is updated by the communication controller with respect to the assigned channel(s) latest at the end of the slot following the slot assigned to the message buffer. The flags are updated only when the communication controller is in NORMAL_ACTIVE or NORMAL_PASSIVE state. If only one channel (A or B) is assigned to a message buffer, the channel-specific status flags of the other channel are written to zero. If both channels are assigned to a message buffer, the channel-specific status flags of both channels are updated. The message buffer status is updated only when the slot counter reached the configured frame ID and when the cycle counter filter matched. When the Host updates a message buffer via Input Buffer, all MBS flags are reset to zero independent of which IBCM bits are set or not.

Whenever the Message Handler changes one of the flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB the respective message buffer's MBC flag in registers MBSC1/2/3/4 is set.

Figure 17-183 and Table 17-160 illustrate this register.

Figure 17-183. Message Buffer Status Register (MBS) [offset_CC = 0x70C]

31	30	29	28	27	26	25	24	23	22	21	16
Reserved		RESS	PPIS	MFIS	SYNS	SFIS	RCIS	Reserved		CCS(5-0)	
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0		R-0	

15	12	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTB	FTA	Reserved	MLST	ESB	ESA	TCIB	TCIA	SVOB	SVOA	CEOB	CEOA	SEOB	SEOA	VFRB	VFRA
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R = Read; W = write; -*n* = Value after reset

Table 17-160. Message Buffer Status Register (MBS) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Reads are zeros and writes have no effect
29	RESS		Reserved bit status. Reflects the state of the received reserved bit. The reserved bit is transmitted as ’0’.
28	PPIS		Payload preamble indicator status. The payload preamble indicator defines whether a network management vector or message ID is contained within the payload segment of the received frame.
		0	The payload segment of the received frame does not contain a network management vector or a message ID
		1	Static segment: Network management vector at the beginning of the payload Dynamic segment: Message ID at the beginning of the payload
27	NFIS		Null frame indicator status. If set to ’0’ the payload segment of the received frame contains no usable data.
		0	Received frame is a null frame
		1	Received frame is not a null frame

Table 17-160. Message Buffer Status Register (MBS) Field Descriptions (Continued)

Bit	Name	Value	Description
26	SYNS		Sync frame indicator status. A sync frame is marked by the sync frame indicator.
		0	No sync frame received
		1	The received frame is a sync frame
25	SFIS		Startup frame indicator status. A startup frame is marked by the startup frame indicator.
		0	No startup frame received
		1	The received frame is a startup frame
24	RCIS		Received on channel indicator status. Indicates the channel on which the frame was received.
		0	Frame received on channel B
		1	Frame received on channel A
23-22	Reserved		Reads are zeros and writes have no effect
21-16	CCS(5-0)		Cycle count status. Actual cycle count when status was updated.
15	MTB		Frame transmitted on channel B. Indicates that this node has transmitted a data frame in the configured slot on channel B.
		0	No data frame transmitted on channel B
		1	Data frame transmitted on channel B
14	MTA		Frame transmitted on channel A. Indicates that this node has transmitted a data frame in the configured slot on channel A.
		0	No data frame transmitted on channel A
		1	Data frame transmitted on channel A
13	Reserved		Reads are zeros and writes have no effect
12	MLST		Message lost.The flag is set in case the Host did not read the message before the message buffer was updated from a received data frame. Not affected by reception of null frames except for message buffers belonging to the receive FIFO. The flag is reset by a host write to the message buffer via IBF or when a new message is stored into the message buffer after the message buffers ND flag was reset by reading out the message buffer via OBF.
		0	No message lost
		1	Unprocessed message was overwritten
11	ESB		Empty slot channel B. In an empty slot there is no activity on the bus. The condition is checked in static and dynamic slots.
		0	Bus activity detected in the configured slot on channel B
		1	No bus activity detected in the configured slot on channel B

Table 17-160. Message Buffer Status Register (MBS) Field Descriptions (Continued)

Bit	Name	Value	Description
10	ESA		Empty slot channel A. In an empty slot there is no activity on the bus. The condition is checked in static and dynamic slots.
		0	Bus activity detected in the configured slot on channel A
		1	No bus activity detected in the configured slot on channel A
9	TCIB		Transmission conflict indication channel B. A transmission conflict indication is set if a transmission conflict has occurred on channel B.
		0	No transmission conflict occurred on channel B
		1	Transmission conflict occurred on channel B
8	TCIA		Transmission conflict indication channel A. A transmission conflict indication is set if a transmission conflict has occurred on channel A.
		0	No transmission conflict occurred on channel A
		1	Transmission conflict occurred on channel A
7	SVOB	0	Slot boundary violation observed on channel B. A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel B.
		0	No slot boundary violation observed on channel B
		1	Slot boundary violation observed on channel B
6	SVOA		Slot boundary violation observed on channel A. A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel A.
		0	No slot boundary violation observed on channel A
		1	Slot boundary violation observed on channel A
5	CEOB		Content error observed on channel B. A content error was observed in the configured slot on channel B.
		0	No content error observed on channel B
		1	Content error observed on channel B
4	CEOA		Content error observed on channel A. A content error was observed in the configured slot on channel A.
		0	No content error observed on channel A
		1	Content error observed on channel A
3	SEOB		Syntax error observed on channel B. A syntax error was observed in the assigned slot on channel B.
		0	No syntax error observed on channel B
		1	Syntax error observed on channel B

Table 17-160. Message Buffer Status Register (MBS) Field Descriptions (Continued)

Bit	Name	Value	Description
2	SEOA		Syntax error observed on channel A. A syntax error was observed in the assigned slot on channel A.
		0	No syntax error observed on channel A
		1	Syntax error observed on channel A
1	VFRB		Valid frame received on channel B. A valid frame indication is set if a valid frame was received on channel B.
		0	No valid frame received on channel B
		1	Valid frame received on channel B
0	VFRA		Valid frame received on channel A. A valid frame indication is set if a valid frame was received on channel A.
		0	No valid frame received on channel A
		1	Valid frame received on channel A

Note:
The status bits RESS, PPPIS, NFIS, FYNS, SFIS and RCIS are updated from both valid data and null frames. If no valid frame was received, the previous value is maintained.

Note:
The FlexRay protocol specification requires that FTA, and FTB can only be reset by the CPU. Therefore the Cycle Count Status CCS[5:0] for these bits is only valid for the cycle where the bits are set to ’1’.

17.30.6 Output Buffer Command Mask (OBCM)

Configures how the Output Buffer is updated from the message buffer in the Message RAM selected by bits OBRS(6-0) of the output buffer command request register. Mask bits RDSS and RHSS are copied to the register internal storage when a Message RAM transfer is requested by OBCR.REQ. When OBF host and OBF shadow are swapped, also mask bits RDSH and RHSH are swapped with bits RDSS and RHSS to keep them attached to the respective output buffer transfer.

Figure 17-184 and Table 17-161 illustrate this register.

Figure 17-184. Output Buffer Command Mask Register (OBCM) [offset_CC = 0x700]

31	18	17	16
Reserved		RDSH	RHSH
R-0		R-0	R-0

15	2	1	0
Reserved		RDSS	RHSS
R-0		R/W-0	R/W-0

R = Read; W = write; -*n* = Value after reset

Table 17-161. Output Buffer Command Mask Register (MBS) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads are zeros and writes have no effect
17	RDSH		Read data section host.
		0	Data section is not read
		1	Data section selected for transfer from message RAM to output buffer
16	RHSH		Read header section host.
		0	Header section is not read
		1	Header section selected for transfer from message RAM to output buffer
15–2	Reserved		Reads are zeros and writes have no effect
1	RDSS		Read Data Section shadow.
		0	Data section is not read
		1	Data section selected for transfer from message RAM to output buffer
0	RHSS		Read header section shadow
		0	Header section is not read
		1	Header section selected for transfer from message RAM to output buffer

Note:
After the transfer of the header section from the message RAM to OBF shadow has completed, the message buffer status Changed flag MBS of the selected message buffer in the message buffer Changed 1,2,3,4 registers is cleared. After the transfer of the data section from the message RAM to OBF shadow has completed, the New Data flag ND of the selected message buffer in the New Data 1,2,3,4 registers is cleared.

17.30.7 Output Buffer Command Request (OBCR)

After setting bit REQ to 1 while OBSYS is 0, OBSYS is automatically set to 1, OBRS(6-0) is copied to the register internal storage, mask bits OBCM.RDSS and OBCM.RHSS are copied to register OBCM internal storage, and the transfer of the message buffer selected by OBRS(6-0) from the Message RAM to OBF Shadow is started. When the transfer between the Message RAM and OBF shadow has completed, this is signalled by setting OBSYS back to 0.

By setting bit VIEW to 1 while OBSYS is 0, OBF Host and OBF shadow are swapped. Additionally mask bits OBCM.RDSH and OBCM.RHSH are swapped with the register OBCM internal storage to keep them attached to the respective output buffer transfer. OBRH(6-0) signals the number of the message buffer currently accessible by the Host.

If bits REQ and VIEW are set to 1 with the same write access while OBSYS is 0, OBSYS is automatically set to 1 and OBF shadow and OBF host are swapped. Additionally mask bits OBCM.RDSH and OBCM.RHSH are swapped with the registers internal storage to keep them attached to the respective output buffer transfer. Afterwards OBRS(6-0) is copied to the register

internal storage, and the transfer of the selected message buffer from the Message RAM to OBF shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF host. When the current transfer between Message RAM and OBF shadow has completed, this is signalled by setting OBSYS back to 0.

Any write access to OBCR(15-8) while OBSYS is set will cause the error flag IOBA in the Error Interrupt Register to be set.

In this case the output buffer will not be changed.

Figure 17-185 and Table 17-162 illustrate this register.

Figure 17-185. Output Buffer Command Mask Register (OBCR) [offset_CC = 0x714]

31 23	22 16
Reserved	OBRH(6–0)
R-0	R/W-0

15	14 10	9	8	7	6 5 4 3 2 1 0
OBSYS	Reserved	REQ	VIEW	Reserve d	OBRS(6–0)
R-0	R-0	R/W-0	R/W-0	R-0	R/W-0

R = Read; W = write; *-n* = Value after reset

Table 17-162. Output Buffer Command Mask Register (OBCR) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads are zeros and writes have no effect

Table 17-162. Output Buffer Command Mask Register (OBCR) Field Descriptions (Continued)

Bit	Name	Value	Description
22–16	OBRH(6–0)	0x0–0x7F	Output buffer request host. Number of message buffer currently accessible by the Host via RDHS[1..3], MBS, and RDDS[1..64]. By writing VIEW to 1 OBF Shadow and OBF host are swapped and the transferred message buffer is accessible by the host.
15	OBSYS		Output buffer shadow busy. Set to 1 after setting bit REQ. When the transfer between the message RAM and OBF shadow has completed, OBSYS is set back to 0.
		0	No transfer in progress.
		1	Transfer between message RAM and OBF shadow in progress
9	REQ		Request message RAM Transfer. Requests transfer of message buffer addressed by OBRS(6–0) from message RAM to OBF shadow. Only writable while OBSYS = 0.
		0	No request
		1	Transfer to OBF shadow requested
8	VIEW		View shadow buffer. Toggles between OBF shadow and OBF host. Only writable while OBSYS = 0.
		0	No action
		1	Swap OBF shadow and OBF
7	Reserved		Reads are zeros and writes have no effect
6–0	OBRS(6–0)	0x0–0x7F	Output buffer request shadow. Number of source message buffer to be transferred from the message RAM to OBF shadow. If the number of the first message buffer of the receive FIFO is written to this register, the message handler transfers the message buffer addressed by the GET Index register (GIDX) to OBF shadow.

17.31 Minimum VBUS Clock Frequency

To calculate the minimum VBUS clock frequency the worst case scenario has to be considered. The worst case scenario depends on the following parameters

- maximum payload length
- minimum minislot length
- number of configured message buffers (excluding FIFO)
- used channels (single/dual channel)

Worst case scenario:

- reception of a message with a maximum payload length in slot n (n is 7,15,23,31,39,...)
- slot n+1 to n+7 are empty dynamic slots (minislot) and configured as receive buffer
- the find-sequence (usually started in slot 8,16,24,32,40,...) has to scan the maximum number of configured buffers
- the number of concurrent tasks has its maximum value of 3

The find-sequence is executed each 8 slots (slot 8,16,24,32,40,...). It has to be finished until the next find-sequence is requested.

The duration of a Transient Buffer RAM (TBF) transfer to the Message Buffer RAM (MBF) varies from 4 (header section only) to 68 (header + maximum data section) time steps plus a setup time of 6 time steps.

$\text{VBUScycles}_{\text{t2m}}$ = (number of concurrent tasks) x (6 + (number of 4-byte words))

A Slot Status (SS) transfer to the Message Buffer RAM (MBF) has a length of 1 time step plus a setup time of 4 time steps.

$\text{VBUSclk}_{\text{ss2m}}$ = (number of concurrent tasks) x 5

The find sequence has a maximum length of 128 (maximum number of buffers) time steps plus a setup time of 2 time steps.

$\text{VBUSclk}_{\text{find}}$ = (number of concurrent tasks) x (2 + (number of configured buffers))

A minislot has a length of 2 to 63 macroticks (MTicks). The minimum nominal macrotick period (MTcycle) is 1µs. A sequence of 8 minislots has a length of

$t_{\text{8minislots}}$ = 8 x MTicks x MTcycle

The worst case VCLK cycle period can be calculated as follows:

$$t_{\text{8minislots}} \geq \frac{VBUScycles_{\text{t2m}} + (7 \times VBUScycles_{\text{ss2m}}) + \text{VBUScycles}_{\text{find}}}{VBUSclk}$$

$$\frac{1}{VBUSclk} \leq \frac{t_{\text{8minislots}}}{VBUScycle_{\text{t2m}} + (7 \times VBUScycle_{\text{ss2m}}) + \text{VBUScycle}_{\text{find}}} \quad [\mu s]$$

minimum $t_{8minislots}$ = 8 * 2 * 1 µs = 16 µs
maximum VBUScycle_{t2m} = 3 * (6 + 68) = 222
maximum VBUScycle_{ss2m} = 3 * 5 = 15
maximum VBUScycle_{find} = 3 * (2 + 128) = 390

$$\frac{1}{VBUSclk} \leq = \frac{16\mu s}{222 + 7 \times 15 + 390} = 22.32ns$$

The minimum VBUSclk frequency for this worst case scenario is 44,8125 MHz.

Chapter 18
SPNU489C-February 2012

High-End Timer (NHET) Module

This document provides a general description of the High-End Timer (NHET). The NHET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 22 instructions. The NHET micromachine is connected to a port of input/output (I/O) pins.

Topic **Page**

18.1 Features

- Programmable timer for input and output timing functions
- Reduced instruction set (22 instructions) for dedicated time and angle functions
- 128 words of instruction RAM protected by parity
- User definable number 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 limiting frequency
- Low CPU overhead and interrupt load
- Efficient data transfer to or from the CPU memory with dedicated High-End-Timer Transfer Unit (HTU) or DMA
- Diagnostic capabilities with different loopback mechanisms and pin status readback functionality

18.2 Overview

The NHET is a fourth-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 NHET breakpoint feature, combined with various stop capabilities, makes the NHET software application easy to debug.

18.2.1 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, etc.); 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 NHET program itself. More complex algorithms can take advantage of the CPU access to the NHET 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 NHET RAM. CPU access to the NHET RAM also improves the debug and development of timer programs. The CPU program can stop the NHET and view the contents of the program, control, and data fields that reside in the NHET 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.

18.2.2 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set. Most of the data and the arithmetic and logical unit (ALU) are 32 bits wide. Two 25-bit registers and one 32-bit register 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 CPU 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 instructions is made up of a 32-bit program field, a 32-bit control field and a 32-bit data field. The NHET execution unit fetches the complete 96-bit instruction in one cycle and executes it. The instructions include a 9-bit field for specifying the address of the next instruction to be executed. This means that an application program sequence is not controlled by a program counter (PC), but by the actual content of each instruction. This offers greater flexibility in going back and forth in the memory during program execution.

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 or DMA 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 instruction sequence implements a timer. A PWM would need a two-instruction sequence:

Count and Compare. 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 NHET rolls over to the first instruction, the timer waits for the resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock.

The execution time of this main loop is the sum of all the cycles required for the instructions that the loop executes. The longest path through the instruction sequence must be completed within the loop resolution clock (LRP). It has to be ensured that the timer functions implemented in the algorithm complete within the 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. To overcome this limitation, some of the most commonly used instructions can access the high resolution (HR) hardware. This access allows pulse or period measurements, time compare, and PWM output waveforms at the resolution of the *high resolution clock* instead of the *loop resolution clock*.

Certain instructions (MOV32, ADM32, ...) allow the manipulation of certain fields of other instructions. These built-in move instructions actually transfer new data into the active compare field of an instruction synchronously with the resolution clock. This synchronization method makes it unnecessary to use interrupts to avoid such problems as incorrect pulse widths when the CPU updates the data asynchronously.

18.2.3 Performance

Most instructions execute in one cycle, but a few take two or three cycles. The average cycle per instruction (CPI) measured on various complex benchmarks is approximately 1.3.

The NHET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (e.g., missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed. The minimal interrupts frees the CPU bandwidth to perform other tasks.

18.2.4 NHET compared to previous generation HET

The NHET contains the following improvements compared to the previous generation HET:

- New Interrupt Enable Set and Clear registers
- Capability to generate requests to the DMA module or 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
- Changed edge detection to be independent on the previous bit
- 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 either the HET Transfer Unit (HTU) or the DMA module
- 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

18.2.5 Instructions Features

The NHET has the following instructions features:

- NHET uses a RISC-based specialized timer micromachine to carry out a set of 22 instructions
- Instructions are implemented in a Very Long Instruction Word (VLIW) format (96 bits wide)
- The NHET 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

18.2.6 Program Usage

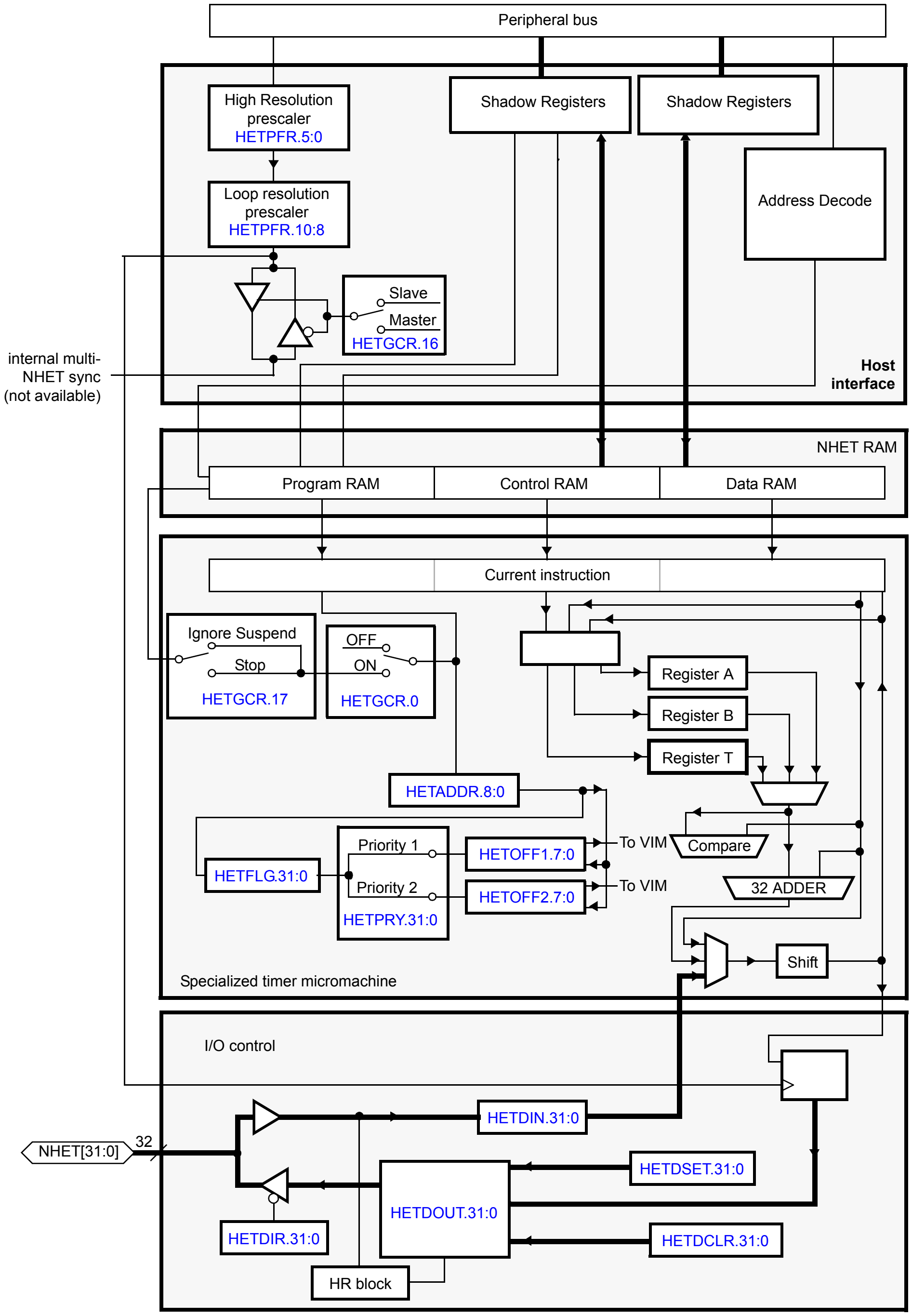
The NHET instructions/program can be assembled with the NHET assembler. The assembler generates a C-structure which can be included into the main application program. The application has to copy the content of the structure into the NHET RAM, set up necessary registers and start the NHET program execution. 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.

18.3 Block Diagram

The NHET module (see Figure 18-1) comprises four separate components:

- Host interface
- NHET RAM
- Specialized timer micromachine
- I/O control (the NHET is attached to an I/O port of up to 32 pins)

Figure 18-1. NHET Block Diagram



18.4 NHET Functional Description

The NHET contains RAM into which NHET code is loaded. The NHET 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.

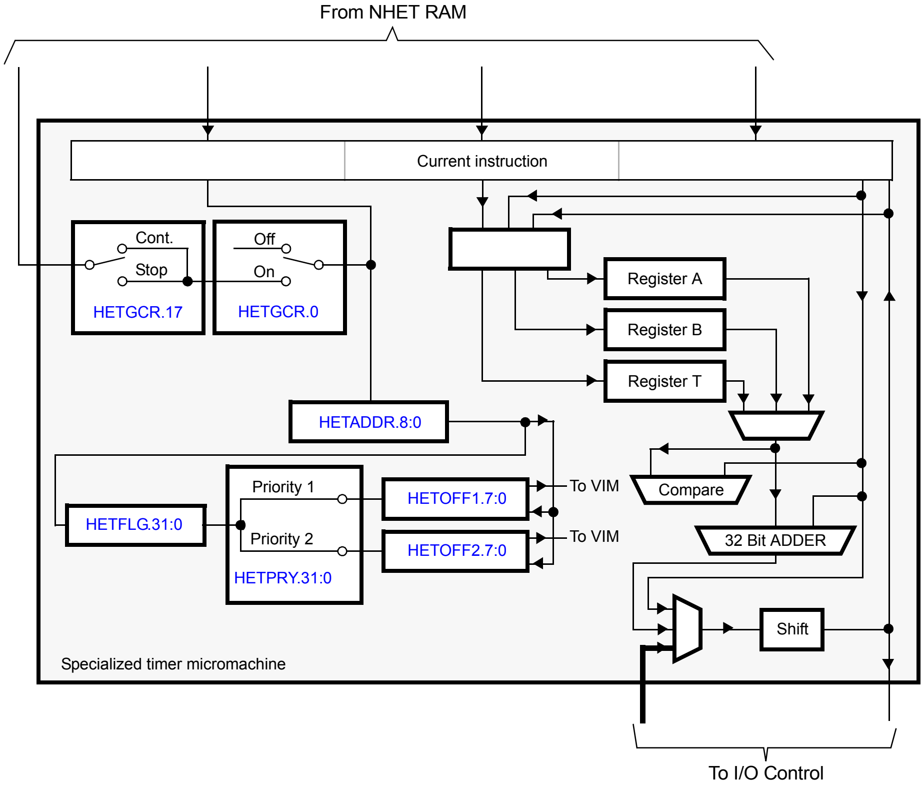
18.4.1 Specialized Timer Micromachine

The NHET has its own instruction set, detailed in Section 18.7.1. The timer micromachine reads each instruction from the NHET 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 NHET RAM sequentially. The NHET 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 18-2 shows some of the major operations that the NHET can carry out, namely compares, captures, angle functions, additions, and shifts. The NHET contains three registers (A, B, and T) used to hold compare or counter values and are used by the NHET 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 18-2. Specialized Timer Micromachine



18.4.1.1 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 NHET clock accuracy) on the HR I/O pins. For more information, see Section 18.4.5.

18.4.1.2 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 depending if special routines are executed.

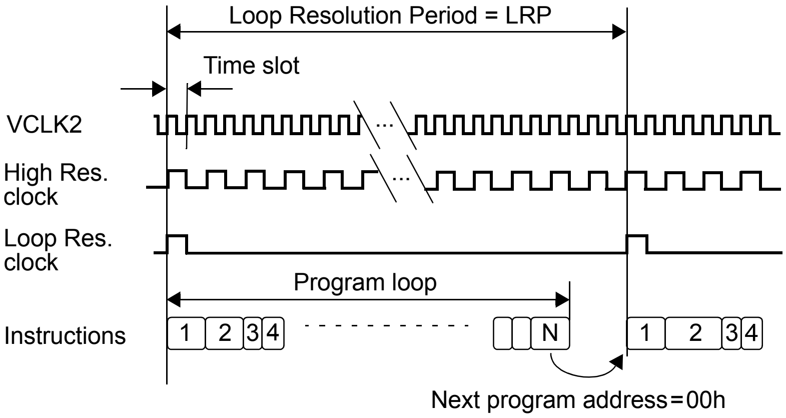
The timer program restarts on every resolution loop. The start address is fixed at RAM **address 00h**. The longest loop in a program must fit within one loop resolution period to guarantee complete accuracy.

The last instruction of a program must point to the start address (next program address= 00h).

If a program loop is shorter than the selected loop resolution, the program waits until the end of the loop resolution prior to starting the next loop.

The timing diagram below illustrates the program flow execution.

Figure 18-3. Program Flow Timings



18.4.1.3 Instruction Execution Sequence

The execution of a NHET program begins with the first occurrence of the loop resolution clock, after the NHET 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 NHET goes into a wait state until the occurrence of the loop resolution clock and resumes normal execution.

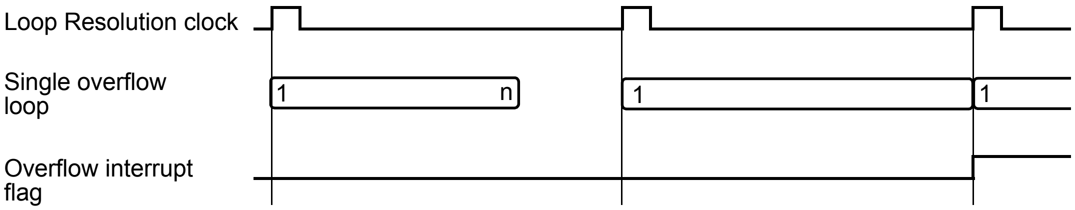
The number of instructions executed by a program must be between two loop resolution clocks. The program executes its first instruction in the high phase of the loop resolution clock. If the total time of execution for the program exceeds the limit, the program overflow flag is asserted and the execution of the program is stopped. The execution of the program is resumed by prefetching the instruction at location address 00h.

18.4.1.4 Program Overflow

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 HETEXC2. To maintain synchronization of the I/Os, this condition should never occur in a normal operation. During the software debug phase, this flag may be used to debug the programmed loop resolution value.

In case of program overflow, the NHET sequence is started again at address 00h.

Figure 18-4. Use of the Overflow Interrupt Flag



Note: Limitation on instruction falling on a program overflow

The pin action will not be taken if the instruction that caused the pin action falls on a program overflow. All other instructions such as updating the A, B, and T registers and the RAM will still be performed.

18.4.1.5 Architectural Restrictions

Note:

The size of a NHET program should be greater than one instruction, otherwise this instruction will be executed twice.

Note:

Any MOV instruction that modifies a field in the next instruction to be executed will incur a wait cycle.

Note:

Only one instruction (using high resolution) is allowed per high resolution pin. To enforce this restriction, the HR I/O architecture is programmed only once per resolution loop. This occurs when the first instruction is executed. It needs to be ensured that only one instruction accesses a given pin in HR mode. The HR structure takes the information from the first instruction and ignores the remaining instructions. Note that the succeeding instructions are ignored even if the first instruction uses en_pin_action = OFF (with hr_lr = HIGH). The hr_lr parameter must be set to LOW (independent of en_pin_action) for the first instruction to enable the succeeding instructions, which are assigned to the same pin.

Note:

Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (e.g. at addresses 1, 3, 5, 7, etc.)

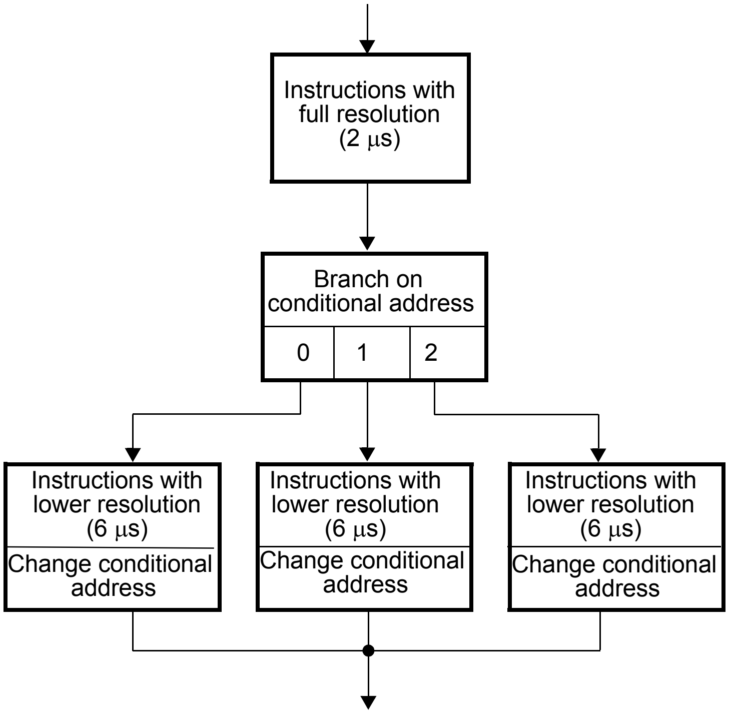
18.4.1.6 Multi-Resolution Scheme

The NHET has the capability to virtually extend the counter width by executing instructions not in every loop resolution period. This will decrease the timer accuracy, but help generating or measuring slow signals.

A lower resolution sequence consists of instructions that are not executed on every resolution loop, but only on every N resolutions. Unconditional Branch instruction and an index sequence, using a MOV64 instruction in each low resolution loop, is required to control this particular program flow. See Figure 18-5.

Note:
HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

Figure 18-5. Multi-Resolution Operation Flow Example

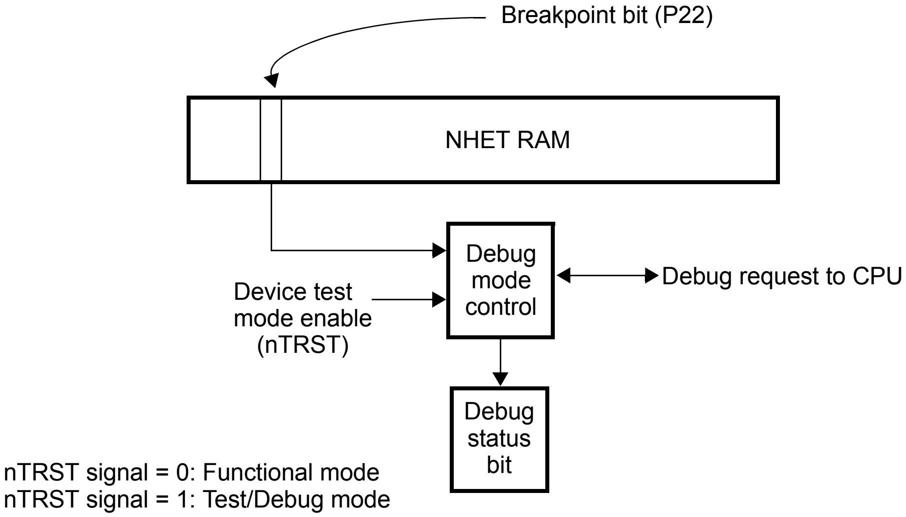

Instructions with full resolution (2 µs)
Branch on conditional address
0
1
2
Instructions with lower resolution (6 µs)
Change conditional address
Instructions with lower resolution (6 µs)
Change conditional address
Instructions with lower resolution (6 µs)
Change conditional address

18.4.1.7 *Debug Capability*

The NHET has built-in debug capability in its architecture that allows you to more easily debug your NHET program. See Figure 18-6.

This debug capability is based on the breakpoint bit that is included as part of each NHET instruction (bit P22 of any instruction). This bit can be programmed to a 1 to cause the NHET program to freeze after the current instruction is executed. This action will also cause all internal NHET state machines to freeze and send a debug request to the CPU. During the NHET debug (i.e., NHET breakpoint reached), you may still access the contents of all NHET control registers, including a register that contains the current NHET address to allow you to determine which instruction caused the breakpoint condition. If the Breakpoint bit is programmed to a 0, the instruction will then be executed and resume program operation.

Figure 18-6. Debug Control Configuration


Breakpoint bit (P22)
NHET RAM
Debug mode control
Debug request to CPU
Device test mode enable (nTRST)
Debug status bit
nTRST signal = 0: Functional mode
nTRST signal = 1: Test/Debug mode

When the device test mode is enabled by pulling the nTRST signal high and a breakpoint is reached in the NHET program, the debug status bit (Exception Control Register 2 (HETEXC2)) will be set and the debug request signal will be sent to the CPU. The CPU will wait for the CPU instruction boundary and then send the debug acknowledge signal back to the NHET. At this time the debugger may clear the debug status bit by writing a 1 to clear the bit in the NHET peripheral frame. This will allow the NHET to exit its debug mode when the debug acknowledge signal is de-asserted by the CPU.

This debug feature may be used to conditionally freeze the NHET operation by branching to a breakpoint instruction when a compare or capture condition exists. For example, an equality compare (ECMP) instruction could be used to branch to an instruction with the breakpoint bit set when the compare is equal. A software capture word (WCAP) instruction could be used to branch to an instruction with the breakpoint bit set when a certain external pin condition exists. When a breakpoint instruction is executed, the NHET freezes and the CPU enters debug mode. At this time the user may verify the state of the NHET RAM or peripheral register and check the current address to determine the source of the breakpoint.

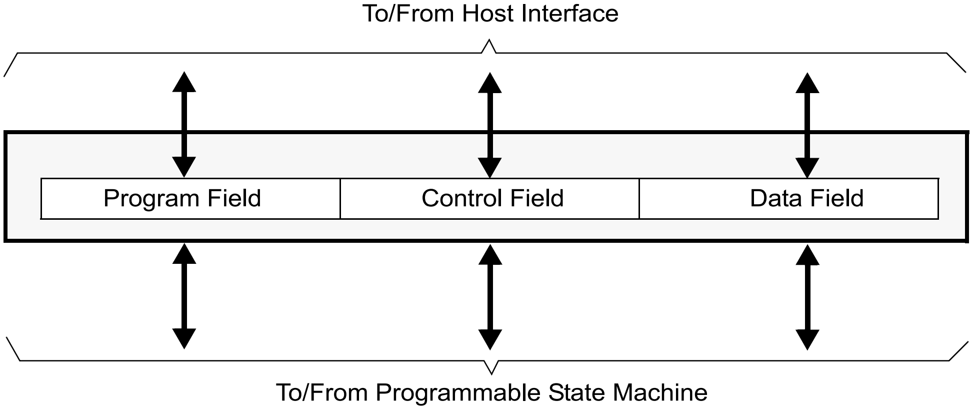
Note:
Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (e.g. at addresses 1, 3, 5, 7, etc.)

18.4.2 NHET RAM

The timer RAM uses 4 RAM banks, where each bank has two port access capability (see Section 18.4.2.1). This means that one RAM address may be written while another address is read. This gives the capability of writing back data into the current instruction while reading the next instruction. The RAM words are 96-bits wide, which are split into three 32-bit fields (program, control, and data). On application startup, the NHET program needs to be loaded into the NHET RAM, before the NHET program execution is switched on.

Note:
Write accesses: Only 32-bit write accesses to NHET RAM are allowed
Read accesses: 8-bit, 16-bit and 32-bit read assesses from NHET RAM are allowed

Figure 18-7. NHET RAM



18.4.2.1 NHET Memory Map and RAM Banking

Table 18-1 describes the NHET memory map.

Table 18-1. NHET Memory Map

Instruction Address	Program Field Address	Control Field Address	Data Field Address	Reserved Address	NHET 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

The NHET RAM consists of 4 RAM banks A, B, C and D and the succeeding 96 bit wide NHET instructions are placed into these banks in an interleaved order shown in Table 18-1. The reserved address of each instruction is not implemented as RAM.

Each bank is a two port RAM allowing a read and a write access in the same cycle. Therefore in the same cycle there could be several simultaneous operations:

– The currently executed instruction is written back to its bank X

– The next instruction is prefetched (i.e. read) from its bank Y

(See Section 18.4.1.3 for Instruction execution sequence information.)

Assuming a code execution, where X and Y are different banks, a NHET-external master (CPU, HTU, DMA or other master) can perform a third access to the NHET RAM in the same cycle: A read access from bank

X, a write access to bank Y or any access from the other two banks which are currently not used by the NHET execution.

If the external master performs the same type of access (read or write) to the same bank in the same cycle as the NHET execution unit, the NHET will assert wait states to this master as long as this bank is used by the NHET execution unit. After the NHET has completed accessing this bank it will then allow the master to access this bank.

The availability of 4 banks which are mapped to the NHET instruction addresses in the above described way and the possibility to adapt the execution flow of the NHET program during it's development enables to minimize the number of wait states for masters accessing the NHET RAM.

In general the advantage of this banked RAM concept is, that there is no need to stop the NHET execution during accesses of NHET-external masters.

Only in special situations of automatic read clear accesses there could be the need to insert wait cycles for the NHET program execution (see Section 18.4.4.3).

18.4.2.2 NHET Memory Map including Parity RAM

The NHET RAM can be protected by parity in the NHET Parity Control Register (HETPCR). Table 18-2 shows the address offsets for NHET RAM and NHET parity RAM from the NHET RAM start address of 0xFF460000.

Table 18-2. NHET memory map including parity RAM

Offset Address	
0x0000	NHET RAM
0x2000	NHET Parity RAM

The NHET implementation on this device allows for 128 instructions of 16 bytes each (including the 32-bit reserved field). The address range for the NHET RAM is from offset 0x0000 to 0x07FF.

The NHET Parity RAM covers and address range from offset 0x2000 to 0x27FF. (see also Section 18.4.2.4).

18.4.2.3 Parity Checking

The NHET module can detect parity errors in NHET RAM. As described in Section 18.4.2 the NHET allows 32-bit writes only. Therefore NHET RAM parity checking is implemented using one parity bit per 32-bit field in NHET RAM.

Even or odd parity selection for NHET 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 NHET 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 NHET execution unit makes a read access to NHET RAM, but also when a different master (e.g. CPU, HTU, DMA,...) performs the read access. If any 32-bit-word fails the parity check then an error is signaled to the ESM module. The NHET 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 NHET 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 NHET 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 NHET address is always aligned to a 32-bit word boundary.

The detection of a NHET parity error causes the following actions:

- An error is signaled to the ESM module.
- The Parity Address Register (HETPAR) is loaded with the address of the faulty NHET field.
- The NHET execution immediately stops and the instruction with the parity error and all following instructions in the current NHET loop are no more executed. The Turn-On/Off-Bit in the Global Configuration Register (HETGCR) is automatically cleared, so no further program loops are executed. All NHET internal flags are cleared.
- After a parity error all NHET pins, which are defined as output pins in the NHET Direction Register (HETDIR), which are not defined as open drain pins in the NHET Open Drain Register (HETPDR) and which are selected with the NHET Parity Pin Register (HETPPR), will remain outputs, but automatically change their levels in the following way:
  - If the NHET Pull Select Register (HETPSL) specifies 0 for the pin, it will switch to low level.
  - If the NHET Pull Select Register (HETPSL) specifies 1 for the pin, it will switch to high level.

  This behavior is independent on the value, which the NHET Pull Disable Register (HETPULDIS) specifies for the corresponding pin. The automatic level switch will be reflected in the NHET Data Output Register (HETDOUT).
- After a parity error all NHET pins, which are defined as open drain pins by the NHET Open Drain Register (HETPDR) and which are selected with the NHET Parity Pin Register (HETPPR), will switch to high impedance state. The automatic level switch will be reflected in the NHET Data Output Register (HETDOUT).

While debugging an application, the parity checking will still be performed if the access to the NHET RAM is done by the NHET or another master. If the debugger does the access, no parity error will be generated.

18.4.2.4 Testing Parity

To test the parity checking mechanism, the parity RAM has to be made accessible in order to allow manually inserting faults. This is done by a control bit in the NHET Parity Control Register (HETPCR). Only when this bit is set, the parity bits are accessible and mapped according to Table 18-3.

When in test mode (i.e. 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 NHET RAM.

Table 18-3. NHET Parity Bit Mapping

	Bits		
Address	**31**	**1**	**0**
offset + 0x00	Read 0		Program Field Instruction 0
offset + 0x04	Read 0		Control Field Instruction 0
offset + 0x08	Read 0		Data Field Instruction 0
offset + 0x0C	Read 0		Read 0
offset + 0x10	Read 0		Program Field Instruction 1
....	...		...

Each 32-bit NHET field has its own parity bit in the NHET Parity RAM as shown in Table 18-3. There are no parity bits for the reserved fields, since there is no physical NHET RAM for these fields.

18.4.2.5 Initializing Parity Bits

After device power up, the NHET RAM content including the parity bit cannot be guaranteed. In order to avoid parity failures, when reading NHET RAM, the RAM has to be initialized first. In order to allow for parity calculation during initialization, the parity functionality has to be enabled first. Initialization can simply be done by writing known values into the RAM by software and the corresponding parity bit will be automatically calculated. Section 18.2.2 describes that the NHET RAM must be loaded with the NHET program before the NHET could be started. This step will set the parity bits to the correct values. If the NHET program is smaller than the NHET RAM available on the device, then there is a remaining section in NHET RAM, where the parity bits are not initialized.

Another possibility to initialize the NHET 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.

18.4.3 Time Base

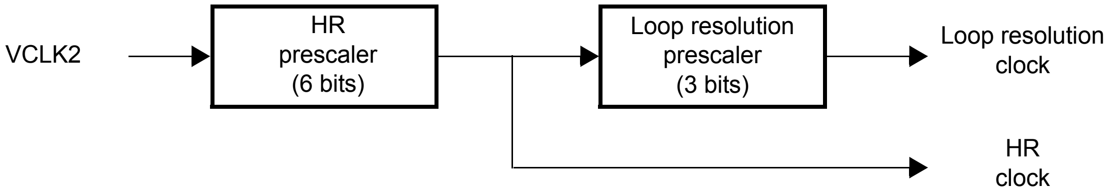
All NHET timings are derived from VCLK2 (see Figure 18-8).

Two prescalers are available to adjust the timer resolution clock for the program loop, and the high resolution (HR) clock for the HR I/O counters.

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 an 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 18-4.

Loop Resolution Clock The loop resolution clock defines the timebase for executing all instructions in a NHET program. Since instructions can be conditionally executed, the longest path through the NHET 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 18-5.

Figure 18-8. Prescaler Configuration



The loop resolution period (LRP) is typically determined by the number of cycles required to complete the worst-case NHET application software loop. Therefore, the prescaler numbers have to be chosen such that the number of time slots (ts) in LRP is greater than the number of cycles required in a loop. Typically, the smallest HR prescaler divide rate (hr) is used, that will still allow the appropriate number of cycles.

Following abbreviations and relations are used in this document:

- hr = high resolution prescale divide rate
  hr = 1, 2, 3, 4,..., 63, 64
- lr = loop resolution prescale divide rate
  lr = 1, 2, 4, 8, 16, 32, 64,128
- ts = number of time slots available in one NHET loop (LRP)

$$\mathrm{ts} = \mathrm{hr} \times \mathrm{lr} \qquad \text{(EQ 1)}$$

- HRP = high resolution clock period

$$\mathrm{HRP} = \frac{\mathrm{hr}}{\mathrm{VCLK2}} \qquad \text{(EQ 2)}$$

– LRP = loop resolution clock period

$$LRP = \frac{hr \times lr}{VCLK2} = lr \times HRP \qquad \text{(EQ 3)}$$

The divide rates hr and lr can be defined in the HETPFR register according to the tables given in the following:

Table 18-4. HR Prescale Factor Codes

HR Prescale Factor Code						HR Prescale Divide Rate hr
5	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

Table 18-5. Loop Resolution Prescale Factor Codes

Loop Resolution Prescale Factor Code			Loop-Resolution Prescale Divide Rate lr
2	**1**	**0**	
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

18.4.3.1 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 NHET 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:

$$hr = 2 \rightarrow HRP = \frac{hr}{VCLK2} = \frac{2}{32MHz} = 62.5ns$$

$$\mathrm{lr} = 128 \rightarrow \mathrm{lr} \times \mathrm{HRP} = 128 \times 62.5\mathrm{ns} = 8\mu\mathrm{s}$$

$$\mathrm{ts} = \mathrm{hr} \times \mathrm{lr} = 2 \times 128 = 256$$

$$\mathrm{hr} = 2, \mathrm{lr} = 128 \rightarrow \mathrm{HETPFR}[31:0] = 0\mathrm{x}00000701$$

If in the example above, the resolution needs to move from 8 μs to 4 μs, then only 128 time slots will be available.

18.4.3.2 The 7-Bit HR Data Field

The instruction execution examples of ECMP (Section 18.4.5.8), MCMP (Section 18.4.5.9), PCNT (Section 18.4.5.11), PWCNT (Section 18.4.5.10), and WCAP (Section 18.4.5.12) show, that the 7-bit HR data field can generate or measure high resolution delays within one NHET loop LRP. The last section showed that

$$\mathrm{LRP} = \mathrm{lr} \times \mathrm{HRP}$$

So there are lr high resolution clocks (HRP) within the NHET loop 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. Table 18-6 shows which bits of the HR data field are not needed by the hardware, 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 (e.g. for PCNT, WCAP)
- Or interpreted as 0 for HR compare (e.g. for ECMP, MCMP. PWCNT)

Table 18-6. Interpretation of the 7-Bit HR Data Field

Loop Resolution Prescale divide rate (lr)	Bits of the HR data field							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	V	X	X	X	X	X	X	0 to 1
4	V	V	X	X	X	X	X	0 to 3
8	V	V	V	X	X	X	X	0 to 7
16	V	V	V	V	X	X	X	0 to 15
32	V	V	V	V	V	X	X	0 to 31
64	V	V	V	V	V	V	X	0 to 63
128	V	V	V	V	V	V	V	0 to 127
Weight factor as fraction of HR cycles in one loop	1/2	1/4	1/8	1/16	1/32	1/64	1/128	

V = Valid bit; X = Non-relevant bit (ignored)

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
$= lr \cdot (D[6] \cdot 1/2 + D[5] \cdot 1/4 + D[4] \cdot 1/8 + D[3] \cdot 1/16 + D[2] \cdot 1/32 + D[1] \cdot 1/64 + D[0] \cdot 1/128)$
$= 8 \cdot (1 \cdot 1/2 + 0 \cdot 1/4 + 1 \cdot 1/8 + 0 \cdot 1/16 + 0 \cdot 1/32 + 0 \cdot 1/64 + 0 \cdot 1/128)$
= 5 HRPs

18.4.4 Host Interface

The host interface controls all communications between timer-RAM and masters accessing the NHET RAM. It includes following components:

- Two user programmable 6-bit (HETPFR.5:0) and 3-bit (HETPFR.10:8) clock pre-scalers for high resolution (HR) and loop resolution clocks (LR).
- A bit (HETGCR.16) for controlling whether the NHET is configured as a master or a slave.

18.4.4.1 CPU Access to Timer-RAM

Note:
Write accesses: Only 32-bit write accesses to NHET RAM are allowed.
Read accesses: 8-bit, 16-bit and 32-bit read accesses from NHET RAM are allowed

18.4.4.2 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 (e.g.CPU, DMA or any other master) 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 NHET 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. At a time 3 masters can do 64-bit reads in an interleaved manner (e.g. 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. At the same time, read to PF field and writes to all the fields are allowed by the 4th master in this scenario.

18.4.4.3 Automatic Read Clear Feature

The NHET provides a feature allowing to automatically clear the data field immediately after the data field is read by an external master. 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.

Automatic read clear with 32-bit read access:

1. Set the control bit of instruction X
2. Master reads the data field of instruction X. This data field is automatically cleared in NHET RAM.

Automatic read clear with 64-bit read access:

1. Set the control bit of instruction X
2. Master reads the control field of instruction X. At this time the data field of instruction X is loaded into the shadow register (see description of the 64-bit-read access) and the data field location in the NHET RAM is automatically cleared.

In order not to loose captured data it is ensured that the NHET program execution will capture a new value to a data field if the master by accident performs an auto-read clear access at the instruction which is currently executed.

Application example for the automatic read clear feature:

The master reads a period or pulse value from a PCNT data field which is then automatically cleared. This means that the master can detect if the value is new or was already read by the master.

18.4.4.4 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the 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.

18.4.4.5 Power-Down

The NHET uses one clock to drive both registers and logic. Local low power mode is asserted by setting the corresponding bit in the PCR module, which switches off the clock. This leaves the module in a static state in which it consumes the lowest possible current. Local power down mode allows still accessing the registers, since the clock will be temporarily switched on during a register access.

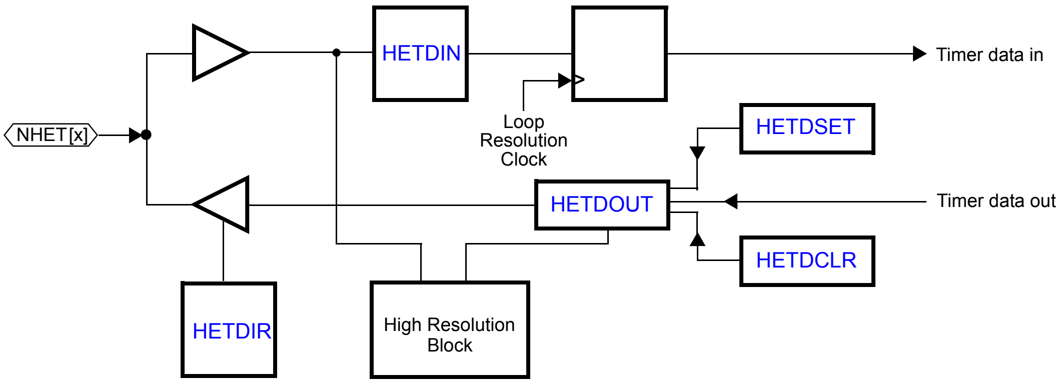
After setting the turn-off bit in the Global Configuration Register (HETGCR), it is required to delay until the end of the timer program loop before putting the NHET in power-down mode. This can be done by waiting until the NHET Current Address (HETADDR) becomes zero.

18.4.5 I/O Control

The NHET has up to 32 pins. Refer to device specific data sheets for information concerning the number of NHETIO available. All the NHET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins NHET[31] to NHET[0]. This structure allows any NHET instruction to use these I/Os with a loop resolution accuracy. See Figure 18-9 for an illustration of the I/O control. In addition all 32 I/Os have a special HR structure based on the HR clock.

Figure 18-9. I/O Control



The general purpose I/Os, pins NHET [31] to NHET [0], can be used by the CPU as general purpose inputs or outputs using the NHET Data Input Register (HETDIN) for reading and NHET Data Output Register (HETDOUT), NHET Data Set Register (HETDSET) or NHET Data Clear Register (HETDCLR) for writing, depending on the type of action to perform. The NHET pins used as general purpose inputs are sampled on each VCLK2 period.

Note: Using HETDSET and HETDCLR

The NHET Data Clear Register (HETDCLR) and NHET Data Set Register (HETDSET) can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. Actually, when the application needs to set or to reset multiple pins without changing the value of the others pins, the first possibility is to read NHET Data Output Register (HETDOUT), modify the content (AND, OR, etc.), and write the result into NHET Data Output Register (HETDOUT). This solution could be interrupted by a function modifying the same register, which will result in a data coherency problem.

Using the NHET Data Set Register (HETDSET) or NHET 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.

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 */
```

The following chapters describe the pin structures when used in functional mode.

18.4.5.1 Loop Resolution Structure

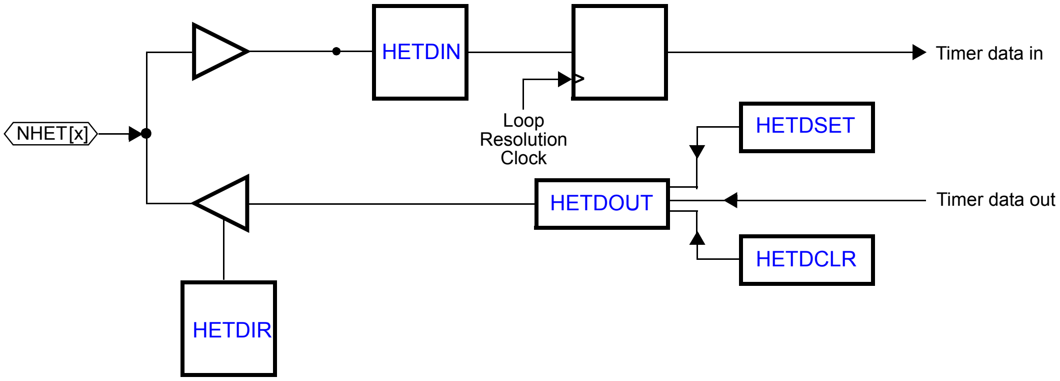
The NHET uses the pins NHET [31:0] as input and/or output by the way of the instruction set. Actually, each pin could monitor the NHET program or could be monitored by the NHET program. By using the I/O register of the NHET, the CPU is able to interact with the NHET program flow.

When an action (set or reset) is taken on a pin by the NHET program, the NHET will modify the pin at the rising edge of the next resolution clock.

When an event occurs on a NHET I/O pin, it is taken into account at the next rising edge of the resolution clock.

The structure of each pin is shown in Figure 18-10.

Figure 18-10. NHET Loop Resolution Structure for Each Bit



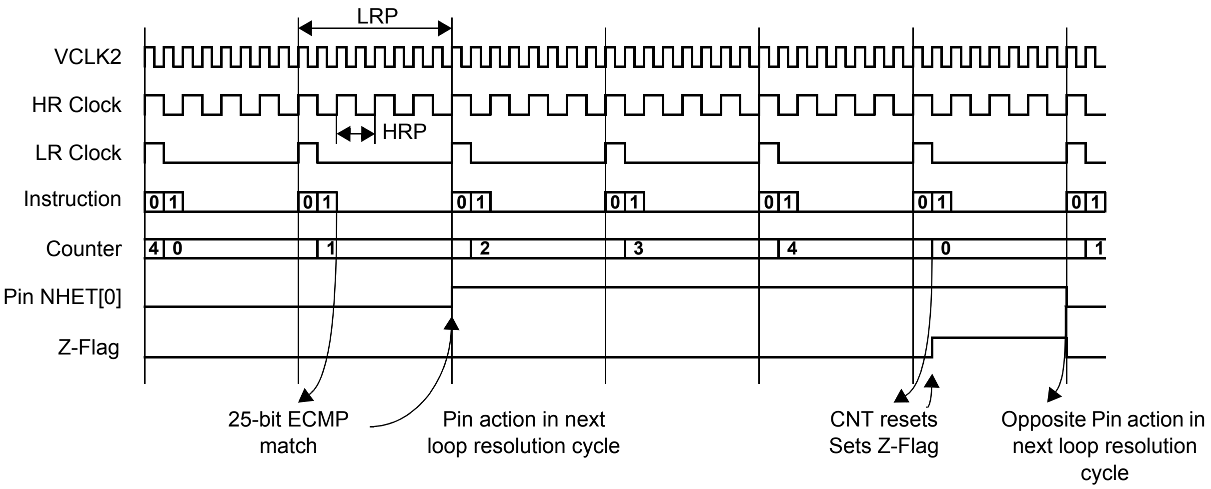
The example in Figure 18-11 shows a simple PWM generation with loop resolution accuracy. The corresponding program can be found below.

```
HETPFR[31:0] register = 0x201 → lr=4 and hr=2 → ts = 8

NHET 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=CC0,
             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
```

Note: NHET[x] pin is referenced as CC[x] in the NHET assembler mnemonic.

The CNT and ECMP instructions are executed once each loop resolution cycle. When the CNT instruction is executed, the specified register (A) and its data field is 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 keeps on incrementing each loop resolution cycle. When the data field overflows (max + 1), then the Z-flag is set by the 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 18-11. Loop Resolution Instruction Execution Example



18.4.5.2 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 PCNT or WCAP) or HR output compares (using ECMP, MCMP or PWCNT).

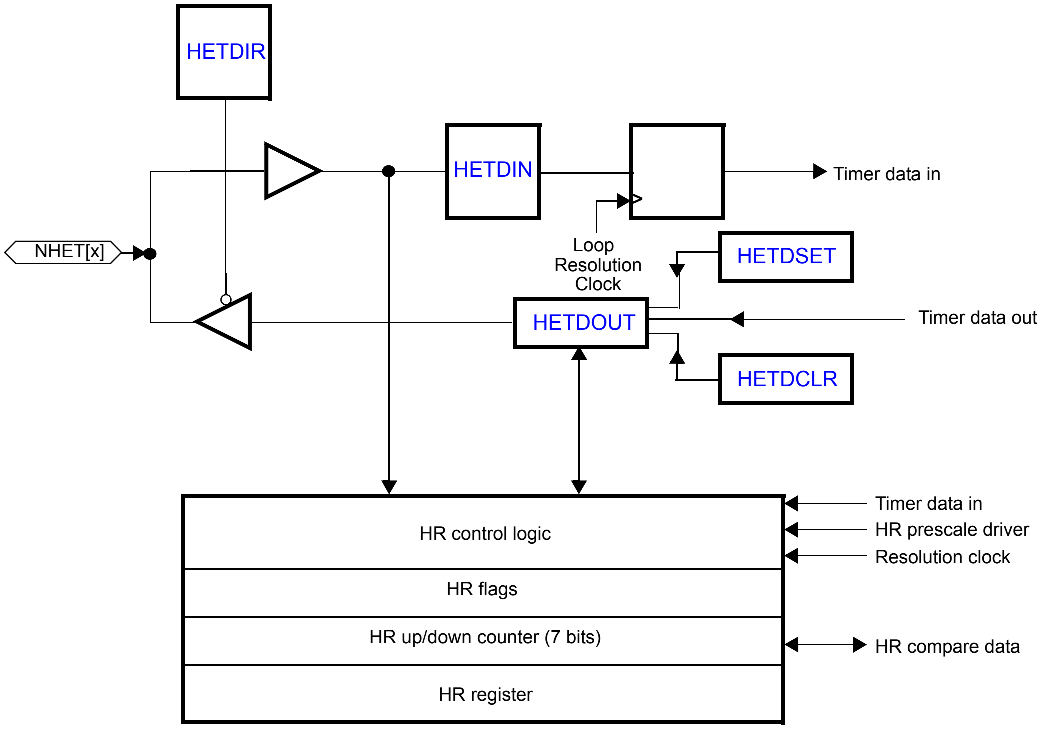
Four HR instructions have a dedicated hr_lr bit (HR/low resolution; program field bit 8) allowing operation either in HR mode or in standard resolution mode by ignoring the HR field. Those instructions are ECMP, MCMP, PWCNT and WCAP. By default, the hr_lr bit value is 0 which implies HR operation mode. However, setting this bit to one, allows the use of several HR instructions on a single HR pin. Only one instruction will be allowed to operate in HR mode (i.e., bit set to 0), but the others instructions can be used in standard resolution mode (i.e., bit set to 1).

18.4.5.3 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR block 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:
Only one instruction (using high resolution) is allowed per high resolution pin. To enforce this restriction, the architecture shown below is programmed only once per resolution loop. This occurs when the first instruction is executed. You must ensure that only one instruction accesses a given pin in HR mode. The HR structure takes the information from the first instruction and ignores the remaining instructions. Note that the succeeding instructions are ignored even if the first instruction uses en_pin_action = OFF (with hr_lr = HIGH). The hr_lr parameter must be set to LOW (independent of en_pin_action) for the first instruction to enable the succeeding instructions, which are assigned to the same pin.

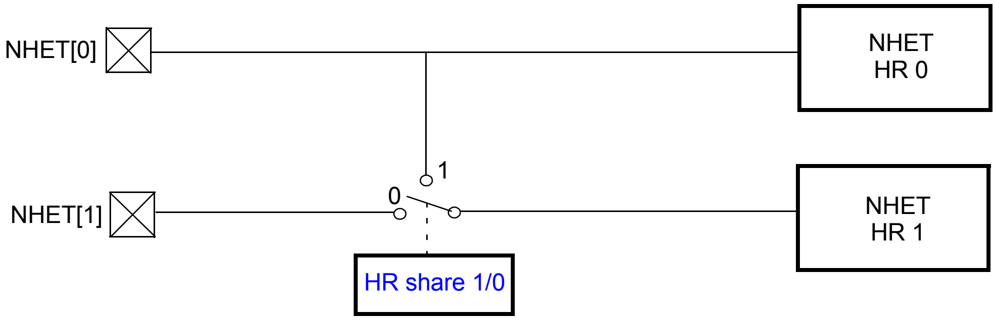
Figure 18-12. HR I/O Architecture



18.4.5.4 HR Structures Sharing

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 18-13.

Figure 18-13. Example of HR Structure Sharing for NHET Pins 0/1



The following program gives an example how the HR share feature (NHET[0] HR structure and NHET[1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=CC0 }
L01 PCNT { next=L00, type=fall2rise, pin=CC1 }
```

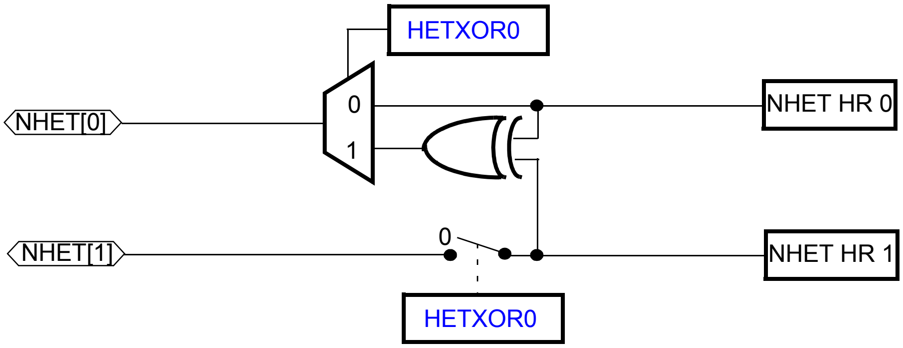
The NHET[1] HR structure is also connected to the NHET[0] pin. The L00_PCNT data field is able to capture a high pulse and the L01_PCNT captures a low pulse on the **same** pin (NHET [0] pin).

18.4.5.5 XOR-shared HR Structure

Usually the NHET 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 18-14. 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 18-15.

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 18-14. XOR-shared HR I/O



The following NHET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in Figure 18-15.

```
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 }
```

NHET Settings and output signal calculation for this example program:

- Pin NHET[0] and NHET[1] are XOR-shared.
- HETPFR[31:0] register = 0x700 → lr=128 and hr=1 → time slots ts = 128
- PWM period (determined by CNT_max field) = (22+1) · LRP = 2944 HRP
- Length of high pulse of (NHET[0] XOR NHET[1]) =

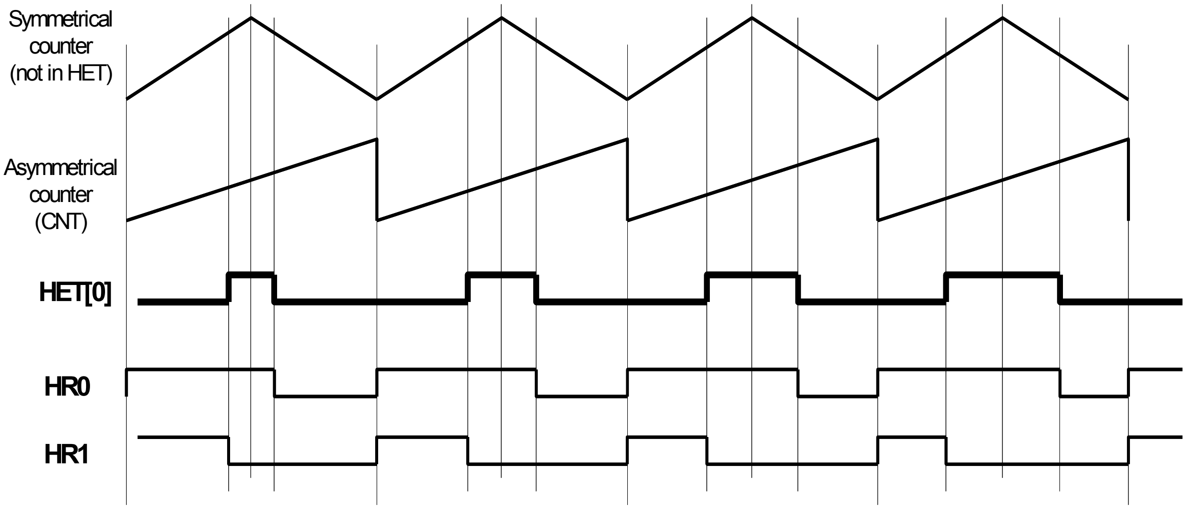
  LH = (17·LRP+115·HRP) - (5·LRP+13·HRP)

  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 18-15 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 NHET. 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 NHET[0]. Notice that the pulses of signal NHET[0] are centered about the axis of symmetry.

Figure 18-15. Symmetrical PWM with XOR-sharing Output



18.4.5.6 Loop Back Mode

The loop back feature can be used by the application to monitor an NHET 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.

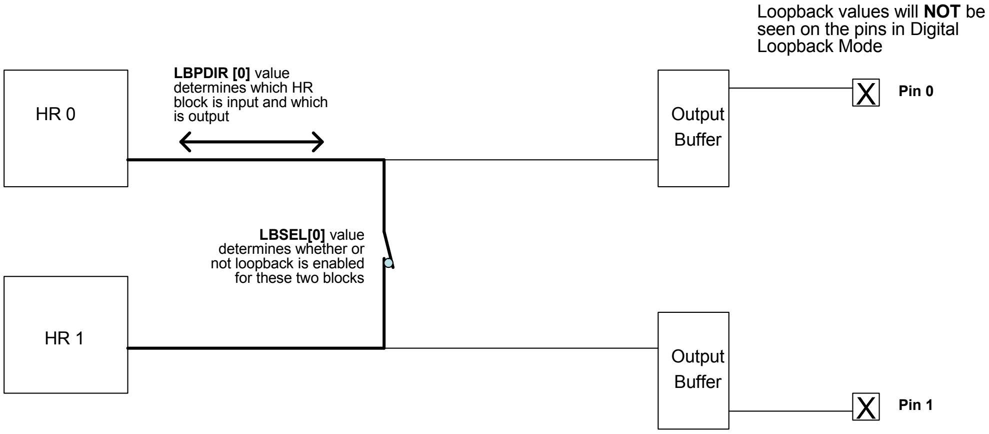
Loopback mode is activated between two high resolution structures by setting LBPSEL[x] to ‘1’ in the HETLBPSEL Register for the corresponding structure pair. The **direction** of the loopback between the two structures in the structure pair is determined by the value of LBPDIR[x] in the HETLBPDIR Register.

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 18-16 below 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.

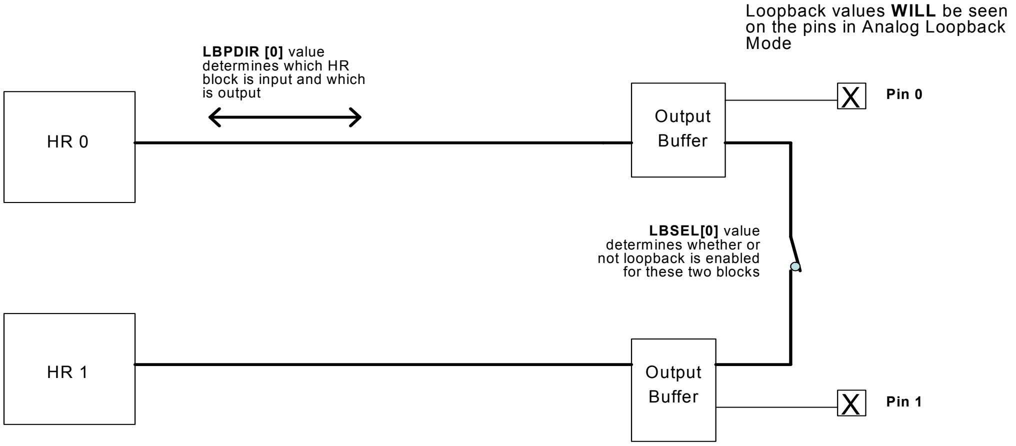
Figure 18-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = ‘0’


Loopback values will **NOT** be seen on the pins in Digital Loopback Mode
HR 0
LBPDIR [0] value determines which HR block is input and which is output
Output Buffer
Pin 0
LBSEL[0] value determines whether or not loopback is enabled for these two blocks
HR 1
Output Buffer
Pin 1

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 20 below 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 18-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = ‘1’


Loopback values WILL be seen on the pins in Analog Loopback Mode
LBPDIR [0] value determines which HR block is input and which is output
HR 0
Output Buffer
Pin 0
LBSEL[0] value determines whether or not loopback is enabled for these two blocks
HR 1
Output Buffer
Pin 1

Note:

- o The loop back direction can be selected independent of the HETDIR register setting.
- o The pin which is not driven by the NHET output pin actions can still be used as normal GIO pin.

18.4.5.7 Edge Detection

The following shows that the NHET allows maximum one edge per NHET loop for input signals, which limits the maximum frequency for input signals to

$$f\_max = 1/(2 \times LRP)$$

and both the pulse high and pulse low durations need to be at least one LRP in duration in order for the edge detect algorithm to work.

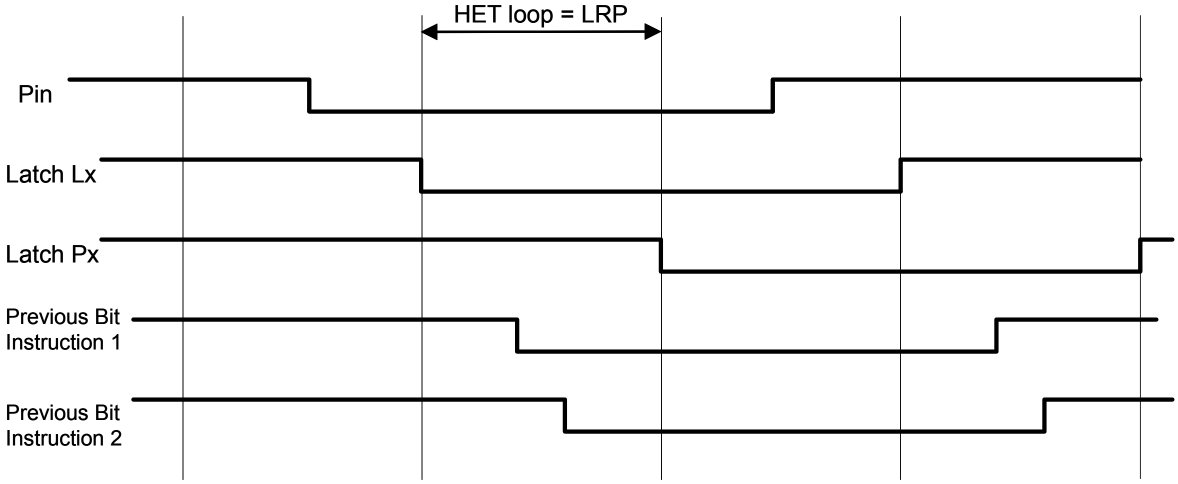
Apart from that the user does not have to care about how edge detection works as long as the instructions “previous bit” information is not used, which is useful for some special applications.

The following NHET instructions need to detect if a signal edge occurred in the preceding NHET loop on the assigned NHET pin:

ACNT, APCNT, BR, ECNT, PCNT, WCAP, WCAPE

Figure 18-18 illustrates how the edge detection works. Every NHET pin x has an internal latch Lx, which synchronizes the pin signal to the next NHET loop, and an internal latch Px, which synchronizes it to the NHET loop following the one in which the signal has been latched to Lx. Once the instruction is executed it will decode Lx/Px in the following way: If Lx=Px then no edge has occurred, if Lx/Px=0/1 then a falling edge has occurred, if Lx/Px=1/0 then a rising edge has occurred.

Figure 18-18. Edge Detection



The above listed instructions have a so called previous bit (C25) in the control field. Every time the instruction is executed the previous bit is updated as follows: If Lx/Px=0/1 or Lx/Px=0/0 then the previous bit is set to zero. If Lx/Px=1/0 or Lx/Px=1/1 then the previous bit is set to one.

So the previous bit is not used as input information for the internal edge detection. Instead the edge detection uses the Lx and Px latches. The previous bit just stores the type of the last edge (rising or falling) in the control field.

Application examples, which use the previous bit information:

- Two instructions x and y are assigned to the same input signal on the same pin (e.g. using the HRSHARE feature). An external master (CPU, DMA, ...) first reads the control and data field of instruction x in a 64-bit access and then reads control and data field of instruction y in a 64-bit access. To determine if the captured values of data field x and y belong to the same edge type (rising or falling) the master can compare the previous bits of instruction x and y.
- The event-parameter of a WCAP instruction is set to BOTH, which means that the WCAP captures time stamps after rising and falling edges to its data field. If an external master reads the control and data field of WCAP in a 64-bit access then it can find out from the previous bit whether the captured data belongs to a rising or falling edge.

At the time the turn on bit is set in the Global Configuration Register (HETGCR) the hardware ensures that latches Lx and Px are correctly initialized according to the current level of the pin. From this time the edge detection is performed as described above if the pin is configured as input pin.

18.4.5.8 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 nsec

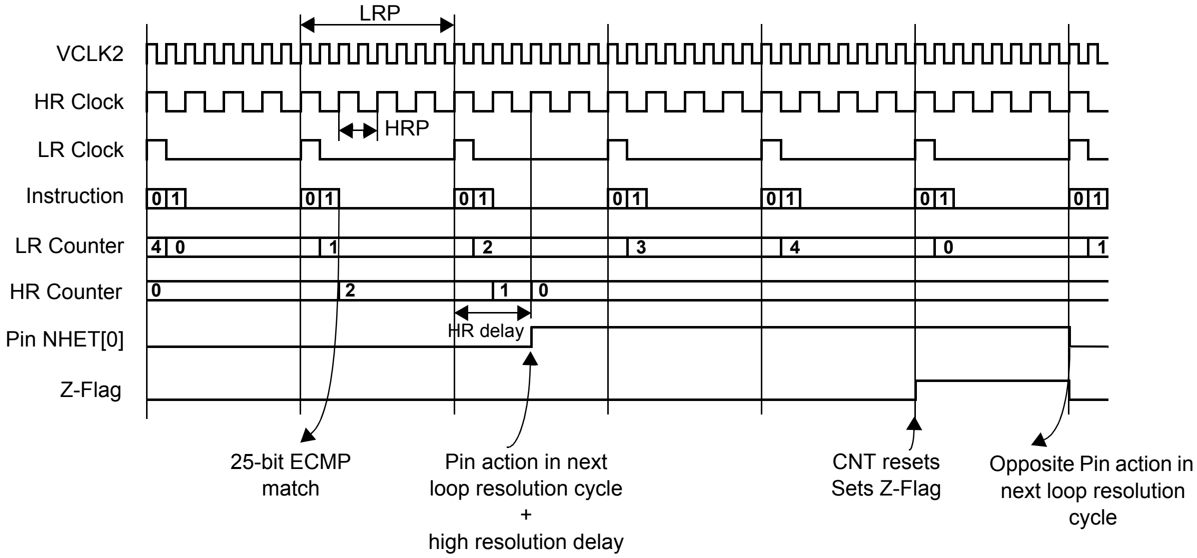
LRP = (hr · lr) / VCLK2 = 8 / 32 MHz = 250 nsec

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 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 18-19 the 25-bit compare value is one and the 7-bit HR compare value is two. According to Section 18.4.3.2, 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 18-19. ECMP Execution Timings



```
HETPFR[31:0] register = 0x201 → lr=4 and hr=2 → ts = 8

NHET 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=CC0,
             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 a 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.

18.4.5.9 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 resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

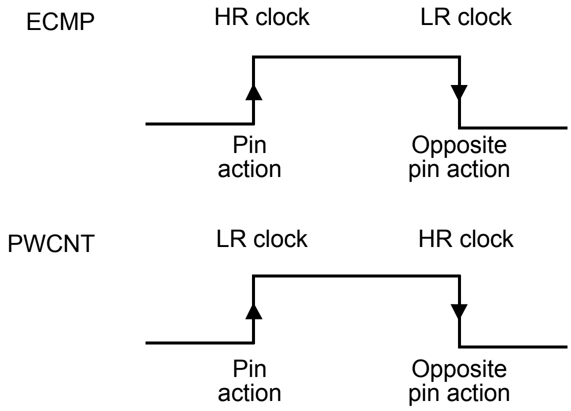
The MCMP instruction avoids the missing pulse problem of the ECMP instruction (see previous example), however the duty cycle of the signal might not be exact for one PWM period. The benefit of the MCMP is that it avoids adding another instruction to do the duty cycle update synchronously.

18.4.5.10 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 18-20. 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 18-20. High/Low Resolution Modes for ECMP and PWCNT



18.4.5.11 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 18-21, 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, 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 “The 7-bit HR data field”)

Figure 18-21 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 18-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)

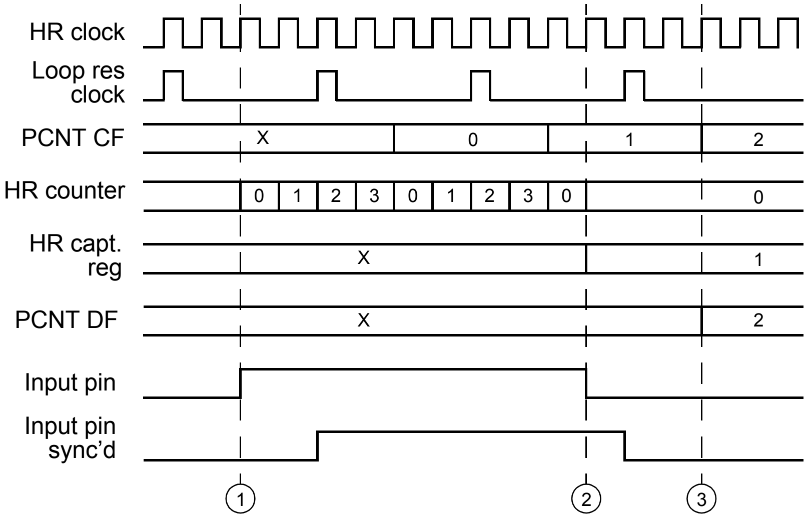
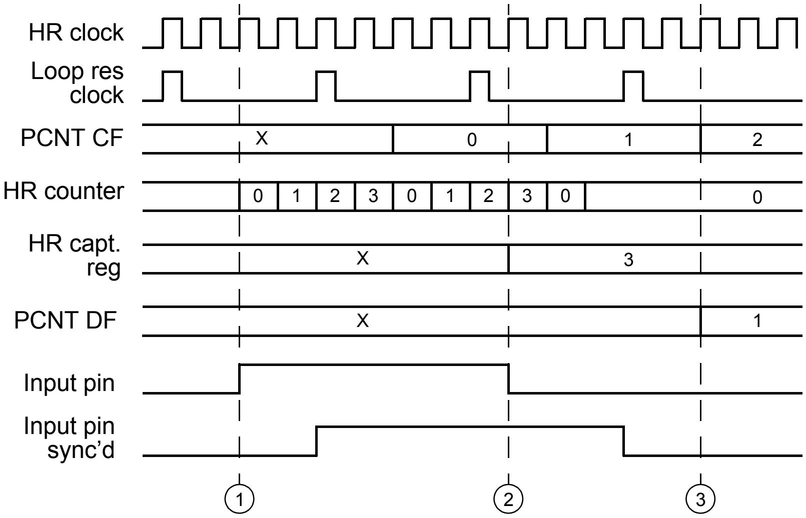


Figure 18-22 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

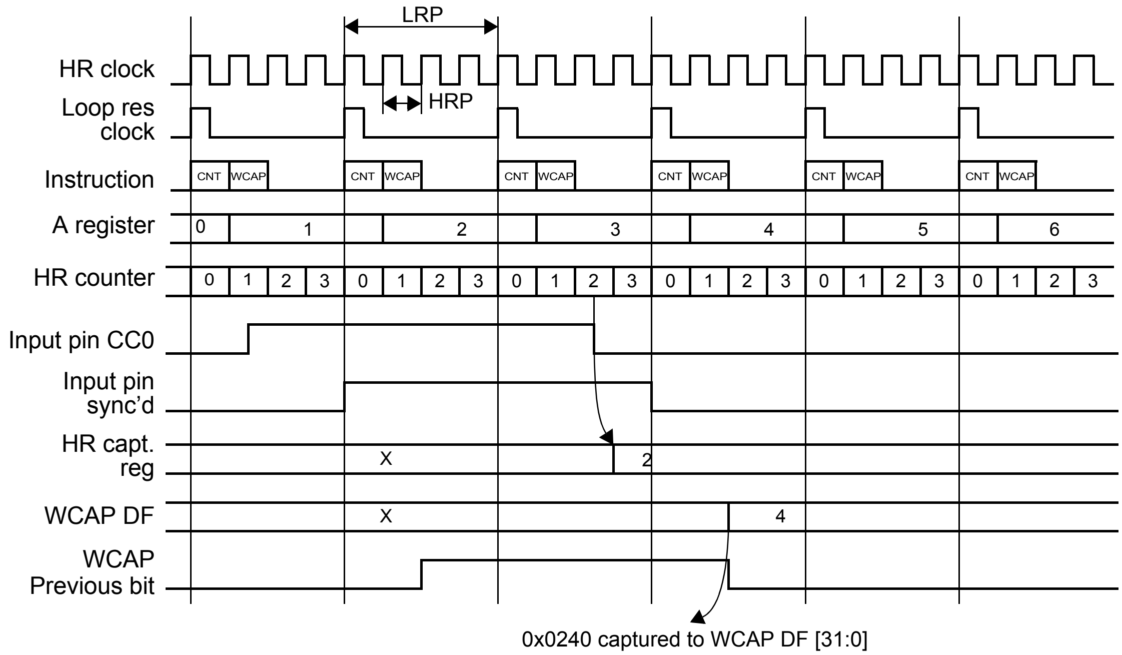
Figure 18-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)



18.4.5.12 *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 18-23).

Figure 18-23. WCAP Instruction Timing



```
HETPFR_register = 0x0200 → lr = 4, hr = 1, ts = 4
```

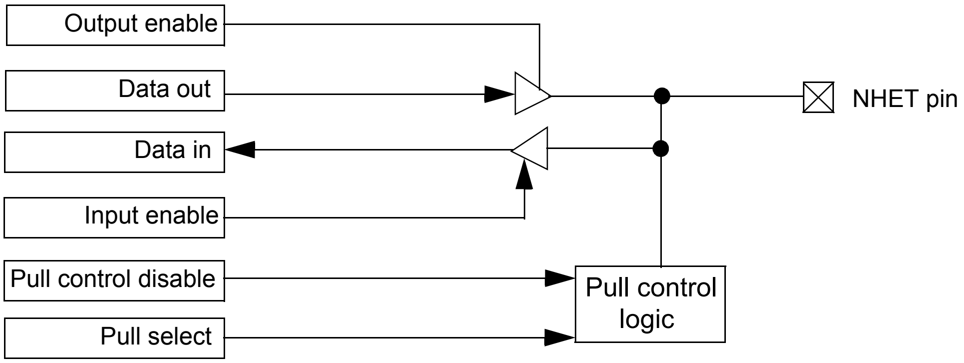
NHET Program:

```
L00 CNT  {reg=A, max=01ffffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=CC0,
          data=0}
```

In the timing of 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 evaluates an edge by comparing its Previous bit with the sync'd input signal. In Figure 18-23, 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].

18.4.5.13 I/O Pull Control Feature

Figure 18-24. 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 NHET Pull Disable Register (HETPULDIS). In this case, if the corresponding bit in the NHET Pull Select Register (HETPSL) is set, the pin will have a pull-up; if the bit in the NHET Pull Select Register (HETPSL) is cleared, the pin will have a pull-down. If the bit in the NHET Pull Disable Register (HETPULDIS) is set, there is no pull-up or pull-down on the pin.
- Input buffer: The input buffer is disabled only if the pin direction is set as input AND the pull control is disabled AND pull down is selected as the pull bias. In all other 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 NHET Direction Register (HETDIR) is set AND the open-drain feature (NHET Open Drain Register (HETPDR)) is not enabled.

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 18-7.

Table 18-7. Input Buffer, Output Buffer and Pull Control Behavior

Device under Reset?	Pin Direction (DIR)	Pull Disable (PULDIS)	Pull Select (PULSEL)	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Device- and module-specific	Disabled	Pulldown
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

18.4.5.14 Open-Drain Feature

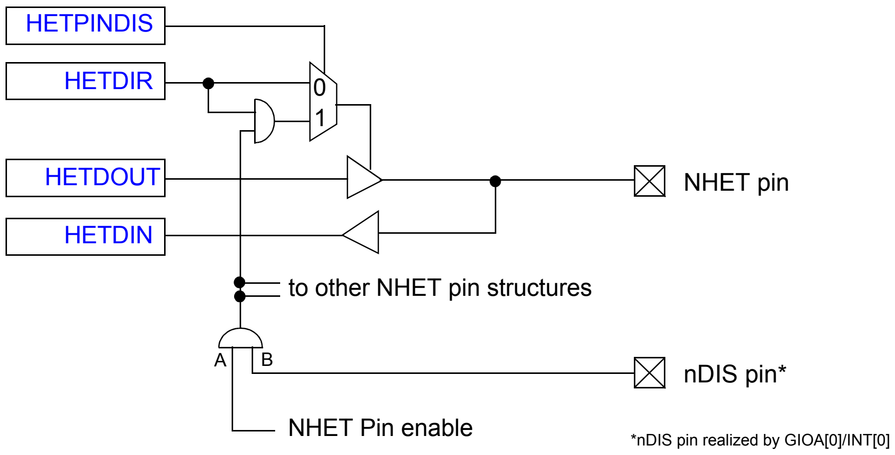
The following apply if the open-drain feature is enabled on a pin, i.e. the corresponding bit in the NHET Open Drain Register (HETPDR) is set

- Output buffer is enabled if a low signal is being driven internally to the pin.
- The output buffer is disabled if a high signal is being driven internally to the pin.

18.4.5.15 NHET 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 18-25. NHET Pin Disable Feature Diagram


HETPINDIS
HETDIR
0
1
HETDOUT
HETDIN
NHET pin
to other NHET pin structures
A
B
nDIS pin*
NHET Pin enable
*nDIS pin realized by GIOA[0]/INT[0]

The following truth-table shows the conditions for the output buffer to be enabled/disabled.

Table 18-8. NHET Pin Disable Feature

HETPINDIS.x	nDIS Pin (Input)[a]	NHET PIN ENA	HETDIR.x	Output Buffer
0	X	X	0	Disabled
0	X	X	1	Enabled

Table 18-8. NHET Pin Disable Feature

HETPINDIS.x	nDIS Pin (Input)[a]	NHET PIN ENA	HETDIR.x	Output Buffer
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

[a] X means state of the pin is not relevant

An interrupt capable device I/O pin (GIOA[0]/INT[0]) is used for the nDIS signal, which is connected internally to the NHET (see above diagram). An active low level on nDIS is intended to signal an abnormal situation as described above. All NHET pins, which are selected with the NHET 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:

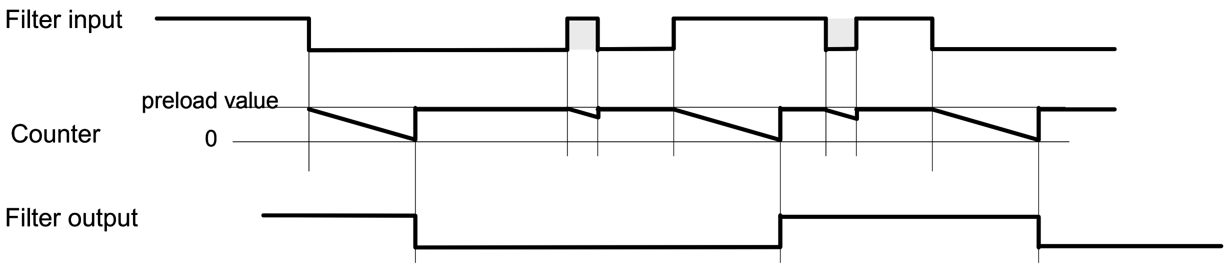
- Software detects, by reading the HETDIN register of the GIO pin, that the level on nDIS is inactive (i.e. high).
- Software sets bit HET PIN ENA to deactivate the high impedance state of the pins.

18.4.6 Suppression Filters

Each NHET pin is equipped with a suppression filter. If the pin is configured as an input it enables to filter out pulses, which are smaller than a programmable duration. Each filter consists of a 10-bit down counter, which starts counting at a programmable preload value and is decremented with VCLK2.

- 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 output 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 is not taking place. The counter resumes at the preload value until it detects an opposite pin action on the input signal again.
- The filter output signal is delayed compared to the filter input signal, depending on the counter clock frequency (VCLK2) and the programmed preload value.
- The accuracy of the output signal is +/- the counter clock frequency.

Figure 18-26. Suppression Filter Counter Operation



The following table gives examples for a 100 MHz VCLK2 frequency.

Table 18-9. 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 nsec, 20 nsec, …, 10.22 usec, 10.23 usec 50 MHz, 25 MHz, …, 48.924 kHz, 48.876 kHz
2	50.0 MHz	20 nsec, 40 nsec, …, 20.44 usec, 20.48 usec 25 MHz, 12.5 MHz, …, 24.462 kHz, 24.414 kHz
3	33.3 MHz	30 nsec, 60 nsec, …, 30.66 usec, 30.69 usec 16.7 MHz, 8.3 MHz, …, 16.308 kHz, 16.292 kHz

18.4.7 Interrupts and Exceptions

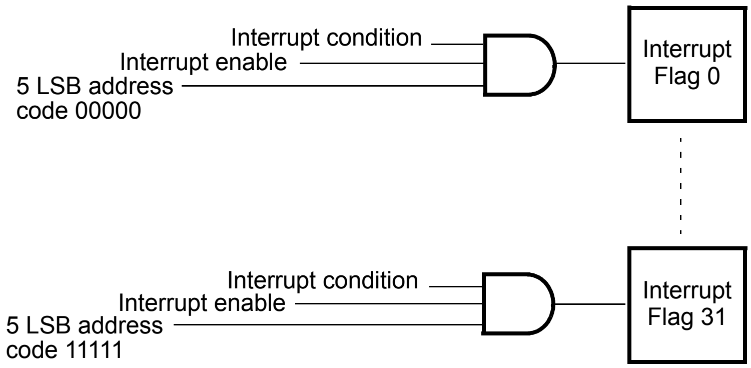
NHET interrupts can be generated by any instruction that has an interrupt enable bit in its instruction format. When the interrupt condition in an instruction is true and the interrupt enable bit of that instruction is set, an interrupt flag is then set in the NHET Interrupt Flag Register (HETFLG). The address code for this flag is determined by the five LSBs of the current timer program address. The flag in the NHET Interrupt Flag Register (HETFLG) is set even if the corresponding bit in the NHET Interrupt Enable Set Register (HETINTENAS) is zero. To generate an interrupt the corresponding bit in the NHET Interrupt Enable Set Register (HETINTENAS) must be one.To execute the interrupt service routine, the main CPU must first determine which source created the interrupt request. This is done by reading the NHET Offset Index Priority Level 1 Register (HETOFF1) or NHET Offset Index Priority Level 2 Register (HETOFF2) which provides the number of the source. Reading the offset register will automatically clear the corresponding interrupt flag which created the request. However, if the offset registers to check the interrupt source are not used, the flag should be cleared once the interrupt has been serviced.

Table 18-10. 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 18-48.

Figure 18-27. 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. Two interrupt requests are generated by the NHET to the Vectored Interrupt Module (VIM).

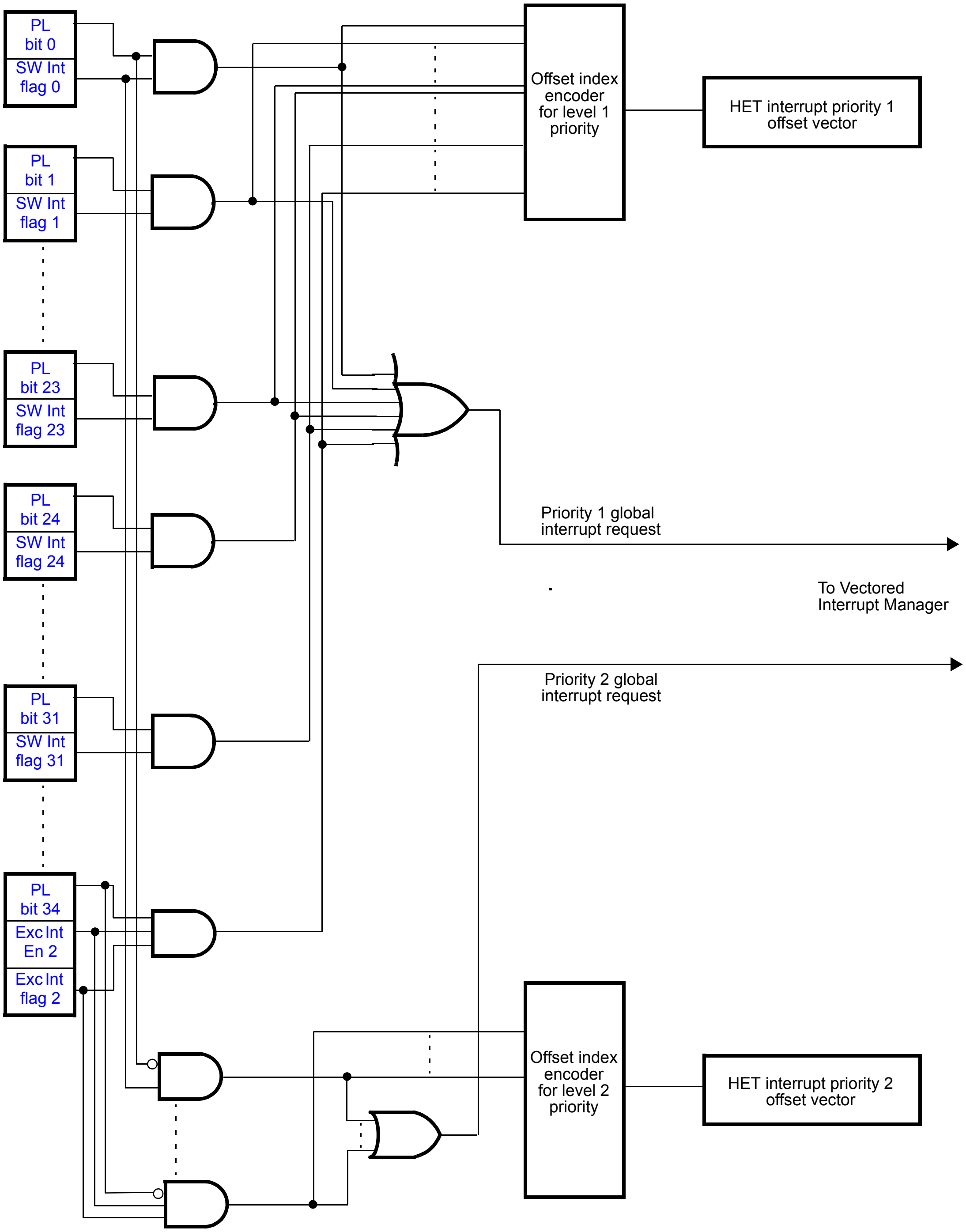
In addition to the interrupts generated by the instructions the NHET can generate three additional exceptions:

- Program overflow
- APCNT underflow (see Section 18.5.1.2)
- APCNT overflow (see Section 18.5.1.3)

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 18-28.

Figure 18-28. Interrupt Flag/Priority Level Architecture


PL
bit 0
SW Int
flag 0
PL
bit 1
SW Int
flag 1
PL
bit 23
SW Int
flag 23
PL
bit 24
SW Int
flag 24
PL
bit 31
SW Int
flag 31
PL
bit 34
Exc Int
En 2
Exc Int
flag 2
Offset index encoder for level 1 priority
HET interrupt priority 1 offset vector
Priority 1 global interrupt request
To Vectored Interrupt Manager
Priority 2 global interrupt request
Offset index encoder for level 2 priority
HET interrupt priority 2 offset vector

18.4.8 NHET Requests to DMA and HTU

As described in Section 18.7.3 the majority of the NHET instructions is able to generate a transfer request to the High End Timer Transfer Unit (HTU) and/or to the DMA module when a instruction specific condition is true. One NHET instruction can select one of 8 request lines by programming the “reqnum” parameter (see Section 18.7.3). The “request” field in an instruction is used to disable, enable the selected request line or to generate a quiet request (see Section 18.7.2). Quiet requests can be used by the HTU, but not by the DMA. For quiet request please refer to the HTU users guide.

The configuration of the NHET Request Destination Select Register (HETREQDS) bits determines if a request line triggers a HTU-DCP, a DMA channel or both. This means the register bits will determine whether a NHET instruction triggers DMAREQ[x], HTUREQ[x] or both signals (shown in Figure 18-29). The request line number x corresponds to the “reqnum” parameter used in the instruction.

Figure 18-29. Request line assignment example

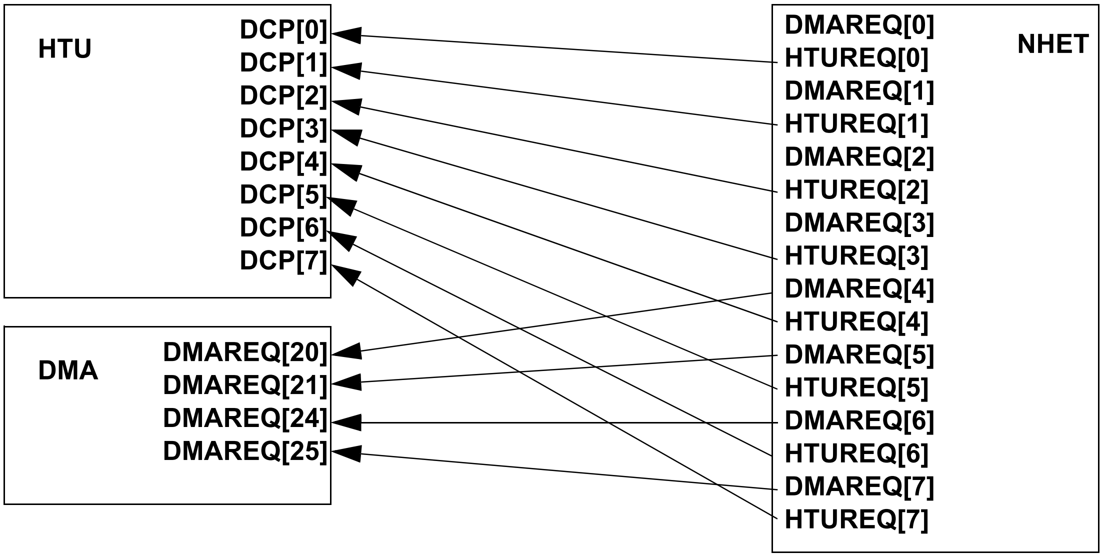


Figure 18-29 shows that not all combinations, which are possible with the NHET Request Destination Select Register (HETREQDS) bits, must be supported by a given device configuration.

18.5 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The NHET has a method to provide such a time base for low-end engine systems. The reference is created by the NHET using three dedicated instructions with fractional angle steps equal to /8, / 16, /32, /64.

18.5.1 Software Angle Generator

The NHET 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.

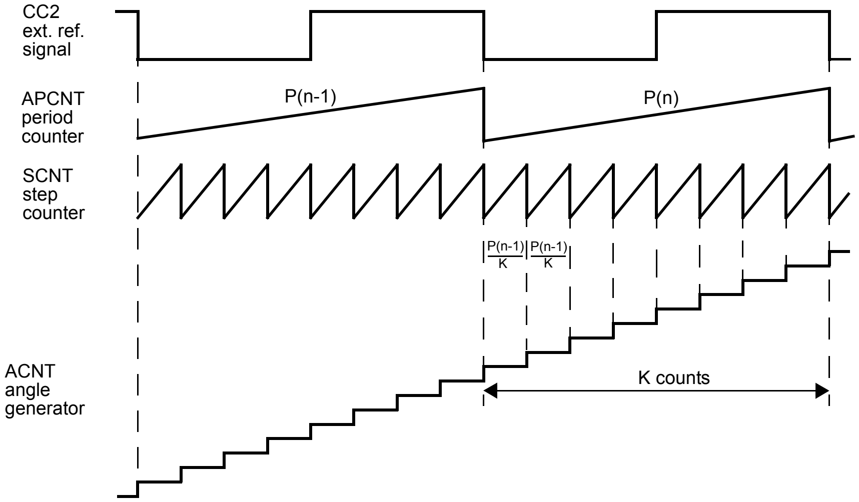
The time base is used to generate fractional angle steps between the reference points. The step width K (= 8, 16, 32, or 64) programmed by the user defines the angle accuracy of the time base. These fractional steps are then accumulated in an angle counter to form the absolute angle value.

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 18-30 illustrates the basic operation of APCNT, SCNT, and ACNT.

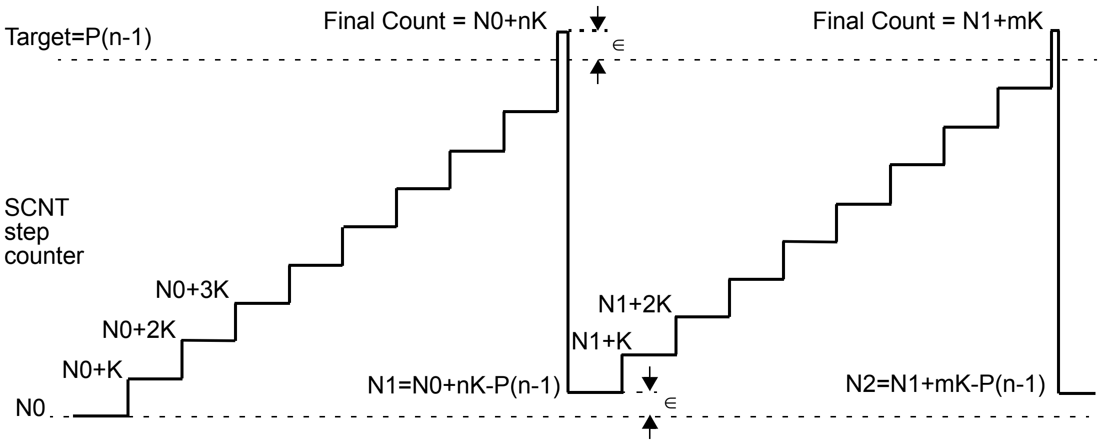
A NHET timer program can only have one angle generator.

Figure 18-30. Operation of NHET Count Instructions



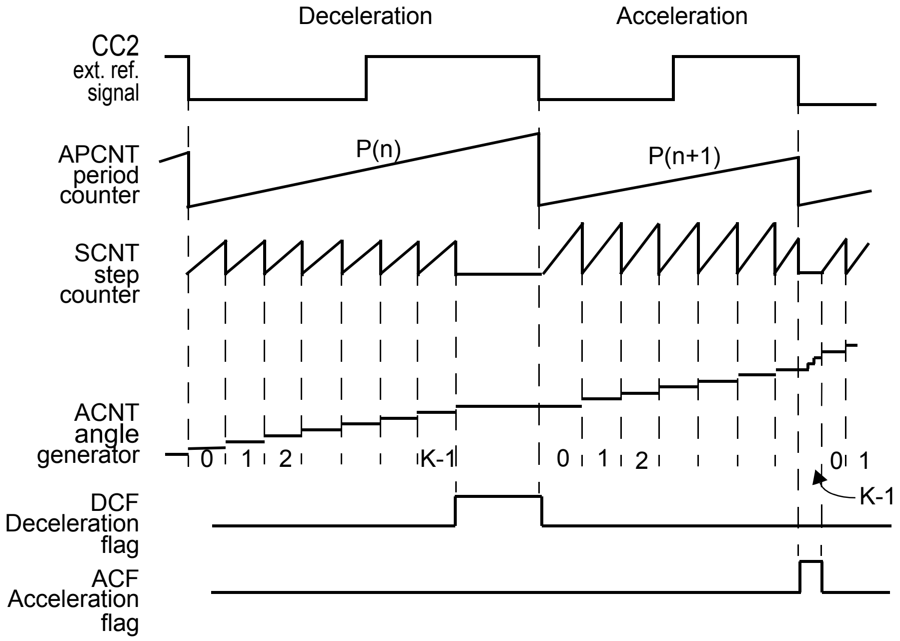
Due to stepping, the final count of SCNT does not usually exactly match the target value P(n-1). Figure 18-31 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

Figure 18-31. 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 18-32 illustrates how the compensations for acceleration and deceleration operate.

Figure 18-32. ACNT Period Variation Compensations



18.5.1.1 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 18-33 and Figure 18-34 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the NHET.

Figure 18-33. NHET Timings Associated with the Gap Flag (ACNT Deceleration)7

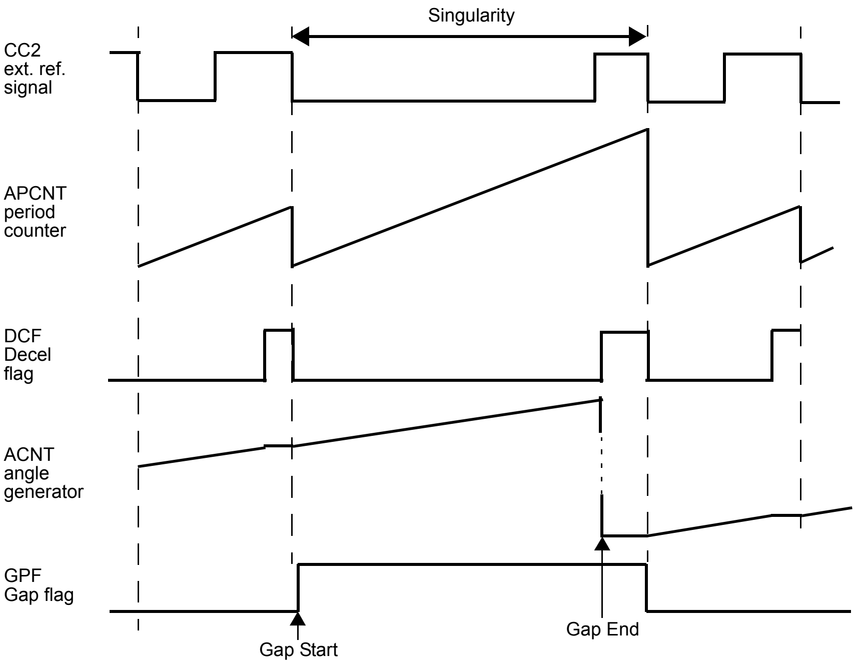
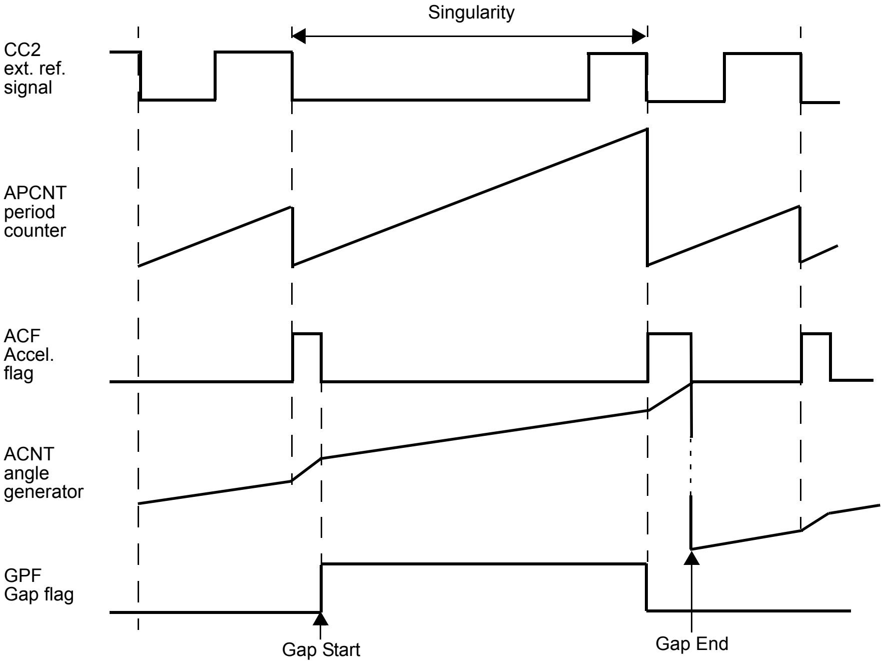


Figure 18-34. NHET Timings Associated with the Gap Flag (ACNT Acceleration)



18.5.1.2 APCNT Underflow

The fastest valid external signal APCNT can accept must satisfy the following condition:

$$\text{Step Width K} < \frac{\text{Period Min.}}{\text{Resolution (LRP)}}$$

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.

18.5.1.3 APCNT Overflow

The slowest valid external signal APCNT can measure must satisfy the following condition:

$$\frac{\text{Period Max}}{\text{Resolution}} < 33554431$$

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.

18.6 NHET Control Registers

18.6.1 NHET Register Summary

Table 18-35 summarizes all the NHET registers. The base address for the control registers is 0xFFF7 B800.

Figure 18-35. NHET Registers

<table>
<tr><th>Offset Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x00
HETGCR
Page 1282</td><td colspan="7">Reserved</td><td>HET PIN ENA</td><td>Reserved</td><td colspan="2">MP</td><td colspan="2">Reserved</td><td>PPF</td><td>IS</td><td>CMS</td></tr>
<tr><td colspan="15">Reserved</td><td>TO</td></tr>
<tr><td rowspan="2">0x04
HETPFR
Page 1285</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="5">Reserved</td><td colspan="3">LRPFC</td><td colspan="2">Reserved</td><td colspan="6">HRPFC</td></tr>
<tr><td rowspan="2">0x08
HETADDR
Page 1287</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">HETADDR.8:0</td></tr>
<tr><td rowspan="2">0x0C
HETOFF1
Page 1288</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="10">Reserved</td><td colspan="6">OFFSET1</td></tr>
<tr><td rowspan="2">0x10
HETOFF2
Page 1289</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="10">Reserved</td><td colspan="6">OFFSET2</td></tr>
<tr><td rowspan="2">0x14
HET
INTENAS
Page 1290</td><td colspan="16">HETINTENAS[31:16]</td></tr>
<tr><td colspan="16">HETINTENAS[15:0]</td></tr>
<tr><td rowspan="2">0x18
HET
INTENAC
Page 1291</td><td colspan="16">HETINTENAC[31:16]</td></tr>
<tr><td colspan="16">HETINTENAC[15:0]</td></tr>
</table>

Figure 18-35. NHET Registers (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C HETEXC1 Page 1292	Reserved							APCNT OVRFL ENA	Reserved							APCNT UNRFL ENA
	Reserved							PRGM OVRFL ENA	Reserved					APCNT OVRFL PRY	APCNT UNRFL PRY	PRGM OVRFL PRY
0x20 HETEXC2 Page 1293	Reserved															
	Reserved							Debug Status Flag	Reserved					APCNT OVRFL FLAG	APCNT UNRFL FLAG	PRGM OVRFL FLAG
0x24 HETPRY Page 1295	HETPRY.31:16															
	HETPRY.15:0															
0x28 HETFLG Page 1296	HETFLAG.31:16															
	HETFLAG.15:0															
0x34 HETHRSH Page 1297	Reserved															
	HRSHARE 31/30	HRSHARE 29/28	HRSHARE 27/26	HRSHARE 25/24	HRSHARE 23/22	HRSHARE 21/20	HRSHARE 19/18	HRSHARE 17/16	HRSHARE 15/14	HRSHARE 13/12	HRSHARE 11/10	HRSHARE 9/8	HRSHARE 7/6	HRSHARE 5/4	HRSHARE 3/2	HRSHARE 1/0
0x38 HETXOR Page 1298	Reserved															
	XORSHARE 31/30	XORSHARE 29/28	XORSHARE 27/26	XORSHARE 25/24	XORSHARE 23/22	XORSHARE 21/20	XORSHARE 19/18	XORSHARE 17/16	XORSHARE 15/14	XORSHARE 13/12	XORSHARE 11/10	XORSHARE 9/8	XORSHARE 7/6	XORSHARE 5/4	XORSHARE 3/2	XORSHARE 1/0
0x3C HETREQENS Page 1299	Reserved															
	Reserved								REQENA 7	REQENA 6	REQENA 5	REQENA 4	REQENA 3	REQENA 2	REQENA 1	REQENA 0

Figure 18-35. NHET Registers (Continued)

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
0x40 HETREQENC Page 1300	Reserved															
	Reserved								REQ DIS 7	REQ DIS 6	REQ DIS 5	REQ DIS 4	REQ DIS 3	REQ DIS 2	REQ DIS 1	REQ DIS 0
0x44 HETREQDS Page 1301	Reserved								TDBS 7	TDBS 6	TDBS 5	TDBS 4	TDBS 3	TDBS 2	TDBS 1	TDBS 0
	Reserved								TDS 7	TDS 6	TDS 5	TDS 4	TDS 3	TDS 2	TDS 1	TDS 0
0x4C HETDIR Page 1302	HETDIR.31:16															
	HETDIR.15:0															
0x50 HETDIN Page 1303	HETDIN.31:16															
	HETDIN.15:0															
0x54 HETDOUT Page 1304	HETDOUT.31:16															
	HETDOUT.15:0															
0x58 HETDSET Page 1305	HETDSET.31:16															
	HETDSET.15:0															
0x5C HETDCLR Page 1306	HETDCLR.31:16															
	HETDCLR.15:0															
0x60 HETPDR Page 1307	HETPDR.31:16															
	HETPDR.15:0															

Figure 18-35. NHET Registers (Continued)

<table>
<tr><th>Offset Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x64
HET
PULDIS
Page 1308</td><td colspan="16">HETPULDIS.31:16</td></tr>
<tr><td colspan="16">HETPULDIS.15:0</td></tr>
<tr><td rowspan="2">0x68
HETPSL
Page 1309</td><td colspan="16">HETPSL.31:16</td></tr>
<tr><td colspan="16">HETPSL.15:0</td></tr>
<tr><td rowspan="2">0x74
HETPCR
Page 1310</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="7">Reserved</td><td>TEST</td><td colspan="4">Reserved</td><td colspan="4">PARITY_ENA</td></tr>
<tr><td rowspan="2">0x78
HETPAR
Page 1311</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="11">PAOFF</td><td>0</td><td>0</td></tr>
<tr><td rowspan="2">0x7C
HETPPR
Page 1312</td><td colspan="16">HETPPR[31:16]</td></tr>
<tr><td colspan="16">HETPPR[15:0]</td></tr>
<tr><td rowspan="2">0x80
HETSFPRLD
Page 1313</td><td colspan="14">Reserved</td><td colspan="2">CCDIV</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">CPRLD</td></tr>
<tr><td rowspan="2">0x84
HETSFENA
Page 1314</td><td colspan="16">HETSFENA.31:16</td></tr>
<tr><td colspan="16">HETSFENA.15:0</td></tr>
<tr><td rowspan="2">0x84
HETSFENA
Page 1314</td><td colspan="16">HETSFENA.31:16</td></tr>
<tr><td colspan="16">HETSFENA.15:0</td></tr>
</table>

Figure 18-35. NHET Registers (Continued)

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
0x8C HETLBPSEL Page 1315	LBP TYPE 31/30	LBP TYPE 29/28	LBP TYPE 27/26	LBP TYPE 25/24	LBP TYPE 23/22	LBP TYPE 21/20	LBP TYPE 19/18	LBP TYPE 17/16	LBP TYPE 15/14	LBP TYPE 13/12	LBP TYPE 11/10	LBP TYPE 9/8	LBP TYPE 7/6	LBP TYPE 5/4	LBP TYPE 3/2	LBP TYPE 1/0
	LBP SEL 31/30	LBP SEL 29/28	LBP SEL 27/26	LBP SEL 25/24	LBP SEL 23/22	LBP SEL 21/20	LBP SEL 19/18	LBP SEL 17/16	LBP SEL 15/14	LBP SEL 13/12	LBP SEL 11/10	LBP SEL 9/8	LBP SEL 7/6	LBP SEL 5/4	LBP SEL 3/2	LBP SEL 1/0
0x90 HET LBPDIR Page 1316	Reserved							Reserved					IODFTENA			
	LBP DIR 31/30	LBP DIR 29/28	LBP DIR 27/26	LBP DIR 25/24	LBP DIR 23/22	LBP DIR 21/20	LBP DIR 19/18	LBP DIR 17/16	LBP DIR 15/14	LBP DIR 13/12	LBP DIR 11/10	LBP DIR 9/8	LBP DIR 7/6	LBP DIR 5/4	LBP DIR 3/2	LBP DIR 1/0
0x94 HET PINDIS Page 1317	HETPINDIS[31:16]															
	HETPINDIS[15:0]															

18.6.2 Global Configuration Register (HETGCR)

Figure 18-36. Global Configuration Register (HETGCR) [0xFFF7B800]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							HET PIN ENA	Reserved	MP		Reserved		PPF	IS	CMS
R-0							RW-1	R-0	RW-0		R-0		RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															TO
R-0															RW-0

RW = Read/Write, RC = Read/Clear, -*n* = value after reset

.

Table 18-11. *Global Configuration Register (HETGCR)*

Bit	Name	Value	Description
31-25	Reserved		Reads return zero, 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		Reads return zero, writes have no effect.
22-21	MP		Master Priority The NHET can prioritize master accesses to NHET 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:
		00	The HTU has lower priority to access the NHET RAM than the arbiter output.
		01	The HTU has higher priority to access the NHET RAM than the arbiter output.
		10	The HTU and the arbiter output use a round robin scheme to access the NHET RAM.
		11	reserved
20-19	Reserved		Reads return zero, writes have no effect.

Table 18-11. *Global Configuration Register (HETGCR)* (Continued)

Bit	Name	Value	Description
18	PPF		Protect Program Fields
			The PPF bit together with the Turn On/Off bit (TO) allows to protect the program fields of all instructions in NHET RAM.
		0 (TO = 0)	All masters can read and write the program fields.
		0 (TO = 1)	All masters can read and write the program fields.
		1 (TO = 0)	All masters can read and write the program fields.
		1 (TO = 1)	The program fields are readable but not writable for all masters, which could access the NHET RAM. Possible masters are the CPU, HTU, DMA and a secondary CPU (if available). When one of these masters writes to the program field of a instruction, it will remain unchanged.
17	IS		Ignore Suspend
			When Ignore Suspend = 0, the timer operation is stopped on suspend (the current timer instruction is completed). Timer RAM can be freely accessed during suspend. When set to 1, the suspend is ignored and the NHET continues operating.
		0	NHET stops when in suspend mode.
		1	NHET ignores suspend mode and continues operation.
16	CMS		Clk_master/slave
			This bit is used to synchronize multi-NHETs. If set (NHET is master), the NHET outputs a signal to synchronize the prescalers of the slave NHET. By default, this bit is reset, which means a slave configuration.
			Note: This bit must be set to one (1) for single-NHET configuration.
		0	NHET is configured as a slave.
		1	NHET is configured as a master.
15-1	Reserved		Reads return zero, writes have no effect.

Table 18-11. *Global Configuration Register (HETGCR)* (Continued)

Bit	Name	Value	Description
0	TO		Turn On/Off When TO = 0, 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. 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. When TO = 1, timer program execution starts synchronously to the Loop clock. In case of multiple NHETs configuration, the slave NHETs 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).
		0	NHET is OFF.
		1	NHET is ON.

18.6.3 Prescale Factor Register (HETPFR)

Figure 18-37. Prescale Factor Register (HETPFR) [0xFFF7B804]

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
Reserved					LRPFC			Reserved		HRPFC					
R-0					RWP-0			R-0		RWP-0					

RWP = Read in all modes, write in privilege mode only, -*n* = Value after reset

The number of time slots ts, which are available in one NHET loop for the program execution is the product of the high resolution prescale divide rate hr and the loop resolution prescale divide rate lr: ts = hr · lr

Both divide rates hr and lr must be defined in the HETPFR register.

.

Table 18-12. *Prescale Factor Register (HETPFR)*

Bit	Name	Value	Description
31-11	Reserved		Reads return zero, writes have no effect.
10-8	LRPFC		Loop Resolution Pre-scale Factor Code The binary code programmed LRPFC determines the loop resolution prescale divide rate (lr) according to Table 18-13. **Note:** LRPFC is readable in all modes, but writable in privileged mode only.
7-6	Reserved		Reads return zero, writes have no effect.
5-0	HRPFC		High Resolution Pre-scale Factor Code The binary code programmed HRPFC determines the high resolution prescale divide rate (hr) according to Table 18-14. **Note:** HRPFC is readable in all modes, but writable in privileged mode only.

Table 18-13. Loop Resolution Encoding Format

Loop Resolution Prescale Factor Code			Loop-Res Prescale Divide Rate lr
2	1	0	
0	0	0	1
0	0	1	2
0	1	0	4

Table 18-13. Loop Resolution Encoding Format (Continued)

Loop Resolution Prescale Factor Code			Loop-Res Prescale Divide Rate lr
2	1	0	
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Table 18-14. HR Encoding Format

HR Prescale Factor code						HR Prescale Divide Rate hr
5	4	3	2	1	0	
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
0	0	0	1	0	0	5
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	0	1	62
1	1	1	1	1	0	63
1	1	1	1	1	1	64

18.6.4 NHET Current Address Register (HETADDR)

Figure 18-38. NHET Current Address (HETADDR) [0xFFF7B808]

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
Reserved							HETADDR[8:0]								
R-0							R-0								

R = Read, -*n* = Value after reset

.

Table 18-15. *NHET Current Address (HETADDR)*

Bit	Name	Value	Description
31-9	Reserved		Reads return zero, writes have no effect.
8-0	HETADDR[8:0]		NHET Current Address Read: Returns the current NHET program address. Write: Writes have no effect.

18.6.5 *Offset Index Priority Level 1 Register (HETOFF1)*

Figure 18-39. ***Offset Index Priority Level 1 Register (HETOFF1)*** **[0xFFF7B80C]**

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
Reserved										Offset1					
R-0										R-0					

R = Read, -*n* = Value after reset

.

Table 18-16. ***Offset Index Priority Level 1 Register (HETOFF1)***

Bit	Name	Value	Description
31-6	Reserved		Reads return zero, writes have no effect.
5-0	Offset1		HETOFF1[5:0] indexes the currently pending high-priority interrupt. Offset values and sources are listed below and the interrupt encoding format is presented in Table 18-17. Read: Read of these bits determines the pending NHET interrupt. Write: Writes have no effect. **Note:** In any read operation mode, the corresponding flag (in the Interrupt Flag Register (HETFLG)) is also cleared. In Emulation mode the corresponding flag is not cleared.

Table 18-17. Interrupt Offset Encoding Format

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

18.6.6 Offset Index Priority Level 2 Register (HETOFF2)

Figure 18-40. Offset Index Priority Level 2 Register (HETOFF2) [0xFFF7B810]

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
Reserved										Offset2					
R-0										R-0					

R = Read, *-n* = Value after reset

.

Table 18-18. *Offset Index Priority Level 2 Register (HETOFF2)*

Bit	Name	Value	Description
31-6	Reserved		Reads return zero, writes have no effect.
5-0	Offset2		HETOFF2[5:0] indexes the currently pending low-priority interrupt. Offset values and sources are listed below and the interrupt encoding format is presented in Table 18-17. Write: Writes have no effect. Read: Read of these bits determines the pending NHET interrupt. **Note:** In any read operation mode, the corresponding flag (in the Interrupt Flag Register (HETFLG)) is also cleared. In Emulation mode the corresponding flag is not cleared.

18.6.7 Interrupt Enable Set Register (HETINTENAS)

Figure 18-41. NHET Interrupt Enable Set (HETINTENAS) [0xFFF7B814]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETINTENAS[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETINTENAS[15:0]															
RW-0															

RW = Read/Write, -*n* = value after reset

.

Table 18-19. Interrupt Enable Set (HETINTENAS)

Bit	Name	Value	Description
31-0	HETINTENAS		Interrupt Enable Set bits HETINTENAS is readable and writable in any operation mode. Writing a one to bit x enables the interrupts of the NHET instructions at NHET 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... To avoid ambiguity only one of the instructions x+0, x+32, x+64... should have the interrupt enable bit (inside the instruction) set. Writing a zero to HETINTENAS has no effect. When reading from HETINTENAS bit x gives the information, if NHET instructions x+0, x+32, x+64... have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: Writes have no effect.
		1	Read: Interrupt is enabled. Write: Interrupt will be enabled.

18.6.8 Interrupt Enable Clear Register (HETINTENAC)

Figure 18-42. NHET Interrupt Enable Clear (HETINTENAC) [0xFFF7B818]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETINTENAC[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETINTENAC[15:0]															
RW-0															

RW = Read/Write, *-n* = value after reset

.

Table 18-20. NHET Interrupt Enable Clear (HETINTENAC)

Bit	Name	Value	Description
31-0	HETINTENAC		Interrupt Enable Clear bits HETINTENAC is readable and writable in any operation mode. Writing a one to bit x disables the interrupts of the NHET instructions at NHET addresses x+0, x+32, x+64... (See also description for Interrupt Enable Set Register (HETINTENAS)) Writing a zero to HETINTENAC has no effect. When reading from HETINTENAC bit x gives the information, if NHET instructions x+0, x+32, x+64... have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: Writes have no effect.
		1	Read: Interrupt is enabled. Write: Interrupt will be enabled.

18.6.9 Exception Control Register 1 (HETEXC1)

Figure 18-43. Exception Control Register (HETEXC1) [0xFFF7B81C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							Prgm Ovrfl Ena	Reserved					APCNT Ovrfl Pry	APCNT Undrfl Pry	Prgm Ovrfl Pry
R-0							RW-0	R-0					RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

.

Table 18-21. *Exception Control Register 1 (HETEXC1)*

Bit	Name	Value	Description
31-25	Reserved		Reads return zero, writes have no effect.
24	APCNT Ovrfl Ena		APCNT Overflow Enable
		0	APCNT overflow exception is not enabled.
		1	Enables the APCNT overflow exception.
15-9	Reserved		Reads return zero, 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		Reads return zero, writes have no effect.
2-0	APCNT Ovrfl Pry APCNT Undrfl Pry Prgm Ovrfl Pry		Exception Priority Level Register bits Used to select the priority of any of the three potential exception sources.
		0	Exception priority level 2
		1	Exception priority level 1

18.6.10 Exception Control Register 2 (HETEXC2)

Figure 18-44. Exception Control Register 2 (HETEXC2) [0xFFF7B820]

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
Reserved							Debug Status Flag	Reserved					APCNT Ovrfl flg	APCNT Undfl flg	Prgm Ovrfl flg
R-0							RC-0	R-0					RC-0	RC-0	RC-0

RC = Read/Clear, -*n* = Value after reset

.

Table 18-22. *Exception Control Register 2 (HETEXC2)*

Bit	Name	Value	Description
31-9	Reserved		Reads return zero, writes have no effect.
8	Debug Status Flag		Debug Status Flag This flag is set when the device test mode is set and a breakpoint is reached in the NHET program. A debug request signal is asserted to the main CPU.
		0	Read: No NHET instruction with a breakpoint has been reached since the flag was last cleared. Write: No effect.
		1	Read: A NHET instruction with a breakpoint has been reached since the flag was last cleared. Write: Clears the bit.
7-3	Reserved		Reads return zero, writes have no effect.
2	APCNT Ovrfl Flg		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 Undrfl flg		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.

Table 18-22. *Exception Control Register 2 (HETEXC2)* (Continued)

Bit	Name	Value	Description
0	Prgm Overfl flg		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.

18.6.11 Interrupt Priority Register (HETPRY)

Figure 18-45. Interrupt Priority Register (HETPRY) [0xFFF7B824]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPRY[31:16]															
RWP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETPRY[15:0]															
RWP-0															

RWP = Read in all modes, write in privilege mode only, -*n* = Value after reset

.

Table 18-23. Interrupt Priority Register (HETPRY)

Bit	Name	Value	Description
31-0	HETPRY		HET Priority Level bits Used to select the priority of any of the 32 potential interrupt sources coming from the instructions.
		0	Exception priority level 2.
		1	Exception priority level 1.

18.6.12 Interrupt Flag Register (HETFLG)

Figure 18-46. Interrupt Flag Register (HETFLG) [0xFFF7B828]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETFLAG [31:16]															
RC-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETFLAG [15:0]															
RC-0															

RC = Read/Clear, -*n* = Value after reset

.

Table 18-24. Interrupt Flag Register (HETFLG)

Bit	Name	Value	Description
31-0	HETFLAG[31:0]		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.... 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 one 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 NHET instruction with an interrupt has been reached since the flag was cleared. Write: No effect.
		1	Read: A NHET instruction with an interrupt has been reached since the flag was cleared. Write: Clears the bit.

18.6.13 HR Share Control Register (HETHRSH)

Figure 18-47. HR Share Control Register (HETHRSH) [0xFFF7B834]

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
HR Share 31/30	HR Share 29/28	HR Share 27/26	HR Share 25/24	HR Share 23/22	HR Share 21/20	HR Share 19/18	HR Share 17/16	HR Share 15/14	HR Share 13/12	HR Share 11/10	HR Share 9/8	HR Share 7/6	HR Share 5/4	HR Share 3/2	HR Share 1/0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

Table 18-25. HR Share Control Register (HETHRSH)

Bit	Name	Value	Description
31-16	Reserved		Reads return zero, writes have no effect.
15-0	HR Share[15:0]		HR Share Bits Enables the share of the same pin for two HR structures. For example, if bit HR share 1/0 is set, the pin NHET[0] will then be connected to both HR structures 0 and 1. **Note:** If HR share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs.
		0	Pins are not shared.
		1	Pins are shared.

18.6.14 HR XOR-share Control Register (HETXOR)

Figure 18-48. HR XOR-Share Control Register (HETXOR) [0xFFF7B838]

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
HR XOR Share 31/30	HR XOR Share 29/28	HR XOR Share 27/26	HR XOR Share 25/24	HR XOR Share 23/22	HR XOR Share 21/20	HR XOR Share 19/18	HR XOR Share 17/16	HR XOR Share 15/14	HR XOR Share 13/12	HR XOR Share 11/10	HR XOR Share 9/8	HR XOR Share 7/6	HR XOR Share 5/4	HR XOR Share 3/2	HR XOR Share 1/0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

Table 18-26. HR XOR-share Control Register (HETXOR)

Bit	Name	Value	Description
31-16	Reserved		Reads return zero, writes have no effect.
15-0	HR XOR-Share[15:0]		HR XOR-Share Bits Enable the XOR-share of the same pin for two HR structures. For example, if bit HR XOR-share 1/0 is set, the pin NHET[0] will then be commanded by a logical XOR of both HR structures 0 and 1. **Note:** If HR XOR-share bits are used, pins not connected to HR structures can be accessed as general inputs/outputs.
		0	Pins are not XOR-shared.
		1	Pins are XOR-shared.

18.6.15 Request Enable Set Register (HETREQENS)

Figure 18-49. Request Enable Set Register (HETREQENS) [0xFFF7B83C]

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
Reserved								REQ ENA 7	REQ ENA 6	REQ ENA 5	REQ ENA 4	REQ ENA 3	REQ ENA 2	REQ ENA 1	REQ ENA 0
R-0								RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, *-n* = Value after reset

Table 18-27. Request Enable Set Register (HETREQENS)

Bit	Name	Value	Description
31-8	Reserved		Reads return zero, writes have no effect.
7-0	REQENA[7:0]		Request Enable Bits
		0	Read: Returns the information that request line x is disabled. Write: Writing a 0 has no effect.
		1	Read: Returns the information that request line x is enabled. Write: Writing a 1 to bit x enables the NHET request line x, which must be enabled in (only) one NHET instruction. **Note:** The request line can trigger a DMA control packet (i.e. DMA channel), a HTU double control packet (DCP) or both simultaneously. The HETREQDS register determines to which module(s) the NHET request line x is assigned. **Note:** A disabled request line does not memorize old requests. So there are no pending requests to service after enabling request line x.

18.6.16 Request Enable Clear Register (HETREQENC)

Figure 18-50. Request Enable Clear Register (HETREQENC) [0xFFF7B840]

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
Reserved								REQ DIS 7	REQ DIS 6	REQ DIS 5	REQ DIS 4	REQ DIS 3	REQ DIS 2	REQ DIS 1	REQ DIS 0
R-0								RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

Table 18-28. Request Enable Clear Register (HETREQENC)

Bit	Name	Value	Description
31-8	Reserved		Reads return zero, writes have no effect.
7-0	REQDIS [7:0]		Request Disable Bits
		0	Read: Returns the information that request line x is disabled. Write: Writing a 0 has no effect.
		1	Read: Returns the information that request line x is enabled. Write: Writing a 1 to bit x disables the NHET request line x, which must be enabled in (only) one NHET instruction.

18.6.17 Request Destination Select Register (HETREQDS)

Figure 18-51. Request Destination Select Register (HETREQDS) [0xFFF7B844]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TDBS 7	TDBS 6	TDBS 5	TDBS 4	TDBS 3	TDBS 2	TDBS 1	TDBS 0
R-0								RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TDS 7	TDS 6	TDS 5	TDS 4	TDS 3	TDS 2	TDS 1	TDS 0
R-0								RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

Table 18-29. Request Destination Select Register (HETREQDS)

Bit	Name	Value	Description
31-24	Reserved		Reads return zero, writes have no effect.
23-16	TDBS [7:0]		HTU, DMA or Both Select Bits
		0	NHET request line x is assigned to the module specified by TDS bit x.
		1	NHET request line x is assigned to both DMA and HTU. TDS bit x is ignored in this case.
15-8	Reserved		Reads return zero, writes have no effect.
7-0	TDS [7:0]		HTU or DMA Select Bits **Note:** It must be ensured in the NHET program, that one request line is triggered by only one NHET instruction.
		0	NHET request line x is assigned to HTU (TDBS bit x is zero).
		1	NHET request line x is assigned to DMA (TDBS bit x is zero).

Note:
Please refer to the device data sheet how each of the 8 NHET request lines are connected to these modules. See also Section 18.4.8.

18.6.18 NHET Direction Register (HETDIR)

Figure 18-52. NHET Direction Register (HETDIR) [0xFFF7B84C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">HETDIR[31:16]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">HETDIR[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

RW = Read/Write, -*n* = Value after reset

.

Table 18-30. NHET Direction Register (HETDIR)

Bit	Name	Value	Description
31-0	HET DIR[31:0]		Data direction of NHET pins
		0	The pin is an input **Note: If the pin direction is set as an input, the output buffer is tristated.**
		1	The pin is an output

Note:
Table 18-7 shows how the register bits of DIR, PULDIS and PULSEL are affecting the NHET pins.

18.6.19 NHET Data Input Register (HETDIN)

Figure 18-53. NHET Data Input Register (HETDIN) [0xFFF7B850]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">HETDIN[31:16]</td></tr>
<tr><td colspan="16">R-x</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">HETDIN[15:0]</td></tr>
<tr><td colspan="16">R-x</td></tr>
</table>

R = Read, -*n* = Value after reset

Table 18-31. NHET Data Input Register (HETDIN)

Bit	Name	Value	Description
31-0	HETDIN[31:0]		Data input. This bit displays the logic state of the pin.
		0	The pin is at logic low (0)
		1	The pin is at logic high (1)

18.6.20 NHET Data Output Register (HETDOUT)

Figure 18-54. NHET Data Output Register (HETDOUT) [0xFFF7B854]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDOUT[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDOUT[15:0]															
RW-0															

RW = Read/Write, -*n* = Value after reset

Table 18-32. NHET Data Output Register (R-Write) (HETDOUT)

Bit	Name	Value	Description
31-0	HETDOUT[31:0]		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	The pin is at logic low (0).
		1	The pin is at logic high (1) if the HETPDRx bit = 0 or the output is in high impedance state if the HETPDRx bit = 1

18.6.21 NHET Data Set Register (HETDSET)

Figure 18-55. NHET Data Set Register (HETDSET) [0xFFF7B858]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDSET[31:16]															
RS-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDSET[15:0]															
RS-0															

RS = Read/Set, -*n* = Value after reset

Table 18-33. NHET Data Set Register (R-Set) (HETDSET)

Bit	Name	Value	Description
31-0	HETDSET[31:0]		Data set for NHET pins. This bit drives the output of the pin high.
		0	*Write:* Writing a zero has no effect.
		1	*Write:* The pin is driven to logic high (1)
			Note: The current logic state of the HETDOUT bit will also be displayed by this bit.

18.6.22 NHET Data Clear Register (HETDCLR)

Figure 18-56. NHET Data Clear Register (HETDCLR) [0xFFF7B85C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETDCLR[31:16]															
RC-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETDCLR[15:0]															
RC-0															

RC = Read/Clear, -*n* = Value after reset

Table 18-34. NHET Data Clear Register (R-Clear) (HETDCLR)

Bit	Name	Value	Description
31-0	HETDCLR[31:0]		Data clear for NHET pins. This bit drives the output of the pin low.
		0	*Write:* Writing a zero has no effect.
		1	*Write:* The pin is driven to logic low (0)
			Note: The current logic state of the HETDOUT bit will also be displayed by this bit.

18.6.23 NHET Open Drain Register (HETPDR)

Values in this register enable or disable the open drain capability of the data pins.

Figure 18-57. NHET Open Drain Register (HETPDR) [0xFFF7B860]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPDR[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETPDR[15:0]															
RW-0															

RW = Read/Write, -*n* = Value after reset

Table 18-35. NHET Open Drain Register (HETPDR)

Bit	Name	Value	Description
31-0	HETPDR[31:0]		Open drain for NHET 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: HETDOUTx = 0 The output buffer is driven low HETDOUTx = 1 The output buffer is tristated

18.6.24 NHET Pull Disable Register (HETPULDIS)

Values in this register enable or disable the pull-up/-down functionality of the pins.

Figure 18-58. NHET Pull Disable Register (HETPULDIS) [0xFFF7B864]

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*
HETPULDIS[31:16]															
RW-n															

15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
HETPULDIS[15:0]															
RW-n															

RW = Read/Write, *-n* = Value after reset, n is device dependent, see device specific data sheet

Table 18-36. NHET Pull Disable Register (HETPULDIS)

Bit	Name	Value	Description
31-0	HETPULDIS[31:0]		Pull disable for NHET pins
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.

Note:
See device data sheet which pins provide programmable pullups/pulldowns

Note:
Table 18-7 shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the NHET pins.

18.6.25 NHET Pull Select Register (HETPSL)

Values in this register select the pull-up or pull-down functionality of the pins.

Figure 18-59. NHET Pull Select Register (HETPSL) [0xFFF7B868]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPSL[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETPSL[15:0]															
RW-0															

RW = Read/Write, -*n* = Value after reset

Table 18-37. NHET Pull Select Register (HETPSL)

Bit	Name	Value	Description
31-0	HETPSL[31:0]		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 sheet which pins provide programmable pullups/pulldowns

Note:
Table 18-7 shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the NHET pins.

Note:

The information of this register is also used to define the pin states after a parity error:

After a parity error all NHET 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.

18.6.26 Parity Control Register (HETPCR)

Figure 18-60. Parity Control Register (HETPCR) [0xFFF7B874]

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
Reserved							TEST	Reserved				PARITY_ENA			
R-0							RWP-0	R-0				RWP-0101			

RWP = Read in all modes, write in privilege mode only, -*n* = Value after reset

Table 18-38. Parity Control Register (HETPCR)

Bit	Name	Value	Description
31-9	Reserved		Reads return zero, 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		Reads return zero, 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 NHET_UERR signal is activated.
		0101	Read: Parity check is disabled. Write: Disable checking.
		Others	Read: Parity check is enabled. Write: Enable checking.

Note:

It is recommended to write a ‘1010’ to enable error detection, to guard against soft errors flipping PARITY_ENA to a disable state.

18.6.27 Parity Address Register (HETPAR)

Figure 18-61. Parity Address Register (HETPAR) [0xFFF7B878]

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
Reserved			PAOFF											0	0
R-0			R-X											R-0	

R = Read, -*n* = Value after reset (X = Value unchanged after reset)

Table 18-39. Parity Address Register (HETPAR)

Bit	Name	Value	Description
31-13	Reserved		Reads return zero, 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 NHET 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 NHET RAM parity error, PAOFF will contain the offset address of the erroneous 32-bit NHET RAM field counted from the beginning of the NHET 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 NHET RAM. Write: Writes have no effect.
1-0			Reads return zero, writes have no effect.

Note:
The Parity Error Address Register will not be reset, neither by PORRST nor by any other reset source.

18.6.28 Parity Pin Register (HETPPR)

Figure 18-62. Parity Pin Register (HETPPR) [0xFFF7B87C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPPR[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETPPR[15:0]															
RW-0															

RW = Read/Write, *-n* = Value after reset

Table 18-40. Parity Pin Register (HETPPR)

Bit	Name	Value	Description
31-0	HETPPR[31:0]		NHET Parity Pin Select Bits If bit x is set to one and a parity error is detected, then NHET pin x will be affected in the following way: • If NHET pin x is defined as output pin in the NHET Direction Register (HETDIR) (and not defined as an open drain pin in the NHET Open Drain Register (HETPDR)), it will remain an output, but it will automatically change its level in the following way: –If the NHET Pull Select Register (HETPSL) specifies bit x as 0, it will switch to low level. –If the NHET Pull Select Register (HETPSL) specifies bit x as 1, it will switch to high level. This behavior is independent on the value, which the NHET Pull Disable Register (HETPULDIS) specifies for pin x. • If NHET pin x is defined as open drain pin in the NHET Open Drain Register (HETPDR), it will switch to high impedance state. If bit x is set to **zero**, a parity error will not affect the pin in the above described way.

18.6.29 Suppression Filter Preload Register (HETSFPRLD)

Figure 18-63. Suppression Filter Preload Register (HETSFPRLD) [0xFFF7B880]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="14">Reserved</td><td colspan="2">CCDIV</td></tr>
<tr><td colspan="14">R-0</td><td colspan="2">RW-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">CPRLD</td></tr>
<tr><td colspan="6">R-0</td><td colspan="10">RW-0</td></tr>
</table>

RW = Read/Write, *-n* = Value after reset

Table 18-41. Suppression Filter Preload Register (HETSFPRLD)

Bit	Name	Value	Description
31-18	Reserved		Reads return zero, writes have no effect.
17-16	CCDIV		Counter Clock Divider CCDIV determines the ratio between the counter clock and VCLK2.
		00	CCLK = VCLK2
		01	CCLK = VCLK2 / 2
		10	CCLK = VCLK2 / 3
		11	CCLK = VCLK2 / 4
15-10	Reserved		Reads return zero, writes have no effect.
9-0	CPRLD		Counter Preload Value CPRLD contains the preload value for the counter clock.

18.6.30 Suppression Filter Enable Register (HETSFENA)

Figure 18-64. Suppression Filter Enable Register (HETSFENA) [0xFFF7B884]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETSFENA[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETSFENA[15:0]															
RW-0															

RW = Read/Write, -*n* = Value after reset

Table 18-42. Suppression Filter Enable Register (HETSFENA)

Bit	Name	Value	Description
31-0	HETSFENA[31:0]		Suppression Filter Enable Bits **Note:** If the pin is configured as an output by the NHET Direction Register (HETDIR), the filter is automatically disabled independent on the bit in HETSFENA.
		0	The suppression filter for NHET pin x is disabled.
		1	The suppression filter for NHET pin x is enabled.

18.6.31 Loop Back Pair Select Register (HETLBPSEL)

Figure 18-65. Loop Back Pair Select Register (HETLBPSEL) [0xFFF7B88C]

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*
LBP TYPE 31/30	LBP TYPE 29/28	LBP TYPE 27/26	LBP TYPE 25/24	LBP TYPE 23/22	LBP TYPE 21/20	LBP TYPE 19/18	LBP TYPE 17/16	LBP TYPE 15/14	LBP TYPE 13/12	LBP TYPE 11/10	LBP TYPE 9/8	LBP TYPE 7/6	LBP TYPE 5/4	LBP TYPE 3/2	LBP TYPE 1/0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
LBP SEL 31/30	LBP SEL 29/28	LBP SEL 27/26	LBP SEL 25/24	LBP SEL 23/22	LBP SEL 21/20	LBP SEL 19/18	LBP SEL 17/16	LBP SEL 15/14	LBP SEL 13/12	LBP SEL 11/10	LBP SEL 9/8	LBP SEL 7/6	LBP SEL 5/4	LBP SEL 3/2	LBP SEL 1/0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RW = Read/Write, -*n* = Value after reset

Table 18-43. Loop Back Pair Select Register (HETLBPSEL)

Bit	Name	Value	Description
31-16	LBPTYPE[15:0]		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 (2.x) and (2.x+1).
		1	Analog loopback is selected for HR structures (2.x) and (2.x+1).
15-0	LBPSEL[15:0]		Loop Back Pair Select Bits These bits are valid only when IODFT mode is enabled. (HETLBPDIR[19:16] = “1010”). If bit x is set, the HR structures (2·x) and (2·x+1) are connected in a loop back mode. The direction is given by LBPDIR[x] and type is selected by LBPTYPE[x]. The pin which is not driven by the NHET pin actions can still be used as normal GIO pin.

18.6.32 Loop Back Pair Direction Register (HETLBPDIR)

Figure 18-66. Loop Back Pair Direction Register (HETLBPDIR) [0xFFF7B890]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												IODFTENA			
R-0												RWP-0101			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LBP DIR 31/30	LBP DIR 29/28	LBP DIR 27/26	LBP DIR 25/24	LBP DIR 23/22	LBP DIR 21/20	LBP DIR 19/18	LBP DIR 17/16	LBP DIR 15/14	LBP DIR 13/12	LBP DIR 11/10	LBP DIR 9/8	LBP DIR 7/6	LBP DIR 5/4	LBP DIR 3/2	LBP DIR 1/0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

RWP = Read in all modes, write in privilege mode only, -*n* = Value after reset

Table 18-44. Loop Back Pair Direction Register (HETLBPDIR)

Bit	Name	Value	Description
31-20	Reserved		Reads return zero, writes have no effect.
19-16	IODFTENA		Module IODFT Enable Key Reset Value is “0101” i.e. I/O DFT is enabled.
		1010	I/O DFT is enabled.
		Others	I/O DFT is disabled.
15-0	LBPDIR[15:0]		Loop Back Pair Direction Bits
		0	The HR structures (2·x) and (2·x+1) are internally connected with (2·x) as input and (2·x+1) as output.
		1	The HR structures (2·x) and (2·x+1) are internally connected with (2·x) as output and (2·x+1) as input.

Note:
The loop back direction can be selected independent on the HETDIR register setting.

18.6.33 NHET Pin Disable Register (HETPINDIS)

Figure 18-67. NHET Pin Disable Register (HETPINDIS) [0xFFF7B894]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETPINDIS[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETPINDIS[15:0]															
RW-0															

RW = Read/Write, *-0* = Value after reset

Table 18-45. NHET Pin Disable Register (HETPINDIS)

Bit	Name	Value	Description
31-0	HETPINDIS[31:0]		NHET Pin Disable Bits
		0	Logic low : No affect on the output buffer enable of the pin (is controlled by the value of the HETDIR.x bit).
		1	Logic high : Output buffer of the pin is enabled if pin nDIS = 1, HET PIN ENA = 1 and HETDIR = 1, or else disabled if nDIS = 0 or HETDIR = 0 or HET PIN ENA = 0.

18.7 Instruction Set

18.7.1 Instruction Summary

Table 18-46 presents a list of the instructions in the NHET instruction set. The pages following describe each instruction in detail.

Table 18-46. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles	Pages
ACMP	Angle Compare	Ch	-	1	1324
ACNT	Angle Count	9h	-	2	1326
ADCNST	Add Constant	5h	-	2	1330
ADM32	Add Move 32	4h	C5=1	1 or 2	1332
APCNT	Angle Period Count	Eh	-	1 or 2	1336
BR	Branch	Dh	-	1	1339
CNT	Count	6h	-	1 or 2	1341
DADM64	Data Add Move 64	2h	-	2	1344
DJZ	Decrement and Jump if -zero	Ah	P[7-6]=10	1	1346
ECMP	Equality Compare	0h	C[6-5]=00	1	1348
ECNT	Event Count	Ah	P[7-6]=01	1	1351
MCMP	Magnitude Compare	0h	C6=1	1	1353
MOV32	Move 32	4h	C5=0	1 or 2	1357
MOV64	Move 64	1h	-	1	1361
PCNT	Period/Pulse Count	7h	-	1	1364
PWCNT	Pulse Width Count	Ah	P[7-6]=11	1	1368
RADM64	Register Add Move 64	3h	-	1	1371
SCMP	Sequence Compare	0h	C[6-5]=01	1	1374
SCNT	Step Count	Ah	P[7-6]=00	3	1377
SHFT	Shift	Fh	-	1	1380
WCAP	Software Capture Word	Bh	-	1	1383
WCAPE	Software Capture Word and Event Count	8h	-	1	1386

Table 18-47. FLAGS Generated by Instruction

	Flag Name	Set / Reset by	Used by
Z	Z flag	CNT, APCNT, PCNT, ACNT, SCNT, SHFT	ACMP, ECMP, SCMP, MCMP, ACNT, BR, SHFT, MOV32
X	X flag	ACMP	SCMP

Table 18-47. FLAGS Generated by Instruction

	Flag Name	Set / Reset by	Used by
SWF 0-1	Step width flags	SCNT	ACNT
NAF	New Angle flag	ACNT	NAF_global
NAF_global	New Angle flag (global)	*HWAG* or NAF	CNT, ECNT, BR, ACMP, ECMP
ACF	Acceleration flag	ACNT	SCNT, ACNT
DCF	Deceleration flag	ACNT	SCNT, ACNT
GPF	Gap flag	ACNT	APCNT, ACNT

The instructions capable of generating SW interrupts are listed in Table 18-48.

Table 18-48. Interrupt Capable Instructions

Interrupt Capable Instructions	Non Interrupt Capable Instructions
ACMP	ADCNST
ECMP	ADM32
SCMP	DADM64
MCMP	MOV32
CNT	MOV64
ECNT	RADM64
ACNT	SCNT
APCNT	
PWCNT	
PCNT	
DJZ	
WCAP	
WCAPE	
SHFT	
BR	

18.7.2 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 either the HTU or the DMA

Default: 0.

Location: Program field [25:23]

request*

Allows to select between no request (NOREQ), request (GENREQ) and quiet request (QUIET). See section 18.4.8 on page 1271.

Default: No request

Location: Control Field [28:27]

Request	C[28]	C[27]	To HTU	To DMA
NOREQ	0	0	no request	no request
	1	0		
GENREQ	0	1	request	request
QUIET	1	1	quiet request	no 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

MSB				LSB	Pin Select
0	0	0	0	0	Selects NHET[0]
0	0	0	0	1	Selects NHET[1]
(each pin may be selected by writing its number in binary)					
1	1	1	1	0	Selects NHET[30]
1	1	1	1	1	Selects NHET[31]

Reg* Register Select: Selects the register for data comparison and storage.
Default: No register (None).
Location: Control field [2:1] except CNT

Register	C[2]	C[1]
A	0	0
B	0	1
T	1	0
None	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. Pulse high occurs when the pin is set on the compare and toggles at the overflow.

Default: Clear.

Location: Control Field [4:3]

Action	C[4]	C[3]
Clear	0	0
Set	1	0
Pulse Low	0	1
Pulse High	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 18.4.5.7, *Edge Detection*). 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 NHET the first time the instruction is executed.

Default: OFF.

Location: Control Field [25]

cntl_val* Available for the DADM64, MOV64, and RADM64, this bits field allows you 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. The field ensures that the sub-opcodes are moved correctly.

Default: ECMP.

Location: Control Field [6:5]

comp_mode	C[6]	C[5]	order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

18.7.3 Instruction Description

The following sections provide information for individual instructions.

Parameters in [] are optional. Please refer to the NHET assembler user guide for the default values when parameters are omitted.

18.7.3.1 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 | T | NONE**}
[irq ={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
}

Opcode Ch, [P12:P9]

Figure 18-68. ACMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	Opcode	Reserved
6	3	1	9	4	9

Figure 18-69. ACMP Control Field (C31:C0)

31 29	28 27	26	25	24 23	22	21 13	12 8	7 5	4	3	2 1	0
Res.	Request type	Control	Cout prv	Res.	En. pin action	Conditional address	Pin select	Res.	Pin action	Res.	Register select	Int. ena.
3	2	1	1	2	1	9	5	3	1	1	2	1

Figure 18-70. ACMP Data Field (D31:D0)

31 7	6 0
Data.	Reserved
25	7

Cycles One

Register modified Selected register (A, B, 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 field value ≤ Selected register value)
      Cout = 0;

else
      Cout = 1;

If (Z == 0 AND (Selected register value - Angle Inc. < Data field
value) 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)
          SW interrupt flag = 1;
      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.

18.7.3.2 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*}**]**
}

Opcode **9h, [P12:P9]**

Figure 18-71. ACNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 1	0
0	Request Number	BRK	Next program address	Opcode	Edge-select.	Reserved	Int. ena.
6	3	1	9	4	1	7	1

Figure 18-72. ACNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Prv.	Gap End
3	2	1	1	25

Figure 18-73. ACNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

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 NHET[2] exclusively**.** The edge select must be the same as the NHET[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 (NHET[2]).

Action	P8	Edge Select
Rising	1	Detects a rising edge of NHET[2]
Falling	0	Detects a falling edge of NHET[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 NHET[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 NHET[2]) AND (data
          field register != 0) AND (ACF == 0) AND (angle increment con-
          dition is true))
            DCF = 1;
          }
      If (specified edge is detected on pin NHET[2])
          {
          DCF = 0;
          If ((target_edge_field != 0) AND (DCF == 0))
            ACF = 1;
          If (GPF == 1)
```

```
            {
            GPF = 0;
            Z = 1;
            If (Interrupt Enable == 1)
               SW interrupt flag = 1;
            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 NHET[2] pin;
}

Jump to next program address;
```

18.7.3.3 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*}**]**
}

Opcode 5h, [P12:P9]

Figure 18-74. ADCNST Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	Opcode	Remote address
6	3	1	9	4	9

Figure 18-75. ADCNST Control Field (C31:C0)

31 29	28 27	26	25	24 0
Reserved.		Control	Res.	Minimum offset
3	2	1	1	25

Figure 18-76. ADCNST Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles Two

Register modified Register T (implicitly)

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.

Table 18-77 and Table 18-78 illustrate the behavior of ADCNST if the remote data field is or is not zero.

Figure 18-77. ADCNST Operation If Remote Data Field[31:7] Is Not Zero

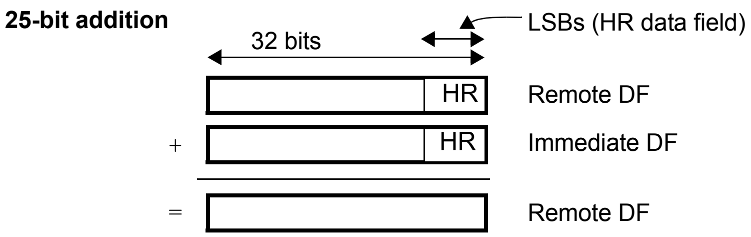
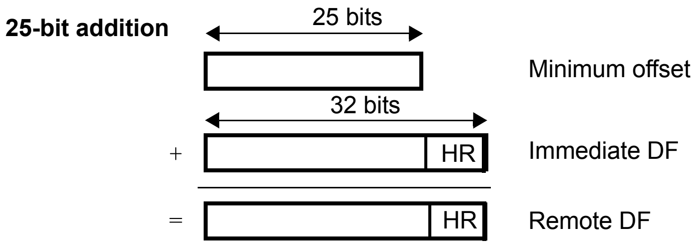


Figure 18-78. ADCNST Operation if Remote Data Field [31:7] Is Zero



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;
```

18.7.3.4 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 | T**}
data={*25-bit unsigned integer*}
[hr_data={*7-bit unsigned integer*}**]**
}

Opcode 4h, [P12:P9];
Sub-Opcode *1h, [C5]*

Figure 18-79. ADM32 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	Opcode	Remote address
6	3	1	9	4	9

Figure 18-80. ADM32 Control Field (C31:C0)

31 27	26	25 7	6	5	4 3	2 1	0
Reserved.	Control	Reserved.	Init flag	Sub-op-code	Move type	Register select	Res.
3 2	1	19	1	1	2	2	1

Figure 18-81. ADM32 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One or two cycles (see Table)

Register modified Selected register (A, B, 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 18-49. Move Types for ADM32

Type	C4	C3	Add		Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field	+ Reg. A, B, or T	Register A, B, or T	1
REM&REGTOREG	0	1	Remote data field	+ Reg. A, B, or T	Register A, B, or T	2
IM&REMTOREG	1	0	Imm. data field	+Remote data field	Register A, B, or T	2
IM&REGTOREM	1	1	Imm. data field	+ Reg. A, B, or T	Remote data field	1

If selected register is 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 18-82 through Figure 18-85 illustrate the ADM32 operation for various cases.

Figure 18-82. ADM32 Add and Move Operation for IM&REGTOREM (Case 00)

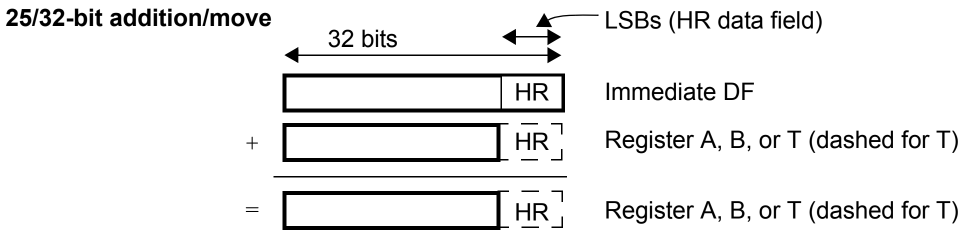


Figure 18-83. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)

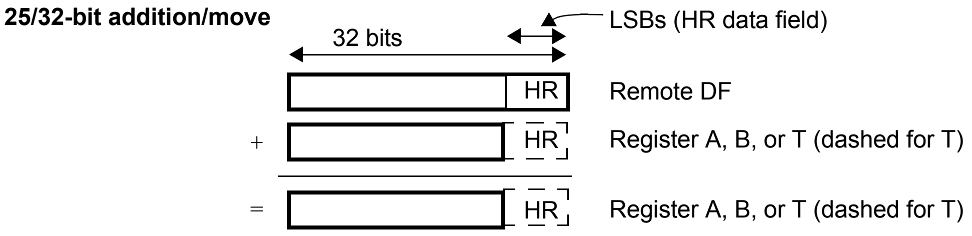

25/32-bit addition/move
32 bits
LSBs (HR data field)
HR
Remote DF
+
HR
Register A, B, or T (dashed for T)
=
HR
Register A, B, or T (dashed for T)

Figure 18-84. ADM32 Add and Move Operation for IM&REMTOREG (Case 10)

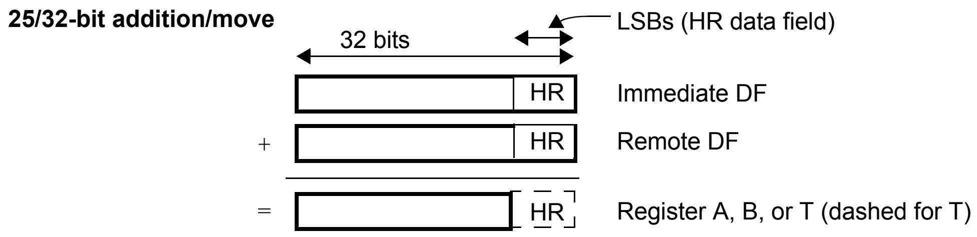
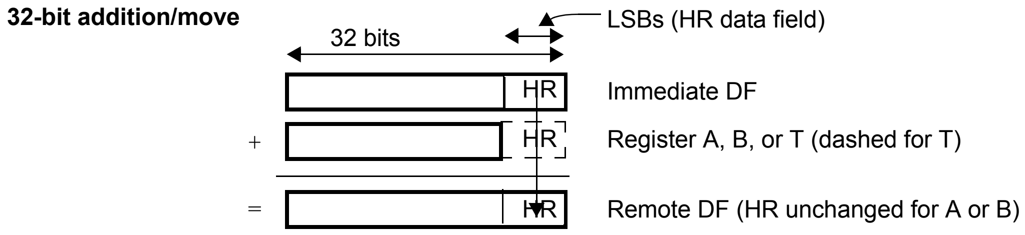


Figure 18-85. ADM32 Add and Move Operation for IM&REGTOREM (Case 11)


32-bit addition/move
32 bits
LSBs (HR data field)
HR
Immediate DF
+
HR
Register A, B, or T (dashed for T)
=
HR
Remote DF (HR unchanged for A or B)

18.7.3.5 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*}**]**
}

Opcode Eh, [P12:P9]

Figure 18-86. APCNT 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	Opcode	Int. ena.	Edge select	Reserved
6	3	1	9	4	1	2	6

Figure 18-87. APCNT 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 18-88. APCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles

One or two cycles
One cycle (normal operation) two cycles (edge detected)

•Cycle 1: edge detected (normal operation)

•Cycle 2: edge detected and GPF = 1 and underflow condition is true

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 NHET[2]. The toothed wheel must then be connected to pin NHET[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 .

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 18-50. 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 NHET[2])
       {
       Register A = 1FFFFFFh;
       APCNT Ovflw flag = 1;
       }
If (specified edge detected on NHET[2])
       {
       Z = 1;
       If (Data field register == 1FFFFFFh)
            {
```

```
           Register A = 1FFFFFFh;
           Register T = 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)
             SW interrupt flag = 1;
           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 NHET[2] pin;
Jump to Next Program Address;
```

18.7.3.6 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**}
[irq={**OFF | ON**}**]**
}

Opcode Dh, [P12:P9]

Figure 18-89. BR Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	Opcode	Reserved
6	3	1	9	4	9

Figure 18-90. BR Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 13	12 8	7 5	4 1	0
Res.	Request type	Control	Prv	Res.	Conditional address	Pin select	Branch cond.	Res.	Int. ena.
3	2	1	1	3	9	5	3	4	1

Figure 18-91. BR Data Field (D31:D0)

31 0
Reserved
32

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.

event Specifies the event that triggers a jump to the indexed program address.

Default: FALL

Table 18-51 provides the branch condition encoding.

Table 18-51. Branch Condition Encoding for BR

event	C7	C6	C5	Branch Condition
NOCOND	0	0	0	Always
FALL	0	0	1	On falling edge on the selected pin
RISE	0	1	0	On rising edge on selected pin
BOTH	0	1	1	On rising or falling edge on selected pin
ZERO	1	0	0	If Zero flag is set
NAF	1	0	1	If NAF_global flag is set
LOW	1	1	0	On LOW level on selected pin
HIGH	1	1	1	On HIGH level on selected pin

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.

Execution

```
If (Condition is true)
      {
      If (Interrupt Enable == 1)
            SW interrupt flag = 1;
      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;
```

18.7.3.7 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***]**
}

Opcode 6h, [P12:P9]

Figure 18-92. CNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5	4 1	0
0	Request Number	BRK	Next program address	Opcode	Angle count.	Register	Comp. Select	Reserved	Int. ena.
6	3	1	9	4	1	2	1	4	1

Figure 18-93. CNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Res.	Max Count
3	2	1	1	25

Figure 18-94. CNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

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 resolution 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)
```

```
               SW interrupt flag = 1;
            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 (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)
               SW interrupt flag = 1;
          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;
```

18.7.3.8 DADM64 (Data Add Move)

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 | 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*}**]**
}

Opcode 2h, [P12:P9]

Figure 18-95. DADM64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	Opcode	Remote Address
6	3	1	9	4	9

Figure 18-96. DADM64 Control Field (P31:P0)

31 29	28 27	26	25 23	22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Comp. Mode	Action	Register select	Int. ena.
3	2	1	3	1	9	5	1	2	2	2	1

Figure 18-97. 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 18-98 shows the DADM64 add and move operation.

Figure 18-98. DADM64 Add and Move Operation

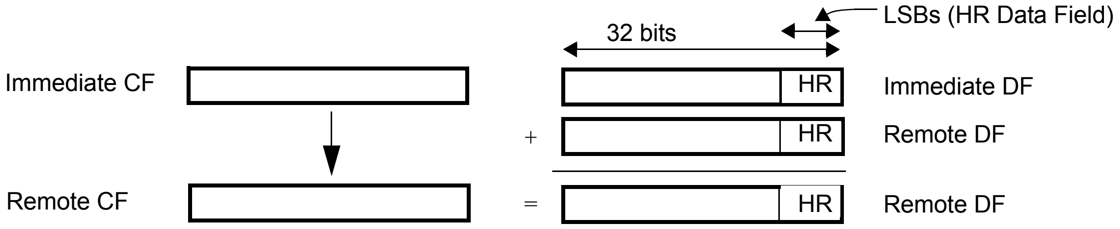


Table 18-52. DADM64 Control Field Description

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
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;
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

18.7.3.9 DJZ (Decrement and Jump if Zero)†

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*}**]**
}

Opcode Ah, [P12:P9];
Sub-Opcode *Sub-opcode [P6-P5]=10*

Figure 18-99. 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	Opcode	Res.	Sub-op-code	Reserved
6	3	1	9	4	1	2	6

Figure 18-100. DJZ Control Field (C31:C0)

31–29	28–27	26	25–22	21–13	12–3	2–1	0
Res.	Request type	Control	Res.	Conditional address	Reserved	Register select	Int. ena.
3	2	1	4	9	10	2	1

Figure 18-101. DJZ Data Field (D31:D0)

31–7	6–0
Data	Reserved
25	7

Cycles One

Register modified selected register (A, B, or T)

† DJNZ is also supported syntax. The functionality of the two instruction names is identical.

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 value != 0)
      {
      Selected register  = Data field value - 1;
      Down Counter value = Data field value - 1;
      Jump to Next Program Address;
      }
else
      {
      Selected register = 000000h;

      If (Interrupt Enable == 1)
         SW interrupt flag = 1;
      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;
      }
```

18.7.3.10 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 | T | NONE**}
[irq={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
[hr_data={*7-bit unsigned integer*}**]**
}

Opcode 0h, [P12:P9];
Sub-Opcode *00h, [C6:C5]*

Figure 18-102. ECMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7	6 0
0	Request Number	BRK	Next program address	Opcode	hr_lr	Angle comp.	Reserved
6	3	1	9	4	1	1	7

Figure 18-103. ECMP Control Field (P31:P0)

31 29	28 27	26	25 23	22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Sub-opcode	Action	Register select	Int. ena.
3	2	1	3	1	9	5	1	2	2	2	1

Figure 18-104. ECMP Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified Register 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, or T).

If T-register is selected, and if the 25-bit data field matches, ECMP updates T-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 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)
              SW interrupt flag = 1;
           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 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
           Jump to Next Program Address;
      }
If (Angle Comp. bit == 1 AND NAF_global == 0)
      Jump to Next Program Address;
```

18.7.3.11 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 | T | NONE**}**]**
[irq={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
}

Opcode Ah;

Sub-Opcode *01h, [P7-P6]*

Figure 18-105. 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	Opcode	Res.	Sub-op-code	Reserved
6	3	1	9	4	1	2	6

Figure 18-106. ECNT Control Field (P31:P0)

31 29	28 27	26	25	24 22	21 13	12 8	7	6 4	3	2 1	0
Res.	Request type	Control	Prv.	Res.	Conditional address	Pin select	Res.	Event	Res.	Register select	Int. ena.
3	2	1	1	3	9	5	1	3	1	2	1

Figure 18-107. ECNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One cycle

Register modified None

Description This instruction defines a specialized 25-bit virtual counter used as an event counter or pulse accumulator (see Table 18-53). The counter value is stored in the data field [D31:D7].

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 18-53. 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 count condition is true according to bits [C6:C4] (see table 1-17))

```
{
      Selected register    = Immediate Data Field + 1;
      Immediate Data Field = Immediate Data Field + 1;

      If (Interrupt Enable == 1)
         SW interrupt flag = 1;
      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;
```

18.7.3.12 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 | T | NONE**}
[irq={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
[hr_data={*7-bit unsigned integer*}**]**
}

Opcode 0h, [P12–P9];
Sub-Opcode *1h, C6*

Figure 18-108. 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	Opcode	hr_lr	Angle comp.	Res.	Save sub.	Reserved
6	3	1	9	4	1	1	1	1	5

Figure 18-109. MCMP Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 13	12 8	7	6	5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Sub-op code	Order	Action	Register select	Int. ena.
3	2	1	3	1	9	5	1	1	1	2	2	1

Figure 18-110. MCMP Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified Register T (implicitly)

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, or T).

Note: The Difference Between Compare Values
The difference between the two data values must not exceed (2^{24}) - 1.

If the hr_lr bit is reset, pin action will occur after a delay from the next 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 (C4). The pin will not change states if the enable pin action bit (C22) is reset.

MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P5) provides the option to save the result of a subtraction into register T.

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 18-54. 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 Comp. bit == 0 OR (Angle Comp. bit == 1 AND NAF_global ==
1))
      {
      If ((C5 == 1 AND (Immediate data value - Selected register) ≥0)
      OR (C5 == 0 AND (Selected register - Immediate data value) ≥0))
          {
          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 (Interrupt Enable == 1)
             SW interrupt flag = 1;
          If ([C28:C27] == 01)
             Generate request on request line [P25:P23];
          If ([C28:C27] == 11)
             Generate quiet request on request line
             [P25:P23];


          If (Save sub. == 1)
            {
            If ([C5] == 1)
              T register[31:7] = Immediate data value - Selected register;
              T-register[6:0]  = 0x00;
            If ([C5] == 0)
              T register[31:7] = Selected register - Immediate data
              value;
              T register[6:0]  = 0x00;
            }
          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
                    Jump to Next Program Address;
                }
        If (Angle Comp. bit == 1 AND NAF_global == 0)
                Jump to Next Program Address;
```

18.7.3.13 MOV32 (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**}**]**
type={**IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG**}
reg={**A | B | T | NONE**}
[data={*25-bit unsigned integer*}**]**
[hr_data={*7-bit unsigned integer*}**]**
}

Opcode 4h, [P12:P9];
Sub-Opcode *C5=0*

Figure 18-111. MOV32 Program Field (P31:P0)

31–26	25–23	22	21–13	12–9	8–0
0	Reserved	BRK	Next program address	Opcode	Remote address
6	3	1	9	4	9

Figure 18-112. MOV32 Control Field (C31:C0)

31–27	26	25–23	22	21–7	6	5	4–3	2–1	0
Reserved	Control	Res.	Z Fl. Cond.	Reserved	Init flag	Sub-op code	Move type	Register select	Res.
5	1	3	1	15	1	1	2	2	1

Figure 18-113. MOV32 Data Field (D31:D0)

31–7	6–0
Data	HR Data
25	7

Cycles One or two cycles

Register modified Selected register (A, B 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 18-114 through Figure 18-117 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

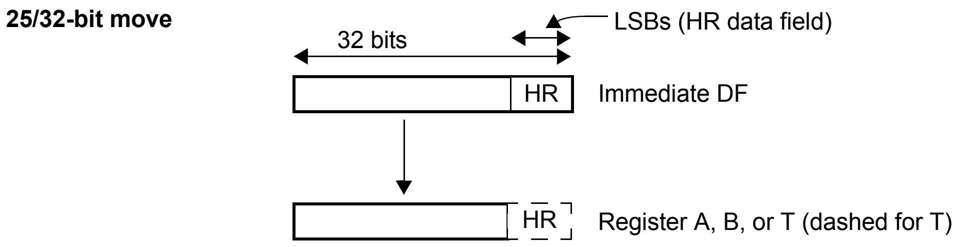
A value of OFF results in no change to the system flags.

type Specifies the move type to be executed.

Table 18-55. Move Type Encoding Selection

Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field & register A, B, or T	1
REGTOREM	1	0	Register A, B, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, or T	2

Figure 18-114. 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 18-115. MOV32 Move Operation for IMTOREG&REM (Case 01)

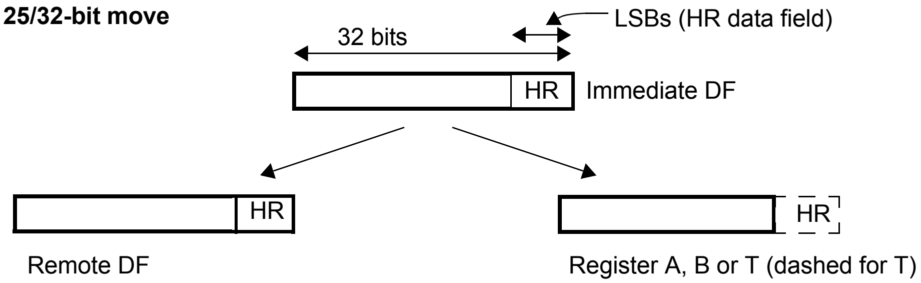


Figure 18-116. MOV32 Move Operation for REGTOREM (Case 10)

25/32-bit move

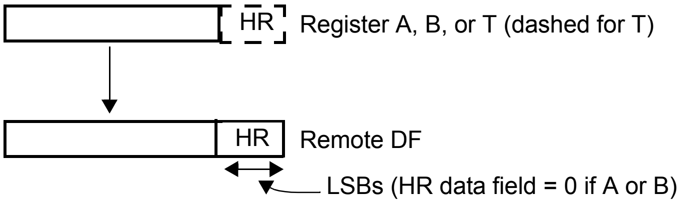
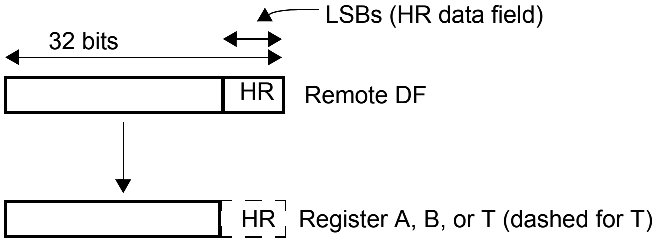

HR
Register A, B, or T (dashed for T)
HR
Remote DF
LSBs (HR data field = 0 if A or B)

Figure 18-117. MOV32 Move Operation for REMTOREG (Case 11)

25/32-bit move



Execution

```
If [(C22 ==0)   OR   ((C22 == 1) AND (Z Flag == 1))]
{
       switch (C4:C3)
       {
       case 00:
            Selected register = Immediate Data Field;
       case 01:
            Selected register = Immediate Data Field;
            Remote Data Field = Immediate Data Field;
       case 10:
            Remote Data Field = Selected register;
       case 11:
            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;
```

18.7.3.14 MOV64 (Data Move)

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 | 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*}**]**

}

Opcode 1h, [P12:P9]

Figure 18-118. MOV64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	Opcode	Remote address
6	3	1	9	4	9

Figure 18-119. MOV64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Comp. mode	Action	Register select	Int. ena.
3	2	1	3	1	9	5	1	2	2	2	1

Figure 18-120. 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 18-121.

Figure 18-121. MOV64 Move Operation

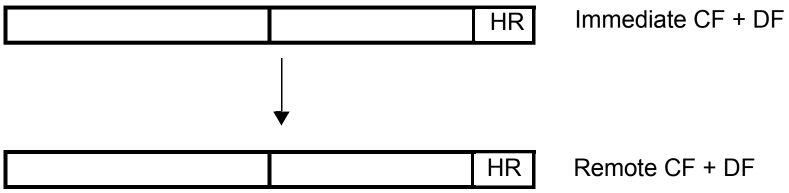


Table 18-56. MOV64 Control Field Descriptions

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	Maintains the control field for the remote instruction.
comp_mode	Selects the comparison mode type to be used by 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 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 18-57. 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;
```

18.7.3.15 PCNT (Period/Pulse Count)

Syntax

PCNT {
[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*}**]**
}

Opcode 7h, [P12:P9]

Figure 18-122. 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	Opcode	Int. en.	Type select	Res.	Pin select
6	3	1	9	4	1	2	1	5

Figure 18-123. 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 18-124. 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 [C24:C0] and in the register A is incremented each NHET loop. PCNT uses the HR structure on the pin to measure an HR period/pulse count value.

Table 18-58. Period/Pulse Count

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 18-59. 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

Table 18-60. Counter Type

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 NHET is used mainly for resetting the counter and start the period count.

Execution

```
Z = 0;
If (Period/Pulse select [P7,P6] == 1X)
      {
      If (Period value != 1FFFFFFh)
          {
          Register A   = Period value + 1;
          Period value = Period value + 1;
          }
      elseIf (specified edge not detected on selected pin)
          Register A   = 1FFFFFFh;

      If (specified edge detected on selected pin)
          {
          Z = 1;
          If (Period value == 1FFFFFFh)
            {
            Register A = 1FFFFFFh;
            Data field = 1FFFFFFh;
            HR Capture Value = 7Fh;
            }
          else
            {
            Data field = Period value + 1;
            HR capture value = selected HR counter;
            Register A = Period Value + 1;
            Period value = 0000000h;

            If (Interrupt Enable == 1)
               SW interrupt flag = 1;
            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 Next Program Address;
            }
          }
      }
else /*** Pulse mode ***/
      {
      If (Period value != 1FFFFFFh)
          {
          Register A   = Period value + 1;
          Period value = Period value + 1;
          }
      If (specified reset edge is detected on selected pin)
          Period value = 0000000h;
      If (specified capture edge is detected on selected pin)
          {
          Z = 1;
          If (Period value == 1FFFFFFh)
            {
            Register A = 1FFFFFFh;
            Data field = 1FFFFFFh;
            HR capture value = 1111111;
            }
          else
            {
            Data field = Period value + 1;
            HR capture value = Selected HR Counter;

            If (Interrupt Enable == 1)
               SW interrupt flag = 1;
            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 Next Program Address;
Prv bit = Current Lx value of selected pin;
```

18.7.3.16 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*}**]**
}

Opcode Ah, [P12:P9];
Sub-Opcode *11h, [P7-P6]*

Figure 18-125. 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	Opcode	hr_lr	Sub-op-code	Reserved
6	3	1	9	4	1	2	6

Figure 18-126. PWCNT Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 13	12 8	7 5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Action	Register select	Int. ena.
3	2	1	3	1	9	5	3	2	2	1

Figure 18-127. 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 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)
             SW interrupt flag = 1;
          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
      }
```

18.7.3.17 RADM64 (Register Add Move)

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 | 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*}**]**
}

Opcode 3h, [P12:P9]

Figure 18-128. RADM64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	Opcode	Remote address
6	3	1	9	4	9

Figure 18-129. RADM64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Res.	En. pin action	Conditional address	Pin select	Res.	Comp. mode	Action	Register select	Int. ena.
3	2	1	3	1	9	5	1	2	2	2	1

Figure 18-130. RADM64 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

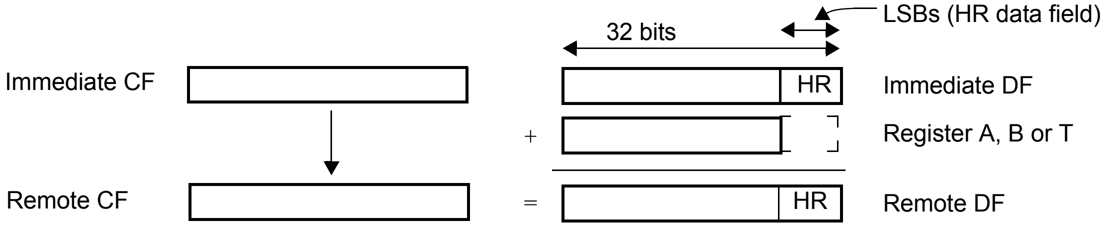
Cycles One

Register modified Selected register (A, B or T)

Description This instruction 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 of the T-register 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 18-131.

Figure 18-131. RADM64 Add and Move Operation



comp_mode Selects the comparison mode type to be used.

Table 18-61. 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

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 data field);
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

18.7.3.18 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*}
}

Opcode 0h, [P12–P9];
Sub-Opcode *[C6-C5]=01*

Figure 18-132. SCMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	Opcode	Reserved
6	3	1	9	4	9

Figure 18-133. SCMP Control Field (C31:C0)

31 29	28 27	26	25	24 23	22	21 13	12 8	7	6 5	4	3 2	1	0
Res.	Request type	Control	Cout prv	Res.	En. pin action	Conditional address	Pin select	Re s.	Sub-opcode	Action	Res.	Restart enable	Int. ena.
3	2	1	1	2	1	9	5	1	2	1	2	1	1

Figure 18-134. 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 is selected (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)
             SW interrupt flag = 1;
          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
          Jump to Next Program Address;
      }
If (Time Mode is selected (C13 == 1)) AND ((Restart En. == 1 AND X ==
0) OR Restart En. == 0)
      {
      Register T = Register A - Immediate Data Field;
      (Result of subtract must not exceed (2^24) - 1)
      Jump to Conditional Program Address;
      }
Cout_prv = Cout; (always executed)
```

18.7.3.19 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*}**]**
}

Opcode Ah, [P12:P9];
Sub-Opcode *Sub-opcode [P7-P6]=00*

Figure 18-135. SCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 4	3 0
0	Reserved	BRK	Next program address	Opcode	Res.	Sub-op-code	Step width	Reserved
6	3	1	9	4	1	2	2	4

Figure 18-136. SCNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.		Control	Res.	Gap start
3	2	1	1	25

Figure 18-137. SCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

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 18-62:

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 the Table 18-62.

Table 18-62. 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 x (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)
      Jump to Next Program Address;
else
      {
      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;
```

18.7.3.20 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 | T | NONE**}**]**
[irq={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
}

Opcode Fh, [P12:P9]

Figure 18-138. 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	Opcode	Reserved	Smode
6	3	1	9	4	5	4

Figure 18-139. SHFT Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Prv.	Res.	Conditional address	Pin select	Res.	Shift condition	Res.	Register select	Int. ena.
3	2	1	1	3	9	5	1	2	2	2	1

Figure 18-140. SHFT 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 shifts the data field of the Instruction. NHET 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 NHET[0] or shift always), register for data storage (A, B, or T), and the data pin.

smode Shift mode.

Table 18-63. 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 18-64. SHIFT Condition Encoding

C6	C5	Shift condition
0	X	Always
1	0	Rising edge of NHET[0]
1	1	Falling edge of NHET[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 CC0 rising edge)
OR (SHIFT condition == 11 AND CC0 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;
          }
    Shift Immediate Data Field by one bit according to bits [P3:P0];
    (see Table 18-63)
    Immediate Data Field = Result of the shift;
    Selected register    = Result of the shift;
    }
If ((Immediate Data Field == all 0's OR
  (Immediate Data Field == all 1's))
    {
    Jump to Conditional Address;
    If (Interrupt Enable == 1)
        {
        SW interrupt flag = 1;
        }
    }
else
    {
    Jump to Next Program Address;
    }

Prv. bit = CC0 Pin level; (Always executed)
Jump to Next Program Address;
```

Note:
The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

18.7.3.21 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 | T | NONE**}
[irq={**OFF | ON**}**]**
data={*25-bit unsigned integer*}
[hr_data={*7-bit unsigned integer*}**]**
}

Opcode Bh, [P12:P9]

Figure 18-141. WCAP Program Field (P31:P0)

31-26	25-23	22	21-13	12-9	8	7-0
0	Request Number	BRK	Next program address	Opcode	hr_lr	Reserved
6	3	1	9	4	1	8

Figure 18-142. WCAP Control Field (C31:C0)

31-29	28-27	26	25	24-22	21-13	12-8	7	6-5	4-3	2-1	0
Res.	Request type	Control	Prv.	Res.	Conditional address	Pin select	Res.	Capture condition	Res.	Register select	Int. ena.
3	2	1	1	3	9	5	1	2	2	2	1

Figure 18-143. WCAP Data Field (D31:D0)

31-7	6-0
Data	HR Data
25	7

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 18-65. 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 edges

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 NHET 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)
       SW interrupt flag = 1;
    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;
```

18.7.3.22 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 | T | NONE}
[irq={OFF | ON}]
[ts_data={*25-bit unsigned integer***}]**
[ec_data={*7-bit unsigned integer***}]**
}

Opcode 8h, [P12:P9]

Figure 18-144. WCAPE Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	Opcode	Reserved
6	3	1	9	4	9

Figure 18-145. WCAPE Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 13	12 8	7	6 5	4 3	2 1	0
Res.	Request type	Control	Prv.	Res.	Conditional address	Pin select	Res.	Capture condition	Res.	Register select	Int. ena.
3	2	1	1	3	9	5	1	2	2	2	1

Figure 18-146. 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 18-66. 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 edges

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF
Default: OFF.

ts_data Specifies the initial 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)
          SW interrupt flag = 1;
    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;

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Chapter 19

SPNU489C-February 2012

High End Timer Transfer Unit (HTU) Module

This reference guide describes High-End Timer Transfer Unit (HTU). The HTU is similar to a DMA (Direct Memory Access) module, but it is specialized to transfer NHET (High End Timer) data to or from the microcontroller RAM.

Topic Page

19.1 Overview

The HET transfer unit is a dedicated direct memory access controller which transfers data between the NHET RAM and RAM buffers located in the main memory address range. This eliminates time consuming CPU accesses to the NHET RAM to gather measurement data or creating output waveforms and thus freeing up the CPU to perform other tasks.

19.1.1 Features

- Independently transfers data between the NHET and the main memory
- 8 double control packets supporting dual buffer configuration
- Transfer requests generated by NHET 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

19.2 Module Operation

The HTU is tightly coupled to the NHET and is not intended to transfer data from other peripheral modules. It initiates transfers with the help of requests generated by the NHET program and configurable control packets. Figure 19-1 shows a system blockdiagram 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 NHET RAM, it is free to do so.

Figure 19-1. System Block Diagram

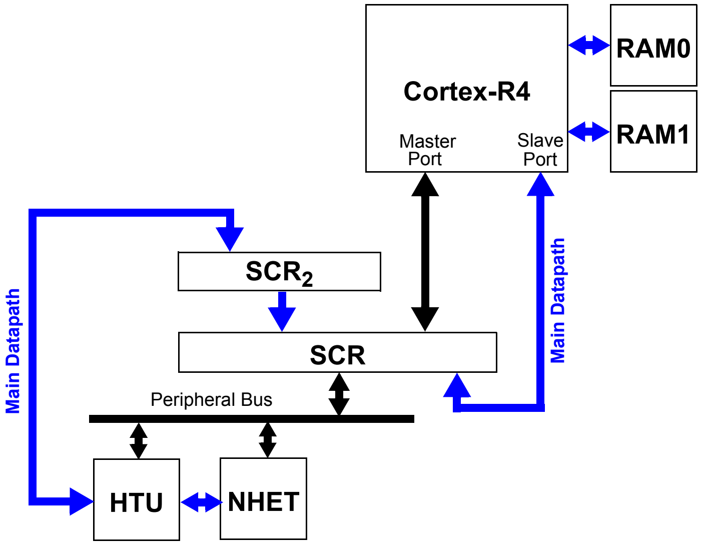
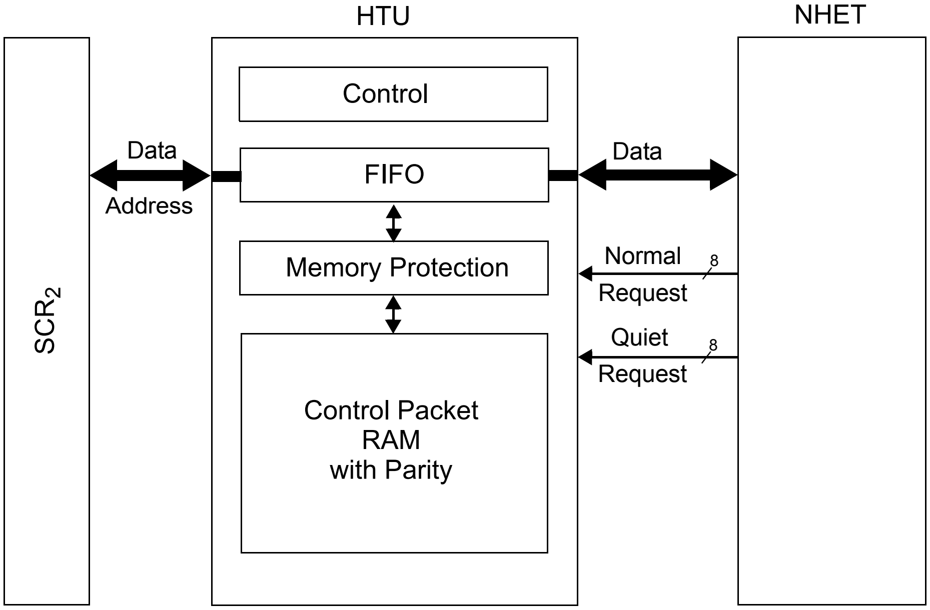


Figure 19-2 shows a more detailed blockdiagram of the HTU module.

Figure 19-2. HTU Block Diagram



Transfers between NHET RAM and the main memory are triggered by 8 different normal NHET requests. Quiet requests are used for specific cases and are discussed in Section 19.2.4.1. Control packets, which store the source and destination addresses, the transfer count and other information (see Section 19.5), 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.

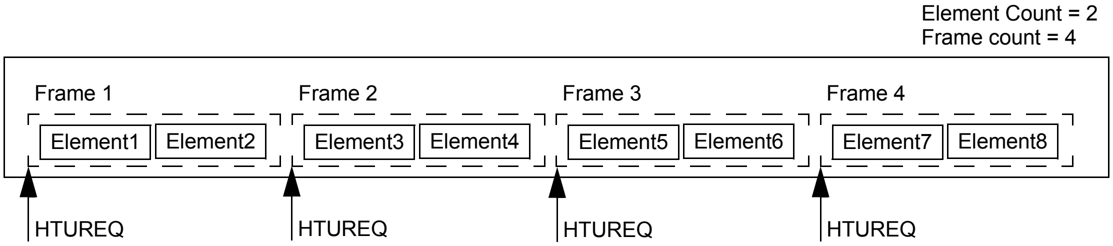
Control packets are implemented as double control packets (DCP) which allow to specify two buffers for the data transfer. This enables the CPU to work with one buffer, while new data is transferred to/from the other buffer.

The control packet defines:

- the start address of the source/destination buffers
- the NHET instruction address location
- how many elements need to be transferred
- the buffer handling

A transfer is triggered when a certain condition (e.g. capture, compare condition) is detected by a NHET instruction. The NHET 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 19-3. Example of a HTU Transfer



19.2.1 Data Transfers between Main RAM and NHET RAM

19.2.1.1 Addressing Modes

The addressing modes of a control packet need to be distinguished between the main RAM of the CPU and the NHET RAM.

Main RAM

For each double control packet (see Section 19.2.1.3) 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 Section 19.3 illustrate the post-increment mode, where the elements of consecutive frames are transferred to/from consecutive locations in the main RAM buffer.

NHET RAM

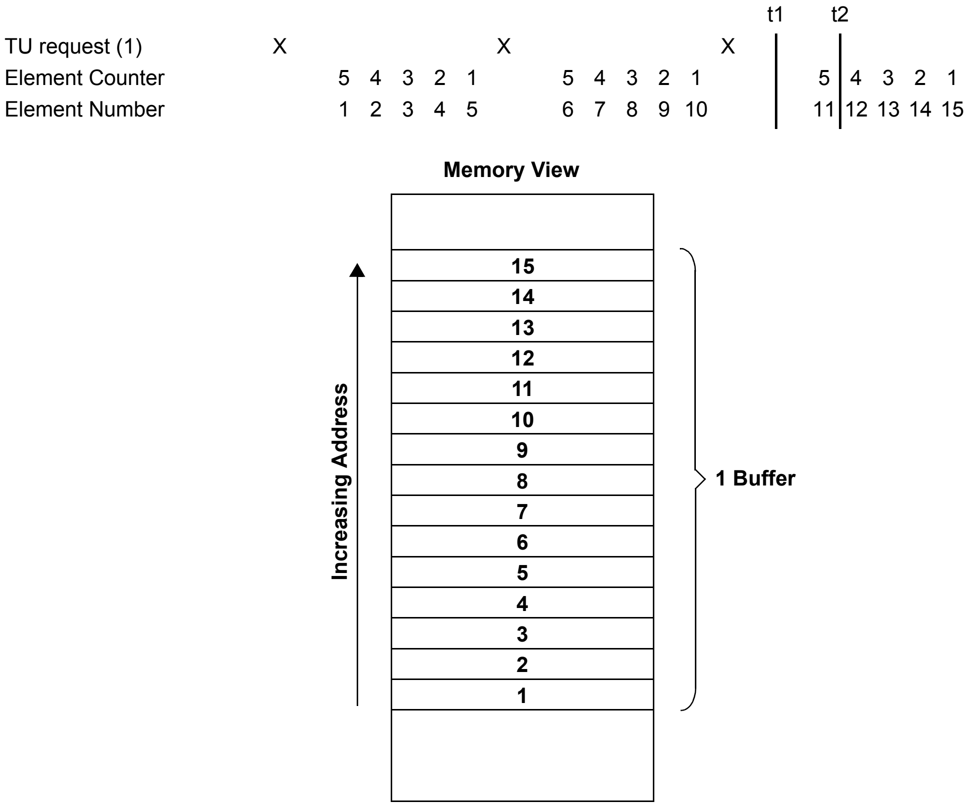
How a DCP addresses the NHET RAM is determined by the initial NHET address, the initial element counter (IETCOUNT) and the NHET addressing mode (ADDMH). The main difference to the main RAM addressing mode is that the HET address is reset to the initial HET address for every first element of a frame. 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.

19.2.1.2 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 19-4 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 (i.e. element count = 5).

Figure 19-4. Single Buffer Timing and Memory Representation



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

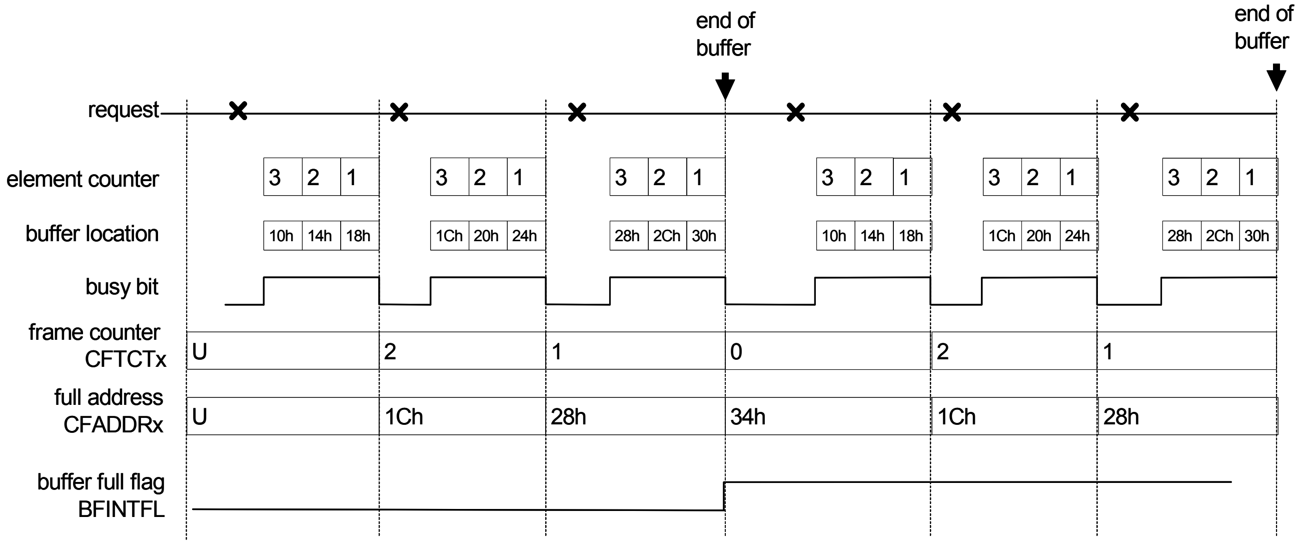
If TMBA or TMBB is set to one shot buffer mode then the data stream will stop after all elements of buffer A or buffer B have been transferred. This means that the corresponding DCP will be disabled after the last frame was transferred to/from buffer A or B and CFTCTA or CFTCTB decrements to zero.

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 Figure 19-5 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 19-5. Timing Example for Circular Buffer Mode



19.2.1.3 Dual Buffer Implementation

The transfer unit provides **double control packets (DCPs)** supporting the use of two buffers per data stream (i.e. per HTU request source). If one buffer should be read by the CPU or DMA, 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 or DMA reads this buffer.

Figure 19-6 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 19-6. Dual Buffer Timing

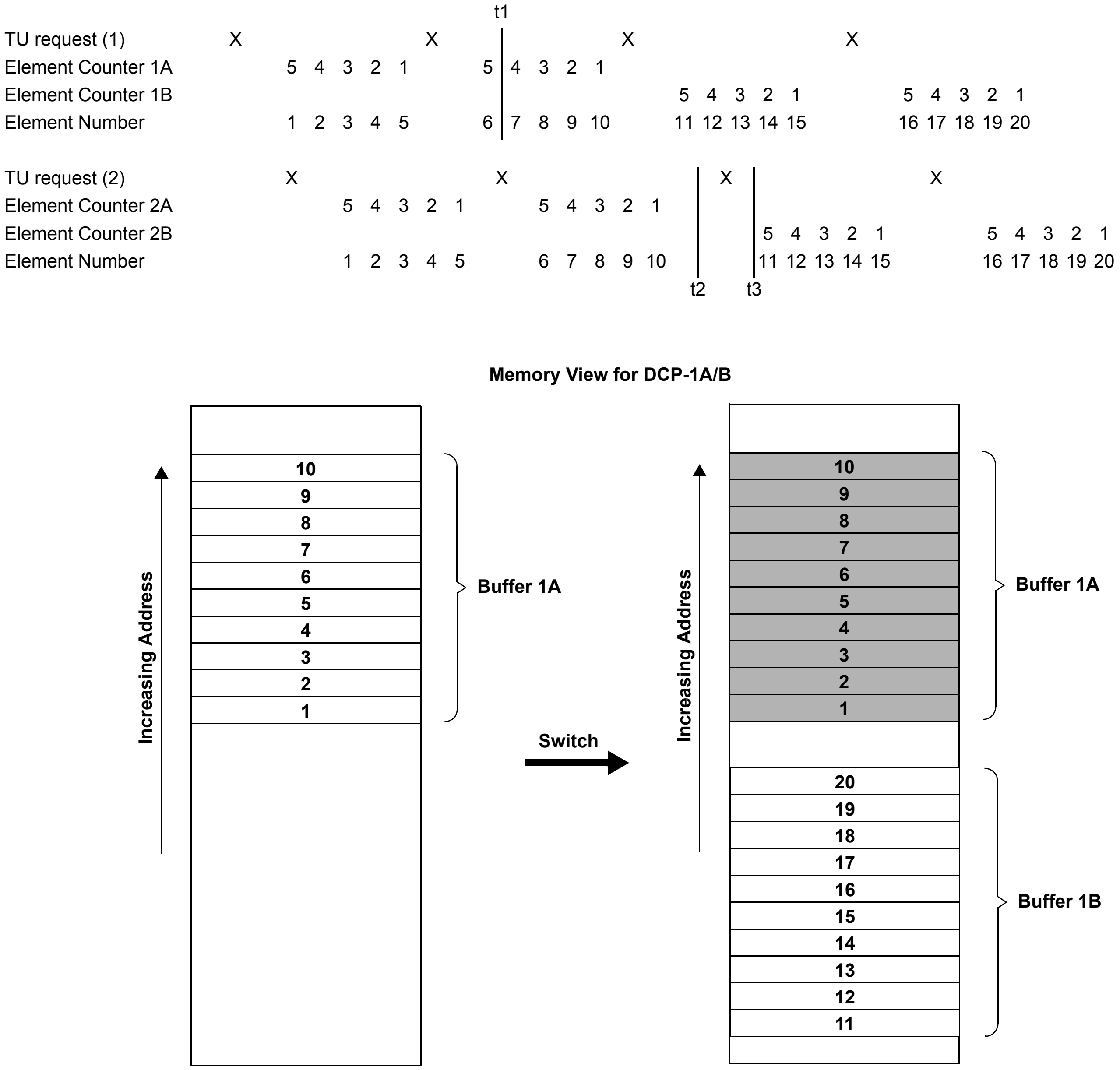

t1
TU request (1) X X X X
Element Counter 1A 5 4 3 2 1 5 4 3 2 1
Element Counter 1B 5 4 3 2 1 5 4 3 2 1
Element Number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
TU request (2) X X X X
Element Counter 2A 5 4 3 2 1 5 4 3 2 1
Element Counter 2B 5 4 3 2 1 5 4 3 2 1
Element Number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
t2
t3
Memory View for DCP-1A/B
Increasing Address
10
9
8
7
6
5
4
3
2
1
Buffer 1A
Switch
Increasing Address
10
9
8
7
6
5
4
3
2
1
Buffer 1A
20
19
18
17
16
15
14
13
12
11
Buffer 1B

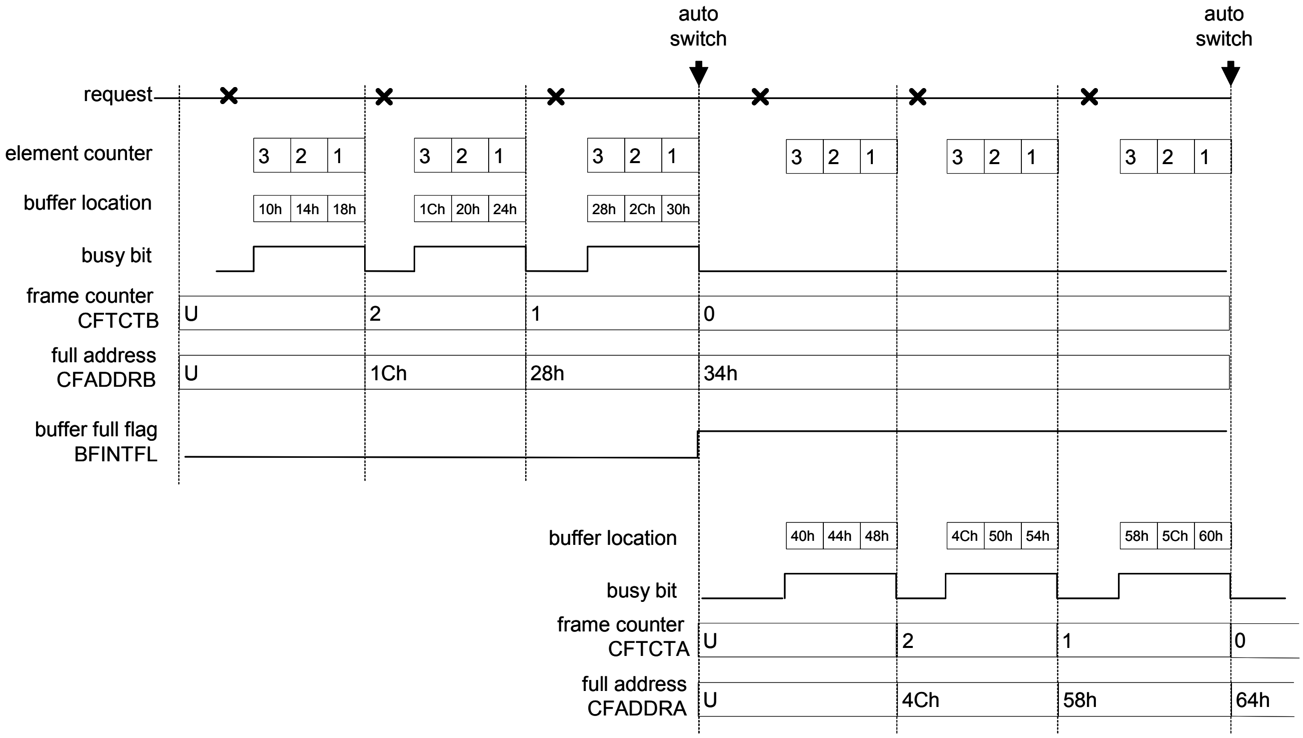
Figure 19-6 shows a switch at time t1, where buffer 1A is frozen and data stream 1 is directed to buffer 1B, but only after the frame has been completed. It also shows the time (t2 or t3) where 2A is frozen and data stream 2 is directed to buffer 2B. If the switch happens between the request and the start of the frame (e.g. 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 19-7 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 19-7. Timing Example for Auto Switch Buffer Mode



19.2.1.4 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. But 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 19-1.

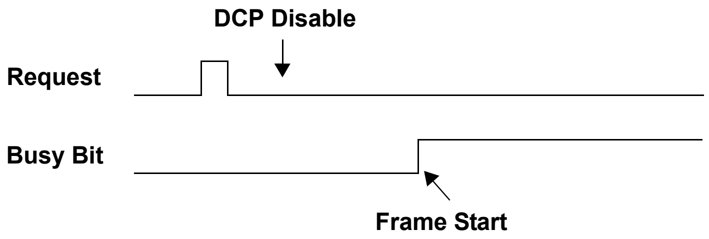
Table 19-1. CPENA / TMBx Priority Rules

Write access to CPENA bits (2·x+1) and (2·x) during the frame with frame counter = 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.

(*) See read table of CPENA register

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 19-8 shows this scenario.

Figure 19-8. 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 NHET, but it will not be able to transfer any data to or from the main memory, while the VBUSHOLD bit is set.

19.2.2 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 NHET instruction) is serviced first.

19.2.3 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 zero).
- Parity Error of DCP x (with parity check enabled and COPE bit set to zero). See also Section 19.2.6, *Control Packet RAM Parity Checking*
- Bus Error of DCP x.
- Memory Protection Error of DCP x (with memory protection enabled). See also Section 19.2.5, *Memory Protection*
- Writing a one to a BUSY bit (belonging to DCP x) if that bit is one. There is no effect if the BUSY bit is zero.
- Writing a one 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.

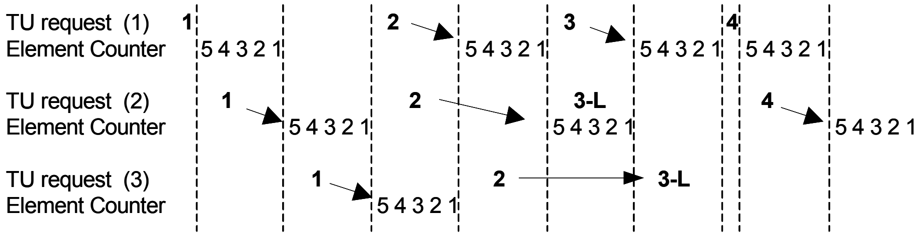
In case of the Bus Error, the counter for the element, which follows the element that caused the error, is captured to the ERRETC register field.

If the HTUEN bit is cleared during a frame is transferred, then the frame will be completed before the HTU is disabled.

19.2.4 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 19-9 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 NHET instruction generates a request on DCP x. The arrows in Figure 19-9 point to the associated frame, which could be delayed compared to the request. The delays are caused by different frames, which are currently processed. Figure 19-9 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 19-9. Timing Example Including Lost Requests



Lost requests are signalled 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.

19.2.4.1 Requests and Quiet Requests

In addition to generating too many transfer request 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 19-10 the L3-Instruction generates the HTU request at time t2, t7, etc. and the according frame (at t3). The frame is delayed because of the HTU load. However as shown in Figure 19-10, 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 19-11) then the frame could fall into the NHET loop (LRP), in which the NHET 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 19-10. Timing, which generates no request lost error

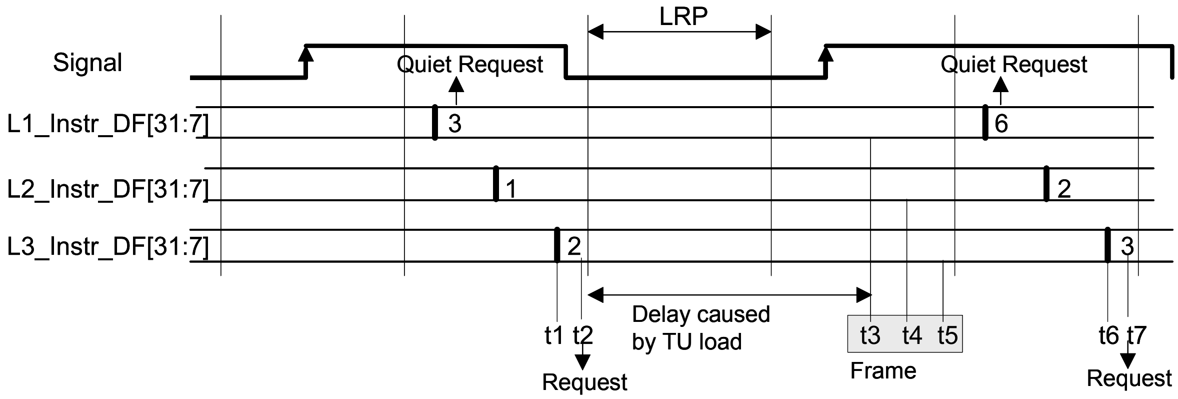
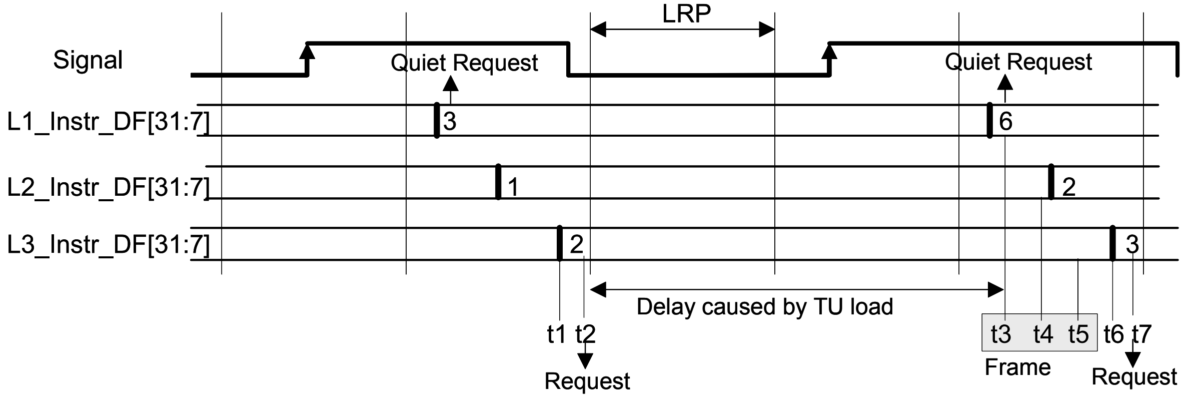


Figure 19-11. Timing, which generates a request lost error



To ensure consistent data the NHET 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 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 19-10 and Figure 19-11 instruction L1 should be configured to generate a quiet request and instruction L3 to generate a normal request. In the case of Figure 19-11 the corresponding bit in the RLOSTFL register will be set.

It is the responsibility of the NHET 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 NHET 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 19-2). A 3-bit field in the program field will select which of the 8 Double Control Packets will be triggered by the request.

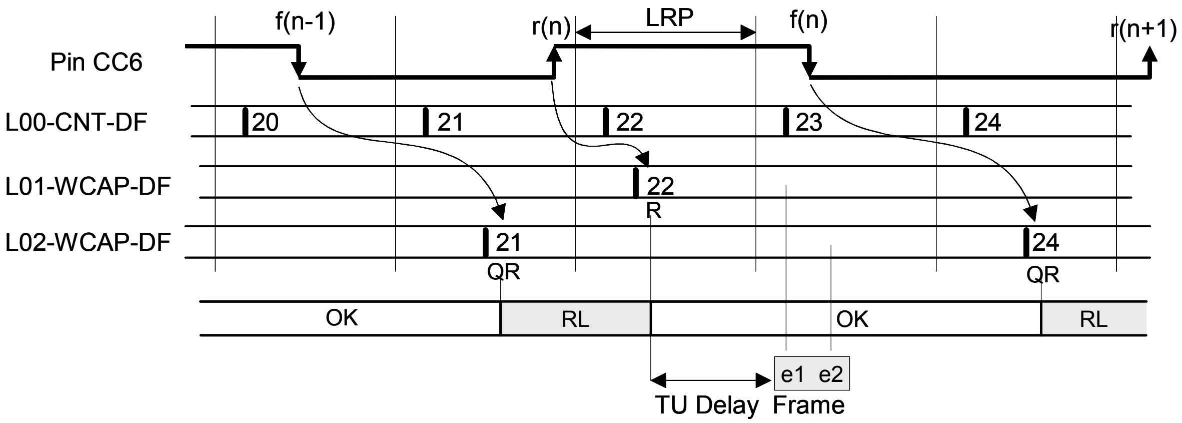
Table 19-2. Triggered Control Packets

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 or DMA request line
0	1	Generate normal request	
1	1	Generate quiet request	

In the case of very light HTU load, but higher signal requirements (e.g. 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
```

Figure 19-12. Timing Example for two WCAP Instructions



The HTU frame will have two elements: The first gives the time stamp of the rising edge r(n) and the second the one 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 19-12 and a request lost will only be signalled 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 [22,24] could be read, but this will be signalled by a request lost error, since at least e2 falls into the RL period.

19.2.5 Memory Protection

This feature allows to restrict 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 which start and end addresses can be configured. With the memory protection feature enabled, read and write accesses by the HTU inside the defined regions are allowed. 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 zero. Bits ACCR01, INTENA01, and register settings of MP1S, and MP1E will be ignored.

To use both regions below rules must followed:

1. Memory mapped region 0 covers a lower memory area as Memory mapped region 1.
2. REG01ENA is a one and REG0ENA is a zero.
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

19.2.6 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 (e.g. 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 0x000, but there is a parity error at byte address 0x00C then the parity error will occur and the captured parity address will be 0x00C and not 0x000. 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 zero 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 (e.g. 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 zero, 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 one 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.

19.2.6.1 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 0xFF4E0200.

When in test mode (i.e. 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 19-3 and Table 19-4 show how the corresponding parity bits of the DCP RAM bytes are mapped into the memory.

Table 19-3. DCP RAM

Bit	31 24	23 16	15 8	7 0
0xFF4E0000	Byte 0	Byte 1	Byte 2	Byte 3
0xFF4E0004	Byte 4	Byte 5	Byte 6	Byte 7
0xFF4E0008	Byte 8	Byte 9	Byte 10	Byte 11
0xFF4E000C	Byte 12	Byte 13	Byte 14	Byte 15

Table 19-4. DCP Parity RAM

Bit		24		16		8		0
0xFF4E0200		P0		P1		P2		P3
0xFF4E0204		P4		P5		P6		P7
0xFF4E0208		P8		P9		P10		P11
0xFF4E020C		P12		P13		P14		P15

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.

19.2.6.2 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 one 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 one 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.

19.3 Use Cases

19.3.1 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 NHET RAM to the buffer RAM. So over time consecutive locations in the RAM buffer can be filled with consecutive measurement values captured into the NHET RAM data field of the same PCNT instruction without loading or interrupting the CPU.

19.3.2 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 NHET input signal (on one pin): Time stamp values (WCAP), edge counter values (ECNT) and last period values (PCNT).

Figure 19-13 shows the timing and Figure 19-14 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 NHET pin.

Figure 19-13. Timing of the WCAP, ECNT, PCNT Example

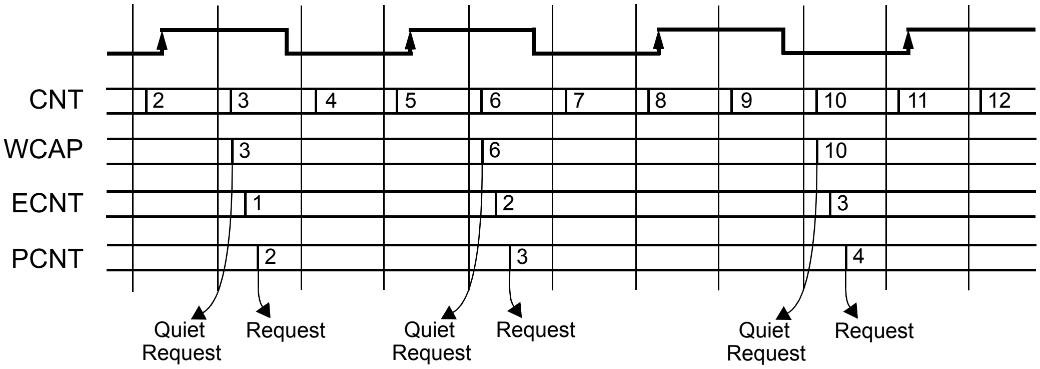


Figure 19-14. Field Addresses of the WCAP, ECNT, PCNT Example

	PF	CF	DF	res
WCAP	0x30	0x34	0x38	0x3C
ECNT	0x40	0x44	0x48	0x4C
PCNT	0x50	0x54	0x58	0x5C

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:

The following diagram 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 19-5. 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)	0x38	0x48	0x58	0x38	0x48	0x58	0x38	0x48	0x58
Destination Address (main CPU RAM)	0x70	0x74	0x78	0x7c	0x80	0x84	0x88	0x8c	0x90

The destination buffer is filled with the WCAP, ECNT and PCNT data field values as shown below.

Table 19-6. Destination Buffer Values

Address	Frame Count	Instruction	Value
0x70	3	WCAP	3
0x74	3	ECNT	1
0x78	3	PCNT	2
0x7C	2	WCAP	6
0x80	2	ECNT	2
0x84	2	PCNT	3
0x88	1	WCAP	10
0x8C	1	ECNT	3
0x90	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]
```

19.3.3 Example: 64-Bit-Transfer of control field and data fields

The following diagram shows how the internal element counter, frame counter and the address registers change over time assuming the same example as in Section 19.3.2, 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 19-7. 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	0x34	0x44	0x54	0x34	0x44	0x54	0x34	0x44	0x54
Full (Destination) Address	0x70	0x78	0x80	0x88	0x90	0x98	0xA0	0xA8	0xB0

[1] The diagram shows the byte addresses

The destination buffer is filled with the WCAP, ECNT and PCNT control and data field values as shown on the right.

Table 19-8. Destination Buffer Values

Address	Frame Count	Instruction	Value
0x70	3	WCAP	Control Field Value
0x74	3	WCAP	3
0x78	3	ECNT	Control Field Value
0x7C	3	ECNT	1
0x80	3	PCNT	Control Field Value
0x84	3	PCNT	2
0x88	2	WCAP	Control Field Value
0x8C	2	WCAP	6
0x90	2	ECNT	Control Field Value
0x94	2	ECNT	2
0x98	2	PCNT	Control Field Value
0x9C	2	PCNT	3
0xA0	1	WCAP	Control Field Value
0xA4	1	WCAP	10
0xA8	1	ECNT	Control Field Value
0xAC	1	ECNT	3
0xB0	1	PCNT	Control Field Value
0xB4	1	PCNT	4

The necessary setup of the HTU control packet (see Section 19.5) 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.

19.4 Control Registers

Figure 19-15 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 0xFFF7 A400. The address locations not listed, are reserved.

Figure 19-15. Control Register Mapping

Offset Address† Register	31 15	30 14	29 13	8 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
0x00 GC Page 1411	Reserved							VBUS HOLD	Reserved							HTU EN
	Reserved							DEBM	Reserved							HTURES
0x04 CPENA Page 1414	Reserved															
	CPENA(15:0)															
0x08 BUSY0 Page 1416	Reserved							BUSY 0A	Reserved							BUSY 0B
	Reserved							BUSY1 A	Reserved							BUSY1 B
0x0C BUSY1 Page 1417	Reserved							BUSY2 A	Reserved							BUSY2 B
	Reserved							BUSY3 A	Reserved							BUSY3 B
0x10 BUSY2 Page 1418	Reserved							BUSY4 A	Reserved							BUSY4 B
	Reserved							BUSY5 A	Reserved							BUSY5 B
0x14 BUSY3 Page 1419	Reserved							BUSY6 A	Reserved							BUSY6 B
	Reserved							BUSY7 A	Reserved							BUSY7 B
0x18 ACP Page 1420	ERRF			ERRETC(4:0)					Reserved				ERRCPN(3:0)			
	TIPF	BUS BUSY		CETCOUNT(4:0)					Reserved				NACP(3:0)			

Figure 19-15. Control Register Mapping (Continued)

Offset Address† Register	31	30	29	8	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20 RLBECTRL Page 1423	Reserved															BER INT ENA
	Reserved							CO RL	Reserved							RL INT ENA
0x24 BFINTS Page 1424	Reserved															
	BFINTENA(15:0)															
0x28 BFINTC Page 1425	Reserved															
	BFINTDIS(15:0)															
0x2C INTMAP Page 1426	Reserved															MAPSEL
	CPINTMAP(15:0)															
0x34 INTOFF0 Page 1427	Reserved															
	Reserved						INTTYPE0 (1:0)		Reserved				CPOFF0(3:0)			
0x38 INTOFF1 Page 1429	Reserved															
	Reserved						INTTYPE1 (1:0)		Reserved				CPOFF1(3:0)			
0x3C BIM Page 1431	Reserved															
	Reserved								BIM(7:0)							
0x40 RLOSTFL Page 1434	Reserved															
	CPRLFL(15:0)															

Figure 19-15. Control Register Mapping (Continued)

<table>
<tr><th>Offset
Address† Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>8
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x44
BFINTFL
Page 1435</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">BFINTFL(15:0)</td></tr>
<tr><td rowspan="2">0x48
BERINTFL
Page 1437</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">BERINTFL(15:0)</td></tr>
<tr><td rowspan="2">0x4C
MP1S
Page 1438</td><td colspan="16">STARTADDRESS1(31:16)</td></tr>
<tr><td colspan="16">STARTADDRESS1(15:0)</td></tr>
<tr><td rowspan="2">0x50
MP1E
Page 1439</td><td colspan="16">ENDADDRESS1(31:16)</td></tr>
<tr><td colspan="16">ENDADDRESS1(15:0)</td></tr>
<tr><td rowspan="2">0x54
DCRTL
Page 1440</td><td colspan="4">Reserved</td><td colspan="4">CPNUM(3:0)</td><td colspan="7">Reserved</td><td>HTUD-
BGS</td></tr>
<tr><td colspan="15">Reserved</td><td>DBGEN</td></tr>
<tr><td rowspan="2">0x58
WPR
Page 1442</td><td colspan="16">WP(31:16)</td></tr>
<tr><td colspan="16">WP(15:0)</td></tr>
<tr><td rowspan="2">0x5C
WMR
Page 1443</td><td colspan="16">WM(31:16)</td></tr>
<tr><td colspan="16">WM(15:0)</td></tr>
<tr><td rowspan="2">0x60
ID
Page 1444</td><td colspan="8">Reserved</td><td colspan="8">CLASS</td></tr>
<tr><td colspan="8">TYPE</td><td colspan="8">REV</td></tr>
</table>

Figure 19-15. Control Register Mapping (Continued)

Offset Address† Register	31	30	29	8	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64 PCR Page 1445	Reserved															CO PE
	Reserved							TEST	Reserved				PARITY_ENA			
0x68 PAR Page 1447	Reserved															PE FT
	Reserved						PAOFF									
0x70 MPCS Page 1448	Reserved				CPNUM0(3:0)				Reserved						MPEF T1	MPEFT 0
	Reserved				CPNUM1(3:0)						INT ENA01	ACCR 01	REG01 ENA	IINT ENA0	ACCR 0	REG0 ENA
0x74 MP0S Page 1451	STARTADDRESS0(31:16)															
	STARTADDRESS0(15:0)															
0x78 MP0E Page 1452	ENDADDRESS0(31:16)															
	ENDADDRESS0(15:0)															

19.4.1 Global Control Register (HTU GC)

Figure 19-16. Global Control Register (HTU GC) [offset = 0x00]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DEBM	Reserved							HTURES
R-0							RWP-0	R-0							RWP-0

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-9. Global Control Register (HTU GC) Field Descriptions

Bit	Name	Value	Description
31-25	Reserved		Reads return zeros and 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 NHET module. When the BUS_BUSY bit is zero 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 zero. Request lost conditions will be detected and interrupts generated if they are enabled.
23-17	Reserved		Reads return zeros and writes have no effect.

Table 19-9. Global Control Register (HTU GC) Field Descriptions (Continued)

Bit	Name	Value	Description
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 one to it from carrying out unintended bus transactions. If the HTUEN bit is cleared 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 Section 19.2.6.2, *Initializing Parity Bits*). If HTUEN is one 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 one 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		Reads return zeros and 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		Reads return zeros and writes have no effect.

Table 19-9. Global Control Register (HTU GC) Field Descriptions (Continued)

Bit	Name	Value	Description
0	HTURES		HTU Software Reset
		0	The Transfer Unit is not in reset state
		1	The Transfer Unit is enabled 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.

19.4.2 *Control Packet Enable Register (HTU CPENA)*

This register enables or disables the individual double control packets (DCP).

Figure 19-17. Control Packet Enable Register (HTU CPENA) [offset = 0x04]

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*
CPENA(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-10. *Control Packet Enable Register (HTU CPENA)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and 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 19-11 for write rules See Table 19-12 for read rules

Table 19-11. 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 19-12. 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	Can not be read

- The conditions listed in Section 19.2.3, *Conditions for Frame Transfer Interruption* can automatically disable DCP x. In this case bits (2·x) and (2·x+1) are both automatically set to zero.

- 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 19-1.

19.4.3 Control Packet (CP) Busy Register 0 (HTU BUSY0)

This register displays the status of individual control packets.

Figure 19-18. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 0x08]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BUSY 1A	Reserved							BUSY 1B
R-0							RCP-0	R-0							RCP-0

R = Read in all modes, CP = Write 1 in privilege mode to clear the bit, writing zero has no effect, -n = Value after reset

Table 19-13. *Control Packet (CP) Busy Register 0 (HTU BUSY0)* Field Descriptions

Bit	Name	Value	Description
31-25	Reserved		Reads return zeros and writes have no effect.
24	BUSY0A		Busy Flag for CP A of DCP 0
23-17	Reserved		Reads return zeros and writes have no effect.
16	BUSY0B		Busy Flag for CP B of DCP 0
15-9	Reserved		Reads return zeros and writes have no effect.
8	BUSY1A		Busy Flag for CP A of DCP 1
7-1	Reserved		Reads return zeros and 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**. (As shown in the diagram below, 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 zero.
3. At the conditions listed in Section 19.2.3, *Conditions for Frame Transfer Interruption*

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.

19.4.4 Control Packet (CP) Busy Register 1 (HTU BUSY1)

This register displays the status of individual control packets.

Figure 19-19. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0x0C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BUSY 2A	Reserved							BUSY 2B
R-0							RCP-0	R-0							RCP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BUSY 3A	Reserved							BUSY 3B
R-0							RCP-0	R-0							RCP-0

R = Read in all modes, CP = Write 1 in privilege mode to clear the bit, writing zero has no effect, -n = Value after reset

Table 19-14. *Control Packet (CP) Busy Register 1 (HTU BUSY1)* Field Descriptions

Bit	Name	Value	Description
31-25	Reserved		Reads return zeros and writes have no effect.
24	BUSY2A		Busy Flag for CP A of DCP 2
23-17	Reserved		Reads return zeros and writes have no effect.
16	BUSY2B		Busy Flag for CP B of DCP 2
15-9	Reserved		Reads return zeros and writes have no effect.
8	BUSY3A		Busy Flag for CP A of DCP 3
7-1	Reserved		Reads return zeros and writes have no effect.
0	BUSY3B		Busy Flag for CP B of DCP 3

See Section 19.4.3, *Control Packet (CP) Busy Register 0 (HTU BUSY0)* for more details.

19.4.5 Control Packet (CP) Busy Register 2 (HTU BUSY2)

Figure 19-20. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BUSY 4A	Reserved							BUSY 4B
R-0							RCP-0	R-0							RCP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BUSY 5A	Reserved							BUSY 5B
R-0							RCP-0	R-0							RCP-0

R = Read in all modes, CP = Write 1 in privilege mode to clear the bit, writing zero has no effect, -n = Value after reset

Table 19-15. *Control Packet (CP) Busy Register 2 (HTU BUSY2)* Field Descriptions

Bit	Name	Value	Description
31-25	Reserved		Reads return zeros and writes have no effect.
24	BUSY4A		Busy Flag for CP A of DCP 4
23-17	Reserved		Reads return zeros and writes have no effect.
16	BUSY4B		Busy Flag for CP B of DCP 4
15-9	Reserved		Reads return zeros and writes have no effect.
8	BUSY5A		Busy Flag for CP A of DCP 5
7-1	Reserved		Reads return zeros and writes have no effect.
0	BUSY5B		Busy Flag for CP B of DCP 5

See Section 19.4.3, *Control Packet (CP) Busy Register 0 (HTU BUSY0)* for more details.

19.4.6 Control Packet (CP) Busy Register 3 (HTU BUSY3)

Figure 19-21. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 0x14]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BUSY 6A	Reserved							BUSY 6B
R-0							RCP-0	R-0							RCP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							BUSY 7A	Reserved							BUSY 7B
R-0							RCP-0	R-0							RCP-0

R = Read in all modes, CP = Write 1 in privilege mode to clear the bit, writing zero has no effect, -n = Value after reset

Table 19-16. *Control Packet (CP) Busy Register 3 (HTU BUSY3)* Field Descriptions

Bit	Name	Value	Description
31-25	Reserved		Reads return zeros and writes have no effect.
24	BUSY6A		Busy Flag for CP A of DCP 6
23-17	Reserved		Reads return zeros and writes have no effect.
16	BUSY6B		Busy Flag for CP B of DCP 6
15-9	Reserved		Reads return zeros and writes have no effect.
8	BUSY7A		Busy Flag for CP A of DCP 7
7-1	Reserved		Reads return zeros and writes have no effect.
0	BUSY7B		Busy Flag for CP B of DCP 7

19.4.7 Active Control Packet and Error Register (HTU ACPE)

Figure 19-22. Active Control Packet and Error Register (HTU ACPE) [offset = 0x18]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERRF	Reserved		ERRETC(4:0)					Reserved				ERRCPN(3:0)			
RCP-0	R-0		R-0					R-0				R-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIPF	BUS BUSY	Reserved	CETCOUNT(4:0)					Reserved				NACP(3:0)			
R-0	R-0	R-0	R-0					R-0				R-0			

R = Read in all modes, CP = Write 1 in privilege mode to clear the bit, writing zero has no effect, -n = Value after reset

Table 19-17. *Active Control Packet and Error Register (HTU ACPE)* Field Descriptions

Bit	Name	Value	Description
31	ERRF		Error Flag
		0	No error occured
		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 one to ERRF. ERRF can be used to indicate if ERRETC and ERRCPN contain new unread data.
30-29	Reserved		Reads return zeros and writes have no effect.

Table 19-17. *Active Control Packet and Error Register (HTU ACPE)* Field Descriptions (Continued)

Bit	Name	Value	Description
28-24	ERRETC		Error Element Transfer Counter 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 also section 19.2.3, *Conditions for Frame Transfer Interruption*, on page 1399. • 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 one to a BUSY bit, which belongs to the control packet specified by ERRCPN, if that bit is one. There is no effect if the BUSY bit is zero. 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		Reads return zeros and 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.

Table 19-17. *Active Control Packet and Error Register (HTU ACPE)* Field Descriptions (Continued)

Bit	Name	Value	Description
14	BUSBUSY		Bus Is Busy
		0	Bus between NHET and HTU is not busy
		1	When BUSBUSY is one the bus is busy with a transfer. It is different from TIPF above because BUSBUSY will go low after VBUSHOLD is set to one and no transfers are pending between the HTU and the main memory. TIPF will remain one if a transfer is still pending and VBUSPHOLD is one.
13	Reserved		Reads return zeros and writes have no effect.
12-8	CETCOUNT		Current Element Transfer Count CETCOUNT shows the current element transfer counter for the frame which is currently processed. If the HTU doesn't currently transfer any frame CETCOUNT is zero. 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 zero and the HTU is finished updating the current DCP (and the CPENA registers if the required conditions are fulfilled).
7-4	Reserved		Reads return zeros and writes have no effect.
3-0	NACP		Number of Active Control Packet Indicates which CP currently processes a frame. **Active or Recent DCP** **NACPValue** 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.

19.4.8 *Request Lost and Bus Error Control Register (HTU RLBECTRL)*

Figure 19-23. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 0x20]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CORL	Reserved							RL INT ENA
R-0							RWP-0	R-0							RWP-0

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-18. *Request Lost and Bus Error Control Register (HTU RLBECTRL)* Field Descriptions

Bit	Name	Value	Description
31-17	Reserved		Reads return zeros and 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		Reads return zeros and writes have no effect.
8	CORL		Continue On Request Lost Error
		0	Stop current frame on request lost detection. Please see section 19.2.3, *Conditions for Frame Transfer Interruption*, on page 1399
		1	If CORL is one 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		Reads return zeros and 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).

19.4.9 *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 19-24. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 0x24]

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*
BFINTENA(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-19. *Buffer Full Interrupt Enable Set Register (HTU BFINTS)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
16-0	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, i.e. once the frame counter CFTCTA decrements to zero. The same applies for CP B (and CFTCTB).
		0	Interrupt disabled. Writing a zero 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.

19.4.10 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 19-25. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 0x28]

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
BFINTDIS(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-20. *Buffer Full Interrupt Enable Clear Register (HTU BFINTC)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	BFINTDIS		Buffer Full Interrupt Disable Bits
		0	Interrupt disabled. Writing a zero 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.

19.4.11 Interrupt Mapping Register (HTU INTMAP)

Figure 19-26. Interrupt Mapping Register (HTU INTMAP) [offset = 0x2C]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPINTMAP(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-21. *Interrupt Mapping Register (HTU INTMAP)* Field Descriptions

Bit	Name	Value	Description
31-17	Reserved		Reads return zeros and writes have no effect.
16	MAPSEL		Interrupt Mapping Select Bit
		0	If MAPSEL is zero 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 one 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.

19.4.12 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 (i.e. higher row in the table next page) have higher priority.

Figure 19-27. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 0x34]

31-16
Reserved
R-0

15-10	9-8	7-4	3-0
Reserved	INTTYPE0 (1:0)	Reserved	CPOFF0(3:0)
R-0	R-0	R-0	R-0

R = Read in all modes, -n = Value after reset

Table 19-22. *Interrupt Offset Register 0 (HTU INTOFF0)* Field Descriptions

Bit	Name	Value	Description
31-11	Reserved		Reads return zeros and writes have no effect.
10-8	INTTYPE0		Interrupt Type of Interrupt Line 0. Indicates whether a buffer-full, RLOST or BER interrupt - assigned to interrupt line 0 - is currently pending.
		00	No interrupt
		01	Interrupt caused by full buffer on CP/DCP specified by CPOFF0
		10	RLOST interrupt generated by CP/DCP specified by CPOFF0
		11	BER interrupt generated by CP/DCP specified by bits CPOFF0
7-5	Reserved		Reads return zeros and writes have no effect.

Table 19-22. *Interrupt Offset Register 0 (HTU INTOFF0)* Field Descriptions (Continued)

Bit	Name	Value	Description
4-0	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 0 0 0	DCP 0, CP A
		0 0 0 1	DCP 0, CP B
		0 0 1 0	DCP 1, CP A
		0 0 1 1	DCP 1, CP B
		0 1 0 0	DCP 2, CP A
		0 1 0 1	DCP 2, CP B
		0 1 1 0	DCP 3, CP A
		0 1 1 1	DCP 3, CP B
		1 0 0 0	DCP 4, CP A
		1 0 0 1	DCP 4, CP B
		1 0 1 0	DCP 5, CP A
		1 0 1 1	DCP 5, CP B
		1 1 0 0	DCP 6, CP A
		1 1 0 1	DCP 6, CP B
		1 1 1 0	DCP 7, CP A
		1 1 1 1	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

Note:
In order to read INTTYPE0 and CPOFF0 simultaneously always read this register using word or half-word but not using byte accesses.

19.4.13 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 19-28. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 0x38]

31-16
Reserved
R-0

15-10	9-8	7-4	3-0
Reserved	INTTYPE1 (1:0)	Reserved	CPOFF1(3:0)
R-0	R-0	R-0	R-0

R = Read in all modes, -n = Value after reset

Table 19-23. *Interrupt Offset Register 1 (HTU INTOFF1)* Field Descriptions

Bit	Name	Value	Description
31-11	Reserved		Reads return zeros and writes have no effect.
10-8	INTTYPE1		Interrupt Type of Interrupt Line 1. Indicates whether a buffer-full, RLOST or BER interrupt - assigned to interrupt line 1 - is currently pending.
		00	No interrupt
		01	Interrupt caused by full buffer on CP/DCP specified by CPOFF1
		10	RLOST interrupt generated by CP/DCP specified by CPOFF1
		11	BER interrupt generated by CP/DCP specified by bits CPOFF1
7-5	Reserved		Reads return zeros and writes have no effect.

Table 19-23. *Interrupt Offset Register 1 (HTU INTOFF1)* Field Descriptions (Continued)

Bit	Name	Value	Description
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 0 0 0	DCP 0, CP A
		0 0 0 1	DCP 0, CP B
		0 0 1 0	DCP 1, CP A
		0 0 1 1	DCP 1, CP B
		0 1 0 0	DCP 2, CP A
		0 1 0 1	DCP 2, CP B
		0 1 1 0	DCP 3, CP A
		0 1 1 1	DCP 3, CP B
		1 0 0 0	DCP 4, CP A
		1 0 0 1	DCP 4, CP B
		1 0 1 0	DCP 5, CP A
		1 0 1 1	DCP 5, CP B
		1 1 0 0	DCP 6, CP A
		1 1 0 1	DCP 6, CP B
		1 1 1 0	DCP 7, CP A
		1 1 1 1	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

Note:
In order to read INTTYPE1 and CPOFF1 simultaneously always read this register using word or half-word but not using byte accesses.

19.4.14 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.

Figure 19-29. Buffer Initialization Mode Register (HTU BIM) [offset = 0x3C]

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
Reserved								BIM(7:0)							
R-0								RWP-0							

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-24. *Buffer Initialization Mode Register (HTU BIM)* Field Descriptions

Bit	Name	Value	Description
31-8	Reserved		Reads return zeros and 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 (i.e. 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 Table 19-25, *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 19-25 shows more details on the BIM usage.

Table 19-25. Buffer Initialization

Case	Change of CPENA bits (2·x+1) and (2·x)		Action on buffer A or B (of DCP x)		
	Old state (*)	**New state (**)**		**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 (***)	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 (***)	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

(*) See read table of CPENA register

(**) See write table of CPENA register

(***) This is regardless of whether the switch is done by a write acess 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 zero 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 19-25)

```
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 in Table 19-25 and not for all the other cases shown in the table. 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 one 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 one 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 19-25 also apply for this situation. The software should use the procedures explained in the first note before setting HTUEN.

19.4.15 Request Lost Flag Register (HTU RLOSTFL)

Figure 19-30. Request Lost Flag Register (HTU RLOSTFL) [offset = 0x40]

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
CPRLFL(15:0)															
RC-0															

R = Read in all modes, WP = Write 1 in privilege mode to clear the bit only, -n = Value after reset

Table 19-26. *Request Lost Flag Register (HTU RLOSTFL)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	CPRLFL		CP Request Lost Flags
		0	No request was lost. Writing a zero 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 one clears the corresponding flag.

19.4.16 Buffer Full Interrupt Flag Register (HTU BFINTFL)

Figure 19-31. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 0x44]

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
BFINTFL(15:0)															
RWP-0															

R = Read in all modes, WP = Write 1 in privilege mode to clear the bit only, -n = Value after reset

Table 19-27. *Buffer Full Interrupt Flag Register (HTU BFINTFL)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	BFINTFL		Buffer Full Interrupt Flags
		0	No buffer full condition detected. Writing a zero 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 one clears the corresponding flag. **Note:** For writing data to system memory from NHET modules: The buffer full Interrupt flag is set when all data transfer (last frame) is completed on that buffer (without waiting for write status information). If write status comes back after the frame data transfer completion and assume there is an error, then, the buffer full flag interrupt will already be set. If write status comes back during the frame data transfer and assume there is an error, the buffer full flag will not be updated. Buffer full flag must not be used as an indication for actual completion of the DCPx last frame without error. The last frame may or may not have bus errors. Software must poll for busy the busy bit and not buffer full flag for each DCPx (CPA or CPB) and poll error status from other registers. For reading data from system memory and transfer to NHET module, the buffer full flag should not be set as long as there is a read bus error. Read bus error always come in the same cycle as read data so we do not encounter the same limitation as write.

19.4.17 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 the Section 19.2.3, *Conditions for Frame Transfer Interruption*

Figure 19-32. BER Interrupt Flag Register (HTU BERINTFL) [offset = 0x48]

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
BERINTFL(15:0)															
RWP-0															

R = Read in all modes, WP = Write 1 in privilege mode to clear the bit only, -n = Value after reset

Table 19-28. *BER Interrupt Flag Register (HTU BERINTFL)* Field Descriptions

Bit	Name	Value	Description
31-16	Reserved		Reads return zeros and writes have no effect.
15-0	BERINTFL		Bus Error Interrupt Flags
		0	No bus error condition detected. Writing a zero 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 one clears the corresponding flag.

19.4.18 Memory Protection 1 Start address Register (HTU MP1S)

This register configures the start address of memory protection region 1.

Figure 19-33. Memory Protection 1 Start address Register (HTU MP1S) [offset = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS1(31:16)															
RWP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS1(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-29. ***Memory Protection 1 Start address Register (HTU MP1S)*** **Field Descriptions**

Bit	Name	Value	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.

19.4.19 Memory Protection 1 End Address Register (HTU MP1E)

Figure 19-34. Memory Protection 1 End Address Register (HTU MP1E) [offset = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENDADDRESS1(31:16)															
RWP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDADDRESS1(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-30. *Memory Protection 1 End Address Register (HTU MP1E)* Field Descriptions

Bit	Name	Value	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.

19.4.20 Debug Control Register (HTU DCTRL)

This register allows to create watchpoints on access to a certain location. It is intended to help debug the application execution during program development.

Figure 19-35. Debug Control Register (HTU DCTRL)[offset = 0x54]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="4">CPNUM(3:0)</td><td colspan="7">Reserved</td><td>HTU DBGS</td></tr>
<tr><td colspan="4">R-0</td><td colspan="4">R-0</td><td colspan="7">R-0</td><td>RWS-0</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>DBR EN</td></tr>
<tr><td colspan="15">R-0</td><td>RWS-0</td></tr>
</table>

R = Read in all modes, WS = Write in suspend mode only, U = Undefined; -n = Value after test reset

Table 19-31. *Debug Control Register (HTU DCTRL)* Field Descriptions

Bit	Name	Value	Description
31-28	Reserved		Reads return zeros and writes have no effect.
27-24	CPNUM		CP Number. These bit fields indicate the CP which should cause the watch point to match.
		0000	CP A of DCP0
		0001	CP B of DCP0
		0010	CP A of DCP1
		...	...
		1110	CP A of DCP7
		1111	CP B of DCP7
23-17	Reserved		Reads return zeros and 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 one 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		Reads return zeros and writes have no effect.

Table 19-31. *Debug Control Register (HTU DCTRL)* Field Descriptions (Continued)

Bit	Name	Value	Description
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.

19.4.21 Watch Point Register (HTU WPR)

This register defines the main memory address of the watchpoint.

Figure 19-36. Watch Point Register (HTU WPR) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WP(31:16)															
RWS-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WP(15:0)															
RWS-0															

R = Read in all modes, WS = Write in debug mode only, -n = Value after test reset

Table 19-32. *Watch Point Register (HTU WPR)* Field Descriptions

Bit	Name	Value	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 exection 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.

19.4.22 Watch Mask Register (HTU WMR)

This register defines a mask of the main memory address of the watchpoint. It can be used to define a memory range in conjunction with the WPR register.

Figure 19-37. Watch Mask Register (HTU WMR) [offset = 0x5C]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">WM(31:16)</td></tr>
<tr><td colspan="16">RWS-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">WM(15:0)</td></tr>
<tr><td colspan="16">RWS-0</td></tr>
</table>

R = Read in all modes, WS = Write in debug mode only, -n = Value after test reset

Table 19-33. *Watch Mask Register (HTU WMR)* Field Descriptions

Bit	Name	Value	Description
31-0	WM		Watch Mask Register Setting a bit in the WMR register to one 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.

19.4.23 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 19-38. Module Identification Register (HTU ID) [offset = 0x60]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE								REV							
R - Class Subtype Number								R - Module Revision Number							

R = Read in all modes, -n = Value after reset

Table 19-34. ***Module Identification Register (HTU ID*** **Field Descriptions**

Bit	Name	Value	Description
31-17	Reserved		Reads return zeros and 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.

19.4.24 Parity Control Register (HTU PCR)

Figure 19-39. Parity Control Register (HTU PCR) [offset = 0x64]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TEST	Reserved				PARITY_ENA			
R-0							RWP-0	R-0				RWP-0101			

R = Read in all modes, WP = Write in privilege mode, -n = Value after reset

Table 19-35. *Parity Control Register (HTU PCR)* Field Descriptions

Bit	Name	Value	Description
31-17	Reserved		Reads return zeros and 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 (e.g. 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		Reads return zeros and 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. t
		0	Parity bits are not memory mapped
		1	Parity bits are memory mapped
7-4	Reserved		Reads return zeros and writes have no effect.

Table 19-35. ***Parity Control Register (HTU PCR)*** **Field Descriptions (Continued)**

Bit	Name	Value	Description
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.
		0101	Parity check is disabled
		All Others	Parity check is enabled **Note:** It is recommended to write a ‘1010’ to enable error detection, to guard against single bit changes from flipping PARITY_ENA to a disable state.

19.4.25 Parity Address Register (HTU PAR)

Figure 19-40. Parity Address Register (HTU PAR) [offset = 0x68]

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

15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Reserved							PAOFF								
R-0							R-X								

R = Read in all modes, -n = Value after reset (X= Value unchanged after reset)
CP = Write 1 in privilege mode to clear the bit, writing zero has no effect,

Table 19-36. *Parity Address Register (HTU PAR)* Field Descriptions

Bit	Name	Value	Description
31-10	Reserved		Reads return zeros and 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 detected
		1	Fault 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		Reads return zeros and 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.

19.4.26 Memory Protection Control and Status Register (HTU MPCS)

Figure 19-41. Memory Protection Control and Status Register (HTU MPCS) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CPNUM0(3:0)				Reserved						MPEFT1	MP EFT0
R-0				R-0				R-0						RCP-0	RCP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CPNUM1(3:0)						INT ENA01	ACCR01	REG01E NA	INT ENA0	ACCR0	REG0 ENA
R-0				R-0				R-0		RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset
CP = Write 1 in privilege mode to clear the bit, writing zero has no effect,

Table 19-37. *Memory Protection Control and Status Register (HTU MPCS)* Field Descriptions

Bit	Name	Value	Description
31-28	Reserved		Reads return zeros and 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.
		0000	CP A of DCP0
		0001	CP B of DCP0
		0010	CP A of DCP1
		...	...
		1110	CP A of DCP7
		1111	CP B of DCP7
23-18	Reserved		Reads return zeros and 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 MP0E registers, when the access violates the rights defined by ACCR01 and when the REG01ENA bit is set.
		0	No fault detected. Writing a zero has no effect
		1	Fault detected. Writing a one will clear the bit.

Table 19-37. *Memory Protection Control and Status Register (HTU MPCS)* Field Descriptions (Continued)

Bit	Name	Value	Description
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 detected. Writing a zero has no effect
		1	Fault detected. Writing a one will clear the bit.
15-12	Reserved		Reads return zeros and 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.
		0000	CP A of DCP0
		0001	CP B of DCP0
		0010	CP A of DCP1
		...	...
		1110	CP A of DCP7
		1111	CP B of DCP7
7-6	Reserved		Reads return zeros and 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 signalling is disabled
		1	Error signalling 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 MP0E 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

Table 19-37. *Memory Protection Control and Status Register (HTU MPCS)* Field Descriptions (Continued)

Bit	Name	Value	Description
3	REG01ENA		Region Enable 01. This bit needs to be set when working with two memory mapped regions. **REG0ENA must be set to zero** if this bit is set to a one. 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 MP0E 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 signalling is disabled
		1	Error signalling 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
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.

19.4.27 Memory Protection Start Address Register 0 (HTU MP0S)

This register configures the start address of memory protection region 0

Figure 19-42. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS0(31:16)															
RWP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS0(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-38. *Memory Protection 0 Start address Register (HTU MP0S)* Field Descriptions

Bit	Name	Value	Description
31-0	STARTADDRESS0		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.

19.4.28 Memory Protection End Address Register (HTU MP0E)

Figure 19-43. Memory Protection End Address Register (HTU MP0E) [offset = 0x78]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENDADDRESS0(31:16)															
RWP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDADDRESS0(15:0)															
RWP-0															

R = Read in all modes, WP = Write in privilege mode only, -n = Value after reset

Table 19-39. *Memory Protection End Address Register (HTU MP0E)* Field Descriptions

Bit	Name	Value	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.

19.5 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 zero to ensure future compatibility. The HTU RAM can be cleared with the system RAM initialization function.

Figure 19-45 provides a summary of the memory configuration. The base address for DCPs is 0xFF4E 0000.

Figure 19-44. Double Control Packet Memory Map

Offset Address	Name	Control Packet	Description
0x000	IFADDRA	Primary DCP0	Initial main memory address Control Packet A
0x004	IFADDRB		Initial main memory address Control Packet B
0x008	IHADDRCT		Initial NHET address and control
0x00C	ITCOUNT		Initial transfer count
0x010..0x01C		Primary DCP1	
0x020..0x02C		Primary DCP2	
0x030..0x03C		Primary DCP3	
0x040..0x04C		Primary DCP4	
0x050..0x05C		Primary DCP5	
0x060..0x06C		Primary DCP6	
0x070..0x07C		Primary DCP7	
0x080..0x0FC	Reserved		
0x100	CFADDRA	Current DCP0	Current main memory address Control Packet A
0x104	CFADDRB		Current main memory address Control Packet B
0x108	CFCOUNT		Current frame count
0x10C	Reserved		
0x110..0x11C		Current DCP1	
0x120..0x12C		Current DCP2	
0x130..0x13C		Current DCP3	
0x140..0x14C		Current DCP4	
0x150..0x15C		Current DCP5	
0x160..0x16C		Current DCP6	
0x170..0x17C		Current DCP7	

The control packet configuration overview below and description on the next pages shows the configuration for a single DCP. All other double control packets have the same structure.

Figure 19-45. Double Control Packet Configuration

Offset Address	31 15	30 14	29 13	8 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
0x000 IFADDRA Page 1455	IFADDRA(31:16)															
	IFADDRA(15:0)															
0x004 IFADDRB Page 1456	IFADDRB(31:16)															
	IFADDRB(15:0)															

Figure 19-45. Double Control Packet Configuration (Continued)

<table>
<tr><th>Offset Address</th><th>31
15</th><th>30
14</th><th>29
13</th><th>8
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x008
IHADDRCT
Page 1457</td><td colspan="8">Reserved</td><td>DIR</td><td>SIZE</td><td>ADDMH</td><td>ADDMF</td><td colspan="2">TMBA(1:0)</td><td colspan="2">TMBB(1:0)</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="11">IHADDR(10:0)</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="2">0x00C
ITCOUNT
Page 1460</td><td colspan="11">Reserved</td><td colspan="5">IETCOUNT(4:0)</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">IFTCOUNT(7:0)</td></tr>
<tr><td rowspan="2">0x100
CFADDRA
Page 1461</td><td colspan="16">CFADDRA(31:16)</td></tr>
<tr><td colspan="16">CFADDRA(15:0)</td></tr>
<tr><td rowspan="2">0x104
CFADDRB
Page 1462</td><td colspan="16">CFADDRB(31:16)</td></tr>
<tr><td colspan="16">CFADDRB(15:0)</td></tr>
<tr><td rowspan="2">0x108
CFCOUNT
Page 1463</td><td colspan="8">Reserved</td><td colspan="8">CFTCTA(7:0)</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">CFTCTB(7:0)</td></tr>
</table>

19.5.1 Initial Full Address A Register (HTU IFADDRA)

Figure 19-46. Initial Full Address A Register (HTU IFADDRA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IFADDRA(31:16)															
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IFADDRA(15:0)															
RWP-X															

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-40. *Initial Full Address A Register (HTU IFADDRA)* Field Descriptions

Bit	Name	Value	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.

19.5.2 Initial Full Address B Register (HTU IFADDRB)

Figure 19-47. Initial Full Address B Register (HTU IFADDRB)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IFADDRB(31:16)															
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IFADDRB(15:0)															
RWP-X															

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-41. *Initial Full Address B Register (HTU IFADDRB)* Field Descriptions

Bit	Name	Value	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.

19.5.3 Initial NHET Address and Control Register (HTU IHADDRCT)

Figure 19-48. Initial HET Address and Control Register (HTU IHADDRCT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								DIR	SIZE	ADDMH	ADDMF	TMBA(1:0)		TMBB(1:0)	
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			IHADDR(10:0)											Reserved	
RWP-X			RWP-X											RWP-X	

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-42. *Initial HET Address and Control Register (HTU IHADDRCT)* Field Descriptions

Bit	Name	Value	Description
31-24	Reserved		Reads return zeros and writes have no effect.
23	DIR		Direction of Transfer
		0	NHET address is read and main memory address is written
		1	Main memory address is read and NHET address is written
22	SIZE		Size of Transferred Data
		0	32-bit transfer
		1	64-bit transfer
			64-bit transfer examples: If the NHET address points to the NHET instruction Control Field (CF), then the CF and Data Field (DF) will be transferred. If the NHET address points to the Program Field (PF), then the PF and CF will be transferred.

Table 19-42. *Initial HET Address and Control Register (HTU IHADDRCT)* Field Descriptions (Continued)

Bit	Name	Value	Description
21	ADDMH		Addressing Mode NHET Address. This bit determines the NHET address index from one to the next element of a frame.
		0	Increment by 16 bytes **Examples:** If the initial NHET address points to data field of instruction (n). Then the NHET 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 NHET address points to control field of instruction (n), then the NHET 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 NHET instruction blocks into the NHET 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
		00	One Shot buffer mode
		01	Circular buffer mode
		All Others	Auto Switch mode
17-16	TMBB		Transfer Mode for Buffer B
		00	One Shot buffer mode
		01	Circular buffer mode
		All Others	Auto Switch mode
15-13	Reserved		Reads return zeros and writes have no effect.

Table 19-42. *Initial HET Address and Control Register (HTU IHADDRCT)* Field Descriptions (Continued)

Bit	Name	Value	Description
12-2	IHADDR		Initial NHET Address The initial NHET Address points to the NHET field, which is the first element of the frame. The NHET address (Bits 12:2) increments by one for each 32-bit NHET field and starts with zero at the first 32-bit field in the NHET RAM. **Note:** When the HTU addresses the NHET RAM it uses only the number of address bits required for the actual NHET RAM size. If the NHET address exceeds the actual NHET RAM size, the unused MSB bits of the address will be ignored and the address rolls over to the start of the NHET RAM.
1-0	Reserved		Reads return zeros and writes have no effect.

19.5.4 Initial Transfer Count Register (HTU ITCOUNT)

Figure 19-49. Initial Transfer Count Register (HTU ITCOUNT)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="11">Reserved</td><td colspan="5">IETCOUNT(4:0)</td></tr>
<tr><td colspan="11">RWP-X</td><td colspan="5">RWP-X</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">IFTCOUNT(7:0)</td></tr>
<tr><td colspan="8">RWP-X</td><td colspan="8">RWP-X</td></tr>
</table>

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-43. *Initial Transfer Count Register (HTU ITCOUNT)* Field Descriptions

Bit	Name	Value	Description
31-21	Reserved		Reads return zeros and writes have no effect.
20-16	IETCOUNT		Initial Element Transfer Count Defines the number of element transfers.
15-8	Reserved		Reads return zeros and writes have no effect.
7-0	IFTCOUNT		Initial Frame Transfer Count Defines the number of frame transfers.

19.5.5 Current Full Address A Register (HTU CFADDRA)

Figure 19-50. Current Full Address A Register (HTU CFADDRA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFADDRA(31:16)															
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFADDRA(15:0)															
RWP-X															

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-44. *Current Full Address A Register (HTU CFADDRA)* Field Descriptions

Bit	Name	Value	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 Section 19.2.3, *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 zero).
- 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.

19.5.6 Current Full Address B Register (HTU CFADDRB)

Figure 19-51. Current Full Address B Register (HTU CFADDRB)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFADDRB(31:16)															
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFADDRB(15:0)															
RWP-X															

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-45. *Current Full Address B Register (HTU CFADDRB)* Field Descriptions

Bit	Name	Value	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 Section 19.2.3, *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 zero).

– 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.

19.5.7 *Current Frame Count Register (HTU CFCOUNT)*

The current frame count register enables the software to find out the recent element in the frozen buffer while the counter of the active buffer decrements.

Figure 19-52. Current Frame Count Register (HTU CFCOUNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CFTCTA(7:0)							
RWP-X								RWP-X							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CFTCTB(7:0)							
RWP-X								RWP-X							

R = Read in all modes, WP = Write in privilege mode only, X = Unknown; -n = Value after reset

Table 19-46. *Initial Transfer Count Register (HTU ITCOUNT)* Field Descriptions

Bit	Name	Value	Description
31-24	Reserved		Reads return zeros and 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		Reads return zeros and writes have no effect.
7-0	CFTCTB		Current Frame Transfer Count for CP B. It is updated at the end of each frame.

19.6 Examples

19.6.1 Application examples for setting the transfer modes of CP A and B of a DCP

Table 19-47. 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 (i.e. determined by the frequency of the NHET transfer requests).

19.6.2 Software example sequence assuming circular mode for both CP A and B

The example assumes the NHET 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 zero
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 zero. 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 zero

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 zero. 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

19.6.3 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 = 1010 shows that DCP 5 / CP A has caused the RLOSTFL interrupt. The hardware automatically clears bit (2·5+0) in RLOSTFL.
- Reading RLOSTFL= 00000000 10000100 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 00000000 10000100 to RLOSTFL clears bits 2 and 7 and the according pending interrupts.

Chapter 20

SPNU489C-February 2012

Direct Memory Access Controller (DMA) Module

This document describes the TMSx70 Platform direct memory access (DMA) controller.

Topic **Page**

20.1 Overview

The DMA controller is used to transfer data between two locations in the memory map in the background of CPU operations. Typically, the DMA is used to:

- Transfer blocks of data between external and internal data memories
- Restructure portions of internal data memory
- Continually service a peripheral
- Page program sections to internal program memory

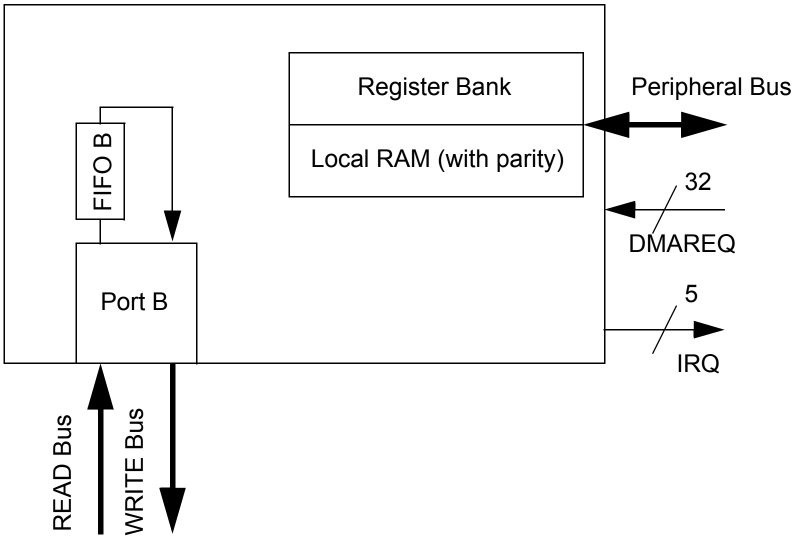
20.1.1 Main Features

- CPU independent data transfer
- One master port - PortB (64 bits wide) that interfaces microcontrollers Memory System.
- FIFO buffer(4 entries deep and each 64bit wide)
- Channel control information is stored in RAM protected by parity
- 16 channels with individual enable
- Channel chaining capability
- 32 peripheral DMA requests
- Hardware and Software DMA requests
- 8, 16, 32 or 64-bit transactions supported
- Multiple addressing modes for source/destination (fixed, increment, offset)
- Auto-initiation
- Power-management mode
- Memory Protection for the address range DMA can access with four configurable memory regions

20.2 Block Diagram

Figure 20-1 gives a detailed view of the DMA internal architecture. DMA data read and write access happens through Port B. FIFO B is 4 levels deep and 64-bits wide.32 DMA requests go into the DMA that can trigger DMA transfers. Five interrupt request lines go out of the DMA to signal that a certain transfer status is reached. Register banks hold the memory mapped DMA configuration registers. Local RAM consists of DMA control packets and is secured by parity. All the programming / configuration of the DMA controller is done via the Peripheral bus.

Figure 20-1. DMA Block Diagram



20.3 Module Operation

The DMA acts as an independent master in the TMSx70 platform architecture. All DMA memory and register accesses are performed in user mode. If the DMA writes to registers which are only accessible in privileged mode, the write will not be performed.

The DMA registers and its local RAM can only be accessed in privilege mode. Therefore, it is not possible for the DMA to reprogram itself. Due to the decoupling of the read and write bus, it is possible to execute a maximum of two transfers in parallel, if source and destination of the transfer are not on the same bus system.

20.3.1 Memory Space

The DMA controller makes no distinction between program memory and data memory. The DMA controller can transfer to and from any space within the 4 Gbyte physical address map, by programming the absolute address for the source and destination in the control packet.

20.3.2 DMA Data Access

The DMA controller refers to data in three levels of granularity:

- **Element:** Depending on the programmed data type, an 8-bit, 16-bit, 32-bit, or a 64-bit value. The type can be individually selected for the source (read) and destination (write). See Figure 20-2 and Figure 20-3 for an example of the use of elements. An element transfer cannot be interrupted.
- **Frame:** One or more elements to be transferred as a unit. A frame transfer can be interrupted between element transfers. See Figure 20-2 for an example.
- **Block:** One or more frames to be transferred as a unit. Each channel can transfer one block of data (once or multiple times). See Figure 20-3 for an example.

Figure 20-2. Example of a DMA Transfer Using Frame Trigger Source

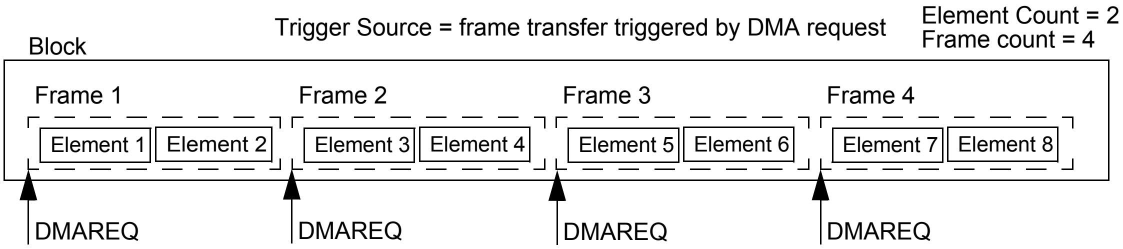
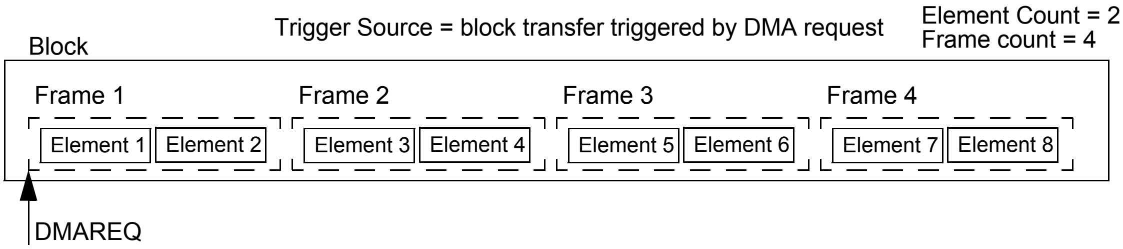


Figure 20-3. Example of a DMA Transfer Using Block Trigger Source



20.3.3 Addressing Modes

There are three addressing modes supported by the DMA controller that can be setup independent for the source and the destination address:

- Constant — source and/or destination addresses do not change.
- Post incremented — source and/or destination address are post-incremented by the element size.
- Indexed — source and/or destination address is post-incremented as defined in the element index offset register(EIOFF) and the frame index offset register (FIOFF).

An unaligned address with respect to the element size is not supported. An address chosen during transfer must be divisible by the element size.

20.3.4 DMA Channel Control Packets

There are a total of 16 control packets. Each control packet is associated with a channel in a fixed order. For example, control packet 0 stores channel information for channel 0. The DMA requests can be mapped to the individual channels as described in Section 20.3.7. Figure 20-4 illustrates the mapping scheme between DMA requests and channels. Each control packet contains nine fields. The first six fields compose the primary control packet and are programmable during DMA setup. The last three fields compose working control packet and are only readable by the CPU. The working control packets are used to support auto-initiation. The organization of control packets is shown in Figure 20-5.

The primary control packet contains channel information such as source address, destination address, transfer count, element/frame index pointer and channel configuration. Source address, destination address and transfer count also have their respective working images. The three fields of working images compose a working control packet and are not accessible to the CPU in write access.

The first time a DMA channel is selected for a transaction, the following process occurs:

1. The primary control packet is first read by the DMA state machine.
2. Once the channel is arbitrated, the current source address, destination address and transfer count are then copied to their respective working images.
3. When the channel is serviced again by the DMA, the state machine will read both the primary control packet and the working control packet to continue the DMA transaction until the end of an entire block transfer.

When the same channel is requested again, the state machine will start again by reading only the primary control packet and then continue the same process described above. The user software need not set up control packets again because the contents of the primary control packet were never lost. The working images of the control packets are reducing the software overhead and interaction with the DMA module to a minimum.

Note:
Changing the contents of a channel control packet will clear the corresponding pending bit (PEND) if the channel has a pending status. If the control packet of an active channel (as indicated in DMASTAT) is changed, then the channel will stop immediately at an arbitration boundary. When the same channel is triggered again, it will begin with the new control packet information.

Figure 20-4. DMA Request Mapping and Control Packet Organization

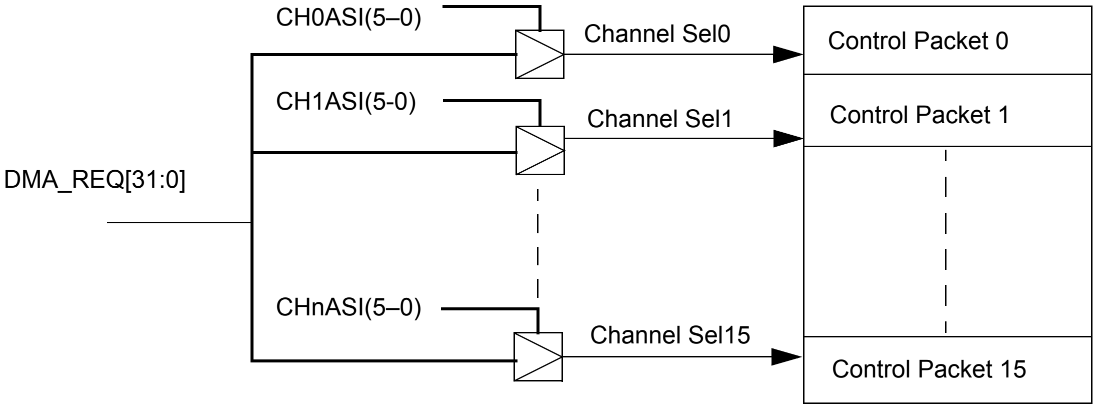
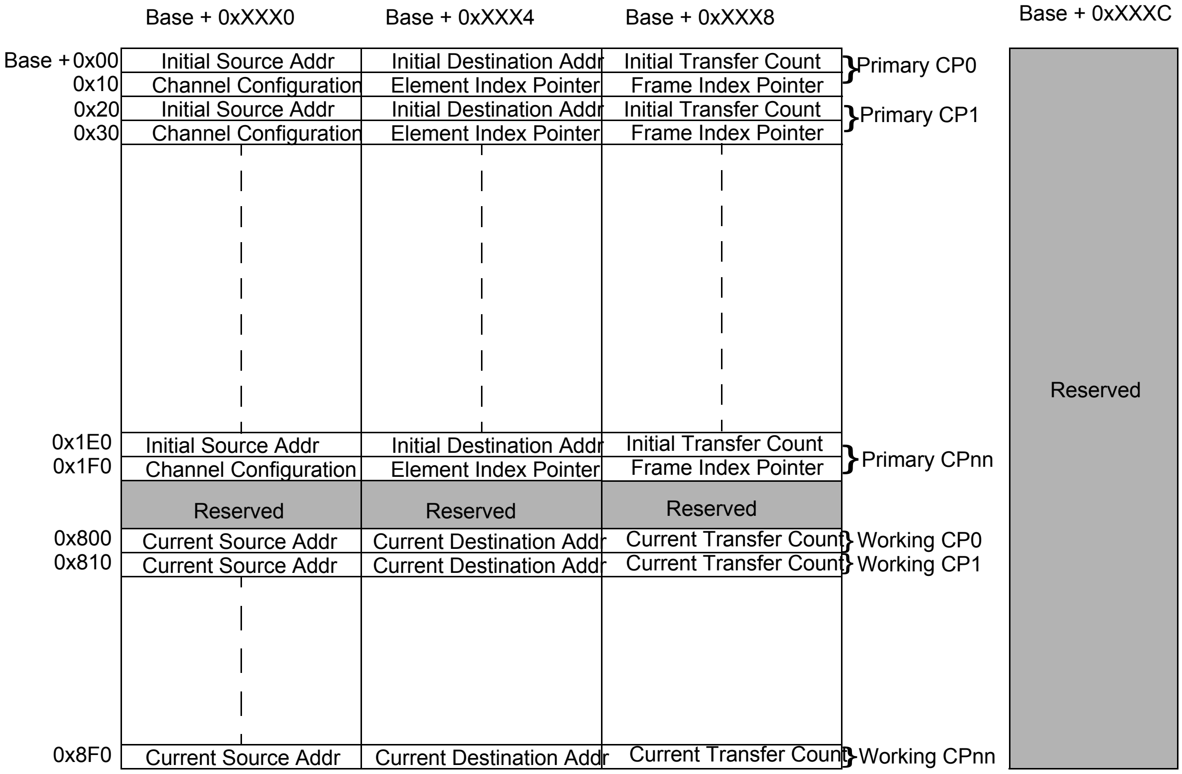


Figure 20-5. Control Packet Organization and Memory Map



20.3.4.1 Initial Source Address

This field stores the absolute 32-bit source address of the DMA transfer.

20.3.4.2 Initial Destination Address

This field stores the absolute 32-bit destination address of the DMA transfer.

20.3.4.3 Initial Transfer Count

The transfer count field is composed of two parts. The frame transfer count value and the element transfer count value. Each count value is 13 bits wide. As a Single Block transfer maximum of 512 Mbytes of data can be transferred. Element count and frame count are programmed according to the source data structure.

The total transfer size is calculated as stated below:

$$\boxed{T_{sz} = E_{rsz} * E_{tc} * F_{tc}} \qquad \text{(EQ 1)}$$

where

T_{sz} = Total transfer size
E_{rsz} = Read Element Size
E_{tc} = Element Transfer Count
F_{tc} = Frame Transfer Count

Note:
A zero element count with a non-zero frame count or a non-zero element count with a zero frame count are all considered as zero total transfer count. No DMA transaction is initiated with any of the counters set to 0.

20.3.4.4 Channel Configuration Word

The channel configuration defines the following individual parameters

- Read element size
- Write element size
- Trigger type (frame or block)
- Addressing mode for source
- Addressing mode for destination
- Auto-initiation mode
- Next control packet to be triggered at control packet finish(Channel Chaining)

20.3.4.5 Element/Frame Index Pointer

There are 4 index pointers that allow the creation of different types of buffers in RAM and address registers in a structured manner: an element index pointer for source and destination and a frame index pointer for source and destination.

The element index pointer for source and/or destination defines the offset to be added after each element transfer to the source and/or destination address. The frame index pointer for source and/or destination defines the offset to be added to the source and/or destination address after the element count reaches zero. The element and frame index pointers must be defined in terms of the number of bytes of offset. The DMA controller does not adjust the element/frame index number according to the element size. An index of 2 means *increment the address by 2* and not by 16 when the element size is 64 bits.

20.3.4.6 Current Source Address

The current source address field contains the current working source address during a DMA transaction. The current source address is incremented during post increment addressing mode or indexing mode.

20.3.4.7 Current Destination Address

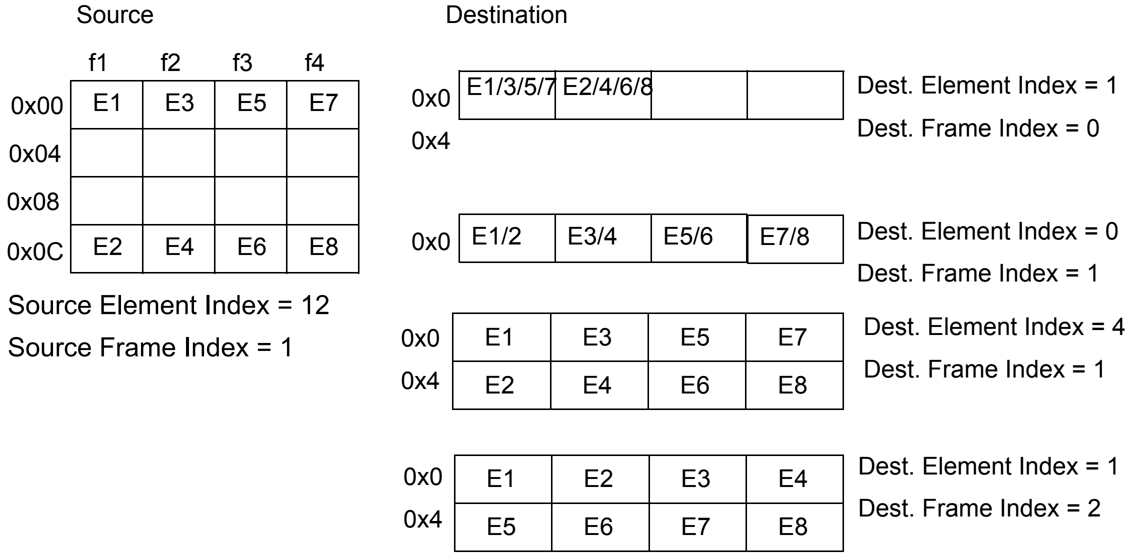
The current destination address field contains the current working destination address during a DMA transaction. The current destination address is incremented during post-increment addressing mode or indexing mode.

20.3.4.8 Current Transfer Count

The current transfer count stores the remaining number of elements to be transferred in a block. It is decremented by one for each element read from the source location.

Figure 20-6, Figure 20-7, and Figure 20-8 show some examples of DMA transfers.

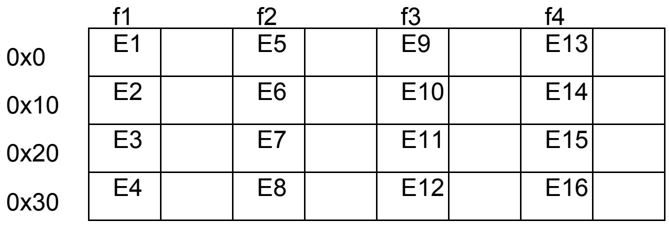
Figure 20-6. DMA Transfer Example 1



The example assumes the following setup.

Read Element Size = 8 bit
Write Element Size = 8 bit
Element Count = 2
Frame Count = 4

Figure 20-7. DMA Indexing Example 1



Element Index = 16

Frame Index = 4

This example can be applied to either source or destination indexing and assumes the following setup.

Element Size = 16 bit
Element Count = 4
Frame Count = 4

Figure 20-8. DMA Indexing Example 2

	f1	f2	f3	f4	f5	f6	f7	f8
0x0	E1	E4	E7	E10	E13	E16	E19	E22
0x20								
0x40	E2	E5	E8	E11	E14	E17	E20	E23
0x60								
0x80	E3	E6	E9	E12	E15	E18	E21	E24

Element Index = 64
Frame Index = 4
This example can be applied to either source or destination indexing and assumes the following setup.

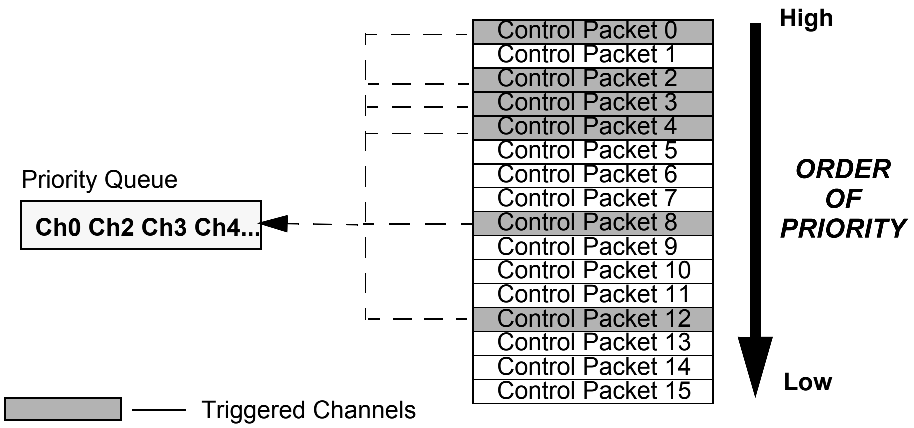
Element Size = 32 bit
Element Count = 3
Frame Count = 8

20.3.5 Priority Queue

User can assign channels in to priority queues to facilitate request handling during arbitration. The port has two priority queues: a high and a low priority queue. The queue can be configured to follow a fixed or rotating priority scheme. Fixed priority is such that the lower the channel number(Figure 20-9), the higher its priority. Rotating priority is based on a round-robin scheme. Initially, the priority list is sorted according to the fixed priority scheme. Channels assigned to the high priority queue are always serviced first according to the selected priority scheme before channels in the low priority queue are serviced. Table 20-1 describes how arbitration is performed according to different priority schemes.

Note:
Since the DMA controller provides the capability to map any one of the 32 hardware DMA request lines to any channel, the numerical order of the hardware DMA request does not imply any priority. The priority of each hardware DMA request is programmed and determined by software.

Figure 20-9. Fixed Priority Scheme



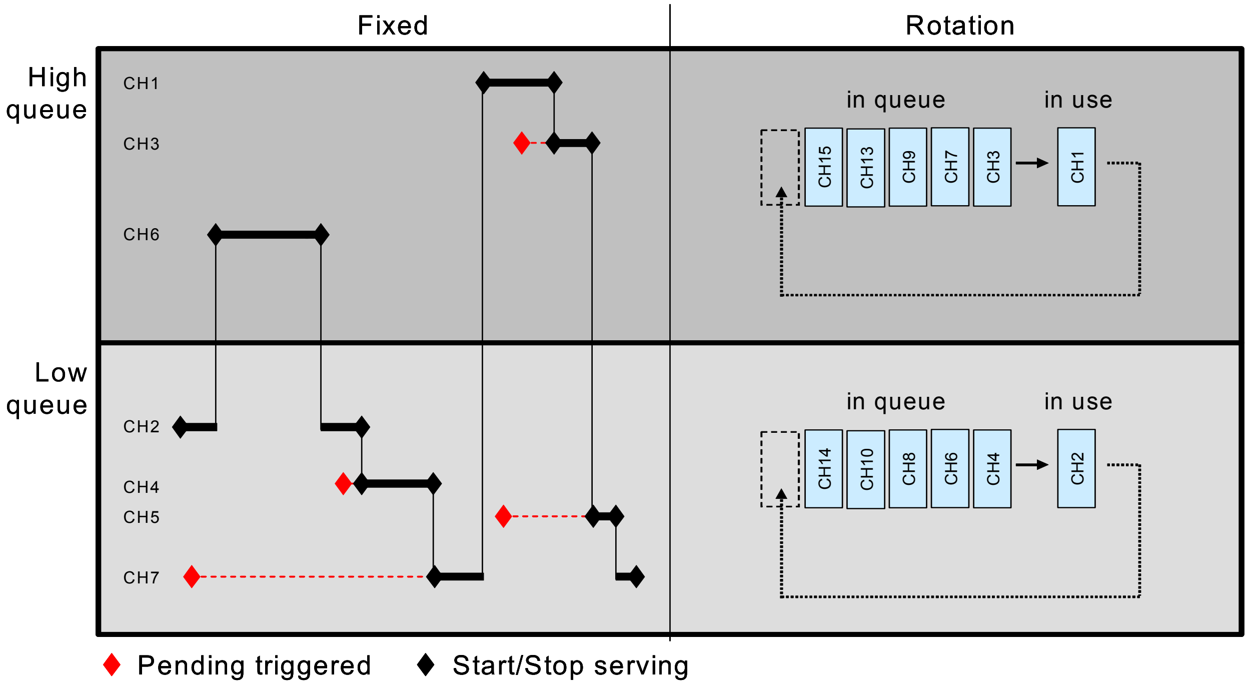
The above figure illustrates that by default Lower the channel number, higher the Priority.

Table 20-1. Arbitration According to Priority Queues and Priority Schemes

Queue	Priority Scheme	Remark
High priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number, the higher the priority. A channel will be arbitrated out whenever there is a higher pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero before the next highest pending channel is serviced. When there is no pending channels left in high queue then the DMA switches to service low queue channels.
	Rotating	Channels are arbitrated by using the round-robin scheme. Arbitration is performed when the FIFO is empty. When there are no pending channels left in high queue then the DMA switches to service low queue channels.
Low priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number the higher the priority. A channel will be arbitrated out whenever there is a higher-priority pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero, before the next highest pending channel is serviced. If there is a pending channel in the high-priority queue while DMA is servicing a low queue channel then DMA will switch back to service high queue channel after an arbitration boundary.
	Rotating	Channels are arbitrated by using round-robin scheme. Arbitration is performed when the FIFO is empty.

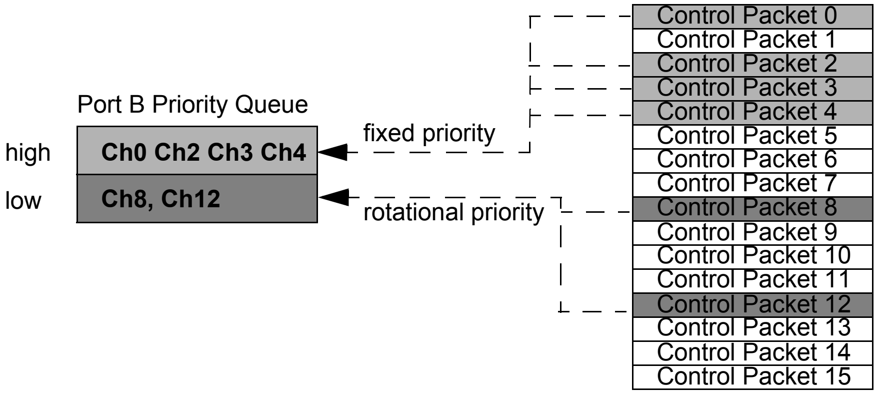
A Simple Priority Queues example in both Fixed and Rotation Scheme is shown in Figure 20-10.

Figure 20-10. Example of Priority Queues



For optimal system performance, the high priority channels should be put in fixed arbitration scheme and low priority channels in the rotating priority scheme as illustrated in Figure 20-11.

Figure 20-11. Example Channel Assignments



1 The above figure illustrates the channel assignments in a system with 16 channels. This approach can be scaled dependent on the total channels available.

20.3.6 Data Packing and Unpacking

The DMA controller automatically performs the necessary data packing and unpacking when the read element size differs from the write element size. Data packing is required when the read element size is smaller than the write element size; data unpacking is required when the read element size is larger than the write element size. When the read element size is equal to the write element size, no packing is performed during read, nor is any unpacking performed during write.

Figure 20-12 shows an example of data unpacking in which the DMA is used to transfer 128 transmit data elements to the MibSPI FIFO buffer. In this example, data unpacking is required because the read element size is 64 while the write element size is 16. The DMA first performs an 64-bit read from the source into its FIFO buffer. After the 64-bit data is read into the DMA FIFO buffer, it must unpack the data into four 16-bit data elements before writing out to the destination. Therefore the DMA would need to perform four 16 bit write operations to the destination.

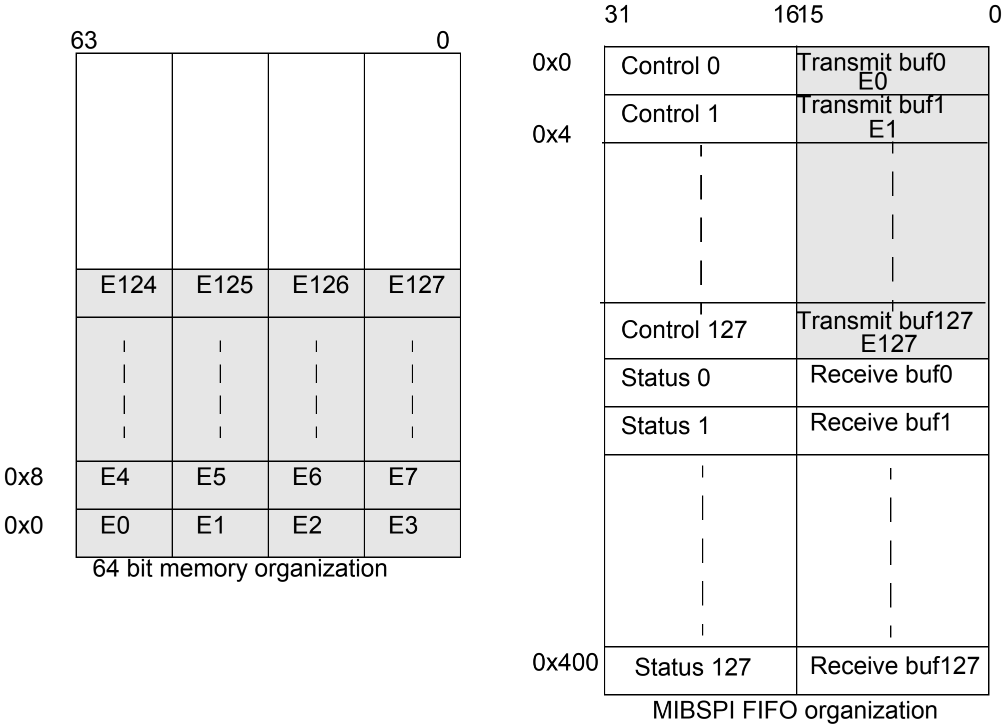
Note:

In the example in Figure 20-12, to transmit data at the lower bits of the MibSPI, bits 15:0, the destination address should be incremented by a factor of 2.

Note:

1) The element Count (ITCOUNT) refers only to the read element.
2) Data unpacking does not require the DMA request. Once the DMA request is received, data from Source is moved in to FIFO and unpacking happens until the FIFO is empty.
3) DMA assumes the destination is always ready and will perform write immediately. In case of data unpacking and Constant Addressing Mode write (ADDMW(1–0) = 0) the destination data will be overwritten by next data or next data might be skipped in case the destination has overflow protection (e.g.., SCITD register). User should configure DMA to avoid data unpacking if the Destination is configured as Constant Addressing Mode write to avoid data loss.

Figure 20-12. Example of DMA Data Unpacking



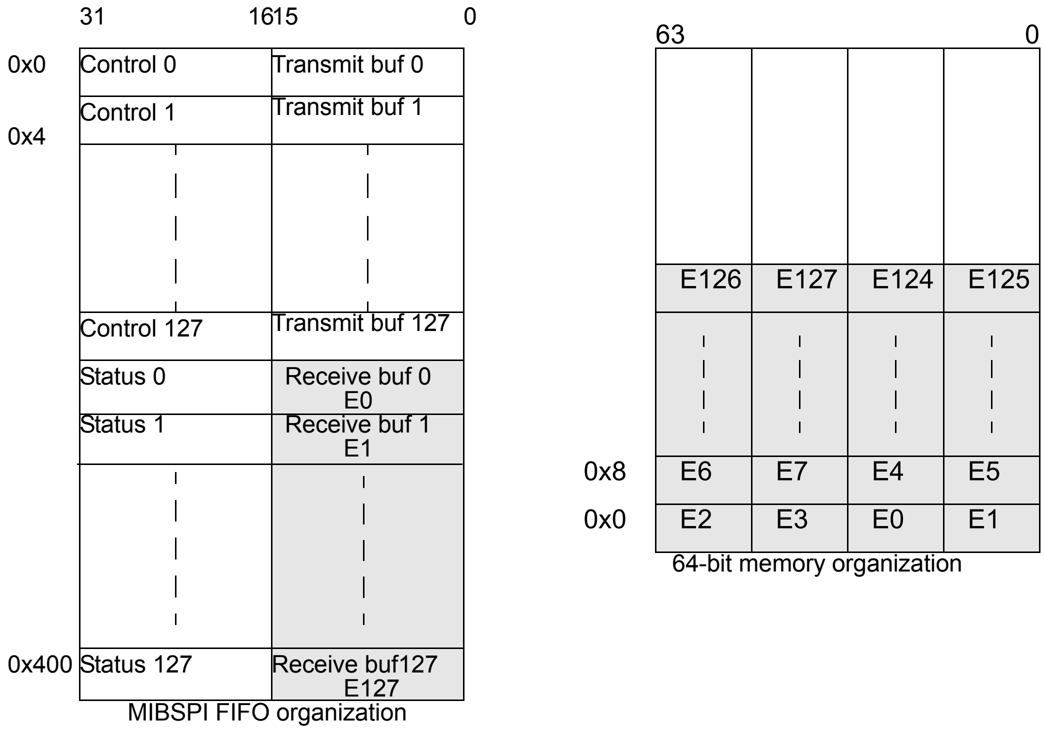
In this example, initialization of the MIBSPI FIFO is illustrated and assumes the following setup:

Read Element Size = 64 bit
Write Element Size = 16 bit
Element Count = 32
Frame Count = 1
Source Element Index = n/a, use post increment addressing mode
Source Frame Index = n/a, use post increment addressing mode
Destination Element Index = 4
Destination Frame Index = 0

When the read element size is smaller than the write element size, the DMA controller needs to perform data packing. The number of elements to pack is equal to the ratio between the write element size and read element size. In the example in Figure 20-13, the read element size is 16 bits and the write element size is 64 bits. The DMA controller would first pack the first four elements by performing four consecutive 16-bit read accesses of E0, E1, E2, and E3 into the first word of the DMA's internal FIFO. The DMA controller would then perform one single 64-bit write operation to transfer the data to the 64-bit destination memory.

Normally, the DMA controller carries out bus transactions on the bus according to the element size. For example, the DMA controller would perform a 16-bit read transaction if the read element size is programmed as 16 bits, or an 8-bit write transaction if the write element size is programmed as 8 bit. The exception is when the total transfer size is as defined in (EQ 1) is not a multiple of the write element size.

Figure 20-13. Example of DMA Data Packing



In this example, a read of the MIBSPI FIFO is illustrated and assumes the following setup: This example

Read Element Size = 16 bit
Write Element Size = 64 bit
Element Count = 128
Frame Count = 1
Source Element Index = 4
Source Frame Index = 0
Destination Element Index = n/a, use post increment addressing mode
Destination Frame Index = n/a, use post increment addressing mode

For example, if the read element size is 8 bits, the element transfer count is equal to 9, and the write element size is 64 bit. The DMA controller would first perform eight 8-bit read transactions from the source. It would then perform a 64-bit write to the destination. When the same channel wins arbitration again, the DMA controller would first perform one 8-bit read from the source, followed by one 8-bit write to the destination, even though the write element size is 64 bit.

Note:
Since peripherals are slower, it is advised to use data packing feature with caution for reading data from peripherals. Improper use might delay servicing other pending DMA channels.

20.3.7 DMA Request

There are three ways to start a DMA transfer:

- **Software request:** The transfer will be triggered by writing to SWCHENAS register. The software request can trigger either a block or a frame transfer depending on what the trigger type (TTYPE) bit is set to in the CHCTRL register.

- **Hardware request**: The DMA controller can handle up to 32 DMA Request lines. A hardware request can trigger either a frame or a block transfer dependent on the setting of TTYPE control bit of the CHCTRL register.
- **Triggered by other control packet:** When a control packet finishes the programmed number of transfers it can trigger another channel to initiate its transfers.

Each time a DMA request is made, either one frame transfer or one block transfer can be chosen. An active DMA request signal will trigger a DMA transaction.

The DMA controller has a two-level buffer to capture HW requests per channel. When a HW request is generated and the channel is enabled, the corresponding bit in the DMA status register (see DMASTAT) is set. The pending register acts as a first-level buffer. Typically, a peripheral acting as a source of a transfer would initiate another request after its data registers have been read out by DMA, even though that data has not been completely transferred to the destination. If a second HW request is generated by the peripheral, the DMA controller has an extra request buffer to capture this second request and service it after the first request is complete.

Note:
The DMA cannot capture more than three requests if its request buffers are already full. If any request occur during this moment DMA will discard it.

The DMA controller also supports a mix of hardware and software requests on the same channel. If a software request is generated, the corresponding bit in the Channel Pending register (see PEND) is set accordingly. If the pending request is not completely serviced by the DMA and a hardware request is generated by a peripheral onto the same channel, the DMA will capture and recognize this hardware request into its request buffer.

Note:
The DMA controller cannot recognize two software requests on the same channel if the first software request is still pending. If such request occur DMA will discard it. Therefore the user S/W should check the pending register before issue a new software request.

20.3.8 Auto-Initiation

When Auto-initiation Mode (AIM) bit of CHCTRL register is enabled for a channel and the channel is triggered by a software request for a block transfer, the channel will restart again using the same channel information stored at the respective control packet after one block transfer is completed. In the case of Hardware Request the channel needs to be retriggered each time after a block is complete even if auto-initiation is enabled.

20.3.9 Interrupts

Each channel can be configured to generate interrupts on several transfer conditions:

- Frame transfer complete (FTC) interrupt: an interrupt is issued after the last element of a frame has been transferred.
- Last frame transfer started (LFS) interrupt: an interrupt is issued before the first element of the last frame of a block transfer has started.
- First half of block complete (HBC) interrupt: an interrupt is issued if more than half of the block is transferred.
  – If the number of frames *n* is odd, then the HBC interrupt is generated at the beginning of the frame when *(n-1) / 2* number of frames are left in the block.
  – If the number of frames *n* is even, then the HBC interrupt is generated at the beginning of the frame after *n/2* number of frames are left in the block.

- Block transfer complete (BTC) interrupt: an interrupt is issued after the last element of the last frame has been transferred.
- Bus error (BER) interrupt: an interrupt is issued when DMA detects an error on the bus. The BER interrupt is connected to the ESM module.
- Memory Protection Unit error (MPU): an interrupt is issued when the DMA detects that the access falls outside of a memory region programmed in the MPU registers of the DMA. The MPU interrupt is connected to the ESM module.
- Parity error (PAR): an interrupt is issued when the DMA detects a parity error when reading one of the control packets. The PAR interrupt is connected to the ESM module.

The DMA outputs 5 interrupt lines for control packet handling, a parity interrupt and a memory protection interrupt(Figure 20-14). Each type of transfer interrupt condition is grouped together. For example, all block-transfer complete interrupts that are routed to a port are combined (OR'ed). The channel that caused the interrupt is given in the corresponding interrupt channel offset register. Priority between interrupts among the same interrupt type is resolved by a fixed priority scheme. Priority between different interrupt types is resolved in the Vector Interrupt Manager. Figure 20-15 explains the Frame Transfer Complete Interrupt structure in detail.

Note:
Each Channel Specific interrupts in DMA module are routed towards Group A or B to support two different CPU's individually. For devices with Single CPU or Dual CPU where both CPUs are running same code in delayed lock-step as safety feature
Group A - Interrupts (FTC, LFS, HBC, BTC and BER) are routed to the ARM CPU.
Group B - Interrupts (FTC, LFS, HBC, BTC and BER) are not routed out.
User software should configure only Group A interrupts.

Figure 20-14. DMA Interrupts

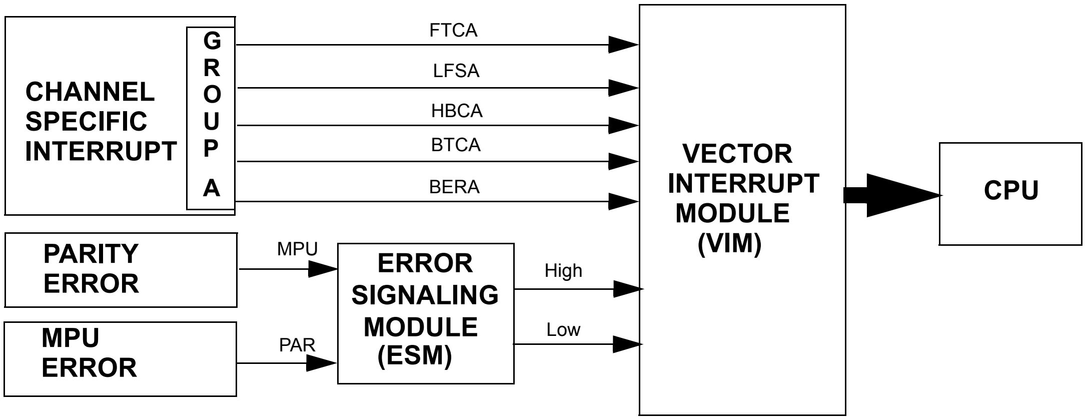
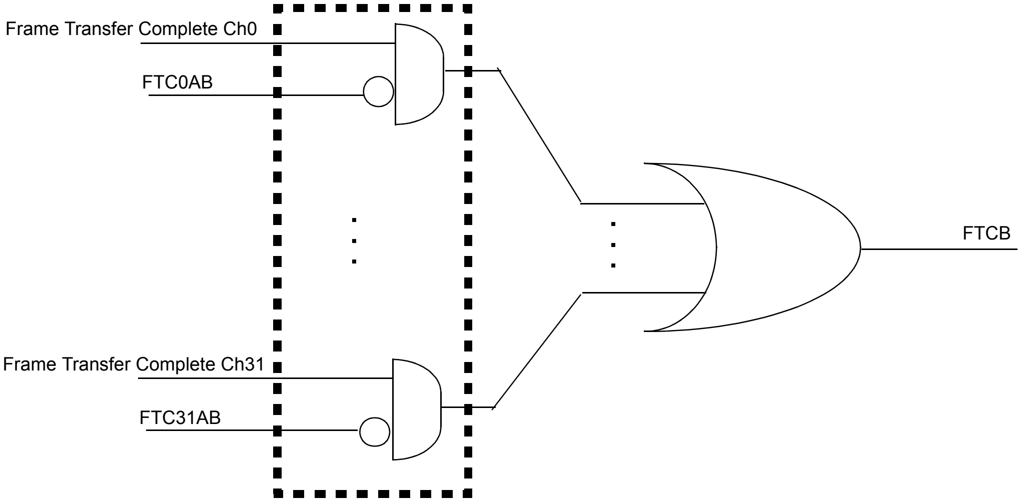

CHANNEL SPECIFIC INTERRUPT
GROUP A
FTCA
LFSA
HBCA
BTCA
BERA
VECTOR INTERRUPT MODULE (VIM)
CPU
PARITY ERROR
MPU ERROR
MPU
PAR
ERROR SIGNALING MODULE (ESM)
High
Low

Figure 20-15. Detailed Interrupt Structure (Frame Transfer Complete Path)



1 This figure is applicable for HBC, LFS, BTC, and BER interrupt

20.3.10 Debugging

The DMA supports four different behaviors in suspend mode. These behaviors can be configured by the user as per the application requirement.

- Immediate stop at a DMA channel arbitration boundary. Please refer to Table 20-2 and Table 20-3 for arbitration boundary definition.
- Finish current frame transfer and continue after suspend ends.
- Finish current block transfer and continue after suspend ends.
- Ignore the suspend. The DMA continues to be operational as in functional mode when debug mode is active.

When the DMA controller enters suspend mode, it continues to sample incoming hardware DMA requests. But the channel pending register (PEND) is frozen from being updated. After the suspend ends, all new requests that were received during suspend mode are reflected in the PEND register.

Except when the DMA controller is configured to ignore suspend mode, no channel arbitration is performed during suspend mode. The current channel under which suspend mode was entered will finish its entire frame or block-transfer after suspend mode ends, depending how the debug option was chosen.

To facilitate debugging, a watch point register (WPR) and a watch mask register (WMR) are used. The watch point register together with the watch mask register can be configured to watch for a unique address or a range of addresses. When the condition to watch is true, the DMA freezes its state and generates a debug request signal to the host CPU so the state of the DMA can be examined.

20.3.11 Power Management

The DMA offers two power-management modes: run and sleep. In run mode, the DMA is fully operational.

The sleep mode shuts down the DMA if no pending channels are waiting to be serviced. If a DMA request is received or a software request is generated by the user software, then the DMA wakes up immediately.

The sleep mode may be used to optimize the DMA module power consumption.

When the system module issues a global low power mode request, the DMA will respond to the system module with an acknowledge if no DMA requests are pending.

Note:
When the DMA is in global low power mode, the clock is stopped and therefore it cannot detect any DMA request. The device must be woken up before a peripheral can generate a DMA request.

20.3.12 FIFO Buffer

DMA FIFO is 4 levels deep and 64-bit wide (can hold upto 4 x 64-bits of data). They are used for Data packing and unpacking.

The DMA FIFO has two states:

- EMPTY : The FIFO contains no data.
- FULL : The FIFO is filled or the element count has reached zero; the read operation has to be stopped.

DMA channels can only be switched when the FIFO is empty. This also implies that arbitration between channels is done when the FIFO is empty.

The FIFO buffer may be bypassed through the use of the bypass feature in the port control register; see PTCRL for register details. Writing 1 to this bit limits the FIFO depth to the size of one element. That means after one element is read the write out to the destination will start. This feature is particularly useful to minimize switching latency in-between channels. When bypass mode is enabled, the DMA will perform minimal bus cycles on AHB bus. In addition, the bypass feature allows arbitration between channels that can be carried out at a source element granularity.

However, it has to be considered that while in bypass mode, the DMA controller does not make optimal use of the bus bandwidth. Since the read and write element sizes can be different, then the number of read and write transactions will be different. Table 20-2 and Table 20-3 show a comparison between the number of read and write transactions performed by the DMA controller from one channel to another before arbitration in non-bypass and bypass mode.

Table 20-2. Maximum Number of DMA Transactions per Channel in Non-Bypass Mode

	Write Element Size	**8 bit**		**16 bit**		**32 bit**		**64 bit**	
Read Element Size	**8 bit**	4 read	4 write	4 read	2 write	4 read	1 write	8 read	1 write
	16 bit	2 read	4 write	4 read	4 write	4 read	2 write	4 read	1 write
	32 bit	1 read	4 write	2 read	4 write	4 read	4 write	4 read	2 write
	64 bit	1 read	8 write	1 read	4 write	2 read	4 write	4 read	4 write

Table 20-3. Maximum Number of DMA Transactions per Channel in Bypass Mode

	Write Element Size	8 bit		16 bit		32 bit		64 bit	
Read Element Size	**8 bit**	1 read	1 write	2 read	1 write	4 read	1 write	8 read	1 write
	16 bit	1 read	2 write	1 read	1 write	2 read	1 write	4 read	1 write
	32 bit	1 read	4 write	1 read	2 write	1 read	1 write	2 read	1 write
	64 bit	1 read	8 write	1 read	4 write	1 read	2 write	1 read	1 write

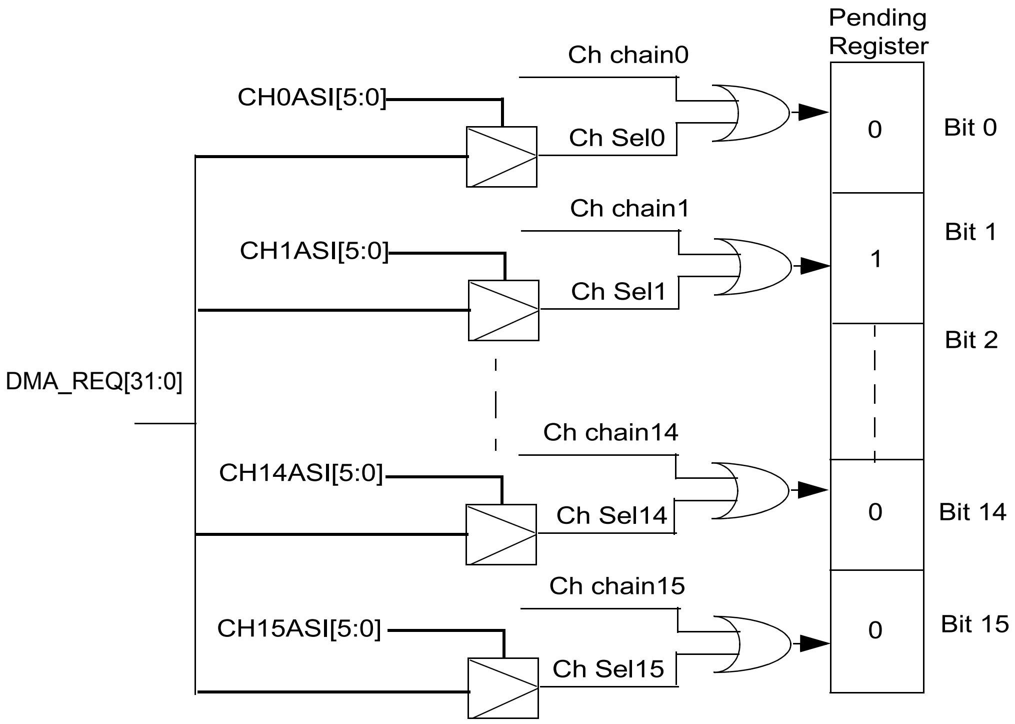
20.3.13 Channel Chaining

Channel chaining is used to trigger a single or multiple channels with out an external DMA request. This is possible by chaining one control packet to other. Chain[5:0] field of the Channel Control Register (CHCTRL) is used to program the chaining control packet. Chained control packets follow arbitration rules within the pending register. For example if CH1, CH2, CH4, CH5 are triggered together and CH3 is chained with CH1. The order of channels serviced in spite of chaining will be CH1 -> CH2 -> CH3 -> CH4 -> CH5.

In order to setup up channel chain feature the HWCHENA register needs to be enabled for all chained channels before triggering first DMA request.

Figure 20-16 illustrates how internally chained request is generated after completing the required transfers and stored in pending register. In this example CH1 is Chained to CH0. When CH0 is triggered CH1 is captured as pending in the Channel Pending Register (PEND) even when it is not triggered.

Figure 20-16. Example of Channel Chaining



20.3.14 Memory Protection

The DMA controller is capable of access to the full address range of the device. The protection mechanism allows the protection of up to four memory regions to restrict accesses to those address ranges. This will allow the application to protect critical application data from unintentionally being accessed by the DMA controller.

20.3.14.1 Protection Mechanism

The memory protection mechanism consists of the access privilege for a given memory region, the start and end address for the region, and notification of an access violation for the protected region.

Each region to be protected is configured by software by writing the start address and end address for each region into the DMA Memory Protection Registers, DMAMPRxS and DMAMPRxE. The definition of these registers can be found starting at Section 20.4.1.57. Any region in the valid address space can be protected from inappropriate accesses.

The access privileges can be set to one of four permission settings as shown below:

- Full access
- Read only access
- Write only access
- No access

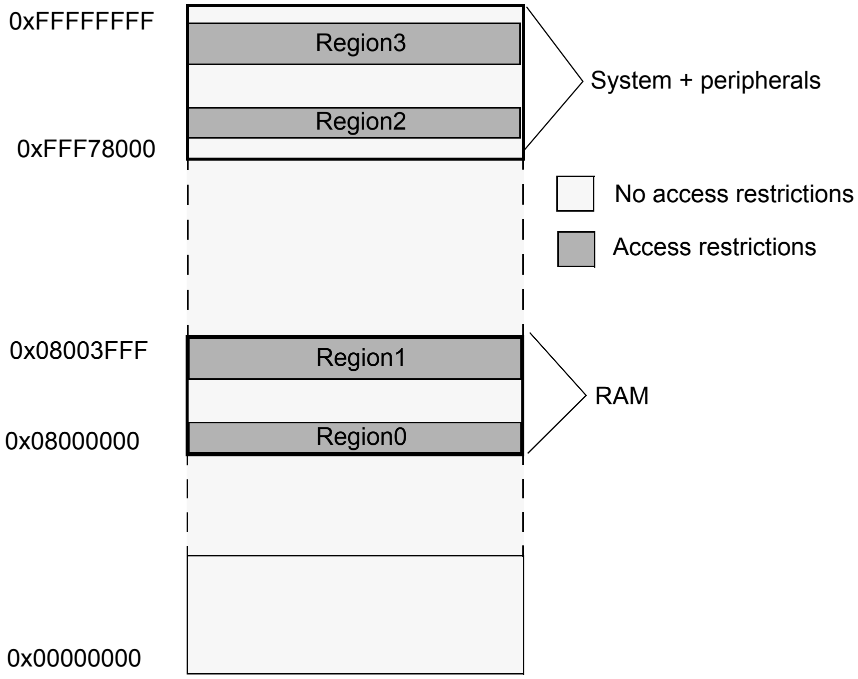
The permissions for a given region are selected by writing the appropriate values in the DMA memory protection control register (DMAMPCR).

Note:
If the regions defined by the start and end addresses overlap, the region defined first in the register space determines the access privilege. For example, if region 0 and region 1 overlap, the access permissions defined for region 0 will take precedence since region 0 registers are before region 1.

In a case where a memory protection violation occurs, a flag will be set and an interrupt will be generated, if interrupts are enabled. The DMA memory protection status register (DMAMPST) contains the status flags for the memory protection mechanism and the DMAMPCR, contains the interrupt enable bits. Upon detection of the memory protection violation, the DMA Channel that caused the violation will be stopped and the next available DMA channel will be serviced.

Figure 20-17 Illustrates a protection mechanism.

Figure 20-17. Example of Protection Mechanism



20.3.15 Parity Checking

Parity checking is implemented using parity on a per-byte basis for DMA Control Packets in the RAM. Checking for even or odd parity can be programmed by a 4-bit key located in the system module that controls the parity configuration on a global basis. This ensures that all modules using parity are acting in the same manner. The default setup after reset is odd parity.

In addition, parity checking can be enabled and disabled within the module by a 4-bit key. The key is located in the DMAPCR register.

During a read access, regardless if it was read by the DMA state machine or another master (CPU, ...), the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check, then a parity error interrupt is generated. The address that generated the error is detected and is captured for host system debugging in the register DMAPAR. The address is frozen from being updated until it is read by the bus master.

Additional error handling is dependent on the requestor.

- DMA reading from a control packet RAM: The transmission requested by DMA request will not take place.
- CPU reading from the control packet RAM: The data will be retrieved by the CPU and a parity error interrupt will be generated.

In both cases, the control packet will be left active or the DMA will be switched off dependent on the ERRA bit in the DMAPCR.

20.3.16 Parity Testing

The parity RAM is accessible to allow manually inserting faults so that the parity checking feature can be tested. Test mode is entered by asserting the TEST bit within the DMAPCR register. Once the bit is set, the parity bits are mapped to the control packet RAM starting address + 0xA00.

Note:
When in test mode, no parity checking will be done when reading from parity memory, but parity checking will be performed on the normal memory.

Each byte in Control Packet RAM has its own parity bit in the Control Packet Parity RAM as shown in Table 20-4, Table 20-5 and Table 20-6. P0 is the parity bit for byte 0, P1 is the parity bit for byte 1 and so on.

Each byte in the control packet RAM has its own parity bit in the control packet parity RAM as shown in Table 20-4 and Table 20-5.

Table 20-4. Control Packet RAM

Bit	31 24	23 16	15 8	7 0
Word0	Byte 0	Byte 1	Byte 2	Byte 3
Word1	Byte 4	Byte 5	Byte 6	Byte 7
Word2	Byte 8	Byte 9	Byte 10	Byte 11
Word3	Byte 12	Byte 13	Byte 14	Byte 15

Table 20-5. Control Packet RAM

Bit	127 96	95 64	63 32	31 0
	Word 3	Word 2	Word 1	Word 0

Table 20-6. Parity RAM

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

20.3.17 Initializing RAM with Parity

After power up, the RAM content including the parity bit cannot be guaranteed. To avoid parity failures when reading RAM, the RAM has to be initialized. The RAM can be initialized by writing known values into it. When the known value is written, the corresponding parity bit will be automatically calculated and updated.

Another possibility to initialize the memory is to set a 4-bit key in the system module. This key triggers the automatic initialization of all RAMs on the microcontroller. The RAM will be initialized to 0. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

To allow for parity calculation during initialization, the parity functionality has to be enabled as discussed in Section 20.3.15.

20.4 Control Registers and Control Packets

This section details the DMA control registers and memory map, summarized in Figure 20-18. Table 20-7 summarizes the control packets. Each register begins on a word boundary. All registers and control packets are accessible in 8, 16, and 32 bit. The base address for the control registers is 0xFFFF F000.

Note:
The register definitions are given for a full DMA module configuration (32 channels, 64 requests, 2 Ports, Dual CPU support). Writes and Reads of bits pertaining to features not included in the DMA implementation as defined in the device specific datasheet are possible without error; however, they will have no affect on device operation.

Note:
Control registers and control packets have two different base addresses

Figure 20-18. DMA Control 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
0x00 GCTRL Page 1498	Reserved															DMA_EN
	Reserved	Bus Busy	Reserved				DEBUG MODE(1–0)		Reserved							DMA RES
0x04 PEND Page 1500	Reserved															
	PEND[15:0]															
0x0C DMASTAT Page 1501	Reserved															
	STCH[15:0]															
0x14 HWCHENAS Page 1502	Reserved															
	HWCHENA[15:0]															
0x1C HWCHENAR Page 1503	Reserved															
	HWCHDIS[15:0]															
0x24 SWCHENAS Page 1504	Reserved															
	SWCHENA[15:0]															

Figure 20-18. DMA Control 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
0x2C SWCHENAR Page 1505	Reserved															
	SWCHDIS[15:0]															
0x34 CHPRIOS Page 1506	Reserved															
	CPS[15:0]															
0x3C CHPRIOR Page 1507	Reserved															
	CPR[15:0]															
0x44 GCHIENAS Page 1508	Reserved															
	GCHIE[15:0]															
0x4C GCHIENAR Page 1509	Reserved															
	GCHID[15:0]															
0x54 DREQASI0 Page 1510	Reserved		CH0ASI(5–0)						Reserved		CH1ASI(5–0)					
	Reserved		CH2ASI(5–0)						Reserved		CH3ASI(5–0)					
0x58 DREQASI1 Page 1512	Reserved		CH8ASI(5–0)						Reserved		CH9ASI(5–0)					
	Reserved		CH10ASI(5–0)						Reserved		CH11ASI(5–0)					
0x5C DREQASI2 Page 1514	Reserved		CH8ASI(5–0)						Reserved		CH9ASI(5–0)					
	Reserved		CH10ASI(5–0)						Reserved		CH11ASI(5–0)					
0x60 DREQASI3 Page 1516	Reserved		CH12ASI(5–0)						Reserved		CH13ASI(5–0)					
	Reserved		CH14ASI(5–0)						Reserved		CH15ASI(5–0)					

Figure 20-18. DMA Control 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
0x64 - 0x90 Reserved	Reserved															
	Reserved															
0x94 PAR0 Page 1518	Reserved	CH0PA(2–0)			Reserved	CH1PA(2–0)			Reserved	CH2PA(2–0)			Reserved	CH3PA(2–0)		
	Reserved	CH4PA(2–0)			Reserved	CH5PA(2–0)			Reserved	CH6PA(2–0)			Reserved	CH7PA(2–0)		
0x98 PAR1 Page 1520	Reserved	C8PA(2–0)			Reserved	C9PA(2–0)			Reserved	CH10PA(2–0)			Reserved	CH11PA(2–0)		
	Reserved	CH12PA(2–0)			Reserved	CH13PA(2–0)			Reserved	CH4PA(2–0)			Reserved	CH5PA(2–0)		
0x9C - 0xB0 Reserved	Reserved															
	Reserved															
0xB4 FTCMAP Page 1522	Reserved															
	FTCAB[15:0]															
0xBC LFSMAP Page 1523	Reserved															
	LFSAB[15:0]															
0xC4 HBCMAP Page 1524	Reserved															
	HBCAB[15:0]															
0xCC BTCMAP Page 1525	Reserved															
	BTCAB[15:0]															
0xD4 BERMAP Page 1526	Reserved															
	BERAB[15:0]															

Figure 20-18. DMA Control 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
0xDC FTCINTENAS Page 1527	Reserved															
	FTCINTENA[15:0]															
0xE4 FTCINTENAR Page 1528	Reserved															
	FTCINTDIS[15:0]															
0xEC LFSINTENAS Page 1529	Reserved															
	LFSINTENA[15:0]															
0xF4 LFSINTENAR Page 1530	Reserved															
	LFSINTDIS[15:0]															
0xFC HBCINTENAS Page 1531	Reserved															
	HBCINTENA[15:0]															
0x104 HBCINTENAR Page 1532	Reserved															
	HBCINTDIS[15:0]															
0x10C BTCINTENAS Page 1533	Reserved															
	BTCINTENA[15:0]															
0x114 BTCINTENAR Page 1534	Reserved															
	BTCINTDIS[15:0]															
0x11C GINTFLAG Page 1535	Reserved															
	GINT[15:0]															

Figure 20-18. DMA Control 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
0x124 FTCFLAG Page 1536	Reserved															
	FTCI[15:0]															
0x12C LFSFLAG Page 1537	Reserved															
	LFSI[15:0]															
0x134 HBCFLAG Page 1538	Reserved															
	HBCI[15:0]															
0x13C BTCFLAG Page 1539	Reserved															
	BTCI[15:0]															
0x144 BERFLAG Page 1540	Reserved															
	BERI[15:0]															
0x14C FTCAOFFSET Page 1541	Reserved															
	Reserved								sbz	sbz	FTCA(5–0)					
0x150 LFSAOFFSET Page 1542																
	Reserved								sbz	sbz	LFSA(5–0)					
0x154 HBCAOFFSET Page 1543																
	Reserved								sbz	sbz	HBCA(5–0)					
0x158 BTCAOFFSET Page 1544																
	Reserved								sbz	sbz	BTCA(5–0)					

Figure 20-18. DMA Control 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
0x15C BERAOFFSET Page 1545																
	Reserved								sbz	sbz	BERA(5–0)					
0x160 FTCBOFFSET Page 1546																
	Reserved								sbz	sbz	FTCB(5–0)					
0x164 LFSBOFFSET Page 1547																
	Reserved								sbz	sbz	LFSB(5–0)					
0x168 HBCBOFFSET Page 1548																
	Reserved								sbz	sbz	HBCB(5–0)					
0x16C BTCBOFFSET Page 1549																
	Reserved								sbz	sbz	BTCB(5–0)					
0x170 BERBOFFSET Page 1550																
	Reserved								sbz	sbz	BERB(5–0)					
0x178 PTCRL Page 1551	Reserved							PEND B	Reserved					BYB	PSFRH QPB	PSFR-LQPB
	Reserved							PEND A	Reserved					BYA	PSFRH QPA	PSFR-LQPA
0x17C RTCTRL Page 1553																
	Reserved															RTC
0x180 DCTRL Page 1554	Reserved			CHNUM(4–0)					Reserved							DMAD-BGS
	Reserved															DBGEN

Figure 20-18. DMA Control Registers

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184 WPR Page 1556	WP[31:16]															
	WP[15:0]															
0x188 WMR Page 1557	WM[31:16]															
	WM[15:0]															
0x198 PBACSADDR Page 1558	PBACSA(31–16)															
	PBACSA(15–0)															
0x19C PBACDADDR Page 1559	PBACDA(31–16)															
	PBACDA(15–0)															
0x1A0 PBACTC Page 1560	Reserved			PBFTCOUNT(12–0)												
	Reserved			PBETCOUNT(12–0)												
0x1A8 DMAPCR Page 1561	Reserved															ERRA
	Reserved							TEST	Reserved				PARITY_ENA(3–0)			
0x1AC DMAPAR Page 1563	Reserved							EDFLAG	Reserved							
	Reserved				ERRORADDRESS(11–0)											
0x1B0 DMAMPCTRL Page 1564	Reserved			INT3AB	INT3 ENA	REG3AP(1–0)		REG3 ENA	Reserved			INT2AB	INT2 ENA	REG2AP(1–0)		REG2 ENA
	Reserved			INT1AB	INT1 ENA	REG1AP(1–0)		REG1 ENA	Reserved			INT0AB	INT0 ENA	REG0AP(1–0)		REG0 ENA
0x1B4 DMAMPST Page 1567	Reserved							REG3FT	Reserved							REG2FT
	Reserved							REG1FT	Reserved							REG0FT

Figure 20-18. DMA Control 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
0x1B8 DMAMPROS Page 1569	STARTADDRESS(31–16)															
	STARTADDRESS(15–0)															
0x1BC DMAMPR0E Page 1570	ENDADDRESS(31–16)															
	ENDADDRESS(15–0)															
0x1C0 DMAMPR1S Page 1571	STARTADDRESS(31–16)															
	STARTADDRESS(15–0)															
0x1C4 DMAMPR1E Page 1572	ENDADDRESS(31–16)															
	ENDADDRESS(15–0)															
0x1C8 DMAMPR2S Page 1573	STARTADDRESS(31–16)															
	STARTADDRESS(15–0)															
0x1CC DMAMPR2E Page 1574	ENDADDRESS(31–16)															
	ENDADDRESS(15–0)															
0x1D0 DMAMPR3S Page 1575	STARTADDRESS(31–16)															
	STARTADDRESS(15–0)															
0x1D4 DMAMPR3E Page 1576	ENDADDRESS(31–16)															
	ENDADDRESS(15–0)															

Table 20-7. Control Packet Memory Map

Address Offset	Mnemonic	Name	Description
0x00 Page 1577	ISADDR	INITIAL SOURCE ADDRESS Register	PRIMARY CONTROL PACKET0
0x04 Page 1578	IDADDR	INITIAL DESTINATION ADDRESS Register	
0x08 Page 1579	ITCOUNT	INITIAL TRANSFER COUNT Register	
0x0C	res	RESERVED	
0x10 Page 1580	CHCTRL	CHANNEL CONTROL Register	
0x14 Page 1582	EIOFF	ELEMENT INDEX OFFSET Register	
0x18 Page 1583	FIOFF	FRAME INDEX OFFSET Register	
0x1C	res	RESERVED	
0x20			PRIMARY CONTROL PACKET1
0x40			PRIMARY CONTROL PACKET2
0x60			PRIMARY CONTROL PACKET3
0x80			PRIMARY CONTROL PACKET4
0xA0			PRIMARY CONTROL PACKET5
0xC0			PRIMARY CONTROL PACKET6
0xE0			PRIMARY CONTROL PACKET7
0x100			PRIMARY CONTROL PACKET8
0x120			PRIMARY CONTROL PACKET9
0x140			PRIMARY CONTROL PACKET10
0x160			PRIMARY CONTROL PACKET11
0x180			PRIMARY CONTROL PACKET12
0x1A0			PRIMARY CONTROL PACKET13
0x1C0			PRIMARY CONTROL PACKET14
0x1E0			PRIMARY CONTROL PACKET15
0x200-0x7FF	res	RESERVED	reserved location

Table 20-7. Control Packet Memory Map

Address Offset	Mnemonic	Name	Description
0x800 Page 1584	CSADDR	CURRENT SOURCE ADDRESS Register	WORKING CONTROL PACKET0 Writing to any working control packet will assert DMA_AERROR
0x804 Page 1585	CDADDR	CURRENT DESTINATION ADDRESS Register	
0x808 Page 1586	CTCOUNT	CURRENT TRANSFER COUNT Register	
0x80C	res	RESERVED	
0x810			WORKING CONTROL PACKET1
0x820			WORKING CONTROL PACKET2
0x830			WORKING CONTROL PACKET3
0x840			WORKING CONTROL PACKET4
0x850			WORKING CONTROL PACKET5
0x860			WORKING CONTROL PACKET6
0x870			WORKING CONTROL PACKET7
0x880			WORKING CONTROL PACKET8
0x890			WORKING CONTROL PACKET09
0x8A0			WORKING CONTROL PACKET10
0x8B0			WORKING CONTROL PACKET11
0x8C0			WORKING CONTROL PACKET12
0x8D0			WORKING CONTROL PACKET13
0x8E0			WORKING CONTROL PACKET14
0x8F0			WORKING CONTROL PACKET15
0x900-0xFFF	res	RESERVED	reserved location

20.4.1 Global Configuration Registers

These registers control the overall behavior of the DMA controller.

20.4.1.1 Global Control Register (GCTRL)

Figure 20-19 and Table 20-8 describe this register.

Figure 20-19. Global Control Register (GCTRL) [offset = 0x00]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Bus Busy	Reserved				DEBUG MODE(1–0)		Reserved							DMA RES
R-0						R/WP-0		R-0							R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-8. Global Control Register (GCTRL) [offset = 0x00] Field Descriptions

Bit	Name	Value	Description
31–17	Reserved		Read returns 0. Writes have no effect.
16	DMA_EN		DMA enable bit. The configuration registers and channel control packets should be setup first before DMA_EN bit is set to one to prevent state machines from carrying out bus transactions. If DMA_EN bit is cleared in the middle of an bus transaction, the state machine will stop at an arbitration boundary.
		0	The DMA is disabled.
		1	The DMA is enabled.
15	Reserved		Read returns 0. Writes have no effect.
14	BUS BUSY		This bit indicates status of DMA external AHB bus status.
		0	DMA's external bus is not busy in data transfers.
		1	DMA's external bus is busy in data transfers.
13-10	Reserved		Read returns 0. Writes have no effect.

Table 20-8. Global Control Register (GCTRL) [offset = 0x00] Field Descriptions (Continued)

Bit	Name	Value	Description
9-8	DEBUG MODE		
		00	Ignore suspend.
		01	Finish current block transfer.
		10	Finish current frame transfer.
		11	Immediately stop at an DMA arbitration boundary and continue after suspend.
7-1	Reserved		Read returns 0. Writes have no effect.
0	DMA RES		DMA software reset
		0	Software reset is disabled.
		1	The DMA state machine and all control registers are in software reset. Control packets are not reset when DMA software reset is active. **Note:** In the event a DMA slave does not respond, the DMA module will respond to the software reset upon reaching an arbitration boundary.

20.4.1.2 Channel Pending Register (PEND)

Figure 20-20 and Table 20-9 describe this register.

Figure 20-20. Channel Pending Register (PEND) [offset = 0x04]

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
PEND[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-9. Channel Pending Register (PEND) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	PEND		Channel pending register. Reading from PEND gives the channel pending information no matter if the channel was initiated by SW or HW. Once set, it remains set even if the corresponding channel is disabled via HWCHENA or SWCHENA. The pending bit is automatically cleared for the following conditions: 1. At the end of a frame or a block transfer depending on how the channel is triggered as programmed in the TTYPE bit field of CHCTRL. 2. The control packet is modified after the pending bit is set. 3. An AHB bus error occurs.
		0	The corresponding channel is inactive.
		1	The corresponding channel is pending and is waiting for service.

20.4.1.3 DMA Status Register (DMASTAT)

Figure 20-21 and Table 20-10 describe this register.

Figure 20-21. DMA Status Register (DMASTAT) [offset = 0x0C]

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
STCH[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-10. DMA Status Register (DMASTAT) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	STCH		Status of DMA channels.
		0	The channel is inactive.
		1	The channel is active; that is, the channel is currently in the DMA's execution queue). **Note: The status of a channel currently in DMA's execution queue remains active even if emulation mode is entered or DMA is disabled via DMA_EN bit.**

20.4.1.4 HW Channel Enable Set and Status Register (HWCHENAS)

Figure 20-22 and Table 20-11 describe this register.

Figure 20-22. HW Channel Enable Set and Status Register (HWCHENAS) [offset = 0x14]

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
HWCHENA[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-11. HW Channel Enable Set and Status Register (HWCHENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	HWCHENA[15:0]		Hardware channel enable bit. An active hardware DMA request cannot initiate a DMA transfer unless the corresponding hardware enable bit is set. The corresponding hardware enable bit is cleared automatically for the following conditions: • At the end of a block transfer if the auto-initiation bit AIM (see CHCTRL) is not active. • If an AHB bus error is detected for an active channel. Reading from HWCHENAS gives the status (enabled/disabled) of channels 0 through 15.
		0	The corresponding channel is disabled for hardware triggering.
		1	The corresponding channel is enabled for hardware triggering.

20.4.1.5 HW Channel Enable Reset and Status Register (HWCHENAR)

Figure 20-23 and Table 20-12 describe this register.

Figure 20-23. HW Channel Enable Reset and Status Register (HWCHENAR) [offset = 0x1C]

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
HWCHDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-12. HW Channel Enable Reset and Status Register (HWCHENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	HWCHDIS[15:0]		HW channel disable bit.
		0	*Read:* The corresponding channel is disabled for HW triggering.
			Write: A write of zero to this bit has no effect.
		1	*Read:* The corresponding channel is enabled for HW triggering.
			Write: The corresponding channel is disabled.

20.4.1.6 SW Channel Enable Set and Status Register (SWCHENAS)

Figure 20-24 and Table 20-13 describe this register.

Figure 20-24. SW Channel Enable Set and Status Register (SWCHENAS) [offset = 0x24]

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
SWCHENA[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-13. SW Channel Enable Set and Status Register (SWCHENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	SWCHENA[15:0]		SW channel enable bit. Writing a one to a bit triggers a SW request on the corresponding channel to start a DMA transaction. The corresponding bit is automatically cleared by the following conditions. 1. The corresponding bit is cleared after one frame transfer if the TTYPE bit in CHCTRL is programmed for frame transfer. 2. The corresponding bit is cleared after one block transfer if the corresponding TTYPE bit is programmed for block transfer and the auto-initiation bit is not enabled. 3. The control packet is modified after the pending bit is set. 4. The corresponding bit is cleared after one block transfer when TTYPE bit is programmed for blocks transfer and if the corresponding bit in HW channel enable register (HWCHENAS) is enabled. When a channel is enabled for both HW and SW, the state machine will initiate transfers based on the SW first. After one block transfer is complete, the corresponding bit in the SWCHENA register is then cleared. The same channel is serviced again by a HW DMA request. 5. The corresponding bit is cleared if an AHB bus error is detected. Reading from SWCHENAS gives the status (enabled/disabled) of channels 0 through 15.
		0	The corresponding channel is not triggered by SW request.
		1	The corresponding channel is triggered by SW request.

20.4.1.7 SW Channel Enable Reset and Status Register (SWCHENAR)

Figure 20-25 and Table 20-14 describe this register.

Figure 20-25. SW Channel Enable Reset and Status Register (SWCHENAR) [offset = 0x2C]

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
SWCHDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-14. SW Channel Enable Reset and Status Register (SWCHENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	SWCHDIS[15:0]		SW channel disable bit.
		0	*Read:* The corresponding channel was not triggered by SW. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The corresponding channel was triggered by SW. *Write:* The corresponding channel is disabled.

20.4.1.8 Channel Priority Set Register (CHPRIOS)

Figure 20-26 and Table 20-15 describe this register.

Figure 20-26. Channel Priority Set Register (CHPRIOS) [offset = 0x34]

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
CPS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-15. Channel Priority Set Register (CHPRIOS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CPH[15:0]		Channel priority set bit. Writing a one to a bit assigns the corresponding channel to the high priority queue.
		0	*Read:* The corresponding channel is assigned to the low priority queue. *Write:* Writing a zero to this bit has no effect.
		1	*Read and write:* The corresponding channel is assigned to high priority queue.

20.4.1.9 Channel Priority Reset Register (CHPRIOR)

Figure 20-27 and Table 20-16 describe this register.

Figure 20-27. Channel Priority Reset Register (CHPRIOR) [offset = 0x3C]

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
CPR[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-16. Channel Priority Reset Register (CHPRIOR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	CPR[15:0]		Channel priority reset bit. Writing a one to a bit assigns the according channel to the low priority queue.
		0	*Read:* The corresponding channel is assigned to the low priority queue. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The corresponding channel is assigned to the high priority queue. *Write:* The corresponding channel is assigned to the low priority queue.

20.4.1.10 Global Channel Interrupt Enable Set Register (GCHIENAS)

Figure 20-28 and Table 20-17 describe this register.

Figure 20-28. Global Channel Interrupt Enable Set Register (GCHIENAS) [offset = 0x44]

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
GCHIE[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-17. Global Channel Interrupt Enable Set Register (GCHIENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	GCHIE[15:0]		Global channel interrupt enable bit.
		0	*Read:* The corresponding channel is disabled for interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read and write:* The corresponding channel is enabled for interrupt.

20.4.1.11 Global Channel Interrupt Enable Reset Register (GCHIENAR)

Figure 20-29 and Table 20-18 describe this register.

Figure 20-29. Global Channel Interrupt Enable Reset Register (GCHIENAR) [offset = 0x4C]

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
GCHID[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-18. Global Channel Interrupt Enable Reset Register (GCHIENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	GCHID		Global channel interrupt disable bit.
		0	*Read:* The corresponding channel is disabled for interrupt. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The corresponding channel is enabled for interrupt. *Write:* The corresponding channel is disabled for interrupt.

20.4.1.12 DMA Request Assignment Register 0 (DREQASI0)

This register assigns the DMA requests to the channels 0–3. Figure 20-30 and Table 20-19 describe this register.

Figure 20-30. DMA Request Assignment Register 0 (DREQASI0) [offset = 0x54]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		CH0ASI(5–0)						Reserved		CH1ASI(5–0)					
R-0		R/WP-0x0						R-0		R/WP-0x1					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH2ASI(5–0)						Reserved		CH3ASI(5–0)					
R-0		R/WP-0x2						R-0		R/WP-0x3					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-19. DMA Request Assignment Register 0 (DREQASI0) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Read returns 0. Writes have no effect.
29–24	CH0ASI(5–0)		Channel 0 assignment. This bit field chooses the DMA request assignment for channel 0.
		0	DMA request line 0 triggers channel 0.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 0.
23–22	Reserved		Read returns 0. Writes have no effect.
21–16	CH1ASI(5–0)		Channel 1 assignment. This bit field chooses the DMA request assignment for channel 1.
		0	DMA request line 0 triggers channel 1.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 1.
15–14	Reserved		Read returns 0. Writes have no effect.
13–8	CH2ASI(5–0)		Channel 2 assignment. This bit field chooses the DMA request assignment for channel 2.
		0	DMA request line 0 triggers channel 2.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 2.

Table 20-19. DMA Request Assignment Register 0 (DREQASI0) Field Descriptions (Continued)

Bit	Name	Value	Description
7–6	Reserved		Read returns 0. Writes have no effect.
5–0	CH3ASI(5–0)		Channel 3 assignment. This bit field chooses the DMA request assignment for channel 3.
		0	DMA request line 0 triggers channel 3.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 3.

20.4.1.13 DMA Request Assignment Register 1 (DREQASI1)

This register assigns the DMA requests to the channels 4–7. Figure 20-31 and Table 20-20 describe this register.

Figure 20-31. DMA Request Assignment Register 1 (DREQASI1) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		CH4ASI(5–0)						Reserved		CH5ASI(5–0)					
R-0		R/WP-0x4						R-0		R/WP-0x5					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH6ASI(5–0)						Reserved		CH7ASI(5–0)					
R-0		R/WP-0x6						R-0		R/WP-0x7					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-20. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Read returns 0. Writes have no effect.
29–24	CH4ASI(5–0)		Channel 4 assignment. This bit field chooses the DMA request assignment for channel 4.
		0	DMA request line 0 triggers channel 4.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 4.
23–22	Reserved		Read returns 0. Writes have no effect.
21–16	CH5ASI(5–0)		Channel 5 assignment. This bit field chooses the DMA request assignment for channel 5.
		0	DMA request line 0 triggers channel 5.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 5.
15–14	Reserved		Read returns 0. Writes have no effect.
13–8	CH6ASI(5–0)		Channel 6 assignment. This bit field chooses the DMA request assignment for channel 6.
		0	DMA request line 0 triggers channel 6.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 6.

Table 20-20. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions (Continued)

Bit	Name	Value	Description
7–6	Reserved		Read returns 0. Writes have no effect.
5–0	CH7ASI(5–0)		Channel 7 assignment. This bit field chooses the DMA request assignment for channel 7.
		0	DMA request line 0 triggers channel 7.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 7.

20.4.1.14 DMA Request Assignment Register 2 (DREQASI2)

This register assigns the DMA requests to the channels 8–11. Figure 20-32 and Table 20-21 describe this register.

Figure 20-32. DMA Request Assignment Register 2 (DREQASI2) [offset = 0x5C]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="6">CH8ASI(5–0)</td><td colspan="2">Reserved</td><td colspan="6">CH9ASI(5–0)</td></tr>
<tr><td colspan="2">R-0</td><td colspan="6">R/WP-0x8</td><td colspan="2">R-0</td><td colspan="6">R/WP-0x9</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="6">CH10ASI(5–0)</td><td colspan="2">Reserved</td><td colspan="6">CH11ASI(5–0)</td></tr>
<tr><td colspan="2">R-0</td><td colspan="6">R/WP-0xA</td><td colspan="2">R-0</td><td colspan="6">R/WP-0xB</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-21. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Read returns 0. Writes have no effect.
29–24	CH8ASI(5–0)		Channel 8 assignment. This bit field chooses the DMA request assignment for channel 8.
		0	DMA request line 0 triggers channel 8.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 8.
23–22	Reserved		Read returns 0. Writes have no effect.
21–16	CH9ASI(5–0)		Channel 9 assignment. This bit field chooses the DMA request assignment for channel 9.
		0	DMA request line 0 triggers channel 9.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 9.
15–14	Reserved		Read returns 0. Writes have no effect.
13–8	CH10ASI(5–0)		Channel 10 assignment. This bit field chooses the DMA request assignment for channel 10.
		0	DMA request line 0 triggers channel 10.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 10.

Table 20-21. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions (Continued)

Bit	Name	Value	Description
7–6	Reserved		Read returns 0. Writes have no effect.
5–0	CH11ASI(5–0)		Channel 11 assignment. This bit field chooses the DMA request assignment for channel 11.
		0	DMA request line 0 triggers channel 11.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 11.

20.4.1.15 DMA Request Assignment Register 3 (DREQASI3)

This register assigns the DMA requests to the channels 12–15.Figure 20-33 and Table 20-22 describe this register.

Figure 20-33. DMA Request Assignment Register 3 (DREQASI3) [offset = 0x60]

31–30	29–24	23–22	21–16
Reserved	CH12ASI(5–0)	Reserved	CH13ASI(5–0)
R-0	R/WP-0xC	R-0	R/WP-0xD

15–14	13–8	7–6	5–0
Reserved	CH14ASI(5–0)	Reserved	CH15ASI(5–0)
R-0	R/WP-0xE	R-0	R/WP-0xF

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-22. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions

Bit	Name	Value	Description
31–30	Reserved		Read returns 0. Writes have no effect.
29–24	CH12ASI(5–0)		Channel 12 assignment. This bit field chooses the DMA request assignment for channel 12.
		0	DMA request line 0 triggers channel 12.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 12.
23–22	Reserved		Read returns 0. Writes have no effect.
21–16	CH13ASI(5–0)		Channel 13 assignment. This bit field chooses the DMA request assignment for channel 13.
		0	DMA request line 0 triggers channel 13.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 13.
15–14	Reserved		Read returns 0. Writes have no effect.
13–8	CH14ASI(5–0)		Channel 14 assignment. This bit field chooses the DMA request assignment for channel 14.
		0	DMA request line 0 triggers channel 14.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 14.

Table 20-22. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions (Continued)

Bit	Name	Value	Description
7–6	Reserved		Read returns 0. Writes have no effect.
5–0	CH15ASI(5–0)		Channel 15 assignment. This bit field chooses the DMA request assignment for channel 15.
		0	DMA request line 0 triggers channel 15.
		. . .	. . .
		1Fh	DMA request line 31 triggers channel 15.

20.4.1.16 Port Assignment Register 0 (PAR0)

Figure 20-34 and Table 20-23 describe this register.

Figure 20-34. Port Assignment Register 0 (PAR0) [offset = 0x94]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CH0PA(2–0)			Reserved	CH1PA(2–0)			Reserved	CH2PA(2–0)			Reserved	CH3PA(2–0)		
R-0	R/WP-0			R-0	R/WP-0			R-0	R/WP-0			R-0	R/WP-0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CH4PA(2–0)			Reserved	CH5PA(2–0)			Reserved	CH6PA(2–0)			Reserved	CH7PA(2–0)		
R-0	R/WP-0			R-0	R/WP-0			R-0	R/WP-0			R-0	R/WP-0		

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-23. Port Assignment Register 0 (PAR0) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return zeros and writes have no effect.
30–28	CH0PA(2–0)		These bit fields determine to which port channel 0 is assigned.
		1xx	Port B
		0xx	Reserved
27	Reserved		Reads return zeros and writes have no effect.
26–24	CH1PA(2–0)		These bit fields determine to which port channel 1 is assigned.
		1xx	Port B
		0xx	Reserved
23	Reserved		Reads return zeros and writes have no effect.
22–20	CH2PA(2–0)		These bit fields determine to which port channel 2 is assigned.
		1xx	Port B
		0xx	Reserved
19	Reserved		Reads return zeros and writes have no effect.
18–16	CH3PA		These bit fields determine to which port channel 3 is assigned.
		1xx	Port B
		0xx	Reserved
15	Reserved		Reads return zeros and writes have no effect.

Table 20-23. Port Assignment Register 0 (PAR0) Field Descriptions (Continued)

Bit	Name	Value	Description
14–12	CH4PA(2–0)		These bit fields determine to which port channel 4 is assigned.
		1xx	Port B
		0xx	Reserved
11	Reserved		Reads return zeros and writes have no effect.
10–8	CH5PA(2–0)		These bit fields determine to which port channel 5 is assigned.
		1xx	Port B
		0xx	Reserved
7	Reserved		Reads return zeros and writes have no effect.
6–4	CH6PA(2–0)		These bit fields determine to which port channel 6 is assigned.
		1xx	Port B
		0xx	Reserved
3	Reserved		Reads return zeros and writes have no effect.
2–0	CH7PA(2–0)		These bit fields determine to which port channel 7 is assigned.
		1xx	Port B
		0xx	Reserved

20.4.1.17 Port Assignment Register 1 (PAR1)

Figure 20-35 and Table 20-24 describe this register.

Figure 20-35. Port Assignment Register 1 (PAR1) [offset = 0x98]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	C8PA(2–0)			Reserved	C9PA(2–0)			Reserved	CH10PA(2–0)			Reserved	CH11PA(2–0)		
R–0	R/WP–0			R–0	R/WP–0			R–0	R/WP–0			R–0	R/WP–0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CH12PA(2–0)			Reserved	CH13PA(2–0)			Reserved	CH4PA(2–0)			Reserved	CH5PA(2–0)		
R–0	R/WP–0			R–0	R/WP–0			R–0	R/WP–0			R-0	R/WP-0		

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-24. Port Assignment Register 1 (PAR1) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return zeros and writes have no effect.
30–28	CH8PA(2–0)		These bit fields determine to which port channel 8 is assigned.
		1xx	Port B
		0xx	Reserved
27	Reserved		Reads return zeros and writes have no effect.
26–24	CH9PA(2–0)		These bit fields determine to which port channel 9 is assigned.
		1xx	Port B
		0xx	Reserved
23	Reserved		Reads return zeros and writes have no effect.
22–20	CH10PA(2–0)		These bit fields determine to which port channel 10 is assigned.
		1xx	Port B
		0xx	Reserved
19	Reserved		Reads return zeros and writes have no effect.
18–16	CH11PA(2–0)		These bit fields determine to which port channel 11 is assigned.
		1xx	Port B
		0xx	Reserved
15	Reserved		Reads return zeros and writes have no effect.

Table 20-24. Port Assignment Register 1 (PAR1) Field Descriptions (Continued)

Bit	Name	Value	Description
14–12	CH12PA(2–0)		These bit fields determine to which port channel 12 is assigned.
		1xx	Port B
		0xx	Reserved
11	Reserved		Reads return zeros and writes have no effect.
10–8	CH13PA(2–0)		These bit fields determine to which port channel 13 is assigned.
		1xx	Port B
		0xx	Reserved
7	Reserved		Reads return zeros and writes have no effect.
6–4	CH14PA(2–0)		These bit fields determine to which port channel 14 is assigned.
		1xx	Port B
		0xx	Reserved
3	Reserved		Reads return zeros and writes have no effect.
2–0	CH15PA(2–0)		These bit fields determine to which port channel 15 is assigned.
		1xx	Port B
		0xx	Reserved

20.4.1.18 FTC Interrupt Mapping Register (FTCMAP)

Figure 20-36 and Table 20-25 describe this register.

Note:
On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-36. FTC Interrupt Mapping Register (FTCMAP) [offset = 0xB4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">FTCAB[15:0]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-25. FTC Interrupt Mapping Register (FTCMAP) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	FTCAB[31:0]		Frame transfer complete (FTC) interrupt to Group A or Group B.
		0	The FTC interrupt of the corresponding channel is routed to Group A.
		1	The FTC interrupt of the corresponding channel is routed to Group B.

20.4.1.19 LFS Interrupt Mapping Register (LFSMAP)

Figure 20-37 and Table 20-26 describe this register.

Note:
On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-37. LFS Interrupt Mapping Register (LFSMAP) [offset = 0xBC]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">LFSAB[15:0]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-26. LFS Interrupt Mapping Register (LFSMAP) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	LFSAB[15:0]		Last frame started (LFS) interrupt to Group A or Group B.
		0	The LFS interrupt of the corresponding channel is routed to Group A.
		1	The LFS interrupt of the corresponding channel is routed to Group B.

20.4.1.20 HBC Interrupt Mapping Register (HBCMAP)

Figure 20-38 and Table 20-27 describe this register.

> **Note:**
> On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-38. HBC Interrupt Mapping Register (HBCMAP) [offset = 0xC4]

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
HBCAB[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-27. HBC Interrupt Mapping Register (HBCMAP) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	HBCAB[15:0]		Half block complete (HBC) interrupt to Group A or Group B.
		0	The HBC interrupt of the corresponding channel is routed to Group A.
		1	The HBC interrupt of the corresponding channel is routed to Group B.

20.4.1.21 BTC Interrupt Mapping Register (BTCMAP)

Figure 20-39 and Table 20-28 describe this register.

Note:
On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-39. BTC Interrupt Mapping Register (BTCMAP) [offset = 0xCC]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">BTCAB[15:0]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-28. BTC Interrupt Mapping Register (BTCMAP) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BTCAB[15:0]		Block transfer complete (BTC) interrupt to Group A or Group B
		0	The BTC interrupt of the corresponding channel is routed to Group A.
		1	The BTC interrupt of the corresponding channel is routed to Group B.

20.4.1.22 BER Interrupt Mapping Register (BERMAP)

Figure 20-40 and Table 20-29 describe this register.

> **Note:**
> On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-40. BER Interrupt Mapping Register (BERMAP) [offset = 0xD4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">BERAB[15:0]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-29. BER Interrupt Mapping Register (BERMAP) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BERAB[15:0]		Bus error (BER) interrupt to Group A or Group B.
		0	The BER interrupt of the corresponding channel is routed to Group A.
		1	The BER interrupt of the corresponding channel is routed to Group B.

20.4.1.23 FTC Interrupt Enable Set (FTCINTENAS)

Figure 20-41 and Table 20-30 describe this register.

Figure 20-41. FTC Interrupt Enable Set (FTCINTENAS) [offset = 0xDC]

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
FTCINTENA[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-30. FTC Interrupt Enable Set (FTCINTENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	FTCINTENA[15:0]		Frame transfer complete (FTC) interrupt enable.
		0	*Read:* The corresponding FTC interrupt of a channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The FTC interrupt of the corresponding channel is enabled.

20.4.1.24 FTC Interrupt Enable Reset (FTCINTENAR)

Figure 20-42 and Table 20-31 describe this register.

> **Note:**
> On this device Group B Interrupts are not Implemented hence user software should configure only Group A interrupts.

Figure 20-42. FTC Interrupt Enable Reset (FTCINTENAR) [offset = 0xE4]

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
FTCINTDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-31. FTC Interrupt Enable Reset (FTCINTENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	FTCINTDIS		Frame transfer complete (FTC) interrupt disable.
		0	*Read:* The corresponding FTC interrupt of a channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The corresponding FTC interrupt of a channel is enabled. *Write:* The corresponding FTC interrupt is disabled.

20.4.1.25 LFS Interrupt Enable Set (LFSINTENAS)

Figure 20-43 and Table 20-32 describe this register.

Figure 20-43. LFS Interrupt Enable Set (LFSINTENAS) [offset = 0xEC]

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
LFSINTENA[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-32. LFS Interrupt Enable Set (LFSINTENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	LFSINTENA		Last frame started (LFS) interrupt enable.
		0	*Read:* The corresponding LFS interrupt of a channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The LFS interrupt of the corresponding channel is disabled.

20.4.1.26 LFS Interrupt Enable Reset (LFSINTENAR)

Figure 20-44 and Table 20-33 describe this register.

Figure 20-44. LFS Interrupt Enable Reset (LFSINTENAR) [offset = 0xF4]

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
LFSINTDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-33. LFS Interrupt Enable Reset (LFSINTENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	LFSINTDIS[15:0]		Last frame started (LFS) interrupt disable.
		0	*Read:* The LFS interrupt of the corresponding channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The LFS interrupt of the corresponding channel is enabled. *Write:* The LFS interrupt of the corresponding channel is disabled.

20.4.1.27 HBC Interrupt Enable Reset (HBCINTENAS)

Figure 20-45 and Table 20-34 describe this register.

Figure 20-45. HBC Interrupt Enable Set (HBCINTENAS) [offset = 0xFC]

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
HBCINTENA[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-34. HBC Interrupt Enable Set (HBCINTENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	HBCINTENA[15:0]		Half block complete (HBC) interrupt enable.
		0	*Read:* The HBC interrupt of the corresponding channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The HBC interrupt of the corresponding channel is enabled.

20.4.1.28 HBC Interrupt Enable Reset (HBCINTENAR)

Figure 20-46 and Table 20-35 describe this register.

Figure 20-46. HBC Interrupt Enable Reset (HBCINTENAR) [offset = 0x104]

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
HBCINTDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-35. HBC Interrupt Enable Reset (HBCINTENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	HBCINTDIS[15:0]		Half block complete (HBC) interrupt disable.
		0	*Read:* The HBC interrupt of the corresponding channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The HBC interrupt of the corresponding channel is enabled. *or write:* The HBC interrupt of the corresponding channel is disabled.

20.4.1.29 BTC Interrupt Enable Set (BTCINTENAS)

Figure 20-47 and Table 20-36 describe this register.

Figure 20-47. BTC Interrupt Enable Set (BTCINTENAS) [offset = 0x10C]

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
BTCINTENA[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-36. BTC Interrupt Enable Reset (BTCINTENAS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BTCINTENA[15:0]		Block transfer complete (BTC) interrupt enable.
		0	*Read:* The BTC interrupt of the corresponding channel is disabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read or write:* The BTC interrupt of the corresponding channel is enabled.

20.4.1.30 BTC Interrupt Enable Reset (BTCINTENAR)

Figure 20-48 and Table 20-37 describe this register.

Figure 20-48. BTC Interrupt Enable Reset (BTCINTENAR) [offset = 0x114]

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
BTCINTDIS[15:0]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-37. BTC Interrupt Enable Reset (BTCINTENAR) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BTCINTDIS[15:0]		Block transfer complete (BTC) interrupt disable.
		0	*Read:* The BTC interrupt of the corresponding channel is enabled. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The BTC interrupt of the corresponding channel is disabled. *Write:* The BTC interrupt of the corresponding channel is disabled.

20.4.1.31 Global Interrupt Flag Register (GINTFLAG)

Figure 20-49 and Table 20-38 describe this register.

Figure 20-49. Global Interrupt Flag Register (GINTFLAG) [offset = 0x11C]

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
GINT[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-38. Global Interrupt Flag Register (GINTFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	GINT[15:0]		Global interrupt flags. A global interrupt flag bit is an OR function of FTC, LFS, HBC, BTC and BER interrupt flags.
		0	No interrupt is pending on the corresponding channel.
		1	One or more of the five interrupt types (FTC, LFS, HBC, BTC, or BER) is pending on the corresponding channel.

20.4.1.32 FTC Interrupt Flag Register (FTCFLAG)

Figure 20-50 and Table 20-39 describe this register.

Figure 20-50. FTC Interrupt Flag Register (FTCFLAG) [offset = 0x124]

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
FTCI[15:0]															
R/WP-0															

R = Read in all modesWP = Write 1 in privilege mode to clear the bit only, -*n* = Value after reset

.

Table 20-39. FTC INTERRUPT FLAG Register (FTCFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	FTCI[15:0]		Frame transfer complete (FTC) flags. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see FTCAOFFSET and FTCBOFFSET).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	*Read:* An FTC interrupt of the corresponding channel is not pending. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* An FTC interrupt of the corresponding channel is pending. *Write:* The flag is cleared.

20.4.1.33 LFS Interrupt Flag Register (LFSFLAG)

Figure 20-51 and Table 20-40 describe this register.

Figure 20-51. LFS Interrupt Flag Register (LFSFLAG) [offset = 0x12C]

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
LFSI[15:0]															
R/WP-0															

R = Read in all modesWP = Write 1 in privilege mode to clear the bit only, -*n* = Value after reset

.

Table 20-40. LFS Interrupt Flag Register (LFSFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	LFSI		Last frame started (LFS) flags. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see LFSAOFFSET and LFSBOFFSET).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	*Read:* An LFS interrupt of the corresponding channel is not pending. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* An LFS interrupt of the corresponding channel is pending. *Write:* The flag is cleared.

20.4.1.34 HBC Interrupt Flag Register (HBCFLAG)

Figure 20-52 and Table 20-41 describe this register.

Figure 20-52. HBC Interrupt Flag Register (HBCFLAG) [offset = 0x134]

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
HBCI[15:0]															
R/WP-0															

R = Read in all modesWP = Write 1 in privilege mode to clear the bit only, -*n* = Value after reset

.

Table 20-41. HBC Interrupt Flag (HBCFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	HBCI[15:0]		Half block transfer (HBC) complete flags. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see HBCAOFFSET and HBCBOFFSET).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	*Read:* An HBC interrupt of the corresponding channel is not pending. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* An HBC interrupt of the corresponding channel is pending. *Write:* The flag is cleared.

20.4.1.35 BTC Interrupt Flag Register (BTCFLAG)

Figure 20-53 and Table 20-42 describe this register.

Figure 20-53. BTC Interrupt Flag Register (BTCFLAG) [offset = 0x13C]

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
BTCI[15:0]															
R/WP-0															

R = Read in all modesWP = Write 1 in privilege mode to clear the bit only, -*n* = Value after reset

.

Table 20-42. BTC Interrupt Flag Register (BTCFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BTCI[15:0]		Block transfer complete (BTC) flags. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see BTCAOFFSET and BTCBOFFSET).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	*Read:* An BTC interrupt of the corresponding channel is not pending. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* An BTC interrupt of the corresponding channel is pending. *Write:* The flag is cleared.

20.4.1.36 BER Interrupt Flag Register (BERFLAG)

Figure 20-54 and Table 20-43 describe this register.

Figure 20-54. BER Interrupt Flag Register (BERFLAG) [offset = 0x144]

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
BERI[15:0]															
R-0															

R = Read in all modesWP = Write 1 in privilege mode to clear the bit only, -*n* = Value after reset

.

Table 20-43. BER Interrupt Flag Register (BERFLAG) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15–0	BERI[15:0]		Bus error (BER) flags. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see BERAOFFSET and BERBOFFSET).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	*Read:* An BER interrupt of the corresponding channel is not pending. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* An BER interrupt of the corresponding channel is pending. *Write:* The flag is cleared.

20.4.1.37 FTCA Interrupt Channel Offset Register (FTCAOFFSET)

Figure 20-55 and Table 20-44 describe this register.

Figure 20-55. FTCA Interrupt Channel Offset Register (FTCAOFFSET) [offset = 0x14C]

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
Reserved								sbz	sbz	FTCA(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-44. FTCA Interrupt Channel Offset Register (FTCAOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7–6	sbz		These bits should always be written as zero.
5–0	FTCA(5–0)		Channel causing FTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see FTCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group A.

20.4.1.38 LFSA Interrupt Channel Offset Register (LFSAOFFSET)

Figure 20-56 and Table 20-45 describe this register.

Figure 20-56. LFSA Interrupt Channel Offset Register (LFSAOFFSET) [offset = 0x150]

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
Reserved								sbz	sbz	LFSA(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-45. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7–6	sbz		These bits should always be programmed as zero.
5–0	LFSA(5–0)		Channel causing LFS interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see LFSFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group A.

20.4.1.39 HBCA Interrupt Channel Offset Register (HBCAOFFSET)

Figure 20-57 and Table 20-46 describe this register.

Figure 20-57. HBCA Interrupt Channel Offset Register (HBCAOFFSET) [offset = 0x154]

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
Reserved								sbz	sbz	HBCA(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-46. HBCA Interrupt Channel Offset Register (HBCAOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7–6	sbz		These bits should always be programmed as zero.
5–0	HBCA(5–0)		Channel causing HBC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see HBCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group A.

20.4.1.40 BTCA Interrupt Channel Offset Register (BTCAOFFSET)

Figure 20-58 and Table 20-47 describe this register.

Figure 20-58. BTCA Interrupt Channel Offset Register (BTCAOFFSET) [offset = 0x158]

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
Reserved								sbz	sbz	BTCA(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-47. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect
7–6	sbz		These bits should always be programmed as zero.
5–0	BTCA(5–0)		Channel causing BTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see BTCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group A.

20.4.1.41 BERA Interrupt Channel Offset Register (BERAOFFSET)

Figure 20-59 and Table 20-48 describe this register.

Figure 20-59. BERA Interrupt Channel Offset Register (BERAOFFSET) [offset = 0x15C]

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
Reserved								sbz	sbz	BERA(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-48. BERA Interrupt Channel Offset Register (BERAOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect
7–6	sbz		These bits should always be programmed as zero.
5–0	BERA(5–0)		Channel causing BER interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see BERFLAG) with the highest priority. Please refer to Table 20-52 to see the interrupt offset index.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group A.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group A.

20.4.1.42 FTCB Interrupt Channel Offset Register (FTCBOFFSET)

Figure 20-60 and Table 20-49 describe this register.

Figure 20-60. FTCB Interrupt Channel Offset Register (FTCBOFFSET) [offset = 0x160]

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
Reserved								sbz	sbz	FTCB(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-49. FTCB Interrupt Channel Offset Register (FTCBOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect
7–6	sbz		These bits should always be programmed as zero.
5–0	FTCB(5–0)		Channel causing FTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see FTCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group B.

20.4.1.43 LFSB Interrupt Channel Offset Register (LFSBOFFSET)

Figure 20-61 and Table 20-50 describe this register.

Figure 20-61. LFSB Interrupt Channel Offset Register (LFSBOFFSET) [offset = 0x164]

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
Reserved								sbz	sbz	LFSB(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-50. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect
7–6	sbz		These bits should always be programmed as zero.
5–0	LFSB(5–0)		Channel causing LFS interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see LFSFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group B.

20.4.1.44 HBCB Interrupt Channel Offset Register (HBCBOFFSET)

Figure 20-62 and Table 20-51 describe this register.

Figure 20-62. HBCB Interrupt Channel Offset Register (HBCBOFFSET) [offset = 0x168]

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
Reserved								sbz	sbz	HBCB(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-51. HBCB Interrupt Channel Offset Register (HBCBOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect
7–6	sbz		These bits should always be programmed as zero.
5–0	HBCB(5–0)		Channel causing HBC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see HBCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group B.

20.4.1.45 BTCB Interrupt Channel Offset Register (BTCBOFFSET)

Figure 20-63 and Table 20-52 describe this register.

Figure 20-63. BTCB Interrupt Channel Offset Register (BTCBOFFSET) [offset = 0x16C]

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
Reserved								sbz	sbz	BTCB(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-52. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7–6	sbz		These bits should always be programmed as zero.
5–0	BTCB(5–0)		Channel causing BTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see BTCFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group B.

20.4.1.46 BERB Interrupt Channel Offset Register (BERBOFFSET)

Figure 20-64 and Table 20-53 describe this register.

Figure 20-64. BERB Interrupt Channel Offset Register (BERBOFFSET) [offset = 0x170]

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
Reserved								sbz	sbz	BERB(5–0)					
R-0								R-0	R-0	R-0					

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-53. BERB Interrupt Channel Offset Register (BERBOFFSET) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7–6	sbz		These bits should always be programmed as zero.
5–0	BERB(5–0)		Channel causing BER interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see BERFLAG) with the highest priority.**
		0	No interrupt is pending.
		1	Channel 0 is causing the pending interrupt Group B.
		. . .	. . .
		10h	Channel 15 is causing the pending interrupt Group B.

20.4.1.47 Port Control Register (PTCRL)

Figure 20-65 and Table 20-54 describe this register.

Figure 20-65. Port Control Register (PTCRL) [offset = 0x178]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							PEND B	Reserved					BYB	PSFRH QPB	PSFR-LQPB
R-0							R-0	R-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
Reserved															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-54. Port Control Register (PTCRL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Read returns 0. Writes have no effect.
24	PENDB		Transfers pending for port B. This flag determines if transfers are ongoing on port B. The flag will be cleared if no transfers are performed. It can be used to determine if there is still data transferred while DMA_EN is set to 0 in GCTCRL. In this case, once all transfers are finished, the flag will be set to 0.
		0	No transfers are pending.
		1	Transfers are pending.
23-19	Reserved		Read returns 0. Writes have no effect.
18	BYB		Bypass FIFO B.
		0	FIFO B is not bypassed.
		1	FIFO B is bypassed. Writing 1 to this bit limits the FIFO depth to the size of one element. That means that after one element is read, the write-out to the destination will begin. This feature is particularly useful to minimize switching latency between channels. **Note: This feature does not make optimal use of bus bandwidth.**
17	PSFRHQPB		Priority scheme fix or rotate for high priority queue of Port B.
		0	Fixed priority is used.
		1	Rotation priority is used.

Table 20-54. Port Control Register (PTCRL) Field Descriptions (Continued)

Bit	Name	Value	Description
16	PSFRLQPB		Priority scheme fix or rotate for low priority queue of Port B.
		0	The fixed priority scheme is used.
		1	The rotation priority scheme is used.
15–0	Reserved		Read returns 0. Writes have no effect.

20.4.1.48 RAM Test Control (RTCTRL)

Figure 20-66 and Table 20-55 describe this register.

Figure 20-66. RAM Test Control (RTCTRL) [offset = 0x17C]

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
Reserved															RTC
R-0															R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-55. RAM Test Control (RTCTRL) [offset = 0x17C] Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Read returns 0. Writes have no effect.
0	RTC		RAM test control. Writing a one to this bit opens the write access to the reserved locations of control packet RAM as defined in the memory map. **Note: This bit should be set to zero during normal operation.**
		0	RAM Test Control Disabled.
		1	RAM Test Control Enabled..

20.4.1.49 Debug Control (DCTRL)

Figure 20-67 and Table 20-56 describe this register.

Figure 20-67. Debug Control (DCTRL) [offset = 0x180]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CHNUM(4–0)					Reserved							DMAD-BGS
R-0			R-0					R-0							R/C-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															DBGEN
R-0															R/C-0

R = Read; W = Write;C = Clear; *-n* = Value after reset

Table 20-56. Debug Control (DCTRL) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Read returns 0. Writes have no effect.
28–24	CHNUM(4–0)	0–1Fh	Channel number. This bit field indicates the channel number that causes the watchpoint to match.
23–17	Reserved		Read returns 0. Writes have no effect.
16	DMADBGS		DMA debug status. When a watchpoint is set up to watch for a unique bus address or a range of addresses is true on one of the three bus ports, then the DMA debug status bit is set to 1 and a debug request signal is asserted to the main CPU. The CPU must write a 1 to clear this bit for the DMA controller to release the debug request signal.
		0	*Read:* No watchpoint condition is detected. *Write:* Writing a zero to this bit has no effect.
		1	*Read:* The watchpoint condition is detected. *Write:* The bit is cleared.
15–1	Reserved		Read returns 0. Writes have no effect.

Table 20-56. Debug Control (DCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
0	DBGEN		Debug enable. **Note: This bit can only be set when using a debugger.** **Note: This bit is reset when Test reset (nTRST) is low.**
		0	Debug is disabled.
		1	The watchpoint checking logics is enabled.

20.4.1.50 Watch Point Register (WPR)

Figure 20-68 and Table 20-57 describe this register.

Figure 20-68. Watch Point Register (WPR) [offset = 0x184]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">WP[31:16]</td></tr>
<tr><td colspan="16">R/W-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">WP[15:0]</td></tr>
<tr><td colspan="16">R/W-0</td></tr>
</table>

R = Read; W = Write; *-n* = Value after test reset

Table 20-57. Watch Point Register (WPR) Field Descriptions

Bit	Name	Value	Description
31–0	WP(31–0)	0–FFFF FFFFh	Watch point. **Note: These bits can only be set when using a debugger.** This register is only reset by a test reset (nTRST). A 32-bit address can be programmed into this register as a watchpoint. This register is used with the watch mask register (WMR). When the DBGEN bit in the DCTRL register is set and a unique address or a range of addresses are detected on the AHB address bus of Port B, a debug request signal is sent to the ARM CPU. The state machine of the port in which the watchpoint condition is true is frozen.

20.4.1.51 Watch Mask Register (WMR)

Figure 20-69 and Table 20-58 describe this register.

Figure 20-69. Watch Mask Register (WMR) [offset = 0x188]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WM[31:16]															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WM[15:0]															
RW-0															

R = Read; W = Write; -*n* = Value after test reset

.

Table 20-58. Watch Mask Register (WMR) Field Descriptions

Bit	Name	Value	Description
31–0	WM[31:0]		Watch mask. **Note: These bits can only be set when using a debugger.** This register is only reset by a test reset (nTRST).
		0	Setting a bit to 0 allows the bit in the WPR register to be used for address matching for a watch point.
		1	Setting a bit to 1 in the WMR register masks the corresponding bit in the WPR register and is disregarded in the comparison.

20.4.1.52 Port B Active Channel Source Address Register (PBACSADDR)

Figure 20-70 and Table 20-59 describe this register.

Figure 20-70. Port B Active Channel Source Address Register (PBACSADDR) [offset = 0x198]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBACSA(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PBACSA(15–0)															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-59. Port B Active Channel Source Address Register (PBACSADDR) Field Descriptions

Bit	Name	Value	Description
31–0	PBACSA(31–0)	0–FFFF FFFFh	Port B active channel source address. This register contains the current source address of the active channel as broadcast in DMASTAT register for Port B.

20.4.1.53 Port B Active Channel Destination Address Register (PBACDADDR)

Figure 20-71 and Table 20-60 describe this register.

Figure 20-71. Port B Active Channel Destination Address Register (PBACDADDR) [offset = 0x19C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PBACDA(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PBACDA(15–0)															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-60. Port B Active Channel Destination Address Register (PBACDADDR) Field Descriptions

Bit	Name	Value	Description
31–0	PBACDA(31–0)	0–FFFF FFFFh	Port B active channel destination address. This register contains the current destination address of the active channel as broadcast in DMASTAT register for Port B.

20.4.1.54 Port B Active Channel Transfer Count Register (PBACTC)

Figure 20-72 and Table 20-61 describe this register.

Figure 20-72. Port B Active Channel Transfer Count Register (PBACTC) [offset = 0x1A0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			PBFTCOUNT(12–0)												
R-0			R-0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PBETCOUNT(12–0)												
R-0			R-0												

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-61. Port B Active Channel Transfer Count Register (PBACTC) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Read returns 0. Writes have no effect.
28–16	PBFTCOUNT(12–0)	0–1FFFh	Port B active channel frame count. These bits contain the current frame count value of the active channel as broadcast in DMASTAT register for Port B.
15–13	Reserved		Read returns 0. Writes have no effect.
12–0	PBETCOUNT(12–0)	0–1FFFh	Port B active channel element count. This register contains the current element count value of the active channel as broadcast in DMASTAT register for Port B.

20.4.1.55 Parity Control Register (DMAPCR)

Figure 20-73 and Table 20-62 describe this register.

Figure 20-73. Parity Control Register (DMAPCR) [offset = 0x1A8]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TEST	Reserved				PARITY_ENA(3–0)			
R-0							R/WP-0	R-0				R/WP-0101			

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-62. Parity Control Register (DMAPCR) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved		Read returns 0. Writes have no effect.
16	ERRA		Error action.
		0	If a parity error is detected on control packet x (x = 0, 1, ... n), then the enable/disable state of control packet x remains unchanged.
		1	If a parity error is detected on control packet x (x = 0, 1, ...n), then the DMA controller is disabled immediately. If a frame on control packet x is processed at the time the parity error is detected, then remaining elements of this frame will not be transferred anymore. The DMA will be disabled regardless of whether the error was detected during a read to the control packet RAM performed by the DMA state machine or by a different master.
15–9	Reserved		Read returns 0. Writes have no effect.
8	TEST		When this bit is set, the parity bits are memory mapped to make them accessible by the CPU.
		0	The parity bits are not memory mapped.
		1	The parity bits are memory mapped.
7–4	Reserved		Read returns 0. Writes have no effect.

Table 20-62. Parity Control Register (DMAPCR) Field Descriptions (Continued)

Bit	Name	Value	Description
3–0	PARITY_ENA(3–0)		Parity error detection enable. 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 DMA_UERR signal is activated.
		0101	The parity check is disabled.
		All other values	The parity check is enabled. **Note: It is recommended to write a 1010 to enable parity check, to guard against soft error from flipping PARITY_ENA to a disable state..**

20.4.1.56 DMA Parity Error Address Register (DMAPAR)

Figure 20-74 and Table 20-63 describe this register.

Figure 20-74. DMA Parity Error Address Register (DMAPAR) [offset = 0x1AC]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ERRORADDRESS(11–0)											
R-0				R-X											

R = Read in all modes; WP = Write in privilege mode only; X = Undefined; -n = Value after reset

.

Table 20-63. DMA Parity Error Address Register (DMAPAR) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Read returns 0. Writes have no effect.
24	EDFLAG		Parity Error Detection Flag. This flag indicates if a parity error occurred on reading DMA Control packet RAM.
		0	*Read :* No error occurred. *Write :* No effect
		1	*Read :* Error detected and the address is captured in DMAPAR's ERROR_ADDRESS field *Write :* Clears the bit.
23–12	Reserved		Read returns 0. Writes have no effect.
11–0	ERROR ADDRESS(11–0)	0–FFFh	Error address. These bits hold the address of the first parity error generated in the RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode when SUSPEND is high, this address is frozen even when read. **Note: The error address register will not be reset by PORRST nor by any other reset source.**

20.4.1.57 DMA Memory Protection Control Register (DMAMPCTRL)

Figure 20-75 and Table 20-64 describe this register.

Figure 20-75. DMA Memory Protection Control Register (DMAMPCTRL) [offset = 0x1B0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			INT3AB	INT3 ENA	REG3AP(1–0)		REG3 ENA	Reserved			INT2AB	INT2 ENA	REG2AP(1–0)		REG2 ENA
R-0			R/WP-0	R/WP-0	R/WP-0		R/WP-0	R-0			R/WP-0	R/WP-0	R/WP-00		R/WP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			INT1AB	INT1 ENA	REG1AP(1–0)		REG1 ENA	Reserved			INT0AB	INT0 ENA	REG0AP(1–0)		REG0 ENA
R-0			R/WP-0	R/WP-0	R/WP-0		R/WP-0	R-0			R/WP-0	R/WP-0	R/WP-00		R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-64. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads return zeros and writes have no effect.
28	INT3AB		Interrupt assignment of region 3 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
27	INT3ENA		Interrupt enable of region 3.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
26–25	REG3AP(1–0)		Region 3 access permission. These bits determine the access permission for region 3
		00	All accesses are allowed.
		01	Read only accesses are allowed.
		10	Write only accesses are allowed.
		11	No accesses are allowed.
24	REG3ENA		Region 3 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
23–21	Reserved		Reads return zeros and writes have no effect.

Table 20-64. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
20	INT2AB		Interrupt assignment of region 2 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
19	INT2ENA		Interrupt enable of region 2.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
18–17	REG2AP(1–0)		Region 2 access permission. These bits determine the access permission for region 2.
		00	All accesses are allowed.
		01	Read only accesses are allowed.
		10	Write only accesses are allowed.
		11	No accesses are allowed.
16	REG2ENA		Region 2 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
15–13	Reserved		Reads return zeros and writes have no effect.
12	INT1AB		Interrupt assignment of region 1 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
11	INT1ENA		Interrupt enable of region 1.
		0	The interrupt is disabled.
		1	The interrupt is enabled.

Table 20-64. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
10–9	REG1AP(1–0)		Region 1 access permission. These bits determine the access permission for region 3
		00	All accesses are allowed.
		01	Read only accesses are allowed.
		10	Write only accesses are allowed.
		11	No accesses are allowed.
8	REG1ENA		Region 1 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
7–5	Reserved		Reads return zeros and writes have no effect.
4	INT0AB		Interrupt assignment of region 0 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the DSP CPU (Group B).
3	INT0ENA		Interrupt enable of region 0.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
2–1	REG0AP(1–0)		Region 0 access permission. These bits determine the access permission for region 0.
		00	All accesses are allowed.
		01	Read only accesses are allowed.
		10	Write only accesses are allowed.
		11	No accesses are allowed.
0	REG0ENA		Region 0 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).

20.4.1.58 DMA Memory Protection Status Register (DMAMPST)

Figure 20-76 and Table 20-65 describe this register.

Figure 20-76. DMA Memory Protection Status Register (DMAMPST) [offset = 0x1B4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							REG3FT	Reserved							REG2FT
R-0							RC-0	R-0							RC-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							REG1FT	Reserved							REG0FT
R-0							RC-0	R-0							RC-0

R = Read in all modes; C = Clear; -n = Value after reset

.

Table 20-65. DMA Memory Protection Status Register (DMAMPST) [offset = 0x1B4] Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zeros and writes have no effect.
24	REG3FT		Region 3 fault. This bit determines whether an access permission violation was detected in this region.
		0	No fault was detected.
		1	*Read:* A fault was detected. *Write:* The bit is cleared.
23–17	Reserved		Reads return zeros and writes have no effect.
16	REG2FT		Region 2 fault. This bit determines whether a access permission violation was detected in this region.
		0	No fault was detected.
		1	*Read:* A fault was detected. *Write:* The bit is cleared.
15–9	Reserved		Reads return zeros and writes have no effect.
8	REG1FT		Region 1 fault. This bit determines whether an access permission violation was detected in this region.
		0	No fault was detected.
		1	*Read:* A fault was detected. *Write:* The bit is cleared.
7–1	Reserved		Reads return zeros and writes have no effect.

Table 20-65. DMA Memory Protection Status Register (DMAMPST) [offset = 0x1B4] Field Descriptions

Bit	Name	Value	Description
0	REG0FT		Region 0 fault. This bit determines whether a access permission violation was detected in this region.
		0	No fault was detected.
		1	*Read:* A fault was detected. *Write:* The bit is cleared.

20.4.1.59 DMA Memory Protection Region 0 Start Address Register (DMAMPR0S)

Figure 20-77 and Table 20-66 describe this register.

Figure 20-77. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) [offset = 0x1B8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-66. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) Field Descriptions

Bit	Name	Value	Description
31–0	START ADDRESS(31–0)	0–FFFF FFFFh	Start Address defines the address at which the region begins.

20.4.1.60 DMA Memory Protection Region 0 End Address Register (DMAMPR0E)

Figure 20-78 and Table 20-67 describe this register.

Figure 20-78. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) [offset = 0x1BC]

31–16
ENDADDRESS(31–16)
R/WP-0

15–0
ENDADDRESS(15–0)
R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-67. DMA Memory Protection Region 0 End Address Register (DMAMPR0E)] Field Descriptions

Bit	Name	Value	Description
31–0	END ADDRESS(31–0)	0–FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space.

20.4.1.61 DMA Memory Protection Region 1 Start Address Register (DMAMPR1S)

Figure 20-79 and Table 20-68 describe this register.

Figure 20-79. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) [offset = 0x1C0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 20-68. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) Field Descriptions

Bit	Name	Value	Description
31–0	START ADDRESS(31–0)	0–FFFF FFFFh	Start Address defines the address at which the region begins.

20.4.1.62 DMA Memory Protection Region 1 End Address Register (DMAMPR1E)

Figure 20-80 and Table 20-69 describe this register.

Figure 20-80. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) [offset = 0x1C4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">ENDADDRESS(31–16)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">ENDADDRESS(15–0)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-69. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) Field Descriptions

Bit	Name	Value	Description
31–0	END ADDRESS(31–0)	0–FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise the region will wrap around at the end of the address space.

20.4.1.63 DMA Memory Protection Region 2 Start Address Register (DMAMPR2S)

Figure 20-81 and Table 20-70 describe this register.

Figure 20-81. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) [offset = 0x1C8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-70. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) Field Descriptions

Bit	Name	Value	Description
31–0	START ADDRESS(31–0)	0–FFFF FFFFh	Start Address defines the address at which the region begins.

20.4.1.64 DMA Memory Protection Region 2 End Address Register (DMAMPR2E)

Figure 20-82 and Table 20-71 describe this register.

Figure 20-82. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) [offset = 0x1CC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENDADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-71. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) Field Descriptions

Bit	Name	Value	Description
31–0	END ADDRESS(31–0)	0–FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise the region will wrap around at the end of the address space.

20.4.1.65 DMA Memory Protection Region 3 Start Address Register (DMAMPR3S

Figure 20-83 and Table 20-72 describe this register.

Figure 20-83. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) [offset = 0x1D0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-72. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) Field Descriptions

Bit	Name	Value	Description
31–0	START ADDRESS(31–0)	0–FFFF FFFFh	Start Address defines the address at which the region begins.

20.4.1.66 DMA Memory Protection Region 3 End Address Register (DMAMPR3E)

Figure 20-84 and Table 20-73 describe this register.

Figure 20-84. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) [offset = 0x1D4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENDADDRESS(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDADDRESS(15–0)															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 20-73. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) Field Descriptions

Bit	Name	Value	Description
31–0	END ADDRESS(31–0)	0–FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space.

20.4.2 Channel Configuration

The channel configuration is defined by the channel control packet: channel control, transfer count, index pointers, source/destination address).

- It is stored in local RAM, which is protected by parity.
- Each control packet contains a total of nine fields.
- The first six fields are programmable, while the last three fields are read only.
- The RAM is accessible by queue A and queue B state machines as well as CPU.
- When there are simultaneous accesses, the priority is resolved in a fixed priority scheme with the CPU having the highest priority.

All the control packets look the same. Following, there is the detailed layout of these registers shown for control packet 0.

20.4.2.1 Initial Source Address (ISADDR)

Figure 20-85 and Table 20-74 describe this register.

Figure 20-85. Initial Source Address (ISADDR) [offset = 0x000]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISADDR(31–16)															
R/WP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISADDR(15–0)															
R/WP-X															

R = Read in all modes; WP = Write in privilege mode only; -X = Unknown; -n = Value after reset

.

Table 20-74. Initial Source Address (ISADDR) Field Descriptions

Bit	Name	Value	Description
31–0	ISADDR(31–0)	0–FFFF FFFFh	Initial source address. These bits gives the absolute 32-bit source address (physical).

20.4.2.2 Initial Destination Address Register (IDADDR)

Figure 20-86 and Table 20-75 describe this register.

Figure 20-86. Initial Destination Address Register (IDADDR) [offset = 0x004]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDADDR(31–16)															
RWP-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDADDR(15–0)															
RWP-X															

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-75. Initial Destination Address Register (IDADDR) Field Descriptions

Bit	Name	Value	Description
31–0	IDADDR(31–0)	0–FFFF FFFFh	Initial destination address. These bits give the absolute 32-bit destination address (physical).

20.4.2.3 *Initial Transfer Count Register (ITCOUNT)*

Figure 20-87 and Table 20-76 describe this register.

Figure 20-87. Initial Transfer Count Register (ITCOUNT) [offset = 0x008]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			IFTCOUNT(12–0)												
R-X			RWP-X												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			IETCOUNT(12–0)												
R-X			RWP-X												

R = Read in all modes; WP = Write in privilege mode only; -X = Unknown; -n = Value after reset

.

Table 20-76. Initial Transfer Count Register (ITCOUNT) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads are undefined and writes have no effect.
28-16	IFTCOUNT(12–0)	0–1FFFh	Initial frame transfer count. These bits define the number of frame transfers.
15-13	Reserved		Reads are undefined and writes have no effect.
12-0	IETCOUNT(12–0)	0–1FFFh	Initial element transfer count. These bits define the number of element transfers. The block transfer size will be IETCOUNT x IFTCOUNT

20.4.2.4 Channel Control Register (CHCTRL)

Figure 20-88 and Table 20-77 describe this register.

Figure 20-88. Channel Control Register (CHCTRL) [offset = 0x010]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										CHAIN(5–0)					
R-X										R/WP-X					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES(1–0)		WES(1–0)		Reserved			TTYPE	Reserved			ADDMR(1–0)		ADDMW(1–0)		AIM
R/WP-X		R/WP-X		R-X			R/WP-X	R-X			R/WP-X		R/WP-X		R/WP-X

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

Table 20-77. CHANNEL CONTROL Register (CHCTRL) Field Descriptions

Bit	Name	Value	Description
31–22	Reserved		Reads are undefined and writes have no effect.
21–16	CHAIN(5–0)		Next channel to be triggered. At the end of the programmed number of frames, the specified channel will be triggered. **Note: The programmer must program the CHAIN bits before initiating a DMA transfer.**
		000000	No channel is selected.
		000001	Channel 0 is selected.
		. . .	. . .
		010000	Channel 15 is selected.
		010001–111111	Reserved
15–14	RES(1–0)		Read element size.
		00	The element is byte, 8-bit.
		01	The element is half-word, 16-bit.
		10	The element is word, 32-bit.
		11	The element is double-word, 64-bit.

Table 20-77. CHANNEL CONTROL Register (CHCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
13–12	WES(1–0)		Write element size.
		00	The element is byte, 8-bit.
		01	The element is half-word, 16-bit.
		10	The element is word, 32-bit.
		11	The element is double-word, 64-bit.
11-9	Reserved		Reads are undefined and writes have no effect.
8	TTYPE		Transfer type.
		0	A hardware request triggers one frame transfer.
		1	A hardware request triggers one block transfer.
7-5	Reserved		Reads are undefined and writes have no effect.
4–3	ADDMR(1–0)		Addressing mode read.
		00	Constant
		01	Post-increment
		10	Reserved
		11	Indexed
2–1	ADDMW(1–0)		Addressing mode write.
		00	Constant
		01	Post-increment
		10	Reserved
		11	Indexed
0	AIM		Auto-initiation mode.
		0	Auto-initiation mode is enabled.
		1	Single block transfer mode is enabled.

20.4.2.5 Element Index Offset Register (EIOFF)

Figure 20-89 and Table 20-78 describe this register.

Figure 20-89. Element Index Offset Register (EIOFF) [offset = 0x014]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			EIDXD(12–0)												
R-X			RWP-X												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EIDXS(12–0)												
R-X			RWP-X												

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-78. Element Index Offset Register (EIOFF) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads are undefined and writes have no effect.
28–16	EIDXD(12–0)	0–1FFFh	Destination address element index. These bits define the offset to be added to the destination address after each element transfer.
15–13	Reserved		Reads are undefined and writes have no effect.
12–0	EIDXS(12–0)	0–1FFFh	Source address element index. These bits define the offset to be added to the source address after each element transfer.

20.4.2.6 Frame Index Offset Register (FIOFF)

Figure 20-90 and Table 20-79 describe this register.

Figure 20-90. Frame Index Offset Register (FIOFF) [offset = 0x018]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			FIDXD(12–0)												
R-X			R/WP-X												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			FIDXS(12–0)												
R-X			R/WP-X												

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-79. Frame Index Offset Register (FIOFF) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads are undefined and writes have no effect.
28–16	FIDXD(12–0)	0–1FFFh	Destination address frame index. These bits define the offset to be added to the destination address after element count reached one.
15–13	Reserved		Reads are undefined and writes have no effect.
12–0	FIDXS(12–0)	0–1FFFh	Source address frame index. These bits define the offset to be added to the source address after element count reached one.

20.4.2.7 Current Source Address Register (CSADDR)

Figure 20-91 and Table 20-80 describe this register.

Figure 20-91. Current Source Address Register (CSADDR) [offset = 0x800]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSADDR(31–16)															
R-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSADDR(15–0)															
R-X															

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-80. Current Source Address Register (CSADDR) Field Descriptions

Bit	Name	Value	Description
31–0	CSADDR(31–0)	0–FFFF FFFFh	Current source address. These bits contain the current working absolute 32-bit source address (physical). These bits are only updated after a channel is arbitrated out from the priority queue.

20.4.2.8 Current Destination Address Register (CDADDR)

Figure 20-92 and Table 20-81 describe this register.

Figure 20-92. Current Destination Address Register (CDADDR) [offset = 0x804]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDADDR(31–16)															
R-X															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDADDR(15–0)															
R-X															

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-81. Current Destination Address Register (CDADDR) Field Descriptions

Bit	Name	Value	Description
31–0	CDADDR(31–0)	0–FFFF FFFFh	Current destination address. These bits contain the current working absolute 32-bit destination address (physical). These bits are only updated after a channel is arbitrated out of the priority queue.

20.4.2.9 Current Transfer Count Register (CTCOUNT)

Figure 20-93 and Table 20-82 describe this register.

Figure 20-93. Current Transfer Count Register (CTCOUNT) [offset = 0x808]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CFTCOUNT(12–0)												
R-X			R-X												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CETCOUNT(12–0)												
R-X			R-X												

R = Read in all modes; WP = Write in privilege mode only; X = Unknown; -n = Value after reset

.

Table 20-82. Current Transfer Count Register (CTCOUNT) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Reads are undefined and writes have no effect.
28-16	CFTCOUNT(12–0)	0–1FFFh	Current frame transfer count.Returned the current remaining frame counts.
15-13	Reserved		Reads are undefined and writes have no effect.
12–0	CETCOUNT(12–0)	0–1FFFh	Current element transfer count. These bits return the current remaining element counts. CTCOUNT register is only updated after a channel is arbitrated out of the priority queue.

Chapter 21
SPNU489C-February 2012

Real-Time Interrupt (RTI) Module

This document describes the functionality of the real-time interrupt (RTI) module. It is specifically designed to support time-triggered operating systems and real-time operating systems.

Topic ... Page

21.1 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 timebases 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.

In addition the RTI provides a mechanism to synchronize the operating system to the FlexRay communication cycle. Clock supervision allows to detect issues on the FlexRay bus with an automatic switch to an internally generated timebase.

21.1.1 Features

The RTI module has the following features:

- Two independent 64 bit counter blocks
- Four configurable compares for generating operating system ticks or DMA requests. Each event can be driven by either counter block 0 or counter block 1.
- One counter block usable for application synchronization to FlexRay network including clock supervision
- Fast enabling/disabling of events
- Two time stamp (capture) functions for system or peripheral interrupts, one for each counter block

21.1.2 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.

21.2 Module Operation

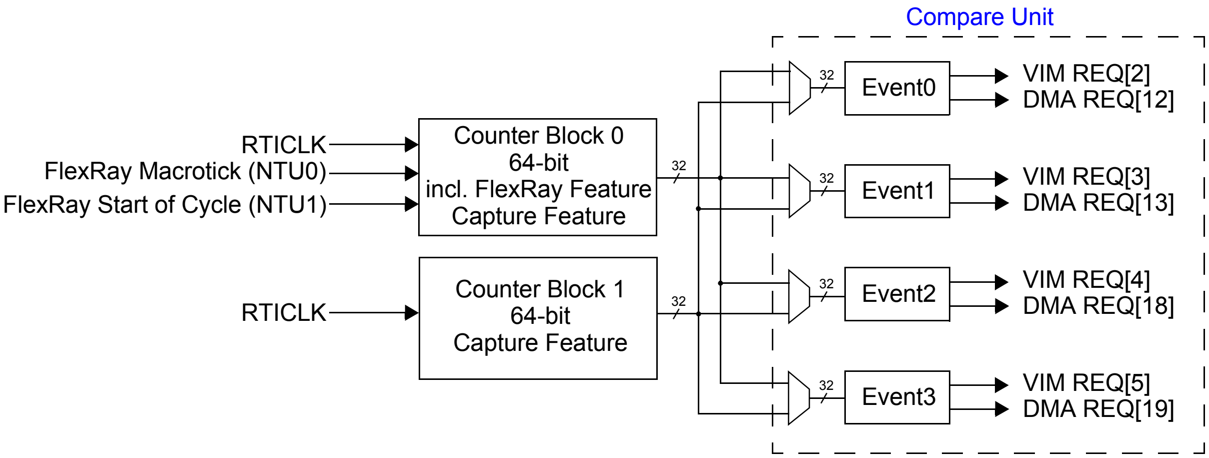
Figure 21-1 illustrates the high level block diagram of the RTI module.

The RTI module has two independent counter blocks for generating different timebases: counter block 0 and counter block 1. The two counter blocks provide the same basic functionality, but counter block 0 has the additional functionality of being able to work with the FlexRay Macrotick (NTU0) or Start of Cycle (NTU1) and do a clock supervision to detect a missing signal.

A compare unit compares the counters with programmable values and generates four independent interrupt or DMA requests 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 21-1. RTI Block Diagram

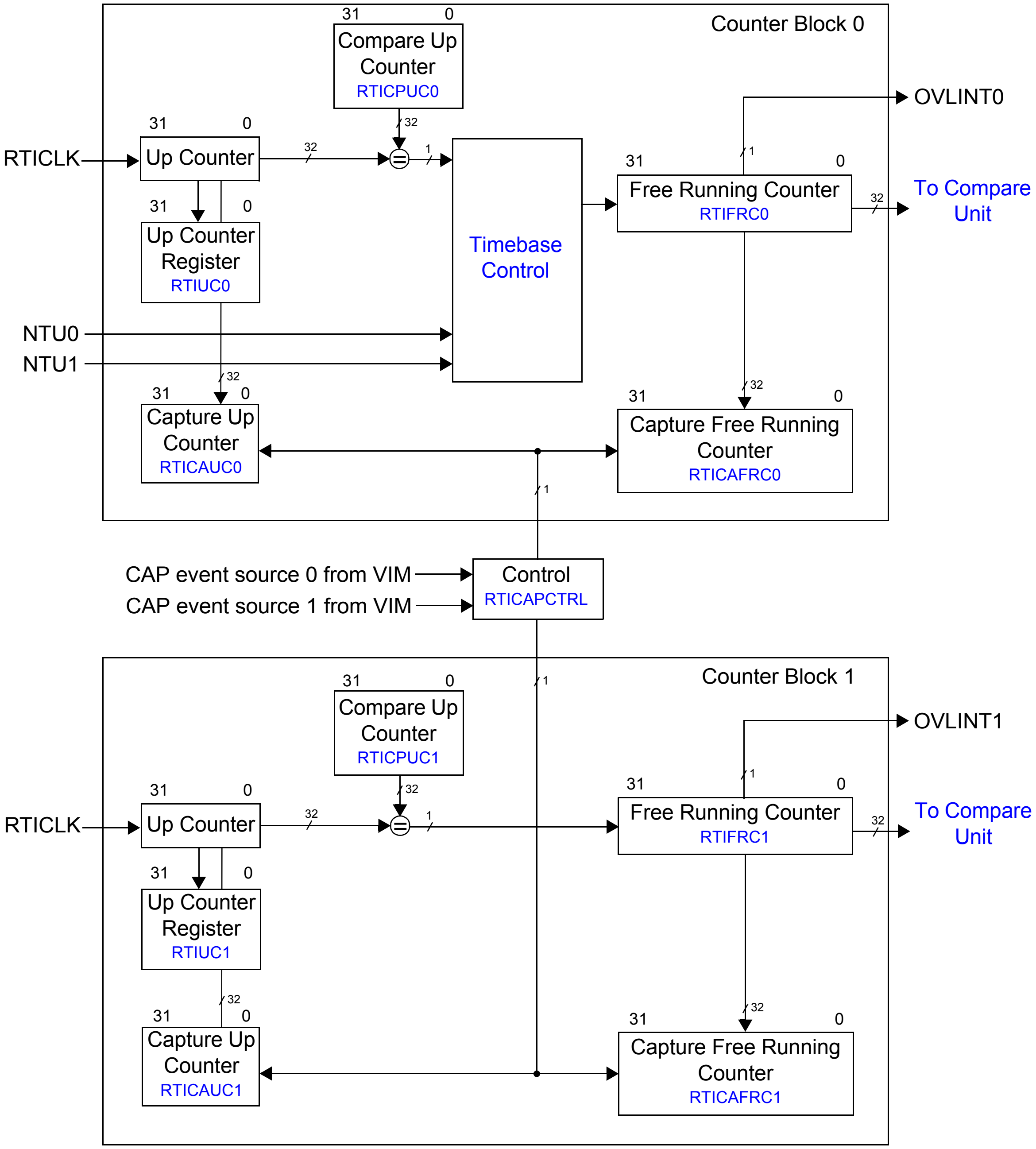


21.2.1 Counter Operation

Each counter block consists of the following (see Figure 21-2):

- One 32-bit prescale counter (RTIUC0 or RTIUC1)
- One 32-bit free running counter (RTIFRC0 or RTIFRC1)

Figure 21-2. Counter Block Diagram


Counter Block 0
Compare Up Counter RTICPUC0
31 0
RTICLK
Up Counter
Up Counter Register RTIUC0
32
1
Timebase Control
Free Running Counter RTIFRC0
OVLINT0
To Compare Unit
NTU0
NTU1
Capture Up Counter RTICAUC0
Capture Free Running Counter RTICAFRC0
CAP event source 0 from VIM
CAP event source 1 from VIM
Control RTICAPCTRL
Counter Block 1
Compare Up Counter RTICPUC1
Up Counter
Up Counter Register RTIUC1
Free Running Counter RTIFRC1
OVLINT1
To Compare Unit
Capture Up Counter RTICAUC1
Capture Free Running Counter RTICAFRC1

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 timebase for the operating system. See Section

21.2.2 for the timebase 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:

$$f_{RTIFRCx} = \begin{cases} \dfrac{f_{RTICLK}}{RTICPUCx + 1} & \text{when RTICPUCx} \neq 0 \\ \dfrac{f_{RTICLK}}{2^{32}} & \text{when RTICPUCx } = 0 \end{cases}$$

The counter values can be determined by reading the respective counter registers or by generating a hardware event which captures the counter value into the respective capture register. Both functions are described in the following sections.

21.2.1.1 Counter and Capture Read Consistency

The device internal databus is 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).

> 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).

> 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.

21.2.1.2 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 21.2.1.1).

21.2.2 Interrupt/DMA Requests

There are four compare registers (RTICOMPy) to generate interrupt requests to the VIM or DMA requests to the DMA controller. The interrupts can be used to generate different timebases 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 matched. The period of the generated interrupt/DMA request can be calculated with:

$$t_{COMPx} = t_{RTICLK} \times (RTICPUCy + 1) \times RTIUDCPy$$

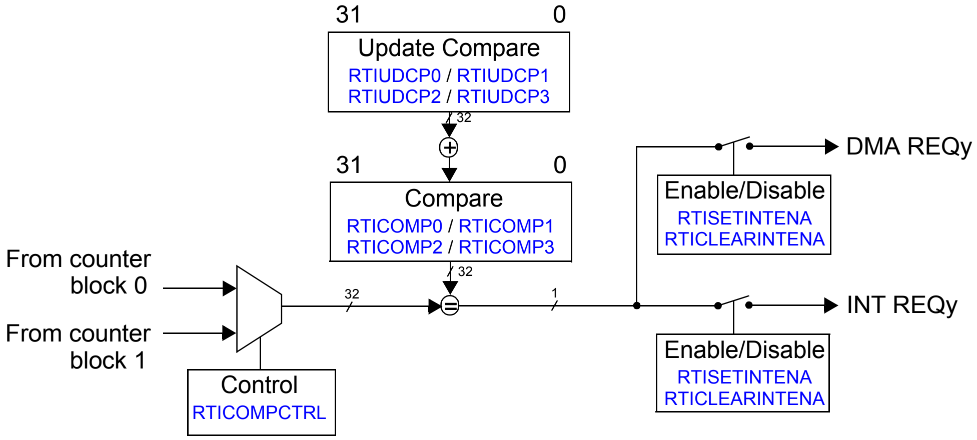
if RTICPUCy = 0,

$$t_{COMPx} = t_{RTICLK} \times 2^{32} \times RTIUDCPy$$

If RTIUDCPy = 0,

$$t_{COMPx} = t_{RTICLK} \times (RTICPUCy + 1) \times 2^{32}$$

Figure 21-3. 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/DMA requests can be enabled in the RTISETINTENA register and disabled in the RTICLEARINTENA register. The RTIINTFLAG register shows the pending interrupts.

21.2.3 RTI Clocking

The counter blocks are clocked with RTICLK (for definition see Section 2.3, "Clock Domains"). Counter block 0 can be clocked in addition by either the FlexRay Macrotick (NTU0) or the FlexRay Start of Cycle (NTU1).

A clock supervision for the NTUx clocking scheme is implemented to avoid missing operating system ticks.

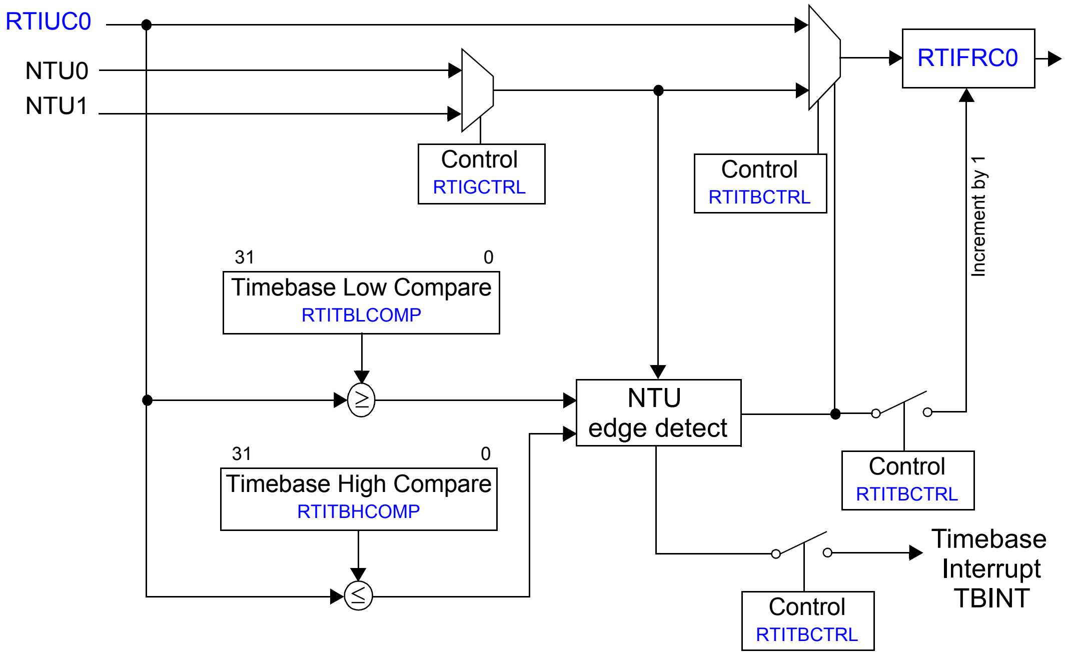
21.2.4 Synchronizing Timer Events to Network Time (NTU)

For applications which are participating on a time-triggered communication bus, it is often beneficial to synchronize the application or operating system to the network time. The RTI provides a feature to increment Free Running Counter 0 (RTIFRC0) by a periodic clock provided by the communication module. In this case two different clocks can be chosen. One is the FlexRay module Macrotick (NTU0) and the other is the Start of Cycle (NTU1) information of the same module.

The application has control over which clock (RTICLK, NTU0, NTU1) should be used for clocking RTIFRC0. If NTUx is used, a clock supervision circuit allows to monitor this clock and provides a fallback solution, should the clock be non-functional (missing). A too fast running NTUx cannot be detected.

RTIUC0 is utilized to monitor the NTUx signal. A detection window can be programmed in which a valid NTU clock pulse needs to occur. If no pulse is detected, the RTI automatically switches back to clock the Free Running Counter 0 with RTIUC0. In order to avoid a big jitter in the operating ticks, in case a switch back to RTIUC0 happens, RTICPUC0 should be set to a value so the clock frequency RTIUC0 outputs is approximately the same as the NTUx frequency.

Figure 21-4. Timebase Control



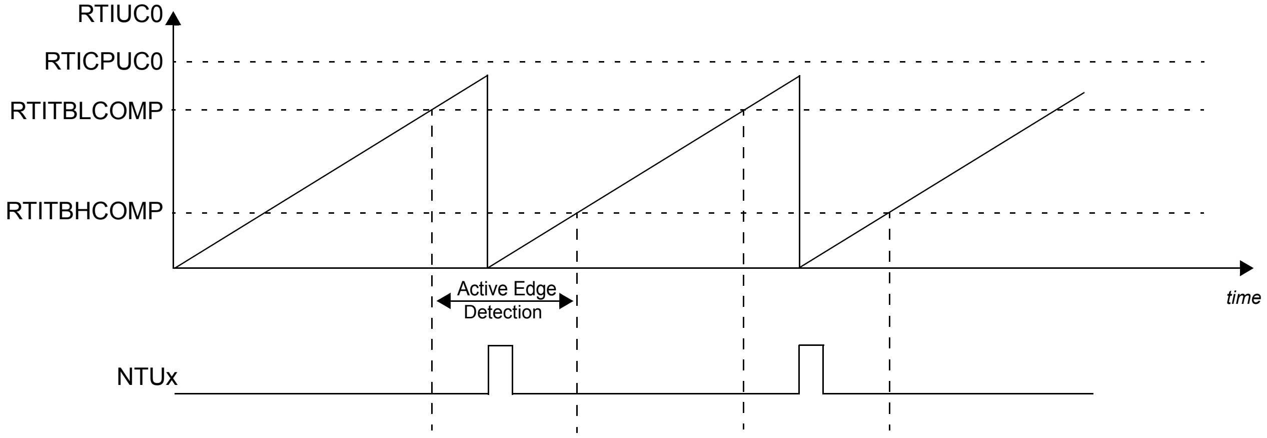
21.2.4.1 Detecting Clock Edges

To detect clock edges on the NTUx signal, the timebase low compare has to be set lower or equal than the value stored in the RTICPUC0 register and the timebase high compare has to be set higher than 0 and lower than the timebase low compare value. This effectively opens a window in which an edge of the NTUx signal is expected (see Figure 21-5). Outside this window, no edges will be detected. If no edge will occur inside the detection window, the multiplexer is switched to internal timebase. The application can select to generate a timebase interrupt (TBINT) and if the INC bit is set, also will automatically increment RTIFRC0 by one to compensate for the missed clock cycle of NTUx. If an edge occurs inside the window, RTIUC0 will be reset to synchronize the two timebases.

In order to make the edge detection work properly, the value in RTICPUC0 needs to be adapted so that RTIUC0 has a similar period as NTUx.

To ensure the NTUx signal is properly detected, the NTUx period must be at least twice as long as the RTICLK period.

Figure 21-5. Clock detection scheme



21.2.4.2 Switching from Internal Source to External Source

If the application switches from an internal source to an external source, the two signals must be synchronized (see Figure 21-6). The synchronization will occur when the TBEXT bit is set. RTIUC0 will be reset and the edge detection circuit will be active for one (RTICPUC0 + RTITBHCOMP) period or until an edge is detected. If there is no pulse during this period, the source will be reset from an external clock source to an internal clock source. If an edge is detected, the windowed edge detection behavior will take place. Setting the TBEXT bit will not increment free running counter 0.

Note:

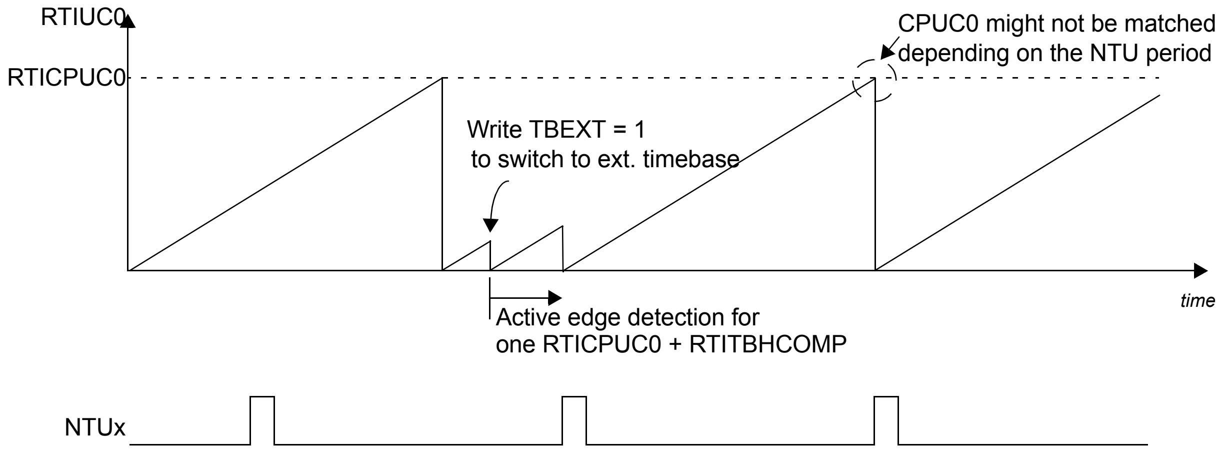
If an external timebase is used, then the software must ensure that timebase low compare and timebase high compare are programmed to a valid state before switching TBEXT to an external source. This state is necessary to allow the timebase control circuit to operate correctly.
The following condition must be met:

•RTITBHCOMP < RTITBLCOMP ≤ RTICPUC0

RTITBHCOMP must be lower than RTICPUC0 because RTIUC0 will be reset if RTICPUC0 is reached. RTITBHCOMP will represent the number of RTICLK cycles from RTICPUC0 until the circuit switches to the internal timebase when no NTU edge is detected.
If an external timebase is used, RTIGCTRL[0] must be set to 1 (enable RTIUC0) to ensure that the timebase control circuit does not wait indefinitely for an incoming signal.

Figure 21-6 shows a timing example for the synchronization phase when the TBEXT bit is set.

Figure 21-6. Switch to NTUx

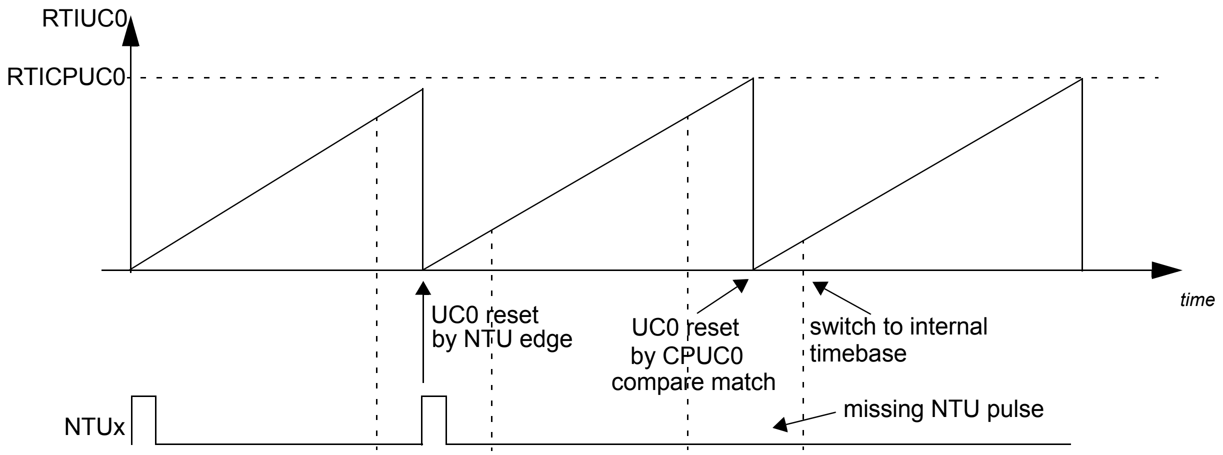

RTIUC0
RTICPUC0
CPUC0 might not be matched
depending on the NTU period
Write TBEXT = 1
to switch to ext. timebase
time
Active edge detection for
one RTICPUC0 + RTITBHCOMP
NTUx

21.2.4.3 *Switching from External Source to Internal Source*

When the edge detection is active (TBEXT = 1) and no clock edge of NTUx is detected inside the programmed detection window, the RTI will automatically switch the timebase to RTIUC0. Figure 21-7 shows a timing example for a missing NTU signal. In the case where the INC bit is set, RTIFRC0 will automatically be incremented by one to compensate for the missed NTU pulse.

Setting TBEXT = 0 will also switch the clock source for RTIFRC0 to RTIUC0.

Figure 21-7. Missing NTUx Signal Example



21.2.5 *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 2.6, *Low Power Modes on TMS570LS20x/10x Safety MCUs*).

The operation of the RTI Module is guaranteed 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, timebase 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 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.

When the device is put into low power mode, the peripheral which is generating the external clock NTU is no longer active, and the timebase control circuitry has to switch to an internal clocking scheme when it detects a missing clock on NTU. The timebase interrupt will wake up the device and the application software has to adapt the periodic interrupt generation to the internal clock source.

DMA transfers will be disabled, and DMA requests will not be generated after device wakeup since the DMA controller will be powered down.

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.

21.2.6 Debug Mode Behavior

Once the system enters debug mode, the behavior of the RTI depends on the COS bit. If the bit is cleared and 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. However if the external timebase (NTU) is used and the system is in debug mode, the timebase control circuit will switch to internal timebase, once it detects the missing NTU signal of the suspended communication controller. This will be signaled with an TBINT interrupt, so that software can resynchronize after the device exits debug mode.

21.3 Control Registers

Figure 21-8 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 0xFFFF FC00. The address locations not listed, are reserved..

Figure 21-8. RTI 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
0x00 RTIGCTRL Page 1602	Reserved														NTUSEL	
	COS	Reserved													CNT1 EN	CNT0 EN
0x04 RTITBCTRL Page 1604	Reserved															
	Reserved														INC	TB EXT
0x08 RTICAPCTRL Page 1605	Reserved															
	Reserved														CAP CNTR1	CAP CNTR0
0x0C RTICOMPCTRL Page 1606	Reserved															
	Reserved			COMP SEL3	Reserved			COMP SEL2	Reserved			COMP SEL1	Reserved			COMP SEL0
0x10 RTIFRC0 Page 1607	FRC0(31–16)															
	FRC0(15–0)															
0x14 RTIUC0 Page 1608	UC0(31–16)															
	UC0(15–0)															
0x18 RTICPUC0 Page 1609	CPUC0(31–16)															
	CPUC0(15–0)															

Figure 21-8. RTI Registers (Continued)

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
0x1C Reserved	Reserved															
	Reserved															
0x20 RTICAFRC0 Page 1610	CAFRC0(31–16)															
	CAFRC0(15–0)															
0x24 RTICAUC0 Page 1611	CAUC0(31–16)															
	CAUC0(15–0)															
0x28 Reserved	Reserved															
	Reserved															
0x2C Reserved	Reserved															
	Reserved															
0x30 RTIFRC1 Page 1612	FRC1(31–16)															
	FRC1(15–0)															
0x34 RTIUC1 Page 1613	UC1(31–16)															
	UC1(15–0)															
0x38 RTICPUC1 Page 1614	CPUC1(31–16)															
	CPUC1(15–0)															

Figure 21-8. RTI Registers (Continued)

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
0x3C Reserved	Reserved															
	Reserved															
0x40 RTICAFRC1 Page 1615	CAFRC1(31–16)															
	CAFRC1(15–0)															
0x44 RTICAUC1 Page 1616	CAUC1(31–16)															
	CAUC1(15–0)															
0x48 Reserved	Reserved															
	Reserved															
0x4C Reserved	Reserved															
	Reserved															
0x50 RTICOMP0 Page 1617	COMP0(31–16)															
	COMP0(15–0)															
0x54 RTIUDCP0 Page 1618	UDCP0(31–16)															
	UDCP0(15–0)															
0x58 RTICOMP1 Page 1619	COMP1(31–16)															
	COMP1(15–0)															

Figure 21-8. RTI Registers (Continued)

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
0x5C RTIUDCP1 Page 1620	UDCP1(31–16)															
	UDCP1(15–0)															
0x60 RTICOMP2 Page 1621	COMP2(31–16)															
	COMP2(15–0)															
0x64 RTIUDCP2 Page 1622	UDCP2(31–16)															
	UDCP2(15–0)															
0x68 RTICOMP3 Page 1623	COMP3(31–16)															
	COMP3(15–0)															
0x6C RTIUDCP3 Page 1624	UDCP3(31–16)															
	UDCP3(15–0)															
0x70 RTITBLCOMP Page 1625	TBLCOMP(31–16)															
	TBLCOMP(15–0)															
0x74 RTITBHCOMP Page 1626	TBHCOMP(31–16)															
	TBHCOMP(15–0)															
0x78 Reserved	Reserved															
	Reserved															

Figure 21-8. RTI Registers (Continued)

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
0x7C Reserved	Reserved															
	Reserved															
0x80 RTISETINTENA Page 1627	Reserved													SET OVL1 INT	SET OVL0 INT	SET TB INT
	Reserved				SET DMA3	SET DMA2	SET DMA1	SET DMA0	Reserved				SET INT3	SET INT2	SET INT1	SET INT0
0x84 RTICLEARINTENA Page 1629	Reserved													CLEAR OVL1 INT	CLEAR OVL0 INT	CLEAR TB INT
	Reserved				CLEAR DMA3	CLEAR DMA2	CLEAR DMA1	CLEAR DMA0	Reserved				CLEAR INT3	CLEAR INT2	CLEAR INT1	CLEAR INT0
0x88 RTIINTFLAG Page 1632	Reserved													OVL1 INT	OVL0 INT	TB INT
	Reserved												INT3	INT2	INT1	INT0
0x8C..A0 Reserved	Reserved															
	Reserved															

21.3.1 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 21-9 and described in Table 21-1.

Figure 21-9. RTI Global Control Register (RTIGCTRL) [offset = 0x00h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														NTUSEL	
R-0														RWP-00	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COS	Reserved													CNT1 EN	CNT0 EN
R/WP-0	R-0													R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-1. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect.
17-16	NTUSEL		Select NTU signal. These bits determine which NTU input signal is used as external timebase
		00	NTU0 (FlexRay Macrotick)
		01	NTU1 (FlexRay Start of cycle)
		10	Reserved
		11	Reserved
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into debug mode or if they continue counting.
		0	Counters are stopped while in debug mode.
		1	Counters are running while in debug mode.
14–2	Reserved		Reads return 0 and 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.

Note:
If the application uses the timebase circuit for synchronization between the communications controller and the operating system and the device enters debug mode, the synchronization may be lost depending on the COS setting in the RTI module and the debug mode behavior of the communications controller.

21.3.2 RTI Timebase Control Register (RTITBCTRL)

The timebase control register selects if the free running counter 0 is incremented by RTICLK or NTU. This register is shown in Figure 21-10 and described in Table 21-2.

Figure 21-10. RTI Timebase Control Register (RTITBCTRL) [offset = 0x04h]

31–16
Reserved
R-0

15–2	1	0
Reserved	INC	TB EXT
R-0	R/WP-0	R/WP-0

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 21-2. RTI Timebase Control Register (RTITBCTRL) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and writes have no effect.
1	INC		Increment free running counter 0. This bit determines whether the free running counter 0 (RTIFRC0) is automatically incremented if a failing clock on the NTU signal is detected.
		0	RTIFRC0 will not be incremented on a failing external clock.
		1	RTIFRC0 will be incremented on a failing external clock.
0	TBEXT		Timebase external. This bit selects whether the free running counter 0 (RTIFRC0) is clocked by the internal up counter 0 (RTIUC0) or from the external signal NTU. Setting the TBEXT bit from 0 to 1 will not increment RTIFRC0, since RTIUC0 is reset. When the timebase supervisor circuit detects a missing clock edge, then the TBEXT bit is reset. Only the software can select whether the external signal should be used.
		0	RTIUC0 clocks RTIFRC0.
		1	NTU clocks RTIFRC0.

21.3.3 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in Figure 21-11 and described in Table 21-3.

Figure 21-11. RTI Capture Control Register (RTICAPCTRL) [offset = 0x08h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="14">Reserved</td><td>CAP CNTR1</td><td>CAP CNTR0</td></tr>
<tr><td colspan="14">R-0</td><td>R/WP-0</td><td>R/WP-0</td></tr>
</table>

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-3. RTI Capture Control Register (RTICAPCTRL) Field Descriptions

Bit	Name	Value	Description
31–2	Reserved		Reads return 0 and 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.

21.3.4 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in Figure 21-12 and described in Table 21-12.

Figure 21-12. RTI Compare Control Register (RTICOMPCTRL) [offset = 0x0Ch]

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
Reserved			COMP SEL3	Reserved			COMP SEL2	Reserved			COMP SEL1	Reserved			COMP SEL0
R-0			R/WP-0	R-0			R/WP-0	R-0			R/WP-0	R-0			R/WP-0

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 21-4. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads return 0 and 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		Reads return 0 and 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		Reads return 0 and 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		Reads return 0 and 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.

21.3.5 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 21-13 and described in Table 21-5.

Figure 21-13. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 0x10h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRC0(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRC0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-5. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions

Bit	Name	Value	Description
31–0	FRC0(31–0)	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.**

21.3.6 RTI Up Counter 0 Register (RTIUC0)

The up counter 0 register holds the current value of prescale counter. This register is shown in Figure 21-14 and described in Table 21-6.

Figure 21-14. RTI Up Counter 0 Register (RTIUC0) [offset = 0x14h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UC0(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UC0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-6. RTI Up Counter 0 Register (RTIUC0) Field Descriptions

Bit	Name	Value	Description
31–0	UC0(31–0)	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.**

21.3.7 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 21-15 and described in Table 21-7.

Figure 21-15. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 0x18h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">CPUC0(31–16)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CPUC0(15–0)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-7. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions

Bit	Name	Value	Description
31–0	CPUC0(31–0)	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} If CPUC0 $\neq$ 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.

21.3.8 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 21-16 and described in Table 21-8.

Figure 21-16. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 0x20h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAFRC0(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAFRC0(15–0)															
R-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-8. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions

Bit	Name	Value	Description
31–0	CAFRC0(31–0)	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.

21.3.9 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 21-17 and described in Table 21-9.

Figure 21-17. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 0x24]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAUC0(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAUC0(15–0)															
R-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-9. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions

Bit	Name	Value	Description
31–0	CAUC0(31–0)	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.

21.3.10 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 21-18 and described in Table 21-10.

Figure 21-18. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 0x30h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">FRC1(31–16)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">FRC1(15–0)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 21-10. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions

Bit	Name	Value	Description
31–0	FRC1(31–0)	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.**

21.3.11 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 21-19 and described in Table 21-11.

Figure 21-19. RTI Up Counter 1 Register (RTIUC1) [offset = 0x34h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UC1(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UC1(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, *-n* = Value after reset

Table 21-11. RTI Up Counter 1 Register (RTIUC1) Field Descriptions

Bit	Name	Value	Description
31–0	UC1(31–0)	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.**

21.3.12 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 21-20 and described in Table 21-12.

Figure 21-20. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 0x38h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUC1(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUC1(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 21-12. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions

Bit	Name	Value	Description
31–0	CPUC1(31–0)	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 set 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_{FRC1} = RTICLK/2^{32} 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.

21.3.13 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 21-21 and described in Table 21-13.

Figure 21-21. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 0x40h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAFRC1(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAFRC1(15–0)															
R-0															

R = Read, -*n* = Value after reset

Table 21-13. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions

Bit	Name	Value	Description
31–0	CAFRC1(31–0)	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.

21.3.14 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 21-22 and described in Table 21-14.

Figure 21-22. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 0x44h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAUC1(31–16)															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAUC1(15–0)															
R-0															

R = Read, WP = Write in privilege mode only, -*n* = Value after reset

Table 21-14. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions

Bit	Name	Value	Description
31–0	CAUC1(31–0)	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.

21.3.15 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in Figure 21-23 and described in Table 21-15.

Figure 21-23. RTI Compare 0 Register (RTICOMP0) [offset = 0x50h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP0(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-15. RTI Compare 0 Register (RTICOMP0) Field Descriptions

Bit	Name	Value	Description
31–0	COMP0(31–0)	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. With this register it is also possible to initiate a DMA request. 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.

21.3.16 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 21-24 and described in Table 21-16.

Figure 21-24. RTI Update Compare 0 Register (RTIUDCP0) [offset = 0x54h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP0(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDCP0(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-16. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP0(31–0)	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.

21.3.17 RTI Compare 1 Register (RTICOMP1)

The compare 1 register holds the value to be compared to the counters. This register is shown in Figure 21-25 and described in Table 21-17.

Figure 21-25. RTI Compare 1 Register (RTICOMP1) [offset = 0x58]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP1(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-17. RTI Compare 1 Register (RTICOMP1) Field Descriptions

Bit	Name	Value	Description
31–0	COMP1(31–0)	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. With this register, it is possible to initiate a DMA request. 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.

21.3.18 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 21-26 and described in Table 21-18.

Figure 21-26. RTI Update Compare 1 Register (RTIUDCP1) [offset = 0x5Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP1(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDCP1(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-18. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP1(31–0)	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.

21.3.19 RTI Compare 2 Register (RTICOMP2)

The compare 2 register holds the value to be compared to the counters. This register is shown in Figure 21-27 and described in Table 21-19.

Figure 21-27. RTI Compare 2 Register (RTICOMP2) [offset = 0x60h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-19. RTI Compare 2 Register (RTICOMP2) Field Descriptions

Bit	Name	Value	Description
31–0	COMP2(31–0)	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. With this register, it is possible to initiate a DMA request. 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.

21.3.20 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 21-28 and described in Table 21-20.

Figure 21-28. RTI Update Compare 2 Register (RTIUDCP2) [offset = 0x64h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP2(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDCP2(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-20. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP2(31–0)	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.

21.3.21 RTI Compare 3 Register (RTICOMP3)

The compare 3 register holds the value to be compared to the counters. This register is shown in Figure 21-29 and described in Table 21-21.

Figure 21-29. RTI Compare 3 Register (RTICOMP3) [offset = 0x68h]

31–16
COMP3(31–16)
R/WP-0

15–0
COMP3(15–0)
R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-21. RTI Compare 3 Register (RTICOMP3) Field Descriptions

Bit	Name	Value	Description
31–0	COMP3(31–0)	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. With this register, it is possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register will provide a new compare value.

21.3.22 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 21-30 and described in Table 21-22.

Figure 21-30. RTI Update Compare 3 Register (RTIUDCP3) [offset = 0x6Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDCP3(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDCP3(15–0)															
R/WP-0															

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-22. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions

Bit	Name	Value	Description
31–0	UDCP3(31–0)	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.

21.3.23 RTI Timebase Low Compare Register (RTITBLCOMP)

The timebase low compare register holds the value to activate the edge detection circuit. This register is shown in Figure 21-31 and described in Table 21-23.

Figure 21-31. RTI Timebase Low Compare Register (RTITBLCOMP) [offset = 0x70h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBLCOMP(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBLCOMP(15–0)															
R/WP-0															

R = Read, WP = Write in privilege mode only, *-n* = Value after reset

Table 21-23. RTI Timebase Low Compare Register (RTITBLCOMP) Field Descriptions

Bit	Name	Value	Description
31–0	TBLCOMP(31–0)	0–FFFF FFFFh	Timebase low compare value. This value determines when the edge detection circuit starts monitoring the NTU signal. It will be compared with RTIUC0. A read of this register will return the current compare value. A write to this register has the following effects: If TBEXT = 0: The compare value is updated. If TBEXT = 1: The compare value is not changed.

21.3.24 RTI Timebase High Compare Register (RTITBHCOMP)

The timebase high compare register holds the value to deactivate the edge detection circuit. This register is shown in Figure 21-32 and described in Table 21-24.

Figure 21-32. RTI Timebase High Compare Register (RTITBHCOMP) [offset = 0x74h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBHCOMP(31–16)															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBHCOMP(15–0)															
R/WP-0															

R = Read, W = Write in privilege mode only, -*n* = Value after reset

Table 21-24. RTI Timebase High Compare Register (RTITBHCOMP) Field Descriptions

Bit	Name	Value	Description
31–0	TBHCOMP(31–0)	0–FFFF FFFFh	Timebase high compare value. This value determines when the edge detection circuit will stop monitoring the NTU signal. It will be compared with RTIUC0. RTITBHCOMP must be less than RTICPUC0 because RTIUC0 will be reset when RTICPUC0 is reached. Example: The NTU edge detection circuit should be active ± 10 RTICLK cycles around RTICPUC0. RTICPUC0 = 0x00000050 RTITBLCOMP = 0x000046 RTITBHCOMP = 0x00000009 A read of this register will return the current compare value. A write to this register has the following effects: If TBEXT = 0: The compare value is updated. If TBEXT = 1: The compare value is not changed.

21.3.25 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 21-33 and described in Table 21-25.

Figure 21-33. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 0x80h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													SET OVL1 INT	SET OVL0 INT	SET TBINT
R-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
Reserved				SET DMA3	SET DMA2	SET DMA1	SET DMA0	Reserved				SET INT3	SET INT2	SET INT1	SET INT0
R-0				R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0				R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-25. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18	SETOVL1INT		Set free running counter 1 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
18	Reserved		Read/writable, but no affect on operation.
17	SETOVL0INT		Set free running counter 0 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
16	SETTBINT		Set timebase interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
15–12	Reserved		Reads return 0 and writes have no effect.
11	SETDMA3		Set compare DMA request 3.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The DMA request is enabled.

Table 21-25. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions (Continued)

Bit	Name	Value	Description
10	SETDMA2		Set compare DMA request 2.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The DMA request is enabled.
9	SETDMA1		Set compare DMA request 1.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The DMA request is enabled.
8	SETDMA0		Set compare DMA request 0.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The DMA request is enabled.
7-4	Reserved		Reads return 0 and writes have no effect.
3	SETINT3		Set compare interrupt 3.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
2	SETINT2		Set compare interrupt 2.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
1	SETINT1		Set compare interrupt 1.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.
0	SETINT0		Set compare interrupt 0.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read or write:* The interrupt is enabled.

21.3.26 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 21-34 and described in Table 21-26.

Figure 21-34. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 0x84h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													CLEAR OVL1 INT	CLEAR OVL0 INT	CLEAR TBINT
R-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
Reserved				CLEAR DMA3	CLEAR DMA2	CLEAR DMA1	CLEAR DMA0	Reserved				CLEAR INT3	CLEAR INT2	CLEAR INT1	CLEAR INT0
R-0				R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0				R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privileged mode only, -*n* = Value after reset

Table 21-26. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect.
18	CLEAROVL1INT		Clear free running counter 1 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
17	CLEAROVL0INT		Clear free running counter 0 overflow interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
16	CLEARTBINT		Clear timebase interrupt.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
15–12	Reserved		Reads return 0 and writes have no effect.

Table 21-26. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions (Continued)

Bit	Name	Value	Description
11	CLEARDMA3		Clear compare DMA request 3.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The DMA request is enabled. *Write:* The DMA request is disabled.
10	CLEARDMA2		Clear compare DMA request 2.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The DMA request is enabled. *Write:* The DMA request is disabled.
9	CLEARDMA1		Clear compare DMA request 1.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The DMA request is enabled. *Write:* The DMA request is disabled.
8	CLEARDMA0		Clear compare DMA request 0.
		0	*Read:* The DMA request is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The DMA request is enabled. *Write:* The DMA request is disabled.
7-4	Reserved		Reads return 0 and writes have no effect.
3	CLEARINT3		Clear compare interrupt 3.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
2	CLEARINT2		Clear compare interrupt 2.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

Table 21-26. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions (Continued)

Bit	Name	Value	Description
1	CLEARINT1		Clear compare interrupt 1.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
0	CLEARINT0		Clear compare interrupt 0.
		0	*Read:* The interrupt is disabled. *Write:* The corresponding bit is unchanged.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

21.3.27 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 21-35 and described in Table 21-27.

Figure 21-35. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 0x88h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													OVL1 INT	OVL0 INT	TBINT
R-0													R/CP-0	R/CP-0	R/CP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												INT3	INT2	INT1	INT0
R-0												R/CP-0	R/CP-0	R/CP-0	R/CP-0

R = Read, CP = Clear in privileged mode only, U = Undefined; -*n* = Value after reset

Table 21-27. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and 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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set 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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set 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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
15–4	Reserved		Reads return 0 and writes have no effect.

Table 21-27. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (Continued)

Bit	Name	Value	Description
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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set 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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.
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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set 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:* The bit is unchanged.
		1	*Read:* An interrupt is pending. *Write:* The bit is set to 0.

Chapter 22

SPNU489C-February 2012

Cyclic Redundancy Check Controller (CRC) Module

Topic Page

22.1 Overview

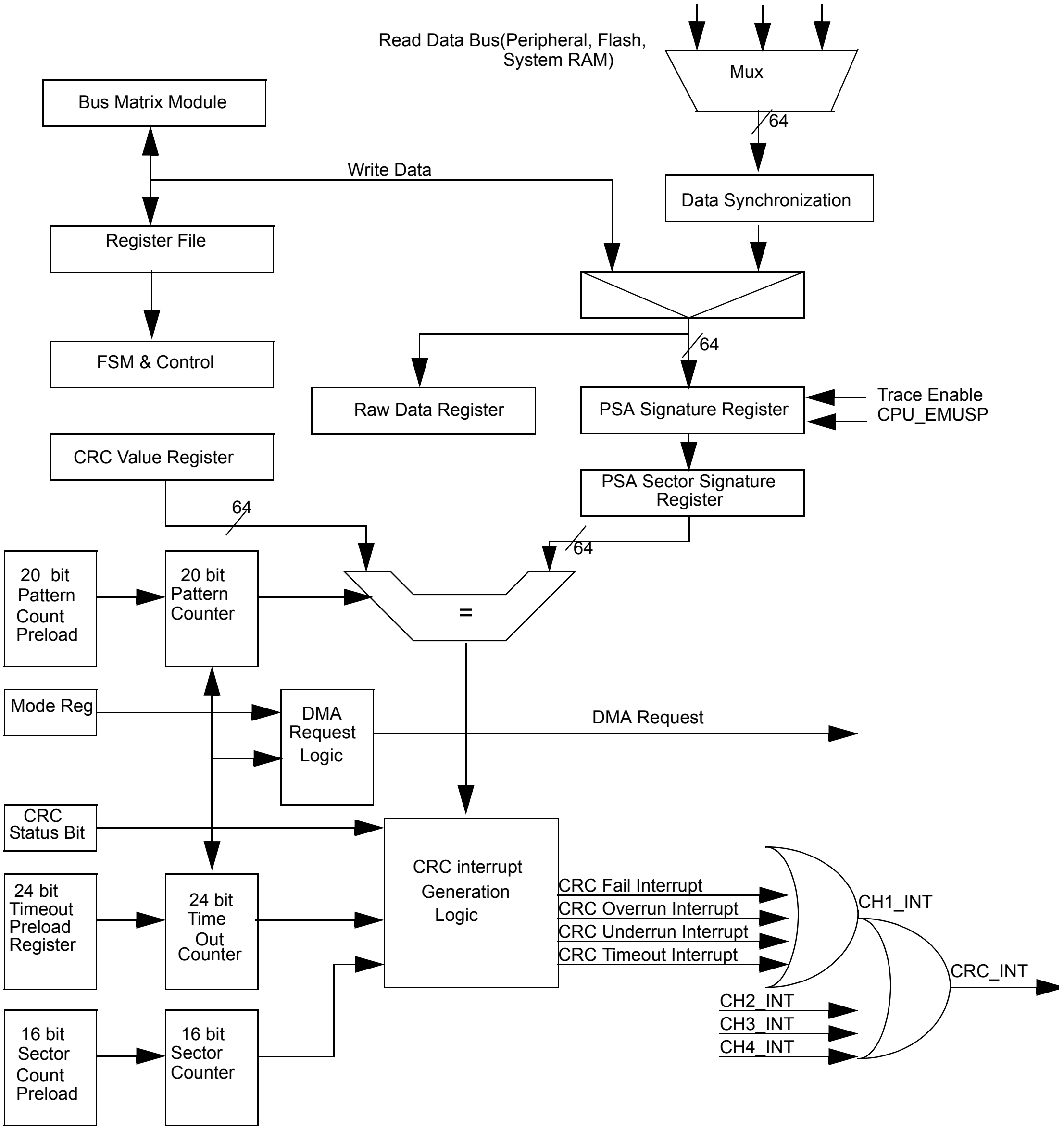
MCRC Controller is a module which is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into MCRC Controller. The responsibility of MCRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a pre-determined good signature value. MCRC controller provides up to four channels to perform CRC calculation on multiple memories in parallel and can be used on any memory system. Channel 1 can also be put into data trace mode. In data trace mode, MCRC controller compresses each data being read through CPU read data bus.

22.2 Features

The CRC (Cyclic Redundancy Check) controller offers:

- Four 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.
- Each channel has a CRC Value Register which contains the pre-determined CRC value.
- Use timed base event trigger from timer to initiate DMA data transfer.
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression.
- Three modes of operation. Auto, Semi-CPU and Full-CPU.
- For each channel, CRC can be performed either by MCRC Controller or by CPU.
- Automatically perform signature verification without CPU intervention in AUTO mode.
- Generate interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself.
- Generate CRC fail interrupt in AUTO mode if signature verification fails.
- Generate Timeout interrupt if CRC is not performed within the time limit.
- Generate DMA request per channel to initiate CRC value transfer.
- Peripheral Bus slave interface.
- Data trace capability on Peripheral Bus Master, Flash and System RAM data buses.

Figure 22-1. MCRC Controller Block Diagram For One Channel



Notes: 1) Only Channel 1 can support data trace.

22.3 Module Operation

22.3.1 General Operation

There are four channels in MCRC controller and for each channel there is a memory mapped PSA (Parallel Signature Analysis) Signature Register and a memory mapped CRC (Cyclic Redundancy Check) Value register. A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be a 8, 16, 32, or 64 bit data. MCRC module performs the signature calculation and compares the signature to a pre-determined value. The PSA Signature Register compresses an incoming data pattern into a signature when it is written. When one sector of data patterns are written into PSA Signature Register, a final signature corresponding to the sector is obtained. CRC Value Register stores the pre-determined signature corresponding to one sector of data patterns. The calculated signature and the pre-determined signature are then compared to each other for signature verification. To minimize CPU's involvement, data patterns transfer can be carried out at the background of CPU using DMA controller. DMA is setup to transfer data from memory of which the contents to be verified to the memory mapped PSA Signature Register. When DMA transfers data to the memory mapped PSA Signature Register, a signature is generated. A programmable 20-bit data pattern counter is used for each channel to define the number of data patterns to calculate for each sector. Signature verification can be performed automatically by MCRC controller in AUTO mode or by CPU itself in Semi-CPU or Full-CPU mode. In AUTO mode, a self sustained CRC signature calculation can be achieved without any CPU intervention. MCRC Controller also provides data trace capability. Channel 1 can perform data trace on CPU data bus. During data trace, channel 1 monitors any data being read on CPU data bus and compresses it. When data trace is enabled for channel 1, all circuits related to DMA request and interrupt generation and counters are disabled.

22.3.2 CRC Modes of Operation

MCRC Controller can operate in AUTO, Semi-CPU and Full-CPU modes.

22.3.2.1 AUTO Mode

In AUTO mode, MCRC Controller in conjunction with DMA controller can perform CRC totally without CPU intervention. A sustained transfer of data to both the PSA Signature Register and CRC Value Register are performed in the background of CPU. When a mismatch is detected, an interrupt is generated to CPU. A 16 bit current sector ID register is provided to identify which sector causes a CRC failure.

22.3.2.2 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register. Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register and compare it to a pre-determined CRC value.

22.3.2.3 Full CPU Mode

In Full-CPU mode, the CPU does the data patterns transfer and signature verification all by itself. When CPU has enough throughput, it can perform data patterns transfer by reading data from the memory system to the PSA Signature Register. After certain number of data patterns are compressed, the CPU can read from the PSA Signature Register and compare the calculated signature to the pre-determined CRC signature value. In Full-CPU mode, neither interrupt nor DMA request is generated. All counters are also disabled.

22.3.2.4 Data Trace Mode

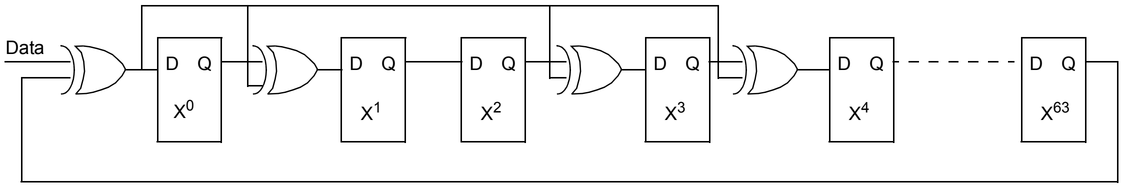
CRC channel 1 can be used to snoop the CPU read data bus. Data read on the CPU bus is converted into a 64-bit data value where the LSB or the MSB bits of the read data bus are padded with zeros. The PSA signature register concatenates the data value with zeros to create a 64-bit word based on the address of the read transaction. For example, the data value 0x12345678 is stored at addresses 0x0 and 0x4. If the CPU reads a word from address 0x0, the value 0x0000000012345678 will be used for the CRC signature generation. If the CPU reads a word from address 0x4, the value 0x1234567800000000 will be used for the signature generation. Each data pattern read by the CPU on its data bus is compressed in the PSA signature register. To enable the data trace on the read bus, 0x0 should be written in CH1_MODE bits. The CH1TRACEEN bit should be set to 0x1. Before the data compression begins, a seed value should be placed in the PSA signature register. To change the seed value, the CH1_MODE bits should be set to 0x00 and data trace bit should be disabled. During data trace mode, all interrupts and DMA request logic are inactive.

22.3.3 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).

$$f(x) = x^{64} + x^4 + x^3 + x + 1 \tag{EQ 1}$$

Figure 22-2. LFSR



The serial implementation of LFSF 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.

The parallel CRC calculation based on the above polynomial can be illustrated in the below 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;
```

Notes:

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. 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. Other modes other than Data Capture will result with the data compressed by PSA Signature Register when it is written. Each channel can be planted with different seed value before compression starts. When PSA Signature Register is read, it gives the calculated signature.

MCRC Controller should be used in conjunction with the on chip DMA controller to produce optimal system performance. The incoming data pattern to PSA Signature Register is typically initiated by the DMA master. When DMA is properly setup, it would read data from the pre-determined memory system and write them to the memory mapped PSA Signature Register. Each time PSA Signature Register is written a signature is generated. CPU itself can also perform data transfer by reading from the memory system and perform write operation to PSA Signature Register if CPU has enough throughput to handle data patterns transfer.

After system reset and when AUTO mode is enabled, MCRC Controller automatically generates a DMA request to request the pre-determined CRC value corresponding to the first sector of memory to be checked.

In AUTO mode, when one sector of data patterns is compressed, the signature stored at the PSA Signature Register is first copied to the PSA Sector Signature Register and PSA Signature Register is then cleared out to all zeros. An automatic signature verification is then performed by comparing the signature stored at the PSA Sector Signature Register to the CRC Value Register. After the comparison the MCRC Controller can generate a DMA request. Upon receiving the DMA request the DMA controller will update the CRC Value Register by transferring the next pre-determined signature value associated with the next sector of memory system. If the signature verification fails then MCRC Controller can generate a CRC fail interrupt.

In Full-CPU mode, no DMA request and interrupt are generated at all. The number of data patterns to be compressed is determined by CPU itself. Full-CPU mode is useful when DMA controller is not available to perform background data patterns transfer. The OS can periodically generate a software interrupt to CPU and use CPU to accomplish data transfer and signature verification.

MCRC Controller supports doubleword, word, half word and byte access to the PSA Signature Register. During a non-doubleword write access, all unwritten byte lanes are padded with zero’s before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register is always in 64 bit because a compressed value is always expressed in 64 bit.

There is a software reset per channel for PSA Signature Register. When set, the PSA Signature Register is reset to all zeros.

PSA Signature Register is reset to zero under the following conditions,

– System reset
– PSA Software reset
– One sector of data patterns are compressed

22.3.4 PSA Sector Signature Register

After one sector of data is compressed, the final resulting signature calculated by PSA Signature Register is transferred to the PSA Sector Signature Register. PSA Signature Register is a read only register. During Semi-CPU mode, the host CPU should read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will in the ISR read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enable to generate when overrun condition occurs.

22.3.5 CRC Value Register

Associated with each channel there is a CRC Value Register. The CRC Value Register stores the pre-determined CRC value. After one sector of data patterns is compressed by PSA Signature Register, MCRC Controller can automatically compare the resulting signature stored at the PSA Sector Signature Register with the pre-determined value stored at the CRC Value Register if AUTO mode is enabled. If the signature verification fails, MCRC Controller can be enabled to generate an CRC fail interrupt. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. CPU should not read from the CRC Value Register during Semi-CPU or Full-CPU mode because the CRC Value Register is not updated during these two modes.

In AUTO mode, for first sector's signature, DMA request is generated when mode is programmed to AUTO. For subsequent sectors, DMA request is generated after each sector is compressed. Responding to the DMA request, DMA controller reloads the CRC Value Register for the next sector of memory system to be checked.

When CRC Value Register is updated with a new CRC value, an internal flag is set to indicate that CRC Value Register contains the most current value. This flag is cleared when CRC comparison is performed. Each time at the end of the final data pattern compression of a sector, MCRC Controller first checks to see if the corresponding CRC Value Register has the most current CRC value stored in it by polling the flag. If the flag is set then the CRC comparison can be performed. If the flag is not set then it means the CRC Value Register contains stale information. A CRC underrun interrupt is generated. When an underrun condition is detected, signature verification is not performed.

MCRC Controller supports doubleword, word, half word and byte access to the CRC Value Register. As noted before comparison between PSA Sector Signature Register and CRC Value Register during AUTO mode is carried out in 64 bit.

22.3.6 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.

22.3.7 Example DMA Controller Setup

DMA controller needs to be setup properly in either AUTO or Semi-CPU mode as DMA controller is used to transfer data patterns. Hardware or a combination of hardware and software DMA triggering are supported.

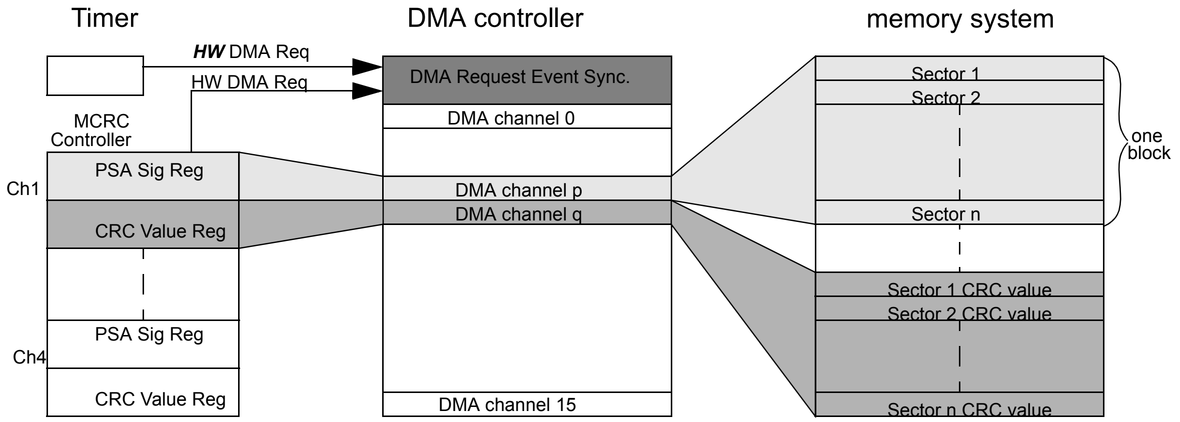
22.3.7.1 AUTO Mode Using Hardware Timer Trigger

There are two DMA channels associated with each CRC channel when in AUTO mode. One DMA channel is setup to transfer data patterns from the source memory to the PSA Signature Register. The second DMA channel is setup to transfer the pre-determined signature to the CRC Value Register. The trigger source for the first DMA channel can be either by hardware or by software. As illustrated in Figure 22-3 a timer can be used to trigger a DMA request to initiate transfer from the source memory system to PSA Signature Register. In AUTO mode, MCRC Controller also generates DMA request after one sector of data patterns is compressed to initiate transfer of the next CRC value corresponding to the next sector of memory. Thus a new CRC value is always updated in the CRC Value Register by DMA synchronized to each sector of memory.

A block of memory system is usually divided into many sectors. All sectors are the same size. The sector size is programmed in the CRC_PCOUNT_REGx and the number of sectors in one block is programmed in the CRC_SCOUNT_REGx of the respective channel. CRC_PCOUNT_REGx multiplies CRC_SCOUNT_REGx and multiplies transfer size of each data pattern should give the total block size in number of bytes.

The total size of the memory system to be examined is also programmed in the respective transfer count register inside DMA module. The DMA transfer count register is divided into two parts. They are element count and frame count. Note that a HW DMA request can be programmed to trigger either one frame or one entire block transfer. In Figure 22-3 a HW DMA request from a timer is used as a trigger source to initiate DMA transfer. If all four CRC channels are active in AUTO mode then a total of four DMA requests would be generated by MCRC Controller.

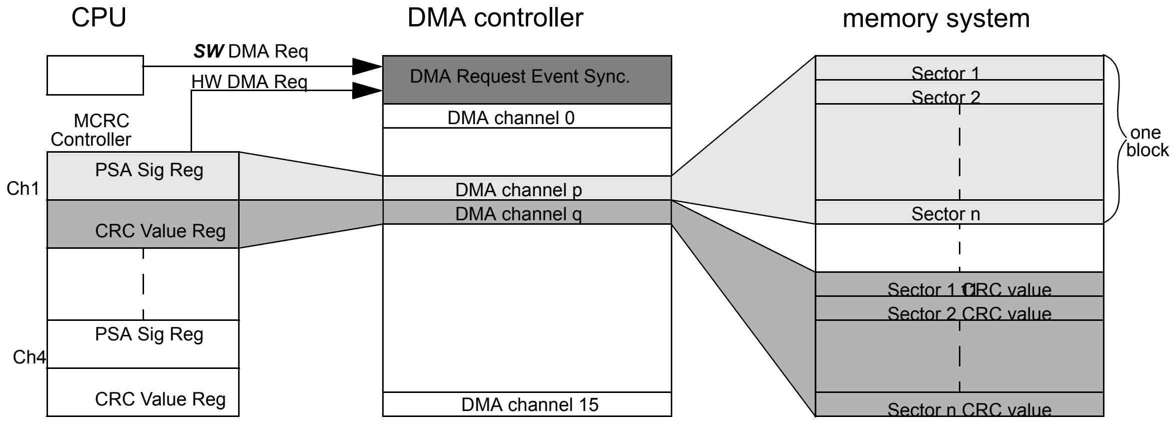
Figure 22-3. AUTO Mode Using Hardware Timer Trigger



22.3.7.2 AUTO Mode Using Software Trigger

The data patterns transfer can also be initiated by software. CPU can generate a software DMA request to activate the DMA channel to transfer data patterns from source memory system to the PSA Signature Register. To generate a software DMA request CPU needs to set the corresponding DMA channel in the DMA software trigger register. Note that just one software DMA request from CPU is enough to complete the entire data patterns transfer for all sectors. Please see Figure 22-4 for illustration.

Figure 22-4. AUTO Mode With Software CPU Trigger



22.3.7.3 Semi-CPU Mode Using Hardware Timer Trigger

During Semi-CPU mode, no DMA request is generated by CRC controller. Therefore, no DMA channel is allocated to update CRC Value Register. CPU should not read from CRC Value Register in semi-CPU mode as it contains stale value. Note that no signature verification is performed at all during this mode. Similar to AUTO mode, either by hardware or by software DMA request can be used as a trigger for data patterns transfer. Figure 1–5. illustrates the DMA setup using semi-CPU mode with hardware timer trigger.

Figure 22-5. Semi-CPU Mode With Hardware Timer Trigger

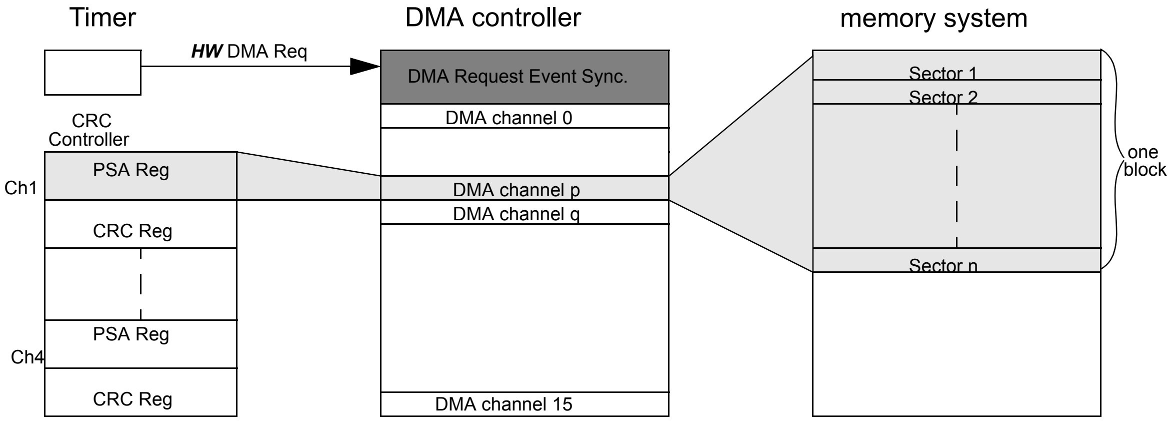


Table 22-1. CRC modes in which DMA request and counter logic are active or inactive

	DMA request	Pattern counter	Sector counter	Timeout counter
AUTO	active	active	active	active
Semi-CPU	inactive	active	active	active
Full-CPU	inactive	inactive	inactive	inactive

22.3.8 Pattern Count Register

There is a 20-bit data pattern counter for every CRC channel. The data pattern counter is a down counter and can be pre-loaded with a programmable value stored in the Pattern Count Register. When the data pattern counter reaches zero, a compression complete interrupt is generated in Semi-CPU mode and an automatic signature verification is performed in AUTO mode. In AUTO only, DMA request is generated to trigger the DMA controller to update the CRC Value Register.

Note: The data pattern count should be divisible by the total transfer count as programmed in DMA controller. The total transfer count is the product of element count and frame count.

22.3.9 Sector Count Register/Current Sector Register

Each channel contains a 16 bit sector counter. The sector count register stores the number of sectors. Sector counter is a free running counter and is incremented by one each time when one sector of data patterns is compressed. When the signature verification fails, the current value stored in the sector counter is saved into current sector register. If signature verification fails, CPU can read from the current sector register to identify the sector which causes the CRC mismatch. To aid and facilitate the CPU in determining the cause of a CRC failure it is advisable to use the following equation during CRC and DMA setup.

$$\text{CRC Pattern Count x CRC Sector Count} = \text{DMA Element Count x DMA Frame Count} \qquad \text{(EQ 2)}$$

The current sector register is frozen from being updated until both the current sector register is read and CRC fail status bit is cleared by CPU. If CPU does not respond to the CRC failure in a timely manner before another sector produces a signature verification failure, the current sector register is not updated with the new sector number. An overrun interrupt is generate instead. If current sector register is already frozen with an erroneous sector and emulation is entered with SUSPEND signal goes to high then the register still remains frozen even it is read.

In Semi-CPU mode, the current sector register is used to indicate the sector for which the compression complete has last happened.

Current sector register is reset when the PSA software reset is enabled.

Note: Both data pattern count and sector count registers must be greater than or equal to one for the counters to count. After reset, pattern count and sector count registers default to zero and the associated counters are inactive.

22.3.10Interrupt

CRC generate several types of interrupts per channel. Associated with each interrupt there is a interrupt enable bit. No interrupt is generated in Full-CPU mode.

– Compression complete interrupt
– CRC fail interrupt
– Overrun interrupt
– Underrun interrupt
– Timeout interrupt

Table 22-2. Modes in which interrupt condition can occur

	AUTO	Semi-CPU	Full-CPU
Compression Complete	no	yes	no
CRC Fail	yes	no	no
Overrun	yes	yes	no
Underrun	yes	no	no
Timeout	yes	yes	no

22.3.10.1 Compression complete interrupt

Compression complete interrupt is generated in Semi-CPU mode only. When the data pattern counter reaches zero, the compression complete flag is set and the interrupt is generated..

22.3.10.2 CRC Fail Interrupt

CRC fail interrupt is generated in AUTO mode only. When the signature verification fails, the CRC fail flag is set,. CPU should take action to address the fail condition and clear the CRC fail flag after it resolves the CRC mismatch.

22.3.10.3 Overrun Interrupt

Overrun Interrupt is generated in either AUTO or Semi-CPU mode. During AUTO mode, if a CRC fail is detected then the current sector number is recorded in the current sector register. If CRC fail status bit is not cleared and current sector register is not read by the host CPU before another CRC fail is detected for another sector then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time the signature is copied into the PSA Sector Signature Register. If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value then an overrun interrupt is generated.

22.3.10.4 Underrun Interrupt

Underrun interrupt only occurs in AUTO mode. The interrupt is generated when the CRC Value Register is not updated with the corresponding signature when the data pattern counter finishes counting. During AUTO mode, MCRC Controller generates DMA request to update CRC Value Register in synchronization to the corresponding sector of the memory. Signature verification is also performed if underrun condition is detected. And CRC fail interrupt is generated at the same time as the underrun interrupt.

22.3.10.5 Timeout Interrupt

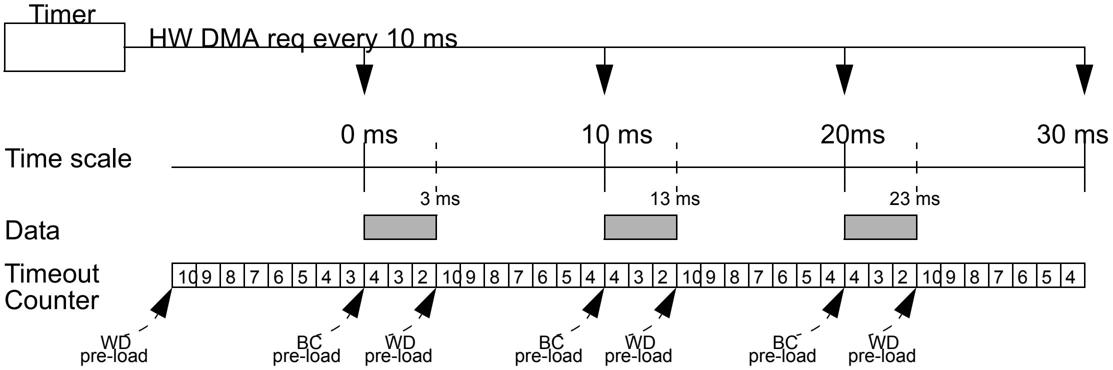
To ensure that the memory system is examined within a pre-defined time frame and no loss of incoming data there is a 24 bit timeout counter per CRC channel. The 24 bit timeout down counter can be pre-loaded with two different pre-load values, watchdog timeout pre-load value (CRC_WDTOPLDx) and block complete timeout pre-load value (CRC_BCTOPLDx). The timeout counter is clocked by a prescaler clock which is permanently running at division 64 of HCLK clock.

Watchdog timeout pre-load register (CRC_WDTOPLDx) is used to check if DMA does supply a block of data responding to a request in a given time frame. Block complete timeout pre-load register (CRC_BCTOPLDx) is used to check if one complete block of data patterns are compressed within a specific time frame. The timeout counter is first pre-loaded with CRC_WDTOPLDx after either AUTO or Semi-CPU mode is selected and starts to down count. If the timeout counter expires before DMA transfers any data pattern to PSA Signature Register then a timeout interrupt is generated. An incoming data pattern before the timeout counter expires will automatically pre-load the timeout counter with CRC_BCTOPLDx the block complete timeout pre-load value.

Block complete timeout pre-load value is used to check it one block of data patterns are compressed within a given time limit. If the timeout counter pre-loaded with CRC_BCTOPLDx value expires before one block of data patterns are compressed a timeout interrupt is generated. When one block (pattern count x sector count) of data patterns are compressed before the counter has expired, the counter is pre-loaded with CRC_WDTOPLDx value again. If the timeout counter is pre-loaded with zero then the counter is disable and no timeout interrupt is generated.

In Figure 22-6 a timer generates DMA request every 10ms to trigger one block (pattern count x sector count) transfer. Since we want to make sure that DMA does start to transfer a block every 10 ms we would set the first pre-load value to 10ms in CRC_WDTOPLDx. We also want to make sure that one block of data patterns are compressed within 4ms. With such a requirement we would set the second pre-load value to 4ms in CRC_BCTOPLDx register.

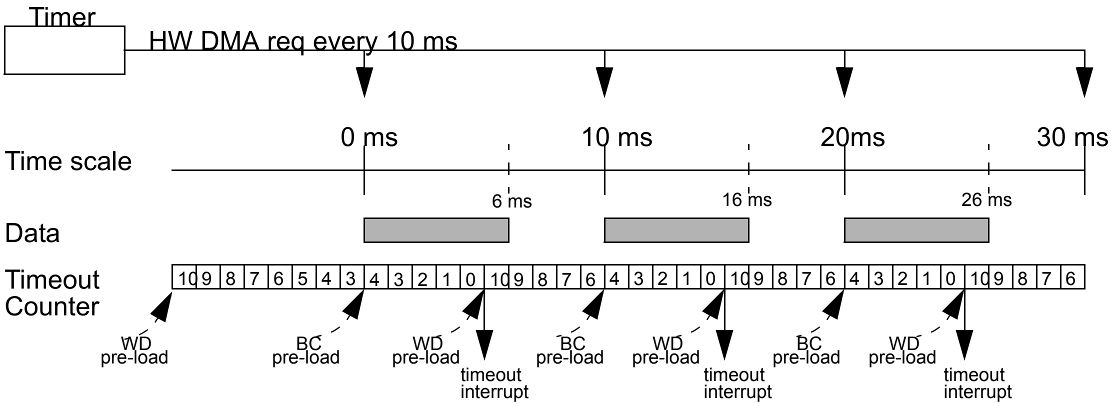
Figure 22-6. Timeout Example 1



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: No timeout interrupt is generated in this example since each block of data patterns are compressed in 3 ms and DMA does initiate a block transfer every 10 ms.

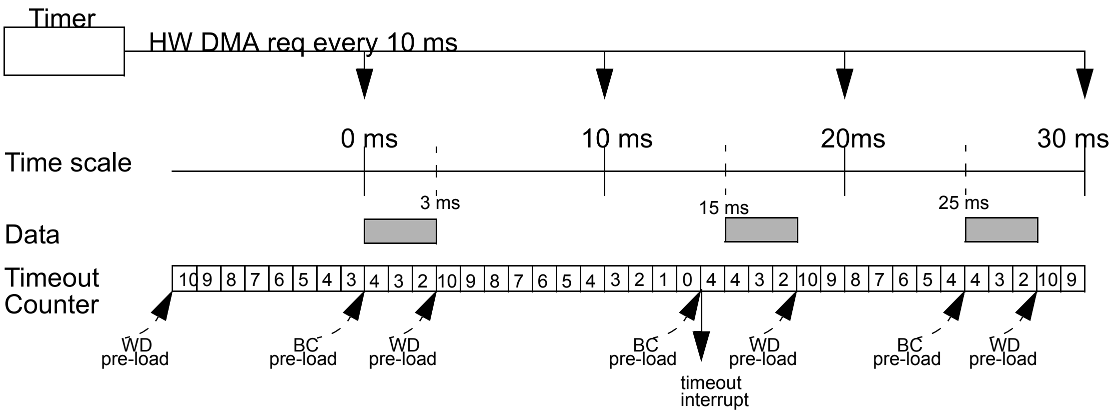
Figure 22-7. Timeout Example 2



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: Timeout interrupt is generated in this example since each block of data patterns are compressed in 6 ms and this is out of the 4ms time frame.

Figure 22-8. Timeout Example 3



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: Timeout interrupt is generated in this example since DMA can not transfer the second block of data within 10ms time limit and the reason may be that DMA is set up in fixed priority scheme and DMA is serving other higher priority channels at the time before it can service the timer request.

22.3.10.6 Interrupt Offset Register

MCRC Controller only generates one interrupt request to interrupt manager. A interrupt offset register is provided to indicate the source of the pending interrupt with highest priority. Table 22-3 shows the offset interrupt vector address of each interrupt condition in an ascending order of priority.

Table 22-3. Interrupt offset mapping

Interrupt Condition	Offset Value
Phantom	0x0
Ch1 CRC Fail	0x1

Ch2 CRC Fail	0x2
Ch3 CRC Fail	0x3
Ch4 CRC Fail	0x4
reserved	0x5-0x8
Ch1 Compression Complete	0x9
Ch2 Compression Complete	0xa
Ch3 Compression Complete	0xb
Ch4 Compression Complete	0xc
reserved	0xd-0x10
Ch1 Overrun	0x11
Ch2 Overrun	0x12
Ch3 Overrun	0x13
Ch4 Overrun	0x14
reserved	0x15-0x18
Ch1 Underrun	0x19
Ch2 Underrun	0x1a
Ch3 Underrun	0x1b
Ch4 Underrun	0x1c
reserved	0x1d-0x20
Ch1 Timeout	0x21
Ch2 Timeout	0x22
Ch3 Timeout	0x23
Ch4 Timeout	0x24

22.3.10.7 Error Handling

When an interrupt is generated, host CPU should take appropriate actions to identify the source of error and restart the respective channel in DMA and MCRC module. To restart a CRC channel user should perform the following steps in the ISR,

1. Write to software reset bit in *CRC_CTRL* register to reset the respective PSA Signature Register.
2. Reset the CHx_MODE bits to “00” in *CRC_CTRL* register as Data capture mode.
3. Set the CHx_MODE bits in *CRC_CTRL* register to desired new mode again.
4. Release software reset.

The host CPU should use byte write to restart each individual channel.

22.3.11CPU Data Trace

CRC channel 1 can be used to snoop Flash, System RAM and Peripheral Bus Master Data buses. However, at any one point only one bus is snooped. It is possible to disable the snooping of any of the buses by programming MCRC_BUS_SEL register. While snooping the data, there is a priority scheme implemented between the buses. Peripheral Bus Master has the highest priority followed by FLash, Even System RAM and Odd System RAM. For each data read by CPU on its data bus the same data is compressed in the PSA Signature Register. A write to PSA Signature Register does not get compressed. Therefore, it is possible to write a seed value into PSA Signature Register before the bus snooping takes place. During data trace mode, all interrupts and DMA request logic are inactive. For non double word read on the data bus, all un-selected byte lanes are padded with zero during compression.

22.3.12Power Down Mode

MCRC 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 MCRC controller is in power down mode, no data tracing alone will happen. However if MCRC registers are accessed then data trace happens from channel 1.

22.3.13Emulation

A read access from a register in functional mode can sometimes trigger a certain internal event to follow. For example, reading an interrupt offset register triggers an event to clear the corresponding interrupt status flag. During emulation when SUSPEND signal is high, a read access from any register should only return the register contents to the bus and should not trigger or mask any event as it would have in functional mode. This is to prevent debugger from reading the interrupt offset register, i.e. during refreshing screen and cause the corresponding interrupt status flag to get cleared. Timeout counters are stopped to generate timeout interrupts in emulation mode. No Peripheral Master bus error should be generated if reading from the unimplemented locations. When channel 1 is placed under data trace, the PSA Signature Register does not compress any data read on CPU data bus when suspend is active.

22.3.14Peripheral Bus Interface

MCRC is a Peripheral slave module. The register interface is similar to other peripheral modules. MCRC supports following features,

- Different sizes of burst operation.
- Aligned and unaligned accesses.
- Returns "Address Error" if the address accessed is beyond specific region. Refer Register table for valid address region.
- CEMUDEBUG signal is used to put MCRC in suspend mode.

22.4 Example

This section illustrates several of the ways in which the TMS470Mx MCRC Controller can be utilized to perform CRC.

22.4.1 Example: Auto Mode Using Time Based Event Triggering

A large memory area with 2Mbyte(256k doubleword) is to be checked in the background of CPU. CRC is to be performed every 1K byte(128 doubleword). Therefore there should be 2048 pre-recorded CRC values. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Let's assume all DMA transfers are carried out in 64-bit transfer size.

22.4.1.1 DMA Setup

- Setup DMA channel 1 with the starting address from which the pre-determined CRC values are stored. Setup the destination address to the memory mapped channel 1 CRC Value Register. Put the source address at post increment addressing mode and put the destination address at constant addressing mode. Use ***hardware*** DMA request for channel 1 to trigger a ***frame*** transfer.
- Setup DMA channel 2 with the source address from which the contents of memory to be verified. Setup the destination address to the memory mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 2048. Put the source address at post increment addressing mode and put the destination address at constant address mode. Use ***hardware*** DMA request for channel 2 to trigger an entire ***block*** transfer.

22.4.1.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

- Setup Timer to generate DMA request associated with DMA channel 2. For example, an OS can setup the timer to generate a DMA request every 10ms.

22.4.1.3 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 2048.
- For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load (CRC_BCTOPLDx) value to 15625(5 ms / (1 HCLK period x 64)) if CRC is operating at 200Mhz.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected MCRC Controller automatically generates a DMA request on channel 1. Around the same time the timer module also generates a DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. MCRC Controller generates a DMA request on DMA channel 1 when one sector of data patterns are compressed. This routine will continue until the entire 2Mbyte are consumed. If the timeout counter reached zero before the entire 2Mbytes are compressed a timeout interrupt is generated. After 2M are transferred, the DMA can generate an interrupt to CPU. The entire operation will continue again when DMA responds to the DMA request from both the timer and MCRC Controller. The CRC is performed totally without any CPU intervention.

22.4.2 Example: Auto Mode Without Using Time Based Triggering

A small but highly secured memory area with 1kbytes is to be checked in the background of CPU. CRC is to be performed every 1Kbytes. Therefore there is only one pre-recorded CRC value. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Assume all transfers carried out by DMA are in 64 bit transfer size.

22.4.2.1 DMA Setup

- Setup DMA channel 1 with the source address from which the pre-determined CRC value is stored. Setup the destination address to the memory mapped channel 1 CRC Value Register. Put the source address at constant addressing mode and put the destination address at constant addressing mode. Use ***hardware*** DMA request for channel 1.
- Setup DMA channel 2 with the source address from which the memory area to be verified. Setup the destination address to the memory mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 1. Put the source address at post increment addressing mode and put the destination address at constant address mode. Generate a ***software*** DMA request on channel 2 after CRC has completed its setup. Enable autoinitiation for DMA channel 2.

22.4.2.2 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 1.
- Leaving the timeout count register with the reset value of zero means no timeout interrupt is generated.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected the MCRC Controller automatically generates a DMA request on channel 1. At the same time the CPU generates a ***software*** DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. MCRC Controller generate a DMA request on DMA channel 1 again after one sector is compressed. After 1kbytes are transferred, the DMA can generate an interrupt to CPU. Responding to the DMA interrupt CPU can restart the CRC routine by generating a software DMA request onto channel 2 again.

22.4.3 Example: Semi-CPU Mode

If DMA controller is available in a system the CRC module can also operate in semi-CPU mode. This means that CPU can still make use of the DMA to perform data patterns transfer to CRC controller in the background. The difference between semi-CPU mode and AUTO mode is that CRC controller does not automatically perform the signature verification. CRC controllers generates a compression complete interrupt to CPU when the one sector of data patterns are compressed. CPU needs to perform the signature verification itself.

A memory area with 2Mbyte is to be verified with the help of the CPU. CRC operation is to be performed every 1K byte. Since there are 2Mbyte(256k doublewords) of memory to be check and we want to perform a CRC every 1Kbyte(128 doublewords) and therefore there should be 2048 pre-recorded CRC values. In Semi-CPU mode, the CRC Value Register is not updated and contains indeterminate data.

22.4.3.1 DMA Setup

Setup DMA channel 1 with the source address from which the memory area to be verified are mapped. Setup the destination address to the memory mapped channel 1 PSA Signature Register. Put the starting address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request to trigger an entire block transfer for channel 1. Disable autoinitiation for DMA channel 1.

22.4.3.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

Setup Timer to generate DMA request associated with DMA channel 1. For example, an OS can setup the timer to generate a DMA request every 10ms.

22.4.3.3 CRC Setup

– Program the pattern count to 128.
– Program the sector count to 2048.
– For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load value to 15625(5 ms / (1 HCLK period x 64)) if CRC is operating at 200Mhz.
– Enable Semi-CPU mode and enable all interrupts.

The timer module first generates a DMA request on DMA channel 1 when it is enabled. When the first incoming data pattern arrives at the PSA Signature Register, the CRC controller will compress it. After one sector of data patterns are compressed, the CRC controller generate a compression complete interrupt. Upon responding to the interrupt the CPU would read from the PSA Sector Signature Register. It is up to the CPU on how to deal with the PSA value just read. It can compare it to a known signature value or it can write it to another memory location to build a signature file or even transfer the signature out of the device via SCI or SPI. This routine will continue until the entire 2Mbyte are consumed. The latency of the interrupt response from CPU can cause overrun condition. If CPU does not read from PSA Sector Signature Register before the PSA value is overridden with the signature of the next sector of memory, an overrun interrupt will be generated by CRC controller.

22.4.4 Example: Full-CPU Mode

In a system without the availability of DMA controller, the CRC routine can be operated by CPU provided the CPU has enough throughput. CPU needs to read from the memory area from which CRC is to be performed.

A memory area with 2Mbyte is to be checked with the help of the CPU. CRC operation is to be performed every 1K byte. In CPU mode, the CRC Value Register is not updated and contains indeterminate data.

22.4.4.1 CRC Setup

– All control registers can be left in their reset state. Only enable Full-CPU mode.

CPU itself reads from the memory and write the data to the PSA Signature Register inside MCRC Controller. When the first incoming data pattern arrives at the PSA Signature Register, the MCRC Controller will compress it. After ***n*** data patterns are compressed, CPU can read from the PSA Signature Register. It is up to the CPU on how to deal with the PSA signature value just read. It can compare it to a known signature value stored at another memory location.

22.5 CRC Control Registers.

All registers are in word boundary. 64, 32, 16, and 8 bit write accesses are supported to all registers. During double word, write CADDRESS[2:0] are ignored from decoding. A double word write access can write to two 32-bit words at the same time.

Write access takes one Peripheral clock and read access takes two Peripheral clock.

Figure 22-9. CRC Control Register

<table>
<tr><th>Offset Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0x00
CRC_CTRL0
Page 1673</td><td colspan="7">Reserved</td><td>CH4_PSA_SWREST</td><td colspan="7">Reserved</td><td>CH3_PSA_SWREST</td></tr>
<tr><td colspan="7">Reserved</td><td>CH2_PSA_SWREST</td><td colspan="7">Reserved</td><td>CH1_PSA_SWREST</td></tr>
<tr><td rowspan="2">0x08
CRC_CTRL1
Page 1675</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="15"></td><td>PWDN</td></tr>
<tr><td rowspan="2">0x10
CRC_CTRL2
Page 1676</td><td colspan="6">Reserved</td><td colspan="2">CH4_MODE</td><td colspan="6">Reserved</td><td colspan="2">CH3_MODE</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="2">CH2_MODE</td><td colspan="3">Reserved</td><td>CH1_TRACEEN</td><td colspan="2">Reserved</td><td colspan="2">CH1_MODE</td></tr>
<tr><td rowspan="2">0x18
CRC_INTS
Page 1678</td><td colspan="3">Reserved</td><td>CH4_TIMEOUTENS</td><td>CH4_UNDERENS</td><td>CH4_OVERENS</td><td>CH4_CRCFAILENS</td><td>CH3_CCITENS</td><td colspan="3">Reserved</td><td>CH3_TIMEOUTENS</td><td>CH3_UNDERENS</td><td>CH3_OVERENS</td><td>CH3_CRCFAILENS</td><td>CH3_CCITENS</td></tr>
<tr><td colspan="3">Reserved</td><td>CH2_TIMEOUTENS</td><td>CH2_UNDERENS</td><td>CH2_OVERENS</td><td>CH2_CRCFAILENS</td><td>CH1_CCITENS</td><td colspan="3">Reserved</td><td>CH1_TIMEOUTENS</td><td>CH1_UNDERENS</td><td>CH1_OVERENS</td><td>CH1_CRCFAILENS</td><td>CH1_CCITENS</td></tr>
<tr><td rowspan="2">0x20
CRC_INTR
Page 1685</td><td colspan="3">Reserved</td><td>CH4_TIMEOUTENR</td><td>CH4_UNDERENR</td><td>CH4_OVERENR</td><td>CH4_CRCFAILENR</td><td>CH4_CCITENR</td><td colspan="3">Reserved</td><td>CH3_TIMEOUTENR</td><td>CH3_UNDERENR</td><td>CH3_OVERENR</td><td>CH3_CRCFAILENR</td><td>CH3_CCITENR</td></tr>
<tr><td colspan="3">Reserved</td><td>CH2_TIMEOUTENR</td><td>CH2_UNDERENR</td><td>CH2_OVERENR</td><td>CH2_CRCFAILENR</td><td>CH2_CCITENR</td><td colspan="3">Reserved</td><td>CH1_TIMEOUTENR</td><td>CH1_UNDERENR</td><td>CH1_OVERENR</td><td>CH1_CRCFAILENR</td><td>CH1_CCITENR</td></tr>
<tr><td rowspan="2">0x28
CRC_STATUS
Page 1692</td><td colspan="3">Reserved</td><td>CH4_TIMEOUT</td><td>CH4_UNDER</td><td>CH4_OVER</td><td>CH4_CRCFAIL</td><td>CH4_CCIT</td><td colspan="3">Reserved</td><td>CH3_TIMEOUT</td><td>CH3_UNDER</td><td>CH3_OVER</td><td>CH3_CRCFAIL</td><td>CH3_CCIT</td></tr>
<tr><td colspan="3">Reserved</td><td>CH2_TIMEOUT</td><td>CH2_UNDER</td><td>CH2_OVER</td><td>CH2_CRCFAIL</td><td>CH2_CCIT</td><td colspan="3">Reserved</td><td>CH1_TIMEOUT</td><td>CH1_UNDER</td><td>CH1_OVER</td><td>CH1_CRCFAIL</td><td>CH1_CCIT</td></tr>
</table>

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
0x30 CRC_INT_OFFSET_REG Page 1696	Reserved															
	Reserved								OFSTREG							
0x38 CRC_BUSY Page 1697	Reserved							CH4_BUSY	Reserved							CH3_BUSY
	Reserved							CH2_BUSY	Reserved							CH1_BUSY
0x40 CRC_PCOUNT_REG1 Page 1698	Reserved												CRC_PAT_COUNT1[19:16]			
	CRC_PAT_COUNT1[15:0]															
0x44 CRC_SCOUNT_REG1 Page 1699	Reserved															
	CRC_SEC_COUNT1[15:0]															
0x48 CRC_CURSEC_REG1 Page 1700	Reserved															
	CRC_CURSEC1[15:0]															
0x4C CRC_WDTOPLD1 Page 1701	Reserved								CRC_WDTOPLD1[23:16]							
	CRC_WDTOPLD1[15:0]															
0x50 CRC_BCTOPLD1 Page 1702	Reserved								CRC_BCTOPLD1[23:16]							
	CRC_BCTOPLD1[15:0]															
0x60 PSA_SIGREGL1 Page 1703	PSASIG1[31:16]															
	PSASIG1[15:0]															

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
0x64 PSA_SIGREGH1 Page 1704	PSASIG1[63:48]															
	PSASIG1[47:32]															
0x68 CRC_REGL1 Page 1705	CRC1[31:16]]															
	CRC1[15:0]															
0x6C CRC_REGH1 Page 1706	CRC1[63:48]															
	CRC1[47:32]															
0x70 PSA_SECSIGREGL1 Page 1707	PSASECSIG1[31:16]															
	PSASECSIG1[15:0]															
0x74 PSA_SECSIGREGH1 Page 1708	PSASECSIG1[63:48]															
	PSASECSIG1[47:32]															
0x78 RAW_DATAREGL1 Page 1709	RAW_DATA1[31:16]															
	RAW_DATA1[15:0]															
0x7C RAW_DATAREGH1 Page 1710	RAW_DATA1[63:48]															
	RAW_DATA1[47:32]															
0x80 CRC_PCOUNT_REG2 Page 1711	Reserved												CRC_PAT_COUNT2[19:16]			
	CRC_PAT_COUNT2[15:0]															

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
0x84 CRC_SCOUNT_REG2 Page 1712	Reserved															
	CRC_SEC_COUNT2[15:0]															
0x88 CRC_CURSEC_REG2 Page 1713	Reserved															
	CRC_CURSEC2[15:0]															
0x8C CRC_WDTOPLD2 Page 1714	Reserved								CRC_WDTOPLD2[23:16]							
	CRC_WDTOPLD2[15:0]															
0x90 CRC_BCTOPLD2 Page 1715	Reserved								CRC_BCTOPLD2[23:16]							
	CRC_BCTOPLD2[15:0]															
0xA0 PSA_SIGREGL2 Page 1716	PSASIG2[31:16]															
	PSASIG2[15:0]															
0xA4 PSA_SIGREGH2 Page 1717	PSASIG2[63:48]															
	PSASIG2[47:32]															
0xA8 CRC_REGL2 Page 1718	CRC2[31:16]]															
	CRC2[15:0]															
0xAC CRC_REGH2 Page 1719	CRC2[63:48]															
	CRC2[47:32]															

<table>
<tr><th>Offset Address
Register</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0xB0
PSA_SECSIGREGL2
Page 1720</td><td colspan="16">PSASECSIG2[31:16]</td></tr>
<tr><td colspan="16">PSASECSIG2[15:0]</td></tr>
<tr><td rowspan="2">0xB4
PSA_SECSIGREGH2
Page 1721</td><td colspan="16">PSASECSIG2[63:48]</td></tr>
<tr><td colspan="16">PSASECSIG2[47:32]</td></tr>
<tr><td rowspan="2">0xB8
RAW_DATAREGL2
Page 1722</td><td colspan="16">RAW_DATA2[31:16]</td></tr>
<tr><td colspan="16">RAW_DATA2[15:0]</td></tr>
<tr><td rowspan="2">0xBC
RAW_DATAREGH2
Page 1723</td><td colspan="16">RAW_DATA2[63:48]</td></tr>
<tr><td colspan="16">RAW_DATA2[47:32]</td></tr>
<tr><td rowspan="2">0xC0
CRC_PCOUNT_REG3
Page 1724</td><td colspan="12">Reserved</td><td colspan="4">CRC_PAT_COUNT3[19:16]</td></tr>
<tr><td colspan="16">CRC_PAT_COUNT3[15:0]</td></tr>
<tr><td rowspan="2">0xC4
CRC_SCOUNT_REG3
Page 1725</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">CRC_SEC_COUNT3[15:0]</td></tr>
<tr><td rowspan="2">0xC8
CRC_CURSEC_REG3
Page 1726</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">CRC_CURSEC3[15:0]</td></tr>
<tr><td rowspan="2">0xCC
CRC_WDTOPLD3
Page 1727</td><td colspan="8">Reserved</td><td colspan="8">CRC_WDTOPLD3[23:16]</td></tr>
<tr><td colspan="16">CRC_WDTOPLD3[15:0]</td></tr>
</table>

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
0xD0 CRC_BCTOPLD3 Page 1727	Reserved								CRC_BCTOPLD3[23:16]							
	CRC_BCTOPLD3[15:0]															
0xE0 PSA_SIGREGL3 Page 1729	PSASIG3[31:16]															
	PSASIG3[15:0]															
0xE4 PSA_SIGREGH3 Page 1730	PSASIG3[63:48]															
	PSASIG3[47:32]															
0xE8 CRC_REGL3 Page 1731	CRC3[31:16]]															
	CRC3[15:0]															
0xEC CRC_REGH3 Page 1732	CRC3[63:48]															
	CRC3[47:32]															
0xF0 PSA_SECSIGREGL3 Page 1733	PSASECSIG3[31:16]															
	PSASECSIG3[15:0]															
0xF4 PSA_SECSIGREGH3 Page 1734	PSASECSIG3[63:48]															
	PSASECSIG3[47:32]															
0xF8 RAW_DATAREGL3 Page 1735	RAW_DATA3[31:16]															
	RAW_DATA3[15:0]															

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
0xFC RAW_DATAREGH3 Page 1736	RAW_DATA3[63:48]															
	RAW_DATA3[47:32]															
0x100 CRC_PCOUNT_REG4 Page 1737	Reserved												CRC_PAT_COUNT4[19:16]			
	CRC_PAT_COUNT4[15:0]															
0x104 CRC_SCOUNT_REG4 Page 1738	Reserved															
	CRC_SEC_COUNT4[15:0]															
0x108 CRC_CURSEC_REG4 Page 1739	Reserved															
	CRC_CURSEC4[15:0]															
0x10C CRC_WDTOPLD4 Page 1740	Reserved								CRC_WDTOPLD4[23:16]							
	CRC_WDTOPLD4[15:0]															
0x110 CRC_BCTOPLD4 Page 1741	Reserved								CRC_BCTOPLD4[23:16]							
	CRC_BCTOPLD4[15:0]															
0x120 PSA_SIGREGL4 Page 1742	PSASIG4[31:16]															
	PSASIG4[15:0]															
0x124 PSA_SIGREGH4 Page 1743	PSASIG4[63:48]															
	PSASIG4[47:32]															

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x128 CRC_REGL4 Page 1744	CRC4[31:16]]															
	CRC4[15:0]															
0x12C CRC_REGH4 Page 1745	CRC4[63:48]															
	CRC4[47:32]															
0x130 PSA_SECSIGREGL4 Page 1746	PSASECSIG4[31:16]															
	PSASECSIG4[15:0]															
0x134 PSA_SECSIGREGH4 Page 1747	PSASECSIG4[63:48]															
	PSASECSIG4[47:32]															
0x138 RAW_DATAREGL4 Page 1748	RAW_DATA4[31:16]															
	RAW_DATA4[15:0]															
0x13C RAW_DATAREGH4 Page 1749	RAW_DATA4[63:48]															
	RAW_DATA4[47:32]															
0x140 MCRC_BUS_SEL Page 1750	Reserved															
	Reserved													MEn	DTC-MEn	ITC-MEn

22.5.1 CRC Global Control Register 0(CRC_CTRL0)

Figure 22-10. CRC Global Control Register 0(CRC_CTRL0) [offset = 0x00]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							CH4_PSA_SWREST	Reserved							CH3_PSA_SWREST
R-0							RW-0	R-0							RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CH2_PSA_SWREST	Reserved							CH1_PSA_SWREST
R-0							RW-0	R-0							RW-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-4. CRC Global Control Register 0(CRC_CTRL0) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Read returns 0. Writes have no effect.
24	CH4_PSA_SWREST		**Channel 4 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. 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 not reset
		1	PSA Signature Register reset
23–17	Reserved		Read returns 0. Writes have no effect.
24	CH3_PSA_SWREST		**Channel 3 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. 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 not reset
		1	PSA Signature Register reset
15–9	Reserved		Read returns 0. Writes have no effect.
8	CH2_PSA_SWREST		**Channel 2 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. 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 not reset
		1	PSA Signature Register reset

Table 22-4. CRC Global Control Register 0(CRC_CTRL0) Field Descriptions (Continued)

Bit	Name	Value	Description
7–1	Reserved		Read returns 0. Writes have no effect.
0	CH1_PSA_SWREST		**Channel 1 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. 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 not reset
		1	PSA Signature Register reset

22.5.2 CRC Global Control Register (CRC_CTRL1)

Figure 22-11. CRC Global Control Register 1(CRC_CTRL1) [offset = 0x08]

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
															PWDN
R-0															RW-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-5. CRC Global Control Register 1(CRC_CTRL1) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Read returns 0. Writes have no effect.
0	PWDN		**Power Down.** When set, MCRC module is put in power down mode
		0	MCRC is not in power down mode
		1	MCRC is in power down mode

22.5.3 CRC Global Control Register 2(CRC_CTRL2)

Figure 22-10 and Table 22-6 describe this register.

Figure 22-12. CRC Global Control Register 2(CRC_CTRL2) [offset = 0x10]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						CH4_MODE		Reserved						CH3_MODE	
R-0						R/WP-0		R-0						R/WP-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CH2_MODE		Reserved			CH1_TRACEEN	Reserved		CH1_MODE	
R-0						R/WP-0					R/W-P-0			R/WP-0	

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-6. CRC Global Control Register 2(CRC_CTRL2) Field Descriptions

Bit	Name	Value	Description
31–26	Reserved		Read returns 0. Writes have no effect.
25-24	CH4_MODE		Channel 4 Mode Selection
		00	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		01	AUTO Mode
		10	Semi-CPU Mode
		11	Full-CPU Mode
23–18	Reserved		Read returns 0. Writes have no effect.
17-16	CH3_MODE		Channel 3 Mode Selection
		00	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		01	AUTO Mode
		10	Semi-CPU Mode
		11	Full-CPU Mode
15–10	Reserved		Read returns 0. Writes have no effect.

Table 22-6. CRC Global Control Register 2(CRC_CTRL2) Field Descriptions (Continued)

Bit	Name	Value	Description
9-8	CH2_MODE		Channel 2 Mode Selection
		00	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		01	AUTO Mode
		10	Semi-CPU Mode
		11	Full-CPU Mode
7–5	Reserved		Read returns 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 snoops on the CPU Peripheral Bus Master, Flash, System RAM buses 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.
		0	Data Trace disable
		1	Data Trace enable
3–2	Reserved		Read returns 0. Writes have no effect.
1-0	CH1_MODE		Channel 1 Mode Selection
		00	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		01	AUTO Mode
		10	Semi-CPU Mode
		11	Full-CPU Mode

22.5.4 CRC Interrupt Enable Set Register(CRC_INTS)

Figure 22-13. CRC Interrupt Enable Set Register(CRC_INTS) [offset = 0x18]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CH4_TIMEOUTENS	CH4_UNDER-ENS	CH4_OVER-ENS	CH4_CRCFAILENS	CH4_CCITENS	Reserved			CH3_TIMEOUTENS	CH3_UNDER-ENS	CH3_OVER-ENS	CH3_CRCFAILENS	CH3_CCITENS
R-0			R/WP-0					R-0			R/WP-0				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CH2_TIMEOUTENS	CH2_UNDER-ENS	CH2_OVER-ENS	CH2_CRCFAILENS	CH2_CCITENS	Reserved			CH1_TIMEOUTENS	CH1_UNDER-ENS	CH1_OVER-ENS	CH1_CRCFAILENS	CH1_CCITENS
R-0			R/WP-0					R-0			R/WP-0				

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Read returns 0. Writes have no effect.
28	CH4_TIMEOUTENS		**Channel 4 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt enable
27	CH4_UNDERENS		**Channel 4 Underrun Interrupt Enable Bit**. Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt enable

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
26	CH4_OVERENS		**Channel 4 Overrun Interrupt Enable Bit.** Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt enable
25	CH4_CRCFAILENS		**Channel 4 CRC Fail Interrupt Enable Bit.** Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt enable
24	CH4_CCITENS		**Channel 4 Compression Complete Interrupt Enable Bit.** Writing a one to this bit enable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt enable
23–21	Reserved		Read returns 0. Writes have no effect.

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
20	CH3_TIMEOUTENS		**Channel 3 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt enable
19	CH3_UNDERENS		**Channel 3 Underrun Interrupt Enable Bit**. Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt enable
18	CH3_OVERENS		**Channel 3 Overrun Interrupt Enable Bit.** Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt enable

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
17	CH3_CRCFAILENS		**Channel 3 CRC Fail Interrupt Enable Bit.** Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt enable
16	CH3_CCITENS		**Channel 3 Compression Complete Interrupt Enable Bit.** Writing a one to this bit enable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt enable
15–13	Reserved		Read returns 0. Writes have no effect.
12	CH2_TIMEOUTENS		**Channel 2 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt enable

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
11	CH2_UNDERENS		**Channel 2 Underrun Interrupt Enable Bit**. Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt enable
10	CH2_OVERENS		**Channel 2 Overrun Interrupt Enable Bit.** Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt enable
9	CH2_CRCFAILENS		**Channel 2 CRC Fail Interrupt Enable Bit.** Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt enable

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
8	CH2_CCITENS		**Channel 2 Compression Complete Interrupt Enable Bit.** Writing a one to this bit enable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt enable
7–5	Reserved		Read returns 0. Writes have no effect.
4	CH1_TIMEOUTENS		**Channel 1 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt enable
3	CH1_UNDERENS		**Channel 1 Underrun Interrupt Enable Bit**. Writing a one to this bit enable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt enable

Table 22-7. CRC Interrupt Enable Set Register(CRC_INTS) Field Descriptions (Continued)

Bit	Name	Value	Description
2	CH1_OVERENS		**Channel 1 Overrun Interrupt Enable Bit.** Writing a one to this bit enable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt enable
1	CH1_CRCFAILENS		**Channel 1 CRC Fail Interrupt Enable Bit.** Writing a one to this bit enable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt enable
0	CH1_CCITENS		**Channel 1 Compression Complete Interrupt Enable Bit.** Writing a one to this bit enable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt enable

22.5.5 CRC Interrupt Enable Reset Register(CRC_INTR)

Figure 22-14. CRC Interrupt Enable Reset Register(CRC_INTR) [offset = 0x20]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CH4_TIMEOUTENR	CH4_UNDERENR	CH4_OVERENR	CH4_CRCFAILENR	CH4_CCITENR	Reserved			CH3_TIMEOUTENR	CH3_UNDERENR	CH3_OVERENR	CH3_CRCFAILENR	CH3_CCITENR
R-0			R/WP-0					R-0			R/WP-0				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CH2_TIMEOUTENR	CH2_UNDERENR	CH2_OVERENR	CH2_CRCFAILENR	CH2_CCITENR	Reserved			CH1_TIMEOUTENR	CH1_UNDERENR	CH1_OVERENR	CH1_CRCFAILENR	CH1_CCITENR
R-0			R/WP-0					R-0			R/WP-0				

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Read returns 0. Writes have no effect.
28	CH4_TIMEOUTENR		**Channel 4 Timeout Interrupt Enable Bit**. Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt disable
27	CH4_UNDERENR		**Channel 4 Underrun Interrupt Enable Bit**. Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt disable

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
26	CH4_OVERENR		**Channel 4 Overrun Interrupt Enable Bit.** Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt disable
25	CH4_CRCFAILENR		**Channel 4 CRC Fail Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt disable
24	CH4_CCITENR		**Channel 4 Compression Complete Interrupt Enable Bit.** Writing a one to this bit disable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt disable
23–21	Reserved		Read returns 0. Writes have no effect.

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
20	CH3_TIMEOUTENR		**Channel 3 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt disable
19	CH3_UNDERENR		**Channel 3 Underrun Interrupt Enable Bit**. Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt disable
18	CH3_OVERENR		**Channel 3 Overrun Interrupt Enable Bit.** Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt disable

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
17	CH3_CRCFAILENR		**Channel 3 CRC Fail Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt disable
16	CH3_CCITENR		**Channel 3 Compression Complete Interrupt Enable Bit.** Writing a one to this bit disable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt disable
15–13	Reserved		Read returns 0. Writes have no effect.
12	CH2_TIMEOUTENR		**Channel 2 Timeout Interrupt Enable Bit**. Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt disable

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
11	CH2_UNDERENR		**Channel 2 Underrun Interrupt Enable Bit**. Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt disable
10	CH2_OVERENR		**Channel 2 Overrun Interrupt Enable Bit.** Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt disable
9	CH2_CRCFAILENR		**Channel 2 CRC Fail Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt disable

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
8	CH2_CCITENR		**Channel 2 Compression Complete Interrupt Enable Bit.** Writing a one to this bit disable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt disable
7–5	Reserved		Read returns 0. Writes have no effect.
4	CH1_TIMEOUTENR		**Channel 1 Timeout Interrupt Enable Bit**. Writing a one to this bit disable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Timeout Interrupt disable
		1	Timeout Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Timeout Interrupt disable
3	CH1_UNDERENR		**Channel 1 Underrun Interrupt Enable Bit**. Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Underrun Interrupt disable
		1	Underrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Underrun Interrupt disable

Table 22-8. CRC Interrupt Enable Clear Register(CRC_INTC) Field Descriptions (Continued)

Bit	Name	Value	Description
2	CH1_OVERENR		**Channel 1 Overrun Interrupt Enable Bit.** Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Overrun Interrupt disable
		1	Overrun Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Overrun Interrupt disable
1	CH1_CRCFAILENR		**Channel 1 CRC Fail Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	CRC Fail Interrupt disable
		1	CRC Fail Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	CRC Fail Interrupt disable
0	CH1_CCITENR		**Channel 1 Compression Complete Interrupt Enable Bit.** Writing a one to this bit disable the CRC compression complete interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
			User and Privileged mode read
		0	Compression Complete Interrupt disable
		1	Compression Complete Interrupt enable
			User and Privileged mode write
		0	Has no effect
		1	Compression Complete Interrupt disable

22.5.6 CRC Interrupt Status Register(CRC_STATUS)

Figure 22-15. CRC Interrupt Status Register(CRC_STATUS) [offset = 0x28]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CH4_TIMEOUT	CH4_UNDER	CH4_OVER	CH4_CRCFAIL	CH4_CCIT	Reserved			CH3_TIMEOUT	CH3_UNDER	CH3_OVER	CH3_CRCFAIL	CH3_CCIT
R-0			R/WP-0					R-0			R/WP-0				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CH2_TIMEOUT	CH2_UNDER	CH2_OVER	CH2_CRCFAIL	CH2_CCIT	Reserved			CH1_TIMEOUT	CH1_UNDER	CH1_OVER	CH1_CRCFAIL	CH1_CCIT
R-0			R/WP-0					R-0			R/WP-0				

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-9. CRC Interrupt Status Register(CRC_STATUS) Field Descriptions

Bit	Name	Value	Description
31–29	Reserved		Read returns 0. Writes have no effect.
28	CH4_TIMEOUT		**Channel 4 CRC Timeout Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in both AUTO and Semi-CPU mode
		0	No timeout interrupt is active
		1	Timeout Interrupt is active
27	CH4_UNDER		**Channel 4 CRC Underrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No Underrun Interrupt is active
		1	Underrun Interrupt is active
26	CH4_OVER		**Channel 4 CRC Overrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in either AUTO or Semi-CPU mode.
		0	No Overrun Interrupt is active
		1	Overrun Interrupt is active
25	CH4_CRCFAIL		**Channel 4 CRC Compare Fail Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No CRC Fail Interrupt is active
		1	CRC Fail Interrupt is active

Table 22-9. CRC Interrupt Status Register(CRC_STATUS) Field Descriptions (Continued)

Bit	Name	Value	Description
24	CH4_CCIT		**Channel 4 CRC Pattern Compression Complete Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is only set in Semi-CPU mode.
		0	Has no effect
		1	Compression Complete Interrupt disable
23–21	Reserved		Read returns 0. Writes have no effect.
20	CH3_TIMEOUT		**Channel 3 CRC Timeout Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in both AUTO and Semi-CPU mode
		0	No timeout interrupt is active
		1	Timeout Interrupt is active
19	CH3_UNDER		**Channel 3 CRC Underrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No Underrun Interrupt is active
		1	Underrun Interrupt is active
18	CH3_OVER		**Channel 3 CRC Overrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in either AUTO or Semi-CPU mode.
		0	No Overrun Interrupt is active
		1	Overrun Interrupt is active
17	CH3_CRCFAIL		**Channel 3 CRC Compare Fail Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No CRC Fail Interrupt is active
		1	CRC Fail Interrupt is active
16	CH3_CCIT		**Channel 3 CRC Pattern Compression Complete Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is only set in Semi-CPU mode.
		0	Has no effect
		1	Compression Complete Interrupt disable
15–13	Reserved		Read returns 0. Writes have no effect.

Table 22-9. CRC Interrupt Status Register(CRC_STATUS) Field Descriptions (Continued)

Bit	Name	Value	Description
12	CH2_TIMEOUT		**Channel 2 CRC Timeout Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in both AUTO and Semi-CPU mode
		0	No timeout interrupt is active
		1	Timeout Interrupt is active
11	CH2_UNDER		**Channel 2 CRC Underrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No Underrun Interrupt is active
		1	Underrun Interrupt is active
10	CH2_OVER		**Channel 2 CRC Overrun Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in either AUTO or Semi-CPU mode.
		0	No Overrun Interrupt is active
		1	Overrun Interrupt is active
9	CH2_CRCFAIL		**Channel 2 CRC Compare Fail Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is set in AUTO mode only.
		0	No CRC Fail Interrupt is active
		1	CRC Fail Interrupt is active
8	CH2_CCIT		**Channel 2 CRC Pattern Compression Complete Status Flag**. This bit is cleared by writing a ’1’ to it only. Writing ’0’ has no effect. This bit is only set in Semi-CPU mode.
		0	Has no effect
		1	Compression Complete Interrupt disable
7–5	Reserved		Read returns 0. Writes have no effect.
4	CH1_TIMEOUT		**Channel 1 Timeout Interrupt Enable Bit**. Writing a one to this bit enable the timeout interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Timeout Interrupt disable

Table 22-9. CRC Interrupt Status Register(CRC_STATUS) Field Descriptions (Continued)

Bit	Name	Value	Description
3	CH1_UNDER		**Channel 1 Underrun Interrupt Enable Bit**. Writing a one to this bit disable the underrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Underrun Interrupt disable
2	CH1_OVER		**Channel 1 Overrun Interrupt Enable Bit.** Writing a one to this bit disable the overrun interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Overrun Interrupt disable
1	CH1_CRCFAIL		**Channel 1 CRC Fail Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	CRC Fail Interrupt disable
0	CH1_CCIT		**Channel 1 Compression Complete Interrupt Enable Bit.** Writing a one to this bit disable the CRC fail interrupt. Writing a zero has no effect. Reading from this bit gives the status (interrupt enable/disable).
		0	Has no effect
		1	Compression Complete Interrupt disable

22.5.7 CRC Interrupt Offset(CRC_INT_OFFSET_REG)

Figure 22-16. CRC Interrupt Offset(CRC_INT_OFFSET_REG) [offset = 0x30]

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
Reserved								OFSTREG							
R-0								R-0							

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-10. CRC Interrupt Offset(CRC_INT_OFFSET_REG) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Read returns 0. Writes have no effect.
7-0	OFSTREG		**CRC Interrupt Offset.** This register indicates the highest priority pending interrupt vector address. Reading the offset register automatically clear the respective interrupt flag. Please reference Table 22-3 for details.

22.5.8 CRC Busy Register(CRC_BUSY)

Figure 22-17. CRC Busy Register(CRC_BUSY) [offset = 0x38]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							CH4_BUSY	Reserved							CH3_BUSY
R-0							R-0	R-0							R-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CH2_BUSY	Reserved							CH1_BUSY
R-0							R-0	R-0							R-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-11. CRC Busy Register(CRC_BUSY) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Read returns 0. Writes have no effect.
24	CH4_BUSY		**CH4_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
23–17	Reserved		Read returns 0. Writes have no effect.
16	CH3_BUSY		**CH3_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
15–9	Reserved		Read returns 0. Writes have no effect.
8	CH2_BUSY		**CH2_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
7–1	Reserved		Read returns 0. Writes have no effect.
0	CH1_BUSY		**CH1_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.

22.5.9 CRC Pattern Counter Preload Register1(CRC_PCOUNT_REG1)

Figure 22-18. CRC Pattern Counter Preload Register1(CRC_PCOUNT_REG1) [offset = 0x40]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												CRC_PAT_COUNT1[19:16]			
R-0												RW-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_PAT_COUNT1[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-12. CRC Pattern Counter Preload Register1(CRC_PCOUNT_REG1) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Read returns 0. Writes have no effect.
19-0	CRC_SEC_COUNT1		**Channel 1 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

22.5.10 CRC Sector Counter Preload Register1(CRC_SCOUNT_REG1)

Figure 22-19. CRC Sector Counter Preload Register1(CRC_SCOUNT_REG1) [offset = 0x44]

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
CRC_SEC_COUNT1[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-13. CRC Sector Counter Preload Register1(CRC_SCOUNT_REG1) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT1		**Channel 1 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

22.5.11 CRC Current Sector Register 1(CRC_CURSEC_REG1)

Figure 22-20. CRC Current Sector Register 1(CRC_CURSEC_REG1) [offset = 0x48]

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
CRC_CURSEC1[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-14. CRC Current Sector Register 1(CRC_CURSEC_REG1) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_CURSEC1		**Channel 1 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number.

22.5.12 CRC Channel 1 Watchdog Timeout Preload Register A(CRC_WDTOPLD1)

Figure 22-21. CRC Channel 1 Watchdog Timeout Preload Register A(CRC_WDTOPLD1) [offset = 0x4C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_WDTOPLD1[23:16]							
R-0								RW-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_WDTOPLD1[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-15. CRC Channel 1 Watchdog Timeout Preload Register A(CRC_WDTOPLD1) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_WDTOPLD1		**Channel 1 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns.

22.5.13 CRC Channel 1 Block Complete Timeout Preload Register B(CRC_BCTOPLD1)

Figure 22-22. CRC Channel 1 Block Complete Timeout Preload Register B(CRC_BCTOPLD1) [offset = 0x50]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_BCTOPLD1[23:16]							
								RW-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_BCTOPLD1[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-16. CRC Channel 1 Block Complete Timeout Preload Register B(CRC_BCTOPLD1) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_BCTCPLD1		**Channel 1 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

22.5.14 Channel 1 PSA signature low register(PSA_SIGREGL1)

Figure 22-23. Channel 1 PSA signature low register(PSA_SIGREGL1) [offset = 0x60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG1[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG1[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-17. Channel 1 PSA signature low register(PSA_SIGREGL1) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG1		**Channel 1 PSA Signature Low Register.** This register contains the value stored at PSASIG1[31:0] register.

22.5.15 Channel 1 PSA signature high register(PSA_SIGREGH1)

Figure 22-24. Channel 1 PSA signature high register(PSA_SIGREGH1) [offset = 0x64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG1[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG1[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-18. Channel 1 PSA signature high register(PSA_SIGREGH1) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG1		**Channel 1 PSA Signature Low Register.** This register contains the value stored at PSASIG1[63:32] register.

22.5.16 Channel 1 CRC value low register(CRC_REGL1)

Figure 22-25. Channel 1 CRC value low register(CRC_REGL1) [offset = 0x68]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">CRC1[31:16]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CRC1[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-19. Channel 1 CRC value low register(CRC_REGL1) Field Descriptions

Bit	Name	Value	Description
31-0	CRC1		**Channel 1 CRC Value Low Register.** This register contains the current known good signature value stored at CRC1[31:0] register.

22.5.17 Channel 1 CRC value high register(CRC_REGH1)

Figure 22-26. Channel 1 CRC value high register(CRC_REGH1) [offset = 0x6C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC1[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC1[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-20. Channel 1 CRC value high register(CRC_REGH1) Field Descriptions

Bit	Name	Value	Description
31-0	CRC1		**Channel 1 CRC Value Low Register.** This register contains the current known good signature value stored at CRC1[63:32] register.

22.5.18 Channel 1 PSA sector signature low register(PSA_SECSIGREGL1)

Figure 22-27. Channel 1 PSA sector signature low register(PSA_SECSIGREGL1) [offset = 0x70]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG1[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG1[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-21. Channel 1 PSA sector signature low register(PSA_SECSIGREGL1) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG1		**Channel 1 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG1[31:0] register.

22.5.19 Channel 1 PSA sector signature high register(PSA_SECSIGREGH1)

Figure 22-28. Channel 1 PSA sector signature high register(PSA_SECSIGREGH1) [offset = 0x74]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG1[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG1[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-22. Channel 1 PSA sector signature high register(PSA_SECSIGREGH1) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG1		**Channel 1 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG1[63:32] register.

22.5.20 Channel 1 Raw Data Low Register(RAW_DATAREGL1)

Figure 22-29. Channel 1 Raw Data Low Register(RAW_DATAREGL1) [offset = 0x78]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA1[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA1[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-23. Channel 1 Raw Data Low Register(RAW_DATAREGL1) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA1		**Channel 1 Raw Data Low Register.** This register contains bit 31:0 of the un-compressed raw data..

22.5.21 Channel 1 Raw Data High Register(RAW_DATAREGH1)

Figure 22-30. Channel 1 Raw Data High Register(RAW_DATAREGH1) [offset = 0x7C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA1[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA1[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-24. Channel 1 Raw Data High Register(RAW_DATAREGH1) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA1		**Channel 1 Raw Data Low Register.** This register contains bit 63:32 of the un-compressed raw data..

22.5.22 CRC Pattern Counter Preload Register2(CRC_PCOUNT_REG2)

Figure 22-31. CRC Pattern Counter Preload Register2(CRC_PCOUNT_REG2) [offset = 0x80]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												CRC_PAT_COUNT2[19:16]			
R-0												RW-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_PAT_COUNT2[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-25. CRC Pattern Counter Preload Register2(CRC_PCOUNT_REG2) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Read returns 0. Writes have no effect.
19-0	CRC_PAT_COUNT 2		**Channel 2 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

22.5.23 CRC Sector Counter Preload Register2(CRC_SCOUNT_REG2)

Figure 22-32. CRC Sector Counter Preload Register2(CRC_SCOUNT_REG2) [offset = 0x84]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CRC_SEC_COUNT2[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-26. CRC Sector Counter Preload Register2(CRC_SCOUNT_REG2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT2		**Channel 2 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

22.5.24 CRC Current Sector Register 2(CRC_CURSEC_REG2)

Figure 22-33. CRC Current Sector Register 2(CRC_CURSEC_REG2) [offset = 0x88]

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
CRC_CURSEC2[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-27. CRC Current Sector Register 2(CRC_CURSEC_REG2) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_CURSEC2		**Channel 2 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number.

22.5.25 CRC Channel 2 Watchdog Timeout Preload Register A(CRC_WDTOPLD2)

Figure 22-34. CRC Channel 2 Watchdog Timeout Preload Register A(CRC_WDTOPLD2) [offset = 0x8C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_WDTOPLD2[23:16]							
R-0								RW-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_WDTOPLD2[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-28. CRC Channel 2 Watchdog Timeout Preload Register A(CRC_WDTOPLD2) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_WDTOPLD2		**Channel 2 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns.

22.5.26 CRC Channel 2 Block Complete Timeout Preload Register B(CRC_BCTOPLD2)

Figure 22-35. CRC Channel 2 Block Complete Timeout Preload Register B(CRC_BCTOPLD2) [offset = 0x90]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_BCTOPLD2[23:16]							

RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_BCTOPLD2[15:0]															

RW-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-29. CRC Channel 2 Block Complete Timeout Preload Register B(CRC_BCTOPLD2) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_BCTCPLD2		**Channel 2 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

22.5.27 Channel 2 PSA signature low register(PSA_SIGREGL2)

Figure 22-36. Channel 2 PSA signature low register(PSA_SIGREGL2) [offset = 0xA0]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">PSASIG2[31:16]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">PSASIG2[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-30. Channel 2 PSA signature low register(PSA_SIGREGL2) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG2		**Channel 2 PSA Signature Low Register.** This register contains the value stored at PSASIG2[31:0] register.

22.5.28 Channel 2 PSA signature high register(PSA_SIGREGH2)

Figure 22-37. Channel 2 PSA signature high register(PSA_SIGREGH2) [offset = 0xA4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG2[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	A2	1	0
PSASIG2[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-31. Channel 2 PSA signature high register(PSA_SIGREGH2) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG2		**Channel 2 PSA Signature Low Register.** This register contains the value stored at PSASIG2[63:32] register.

22.5.29 Channel 2 CRC value low register(CRC_REGL2)

Figure 22-38. Channel 2 CRC value low register(CRC_REGL2) [offset = 0xA8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC2[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC2[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-32. Channel 2 CRC value low register(CRC_REGL2) Field Descriptions

Bit	Name	Value	Description
31-0	CRC2		**Channel 2 CRC Value Low Register.** This register contains the current known good signature value stored at CRC2[31:0] register.

22.5.30 Channel 2 CRC value high register(CRC_REGH2)

Figure 22-39. Channel 2 CRC value high register(CRC_REGH2) [offset = 0xAC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC2[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC2[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-33. Channel 2 CRC value high register(CRC_REGH2) Field Descriptions

Bit	Name	Value	Description
31-0	CRC2		**Channel 2 CRC Value Low Register.** This register contains the current known good signature value stored at CRC2[63:32] register.

22.5.31 Channel 2 PSA sector signature low register(PSA_SECSIGREGL2)

Figure 22-40. Channel 2 PSA sector signature low register(PSA_SECSIGREGL2) [offset = 0xB0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG2[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG2[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-34. Channel 2 PSA sector signature low register(PSA_SECSIGREGL2) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG2		**Channel 2 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG2[31:0] register.

22.5.32 Channel 2 PSA sector signature high register(PSA_SECSIGREGH2)

Figure 22-41. Channel 2 PSA sector signature high register(PSA_SECSIGREGH2) [offset = 0xB4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG2[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG2[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-35. Channel 2 PSA sector signature high register(PSA_SECSIGREGH2) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG2		**Channel 2 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG2[63:32] register.

22.5.33 Channel 2 Raw Data Low Register(RAW_DATAREGL2)

Figure 22-42. Channel 2 Raw Data Low Register(RAW_DATAREGL2) [offset = 0xB8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA2[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA2[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-36. Channel 2 Raw Data Low Register(RAW_DATAREGL2) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA2		**Channel 2 Raw Data Low Register.** This register contains bit 31:0 of the un-compressed raw data..

22.5.34 Channel 2 Raw Data High Register(RAW_DATAREGH2)

Figure 22-43. Channel 2 Raw Data High Register(RAW_DATAREGH2) [offset = 0xBC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA2[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA2[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-37. Channel 2 Raw Data High Register(RAW_DATAREGH2) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA2		**Channel 2 Raw Data Low Register.** This register contains bit 63:32 of the un-compressed raw data..

22.5.35 CRC Pattern Counter Preload Register3(CRC_PCOUNT_REG3)

Figure 22-44. CRC Pattern Counter Preload Register3(CRC_PCOUNT_REG3) [offset = 0xC0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												CRC_PAT_COUNT3[19:16]			
R-0												RW-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_PAT_COUNT3[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-38. CRC Pattern Counter Preload Register3(CRC_PCOUNT_REG3) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Read returns 0. Writes have no effect.
19-0	CRC_SEC_COUNT3		**Channel 3 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

22.5.36 CRC Sector Counter Preload Register3(CRC_SCOUNT_REG3)

Figure 22-45. CRC Sector Counter Preload Register3(CRC_SCOUNT_REG3) [offset = 0xC4]

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
CRC_SEC_COUNT3[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-39. CRC Sector Counter Preload Register3(CRC_SCOUNT_REG3) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT3		**Channel 3 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

22.5.37 CRC Current Sector Register 3(CRC_CURSEC_REG3)

Figure 22-46. CRC Current Sector Register 3(CRC_CURSEC_REG3) [offset = 0xC8]

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
CRC_CURSEC3[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-40. CRC Current Sector Register 3(CRC_CURSEC_REG3) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_CURSEC3		**Channel 3 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number.

22.5.38 CRC Channel 3 Watchdog Timeout Preload Register A(CRC_WDTOPLD3)

Figure 22-47. CRC Channel 3 Watchdog Timeout Preload Register A(CRC_WDTOPLD3) [offset = 0xCC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_WDTOPLD3[23:16]							
R-0								RW-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_WDTOPLD3[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-41. CRC Channel 3 Watchdog Timeout Preload Register A(CRC_WDTOPLD3) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_WDTOPLD3		**Channel 3 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns.

22.5.39 CRC Channel 3 Block Complete Timeout Preload Register B(CRC_BCTOPLD3)

Figure 22-48. CRC Channel 3 Block Complete Timeout Preload Register B(CRC_BCTOPLD3) [offset = 0xD0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_BCTOPLD3[23:16]							

RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_BCTOPLD3[15:0]															

RW-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-42. CRC Channel 3 Block Complete Timeout Preload Register B(CRC_BCTOPLD3) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_BCTCPLD3		**Channel 3 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

22.5.40 Channel 3 PSA signature low register(PSA_SIGREGL3)

Figure 22-49. Channel 3 PSA signature low register(PSA_SIGREGL3) [offset = 0xE0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG3[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG3[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-43. Channel 3 PSA signature low register(PSA_SIGREGL3) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG3		**Channel 3 PSA Signature Low Register.** This register contains the value stored at PSASIG3[31:0] register.

22.5.41 Channel 3 PSA signature high register(PSA_SIGREGH3)

Figure 22-50. Channel 3 PSA signature high register(PSA_SIGREGH3) [offset = 0xE4]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">PSASIG3[63:48]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">PSASIG3[47:32]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-44. Channel 3 PSA signature high register(PSA_SIGREGH3) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG3		**Channel 3 PSA Signature Low Register.** This register contains the value stored at PSASIG3[63:32] register.

22.5.42 Channel 3 CRC value low register(CRC_REGL3)

Figure 22-51. Channel 3 CRC value low register(CRC_REGL3) [offset = 0xE8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC3[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC3[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-45. Channel 3 CRC value low register(CRC_REGL3) Field Descriptions

Bit	Name	Value	Description
31-0	CRC3		**Channel 3 CRC Value Low Register.** This register contains the current known good signature value stored at CRC3[31:0] register.

22.5.43 Channel 3 CRC value high register(CRC_REGH3)

Figure 22-52. Channel 3 CRC value high register(CRC_REGH3) [offset = 0xEC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC3[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC3[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-46. Channel 3 CRC value high register(CRC_REGH3) Field Descriptions

Bit	Name	Value	Description
31-0	CRC3		**Channel 3 CRC Value Low Register.** This register contains the current known good signature value stored at CRC3[63:32] register.

22.5.44 Channel 3 PSA sector signature low register(PSA_SECSIGREGL3)

Figure 22-53. Channel 3 PSA sector signature low register(PSA_SECSIGREGL3) [offset = 0xF0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG3[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG3[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-47. Channel 3 PSA sector signature low register(PSA_SECSIGREGL3) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG3		**Channel 3 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG3[31:0] register.

22.5.45 Channel 3 PSA sector signature high register(PSA_SECSIGREGH3)

Figure 22-54. Channel 3 PSA sector signature high register(PSA_SECSIGREGH3) [offset = 0xF4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG3[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG3[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-48. Channel 3 PSA sector signature high register(PSA_SECSIGREGH3) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG3		**Channel 3 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG3[63:32] register.

22.5.46 Channel 3 Raw Data Low Register(RAW_DATAREGL3)

Figure 22-55. Channel 3 Raw Data Low Register(RAW_DATAREGL3) [offset = 0xF8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA3[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA3[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-49. Channel 3 Raw Data Low Register(RAW_DATAREGL3) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA3		**Channel 3 Raw Data Low Register.** This register contains bit 31:0 of the un-compressed raw data..

22.5.47 Channel 3 Raw Data High Register(RAW_DATAREGH3)

Figure 22-56. Channel 3 Raw Data High Register(RAW_DATAREGH3) [offset = 0xFC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA3[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA3[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-50. Channel 3 Raw Data High Register(RAW_DATAREGH3) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA3		**Channel 3 Raw Data Low Register.** This register contains bit 63:32 of the un-compressed raw data..

22.5.48 CRC Pattern Counter Preload Register4(CRC_PCOUNT_REG4)

Figure 22-57. CRC Pattern Counter Preload Register4(CRC_PCOUNT_REG4) [offset = 0x100]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												CRC_PAT_COUNT4[19:16]			
R-0												RW-0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_PAT_COUNT4[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-51. CRC Pattern Counter Preload Register4(CRC_PCOUNT_REG4) Field Descriptions

Bit	Name	Value	Description
31–20	Reserved		Read returns 0. Writes have no effect.
19-0	CRC_SEC_COUNT4		**Channel 4 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

22.5.49 CRC Sector Counter Preload Register4(CRC_SCOUNT_REG4)

Figure 22-58. CRC Sector Counter Preload Register4(CRC_SCOUNT_REG4) [offset = 0x104]

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
CRC_SEC_COUNT4[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-52. CRC Sector Counter Preload Register4(CRC_SCOUNT_REG4) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_SEC_COUNT4		**Channel 4 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

22.5.50 CRC Current Sector Register 4(CRC_CURSEC_REG4)

Figure 22-59. CRC Current Sector Register 4(CRC_CURSEC_REG4) [offset = 0x108]

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
CRC_CURSEC4[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-53. CRC Current Sector Register 4(CRC_CURSEC_REG4) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Read returns 0. Writes have no effect.
15-0	CRC_CURSEC4		**Channel 4 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number.

22.5.51 CRC Channel 4 Watchdog Timeout Preload Register A(CRC_WDTOPLD4)

Figure 22-60. CRC Channel 4 Watchdog Timeout Preload Register A(CRC_WDTOPLD4) [offset = 0x10C]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">CRC_WDTOPLD4[23:16]</td></tr>
<tr><td colspan="8">R-0</td><td colspan="8">RW-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CRC_WDTOPLD4[15:0]</td></tr>
<tr><td colspan="16">RW-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-54. CRC Channel 4 Watchdog Timeout Preload Register A(CRC_WDTOPLD4) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_WDTOPLD4		**Channel 4 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns.

22.5.52 CRC Channel 4 Block Complete Timeout Preload Register B(CRC_BCTOPLD4)

Figure 22-61. CRC Channel 4 Block Complete Timeout Preload Register B(CRC_BCTOPLD4) [offset = 0x110]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CRC_BCTOPLD4[23:16]							

RW-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_BCTOPLD4[15:0]															

RW-0

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-55. CRC Channel 4 Block Complete Timeout Preload Register B(CRC_BCTOPLD4) Field Descriptions

Bit	Name	Value	Description
31–24	Reserved		Read returns 0. Writes have no effect.
23-0	CRC_BCTCPLD4		**Channel 4 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

22.5.53 Channel 4 PSA signature low register(PSA_SIGREGL4)

Figure 22-62. Channel 4 PSA signature low register(PSA_SIGREGL4) [offset = 0x120]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG4[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG4[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

.

Table 22-56. Channel 4 PSA signature low register(PSA_SIGREGL4) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG4		**Channel 4 PSA Signature Low Register.** This register contains the value stored at PSASIG4[31:0] register.

22.5.54 Channel 4 PSA signature high register(PSA_SIGREGH4)

Figure 22-63. Channel 4 PSA signature high register(PSA_SIGREGH4) [offset = 0x124]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASIG4[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASIG4[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-57. Channel 4 PSA signature high register(PSA_SIGREGH4) Field Descriptions

Bit	Name	Value	Description
31-0	PSASIG4		**Channel 4 PSA Signature Low Register.** This register contains the value stored at PSASIG4[63:32] register.

22.5.55 Channel 4 CRC value low register(CRC_REGL4)

Figure 22-64. Channel 4 CRC value low register(CRC_REGL4) [offset = 0x128]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC4[31:16]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC4[15:0]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-58. Channel 4 CRC value low register(CRC_REGL4) Field Descriptions

Bit	Name	Value	Description
31-0	CRC4		**Channel 4 CRC Value Low Register.** This register contains the current known good signature value stored at CRC4[31:0] register.

22.5.56 Channel 4 CRC value high register(CRC_REGH4)

Figure 22-65. Channel 4 CRC value high register(CRC_REGH4) [offset = 0x12C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC4[63:48]															
RW-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC4[47:32]															
RW-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-59. Channel 4 CRC value high register(CRC_REGH4) Field Descriptions

Bit	Name	Value	Description
31-0	CRC4		**Channel 4 CRC Value Low Register.** This register contains the current known good signature value stored at CRC4[63:32] register.

22.5.57 Channel 4 PSA sector signature low register(PSA_SECSIGREGL4)

Figure 22-66. Channel 4 PSA sector signature low register(PSA_SECSIGREGL4) [offset = 0x130]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">PSASECSIG4[31:16]</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">PSASECSIG4[15:0]</td></tr>
<tr><td colspan="16">R-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-60. Channel 4 PSA sector signature low register(PSA_SECSIGREGL4) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG4		**Channel 4 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG4[31:0] register.

22.5.58 Channel 4 PSA sector signature high register(PSA_SECSIGREGH4)

Figure 22-67. Channel 4 PSA sector signature high register(PSA_SECSIGREGH4) [offset = 0x134]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSASECSIG4[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSASECSIG4[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-61. Channel 4 PSA sector signature high register(PSA_SECSIGREGH4) Field Descriptions

Bit	Name	Value	Description
31-0	PSASECSIG4		**Channel 4 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG4[63:32] register.

22.5.59 Channel 4 Raw Data Low Register(RAW_DATAREGL4)

Figure 22-68. Channel 4 Raw Data Low Register(RAW_DATAREGL4) [offset = 0x138]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA4[31:16]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA4[15:0]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-62. Channel 4 Raw Data Low Register(RAW_DATAREGL4) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA4		**Channel 4 Raw Data Low Register.** This register contains bit 31:0 of the un-compressed raw data..

22.5.60 Channel 4 Raw Data High Register(RAW_DATAREGH4)

Figure 22-10 and Table 22-63 describe this register.

Figure 22-69. Channel 4 Raw Data High Register(RAW_DATAREGH4) [offset = 0x13C]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAW_DATA4[63:48]															
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAW_DATA4[47:32]															
R-0															

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-63. Channel 4 Raw Data High Register(RAW_DATAREGH4) Field Descriptions

Bit	Name	Value	Description
31-0	RAW_DATA4		**Channel 4 Raw Data Low Register.** This register contains bit 63:32 of the un-compressed raw data..

22.5.61 Data Bus Selection Register(MCRC_TRACE_BUS_SEL)

Figure 22-70. Data Bus Selection Register(MCRC_TRACE_BUS_SEL) [offset = 0x140]

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
Reserved													MEn	DTC-MEn	ITCMEn
R-0													RW-1	RW-1	RW-1

R = Read in all modes; WP = Write in privilege mode only; -n = Value after reset

Table 22-64. Channel 4 Raw Data High Register(RAW_DATAREGH4) Field Descriptions

Bit	Name	Value	Description
31–3	Reserved		Read returns 0. Writes have no effect.
2	MEn		Enable/disables the tracing of Peripheral Bus Master
		0	Tracing of Peripheral Bus Master has been disabled
		1	Tracing of Peripheral Bus Master master bus has been enabled
1	DTCMEn		Enable/disables 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		Enable/disables the tracing of instruction TCM
		0	Tracing of Flash data bus has been disabled
		1	Tracing of Flash data bus has been enabled

CPU Compare Module for Cortex™-R4F (CCM-R4F)

This device implements two instances of the Cortex-R4F CPU which are running in lockstep to detect potential faults which can be introduced by the environment the device operates in. The CCM-R4F module detects those faults and signals them to the system.

Topic **Page**

23.1 Main Features

To increase system reliability, safety-critical applications require run-time detection of faults in the Central Processing Unit (CPU). The CPU Compare Module for Cortex-R4F (CCM-R4F) module compares the output of both Cortex-R4F CPUs running in lock step. Any difference in the outputs of the two CPUs is flagged as an error. For diagnostic purposes the CCM-R4F module also incorporates a self test capability to allow for run-time checking of hardware faults within the CCM-R4F module itself.

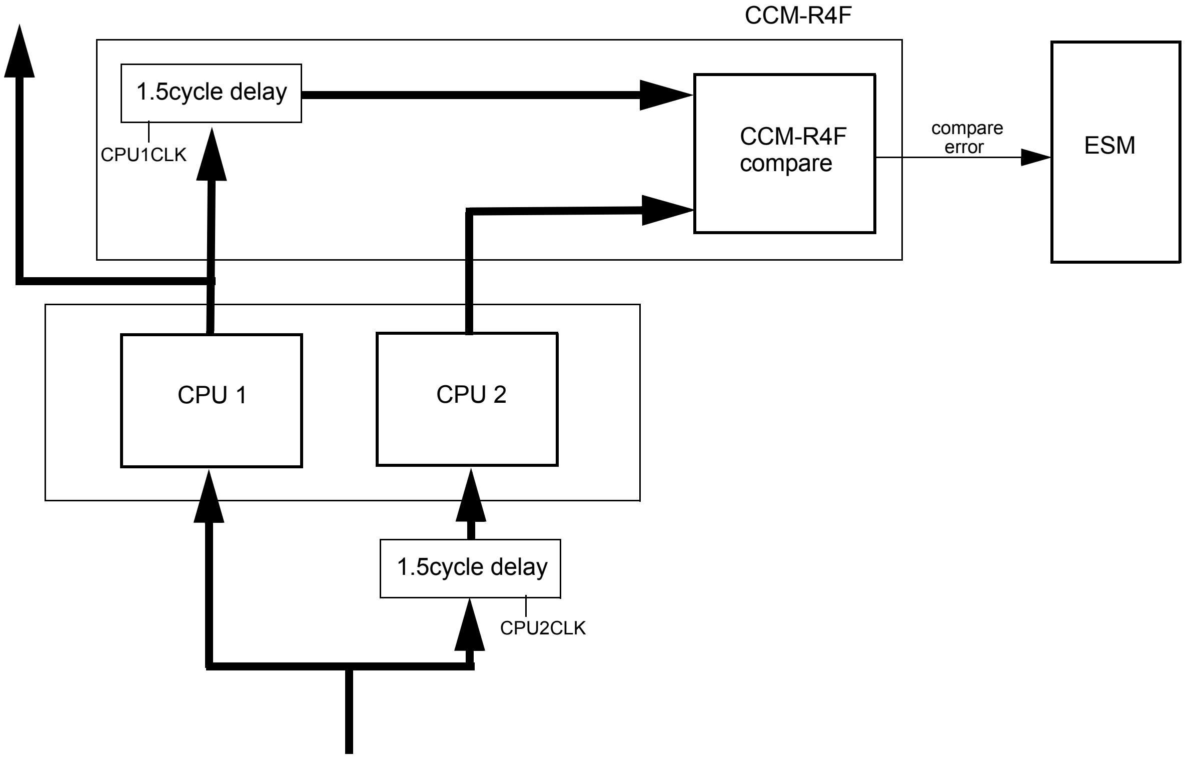
The main features of the CCM-R4F module are:

- run-time detection of faults
- self test capability
- error forcing capability

23.2 Block diagram

Figure 23-1 shows the interconnection diagram of the CCM-R4F module with the two Cortex-R4F CPUs. The output signals of both CPUs are compared in the CCM-R4F unit. To avoid common mode impacts, the signals of the CPUs to be compared are delayed in different ways. The output signals of CPU1 are delayed 1.5 cycles while the input signals of CPU2 are delayed 1.5 cycles.

Figure 23-1. Block Diagram



23.3 Module Operation

The CCM-R4F compares the outputs of the two Cortex-R4F CPUs on the TMS570 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 all the CPU output signals have a known value.

The CCM-R4F module can run in one out of four operating modes:

1. lock step,
2. self test,
3. error forcing, and
4. self test error forcing

The operating mode can be selected by writing a dedicated key to the key register (MKEY).

23.3.1 Lock Step Mode

In the lock step mode, the output signals of both CPUs are compared one-by-one. A difference in the CPU outputs is indicated by ESM error flag “CCM-R4F - compare”.

To avoid an erroneous CCM-R4F 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.

23.3.2 Self Test Mode

In self test mode the CCM-R4F module itself is checked for faults. During self test, the compare error module output signal is deactivated. Any fault detected inside the CCM-R4F module will be flagged by ESM error “CCM-R4F - selftest”.

In self test mode, the CCM-R4F module automatically generates test patterns to look for any hardware faults inside itself. 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 module, 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-R4F remains in self test mode after the test has been terminated or completed, and the application needs to switch the CCM-R4F module mode by writing another key to the mode key register (MKEY). During the self test operation, the compare error signal output to the ESM module is inactive irrespective of the compare result.

There are two types of patterns generated by CCM-R4F during self test mode:

(i) Compare Match Test, and

(ii) Compare Mismatch Test

CCM-R4F 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-R4F’s compare block and clocked for one cycle. The duration of self test is 3615 cycles.

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 CPU register file. The self test is not interruptible.

23.3.2.1 Compare Match Test

During the Compare Match Test, there are four different test patterns generated to stimulate the CCM-R4F module. 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 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 1's on both CPU signal ports
- All 0's on both CPU signal ports
- 0xA's on both CPU signal ports
- 0x5's on both CPU signal ports

These four test patterns will take four clock cycles to complete. Table 23-1 illustrates the sequence of Compare Match Test.

Table 23-1. 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	
1's	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	1	1	0
0's	0	0	0	0	0	0	0	0	0's	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

23.3.2.2 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 1's vector is applied to the CCM-R4F's CPU1 input port and the same pattern is also applied to the CCM-R4F's CPU2 input port but with one bit flipped starting from signal position 0. The un-equal vector will cause the CCM-R4F module to expect a compare mismatch at signal position 0, if the CCM-R4F logic is working correctly. If, however, the CCM-R4F 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-R4F 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 23-2 illustrates the sequence of Compare Mismatch Test.

Table 23-2. 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	1's	1	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	1	0	0
1	1	1's	1	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	0	1	1
1	1	1's	1	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	0	1	1	2
1	1	1's	1	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	0	1	1	1	3
:																						
1	1	1's	1	1	1	1	1	1	1	1	1	0	1's	1	1	1	1	1	1	1	1	n-1
1	1	1's	1	1	1	1	1	1	1	1	0	1	1's	1	1	1	1	1	1	1	1	n
1	1	1's	1	1	1	1	1	1	1	0	1	1	1's	1	1	1	1	1	1	1	1	n+1
1	1	1's	1	1	1	1	1	1	0	1	1	1	1's	1	1	1	1	1	1	1	1	n+2
1	1	1's	1	1	1	1	1	0	1	1	1	1	1's	1	1	1	1	1	1	1	1	n+3
1	1	1's	1	1	1	1	0	1	1	1	1	1	1's	1	1	1	1	1	1	1	1	n+4
:																						

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	0	1’s	1	1	1	1	1	1	1	1	1	1	1’s	1	1	1	1	1	1	1	1	2n-1
0	1	1’s	1	1	1	1	1	1	1	1	1	1	1’s	1	1	1	1	1	1	1	1	2n

23.3.3 Error Forcing Mode

In the error forcing mode, a test pattern is applied to the CPU related inputs of the CCM-R4F compare logic to force an error in the compare error output signal of the compare unit. The ESM error flag “CCM-R4F - compare” is expected after the error forcing mode completes.

Error forcing mode is similar to the Compare Mismatch Test operation of self test mode in which an un-equal vector is applied to the CCM-R4F CPU signal ports. Instead of setting a self test error flag and asserting the self test error signal to ESM module, the error forcing mode forces the compare mismatch to actually assert the compare error output signal. This ensures that any fault in the path between CCM-R4F module and ESM module is detected.

Only one hardcoded test pattern is applied into CCM-R4F during error forcing mode. A repeated 0x5 pattern is applied to CPU1 signal port while a repeated 0xA pattern is applied to the CPU2 signal port. The error forcing mode takes one cycle to complete. Hence, the failing signature is presented for one clock cycle and the mode is automatically switched to lock step mode and the key register (MKEY) shows the lock step key mode. During the one cycle required by the error forcing test, the CPU output signals are not compared. The ESM module is expecting to receive the error signal from the CCM-R4F module once the error forcing mode is entered. If no error signal is detected by ESM module, then a hardware fault is present.

23.3.4 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 lock step mode and the key is switched to lock step after one clock cycle. The ESM error flag “CCM-R4F - selftest” is expected after the self test error forcing mode completes.

23.3.5 Operation During CPU Debug Mode

Certain JTAG operations place the CPU in a debug mode where the code execution is halted. Once this mode is entered, compare errors are not generated, irrespective of the compare mode the CCM-R4F is in. The status flags are also not updated. Normal operation is resumed only after a CPU reset is asserted.

23.4 Control Registers

This section describes the CCM-R4F registers. Each register begins on a word boundary. The registers support 32-bit, 16-bit and 8-bit accesses.

Figure 23-2. Module Registers

<table>
<tr><th>Address
Register Name
Page #</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">0xFFFFF600
CCMSR
Page 1757</td><td colspan="15">Reserved</td><td>CPME</td></tr>
<tr><td colspan="7">Reserved</td><td>STC</td><td colspan="6">Reserved</td><td>STET</td><td>STE</td></tr>
<tr><td rowspan="2">0xFFFFF604
CCMKEYR
Page 1759</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">MKEY[3:0]</td></tr>
</table>

23.4.1 CCM-R4F Status Register (CCMSR – 0xFFFFF600)

Figure 23-3. *CCM-R4F Status Register (CCMSR)*

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							STC	Reserved						STET	STE
R-0							R-0	R-0						R-0	R-0

R = Read, W = Write, P = Privileged Mode, U = Undefined; -*n* = Value after reset

.

Table 1-3. CCM-R4F Status Register (CCMSR) Field Descriptions

Bit	Name	Value	Description
31–17	Reserved	0	Reads return zeros and writes have no effect.
16	CMPE		Compare Error
		0	**User and privileged mode:** Read: CPU signals are identical. **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: CPU signal compare mismatch. **Privileged mode:** Write: Clears the bit.
15-9	Reserved	0	Reads return zeros and 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.
		0	**User and privileged mode:** Read: Self test on-going if self test mode is entered. Write: Writes have no effect.
		1	**User and privileged mode:** Read: Self test is complete. Write: Writes have no effect.
7-2	Reserved	0	Reads return zeros and writes have no effect.

Table 1-3. CCM-R4F Status Register (CCMSR) Field Descriptions (Continued)

Bit	Name	Value	Description
1	STET		Self Test Error Type
		0	**User and privileged mode:** Read: Self test failed during Compare Match Test if STE=1. Write: Writes have no effect.
		1	**User and privileged mode:** 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.
		0	**User and privileged mode:** Read: Self test passed. Write: Writes have no effect.
		1	**User and privileged mode:** Read: Self test failed. Write: Writes have no effect.

23.4.2 *CCM-R4F Key Register (CCMKEYR – 0xFFFFF604)*

Figure 23-4. *CCM-R4F Key Register (CCMKEYR)*

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
Reserved												MKEY[3:0]			
R-0												RWP-0			

R = Read, W = Write, P = Privileged Mode, U = Undefined; -*n* = Value after reset

Table 23-4. CCM-R4F Key Register (CCMKEYR) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved	0	Reads return zeros and writes have no effect.
3-0	MKEY		Mode Key
		0000	**User and privileged mode:** Read: Returns current value of the MKEY. **Privileged mode:** Write: Lock Step mode.
		0110	**User and privileged mode:** Read: Returns current value of the MKEY. **Privileged mode:** Write: Self Test mode.
		1001	**User and privileged mode:** Read: Returns current value of the MKEY. **Privileged mode:** Write: Error Forcing mode.
		1111	**User and privileged mode:** Read: Returns current value of the MKEY. **Privileged mode:** Write: Self Test Error Forcing mode. **Note:** It is recommended writing no other keys than the keys listed. Invalid keys will result in switching operation to lock step mode.

TEXAS INSTRUMENTS

Chapter 24

SPNU489C - February 2012

Vectored Interrupt Manager (VIM) Module

This section describes the behavior of the vectored interrupt manager (VIM) module of the TMS570Px family.

Topic **Page**

24.1 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).

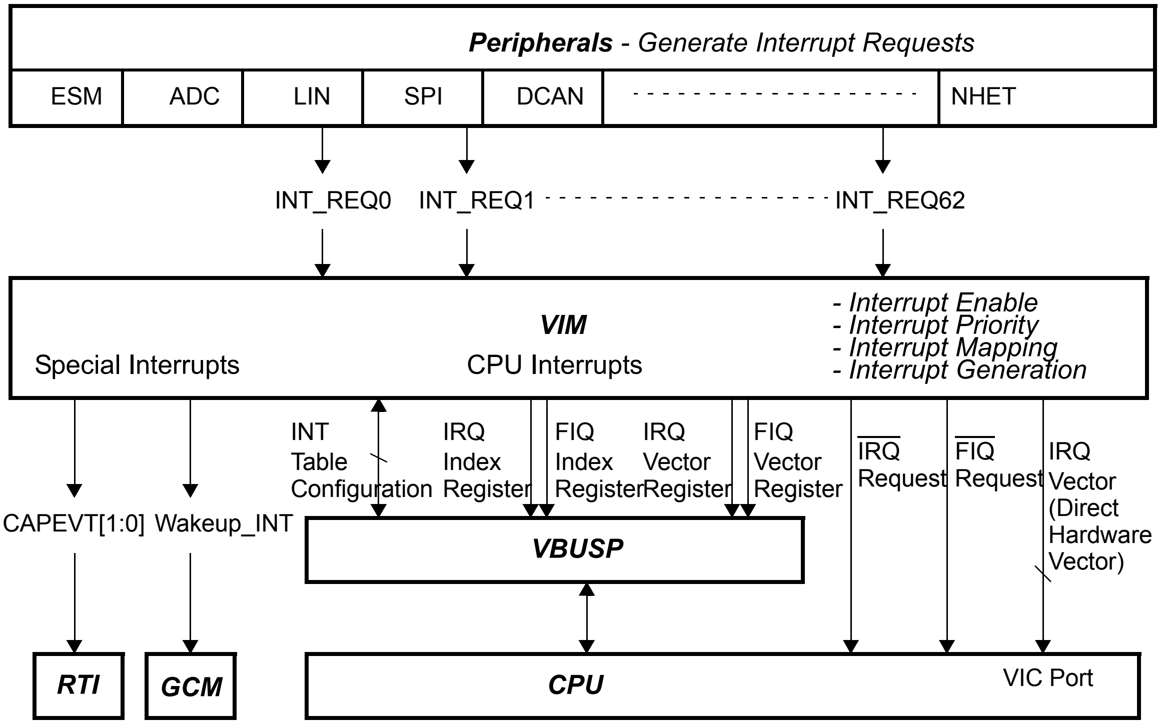
The VIM module has the following features:

- Supports 63 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.

24.2 Device Level Interrupt Management

A block diagram of device level interrupt handling is shown in Figure 24-1. 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 dispatch instructions in the ISR. Section 24.2.1 through Section 24.2.3 provide additional details about these three steps.

Figure 24-1. Device Level Interrupt Block Diagram



24.2.1 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 TMS570Px 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.

24.2.2 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.

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 2 shows how to enable/disable 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 because the FIQ implemented in Cortex-R4F is Non-Maskable Fast Interrupts (NMFI).

24.2.3 Software Interrupt Handling Options

TMS570x supports three different possibilities for software to handle interrupts

1. 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 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 1 shows how to enable the hardware vectored interrupt.

   Note:
   This mode is NOT available for FIQ.

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 3 illustrates the configuration for the exception vectors using this mode.

3. 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, software thus imported will not benefit from the whole range of performance the VIM is capable of.

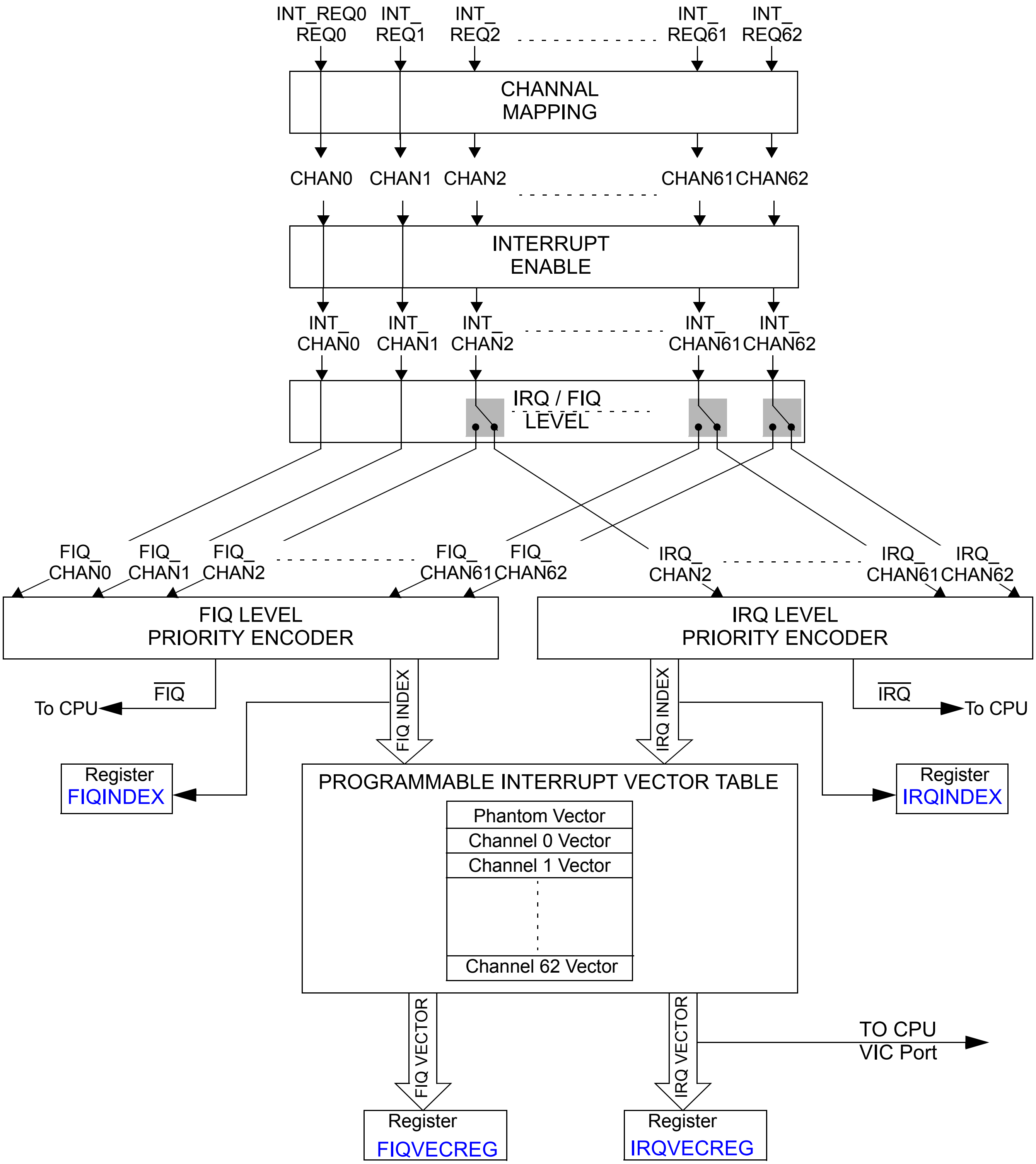
   To port legacy software, the interrupt vector at 0x18 (IRQ) or 0x1C (FIQ) only needs to be a branch statement to a software interrupt table. The software interrupt table reads the pending interrupt from a vector offset register (FIQINDEX[7:0] for FIQ interrupts and IRQINDEX[7:0] for IRQ interrupts). All pending interrupts can be viewed in the INTREQ register. Example 4 shows how to respond to FIQ with short latency in this mode.

   Example 4 shows how to respond to FIQ with short latency in this mode.

24.3 Interrupt Handling Inside VIM

A block diagram of the interrupt handling inside VIM is shown in Figure 24-2

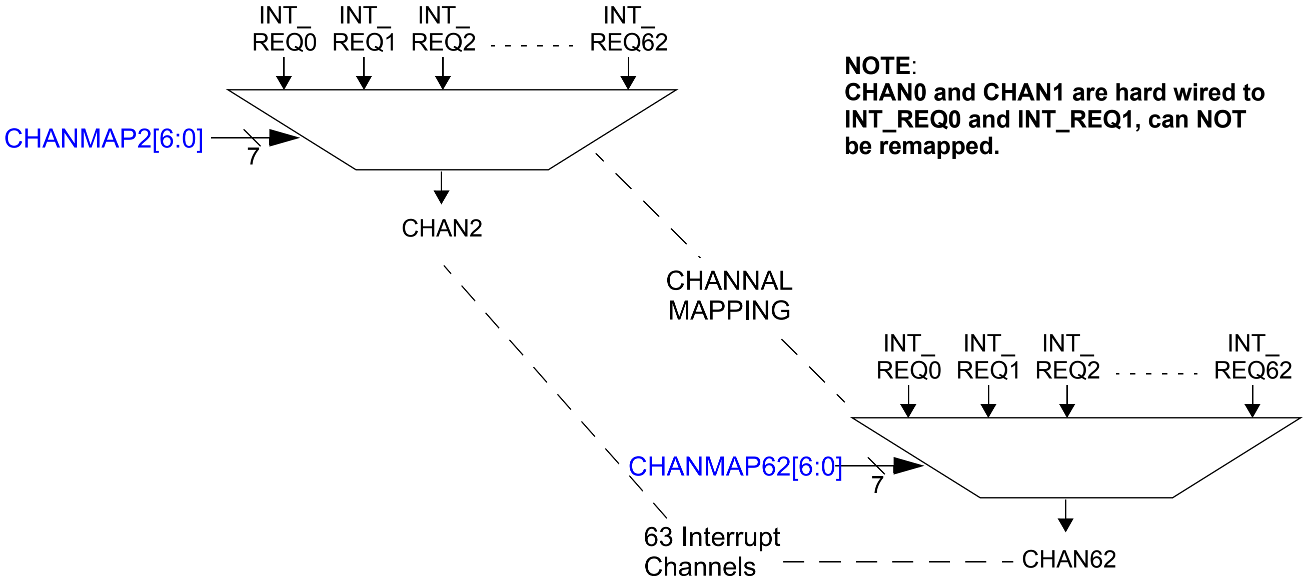
Figure 24-2. VIM Interrupt Handling Block Diagram



24.3.1 VIM Interrupt Channel Mapping

The VIM support 64 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 24-3. For each interrupt channel, CHANx, there is a corresponding mapping register bit field CHANMAPx[6:0], which determines which VIM request the interrupt channel maps to. With this scheme, the same request can be mapped to multiple channels. Since 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 24-3. VIM Channel Mapping



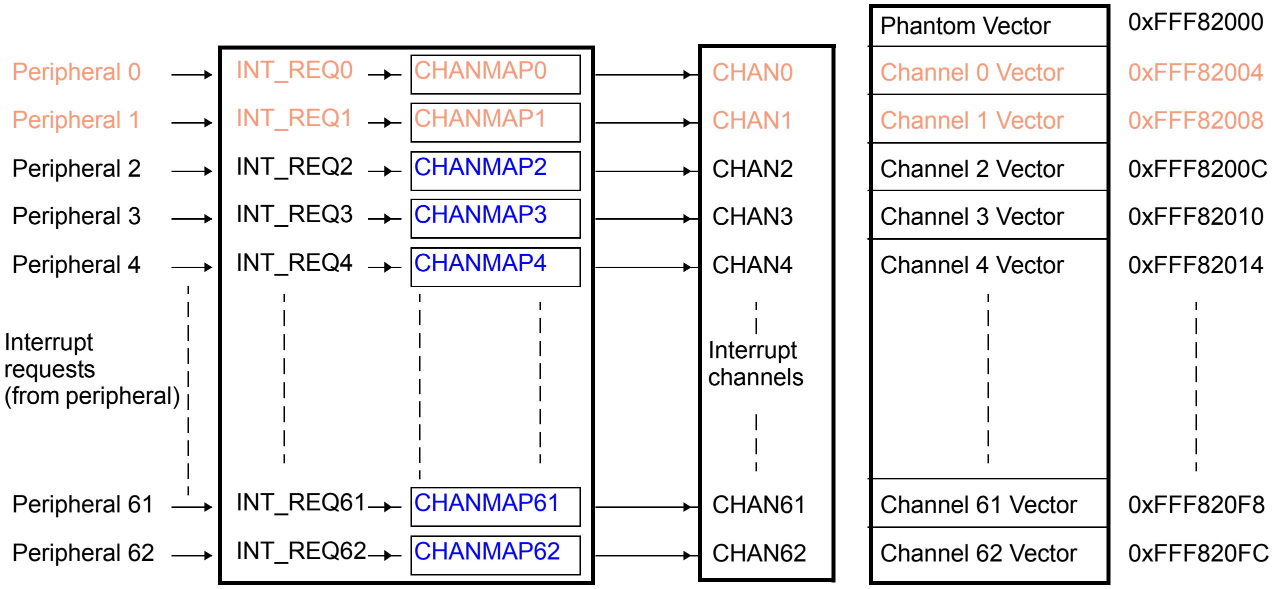
NOTE:
CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, can NOT be remapped.

Note: CHAN63

CHAN63 has no dedicated interrupt vector table entry. Therefore, CHAN63 shall NOT be remapped to other INT_REQ (INT_REQ63 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 24-4 shows the default state following the reset.

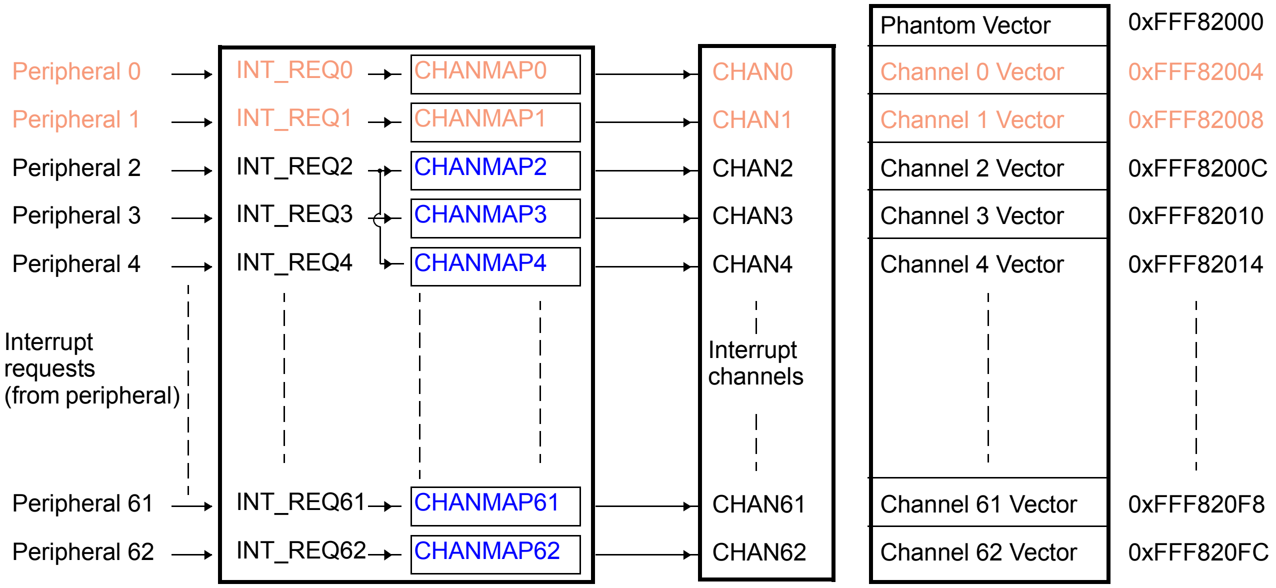
Figure 24-4. VIM in Default State



Note:
CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, can NOT be remapped.

Figure 24-5 shows the VIM INT2 is remapped to both Channel 2 and 4, and INT3 is mapped to channel 3.

Figure 24-5. VIM in a Programmed State



Note:
CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, can NOT be remapped.

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:

a.INT0

b.INT1

c.INT2

d.INT3

Disabling channel 2, the priority becomes:

a.INT0

b.INT1

c.INT3

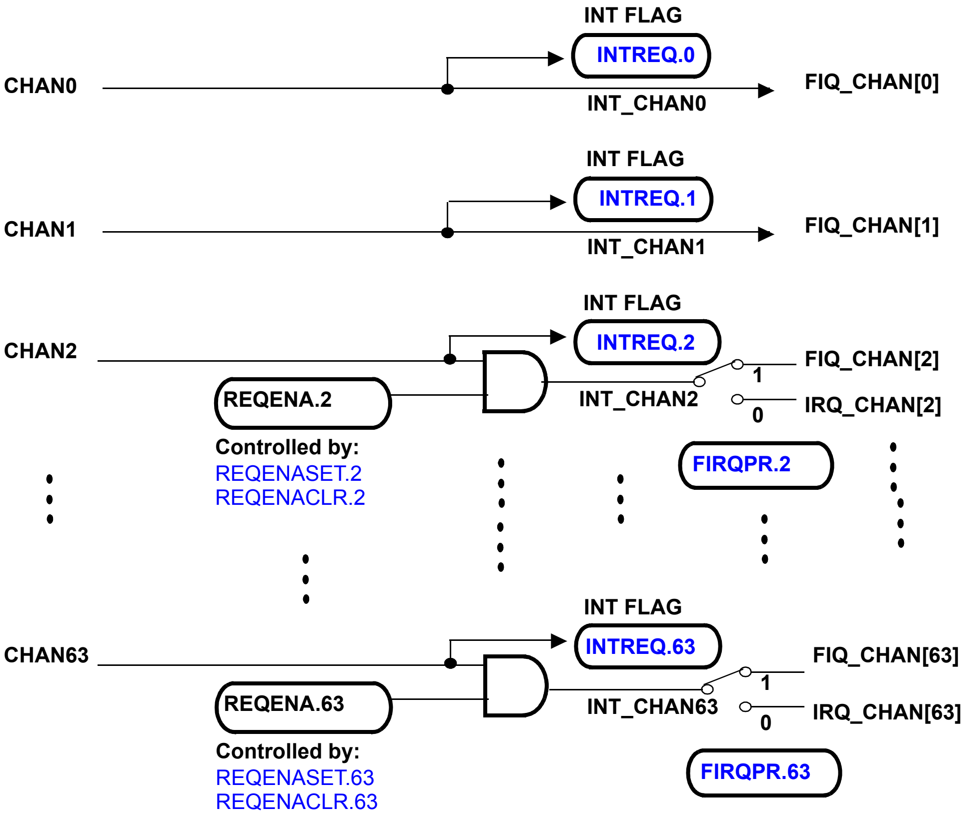
d.INT2

24.3.2 VIM Input Channel Management

As shown in Figure 24-6, 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: Note:
The interrupt ENABLE register does not affect the value of INTREQ.

Figure 24-6. 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 channel 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.

24.4 Interrupt Vector Table (VIM RAM)

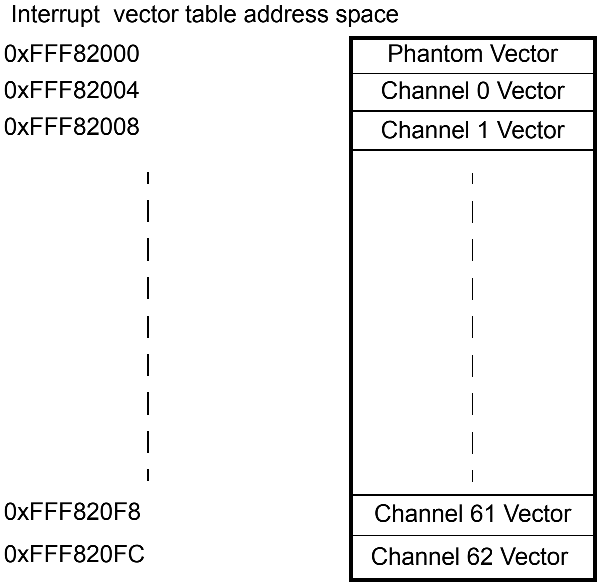
Interrupt vector table stores the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR.

For safety reasons, the interrupt vector table has protection by parity to prevent corruption by soft error. The parity scheme is implemented as a continuous background check based on memory access. Section 24.4.1 through Section 24.4.4 describe how parity works in the interrupt vector table.

24.4.1 Interrupt Vector Table Operation

The interrupt vector table is organized in 64 words of 32 bits. 32-bit, 16-bit, and 8-bit accesses are supported. Figure 24-7 shows the interrupt memory mapping. The table base address is 0xFFF82000.

Figure 24-7. VIM Interrupt Address Memory Map



Note:
The interrupt vector table only has 64 entries, one phantom vector and 63 interrupt channels. Channel 63 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 signalling 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.

24.4.2 Enabling and Controlling the VIM Parity

The polarity of VIM parity is controlled by the DEVCR1 register in the system module (address 0xFFFFFFDC). 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.

24.4.3 Interrupt Vector Table Initialization

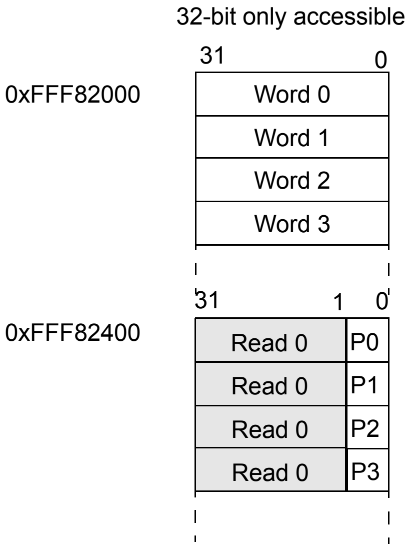
After reset, the interrupt vector table content, including the parity bits is not initialized. Therefore, the CPU need to initialize all of the interrupt addresses into the table, before enabling the corresponding interrupt channel. If parity is required, this initialization should be done after the parity functionality is enabled. In this way, the corresponding parity bit will be automatically updated. For more details on RAM hardware initialization , refer to Section 1.5. This initialization is only required when vectored interrupts are used, index interrupt management does not need the table to be initialized.

24.4.4 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 24-8. Parity Bit Mapping

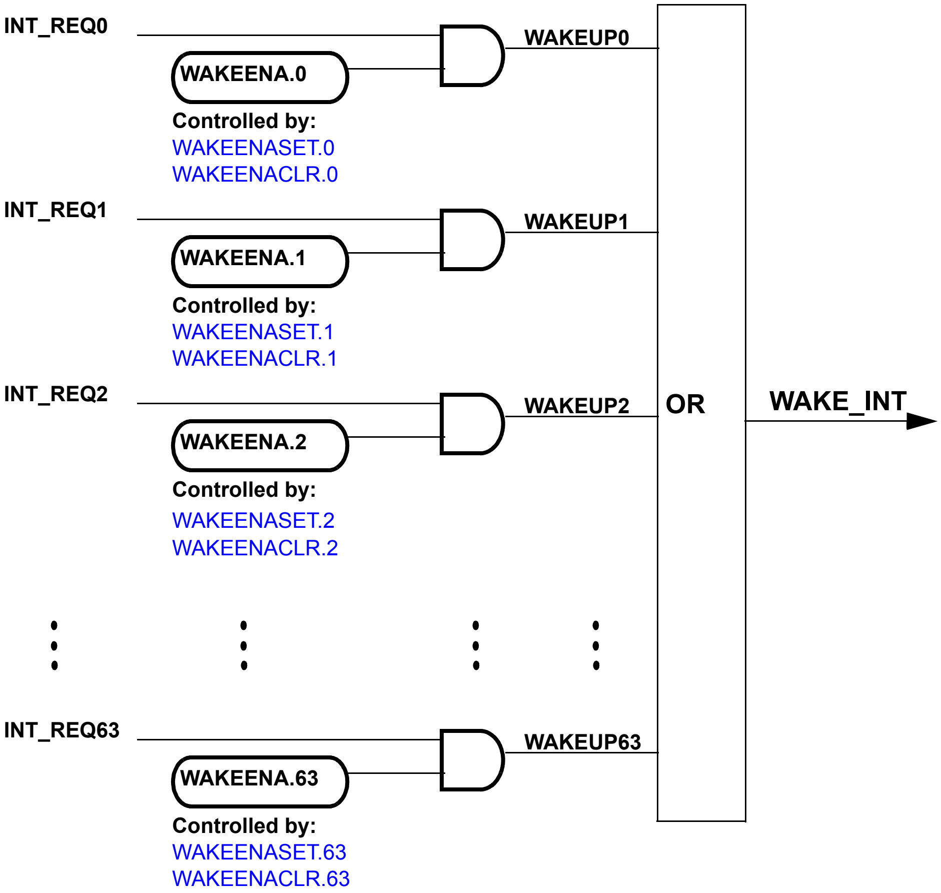


24.5 VIM Wakeup Interrupt

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 24-9, 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.

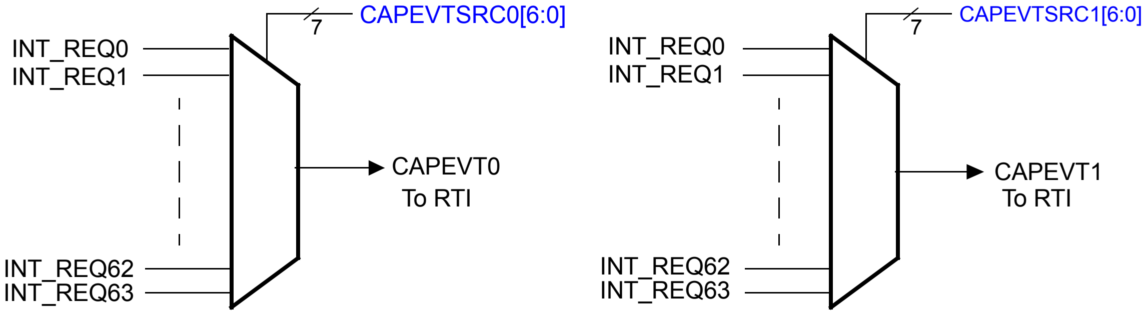
Figure 24-9. Detail of the IRQ Input



24.6 Capture Event Sources

The VIM can select any of the 64 interrupt request to generate capture events for the real-time interrupt (RTI) module (see Figure 24-10). The value in REQENASET / REQENACLR does NOT impact the capture event. Two registers (Section 24.8.16) are available to select, for each capture event source.

Figure 24-10. Capture Event Sources



24.7 Examples

The following sections provide examples about the operation of the VIM.

24.7.1 Examples - Configure CPU To Receive Interrupts

Example 1 shows how to set the vector enable (VE) bit in the CP15 R1 register to enable the hardware vector interrupt. Example 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 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 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 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
```

24.7.2 Examples - Register Vector Interrupt and Index Interrupt Handling

Example 3 illustrates 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 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 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 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 3 and Example 4 are valid. The difference is that the CPU will not read from the 0x18 location on the IRQ interrupt.

24.8 Registers

This section details the VIM module registers, summarized in Figure 24-11. A detailed description of each register and its bits is also provided.

Each register begins on a word boundary. All registers are 32-bit, 16-bit and 8-bit accessible. The start address of the VIM module is 0xFFFFFE00. The parity related VIM registers are placed at 0xFFFFFDEC~0xFFFFFDF8.

Figure 24-11. VIM Control Register Summary

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFFFDEC PARFLG Page 1781	Reserved															
	Reserved															PAR FLG
0xFFFFFDF0 PARCTL Page 1782	Reserved															
	Reserved							TEST	Reserved				PARENA[3:0]			
0xFFFFFDF4 ADDERR Page 1783	Interrupt Vector Table offset (0xFFF820)															
	Interrupt Vector Table offset (0xFFF820)							ADDERR[6:0]							Word offset (0)	
0xFFFFFDF8 FBPARERR Page 1784	FBPARERR[31:16]															
	FBPARERR[15:0]															
0xFFFFFE00 IRQINDEX Page 1786	Reserved															
	Reserved								IRQINDEX[7:0]							
0xFFFFFE04 FIQINDEX Page 1786	Reserved															
	Reserved								FIQINDEX[7:0]							
0xFFFFFE08– FFFFFE0C	Reserved															
	Reserved															

Figure 24-11. VIM Control Register Summary (Continued)

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
0xFFFFFE10 FIRQPR0 Page 1788	FIRQPR[31:16]															
	FIRQPR[15:2]														Reserved	
0xFFFFFE14 FIRQPR1 Page 1788	FIRQPR[63:48]															
	FIRQPR[47:32]															
0xFFFFFE18 - 0xFFFFFE1C	Reserved															
	Reserved															
0xFFFFFE20 INTREQ0 Page 1789	INTREQ[31:16]															
	INTREQ[15:0]															
0xFFFFFE24 INTREQ1 Page 1789	INTREQ[63:48]															
	INTREQ[47:32]															
0xFFFFFE28 - 0xFFFFFE2C	Reserved															
	Reserved															
0xFFFFFE30 REQENASET0 Page 1790	REQENASET[31:16]															
	REQENASET[15:2]														Reserved	
0xFFFFFE34 REQENASET1 Page 1790	REQENASET[63:48]															
	REQENASET[47:32]															

Figure 24-11. VIM Control Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFFFE38 - 0xFFFFFE3C	Reserved															
	Reserved															
0xFFFFFE40 REQENACLR0 Page 1791	REQENACLR[31:16]															
	REQENACLR[15:2]														Reserved	
0xFFFFFE44 REQENACLR1 Page 1791	REQENACLR[63:48]															
	REQENACLR[47:32]															
0xFFFFFE48 - 0xFFFFFE4C	Reserved															
	Reserved															
0xFFFFFE50 WAKEENASET0 Page 1792	WAKEENASET[31:16]															
	WAKEENASET[15:0]															
0xFFFFFE54 WAKEENASET1 Page 1792	WAKEENASET[63:48]															
	WAKEENASET[47:32]															
0xFFFFFE58 - 0xFFFFFE5C	Reserved															
	Reserved															
0xFFFFFE60 WAKEENACLR0 Page 1793	WAKEENACLR[31:16]															
	WAKEENACLR[15:0]															

Figure 24-11. VIM Control Register Summary (Continued)

Offset Address Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFFFE64 WAKEENACLR1 Page 1793	WAKEENACLR[63:48]															
	WAKEENACLR[47:32]															
0xFFFFFE68 - 0xFFFFFE6C	Reserved															
	Reserved															
0xFFFFFE70 IRQVECREG Page 1794	IRQ[31:16]															
	IRQVECREG[15:0]															
0xFFFFFE74 FIQVECREG Page 1795	FIQVECREG[31:16]															
	FIQVECREG[15:0]															
0xFFFFFE78 CAPEVT Page 1796	Reserved									CAPEVTSRC1[6:0]						
	Reserved									CAPEVTSRC0[6:0]						
0xFFFFFE7C	Reserved															
	Reserved															
0xFFFFFE80– 0xFFFFFEBC CHANCTRL[0:15] Page 1797	Res	CHANMAPx$_0$[6:0]							Res	CHANMAP1x$_1$[6:0]						
	Res	CHANMAPx$_2$[6:0]							Res	CHANMAPx$_3$[6:0]						

24.8.1 Interrupt Vector Table Parity Flag Register (PARFLG)

Figure 24-12 and Table 24-1 describe this register.

Figure 24-12. Interrupt Vector Table Parity Flag Register (PARFLG) [0xFFFFFDEC]

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
Reserved															PARFLG
R-0															R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-1. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions

Bit	Name	Value	Description
31–1	Reserved		Reads return zero and 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.

24.8.2 Interrupt Vector Table Parity Control Register (PARCTL)

Figure 24-13. Interrupt Vector Table Parity Control Register (PARCTL) [0xFFFFFDF0]

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
Reserved							TEST	Reserved				PARENA[3:0]			
R-0							R/WP-0	R-0				R/WP-0101			

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-2. Interrupt Vector Table Parity Control Register (PARCTL)Field Descriptions

Bit	Name	Value	Description
31–9	Reserved		Reads return zero and 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		Reads return 0 and writes have no effect.
3–0	PARENA[3:0]		VIM parity enable.
		0101	The VIM parity is disabled.
		all other values	The VIM parity is enabled. **Note: It is recommended to write 1010b to enable PARENA, to guard against soft error from flipping PARENA to a disabled state.**

24.8.3 Address Parity Error Register (ADDERR)

Note: The address parity error register gives the address of the first parity error location. No computation is needed when reading the complete register to retrieve the address in the Interrupt Vector Table.

Note:
This register will never be reset by a power-on reset nor any other reset source.

Figure 24-14. Address Parity Error Register (ADDERR) [0xFFFFFDF4]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt Vector Table offset															
R-FFF8h															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Interrupt Vector Table offset							ADDERR[6:0]							Word offset[1:0]	
R-20h							R-x							R-00	

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-3. Address Parity Error Register (PARERR) Field Descriptions

Bit	Name	Value	Description
31–9	Interrupt Vector Table offset		Interrupt Vector Table offset. Reads are always 0xFFF820; writes have no effect
8–2	ADDERR[6:0]		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 24.8.1)
1–0	Word offset[1:0]		Word offset. Reads are always 0x0; writes have no effect.

24.8.4 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 24-15 and Table 24-4 describe this register.

Note:
This register will never be reset by a power-on reset nor any other reset source.

Figure 24-15. Fall-Back Address Parity Error Register (FBPARERR) [0xFFFFFDF8]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FBPARERR[31:16]															
R/WP-x															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBPARERR[15:0]															
R/WP-x															

R = Read in all modes; WP = Write in privilege mode only; -*x* = Indeterminate

Table 24-4. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions

Bit	Name	Value	Description
31–0	FBPARERR[31:0]	0–0xFFFF FFFF	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.

24.8.5 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 24-5.

Table 24-5. Interrupt Dispatch

IRQINDEX / FIQINDEX Register Bit Field	Highest Priority Pending Interrupt Enabled
0x00	No interrupt
0x01	Channel 0
:	:
0x39	Channel 62
0x40	Channel 63

Note:
Channel 63 has no dedicated interrupt vector table entry. Therefore, Channel 63 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.

24.8.6 *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 24-16 and Table 24-6 describe this register.

Figure 24-16. IRQ Index Offset Vector Register (IRQINDEX) [0xFFFFFE00]

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
Reserved								IRQINDEX(7-0)							
R-0								R-0							

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Table 24-6. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect.
7-0	IRQINDEX(7-0)	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 24-5. When no interrupts are pending, the least significant byte of IRQINDEX is 0x00.

24.8.7 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 24-17 and Table 24-7 describe this register.

Figure 24-17. FIQ Index Offset Vector Register (FIQINDEX) [0xFFFFFE04]

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
Reserved								FIQINDEX(7-0)							
R-0								R-0							

R = Read in all modes; -n = value after reset

Table 24-7. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions

Bit	Name	Value	Description
31–8	Reserved		Reads are undefined and writes have no effect.
7-0	FIQINDEX(7-0)	0–0xFF	FIQ index offset vector. The least significant bits represent the index of the FIQ pending interrupt with the highest precedence, as shown in Table 24-5. When no interrupts are pending, the least significant byte of FIQINDEX is 0x00.

24.8.8 FIQ/IRQ Program Control Registers[0:1] (FIRQPR[0:1])

The FIQ/IRQ program control registers (FIRQPR[0:1]) determine whether a given interrupt request will be either FIQ or IRQ. Figure 24-18-Figure 24-19 and Table 24-8 describe these registers.

Note:
Channel 0 and 1 are FIQ only, not impacted by this register.

Figure 24-18. FIQ/IRQ Program Control Register 0 (FIRQPR0) [0xFFFFFE10]

31-16
FIRQPR[31:16]
R/WP-0

15-2	1-0
FIRQPR[15:0]	Reserved
R/WP-0	R-1

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-19. FIQ/IRQ Program Control Register 1 (FIRQPR1) [0xFFFFFE14]

31-16
FIRQPR[63:48]
R/WP-0

15-0
FIRQPR[47:32]
R/WP-0

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-8. FIQ/IRQ Program Control Registers[0:1] (FIRQPR[0:1]) Field Descriptions

Bit	Name	Value	Description
63-2	FIRQPR[63:2]		FIQ/IRQ program control bits. These bits determine whether an interrupt request from a peripheral is of type FIQ or IRQ. Bit FIRQPRx[63:2] corresponds to request channel[63:2].
		0	Interrupt request is of IRQ type.
		1	Interrupt request is of FIQ type.
1:0	Reserved	1	Read only. A write has no impact.

24.8.9 Pending Interrupt Read Location Registers[0:1] (INTREQ[0:1])

The pending interrupt register gives the pending interrupt requests. The register is updated every vbus clock cycle. Figure 24-20 - Figure 24-21 and Table 24-9 describe this register.

Figure 24-20. Pending Interrupt Read Location Register 0 (INTREQ0) [0xFFFFFE20]

31–16
INTREQ[31:16]
R-0

15–0
INTREQ[15:0]
R-0

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-21. Pending Interrupt Read Location 1 (INTREQ1) Register [0xFFFFFE24]

31–16
INTREQ[63:48]
R-0

15–0
INTREQ[47:32]
R-0

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-9. Pending Interrupt Read Registers[0:1] (INTREQ[0:1]) Field Descriptions

Bit	Name	Value	Description
63–0	INTREQ[63:0]		Pending interrupt bits. These bits determine whether an interrupt request is pending for the request channel between 0 and 63. The interrupt ENABLE register does not affect the value of the interrupt pending bit. Bit INTREQx[63:0] corresponds to request channel[63:0].
		0	No interrupt event has occurred.
		1	An interrupt is pending.

24.8.10 Interrupt Enable Set Registers[0:1] (REQENASET[0:1])

The interrupt register enable selectively enables individual request channels. Figure 24-22 - Figure 24-23 and Table 24-10 describe these registers.

Note:
Channel 0 and 1 are always enabled, not impacted by this register.

Figure 24-22. Interrupt Enable Set Register 0 (REQENASET0) [0xFFFFFE30]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REQENASET[31:16]															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REQENASET[15:2]														Reserved	
R/WP-0														R-1	

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-23. Interrupt Enable Set Register 1 (REQENASET1) [0xFFFFFE34]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REQEnableSET[63:48]															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REQENASET[47:32]															
R/WP-0															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-10. Interrupt Enable Set Register[0:1] (REQENASET[0:1]) Field Descriptions

Bit	Name	Value	Description
63–2	REQENASET[63:2]		Request enable set bits. This vector determines whether the interrupt request channel is enabled. Bit REQENASETx[63:2] corresponds to request channel[63:2].
		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	1	Read only. A write has no impact.

24.8.11 Interrupt Enable Clear Registers[0:1] (REQENACLR[0:1])

The interrupt register enable selectively disables individual request channels. Figure 24-24-Figure 24-25 and Table 24-11 describe these registers.

> **Note:**
> Channel 0 and 1 are always enabled, not impacted by this register.

Figure 24-24. Interrupt Enable Clear Register 0 (REQENACLR0) [0xFFFFFE40]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">REQENACLR[31:16]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">REQENACLR[15:2]</td><td colspan="2">Reserved</td></tr>
<tr><td colspan="14">R/WP-0</td><td colspan="2">R-1</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-25. Interrupt Enable Clear Register 1 (REQENACLR1) [0xFFFFFE44]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">REQENACLR[63:48]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">REQENACLR[47:32]</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-11. Request Enable Clear Register (REQENACLR) Field Descriptions

Bit	Name	Value	Description
63:2	REQENACLR[63:2]		Request enable clear bits. This vector determines whether the interrupt request channel is enabled. Bit REQENACLRx[63:2] corresponds to request channel[63:2].
		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	1	Read only. A write has no impact.

24.8.12 Wake-Up Enable Set Registers[0:1] (WAKEENASET[0:1])

The wake-up enable registers selectively enables individual wake-up interrupt request lines. Figure 24-26-Figure 24-27 and Table 24-12 describe these registers.

Figure 24-26. Wake-Up Enable Set Register 0 (WAKEENASET0) [0xFFFFFE50]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">WAKEENASET[31:16]</td></tr>
<tr><td colspan="16">R/WP-0xFFFF</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">WAKEENASET[15:0]</td></tr>
<tr><td colspan="16">R/WP-0xFFFF</td></tr>
</table>

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-27. Wake-Up Enable Set Register 1 (WAKEENASET1) [0xFFFFFE54]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">WAKEENASET[63:48]</td></tr>
<tr><td colspan="16">R/WP-0xFFFF</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">WAKEENASET[47:32]</td></tr>
<tr><td colspan="16">R/WP-0xFFFF</td></tr>
</table>

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-12. Wake-Up Enable Set Registers[0:1] (WAKEENASET[0:1]) Field Descriptions

Bit	Name	Value	Description
63–0	WAKEENASET[63:0]		Wake-up enable set bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENASETx[63:0] corresponds to interrupt request channel[63:0].
		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.

24.8.13 Wake-Up Enable Clear Registers[0:1] (WAKEENACLR[0:1])

The wake-up enable register selectively disables individual wake-up interrupt request lines. Figure 24-28-Figure 24-29 and Table 24-13 describe these registers.

Figure 24-28. Wake-Up Enable Clear Register 0 (WAKEENACLR0) [0xFFFFFE60]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEENACLR[31:16]															
R/WP-0xFFFF															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEENACLR[15:0]															
R/WP-0xFFFF															

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Figure 24-29. Wake-Up Enable Clear Register 1 (WAKEENACLR1) [0xFFFFFE64]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WAKEENACLR[63:48]															
R/WP-0xFFFF															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEENACLR[47:32]															
R/WP-0xFFFF															

R = Read in all modes; WP = Write in privilege mode only; -n = value after reset

Table 24-13. Wake-Up Enable Clear Registers[0:1] (WAKEENACLR[0:1]) Field Descriptions

Bit	Name	Value	Description
63–0	WAKEENACLR[63:0]		Wake-up enable clear bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENACLRx[63:0] corresponds to interrupt request channel[63:0].
		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.

24.8.14 IRQ Interrupt Vector Register (IRQVECREG)

The interrupt vector register gives the address of the enabled and active IRQ interrupt. Figure 24-30 and Table 24-14 describe these registers.

Figure 24-30. IRQ Interrupt Vector Register (IRQVECREG) [0xFFFFFE70]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQVECREG(15-0)															
R-0															

R = Read in all modes; WP = Write in privileged mode only; -n = value after reset

Table 24-14. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions

Bit	Name	Value	Description
31–0	IRQVECREG(31-0)	From Interrupt Vector Table	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.

24.8.15 FIQ Interrupt Vector Register (FIQVECREG)

The interrupt vector register gives the address of the enabled and active FIQ interrupt. Figure 24-31 and Table 24-15 describe these registers.

Figure 24-31. IRQ Interrupt Vector Register (*FIQVECREG*) [0xFFFFFE74]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIQVECREG(15-0)															
R-0															

R = Read in all modes; WP = Write in privileged mode only; -*n* = value after reset

Table 24-15. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions

Bit	Name	Value	Description
31–0	FIQVECREG(31-0)	From Interrupt Vector Table	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.

24.8.16 Capture Event Register (CAPEVT)

Figure 24-32 and Table 24-16 describe this register.

Figure 24-32. Capture Event Register (CAPEVT) [0xFFFFFE78]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									CAPEVTSRC1(6-0)						
R-U									R/WP-0x0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									CAPEVTSRC0(6-0)						
R-U									R/WP-0x0						

R = Read, W = Write, WP = Write from privileged mode only, S = Set, U = Undefined, -*n* = Value after reset (see table)

Table 24-16. Capture Event Register (CAPEVT) Field Descriptions

Bit	Name	Value	Description
31–23	Reserved		Reads are indeterminate and writes have no effect.
22–16	CAPEVTSRC1[6:0]		Capture event source 1 mapping control. These bits determine which interrupt request maps to the capture event source 1 of the RTI:
		0000000	Interrupt request 0.
		0000001	Interrupt request 1.
		...	...
		0x3E	Interrupt request 63.
15–7	Reserved		Reads are indeterminate and writes have no effect.
6–0	CAPEVTSRC0[6:0]		Capture event source 0 mapping control. These bits determine which interrupt request maps to the capture event source 0 of the RTI:
		0000000	Interrupt request 0.
		0000001	Interrupt request 1.
		...	...
		0x3F	Interrupt request 63.

24.8.17 VIM Interrupt Control Register[0:31] (CHANCTRL[0:15])

Sixteen interrupt control registers control the 64 interrupt channels of the VIM. Each register controls four interrupt channels: each of them is indexed from 0 to 64. Table 24-17 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 24-33 and Table 24-18 describe these registers.

Table 24-17. Interrupt Control Registers Organization

Offset Address[1]	Register Mnemonic	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
0xFFFFFE80	CHANCTRL0	CHANMAP0	CHANMAP1	CHANMAP2	CHANMAP3	0x00010203
0xFFFFFE84	CHANCTRL1	CHANMAP4	CHANMAP5	CHANMAP6	CHANMAP7	0x04050607
:	:	:	:	:	:	:
:	:	:	:	:	:	:
:	:	:	:	:	:	:
0xFFFFFEB8	CHANCTRL14	CHANMAP58	CHANMAP57	CHANMAP58	CHANMAP59	0x38393A3B
0xFFFFFEBC	CHANCTRL15	CHANMAP60	CHANMAP61	CHANMAP62	CHANMAP63	0x3C3D3E3F

Note:
CHANMAP0 and CHANMAP1 are not programable. CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1.

Note:
Do NOT write any value other than 0x3F to CHANMAP63. Channel 63 is reserved because no interrupt vector table entry supports this channel.

Figure 24-33. Interrupt Control Registers[0:31] (CHANCTRL[0:15]) [0xFFFFFE80 to 0xFFFFBC]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	CHANMAPx$_0$[6:0]							Res	CHANMAPx$_1$[6:0]						
R-U	R/WP-see Table 24-17								R/WP-see Table 24-17						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	CHANMAPx$_2$[6:0]							Res	CHANMAPx$_3$[6:0]						
R-U	R/WP-see Table 24-17							R-U	R/WP-see Table 24-17						

R = Read, W = Write, WP = Write from privileged mode only, S = Set, U = Undefined, -*n* = Value after reset (see table)

Table 24-18. Interrupt Control Registers[0:31] (CHANCTL[0:15] Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads are indeterminate and writes have no effect.
30-24	CHANMAPx$_0$[6:0]		CHANMAPx$_0$(6–0). Interrupt CHANx$_0$ mapping control. These bits determine which interrupt request the priority channel CHANx$_0$ maps to:
		0000000	*Read:* Interrupt request 0 maps to channel priority CHANx$_0$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_0$ is set with the interrupt request.
		0000001	*Read:* Interrupt request 1 maps to channel priority CHANx$_0$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_0$ is set with the interrupt request.
		. . .	. . .
		0x3F	*Read:* Interrupt request 63 maps to channel priority CHANx$_0$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_0$ is set with the interrupt request.
23	Reserved		Reads are indeterminate and writes have no effect.

Table 24-18. Interrupt Control Registers[0:31] (CHANCTL[0:15] Field Descriptions (Continued)

Bit	Name	Value	Description
22–16	CHANMAPx$_1$[6:0]		CHANMAPx$_1$(6–0). Interrupt CHANx$_1$ mapping control. These bits determine which interrupt request the priority channel CHANx$_1$ maps to:
		0000000	*Read:* Interrupt request 0 maps to channel priority CHANx$_1$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_1$ is set with the interrupt request.
		0000001	*Read:* Interrupt request 1 maps to channel priority CHANx$_1$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_1$ is set with the interrupt request.
		. . .	. . .
		0x3F	*Read:* Interrupt request 63 maps to channel priority CHANx$_1$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_1$ is set with the interrupt request.
15	Reserved		Reads are indeterminate and writes have no effect.
14–8	CHANMAPx$_2$[6:0]		CHANMAPx$_2$(6–0). Interrupt CHANx$_2$ mapping control. These bits determine which interrupt request the priority channel CHANx$_2$ maps to:
		0000000	*Read:* Interrupt request 0 maps to channel priority CHANx$_2$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_2$ is set with the interrupt request.
		0000001	*Read:* Interrupt request 1 maps to channel priority CHANx$_2$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_2$ is set with the interrupt request.
		. . .	. . .
		0x3F	*Read:* Interrupt request 63 maps to channel priority CHANx$_2$. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx$_2$ is set with the interrupt request.
7	Reserved		Reads are indeterminate and writes have no effect.

Table 24-18. Interrupt Control Registers[0:31] (CHANCTL[0:15] Field Descriptions (Continued)

Bit	Name	Value	Description
6–0	CHANMAPx_3[6:0]		CHANMAPx_3(6–0). Interrupt CHANx_3 mapping control. These bits determine which interrupt request the priority channel CHAN$\mathbf{x_3}$ maps to:
		0000000	*Read:* Interrupt request 0 maps to channel priority CHANx_3. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx_3 is set with the interrupt request.
		0000001	*Read:* Interrupt request 1 maps to channel priority CHANx_3. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx_3 is set with the interrupt request.
		. . .	. . .
		0x3F	*Read:* Interrupt request 63 maps to channel priority CHANx_3. *Write:* The default value of this bit after reset is given in Table 24-17. The channel priority CHANx_3 is set with the interrupt request.

Chapter 25

SPNU489C-February 2012

Error Signaling Module (ESM)

This reference guide provides the specification for a module that indicates a severe device failure at an external signal. The error indication is generated additionally to the error interrupt when the failure is recognized from any detection unit. The external signal is realized as a digital hardware output pin.

Topic   Page

25.1 Overview

The Error Signaling Module (ESM) manages the various error conditions on the TMS570 microcontroller. The error condition is handled based on a severity level assigned to it during the device integration time. Any error condition can be configured to drive a dedicated device pin called $\overline{\text{ERROR}}$ low. This can be used as an indicator to an external monitor circuit to reset the microcontroller if necessary. Therefore the external device is able to keep the entire system in a fail-safe state by disabling the peripherals outside of the ECU.

25.1.1 Feature List

- Up to 96 interrupt/error channels are supported, divided into 3 different groups
  - 32 channels with maskable interrupt and configurable $\overline{\text{ERROR}}$ pin behavior
  - 32 channels with non-maskable interrupt and predefined $\overline{\text{ERROR}}$ pin behavior
  - 32 channels with predefined $\overline{\text{ERROR}}$ pin behavior only
- Dedicated device $\overline{\text{ERROR}}$ pin to signal severe device failure
- Configurable timebase for error signal
- Error forcing capability for testing

25.2 Block Diagram

As shown in Figure 25-1, the ESM channels are divided into three groups. Group1 is implemented with maskable interrupt and configurable ERROR pin behavior. The user shall map the group1 errors critical to their application to the ERROR pin. All the channels in Group2 are considered as critical. Group 2 is implemented with non-maskable interrupt and predefined ERROR pin behavior. Group3 is implemented with predefined ERROR pin behavior only. The total active time of the ERROR pin is controlled by the Low-Time Counter Preload register (LTCP) as shown in Figure 25-2.

Figure 25-1. Block Diagram

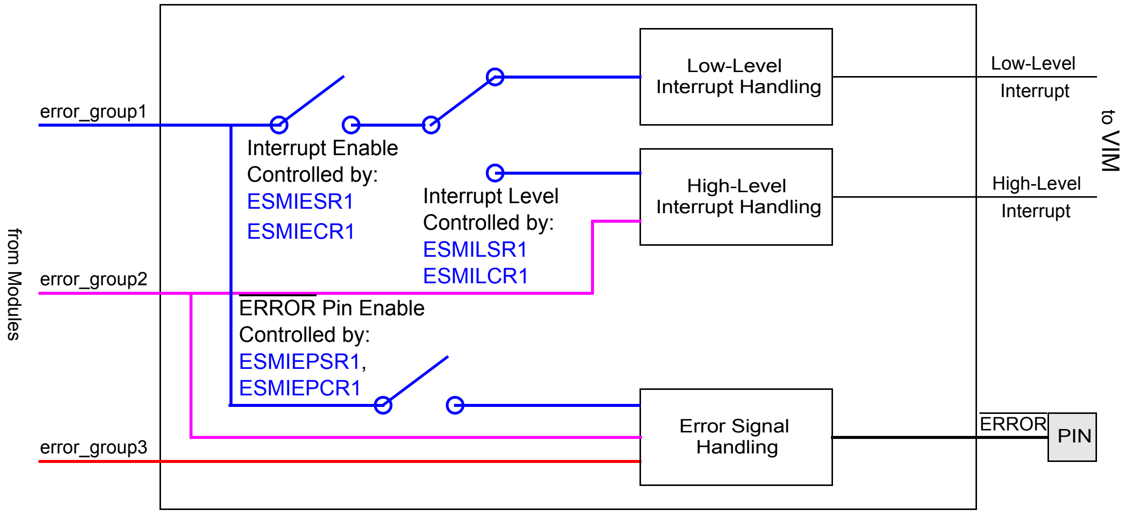
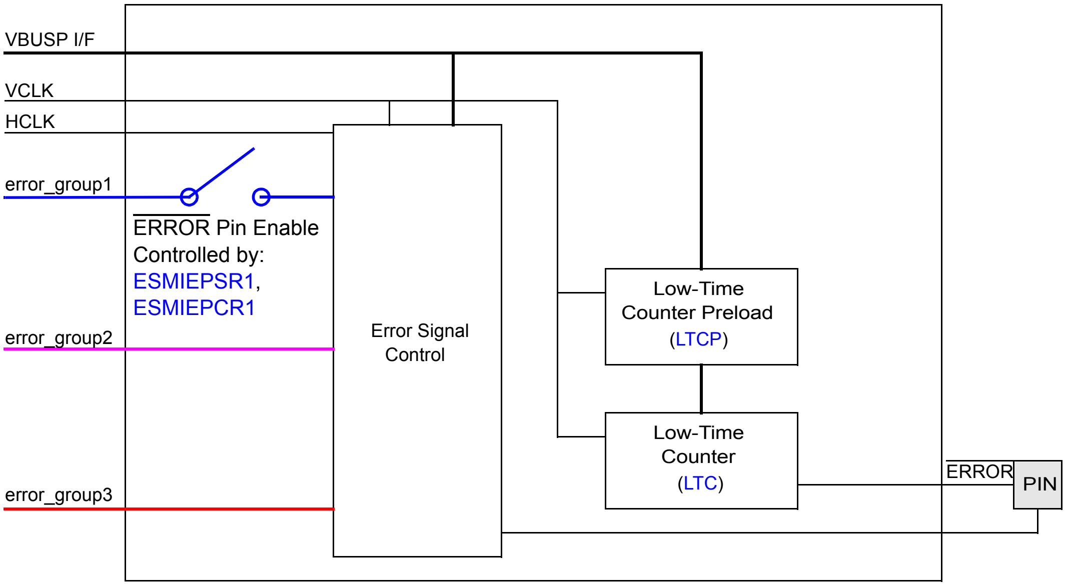


Figure 25-2. Error Signal Handling



25.3 Module Operation

The ESM module maintains the error flags until power-on reset (PORRST) is asserted. This allows the application to identify the source of the error for diagnostic reasons. All error flags can be cleared by the application.

The ESM controls the enabling and disabling functionality of each error signal. The errors deemed critical are not maskable. The ESM module is able to generate a non-maskable interrupt to the CPU in case of critical errors. The ESM also optionally drives a dedicated $\overline{\text{ERROR}}$ device pin to indicate an error condition.

The generating of interrupts and the activation of the $\overline{\text{ERROR}}$ pin is shown in Table 25-1 below:

Table 25-1. ESM interrupt and $\overline{\text{ERROR}}$ pin behavior

Error Group	Interrupt, Level	Influence on the $\overline{\text{ERROR}}$ Pin
1	maskable, low/high	configurable
2	non-maskable, high	fix
3	non-maskable, ---	fix

Please refer to the specific device datasheet for ESM assignment details.

25.3.1 Reset Status

The $\overline{\text{ERROR}}$ pin gets into “output in-active” (high impedance) state with pull-down enabled during power-on reset and into “output active” state with pull-down disabled during any other reset.

After PORRST, all registers in ESM module will be re-initialized to the default value. All the error status registers are cleared to zero.

After nRST, ESMSR1, ESMSSR2, ESMSR3 and ESMEPSR register’s value remains un-changed. The $\overline{\text{ERROR}}$ pin remains unchanged after nRST. Since nRST doesn’t clean the critical failure registers, the user can read those registers to debug the failures even if a nRST is triggered.

After nRST, if one of the flags in ESMSR1 is set, the interrupt service routine will be called once the corresponding interrupt is enabled.

Note:
ESMSR2 is cleared after nRST. The flag in ESMSR2 gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1 and the shadow register ESMSSR2. Reading ESMIOFFLR will not clear the ESMSR1 too.

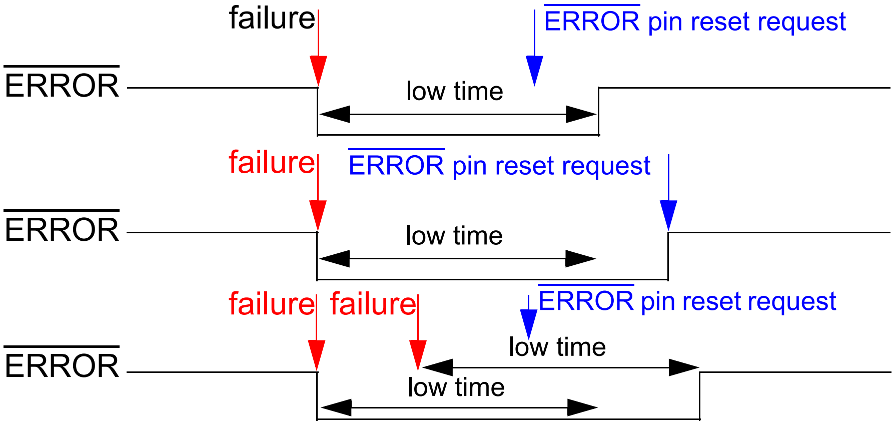
25.3.2 $\overline{\text{ERROR}}$ Pin Timing

The $\overline{\text{ERROR}}$ pin is an active low function. The state of the pin is also readable by software. An internal reset does not affect the state of the pin. 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

$$t_{ERROR\_low} = t_{VCLK} * (LTCP + 1) \qquad \text{(EQ 1)}$$

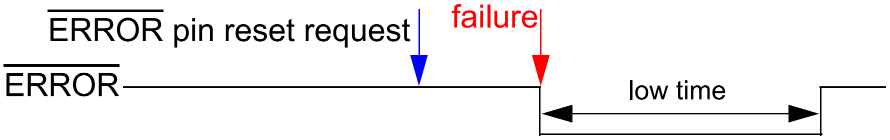
Once this period expires, the $\overline{\text{ERROR}}$ pin is set to high in case the reset of the $\overline{\text{ERROR}}$ pin was requested by the software during the $\overline{\text{ERROR}}$ pin low time by writing an appropriate key to the key register (ESMEKR). If another failure occurs within this time the pin stays low and the low time counter must be reset. The error detection mechanisms cannot be disabled. (see Figure 25-3)

Figure 25-3. ERROR pin timing



If the error low time period expires, the ERROR pin will be set to high in case the reset of the ERROR pin was requested by the software even before the ERROR pin low time has started (Figure 25-4). This case is not recommended and should be avoided by the application.

Figure 25-4. ERROR pin timing



25.3.3 Forcing an Error Condition

The error detection mechanism is testable by software by forcing an error condition. This allows testing not only the ERROR pin functionality but also the signal reaching the ERROR pin activation unit. By writing a dedicated key to the error forcing key register (ESMEKR) the ERROR pin is set to low for the specified time.

Once the application puts the ESM module in the error forcing mode, the application cannot handle the normal error functionality during this time.

If a failure occurs while the ESM module is in the error forcing mode, 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 ERROR pin low until the expiration of the LTC.

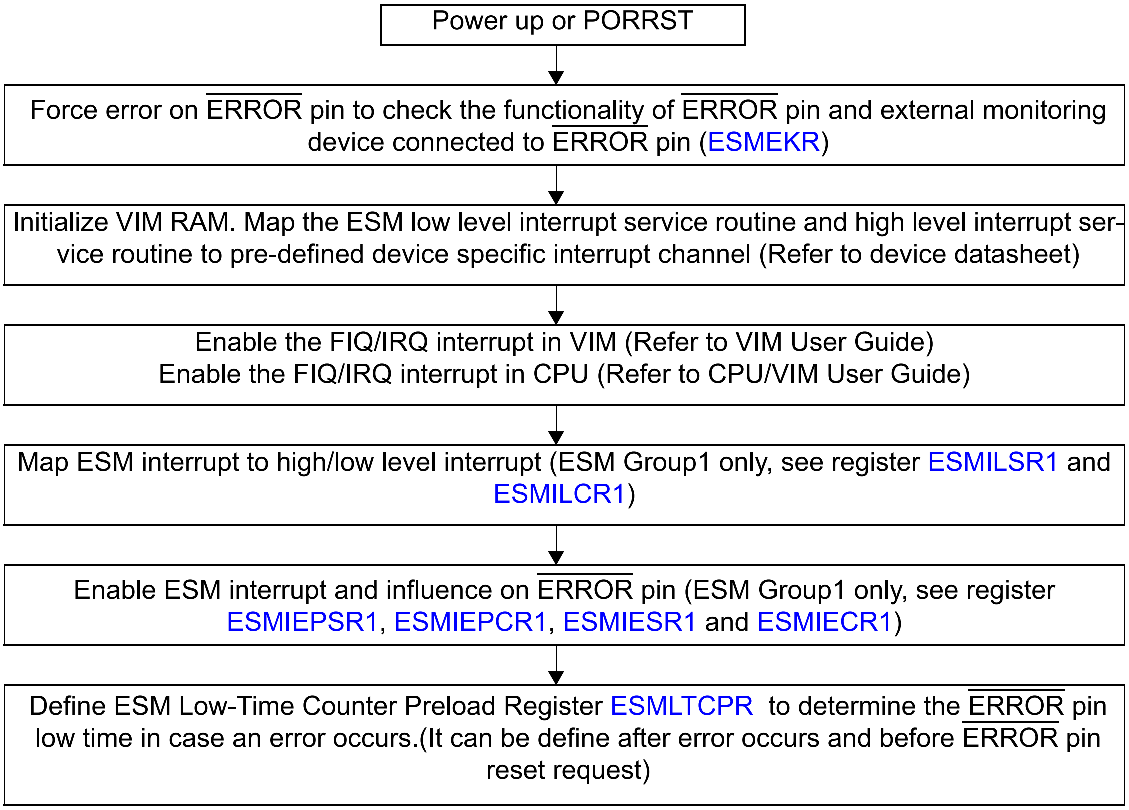
If there are no errors detected while the ESM module is in the error forcing mode, the ERROR pin goes High immediately after exiting the error forcing mode.

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. The application software must check for pending failures before requesting entry into error forcing mode.

25.4 Recommended Programming Procedure

During initialization stage, the application code should follow the recommendations in Figure 25-5 to initialize the ESM. The users shall map the group1 errors critical to their application to the $\overline{\text{ERROR}}$ pin.

Figure 25-5. ESM Initialization



Power up or PORRST

Force error on $\overline{\text{ERROR}}$ pin to check the functionality of $\overline{\text{ERROR}}$ pin and external monitoring device connected to ERROR pin (ESMEKR)

Initialize VIM RAM. Map the ESM low level interrupt service routine and high level interrupt service routine to pre-defined device specific interrupt channel (Refer to device datasheet)

Enable the FIQ/IRQ interrupt in VIM (Refer to VIM User Guide)
Enable the FIQ/IRQ interrupt in CPU (Refer to CPU/VIM User Guide)

Map ESM interrupt to high/low level interrupt (ESM Group1 only, see register ESMILSR1 and ESMILCR1)

Enable ESM interrupt and influence on $\overline{\text{ERROR}}$ pin (ESM Group1 only, see register ESMIEPSR1, ESMIEPCR1, ESMIESR1 and ESMIECR1)

Define ESM Low-Time Counter Preload Register ESMLTCPR to determine the $\overline{\text{ERROR}}$ pin low time in case an error occurs.(It can be define after error occurs and before ERROR pin reset request)

If the ESM indicates detection of a non-recoverable fault within the device, the system shall be held in a fail-safe state (defined by the application) by means of other measures. The external monitoring device can assert nRST to put all I/Os in high impedance mode.

After the $\overline{\text{ERROR}}$ pin outputs low indicating a non-recoverable fault, it can be released from this state by PORRST or write 0x5 to EKEY field in the ESM Error Key Register (ESMEKR).

After an error occurs, the application can read the error status registers (ESMSR1, ESMSR2 and ESMSR3) to debug the error. If a nRST is triggered or the error interrupt has been served, the error flag of group 2 should be read from ESMSSR2 because the error flag in ESMSR2 might be cleared.

25.5 Control Registers

This section describes the ESM registers. Each register begins on a word boundary. The registers support 32-bit, 16-bit and 8-bit accesses. The base address for the control registers is 0xFFFFF500.

Table 25-2. Module 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
0xFFFFF500 ESMIEPSR1 Page 1809	IEPSET131	IEPSET130	IEPSET29	IEPSET28	IEPSET27	IEPSET26	IEPSET25	IEPSET24	IEPSET23	IEPSET22	IEPSET21	IEPSET20	IEPSET19	IEPSET18	IEPSE17	IEPSET16
	IEPSET15	IEPSET14	IEPSET13	IEPSET12	IEPSET11	IEPSET10	IEPSET9	IEPSET8	IEPSET7	IEPSET6	IEPSET5	IEPSET4	IEPSET3	IEPSET2	IEPSET1	IEPSET0
0xFFFFF504 ESMIEPCR1 Page 1810	IEPCLR31	IEPCLR30	IEPCLR29	IEPCLR28	IEPCLR27	IEPCLR26	IEPCLR25	IEPCLR24	IEPCLR23	IEPCLR22	IEPCLR21	IEPCLR20	IEPCLR19	IEPCLR18	IEPCLR17	IEPCLR16
	IEPCLR15	IEPCLR14	IEPCLR13	IEPCLR12	IEPCLR11	IEPCLR10	IEPCLR9	IEPCLR8	IEPCLR7	IEPCLR6	IEPCLR5	IEPCLR4	IEPCLR3	IEPCLR2	IEPCLR1	IEPCLR0
0xFFFFF508 ESMIESR1 Page 1811	INTENSET31	INTENSET30	INTENSET29	INTENSET28	INTENSET27	INTENSET26	INTENSET25	INTENSET24	INTENSET23	INTENSET22	INTENSET21	INTENSET20	INTENSET19	INTENSET18	INTENSET17	INTENSET16
	INTENSET15	INTENSET14	INTENSET13	INTENSET12	INTENSET11	INTENSET10	INTENSET9	INTENSET8	INTENSET7	INTENSET6	INTENSET5	INTENSET4	INTENSET3	INTENSET2	INTENSET1	INTENSET0
0xFFFFF50C ESMIECR1 Page 1812	INTENCLR31	INTENCLR30	INTENCLR29	INTENCLR28	INTENCLR27	INTENCLR26	INTENCLR25	INTENCLR24	INTENCLR23	INTENCLR22	INTENCLR21	INTENCLR20	INTENCLR19	INTENCLR18	INTENCLR17	INTENCLR16
	INTENCLR15	INTENCLR14	INTENCLR13	INTENCLR12	INTENCLR11	INTENCLR10	INTENCLR9	INTENCLR8	INTENCLR7	INTENCLR6	INTENCLR5	INTENCLR4	INTENCLR3	INTENCLR2	INTENCLR1	INTENCLR0
0xFFFFF510 ESMILSR1 Page 1813	INTLVLSET31	INTLVLSET30	INTLVLSET29	INTLVLSET28	INTLVLSET27	INTLVLSET26	INTLVLSET25	INTLVLSET24	INTLVLSET23	INTLVLSET22	INTLVLSET21	INTLVLSET20	INTLVLSET19	INTLVLSET18	INTLVLSET17	INTLVLSET16
	INTLVLSET15	INTLVLSET14	INTLVLSET13	INTLVLSET12	INTLVLSET11	INTLVLSET10	INTLVLSET9	INTLVLSET8	INTLVLSET7	INTLVLSET6	INTLVLSET5	INTLVLSET4	INTLVLSET3	INTLVLSET2	INTLVLSET1	INTLVLSET0
0xFFFFF514 ESMILCR1 Page 1814	INTLVLCLR31	INTLVLCLR30	INTLVLCLR29	INTLVLCLR28	INTLVLCLR27	INTLVLCLR26	INTLVLCLR25	INTLVLCLR24	INTLVLCLR23	INTLVLCLR22	INTLVLCLR21	INTLVLCLR20	INTLVLCLR19	INTLVLCLR18	INTLVLCLR17	INTLVLCLR16
	INTLVLCLR15	INTLVLCLR14	INTLVLCLR13	INTLVLCLR12	INTLVLCLR11	INTLVLCLR10	INTLVLCLR9	INTLVLCLR8	INTLVLCLR7	INTLVLCLR6	INTLVLCLR5	INTLVLCLR4	INTLVLCLR3	INTLVLCLR2	INTLVLCLR1	INTLVLCLR0
0xFFFFF518 ESMSR1 Page 1815	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0

Table 25-2. Module Registers (Continued)

0xFFFFF51C ESMSR2 Page 1816	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
0xFFFFF520 ESMSR3 Page 1817	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
0xFFFFF524 ESMEPSR Page 1818	Reserved															
	Reserved															EPSF
0xFFFFF528 ESMIOFFHR Page 1819	Reserved															
	Reserved									INTOFFH[6:0]						
0xFFFFF52C ESMIOFFLR Page 1820	Reserved															
	Reserved									INTOFFL[6:0]						
0xFFFFF530 ESMLTCR Page 1821	Reserved															
	LTC[15:0]															
0xFFFFF534 ESMLTCPR Page 1822	Reserved															
	LTCP[15:0]															
0xFFFFF538 ESMEKR Page 1823	Reserved															
	Reserved												EKEY[3:0]			
0xFFFFF53C ESMSSR2 Page 1824	ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
	ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0

25.5.1 ESM Influence $\overline{\text{ERROR}}$ Pin Set/Status Register 1 (ESMIEPSR1)

The base address for this register is 0xFFFFF500.

Figure 25-6. ESM Influence $\overline{\text{ERROR}}$ Pin Set/Status Register 1 (ESMIEPSR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEPSET31	IEPSET30	IEPSET29	IEPSET28	IEPSET27	IEPSET26	IEPSET25	IEPSET24	IEPSET23	IEPSET22	IEPSET21	IEPSET20	IEPSET19	IEPSET18	IEPSE17	IEPSET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPSET15	IEPSET14	IEPSET13	IEPSET12	IEPSET11	IEPSET10	IEPSET9	IEPSET8	IEPSET7	IEPSET6	IEPSET5	IEPSET4	IEPSET3	IEPSET2	IEPSET1	IEPSET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-3. ESM Influence $\overline{\text{ERROR}}$ Pin Set/Status Register 1 (ESMIEPSR1)

Bit	Name	Value	Description
31-0	IEPSET		Set Influence on $\overline{\text{ERROR}}$ Pin
		0	**User and privileged mode:** Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR1 register unchanged.
		1	**User and privileged mode:** Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. **Privileged mode:** Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR1 register.

25.5.2 ESM Influence $\overline{\text{ERROR}}$ Pin Clear/Status Register 1 (ESMIEPCR1)

The base address for this register is 0xFFFFF504.

Figure 25-7. ESM Influence $\overline{\text{ERROR}}$ Pin Clear/Status Register 1 (ESMIEPCR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IEPCLR31	IEPCLR30	IEPCLR29	IEPCLR28	IEPCLR27	IEPCLR26	IEPCLR25	IEPCLR24	IEPCLR23	IEPCLR22	IEPCLR21	IEPCLR20	IEPCLR19	IEPCLR18	IEPCLR17	IEPCLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPCLR15	IEPCLR14	IEPCLR13	IEPCLR12	IEPCLR11	IEPCLR10	IEPCLR9	IEPCLR8	IEPCLR7	IEPCLR6	IEPCLR5	IEPCLR4	IEPCLR3	IEPCLR2	IEPCLR1	IEPCLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-4. ESM Influence $\overline{\text{ERROR}}$ Pin Clear/Status Register 1 (ESMIEPCR1)

Bit	Name	Value	Description
31-0	IEPCLR		Clear Influence on $\overline{\text{ERROR}}$ Pin
		0	**User and privileged mode:** Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMIEPSR1 register unchanged.
		1	**User and privileged mode:** Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. **Privileged mode:** Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR1 register.

25.5.3 ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

The base address for this register is 0xFFFFF508.

Figure 25-8. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTEN SET31	INTEN SET30	INTEN SET29	INTEN SET28	INTEN SET27	INTEN SET26	INTEN SET25	INTEN SET24	INTEN SET23	INTEN SET22	INTEN SET21	INTEN SET20	INTEN SET19	INTEN SET18	INTEN SET17	INTEN SET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN SET15	INTEN SET14	INTEN SET13	INTEN SET12	INTEN SET11	INTEN SET10	INTEN SET9	INTEN SET8	INTEN SET7	INTEN SET6	INTEN SET5	INTEN SET4	INTEN SET3	INTEN SET2	INTEN SET1	INTEN SET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-5. ESM Interrupt Enable Set/Status Register 1 (ESMIESR1)

Bit	Name	Value	Description
31-0	INTENSET		Set interrupt Enable
		0	**User and privileged mode:** Read: Interrupt is disabled. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMIECR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt is enabled. **Privileged mode:** Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR1 register.

25.5.4 ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

The base address for this register is 0xFFFFF50C.

Figure 25-9. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTEN CLR31	INTEN CLR30	INTEN CLR29	INTEN CLR28	INTEN CLR27	INTEN CLR26	INTEN CLR25	INTEN CLR24	INTEN CLR23	INTEN CLR22	INTEN CLR21	INTEN CLR20	INTEN CLR19	INTEN CLR18	INTEN CLR17	INTEN CLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN CLR15	INTEN CLR14	INTEN CLR13	INTEN CLR12	INTEN CLR11	INTEN CLR10	INTEN CLR9	INTEN CLR8	INTEN CLR7	INTEN CLR6	INTEN CLR5	INTEN CLR4	INTEN CLR3	INTEN CLR2	INTEN CLR1	INTEN CLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-6. ESM Interrupt Enable Clear/Status Register 1 (ESMIECR1)

Bit	Name	Value	Description
31-0	INTENCLR		Clear interrupt Enable
		0	**User and privileged mode:** Read: Interrupt is disabled. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMIESR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt is enabled. **Privileged mode:** Write: Disables interrupt and clears the corresponding set bit in the ESMIESR1 register.

25.5.5 ESM Interrupt Level Set/Status Register 1 (ESMILSR1)

The base address for this register is 0xFFFFF510.

Figure 25-10. ESM Interrupt Level Set/Status Register *1 (ESMILSR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTLVL SET31	INTLVL SET30	INTLVL SET29	INTLVL SET28	INTLVL SET27	INTLVL SET26	INTLVL SET25	INTLVL SET24	INTLVL SET23	INTLVL SET22	INTLVL SET21	INTLVL SET20	INTLVL SET19	INTLVL SET18	INTLVL SET17	INTLVL SET16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTVLV SET15	INTVLV SET14	INTVLV SET13	INTVLV SET12	INTVLV SET11	INTVLV SET10	INTVLV SET9	INTVLV SET8	INTVLV SET7	INTVLV SET6	INTVLV SET5	INTVLV SET4	INTVLV SET3	INTVLV SET2	INTVLV SET1	INTVLV SET0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-7. ESM Interrupt Level Set/Status Register 1 (ESMILSR1)

Bit	Name	Value	Description
31-0	INTLVLSET		Set Interrupt Level
		0	**User and privileged mode:** Read: Interrupt of channel x is mapped to low level interrupt line. **Privileged mode:** Write: Leaves the bit and the corresponding clear bit in the ESMILCR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt of channel x is mapped to high level interrupt line. **Privileged mode:** Write: Maps interrupt of channel x to high level interrupt line and sets the corresponding clear bit in the ESMILCR1 register.

25.5.6 ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)

The base address for this register is 0xFFFFF514.

Figure 25-11. ESM Interrupt Level Clear/Status Register *1 (ESMILCR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INTLVL CLR31	INTLVL CLR30	INTLVL CLR29	INTLVL CLR28	INTLVL CLR27	INTLVL CLR26	INTLVL CLR25	INTLVL CLR24	INTLVL CLR23	INTLVL CLR22	INTLVL CLR21	INTLVL CLR20	INTLVL CLR19	INTLVL CLR18	INTLVL CLR17	INTLVL CLR16
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTVLV CLR15	INTVLV CLR14	INTVLV CLR13	INTVLV CLR12	INTVLV CLR11	INTVLV CLR10	INTVLV CLR9	INTVLV CLR8	INTVLV CLR7	INTVLV CLR6	INTVLV CLR5	INTVLV CLR4	INTVLV CLR3	INTVLV CLR2	INTVLV CLR1	INTVLV CLR0
RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-8. ESM Interrupt Level Clear/Status Register 1 (ESMILCR1)

Bit	Name	Value	Description
31-0	INTLVLCLR		Clear Interrupt Level
		0	**User and privileged mode:** Read: Interrupt of channel x is mapped to low level interrupt line. **Privileged mode:** Write: Leaves the bit and the corresponding set bit in the ESMILSR1 register unchanged.
		1	**User and privileged mode:** Read: Interrupt of channel x is mapped to high level interrupt line. **Privileged mode:** Write: Maps interrupt of channel x to low level interrupt line and clears the corresponding set bit in the ESMILSR1 register.

25.5.7 ESM Status Register 1 (ESMSR1)

The base address for this register is 0xFFFFF518.

Figure 25-12. ESM Status Register 1 *(ESMSR1)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

R = Read, C = Clear, P = Privileged mode; -*n* = Value after reset/PORRST, *X* = Value unchanged

Table 25-9. ESM Status Register 1 (ESMSR1)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error
		0	**User and privileged mode:** Read: No error occurred; no interrupt is pending **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred; interrupt is pending. **Privileged mode:** Write: Clears the bit.
			Note: After nRST, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.

25.5.8 ESM Status Register 2 (ESMSR2)

The base address for this register is 0xFFFFF51C.

Figure 25-13. ESM Status Register 2 *(ESMSR2)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0	RCP-0

R = Read, C = Clear, P = Privileged mode; -*n* = Value after reset

Table 25-10. ESM Status Register 2 (ESMSR2)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error.
		0	**User and privileged mode:** Read: No error occurred; no interrupt is pending **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred; interrupt is pending. **Privileged mode:** 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.

25.5.9 ESM Status Register 3 (ESMSR3)

The base address for this register is 0xFFFFF520.

Figure 25-14. ESM Status Register 3 *(ESMSR3)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0	RCP-X/0

R = Read, C = Clear, P = Privileged mode; -*n* = Value after reset/PORRST, *X* = Value unchanged

Table 25-11. ESM Status Register 3 (ESMSR3)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Provides status information on a pending error
		0	**User and privileged mode:** Read: No error occurred. **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred. **Privileged mode:** Write: Clears the bit.

25.5.10 ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)

The base address for this register is 0xFFFFF524.

Figure 25-15. ESM $\overline{\text{ERROR}}$ Pin Status Register *(ESMEPSR)*

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
Reserved															EPSF
R-0															R-X/1

R = Read; *-n* = Value after reset/PORRST, *X* = Value unchanged

Table 25-12. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)

Bit	Name	Value	Description
31–1	Reserved	0	These bits are read as zero, and writes have no effect.
0	EPSF		$\overline{\text{ERROR}}$ Pin Status Flag Provides status information for the $\overline{\text{ERROR}}$ Pin
		0	**User and privileged mode:** Read: $\overline{\text{ERROR}}$ Pin is low (active) if any error has occurred. Write: Writes have no effect.
		1	**User and privileged mode:** 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 PORRST. The value will be unchanged after nRST. The $\overline{\text{ERROR}}$ pin status remains un-changed during after nRST.

25.5.11 ESM Interrupt Offset High Register (ESMIOFFHR)

The base address for this register is 0xFFFFF528.

Figure 25-16. ESM Interrupt Offset High Register *(ESMIOFFHR)*

31–16
Reserved
R-0

15–7	6–0
Reserved	INTOFFH[6:0]
R-0	R-0

R = Read; -*n* = Value after reset

Table 25-13. ESM Interrupt Offset High Register (ESMIOFFHR)

Bit	Name	Value	Description
31–7	Reserved	0	These bits are read as zero, and 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 group 2 have higher priority than interrupts of error group 1. 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.
		0000000	No pending interrupt.
		0000001	Interrupt pending for channel 0, error group1.
		...	
		0100000	Interrupt pending for channel 31, error group1.
		0100001	Interrupt pending for channel 0, error group2.
		...	
		1000000	Interrupt pending for channel 31, error group2.
			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.

25.5.12 ESM Interrupt Offset Low Register (ESMIOFFLR)

The base address for this register is 0xFFFFF52C.

Figure 25-17. ESM Interrupt Offset Low Register *(ESMIOFFLR)*

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="9">Reserved</td><td colspan="7">INTOFFL[6:0]</td></tr>
<tr><td colspan="9">R-0</td><td colspan="7">R-0</td></tr>
</table>

R = Read; -*n* = Value after reset

Table 25-14. ESM Interrupt Offset Low Register (ESMIOFFLR)

Bit	Name	Value	Description
31–7	Reserved	0	Reads return 0 and 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.
		0000000	No pending interrupt.
		0000001	Interrupt pending for channel 0, error group1.
		...	
		0100000	Interrupt pending for channel 31, 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.

25.5.13 ESM Low-Time Counter Register (ESMLTCR)

The base address for this register is 0xFFFFF530.

Figure 25-18. ESM Low-Time Counter Register *(ESMLTCR)*

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
LTC[15:0]															
R-0x3FFF															

R = Read; -*n* = Value after reset

Table 25-15. ESM Low-Time Counter Register (ESMLTCR)

Bit	Name	Value	Description
31–16	Reserved	0	These bits are read as zero, and writes have no effect.
15-0	LTC		$\overline{\text{ERROR}}$ Pin Low-Time Counter 16bit pre-loadable down-counter to control low-time of $\overline{\text{ERROR}}$ pin. The low-time counter is triggered by the peripheral clock (VCLK). **Note:** After reset the low time counter is set to the default preload value of the ESMLTCPR to ensure a minimum low time greater than 100us running at a maximum frequency of 100MHz. This register will be updated with the pre-load value of the ESMLTCPR once an error occurs and ESMEKR.EKEY is 5. For example: if current TLTC is 0x3FFF but LTCP is 0x7FFF, the $\overline{\text{ERROR}}$ pin will be clear as zero for 0x7FFF VCLK cycles once error occurs and EKEY is set to 5.

25.5.14 ESM Low-Time Counter Preload Register (ESMLTCPR)

The base address for this register is 0xFFFFF534.

Figure 25-19. ESM Low-Time Counter Preload Register *(ESMLTCPR)*

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
LTCP[15:14]		LTCP[13:0]													
RWP-0		R-3FFF													

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-16. ESM Low-Time Counter Preload Register (ESMLTCPR)

Bit	Name	Value	Description
31–16	Reserved	0	These bits are read as zero, and writes have no effect.
15-0	LTCP		$\overline{\text{ERROR}}$ Pin Low-Time Counter Pre-load Value 16bit pre-load value for the $\overline{\text{ERROR}}$ pin low-time counter. **Note:** Only LTCP.15 and LTCP.14 are configurable (privileged mode write) to ensure a minimum low time greater than 100us running at a maximum frequency of 100MHz.

25.5.15 ESM Error Key Register (ESMEKR)

The base address for this register is 0xFFFFF538.

Figure 25-20. ESM Error Key Register *(ESMEKR)*

31–16
Reserved
R-0

15–4	3–0
Reserved	EKEY[3:0]
R-0	RWP-0

R = Read, W = Write, P = Privileged mode; -*n* = Value after reset

Table 25-17. ESM Error Key Register (ESMEKR)

Bit	Name	Value	Description
31–4	Reserved	0	Reads return 0 and 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):
		0000	Activates normal mode (recommended default mode).
		1010	Forces error on $\overline{\text{ERROR}}$ pin.
		0101	The $\overline{\text{ERROR}}$ pin set to high when the low time counter (LTC) has completed; then the ESMEKR.EKEY bit will switch back to normal mode (EKEY = 0000)
		all other values	Activates normal mode.

25.5.16 ESM Status Shadow Register 2 (ESMSSR2)

The base address for this register is 0xFFFFF53C.

Figure 25-21. ESM Status Shadow Register 2 *(ESMSSR2)*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESF31	ESF30	ESF29	ESF28	ESF27	ESF26	ESF25	ESF24	ESF23	ESF22	ESF21	ESF20	ESF19	ESF18	ESF17	ESF16
RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESF15	ESF14	ESF13	ESF12	ESF11	ESF10	ESF9	ESF8	ESF7	ESF6	ESF5	ESF4	ESF3	ESF2	ESF1	ESF0
RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0	RCP-X/ 0

R = Read, C = Clear, P = Privileged mode; -*n* = Value after reset/PORRST, *X* = Value unchanged

Table 25-18. ESM Status Shadow Register 2 (ESMSSR2)

Bit	Name	Value	Description
31-0	ESF		Error Status Flag Shadow register for status information on pending error.
		0	**User and privileged mode:** Read: No error occurred. **Privileged mode:** Write: Leaves the bit unchanged.
		1	**User and privileged mode:** Read: Error occurred. **Privileged mode:** 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 (PORRST).

TEXAS INSTRUMENTS

Chapter 26

SPNU489C-February 2012

Data Modification Module (DMM)

This document describes the functionality of the Data Modification Module (DMM), which provides the capability to modify data in the entire 4 GB address space of the TMS570 devices from an external peripheral, with minimal interruption of the application.

Topic **Page**

26.1 Overview

26.1.1 Features

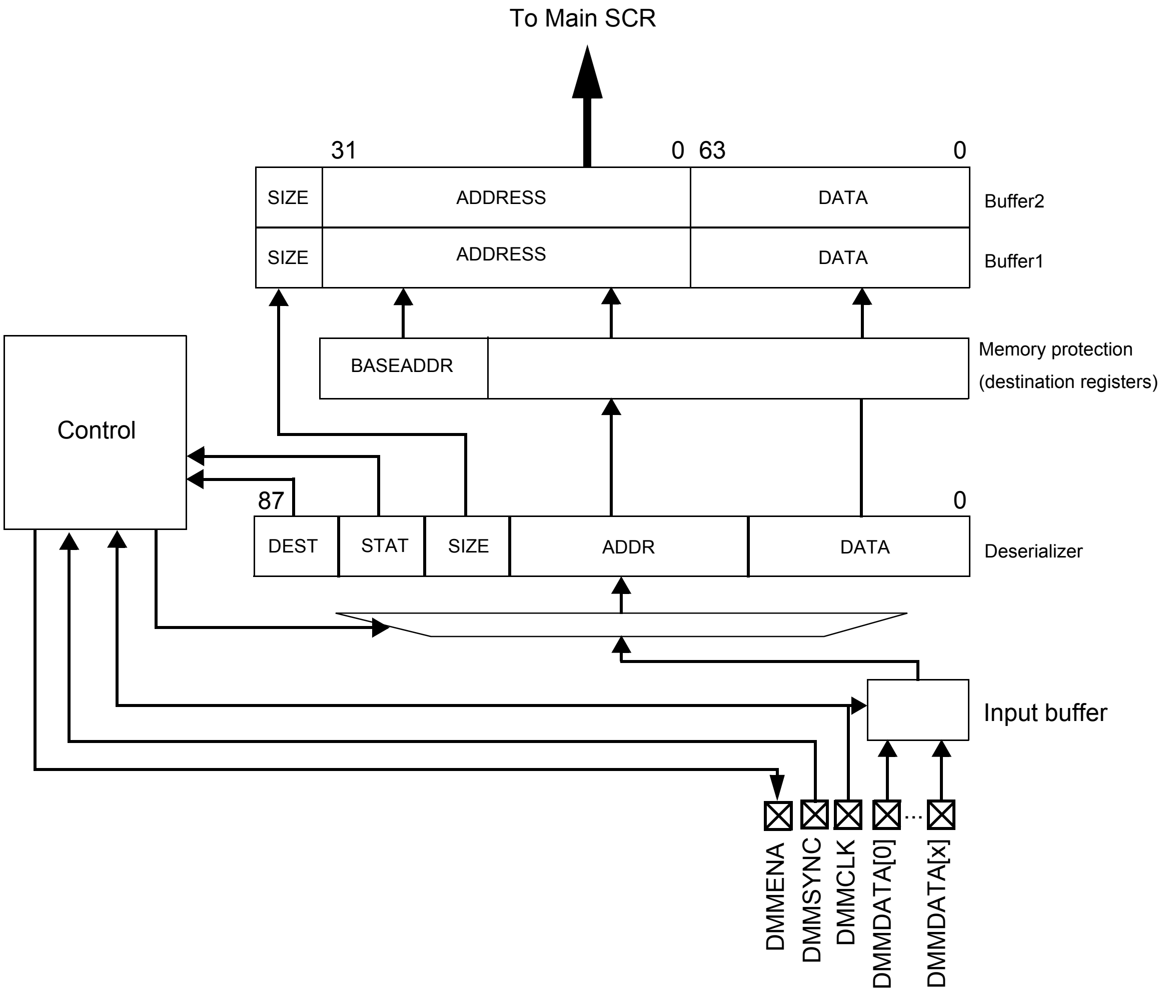
The DMM module has the following features:

- Acts as a bus master, thus enabling direct writes to the 4GB address space without CPU intervention
- Writes to memory locations specified in the received packet (leverages packets defined by trace mode of the RAM trace port (RTP) module
- Writes received data to consecutive addresses, which are specified by the DMM module (leverages packets defined by direct data mode of RTP module)
- Configurable port width (1, 2, 4, 8, 16 pins)
- Up to 100 Mbit/s pin data rate
- Unused pins configurable as GIO pins

26.2 Block Diagram

Figure 26-1 shows the block diagram for the DMM.

Figure 26-1. DMM Block Diagram



26.3 Module Operation

The DMM receives data over the DMM pins from external systems and writes the received data directly to the base address programmed in the module plus offset address given in the packet or into a buffer specified by start address and length. It leverages the protocol defined by the RAM Trace Port (RTP) module to have a common interface definition for external systems. It can also be used to connect an RTP and DMM module together for fast processor intercommunication.

The DMM module provides two modes of operation:

- **Trace Mode:** In this mode, the DMM writes the received data directly to an address which is calculated from the base address programmed into the destination register (Section 26.4.12; Section 26.4.14) plus the offset address contained in the received packet. An interrupt can be generated when data is written the lowest address of a programmed region. This capability enables the sender to raise an interrupt at the receiver while sending specific information.
- **Direct Data Mode:** In this mode, the DMM writes the received data into an address range of the 4GB address space. The buffer start address (Section 26.4.8) and blocksize (Section 26.4.9) is programmable in the DMM module. When the buffer reaches its end address, the buffer pointer wraps around and points to the beginning of the buffer again. The EO_BUFF flag (Section 26.4.5) will be set and if enabled, an interrupt will be generated to indicate a buffer-full condition. Another interrupt, can be configured to indicate different buffer fill levels. This can be accomplished by programming a certain fill level into the DMMINTPT (Section 26.4.11) register. The PROG_BUFF flag (Section 26.4.5) indicates that this level has been reached.

Data will be captured by the input buffer and moved to the appropriate bit field in the deseralizer. When the deseralizer is completely full, the data will be moved to the output buffer register. A two-level buffer is implemented to avoid overflow conditions if the internal bus is occupied by other transactions. In addition the DMMENA signal can be used to signal the external hardware that an overflow might occur if more data is sent. The automatic generation of the DMMENA signal can be configured by setting the ENAFUNC bit (Section 26.4.16). While the DMMENA signal is active, the DMM module will not receive any new data.

The DMM is a bus master and forwards the received data to the bus system. The write operation will be minimally intrusive to the program flow, because the CPU/DMA access will only be blocked if the CPU/DMA accesses the same resource as the DMM.

To prevent an external system from overwriting critical data in the memory while configured in Trace Mode, a memory protection mechanism is implemented via a programmable start address and block size of a region. A maximum of four destinations with two regions each are supported.

For proper operation, at least DMMCLK, DMMSYNC and DMMDATA[0] need to be programmed in functional mode (Section 26.4.16). If a large amount of data should be transmitted in a short time, more data pins should be used in functional mode. The module supports 1, 2, 4, 8, or 16-pin configurations.

The module can be configured to handle a free running clock provided on DMMCLK (Section 26.4.1). Clock pulses between two DMMSYNC pulses which exceed the number of valid clock pulses for a packet will be ignored.

26.3.1 Data Format

Below is a description of the packet and frame format.

26.3.1.1 Clocking Scheme

The DMM supports both continuous and noncontinuous clocking. The clock received on DMMCLK in the continuous clocking scheme is a free-running clock. In noncontinuous clocking scheme, the clock will stop after each packet and will start with the reception of a DMMSYNC signal.

26.3.1.2 Trace Mode Packet

Figure 26-2 illustrates the trace mode packet format. One packet consists of 2 bits (DEST) denoting the destination in which the data is stored, 2 status bits (STAT), the 2-bit SIZE of the data, the 18-bit address of where the data should be written to, and a variable data field.

The DEST bits (Table 26-1) will be used to determine which destination register applies to the transmitted data and the received address determines if the packet falls into a valid region of the destination area. If the address is valid, the base address, programmed in one of the destination registers (DMMDESTxREGy; Section 26.4.12; Section 26.4.14) of this particular region will be applied to create the complete 32-bit address for the destination. The DMM module only takes action on a ‘11’ setting of the STAT bits (Table 26-2). This signals that an overflow in the transmitting hardware module has occurred. If this is the case the SRC_OVF flag (Section 26.4.5) will be set and the received data will be written to the address specified in the packet. The size information of the data transmitted in the packet is denoted in the SIZE bits (Table 26-3) of the packet. Depending on the SIZE information, the module expects to receive only this amount of data.

Figure 26-2. Trace Mode Packet Format

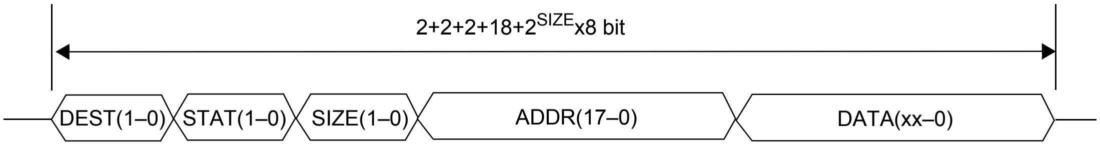


Table 26-1 through Table 26-3 illustrate the encoding of packet format in trace mode.

Table 26-1. Encoding of Destination Bits in Trace Mode Packet Format

DEST[1:0]	Destination
00	Dest 0
01	Dest 1
10	Dest 2
11	Dest 3

Table 26-2. Encoding of Status Bits in Trace Mode Packet Format

STAT[1:0]	Status
00	don’t care
01	don’t care
10	don’t care
11	overflow

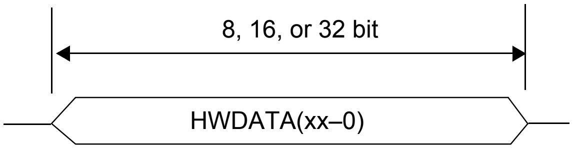
Table 26-3. Encoding of Write Size in Packet Format

SIZE[1:0]	Write Size
00	8 bit
01	16 bit
10	32 bit
11	64 bit

26.3.1.3 Direct Data Mode Packet

Figure 26-3 illustrates the direct data mode packet format.

Figure 26-3. Direct Data Mode Packet Format



The packet consists only of data bits and no header information. It can be 8-, 16- or 32-bit wide. A variable packet width is not supported because the DMM module will check the number of incoming bits (DMMCLK cycles) for error detection. The DMM will write the received data to the destination once the programmed number of bits has been received.

If the programmed word width does not correspond to the received data, the following actions will be taken.

- If the received data is greater than the programmed width, only the configured number of bits are transferred into the RAM buffer, the additional bits are discarded. A packet error will be flagged in non-continuous clock mode immediately upon receiving too many bits. In continuous clock mode, too many received bits cannot be flagged.
- If the received number of bits is smaller than the programmed width, no data will be written to the buffer, because a new DMMSYNC signal has been received before the expected number of bits. In both non-continuous and continuous clock mode, the packet error will be generated.

26.3.2 Data Port

The packet will be received in several subpackets, depending on the width of the external data bus (DMMDATA[y:0]) and the amount of data to be transmitted. Table 26-4 illustrates the number of clock cycles required for a complete packet.

Table 26-4. Number of Clock Cycles per Packet

Port Width/ Pins	Write Size in bits			
	8	16	32	64
1	32	40	56	88
2	16	20	28	44
4	8	10	14	22
8	4	5	7	11
16	2	3	4	6

The user can program the port width in the DMMPC0 register (Section 26.4.16). This feature allows pins that are not used for DMM functionality to be used as GIO pins. Only the pins shown in Table 26-5 can be used for a desired port width.

Table 26-5. Pins Used for Data Communication

Port Width	Pins Used
1	DMMDATA[0]
2	DMMDATA[1:0]

4	DMMDATA[3:0]
8	DMMDATA[7:0]
16	DMMDATA[15:0]

Note:
If pins other than the ones specified in Table 26-5 are programmed as functional pins for a desired port width, the received data will be corrupted and will not be transferred to the deserializer.

Note:
If DMMCLK or DMMSYNC are programmed as nonfunctional pins, functional operation will not occur.

26.3.2.1 Signal Description

DMMSYNC
This signal has to be provided by external hardware. It signals the start of a new packet. It has to be active (high) for one full DMMCLK cycle, starting with the rising edge of DMMCLK.
If the DMMSYNC pulse is longer than a single DMMCLK cycle and two falling edges of DMMCLK see a high pulse on DMMSYNC, the module will treat the second DMMSYNC pulse as the start of a packet and will flag a PACKET_ERR_INT (Section 26.4.5).

DMMCLK
The clock is externally generated and can be suspended between two packets. For this feature, CONTCLK must be set to 0 (Section 26.4.1). If the clock is not stopped between two packets, CONTCLK must be set to 1. Data will be latched on the falling edge of the DMMCLK signal.

$\overline{\text{DMMENA}}$
This signal is pulled high if no new data should be received via the data pins, because of a potential overflow situation.

DMMDATA[15:0]
These pins receive the packet information transmitted by the external hardware. Data is latched on the falling edge of DMMCLK.

Figure 26-4 shows an example of multiple packets received during trace mode, in noncontinuous clock configuration.

Figure 26-4. Packet Sync Signal example

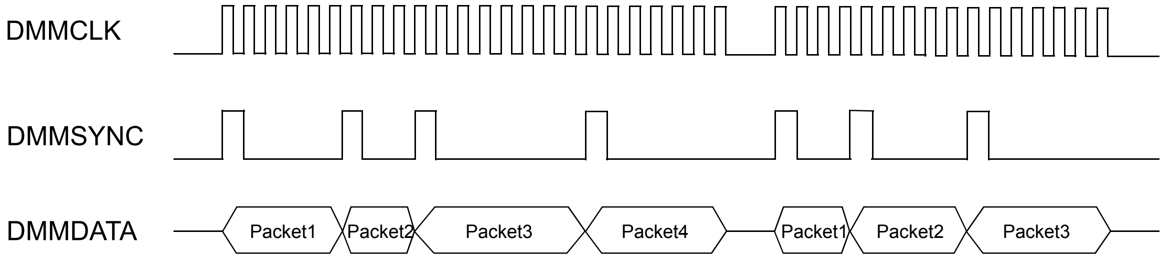
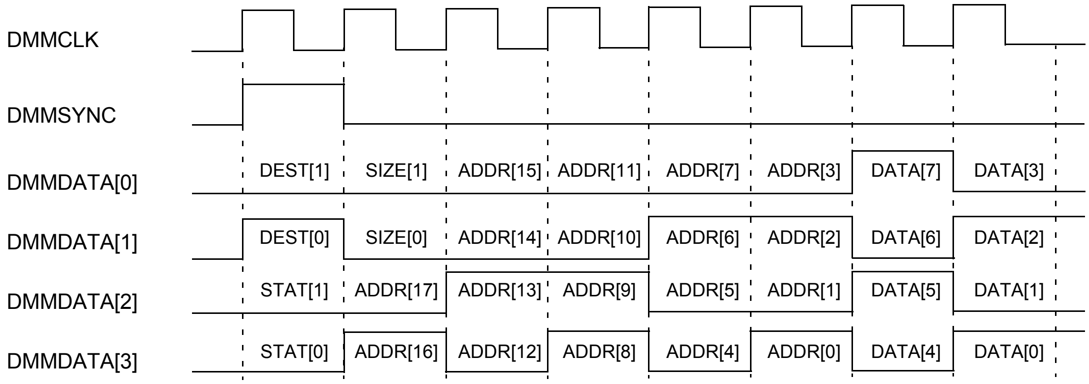


Figure 26-5 shows an example of a 4-bit data port with 8-bit receive data (0xA5) to be written into DEST1 (address 0x12345) on a trace mode packet.

Figure 26-5. Example Single Packet Transmission



26.3.3 Error Handling

The module will generate two different kind of errors. Once an error condition is recognized, an interrupt will be generated if enabled.

26.3.3.1 Overflow Error

This error is signaled when the module has received new data before the previous data was written to the destination address. If the internal buffers are full, the DMMENA signal will go high. If the sending module does not evaluate the DMMENA signal and keeps on sending new frames, the data that was previously received might be overwritten, thus resulting in setting the BUFF_OVF flag (Section 26.4.5).

26.3.3.2 Packet Error

Noncontinuous Clock Mode

The size of the incoming packet is defined by the SIZE information of a trace mode packet or the programmed size of a direct data mode packet. If too many or less than the number of bits are received before the next sync signal, the PACKET_ERR_INT flag will be set (Section 26.4.5). In case of receiving a DMMCLK signal without a corresponding DMMSYNC signal, a packet error will also be generated.

For too few received bits the packet error will be generated with the DMMSYNC signal which cuts the frame short. For too many received bits, the packet error will be generated immediately after receiving the additional bits without corresponding DMMSYNC signal.

Continuous Clock Mode

If less than the expected number of bits are received, the PACKET_ERR_INT flag will be set (Section 26.4.5) when the next DMMSYNC signal is received. Packets with more than the expected number of bits cannot be detected.

The check for packet error is done only after the detection of the first DMMSYNC signal after the DMM is turned on or comes out of suspend mode (with COS = 0; Section 26.4.1) (i.e., before the reception of first DMMSYNC, the toggling of DMMCLK would be ignored).

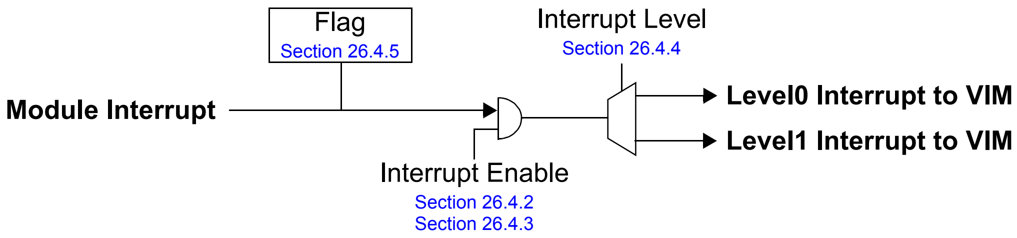
26.3.3.3 Bus Error

If an error occurs on the microcontroller internal bus system while transferring the data from the DMM to the destination, the BUSERROR flag will be set.

26.3.4 Interrupts

The module provides different interrupts. These can be programmed to different interrupt levels independently using DMMINTLVL (Section 26.4.4).

Figure 26-6. Interrupt Structure



Interrupts can be divided into error interrupts and functional interrupts. The error handling is described in Section 26.3.3. Functional interrupts depend on the mode (Trace Mode, Direct Data Mode) the DMM module is used in.

Trace Mode: An interrupt can be enabled whenever an access to the lowest address of a defined region is performed. This address is the starting address programmed in the DMMDESTxREGy register. An interrupt for each of the region can be generated by setting the individual interrupt enable bits.

Direct Data Mode: There are two interrupts which can be individually controlled. One is generated when the buffer pointer reaches the end of the defined buffer and wraps around (EO_BUFF; Section 26.4.2). The other one is generated when the buffer pointer matches the programmed interrupt threshold (PROG_BUFF; Section 26.4.2). The buffer pointer points to the next address to be written, therefor there are (interrupt threshold - 1) values stored in the buffer. The interrupt threshold can be programmed in the DMMINTPT (Section 26.4.11) register.

26.4 Control Registers Summary

This section describes the DMM registers. The registers support 8, 16, and 32-bit writes. The offset is relative to the associated peripheral select. Figure 26-7 provides a summary of the registers and their bits. The base address of the DMM module is 0xFFFFF700.

Figure 26-7. DMM Registers Summary

Offset Address[1]	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
0x00 DMMGLBCTRL	Reserved							BUSY	Reserved					CONT-CLK	COS	RESET
	Reserved					DDM_WIDTH		TM_DDM	Reserved				ON/OFF			
0x04 DMMINTSET	Reserved														PROG_BUFF	EO_BUFF
	DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
0x08 DMMINTCLR	Reserved														PROG_BUFF	EO_BUFF
	DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
0x0C DMMINTLVL	Reserved														PROG_BUFF	EO_BUFF
	DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
0x10 DMMINTFLG	Reserved														PROG_BUFF	EO_BUFF
	DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
0x14 DMMOFF1	Reserved															
	Reserved											OFFSET				
0x18 DMMOFF2	Reserved															
	Reserved											OFFSET				

[1] The offset address is relative to the peripheral's beginning address.

Figure 26-7. DMM Registers Summary (Continued)

Offset Address[1]	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
0x1C DMMDDMDEST	STARTADDR[31:16]															
	STARTADDR[15:0]															
0x20 DMMDDMBL	Reserved															
	Reserved												BLOCKSIZE			
0x24 DMMDDMPT	Reserved															
	Reserved	POINTER[14:0]														
0x28 DMMINTPT	Reserved															
	Reserved	INTPT[14:0]														
0x2C DMMDEST0REG1	BASEADDR[31:18]														BLOCKADDR[17:16]	
	BLOCKADDR[15:0]															
0x30 DMMDEST0BL1	Reserved															
	Reserved												BLOCKSIZE			
0x34 DMMDEST0REG2	BASEADDR[31:18]														BLOCKADDR[17:16]	
	BLOCKADDR[15:0]															
0x38 DMMDEST0BL2	Reserved															
	Reserved												BLOCKSIZE			

[1] The offset address is relative to the peripheral's beginning address.

Figure 26-7. DMM Registers Summary (Continued)

Offset Address[1]	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C DMMDEST1REG1	BASEADDR[31:18]														BLOCK-ADDR[17:16]	
	BLOCKADDR[15:0]															
0x40 DMMDEST1BL1	Reserved															
	Reserved												BLOCKSIZE			
0x44 DMMDEST1REG2	BASEADDR[31:18]														BLOCK-ADDR[17:16]	
	BLOCKADDR[15:0]															
0x48 DMMDEST1BL2	Reserved															
	Reserved												BLOCKSIZE			
0x4C DMMDEST2REG1	BASEADDR[31:18]														BLOCK-ADDR[17:16]	
	BLOCKADDR[15:0]															
0x50 DMMDEST2BL1	Reserved															
	Reserved												BLOCKSIZE			
0x54 DMMDEST2REG2	BASEADDR[31:18]														BLOCK-ADDR[17:16]	
	BLOCKADDR[15:0]															
0x58 DMMDEST2BL2	Reserved															
	Reserved												BLOCKSIZE			

[1] The offset address is relative to the peripheral's beginning address.

Figure 26-7. DMM Registers Summary (Continued)

Offset Address[1]	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
0x5C DMMDEST3REG1	BASEADDR[31:18]														BLOCKADDR[17:16]	
	BLOCKADDR[15:0]															
0x60 DMMDEST3BL1	Reserved															
	Reserved												BLOCKSIZE			
0x64 DMMDEST3REG2	BASEADDR[31:18]														BLOCKADDR[17:16]	
	BLOCKADDR[15:0]															
0x68 DMMDEST3BL2	Reserved															
	Reserved												BLOCKSIZE			
0x6C DMMPC0	Reserved													ENAFUNC	DATA15FUNC	DATA14FUNC
	DATA13FUNC	DATA12FUNC	DATA11FUNC	DATA10FUNC	DATA9FUNC	DATA8FUNC	DATA7FUNC	DATA6FUNC	DATA5FUNC	DATA4FUNC	DATA3FUNC	DATA2FUNC	DATA1FUNC	DATA0FUNC	CLKFUNC	SYNCFUNC
0x70 DMMPC1	Reserved													ENADIR	DATA15DIR	DATA14DIR
	DATA13DIR	DATA12DIR	DATA11DIR	DATA10DIR	DATA9DIR	DATA8DIR	DATA7DIR	DATA6DIR	DATA5DIR	DATA4DIR	DATA3DIR	DATA2DIR	DATA1DIR	DATA0DIR	CLKDIR	SYNCDIR
0x74 DMMPC2	Reserved													ENAIN	DATA15IN	DATA14IN
	DATA13IN	DATA12IN	DATA11IN	DATA10IN	DATA9IN	DATA8IN	DATA7IN	DATA6IN	DATA5IN	DATA4IN	DATA3IN	DATA2IN	DATA1IN	DATA0IN	CLKIN	SYNCIN
0x78 DMMPC3	Reserved													ENAOUT	DATA15OUT	DATA14OUT
	DATA13OUT	DATA12OUT	DATA11OUT	DATA10OUT	DATA9OUT	DATA8OUT	DATA7OUT	DATA6OUT	DATA5OUT	DATA4OUT	DATA3OUT	DATA2OUT	DATA1OUT	DATA0OUT	CLKOUT	SYNCOUT

[1] The offset address is relative to the peripheral's beginning address.

Figure 26-7. DMM Registers Summary (Continued)

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Address[1]	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7C DMMPC4	Reserved													ENASET	DATA15SET	DATA14SET
	DATA13SET	DATA12SET	DATA11SET	DATA10SET	DATA9SET	DATA8SET	DATA7SET	DATA6SET	DATA5SET	DATA4SET	DATA3SET	DATA2SET	DATA1SET	DATA0SET	CLKSET	SYNCSET
0x80 DMMPC5	Reserved													ENACLR	DATA15CLR	DATA14CLR
	DATA13CLR	DATA12CLR	DATA11CLR	DATA10CLR	DATA9CLR	DATA8CLR	DATA7CLR	DATA6CLR	DATA5CLR	DATA4CLR	DATA3CLR	DATA2CLR	DATA1CLR	DATA0CLR	CLKCLR	SYNCCLR
0x84 DMMPC6	Reserved													ENAPDR	DATA15PDR	DATA14PDR
	DATA13PDR	DATA12PDR	DATA11PDR	DATA10PDR	DATA9PDR	DATA8PDR	DATA7PDR	DATA6PDR	DATA5PDR	DATA4PDR	DATA3PDR	DATA2PDR	DATA1PDR	DATA0PDR	CLKPDR	SYNCPDR
0x88 DMMPC7	Reserved													ENAPDIS	DATA15PDIS	DATA14PDIS
	DATA13PDIS	DATA12PDIS	DATA11PDIS	DATA10PDIS	DATA9PDIS	DATA8PDIS	DATA7PDIS	DATA6PDIS	DATA5PDIS	DATA4PDIS	DATA3PDIS	DATA2PDIS	DATA1PDIS	DATA0PDIS	CLKPDIS	SYNCPDIS
0x8C DMMPC8	Reserved													ENAPSEL	DATA15PSEL	DATA14PSEL
	DATA13PSEL	DATA12PSEL	DATA11PSEL	DATA10PSEL	DATA9PSEL	DATA8PSEL	DATA7PSEL	DATA6PSEL	DATA5PSEL	DATA4PSEL	DATA3PSEL	DATA2PSEL	DATA1PSEL	DATA0PSEL	CLKPSEL	SYNCPSEL

[1] The offset address is relative to the peripheral’s beginning address.

26.4.1 DMM Global Control Register (DMMGLBCTRL)

With this register the basic operation of the module is selected.

Figure 26-8. DMM Global Control Register (DMMGLBCTRL) [offset = 00h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							BUSY	Reserved					CONTCLK	COS	RESET
R-0							R-0	R-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
Reserved					DDM_WIDTH(1–0)		TM_DDM	Reserved				ON/OFF			
R-0					R/WP-0		R/WP-0	R-0				R/WP-0101b			

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-6. DMM Global Control Register (DMMGLBCTRL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return 0 and writes have no effect
24	BUSY		Busy indicator.
		0	The DMM does not currently receive data and has no data in its internal buffers, which needs to be transferred.
		1	The module is currently receiving data, or has data in its internal buffers
23-19	Reserved		Reads return 0 and writes have no effect
18	CONTCLK		Continuous DMMCLK input.
			User and privilege mode read, privilege mode write:
		0	DMMCLK is expected to be suspended between two packets.
		1	DMMCLK is expected to be free running between packets.

Table 26-6. DMM Global Control Register (DMMGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
17	COS		Continue on suspend. Influences behavior of module while in debug mode. In all cases the corresponding interrupt will be set.
			User and privilege mode (read):
		0	Packets will not be received during debug mode. Before entering debug mode, the ongoing reception of a packet will be finished and the value will be written to the destination.
		1	Continue receiving packets and update destination, while in debug mode
			Privilege mode (write):
		0	Disable data reception while in debug mode
		1	Enable data reception while in debug mode
16	RESET		Reset. This bit resets the state machine and the registers to its reset value, except the RESET bit itself. It must be cleared by writing to it.
			User and privilege mode (read):
		0	No reset of DMM module
		1	Reset of DMM module
			Privilege mode (write):
		0	No reset of DMM module
		1	Reset DMM module to its reset state
15-11	Reserved		Reads return 0 and writes have no effect
10-9	DDM_WIDTH		Packet Width in direct data mode
			User and privilege mode read and privilege mode write operation:
		Bit Encoding	Transfer Size
		00	8 bit
		01	16 bit
		10	32 bit
		11	Reserved

Table 26-6. DMM Global Control Register (DMMGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
8	TM_DMM		Packet Format.
			User and privilege mode (read):
		0	The DMM module assumes packets in trace mode definition
		1	The DMM module assumes packets in direct data mode definition
			Privilege mode (write):
		0	Enable trace mode
		1	Enable direct data mode
7-4	Reserved		Reads return 0 and writes have no effect
3-0	ON/OFF		Switch module on or off
			User and privilege mode (read):
		All other	The DMM module does not receive data
		1010	The DMM module receives data and writes it to the buffer
			Privilege mode (write):
		All other	Disable receive/write operations. Packets in reception, will still be finished
		1010	Enable receive/write operations. Packets will be received 1 HCLK cycle after enabling the module

Note:
It is recommended to write 0101 to ON/OFF to avoid having a soft error inadvertently enabling the module when a singel bit flips.

Note:
Registers which affect the operation of the module, should be only programmed when the BUSY bit is 0 and the ON/OFF bits are not 1010.

Note:
If the module was in operation, turned off (ON/OFF = all other than 1010) and then turned on (ON/OFF = 1010) again, it is recommended to perform a reset (RESET = 1) of the module before switching it on. This avoids that the state machine is held in an unrecoverable state.

Note:
A write to these register bits while receiving a packet will not have any effect on the received packet. The mode change will be performed after the packet is received.

26.4.2 DMM Interrupt Set Register (DMMINTSET)

This register contains the interrupt set bits for error interrupts and functional interrupts. Only the bits which are relevant for the particular mode (trace mode or direct data mode) will be taken into account for the interrupt generation.

Figure 26-9. DMM Interrupt Set Register (DMMINTSET) [offset = 04h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/C/P-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect
17	PROG_BUFF		Programmable Buffer Interrupt Set. This enables the interrupt generation in case the buffer pointer equals the programmed value in the DMMINTPT register (Section 26.4.11). This bit is only relevant in Direct Data Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on pointer match.
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
16	EO_BUFF		End of Buffer Interrupt Set. This enables the interrupt generation in case data was written to the last entry in the buffer and the pointer wrapped around to the beginning of the buffer. This bit is only relevant in Direct Data Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on writing to the last entry.
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
15	DEST3REG2		Destination 3 Region 2 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
14	DEST3REG1		Destination 3 Region 1 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
13	DEST2REG2		Destination 2 Region 2 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 2 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
12	DEST2REG1		Destination 2 Region 1 Interrupt Set. This enables the interrupt generation in case data was accessed at the start address of Destination 2 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
11	DEST1REG2		Destination 1 Region 2 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
10	DEST1REG1		Destination 1 Region 1 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 1 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
9	DEST0REG2		Destination 0 Region 2 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
8	DEST0REG1		Destination 0 Region 1 Interrupt Set. This enables the interrupt generation in case data was accessed at the startaddress of Destination 0 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
7	BUSERROR		Bus Error Response for errors generated when doing internal bus transfers.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
6	BUFF_OVF		Buffer Overflow. This enables the interrupt generation in case new data is received, while the previous data still has not been transmitted.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
5	SRC_OVF		Source Overflow. This enables an interrupt if the external system experienced and overflow which was signalled in the Trace Mode packet.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
4	DEST3_ERR		Destination 3 Error. This enables the interrupt generation in case data should be written into a address not specified by DMMDEST3REG1/DMMDEST3BL1 or DMMDEST3REG2/ DMMDEST3BL2. If both blocksizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
3	DEST2_ERR		Destination 2 Error Interrupt Set. This enables the interrupt generation in case data should be written into a address not specified by DMMDEST2REG1/DMMDEST2BL1 or DMMDEST2REG2/ DMMDEST2BL2. If both blocksizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
2	DEST1_ERR		Destination 1 Error Interrupt Set. This enables the interrupt generation in case data should be written into a address not specified by DMMDEST1REG1/DMMDEST1BL1 or DMMDEST1REG2/ DMMDEST1BL2. If both blocksizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit

Table 26-7. DMM Interrupt Set Register (DMMINTSET) Field Descriptions (Continued)

Bit	Name	Value	Description
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
1	DEST0_ERR		Destination 0 Error Interrupt Set. This enables the interrupt generation in case data should be written into a address not specified by DMMDEST0REG1/DMMDEST0BL1 or DMMDEST0REG2/ DMMDEST0BL2. If both blocksizes are programmed to 0 or a reserved value, the interrupt will still be generated, the write to the internal RAM however will not take place.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)
0	PACKET_ERR_INT		Packet Error. This enables the interrupt generation in case of an error condition in the packet reception. Please refer to Section 26.3.3 for the error conditions.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Enable interrupt (sets corresponding bit in DMMINTCLR; Section 26.4.4)

26.4.3 DMM Interrupt Clear Register (DMMINTCLR)

This register contains the interrupt clear bits for error interrupts and functional interrupts. Only the bits which are relevant for the particular mode (trace mode or direct data mode) will be taken into account for the interrupt generation

Figure 26-10. DMM Interrupt Clear Register (DMMINTCLR) [offset = 08h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/C/P-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect
17	PROG_BUFF		Programmable Buffer Interrupt Clear. This disables the interrupt generation in case the buffer pointer equals the programmed value in the DMMINTPT register (Section 26.4.11). This bit is only relevant in Direct Data Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
16	EO_BUFF		End of Buffer Interrupt. This disables the interrupt generation in case data was written to the last entry in the buffer and the pointer wrapped around to the beginning of the buffer. This bit is only relevant in Direct Data Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on writing to the last entry.
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
15	DEST3REG2		Destination 3 Region 2 Interrupt Clear. This disables the interrupt generation in case data was accessed at the startaddress of Destination 3 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
14	DEST3REG1		Destination 3 Region 1 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 3 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
13	DEST2REG2		Destination 2 Region 2 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 2 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
12	DEST2REG1		Destination 2 Region 1 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 2 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
11	DEST1REG2		Destination 1 Region 2 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 1 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
10	DEST1REG1		Destination 1 Region 1 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 1 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
9	DEST0REG2		Destination 0 Region 2 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 0 Region 2. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
8	DEST0REG1		Destination 0 Region 1 Interrupt Clear. This disables the interrupt generation in case data was accessed at the start address of Destination 0 Region 1. This bit is only relevant in Trace Mode.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated on a write to the start address of this region
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
7	BUSERROR		Bus Error Response for errors generated when doing internal bus transfers.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
6	BUFF_OVF		Buffer Overflow. This disables the interrupt generation in case new data is received, while the previous data still has not been transmitted.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
5	SRC_OVF		Source Overflow. This disables an interrupt if the external system experienced and overflow which was signalled in the Trace Mode packet.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
4	DEST3_ERR		Destination 3 Error. This disables the interrupt generation in case data should be written into a address not specified by DMMDEST3REG1/DMMDEST3BL1 or DMMDEST3REG2/ DMMDEST3BL2.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
3	DEST2_ERR		Destination 2 Error Interrupt. This disables the interrupt generation in case data should be written into a address not specified by DMMDEST2REG1/DMMDEST2BL1 or DMMDEST2REG2/ DMMDEST2BL2.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
2	DEST1_ERR		Destination 1 Error Interrupt. This disables the interrupt generation in case data should be written into a address not specified by DMMDEST1REG1/DMMDEST1BL1 or DMMDEST1REG2/ DMMDEST1BL2.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

Table 26-8. DMM Interrupt Clear Register (DMMINTCLR) Field Descriptions (Continued)

Bit	Name	Value	Description
1	DEST0_ERR		Destination 0 Error Interrupt. This disables the interrupt generation in case data should be written into a address not specified by DMMDEST0REG1/DMMDEST0BL1 or DMMDEST0REG2/ DMMDEST0BL2.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)
0	PACKET_ERR_INT		Packet Error. This disables the interrupt generation in case of an error condition in the packet reception. Please refer to Section 26.3.3 for the error conditions.
			User and privilege mode (read):
		0	No interrupt will be generated
		1	An interrupt will be generated
			Privilege mode (write):
		0	No influence on bit
		1	Disable interrupt (clears corresponding bit in DMMINTSET; Section 26.4.2)

26.4.4 DMM Interrupt Level Register (DMMINTLVL)

This register contains the interrupt level bits for error interrupts and normal interrupts.

Figure 26-11. DMM Interrupt Level Register (DMMINTLVL) [offset = 0Ch]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/C/P-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-9. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect
17	PROG_BUFF		Programmable Buffer Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
16	EO_BUFF		End of Buffer Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
15	DEST3REG2		Destination 3 Region 2 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
14	DEST3REG1		Destination 3 Region 1 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1

Table 26-9. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions (Continued)

Bit	Name	Value	Description
13	DEST2REG2		Destination 2 Region 2 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
12	DEST2REG1		Destination 2 Region 1 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
11	DEST1REG2		Destination 1 Region 2 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
10	DEST1REG1		Destination 1 Region 1 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
9	DEST0REG2		Destination 0 Region 2 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
8	DEST0REG1		Destination 0 Region 1 Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
7	BUSERROR		BMM Bus Error Response
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1

Table 26-9. DMM Interrupt Level Register (DMMINTLVL) Field Descriptions (Continued)

Bit	Name	Value	Description
6	BUFF_OVF		Write Buffer Overflow Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
5	SCR_OVF		Source Overflow Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
4	DEST3_ERR		Destination 3 Error Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
3	DEST2_ERR		Destination 2 Error Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
2	DEST1_ERR		Destination 1 Error Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
1	DEST0_ERR		Destination 0 Error Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1
0	PACKET_ERR_INT		Packet Error Interrupt Level
			User and privilege mode read, privilege mode write:
		0	Interrupt mapped to level 0
		1	Interrupt mapped to level 1

26.4.5 DMM Interrupt Flag Register (DMMINTFLG)

This register contains the interrupt level bits for error interrupts and normal interrupts.

Figure 26-12. DMM Interrupt Flag Register (DMMINTFLG) [offset = 10h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEST3REG2	DEST3REG1	DEST2REG2	DEST2REG1	DEST1REG2	DEST1REG1	DEST0REG2	DEST0REG1	BUSERROR	BUFF_OVF	SRC_OVF	DEST3_ERR	DEST2_ERR	DEST1_ERR	DEST0_ERR	PACKET_ERR_INT
R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C/P-0	R/C//P-0	R/C/P-0	R/C/P-0	R/C/P-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-10. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions

Bit	Name	Value	Description
31–18	Reserved		Reads return 0 and writes have no effect
17	PROG_BUFF		Programmable Buffer Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
16	EO_BUFF		End of Buffer Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared

Table 26-10. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
15	DEST3REG2		Destination 3 Region 2 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
14	DEST3REG1		Destination 3 Region 1 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
13	DEST2REG2		Destination 2 Region 2 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
12	DEST2REG1		Destination 2 Region 1 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared

Table 26-10. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
11	DEST1REG2		Destination 1 Region 2 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
10	DEST1REG1		Destination 1 Region 1 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
9	DEST0REG2		Destination 0 Region 2 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
8	DEST0REG1		Destination 0 Region 1 Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared

Table 26-10. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
7	BUSERROR		BMM Bus Error Response.
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
6	BUFF_OVF		Write Buffer Overflow Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
5	SRC_OVF		Source Overflow Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
4	DEST3_ERR		Destination 3 Error Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared

Table 26-10. DMM Interrupt Flag Register (DMMINTFLG) Field Descriptions (Continued)

Bit	Name	Value	Description
3	DEST2_ERR		Destination 2 Error Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
2	DEST1_ERR		Destination 1 Error Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
1	DEST0_ERR		Destination 0 Error Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared
0	PACKET_ERR_INT		Packet Error Interrupt Flag
			User and privilege mode (read):
		0	No interrupt occurred
		1	Interrupt occurred
			Privilege mode (write):
		0	No influence on bi
		1	Bit will be cleared

26.4.6 DMM Interrupt Offset 1 Register (DMMOFF1)

This register holds the offset indicating which interrupt occurred on interrupt level 0. The CPU can read this register to determine the source of the interrupt without having to test individual interrupt flags.

Figure 26-13. DMM Interrupt Offset 1 Register (DMMOFF1) [offset = 14h]

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
Reserved											OFFSET				
R-0											R-0				

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-11. DMM Interrupt Offset 1 Register (DMMOFF1) Field Descriptions

Bit	Name	Value	Description
31–5	Reserved		Reads return 0 and writes have no effect
4–0	OFFSET		
			User and privilege mode (read):
		Bit Encoding	Interrupt
		00000	Phantom. All interrupt flags have been cleared before the offset register has been read.
		00001	Packet Error
		00010	Destination 0 Error
		00011	Destination 1 Error
		00100	Destination 2 Error
		00101	Destination 3 Error
		00110	Source Overflow
		00111	Buffer Overflow
		01000	Bus Error
		01001	Destination 0 Region 1
		01010	Destination 0 Region 2
		01011	Destination 1 Region 1
		01100	Destination 1 Region 2

Table 26-11. DMM Interrupt Offset 1 Register (DMMOFF1) Field Descriptions (Continued)

Bit	Name	Value	Description
		01101	Destination 2 Region 1
		01110	Destination 2 Region 2
		01111	Destination 3 Region 1
		10000	Destination 3 Region 2
		10001	End of Buffer
		10010	Programmable Buffer
		10011-11111	Reserved
			Reading the offset will clear the corresponding flag in DMMINTFLG (Section 26.4.5).
			Privilege and user mode writes have no effect

26.4.7 DMM Interrupt Offset 2 Register (DMMOFF2)

This register holds the offset indicating which interrupt occurred on interrupt level 1. The CPU can read this register to determine the source of the interrupt without having to test individual interrupt flags.

Figure 26-14. DMM Interrupt Offset 2 Register (DMMOFF2) [offset = 18h]

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
Reserved											OFFSET				
R-0											R-0				

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-12. DMM Interrupt Offset 2 Register (DMMOFF2) Field Descriptions

Bit	Name	Value	Description
31-5	Reserved		Reads return 0 and writes have no effect
4–0	OFFSET		
			User and privilege mode (read):
		Bit Encoding	Interrupt
		00000	Phantom. All interrupt flags have been cleared before the offset register has been read.
		00001	Packet Error
		00010	Destination 0 Error
		00011	Destination 1 Error
		00100	Destination 2 Error
		00101	Destination 3 Error
		00110	Source Overflow
		00111	Buffer Overflow
		01000	Bus Error
		01001	Destination 0 Region 1
		01010	Destination 0 Region 2
		01011	Destination 1 Region 1
		01100	Destination 1 Region 2

Table 26-12. DMM Interrupt Offset 2 Register (DMMOFF2) Field Descriptions (Continued)

Bit	Name	Value	Description
		01101	Destination 2 Region 1
		01110	Destination 2 Region 2
		01111	Destination 3 Region 1
		10000	Destination 3 Region 2
		10001	End of Buffer
		10010	Programmable Buffer
		10011-11111	Reserved
			Reading the offset will clear the corresponding flag in DMMINTFLG (Section 26.4.5).
			Privilege and user mode writes have no effect

26.4.8 DMM Direct Data Mode Destination Register (DMMDDMDEST)

This register defines the starting address of the buffer used to store the received data in Direct Data Mode. By writing to this register, the DMMDDMPT register (Section 26.4.10) will be set to 0x0000.

Figure 26-15. DMM Direct Data Mode Destination Register (DMMDDMDEST) [offset = 1Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTADDR[31:16]															
R/WP-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDR[15:0]															
R/WP-0															

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-13. DMM Direct Data Mode Destination Register (DMMDDMDEST) Field Descriptions

Bit	Name	Value	Description
31–0	STARTADDR[31:0]		These bits define the starting address of the buffer. The starting address has to be a multiple of the blocksize chosen in DMMDDMBL (Section 26.4.9). User and privilege mode (read): current start address Privilege mode (write): sets start address to value written

26.4.9 DMM Direct Data Mode Blocksize Register (DMMDDMBL)

This register defines the blocksize of the buffer used to store the received data in Direct Data Mode.

Figure 26-16. DMM Direct Data Mode Blocksize Register (DMMDDMBL) [offset = 20h]

31–16
Reserved
R-0

15–4	3–0
Reserved	BLOCKSIZE
R-0	R/WP-0

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-14. DMM Direct Data Mode Blocksize Register (DMMDDMBL) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect
3–0	BLOCKSIZE		These bits define the size of the buffer region
			User and privilege mode (read): current block size
			Privilege mode (write):
		0000	Buffer disabled. No data will be stored.
		0001	32 Byte
		0010	64 Byte
		0011	128 Byte
		0100	256 Byte
		0101	512 Byte
		0110	1 KByte
		0111	2 KByte
		1000	4 KByte
		1001	8 KByte
		1010	16 KByte
		1011	32 KByte
		1100..1111	Reserved

26.4.10 DMM Direct Data Mode Pointer Register (DMMDDMPT)

This register shows the pointer into the buffer programmed by DMMDDMDEST (Section 26.4.8) and DMMDDMBL (Section 26.4.9).

Figure 26-17. DMM Direct Data Mode Pointer Register (DMMDDMPT) [offset = 24h]

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
Reserved	POINTER														
R-0	R-0														

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-15. DMM Direct Data Mode Pointer Register (DMMDDMPT) Field Descriptions

Bit	Name	Value	Description
31–15	Reserved		Reads return 0 and writes have no effect
15–0	POINTER		These bits hold the pointer to the next entry to be written in the buffer. The pointer points to the byte aligned address. If in 16-bit DDM mode, bit 0 will be 0. If in 32-bit DDM mode, bit 0 and 1 will be 0. User and privilege mode (read): next data entry Privilege mode (write): writes have no effect

26.4.11 DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT)

This register can be programmed to hold a threshold to which the DMMDDMPT register (Section 26.4.10) is compared. An interrupt can be generated when both match.

Figure 26-18. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) [offset = 28h]

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
Reserved	INTPT														
R-0	RWP-0														

R = Read, WP = Write in privilege mode only; -*n* = Value after reset

.

Table 26-16. DMM Direct Data Mode Interrupt Pointer Register (DMMINTPT) Field Descriptions

Bit	Name	Value	Description
31–15	Reserved		Reads return 0 and writes have no effect
14–0	INTPT		Interrupt Pointer. When the buffer pointer (DMMDDMPT; Section 26.4.10) matches the programmed value in DMMINTPT and the PROG_BUF interrupt (Section 26.4.2) is set, an interrupt is generated. User and privilege mode (read): current interrupt threshold Privilege mode (write): new interrupt threshold

26.4.12 DMM Destination x Region 1 (DMMDESTxREG1)

This register defines the starting address of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG1 and DMMDESTxBL1, an interrupt (DESTx_ERR) can be generated. The description below is valid for following registers: DMMDEST0REG1, DMMDEST1REG1, DMMDEST2REG1, DMMDEST3REG1.

Figure 26-19. DMM Destination x Region 1 (DMMDESTxREG1) [offset = 2Ch, 3Ch, 4Ch, 5Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:18]														BLOCK-ADDR[17:16]	
R/WP-0														R/WP-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLOCKADDR[15:0]															
R/WP-0															

R = Read, WP = Write in privilege mode only; *-n* = Value after reset

.

Table 26-17. DMM Destination x Region 1 (DMMDESTxREG1) Field Descriptions

Bit	Name	Value	Description
31-18	BASEADDR[31:18]		These bits define the base address of the 256kB region where the buffer is located. User and privilege mode (read): current start address Privilege mode (write): sets base address to value written
17–0	BLOCKADDR[17:0]		These bits define the starting address of the buffer in the 256kB page. The starting address has to be a multiple of the blocksize chosen in DMMDESTxBL1 (Section 26.4.13). User and privilege mode (read): current start address Privilege mode (write): sets start address to value written

26.4.13 DMM Destination x Blocksize 1 (DMMDESTxBL1)

This register defines the blocksize of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG1 and DMMDESTxBL1, an interrupt (DESTx_ERR) can be generated. The description below is valid for following registers: DMMDEST0BL1, DMMDEST1BL1, DMMDEST2BL1, DMMDEST3BL1.

Figure 26-20. DMM Destination x Blocksize 1 (DMMDESTxBL1) [offset = 30h, 40h, 50h, 60h]

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
Reserved												BLOCKSIZE			
R-0												R/WP-0			

R = Read, WP = Write in privilege mode only; *-n* = Value after reset

.

Table 26-18. DMM Destination x Blocksize 1 (DMMDESTxBL1) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect
3-0	BLOCKSIZE		These bits define the length of the buffer region. If all bits are 0, the region is disabled and no data will be stored. User and privilege mode (read): current block size Privilege mode (write):
		0000	Region disabled
		0001	1 KByte
		0010	2 KByte
		0011	4 KByte
		0100	8 KByte
		0101	16 KByte
		0110	32 KByte
		0111	64 KByte
		1000	128 KByte
		1001	256 KByte
		1010..1111	Reserved

26.4.14 DMM Destination x Region 2 (DMMDESTxREG2)

This register defines the starting address of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG2 and DMMDESTxBL2, an interrupt (DESTx_ERR) can be generated. The description below is valid for following registers: DMMDEST0REG2, DMMDEST1REG2, DMMDEST2REG2, DMMDEST3REG2.

Figure 26-21. DMM Destination x Region 2 (DMMDESTxREG2) [offset = 34h, 44h, 54h, 64h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:18]														BLOCK-ADDR[17:16]	
R/WP-0														R/WP-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLOCKADDR[15:0]															
R/WP-0															

R = Read, WP = Write in privilege mode only; *-n* = Value after reset

.

Table 26-19. DMM Destination x Region 2 (DMMDESTxREG2) Field Descriptions

Bit	Name	Value	Description
31-18	BASEADDR[31:18]		These bits define the base address of the 256kB region where the buffer is located. User and privilege mode (read): current start address Privilege mode (write): sets base address to value written
17–0	BLOCKADDR[17:0]		These bits define the starting address of the buffer in the 256kB page. The starting address has to be a multiple of the blocksize chosen in DMMDESTxBL2 (Section 26.4.15). User and privilege mode (read): current start address Privilege mode (write): sets start address to value written

26.4.15 DMM Destination x Blocksize 2 (DMMDESTxBL2)

This register defines the blocksize of the buffer used to store the received data in Trace Mode. If the received data does not fall into the address range defined by DMMDESTxREG2 and DMMDESTxBL2, an interrupt (DESTx_ERR) can be generated. The description below is valid for following registers: DMMDEST0BL2, DMMDEST1BL2, DMMDEST2BL2, DMMDEST3BL2.

Figure 26-22. DMM Destination x Blocksize 2 (DMMDESTxBL2) [offset = 38h, 48h, 58h, 68h]

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
Reserved												BLOCKSIZE			
R-0												R/WP-0			

R = Read, WP = Write in privilege mode only; *-n* = Value after reset

.

Table 26-20. DMM Destination x Blocksize 2 (DMMDESTxBL2) Field Descriptions

Bit	Name	Value	Description
31–4	Reserved		Reads return 0 and writes have no effect
3-0	BLOCKSIZE		These bits define the length of the buffer region. If all bits are 0, the region is disabled and no data will be stored. User and privilege mode (read): current block size Privilege mode (write):
		0000	Region disabled
		0001	1 KByte
		0010	2 KByte
		0011	4 KByte
		0100	8 KByte
		0101	16 KByte
		0110	32 KByte
		0111	64 KByte
		1000	128 KByte
		1001	256 KByte
		1010..1111	Reserved

26.4.16 DMM Pin Control 0 (DMMPC0)

This register defines if the DMM pins are used in functional or GIO mode. It should only be written when ON/OFF = 0101 and the BUSY bit = 0 (Section 26.4.1). If pins other than the pins specified in Table 26-5 are configured, or DMMCLK and DMMSYNC are programmed as non-functional pins, no operation in trace mode or direct data mode is possible.

Figure 26-23. DMM Pin Control 0 (DMMPC0) [offset = 6Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAFUNC	DATA15FUNC	DATA14FUNC
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13FUNC	DATA12FUNC	DATA11FUNC	DATA10FUNC	DATA9FUNC	DATA8FUNC	DATA7FUNC	DATA6FUNC	DATA5FUNC	DATA4FUNC	DATA3FUNC	DATA2FUNC	DATA1FUNC	DATA0FUNC	CLK-FUNC	SYNC-FUNC
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-21. DMM Pin Control 0 (DMMPC0) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENAFUNC		Functional mode of $\overline{\text{DMMENA}}$ pin. This bit defines whether the pin is used in functional mode or in GIO mode.
			User and privilege mode (read):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
			User and privilege mode (write):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode

Table 26-21. DMM Pin Control 0 (DMMPC0) Field Descriptions (Continued)

Bit	Name	Value	Description
17-2	DATAxFUNC		Functional mode of DMMDATA[x] pin.This bit defines whether the pin is used in functional mode or in GIO mode. If pins are configured in functional mode, only pins define in Table 26-5 have to be used for proper operation.
			User and privilege mode (read):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
			User and privilege mode (write):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
1	CLKFUNC		Functional mode of DMMCLK pin. This bit defines whether the pin is used in functional mode or in GIO mode.
			User and privilege mode (read):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
			User and privilege mode (write):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
0	SYNCFUNC		Functional mode of DMMSYNC pin. This bit defines whether the pin is used in functional mode or in GIO mode.
			User and privilege mode (read):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode
			User and privilege mode (write):
		0	Pin is used in GIO mode
		1	Pin is used in Functional mode

26.4.17 DMM Pin Control 1 (DMMPC1)

The bits in this register define the direction of the individual module pins when in GIO mode.

Figure 26-24. DMM Pin Control 1 (DMMPC1) [offset = 70h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENA-DIR	DATA15DIR	DATA14DIR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13DIR	DATA12DIR	DATA11DIR	DATA10DIR	DATA9DIR	DATA8DIR	DATA7DIR	DATA6DIR	DATA5DIR	DATA4DIR	DATA3DIR	DATA2DIR	DATA1DIR	DATA0DIR	CLKDIR	SYN-CDIR
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

Table 26-22. DMM Pin Control 1 (DMMPC1) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect This bit defines whether the pin is used as input or output in GIO mode
18	ENADIR		Direction of $\overline{\text{DMMENA}}$ pin.
			User and privilege mode (read):
		0	Pin is used as input
		1	Pin is used as output
			User and privilege mode (write):
		0	Pin is set to input
		1	Pin is set to output
17-2	DATAxDIR		Direction of DMMDATA[x] pin. This bit defines whether the pin is used as input or output in GIO mode.
			User and privilege mode (read):
		0	Pin is used as input
		1	Pin is used as output
			User and privilege mode (write):
		0	Pin is set to input
		1	Pin is set to output

Table 26-22. DMM Pin Control 1 (DMMPC1) Field Descriptions (Continued)

Bit	Name	Value	Description
1	CLKDIR		Direction of DMMCLK pin.This bit defines whether the pin is used as input or output in GIO mode.
			User and privilege mode (read):
		0	Pin is used as input
		1	Pin is used as output
			User and privilege mode (write):
		0	Pin is set to input
		1	Pin is set to output
0	SYNCDIR		Direction of DMMSYNC pin. This bit defines whether the pin is used as input or output in GIO mode.
			User and privilege mode (read):
		0	Pin is used as input
		1	Pin is used as output
			User and privilege mode (write):
		0	Pin is set to input
		1	Pin is set to output

26.4.18 DMM Pin Control 2 (DMMPC2)

The bits in this register reflect the digital representation of the voltage level at the module pins. Even if a pin is configured to be an output pin, the level can be read back via this register.

Figure 26-25. DMM Pin Control 2 (DMMPC2) [offset = 74h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAIN	DATA15IN	DATA14IN
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13IN	DATA12IN	DATA11IN	DATA10IN	DATA9IN	DATA8IN	DATA7IN	DATA6IN	DATA5IN	DATA4IN	DATA3IN	DATA2IN	DATA1IN	DATA0IN	CLKIN	SYN-CIN
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-23. DMM Pin Control 2 (DMMPC2) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENAIN		$\overline{\text{DMMENA}}$ input. This bit reflects the state of the pin in all modes.
			User and privilege mode (read):
		0	Logic low (input voltage is V_{IL} or lower)
		1	Logic high (input voltage is V_{IH} or higher)
			Privilege mode (write): writes to this bit have no effect.
17-2	DATAxIN		DMMDATA[x] input. This bit reflects the state of the pin in all modes.
			User and privilege mode (read):
		0	Logic low (input voltage is V_{IL} or lower)
		1	Logic high (input voltage is V_{IH} or higher)
			Privilege mode (write): writes to this bit have no effect.
1	CLKIN		DMMCLK input. This bit reflects the state of the pin in all modes.
			User and privilege mode (read):
		0	Logic low (input voltage is V_{IL} or lower)
		1	Logic high (input voltage is V_{IH} or higher)
			Privilege mode (write): writes to this bit have no effect.

Table 26-23. DMM Pin Control 2 (DMMPC2) Field Descriptions (Continued)

Bit	Name	Value	Description
0	SYNCIN		DMMSYNC input. This bit reflects the state of the pin in all modes
			User and privilege mode (read):
		0	Logic low (input voltage is V_{IL} or lower)
		1	Logic high (input voltage is V_{IH} or higher)
			Privilege mode (write): writes to this bit have no effect.

26.4.19 DMM Pin Control 3 (DMMPC3)

The bits in this register set the pin to logic low or high level if the pin is configured as output (Section 26.4.17).

Figure 26-26. DMM Pin Control 3 (DMMPC3) [offset = 78h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAOUT	DATA15OUT	DATA14OUT
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13OUT	DATA12OUT	DATA11OUT	DATA10OUT	DATA9OUT	DATA8OUT	DATA7OUT	DATA6OUT	DATA5OUT	DATA4OUT	DATA3OUT	DATA2OUT	DATA1OUT	DATA0OUT	CLK-OUT	SYN-COUT
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

Table 26-24. DMM Pin Control 3 (DMMPC3) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENAOUT		Output state of $\overline{\text{DMMENA}}$ pin. This bit sets the pin to logic low or high level
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	Logic low (output voltage is set to V_{OL} or lower)
		1	Logic high (output voltage is set to V_{OH} or higher)

Table 26-24. DMM Pin Control 3 (DMMPC3) Field Descriptions (Continued)

Bit	Name	Value	Description
17-2	DATAxOUT		Output state of DMMDATA[x] pin. This bit sets the pin to logic low or high level.
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	Logic low (output voltage is set to V_{OL} or lower)
		1	Logic high (output voltage is set to V_{OH} or higher)
1	CLKOUT		Output state of DMMCLK pin. This bit sets the pin to logic low or high level
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	Logic low (output voltage is set to V_{OL} or lower)
		1	Logic high (output voltage is set to V_{OH} or higher)
0	SYNCOUT		Output state of DMMSYNC pin. This bit sets the pin to logic low or high level.
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	Logic low (output voltage is set to V_{OL} or lower)
		1	Logic high (output voltage is set to V_{OH} or higher)

26.4.20 DMM Pin Control 4 (DMMPC4)

This register allows to set individual pins to a logic high level without having to do a read-modify-write operation as would be the case with the DMMPC3 register (Section 26.4.19). Writing a zero to a bit will not change the state of the pin.

Figure 26-27. DMM Pin Control 4 (DMMPC4) [offset = 7Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENA-SET	DATA15SET	DATA14SET
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13SET	DATA12SET	DATA11SET	DATA10SET	DATA9SET	DATA8SET	DATA7SET	DATA6SET	DATA5SET	DATA4SET	DATA3SET	DATA2SET	DATA1SET	DATA0SET	CLK-SET	SYNC-SET
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-25. DMM Pin Control 4 (DMMPC4) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENASET		Sets output state of $\overline{\text{DMMENA}}$ pin to logic high. Value in the ENASET bit sets the data output control register bit to 1 regardless of the current value in the ENAOUT bit (Section 26.4.19)
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to V_{OH} or higher)

Table 26-25. DMM Pin Control 4 (DMMPC4) Field Descriptions (Continued)

Bit	Name	Value	Description
17-2	DATAxSET		Sets output state of DMMDATA[x] pin to logic high. Value in the DATAxSET bit sets the data output control register bit to 1 regardless of the current value in the DATAxOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to V_{OH} or higher)
1	CLKSET		Sets output state of DMMCLK pin to logic high. Value in the CLKSET bit sets the data output control register bit to 1 regardless of the current value in the CLKOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to V_{OH} or higher)
0	SYNCSET		Sets output state of DMMSYNC pin logic high. Value in the SYNCSET bit sets the data output control register bit to 1 regardless of the current value in the SYNCOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Logic high (output voltage is set to V_{OH} or higher)

26.4.21 DMM Pin Control 5 (DMMPC5)

This register allows to set individual pins to a logic low level without having to do a read-modify-write operation as would be the case with the DMMPC3 register (Section 26.4.19). Writing a one to a bit will change the output to a logic low level, writing a zero will not change the state of the pin.

Figure 26-28. DMM Pin Control 5 (DMMPC5) [offset = 80h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENA-CLR	DATA15CLR	DATA14CLR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13CLR	DATA12CLR	DATA11CLR	DATA10CLR	DATA9 CLR	DATA8 CLR	DATA7 CLR	DATA6 CLR	DATA5 CLR	DATA4 CLR	DATA3 CLR	DATA2 CLR	DATA1 CLR	DATA0 CLR	CLK-CLR	SYNC-CLR
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-26. DMM Pin Control 5 (DMMPC5) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENACLR		Sets output state of $\overline{\text{DMMENA}}$ pin to logic low. Value in the ENACLR bit clears the data output control register bit to 0 regardless of the current value in the ENAOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Clears the pin to logic low (output voltage is set to V_{OL} or lower)

Table 26-26. DMM Pin Control 5 (DMMPC5) Field Descriptions (Continued)

Bit	Name	Value	Description
17-2	DATAxCLR		Sets output state of DMMDATA[x] pin to logic low. Value in the DATAxCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Clears the pin to logic low (output voltage is set to V_{OL} or lower)
1	CLKCLR		Sets output state of DMMCLK pin to logic low. Value in the CLKCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Clears the pin to logic low (output voltage is set to V_{OL} or lower)
0	SYNCCLR		Sets output state of DMMSYNC pin to logic low. Value in the SYNCCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit (Section 26.4.19).
			User and privilege mode (read):
		0	Logic low (output voltage is V_{OL} or lower)
		1	Logic high (output voltage is V_{OH} or higher)
			User and privilege mode (write):
		0	State of the pin is unchanged
		1	Clears the pin to logic low (output voltage is set to V_{OL} or lower)

26.4.22 DMM Pin Control 6 (DMMPC6)

These bits configure the pins in push-pull or open-drain functionality. If configured to be open-drain, the module only drives a logic low level on the pin. An external pull-up resistor needs to be connected to the pin to pull it high when the pin is in high-impedance mode.

Figure 26-29. DMM Pin Control 6 (DMMPC6) [offset = 84h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAPDR	DATA15PDR	DATA14PDR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13PDR	DATA12PDR	DATA11PDR	DATA10PDR	DATA9PDR	DATA8PDR	DATA7PDR	DATA6PDR	DATA5PDR	DATA4PDR	DATA3PDR	DATA2PDR	DATA1PDR	DATA0PDR	CLK-PDR	SYN-CPDR
R/W-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	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-27. DMM Pin Control 6 (DMMPC6) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENAPDR		Open Drain enable. Enables open drain functionality if the pin is configured as GIO output (DMMPC0[18]=0; Section 26.4.16; DMMPC1[18]=1; Section 26.4.17). If the pin is configured as functional pin (DMMPC0[18]=1; Section 26.4.16), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures pin as push/pull
		1	Configures pin as open drain

Table 26-27. DMM Pin Control 6 (DMMPC6) Field Descriptions (Continued)

Bit	Name	Value	Description
17-2	DATAxPDR		Open Drain enable. Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[x]=0; Section 26.4.16; DMMPC1[x]=1; Section 26.4.17). If the pin is configured as functional pin (DMMPC0[x] = 1; Section 26.4.16), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain
1	CLKPDR		Open Drain enable. Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[1]=0; Section 26.4.16; DMMPC1[1]=1; Section 26.4.17). If the pin is configured as functional pin (DMMPC0[1] = 1; Section 26.4.16), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain
0	SYNCPDR		Open Drain enable. Enables open drain functionality on pin if pin is configured as GIO output (DMMPC0[0]=0; Section 26.4.16; DMMPC1[0]=1; Section 26.4.17). If the pin is configured as functional pin (DMMPC0[0] = 1; Section 26.4.16), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain

26.4.23 DMM Pin Control 7 (DMMPC7)

The bits in register control the pullup/down functionality of a pin. The internal pullup/down can be enabled or disabled by this register. The reset configuration of these bits is device implementation dependent. Please consult the device datasheet this information.

Figure 26-30. DMM Pin Control 7 (DMMPC7) [offset = 88h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAP- DIS	DATA1 5PDIS	DATA1 4PDIS
R-0													R/W-x	R/W-x	R/W-x

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1 3PDIS	DATA1 2PDIS	DATA1 1PDIS	DATA1 0PDIS	DATA9 PDIS	DATA8 PDIS	DATA7 PDIS	DATA6 PDIS	DATA5 PDIS	DATA4 PDIS	DATA3 PDIS	DATA2 PDIS	DATA1 PDIS	DATA0 PDIS	CLKP- DIS	SYN- CPDIS
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-28. DMM Pin Control 7 (DMMPC7) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return 0 and writes have no effect
18	ENADIS		Pull disable. Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[18] = 0; Section 26.4.17).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
17-2	DATAxDIS		Pull disable. Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[x] = 0; Section 26.4.17).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality

Table 26-28. DMM Pin Control 7 (DMMPC7) Field Descriptions (Continued)

Bit	Name	Value	Description
1	CLKDIS		Pull disable. Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[1] = 0; Section 26.4.17).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
0	SYNCDIS		Pull disable. Removes internal pullup/pulldown functionality from pin when configured as input pin (DMMPC1[0] = 0; Section 26.4.17).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality

Note:
If the pullup/down is disabled in DMMPC7 and configured as pulldown in DMMPC8 (Section 26.4.24), then the input buffer is disabled.

26.4.24 DMM Pin Control 8 (DMMPC8)

These bits control if the internal pullup or pulldown is configured on the input pin. A secondary function exists when the pull configuration is disabled and a pulldown is selected. This will disable the input buffer.

Figure 26-31. DMM Pin Control 8 (DMMPC8) [offset = 8Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAPSEL	DATA15PSEL	DATA14PSEL
R-0													R/W-1	R/W-1	R/W-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA13PSEL	DATA12PSEL	DATA11PSEL	DATA10PSEL	DATA9PSEL	DATA8PSEL	DATA7PSEL	DATA6PSEL	DATA5PSEL	DATA4PSEL	DATA3PSEL	DATA2PSEL	DATA1PSEL	DATA0PSEL	CLKPSEL	SYNCPSEL
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 26-29. DMM Pin Control 8 (DMMPC8) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return 0 and writes have no effect
18	ENAPSEL		Pull select. Configures pullup or pulldown functionality if DMMPC7[18] = 0 (Section 26.4.23).
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality
17-2	DATAxPSEL		Pull select. Configures pullup or pulldown functionality if DMMPC7[x] = 0 (Section 26.4.23).
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality

Table 26-29. DMM Pin Control 8 (DMMPC8) Field Descriptions (Continued)

Bit	Name	Value	Description
1	CLKPSEL		Pull select. Configures pullup or pulldown functionality if DMMPC7[1] = 0 (Section 26.4.23).
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality
0	SYNCPSEL		Pull select. Configures pullup or pulldown functionality if DMMPC7[0] = 0 (Section 26.4.23).
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality

Note:
If the pullup/down is disabled in DMMPC7 (Section 26.4.23) and configured as pulldown in DMMPC8, then the input buffer is disabled

Chapter 27

SPNU489C-February 2012

RAM Trace Port (RTP) Module

This document describes the functionality of the RAM trace port (RTP) module. It allows to do data trace of CPU or other master accesses to the internal RAM and peripherals.

Topic **Page**

27.1 Overview

This document describes the functionality of the RAM trace port (RTP) module, which provides the features to datalog the RAM contents of the TMS570 devices or accesses to peripherals without program intrusion. It can trace all data write or read accesses to internal RAM. In addition, it provides the capability to directly transfer data to a FIFO to support a CPU-controlled transmission of the data. The trace data is transmitted over a dedicated external interface.

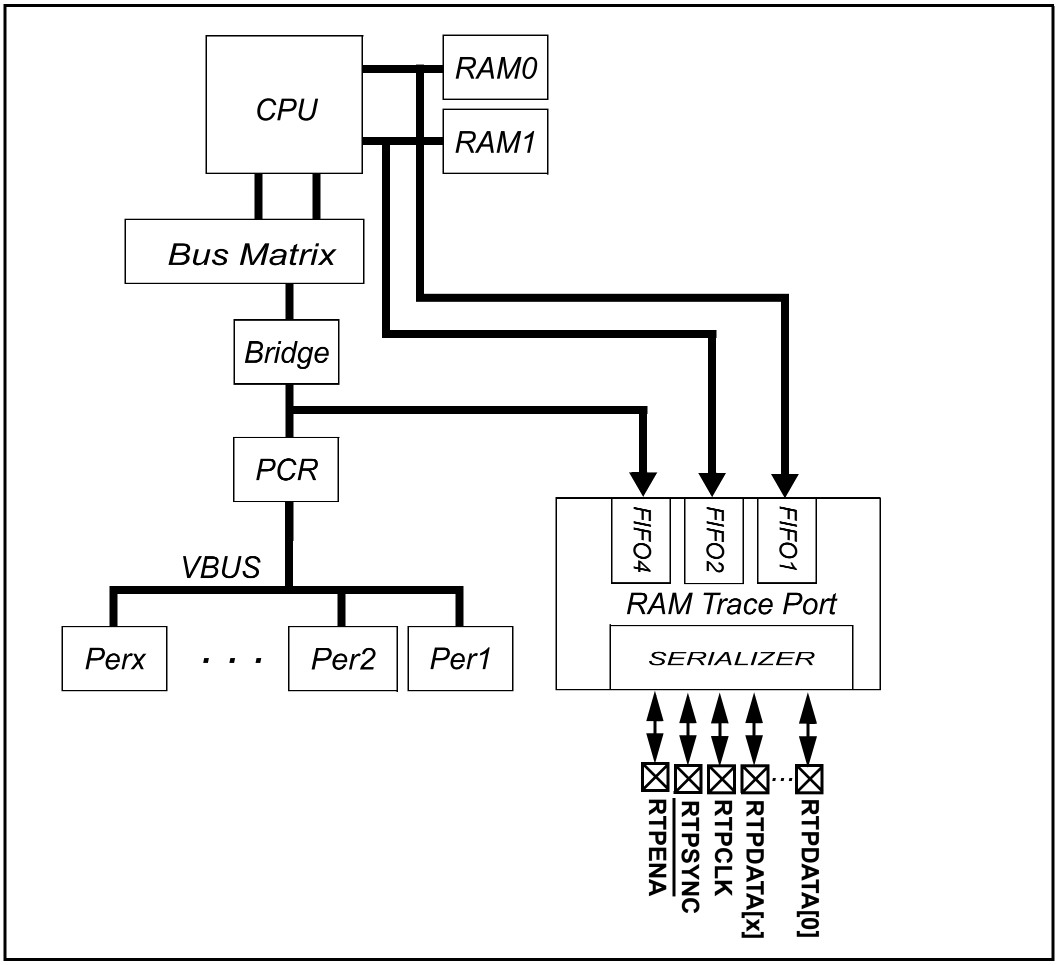
27.1.1 Features

The RTP offers the following features:

- Two modes of operation - Trace Mode and Direct Data Mode
  - Trace Mode
    - Non-intrusive data trace on write or read operation
    - Visibility of RAM content at any time on external capture hardware
    - Trace of peripheral accesses
    - 2 configurable trace regions for each RAM module to limit amount of data to be traced
    - FIFO to store data and address of data of multiple read/write operations
    - Trace of CPU and/or DMA accesses with indication of the master in the transmitted data packet
  - Direct Data Mode
    - Directly write data with the CPU or trace read operations to a FIFO, without transmitting header and address information
- Dedicated synchronous interface to transmit data to external devices
- Free-running clock generation or clock stop mode between transmissions
- up to 100 Mbit per sec/pin transfer rate for transmitting data (up to 133MB/s; see device datasheet for maximum transmission clock frequency)
- Pins not used in functional mode can be used as GIOs

27.2 Block Diagrams

Figure 27-1. Block diagram RAM Trace Port Module



27.3 Module Operation

The RTP module has two modes of operation: Trace mode and Direct Data Mode.

27.3.1 Trace Mode

This mode traces all write or read accesses of CPU and/or a different master to the internal RAMs and the peripheral bus, if the access falls into one of the programmed trace regions. The trace regions allow to restrict the amount of data which is traced. This is done by specifying the start address and the size of the region to be traced. It is not possible to trace write and read operations in the same region at the same time.

Whenever a write or read access occurs, the address, data, size of the access (8, 16, 32, 64 bit), and which module initiated the write or read operation is captured into the FIFO of the corresponding RAM frame. Once new data is in the FIFO and the serializer is empty, the RTP transmits the data into the serializer and starts transmitting it.

The FIFOs are shifting data into the serializer in a round-robin scheme. This means if data is available in multiple FIFOs, the sequence for shifting data into the serializer is FIFO1, FIFO2 and then FIFO4. Only one entry in the respective FIFO is provided to the serializer before switching to the next FIFO. If a FIFO does not hold new data, it will be skipped. This scheme ensures that the FIFOs are drained uniformly.

27.3.1.1 Packet Format in Trace Mode

Figure 27-2 and Figure 27-3 illustrate this format.

Figure 27-2. Packet Format Trace Mode for RAM Locations

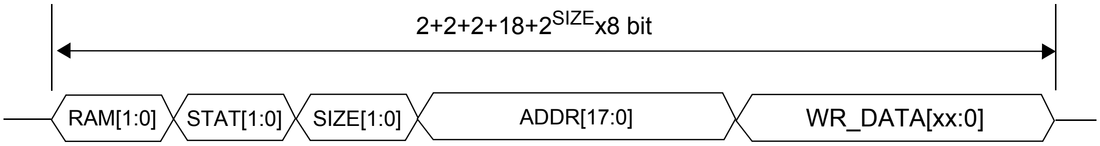
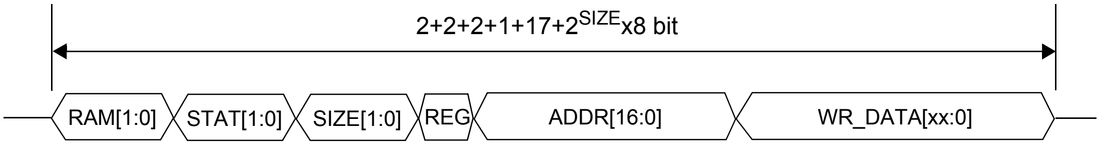


Figure 27-3. Packet Format Trace Mode for Peripheral Locations



When RAM locations are traced, one packet consists of two bits denoting the RAM block in which the data is stored or if the access has been to a peripheral location (Table 27-1), two status bits showing the access initiator or if there was a FIFO overflow (Table 27-2), two bit size (8, 16, 32, or 64 bit) information of the data (Table 27-3), the 18-bit address for RAM accesses and 2^{SIZE}x8 bits of data. If a peripheral location is traced, then the effective address reduces to 17 bits(ADDR[16:0]) and a separate bit (REG) between the SIZE information and the address denotes which programmable region has traced this peripheral access (Table 27-4). With the region identifier, the external hardware can determine which peripheral was traced.

Table 27-1. Encoding of RAM Bits in Trace Mode Packet Format

RAM[1:0]	RAM
00	Cortex-R4 B0TCM
01	Cortex-R4 B1TCM
10	Reserved
11	Peripherals

Table 27-2. Encoding of Status Bits in Trace Mode Packet Format

STAT[1:0]	Status
00	normal entry CPU
01	normal entry other Master
10	reserved
11	overflow of the dedicated FIFO

In the event of a FIFO overflow, an overflow will be signaled in the status bits of the next transmitted packet of that particular FIFO. The last entry in the FIFO will not be overwritten by the new data.

Table 27-3. Encoding of SIZE bits in Trace Mode Packet Format

SIZE[1:0]	Write/Read Size
00	8 bit
01	16 bit
10	32 bit
11	64 bit

Table 27-4. Encoding of REG in Trace Mode Packet Format

REG	Region
0	1
1	2

The packet will be split up into several subpackets when transmitted over the RTP port pins depending on the port width configured. The port width is configured with bits PW[1:0] in the RTPGLBCTRL register. For certain port width configurations and write/read sizes, the number of bits in a packet does not exactly match the port width for the last subpacket. The remaining bits will be filled with zeros.

Table 27-5. Number of Transfers/Packet

	Write/Read Size in bits			
Port Width	8	16	32	64
2	16 → 16	20 → 20	28 → 28	44 → 44
4	8 → 8	10 → 10	14 → 14	22 → 22
8	4 → 4	5 → 5	7 → 7	11 → 11
16	2 → 2	2.5 → 3	3.5 → 4	5.5 → 6

Example: For a 16-bit port and with data of 16-bit, the last transfer has to be padded with eight 0s. This effectively results in a transfer of 48 bits instead of 40. However the whole transfer is completed in 3 RTPCLK cycles.

For a detailed description of the representation of the packet on the RTP port pins, please refer to Section 27.3.5.

27.3.2 Direct Data Mode (DDM)

In this mode, data is written directly by the CPU or other master to a dedicated capture register (RTPDDMW). The data is then transferred from the capture register to the FIFO. In a different configuration the module traces the data on read operations on the RAM directly into the FIFOs. In Direct Data Mode, no information other than the actual data is transmitted. The address of the written data can only be determined by the order of writes or reads by the CPU or other master. This mode is especially useful if a block of data on consecutive addresses has to be transmitted.

The transfer size (8, 16, or 32 bit) is programmable, but cannot be dynamically changed. Data not written/ read in the correct transfer size will be truncated/extended. For example, if the transfer size is programmed to 16 bits and a 32-bit write operation is performed, the data written to the FIFO will be 32-bit wide, however only the upper 16 bits of the FIFO will be transmitted. If an 8-bit operation is performed, bits 8–15 of the FIFO will be indeterminate, so the upper 8 bits of the data transmitted are dependent on the previous contents of the FIFO RAM.

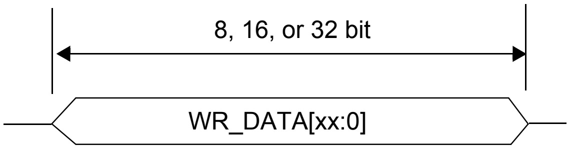
When the module is configured in Direct Data Mode (TM_DDM = 1) to trace write operations (DDM_RW = 1) to the RTPDDMW register, the programming of the trace regions for all FIFOs will be ignored and data tracing, when accessing the addresses defined by the regions, will not occur. If the module is configured in read mode (DDM_RW = 0), and if the read access to a RAM block falls into a valid trace region, the data will be traced into the corresponding FIFO for this RAM block. Since no address information is transmitted in Direct Data Mode, the executing program has to make sure that one FIFO is completely empty (RTPGSR), before new data is traced into the next FIFO.

Note:
Direct Data Mode read operation is not supported on devices with a Cortex-R4 CPU, due to the bus protocol of the TCM interface and certain performance enhancements (e.g. data packing) implemented in the core.

27.3.2.1 Packet Format in Direct Data Mode

Figure 27-4 illustrates this format.

Figure 27-4. Packet Format in Direct Data Mode



In Direct Data Mode write or read operations, only the data written to the RTPDDMW register or the data read from RAM, and therefore captured into the FIFO, will be transmitted. The packet length is programmable (8, 16, or 32 bits).

27.3.3 Trace Regions

To limit the amount of data to be trace, two trace regions per RAM or peripheral are implemented. These can be programmed to specific start addresses and block sizes. Depending on the device configuration (number of RAM blocks), not all regions might be implemented. Trace regions are used in Trace Mode for read or write trace and in Direct Data Mode for read trace. In Direct Data Mode write configuration, the data has to be written directly to RTPDDMW.

The RAM and peripherals start at fixed addresses in the devices memory map. With this the start address of a region does not need to be specified with its full 32-bit address. For RAM regions, only the lower 18-bit need to be programmed. The peripheral address frame covers a wider range and the start address needs to be programmed with the lower 24-bit.

The trace regions do not support a programmable end address, however a block size needs to be specified for each region. The block size can be chosen from as low as 256 Bytes up to 256 kBytes (128 kBytes for peripherals).

27.3.3.1 Inverse Trace Regions

The RTP can be configured to trace accesses which fall into, or are made outside of the specified regions. This can be accomplished by the INV_RGN bit. If this bit is 0, all access which are made inside a region are traced. If the bit is 1, all accesses outside the region are traced. The INV_RGN bit affects all regions of the RTP.

There are certain restrictions when using INV_RGN = 1:

- In this mode up to 2 regions can be excluded from tracing accesses to a particular RAM.
- Inverse trace regions with one or both regions of a RAM programmed with blocksize = 0 is not supported. If only one address range should be excluded from the trace, either the address range has to be covered by both regions (e.g. excluding 1kB range with two 512B regions), or both regions have to be programmed with the same start address and region size. If the whole RAM should be traced, inverse region mode should not be used, instead the 2 regions could be programmed to cover the entire address range with INV_RGN = 0.
- Both regions have to define the same access rights (bits CPU_DMA and RW) for accesses outside of the region of each RAM block, otherwise the result is undefined.
- Peripheral trace in inverse region mode is not supported. The 16 MByte peripheral address range cannot be covered entirely by the 17 bit address definition of the RTP protocol.

27.3.3.2 Overlapping Trace Regions

When in INV_RGN = 0 mode with both regions overlapping and an access is done into the overlapping address range, both regions will be checked for their access rights and if one or both is satisfied, the access will be traced. In the case that both regions would allow the data to be traced, there will still be only one entry into the FIFO.

If accesses to peripherals are done within overlapping regions, the REG bit in the protocol will be 0, denoting Region 1.

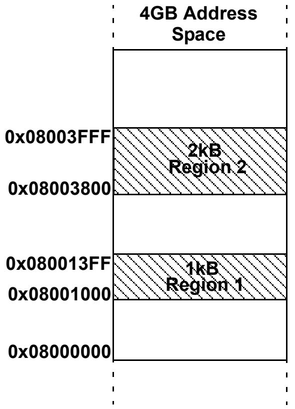
Figure 27-5. Example for Trace Region Setup

2 Trace Regions

- Region 1
  - starts at 0x08001000 with size of 1kB
  - CPU write access are traced
- Region 2
  - starts at 0x08003800 with size of 2kB
  - CPU and other master write accesses are traced

RTPRAM1REG1 = 0x33001000

RTPRAM1REG2 = 0x74003800



27.3.3.3 Cortex-R4 specifics

Due to the bus system used on Cortex-R4, special considerations have to be taken into account. Figure 27-1 shows the block diagram of the RTP connected to the Cortex-R4 RAM interface.

Both interfaces to RAM0 and RAM1 are 64-bit wide. RAM0 and RAM1 are building a consecutive address range, where the 64-bit addresses 0x00, 0x10, 0x20, ... reside in RAM0 and 0x08, 0x18, 0x28, ... reside in RAM1.

Considerations/Restrictions

- To trace a certain address range, the regions of both RAM0 (RTPRAM1REG[1:2]) and RAM1 (RTPRAM2REG[1:2]) need to be set up for the same start address and region size. Otherwise only every other 64-bit access will be traced.
- Direct Data Mode read operation is not supported on Cortex-R4. This is because the CPU uses 64-bit accesses for read operations even though the intended accesses is sub-64-bit wide. Since Direct Data Mode only supports 32-bit data transfers, and the Cortex-R4 RAM interface does not provide information which byte out of the 64-bit is accessed with the read operation, the RTP cannot determine the correct data value.
- If the RAM is protected by ECC, only 64-bit write accesses can be traced. Every 64-bit word in the RAM is protected by a corresponding 8-bit ECC checksum. When a sub-64-bit write access is performed, the Cortex-R4 has to do a read-modify-write operation of the 64-bit word to be modified in order to calculate the corresponding checksum and then write it back to memory. External hardware can still determine which portion of the 64-bit word has been modified, since the other bytes in this word did not change.

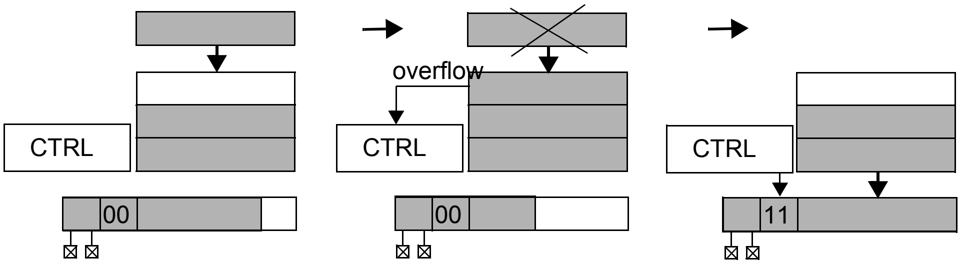
27.3.4 Overflow/Empty Handling

In case the application does RAM accesses faster than the FIFO can be emptied via the external pin interface, the FIFO can overflow. The user can choose whether the program execution/data transfer should be suspended, or an overflow should be signaled in the status bits of the next, to be transmitted, message of this particular FIFO. If program execution is resumed, the data will be lost. The overflow will not be signaled in the message that is already in the serializer and being transmitted when the overflow occurs.

> The status information will only be transmitted in Trace Mode, since the Direct Data Mode packet does not contain any status information.

When an overflow in a FIFO occurs, the corresponding bit in RTPGSR will be set.

Figure 27-6. FIFO Overflow Handling



27.3.5 Signal Description

RTPCLK — This clock signal is used to clock out the data of the serializer. Depending on the CONTCLK bit, the clock can be suspended between packets or it can be free running. The RTPCLK frequency can be adjusted by the PRESCALER bits.

RTPSYNC — The module provides a packet-sync signal. This signal will go high on the rising edge of RTPCLK and will be valid for one RTPCLK cycle to synchronize external hardware to the data stream. The RTPSYNC pulse will be generated for each new packet.

$\overline{\text{RTPENA}}$ This signal is an input and can be used by external hardware to stop the data transmission between packets. When the $\overline{\text{RTPENA}}$ signal goes high, the RTP will finish the current packet transmission and then stop. Once the signal is pulled low again, the RTP will resume the transfer if data is still present in the serializer or FIFOs.
The $\overline{\text{RTPENA}}$ signal does not have to be used for proper module operation. It can be used in GIO mode if the external hardware cannot generate this signal. Overflows of the external system cannot be handled in this case.

RTPDATA[15:0] These pins are used to do the actual data transfer. Data changes with the rising edge of RTPCLK. The port can be configured for different widths (PW[1:0]). The minimum port width supported is 2 pins. See Table 27-9 which pins are used for the port.

Figure 27-7 shows an example of multiple packet transmissions in Trace Mode with an interruption between packets because of $\overline{\text{RTPENA}}$ pulled high.

Figure 27-7. RTP Packet Transfer with Sync Signal

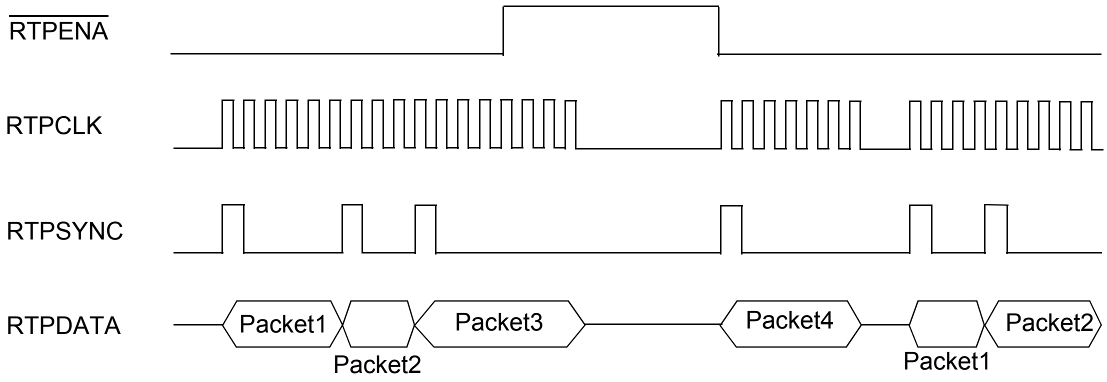
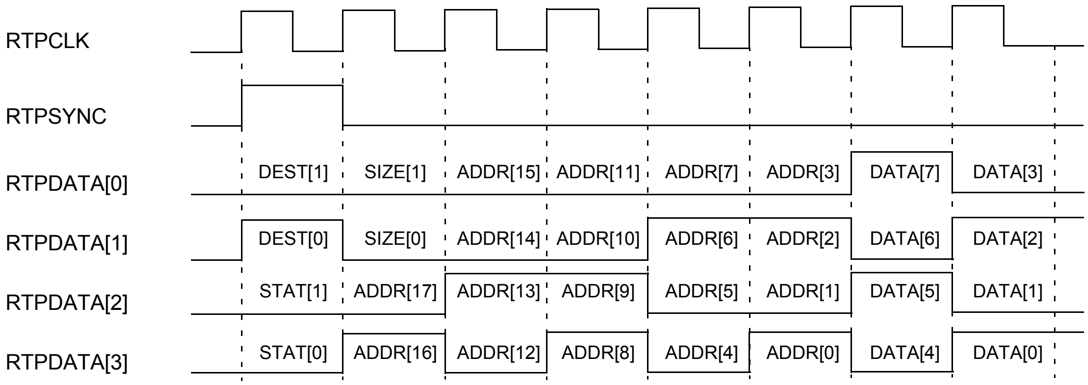


Figure 27-8 shows an example of a 4-bit data port with 8-bit write data (0xA5) written into RAM1 (address 0x12345) with no overflow in trace mode.

Figure 27-8. Packet Format in Trace Mode



27.3.6 Data Rate

The module is configurable to support different RTPCLK frequencies. Please see the device datasheet for the maximum supported frequency. HCLK will be prescaled to achieve the desired RTPCLK frequency. The prescaler supports prescale values from 1 to 8 (RTPGLBCTRL).

The effective bandwidth depends on the configuration of the module and the average data width transmitted in the packets.

27.4 GIO Function

Pins which are not used for RTP functionality can be used as normal GIO pins. If pins should be used in functional mode or GIO mode can be programmed in RTPPC0. The direction of the pins can be chosen in RTPPC1.

Module pins can have either an internal pullup or active pulldown that makes it possible to leave the pins unconnected externally when configured as inputs. The pins can be programmed to have the active pull capability by writing a 0 to the corresponding bit in the RTPPC7 register. Writing a 1 to the corresponding bit disables the active pull functionality of the pin. A pull up can be configured by writing 1 to the corresponding bit in RTPPC8 register. Writing 0 will activate the pulldown capability. The pullup/pulldown is deactivated when a bidirectional pin is configured as an output. If the pullup/down capability is disabled (RTPPC7) and the pull is configured as pulldown (RTPPC8) the input buffer will be disabled.

The GIO pin can be configured to include an open drain functionality when they are configured as output pins. This is done by writing a 1 into the corresponding bit of the RTPPC6 register. When the open drain functionality is enabled, a zero written to the data output register (RTPPC3) forces the pin to a low output voltage (V_{OL} or lower), whereas writing a 1 to the data output register (RTPPC3) forces the pin to a high impedance state. The open drain functionality is disabled when the pin is configured as an input pin.

27.5 Control Registers

This section describes the RTP module registers. The registers support 8-bit, 16-bit, and 32-bit writes. The base address of the RTP module is 0xFFFFFA00.

Table 27-6. Module Registers

Offset Address[1] 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
0x00 RTPGLBCTRL Page 1908	Reserved							TEST	Reserved					PRESCALER[2:0]		
	Reserved		DDM_WIDTH(1–0)		DDM_RW	TM_DDM	PW(1–0)		RESET	CONTCLK	HOVF	Reserved	ON/OFF(2–0)			
0x04 RTPTRENA Page 1913	Reserved							ENA4	Reserved							
	Reserved							ENA2	Reserved							ENA1
0x08 RTPGSR Page 1915	Reserved															
	Reserved			EMPTYSER	EMPTYPER	Reserved	EMPTY2	EMPTY1	Reserved				OVFPER	Reserved	OVF2	OVF1
0x0C–10 RTPRAM1REG[1:2] Page 1917	Reserved	CPU_DMA(1–0)		RW	BLOCKSIZE(3–0)				Reserved						STARTADDR(17–16)	
	STARTADDR(15–0)															
0x14–18 RTPRAM2REG[1:2] Page 1919	Reserved	CPU_DMA(1–0)		RW	BLOCKSIZE(3–0)				Reserved						STARTADDR(17–16)	
	STARTADDR(15–0)															
0x1C–20 Reserved	Reserved															
0x24–28 RTPPERREG[1:2] Page 1921	Reserved	CPU_DMA(1–0)		RW	BLOCKSIZE(3–0)				STARTADDR(23–16)							
	STARTADDR(15–0)															

[1] Address given is offset from start of peripheral frame.

Table 27-6. Module Registers (Continued)

Offset Address[1] 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
0x2C RTPDDMW Page 1923	DATA [31:16]															
	DATA [15:0]															
0x34 RTPPC0 Page 1924	Reserved													ENAFUNC	CLKFUNC	SYNCFUNC
	DATA[15:0]FUNC															
0x38 RTPPC1 Page 1926	Reserved													ENADIR	CLKDIR	SYNCDIR
	DATA[15:0]DIR															
0x3C RTPPC2 Page 1928	Reserved													ENAIN	CLKIN	SYNCIN
	DATA[15:0]IN															
0x40 RTPPC3 Page 1930	Reserved													ENAOUT	CLKOUT	SYNCOUT
	DATA[15:0]OUT															
0x44 RTPPC4 Page 1932	Reserved													ENASET	CLKSET	SYNCSET
	DATA[15:0]SET															
0x48 RTPPC5 Page 1934	Reserved													ENACLR	CLKCLR	SYNCCLR
	DATA[15:0]CLR															
0x4C RTPPC6 Page 1936	Reserved													ENAPDR	CLKPDR	SYNCPDR
	DATA[15:0]PDR															

[1] Address given is offset from start of peripheral frame.

Table 27-6. Module Registers (Continued)

Offset Address[1] 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
0x50 RTPPC7 Page 1938	Reserved													ENAPDIS	CLKPDIS	SYNCPDIS
	DATA[15:0]PDIS															
0x54 RTPPC8 Page 1940	Reserved													ENAP SEL	CLKSEL	SYNCSEL
	DATA[15:0]PSEL															

[1] Address given is offset from start of peripheral frame.

27.5.1 RTP Global Control Register (RTPGLBCTRL)

The configuration of the module can be changed with this register. Figure 27-9 and Table 27-7 illustrate this register.

Figure 27-9. RTP Global Control Register (RTPGLBCTRL) [offset = 00h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="7">Reserved</td><td>TEST</td><td colspan="5">Reserved</td><td colspan="3">PRESCALER(2–0)</td></tr>
<tr><td colspan="7">R-0</td><td>R/WP-0</td><td colspan="5">R-0</td><td colspan="3">R/WP-111b</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">DDM_WIDTH(1–0)</td><td>DDM_RW</td><td>TM_DDM</td><td colspan="2">PW(1–0)</td><td>RESET</td><td>CONTCLK</td><td>HOVF</td><td>INV_RGN</td><td colspan="4">ON/OFF(3–0)</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R/WP-0</td><td>R/WP-0</td><td>R/WP-0</td><td colspan="2">R/WP-0</td><td>R/WP-0</td><td>RWP-0</td><td>R/WP-0</td><td>R/WP-0</td><td colspan="4">R/WP-0101</td></tr>
</table>

R = Read; WP = Write in privilege mode only; -*n* = Value after reset

.

Table 27-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return zeros and writes have no effect.
24	TEST		By setting the bit, the FIFO RAM will be mapped into the SYSTEM Peripheral frame starting at address 0xFFF83000. Each FIFO will start at a 1k boundary. Each FIFO entry will be aligned to a 128 bit boundary. See Table 27-8 for a listing of the FIFOs and their corresponding addresses.
			User and privilege mode (read):
		0	The FIFO RAM is not accessible in the memory map.
		1	The FIFO RAM is mapped to address 0xFFF83000
			Privilege mode (write):
		0	Disables mapping of the FIFO RAM
		1	Enables mapping of the FIFO RAM into address 0xFFF83000

Table 27-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
18-6	PRESCALER(2–0)		The prescaler divides HCLK down to the desired RTPCLK frequency. The maximum RTPCLK frequency specified in the device datasheet must not be exceeded. No dynamic change of RTPCLK is supported. The module should be switched off by the ON/OFF bits in this register before changing the prescaler.
			User and privilege mode read, privilege mode write:
		000	The prescaler is HCLK/1.
		001	The prescaler is HCLK/2.
		010	The prescaler is HCLK/3.
		011	The prescaler is HCLK/4.
		100	The prescaler is HCLK/5.
		101	The prescaler is HCLK/6.
		110	The prescaler is HCLK/7.
		111	The prescaler is HCLK/8.
15–14	Reserved		Reads return 0 and writes have no effect
13–12	DDM_WIDTH(1–0)		Direct data mode word size width. This bit field configures the number of bits that will be transmitted in Direct Data Mode.
			User and privilege mode read, privilege mode write:
		00	The word size width is 8 bits.
		01	The word size width is 16 bits.
		10	The word size width is 32 bits.
		11	Reserved
11	DDM_RW		Direct Data Mode read/write.
			User and privilege mode (read):
		0	Read tracing in Direct Data Mode is enabled.
		1	Write tracing in Direct Data Mode to DDMW register is enabled.
			Privilege mode (write):
		0	Enable read tracing in Direct Data Mode. The RW bits in the RTPRAMxREGy registers to be ignored.
		1	Write tracing in Direct Data Mode to DDMW register is enabled. The RW bits in the RTPRAMxREGy registers to be ignored.

Table 27-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
10	TM_DDM		Trace mode or Direct Data Mode.
			User and privilege mode (read):
		0	Module is configured in Trace Mode.
		1	Module is configured in Direct Data Mode.
			Privilege mode (write):
		0	Configure module to Trace Mode.
		1	Configure module to Direct Data Mode.
9-8	PW(1–0)		Port width. This bit field configures the RTP to the desired port width. Pins that are not used for functional mode can be used as GIO pins. See Table 27-9 which pins are used for the port.
		00	The RTP is 2 pins wide.
		01	The RTP is 4 pins wide.
		10	The RTP is 8 pins wide.
		11	The RTP is 16 pins wide.
7	RESET		This bit resets the state machine and the registers to their reset value. This reset ensures that no data left in the FIFOs is shifted out after switching on the module with the ON/OFF bit.
			User and privilege mode (read):
		0	The RTP module out of reset.
		1	The RTP module is in reset.
			Privilege mode (write):
		0	Do not reset the module.
		1	Reset the module.
6	CONTCLK		Continuous RTPCLK enable.
			User and privilege mode (read):
		0	The RTPCLK is stopped between transmissions.
		1	The RTPCLK is free running.
			Privilege mode (write):
		0	Stop RTPCLK between transmissions.
		1	Configure RTPCLK as free running.

Table 27-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
5	HOVF		Halt on overflow. This bit indicates whether the CPU or DMA is halted while only one location in the FIFO is empty in Trace Mode or Direct Data Mode (read).
			User and privilege mode (read):
		0	The current data transfer to the FIFO will not be suspended in case of a full FIFO.
		1	The current data transfer to the FIFO will be suspended in case of a full FIFO.
			Privilege mode (write):
		0	The halt on FIFO overflow will be disabled. The data transfer will not be suspended and will be discarded. Data written to the RTPDDMW register will overwrite the RTPDDMW register.
		1	The halt on FIFO overflow will be enabled. Data written to the already full FIFO will be written once the FIFO is emptied again. The data transfer to the FIFO will be suspended and signaled to the CPU or other master while there is still data to be shifted out. When there is an empty FIFO location again, the transfer of the data to the FIFO will be finished.
4	INV_RGN		Trace inside or outside of defined trace regions.
			User and privilege mode (read):
		0	Accesses inside the trace regions are traced
		1	Accesses outside of trace regions are traced
			Privilege mode (write):
		0	Allow tracing of accesses inside the regions set in RTPRAMxREGy
		1	Allow tracing of accesses outside the regions set in RTPRAMxREGy

Table 27-7. RTP Global Control Register (RTPGLBCTRL) Field Descriptions (Continued)

Bit	Name	Value	Description
3-0	ON/OFF		On/Off switch.
			User and privilege mode (read):
		1010	Tracing of data is enabled
		all other	Tracing of data is disabled.
			Privilege mode (write):
		1010	Enable Tracing of data. If there is any previous captured data remaining, it will be shifted out.
		all other	Disable tracing of data. If there is still data left in the shift register, it will be shifted out before disabling the shift operations. The data captured in the FIFO remains there until the ON/OFF bits are set to 1010.
			NOTE: It is recommended to write 0101 to disable the module to prevent a soft error from enabling the module inadvertently by a single bit flip.

Table 27-8. FIFO Corresponding Addresses

FIFO	Address
1	**0xFFF83000**
2	**0xFFF83400**
4	**0xFFF83C00**

Table 27-9. Pins Used for Data Communication

Port Width (PW)	Pins Used
00	RTPDATA[1:0]
01	RTPDATA[3:0]
10	RTPDATA[7:0]
11	RTPDATA[15:0]

27.5.2 RTP Trace Enable Register (RTPTRENA)

This register enables/disables tracing of the different RAM blocks or the peripherals individually. Figure 27-10 and Table 27-10 illustrate this register.

Figure 27-10. RTP Trace Enable Register (RTPTRENA) [offset = 04h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							ENA4	Reserved							
R-0							R/WP-0	R-0							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ENA2	Reserved							ENA1
R-0							R/WP-0	R-0							R/WP-0

R = Read, W = Write, P = Privileged, U = Undefined; -*n* = Value after reset

Table 27-10. RTP Trace Enable Register (RTPTRENA) Field Descriptions

Bit	Name	Value	Description
31–25	Reserved		Reads return 0 and writes have no effect
24	ENA4		Enable tracing for peripherals. This bit enables tracing into FIFO4 in trace mode (read/write) or direct data mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO4 will be disabled.
			User and privilege mode (read):
		0	Tracing is disabled.
		1	Tracing is enabled.
			Privilege mode (write):
		0	Disable tracing. If RTPGLBCTRL.ON/OFF = 1010, data already captured in FIFO4 will still be transmitted.
		1	Enable tracing.
23-9	Reserved		Reads return 0 and writes have no effect

Table 27-10. RTP Trace Enable Register (RTPTRENA) Field Descriptions (Continued)

Bit	Name	Value	Description
8	ENA2		Enable tracing for RAM block 2. This bit enables tracing into FIFO2 in Trace Mode (read/write) or Direct Data Mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO2 will be disabled.
			User and privilege mode (read):
		0	Tracing is disabled.
		1	Tracing is enabled.
			Privilege mode (write):
		0	Disable tracing. If RTPGLBCTRL.ON/OFF = 1010, data already captured in FIFO2 will still be transmitted.
		1	Enable tracing.
7-1	Reserved		Reads return 0 and writes have no effect
0	ENA1		Enable tracing for RAM block 1. This bit enables tracing into FIFO1 in Trace Mode (read/write) or Direct Data Mode (read) operations. In Direct Data Mode write operations, this bit will be ignored and tracing into FIFO1 will be disabled.
			User and privilege mode (read):
		0	Tracing is disabled.
		1	Tracing is enabled.
			Privilege mode (write):
		0	Disable tracing. If RTPGLBCTRL.ON/OFF = 1010, data already captured in FIFO1 will still be transmitted.
		1	Enable tracing.

27.5.3 RTP Global Status Register (RTPGSR)

This register provides status information of the module. Figure 27-11 and Table 27-11 illustrate this register.

Figure 27-11. RTP Global Status Register (RTPGSR) [offset = 08h]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td colspan="16">R-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">Reserved</td><td>EMPTY-SER</td><td>EMP-TYPER</td><td>Reserved</td><td>EMPTY 2</td><td>EMPTY 1</td><td colspan="4">Reserved</td><td>OVF-PER</td><td>Reserved</td><td>OVF2</td><td>OVF1</td></tr>
<tr><td colspan="3">R-0</td><td>R-1</td><td>R-1</td><td>R-1</td><td>R-1</td><td>R-1</td><td colspan="4">R-0</td><td>R/WCP-0</td><td>R-0</td><td>R/WCP-0</td><td>R/WCP-0</td></tr>
</table>

R = Read; WCP = Write clear in privilege mode only; -*n* = Value after reset

.

Table 27-11. RTP Global Status Register (RTPGSR) [offset = 08h] Field Descriptions

Bit	Name	Value	Description
31–13	Reserved		Reads return 0 and writes have no effect.
12	EMPTYSER		Serializer empty. This bit determines if there is data left in the serializer.
		0	Serializer holds data which is shifted out
		1	Serializer is empty.
11	EMPTYPER		Peripheral FIFO empty. This bit determines if there are entries left in the FIFO.
		0	FIFO4 contains entries
		1	FIFO4 is empty.
10	Reserved		Reads return 1 and writes have no effect.
9	EMPTY2		RAM block 2 FIFO empty. This bit determines if there are entries left in the FIFO.
		0	FIFO2 contains entries
		1	FIFO2 is empty.
8	EMPTY1		RAM block 1 FIFO empty. This bit determines if there are entries left in the FIFO.
		0	FIFO1 contains entries
		1	FIFO1 is empty.
7-4	Reserved		Reads return 0 and writes have no effect.

Table 27-11. RTP Global Status Register (RTPGSR) [offset = 08h] Field Descriptions (Continued)

Bit	Name	Value	Description
3	OVFPER		Overflow peripheral FIFO. This flag indicates that FIFO4 had all locations full and another attempt to write data to it occurred. The bit will not be cleared automatically if the FIFO is emptied again. The bit will stay set until the CPU clears it.
			User and privilege mode (read):
		0	No overflow occurred.
		1	An overflow occurred.
			Privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	The bit is cleared.
2	Reserved		Reads return 0 and writes have no effect.
1	OVF2		Overflow RAM block 2 FIFO. This flag indicates that FIFO2 had all locations full and another attempt to write data to it occurred. The bit will not be cleared automatically if the FIFO is emptied again. The bit will stay set until the CPU clears it.
			User and privilege mode (read):
		0	No overflow occurred.
		1	An overflow occurred.
			Privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	The bit is cleared.
0	OVF1		Overflow RAM block 1 FIFO. This flag indicates that FIFO1 had all locations full and another attempt to write data to it occurred. The bit will not be cleared automatically if the FIFO is emptied again. The bit will stay set until the CPU clears it.
			User and privilege mode (read):
		0	No overflow occurred.
		1	An overflow occurred.
			Privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	The bit is cleared.

27.5.4 RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2])

Figure 27-12 and Table 27-12 illustrate these registers.

Figure 27-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) [offset = 0Ch, 10h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CPU_DMA(1–0)		RW	BLOCKSIZE(3–0)				Reserved						STARTADDR(17–16)	
R-0	R/WP-00		R/WP-0	R/WP-0				R-0						R/WP-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDR(15–0)															
R/WP-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return 0 and writes have no effect.
30-29	CPU_DMA(1–0)		CPU and/or other master access. This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master.
			User and privilege mode read, privilege mode write:
		00	Read or write operations are traced when coming from the CPU and the other master.
		01	Read or write operations are traced only when coming from the CPU.
		10	Read or write operations are traced only when coming from the other master.
		11	Reserved

Table 27-12. RTP RAM 1 Trace Region [1:2] Register (RTPRAM1REG[1:2]) Field Descriptions (Continued)

Bit	Name	Value	Description
28	RW		Read/Write. This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL[11]), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.
			User and privilege mode (read):
		0	Read operations will be captured.
		1	Write operations will be captured.
			Privilege mode (write):
		0	Trace read accesses.
		1	Trace write accesses.
27-24	BLOCKSIZE(3–0)		These bits define the length of the trace region. Depending on the setting of INV_RGN, accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.
			Region size (in bytes):
		0000	0
		0001	256
		0010	512
		0011	1K
		0100	2K
		...	...
		1010	128K
		1011	256K
		1100–1111	Reserved
23-18	Reserved		Reads return 0 and writes have no effect.
17-0	STARTADDR(17–0)	0–3 FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

27.5.5 RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2])

Figure 27-13 and Table 27-13 illustrate these registers.

Figure 27-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) [offset = 14h, 18h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CPU_DMA(1–0)		RW	BLOCKSIZE(3–0)				Reserved						STARTADDR(17–16)	
R-0	R/WP-00		R/WP-0	R/WP-0				R-0						R/WP-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STARTADDR(15–0)															
R/WP-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; *-n* = Value after reset

.

Table 27-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return 0 and writes have no effect.
30-29	CPU_DMA(1–0)		CPU and/or other master access. This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master.
			User and privilege mode read, privilege mode write:
		00	Read or write operations are traced when coming from the CPU and the other master.
		01	Read or write operations are traced only when coming from the CPU.
		10	Read or write operations are traced only when coming from the other master.
		11	Reserved

Table 27-13. RTP RAM 2 Trace Region [1:2] Register (RTPRAM2REG[1:2]) Field Descriptions (Continued)

Bit	Name	Value	Description
28	RW		Read/Write. This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL[11]), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.
			User and privilege mode (read):
		0	Read operations will be captured.
		1	Write operations will be captured.
			Privilege mode (write):
		0	Trace read accesses.
		1	Trace write accesses.
27-24	BLOCKSIZE(3–0)		These bits define the length of the trace region. Depending on the setting of INV_RGN, accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.
			Region size (in bytes):
		0000	0
		0001	256
		0010	512
		0011	1K
		0100	2K
		...	...
		1010	128K
		1011	256K
		1100–1111	Reserved
23-18	Reserved		Reads return 0 and writes have no effect.
17-0	STARTADDR(17–0)	0–3 FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

27.5.6 RTP Peripheral Trace Region [1:2] Registers (RTPPERREG[1:2])

FIFO4 is dedicated for tracing the peripheral accesses. Since the peripheral frame is 16 Mbytes, the start address has to be defined as a 24-bit value. However, only bits 16 to 0 will be transmitted in the protocol. Bit REG in the protocol will be 0 if there was an access to the range defined by RTPPERREG1. REG will be 1 if the access was into the range defined by RTPPERREG2. Figure 27-14 and Table 27-14 illustrate these registers.

Figure 27-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) [offset = 24h, 28h]

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">CPU_DMA(1–0)</td><td>RW</td><td colspan="4">BLOCKSIZE(3–0)</td><td colspan="8">STARTADDR(23–16)</td></tr>
<tr><td>R-0</td><td colspan="2">R/WP-0</td><td>R/WP-0</td><td colspan="4">R/WP-0</td><td colspan="8">R/WP-0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">STARTADDR(15–0)</td></tr>
<tr><td colspan="16">R/WP-0</td></tr>
</table>

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) Field Descriptions

Bit	Name	Value	Description
31	Reserved		Reads return 0 and writes have no effect.
30-29	CPU_DMA(1–0)		CPU and/or other master access. This bit field indicates if read or write operations are traced either coming from the CPU and/or from the other master. User and privilege mode read, privilege mode write:
		00	Read or write operations are traced when coming from the CPU and the other master.
		01	Read or write operations are traced only when coming from the CPU.
		10	Read or write operations are traced only when coming from the other master.
		11	Reserved

Table 27-14. RTP Peripheral Trace Region [1:2] Register (RTPPERREG[1:2]) Field Descriptions (Continued)

Bit	Name	Value	Description
28	RW		Read/Write. This bit indicates if read or write operations are traced in Trace Mode or Direct Data Mode (read operation). If configured for write in Direct Data Mode (RTPGLBCTRL[11]), the data captured will be discarded. A write operation in Direct Data Mode has to be directly to the RTPDDMW register instead of to RAM. Depending on the INV_RGN bit setting, accesses into or outside the region will be traced.
			User and privilege mode (read):
		0	Read operations will be captured.
		1	Write operations will be captured.
			Privilege mode (write):
		0	Trace read accesses.
		1	Trace write accesses.
27-24	BLOCKSIZE(3–0)		These bits define the length of the trace region. Depending on the setting of INV_RGN, accesses inside or outside the region defined by the start address and blocksize will be traced. If all bits of BLOCKSIZE are 0, the region is disabled and no data will be captured.
			Region size (in bytes):
		0000	0
		0001	256
		0010	512
		0011	1K
		0100	2K
		...	...
		1010	128K
		1011	256K
		1100–1111	Reserved
23-0	STARTADDR(23–0)	0–FF FFFFh	These bits define the starting address of the address region that should be traced. The start address has to be a multiple of the block size chosen. If the start address is not a multiple of the block size, the start of the region will begin at the next lower block size boundary.

27.5.7 RTP Direct Data Mode Write Register (RTPDDMW)

The CPU has to write data to this register if the module is used in Direct Data Mode write configuration. Figure 27-15 and Table 27-15 illustrate this register.

Figure 27-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA(31–16)															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA(16–0)															
R/W-0															

R = Read, W = Write; -*n* = Value after reset

.

Table 27-15. RTP Direct Data Mode Write Register (RTPDDMW) [offset = 2Ch] Field Descriptions

Bit	Name	Value	Description
31–0	DATA(31–0)	0–FFFF FFFFh	This register must be written to in a Direct Data Mode write operation to store the data into FIFO1. Data written must be right-aligned. If the FIFO is full, the reaction depends on the setting of the HOVF bit. If the bit is set, the master writing the data will be waitstated. If the bit is cleared, previous data written to the register will be overwritten. Reads of this register always return 0.

27.5.8 RTP Pin Control 0 Register (RTPPC0)

This register configures the RTP pins as functional or GIO pins. Once the pin is configured in functional mode, it overrides the settings in the RTPPC1 register. Writing to RTPPC3, RTPPC4 and RTPPC5 will have no effect for pins configured as functional pins. Figure 27-16 and Table 27-16 illustrate this register.

Figure 27-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAFUNC	CLK-FUNC	SYNC-FUNC
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]FUNC															
R/W-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h] Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENAFUNC		Functional mode of $\overline{\text{RTPENA}}$ pin.
			User and privilege mode (read):
		0	Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
		0	Configure pin to GIO mode.
		1	Configure pin to functional mode.
17	CLKFUNC		Functional mode of RTPCLK pin.
			User and privilege mode (read):
		0	Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
		0	Configure pin to GIO mode.
		1	Configure pin to functional mode.

Table 27-16. RTP Pin Control 0 Register (RTPPC0) [offset = 34h] Field Descriptions (Continued)

Bit	Name	Value	Description
16	SYNCFUNC		Functional mode of RTPSYNC pin.
			User and privilege mode (read):
		0	Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
		0	Configure pin to GIO mode.
		1	Configure pin to functional mode.
15-0	DATA[15:0]FUNC		Functional mode of RTPDATA[15:0] pins. These bits define whether the pins are used in functional mode or in GIO mode. Each bit represents a single pin.
			User and privilege mode (read):
		0	Pin is used in GIO mode.
		1	Pin is used in functional mode.
			User and privilege mode (write):
		0	Configure pin to GIO mode.
		1	Configure pin to functional mode.

27.5.9 RTP Pin Control 1 Register (RTPPC1)

Once the pin is configured in functional mode (RTPPC0), configuring the corresponding bit in RTPPC1 to 0 will not disable the output driver. Figure 27-17 and Table 27-17 illustrate this register.

Figure 27-17. RTP Pin Control 1 Register (RTPPC1) [offset = 38h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENADIR	CLKDIR	SYN-CDIR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]DIR															
R/W-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-17. RTP Pin Control 1 Register (RTPPC1) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENADIR		Direction of $\overline{\text{RTPENA}}$ pin. This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
			User and privilege mode (write):
		0	Configure pin to input mode.
		1	Configure pin to output mode.
17	CLKDIR		Direction of RTPCLK pin.This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
			User and privilege mode (write):
		0	Configure pin to input mode.
		1	Configure pin to output mode.

Table 27-17. RTP Pin Control 1 Register (RTPPC1) Field Descriptions (Continued)

Bit	Name	Value	Description
16	SYNCDIR		Direction of RTPSYNC pin.This bit defines whether the pin is used as input or output in GIO mode. This bit has no effect when the pin is configured in functional mode.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
			User and privilege mode (write):
		0	Configure pin to input mode.
		1	Configure pin to output mode.
15-0	DATA[15:0]DIR		Direction of RTPDATA[15:0] pins. These bits define whether the pins are used as input or output in GIO mode. These bits have no effect when the pins are configured in functional mode. Each bit represents a single pin.
			User and privilege mode (read):
		0	Pin is used as input.
		1	Pin is used as output.
			User and privilege mode (write):
		0	Configure pin to input mode.
		1	Configure pin to output mode.

27.5.10 RTP Pin Control 2 Register (RTPPC2)

This register represents the input value of the pins if when in GIO or functional mode. Figure 27-18 and Table 27-18 illustrate this register.

Figure 27-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]IN															
R-x															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch] Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENAIN		$\overline{\text{RTPENA}}$ input. This bit reflects the state of the pin in all modes. Writes to this bit have no effect.
			User and privilege mode (read):
		0	The pin is at logic low (0) (input voltage is V_{IL} or lower).
		1	The pin is at logic high (1) (input voltage is V_{IH} or higher).
17	CLKIN		RTPCLK input. This bit reflects the state of the pin in all modes. Writes to this bit have no effect.
			User and privilege mode (read):
		0	The pin is at logic low (0) (input voltage is V_{IL} or lower).
		1	The pin is at logic high (1) (input voltage is V_{IH} or higher).
16	SYNCIN		RTPSYNC input. This bit reflects the state of the pin in all modes. Writes to this bit have no effect.
			User and privilege mode (read):
		0	The pin is at logic low (0) (input voltage is V_{IL} or lower).
		1	The pin is at logic high (1) (input voltage is V_{IH} or higher).

Table 27-18. RTP Pin Control 2 Register (RTPPC2) [offset = 3Ch] Field Descriptions (Continued)

Bit	Name	Value	Description
15-0	DATA[15:0]IN		RTPDATA[15:0] input. These bits reflect the state of the pins in all modes. Each bit represents a single pin. Writes to this bit have no effect.
			User and privilege mode (read):
		0	The pin is at logic low (0) (input voltage is V_{IL} or lower).
		1	The pin is at logic high (1) (input voltage is V_{IH} or higher).

27.5.11 RTP Pin Control 3 Register (RTPPC3)

The bits in this register define the state of the pins when configured in GIO mode as output pins. Once a pin is configured in functional mode (RTPPC0), changing the state of the corresponding bit in RTPPC3 will not affect the pin's state. Figure 27-19 and Table 27-19 illustrate this register.

Figure 27-19. RTP Pin Control 3 Register (RTPPC3) [offset = 40h]

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]OUT															
R/W-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-19. RTP Pin Control 3 Register (RTPPC3) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENAOUT		$\overline{\text{RTPENA}}$ output. This pin sets the output state of the $\overline{\text{RTPENA}}$ pin.
			User and privilege mode (read):
		0	The pin is configured to output logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Set pin to logic low (0) (output voltage is V_{OL} or lower).
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).

Table 27-19. RTP Pin Control 3 Register (RTPPC3) Field Descriptions (Continued)

Bit	Name	Value	Description
17	CLKOUT		RTPCLK output. This pin sets the output state of the RTPCLK pin.
			User and privilege mode (read):
		0	The pin is configured to output logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Set pin to logic low (0) (output voltage is V_{OL} or lower).
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).
16	SYNCOUT		RTPSYNC output. This pin sets the output state of the RTPSYNC pin.
			User and privilege mode (read):
		0	The pin is configured to output logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Set pin to logic low (0) (output voltage is V_{OL} or lower).
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).
15-0	DATA[15:0]OUT		RTPDATA[15:0] output. These bits set the output state of the RTPDATA[15:0] pins. Each bit represents a single pin.
			User and privilege mode (read):
		0	The pin is configured to output logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Set pin to logic low (0) (output voltage is V_{OL} or lower).
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).

27.5.12 RTP Pin Control 4 Register (RTPPC4)

This register provides the option to set pins to a logic 1 level without influencing the state of other pins. It eliminates the read-modify-write operation necessary with RTPPC3. Once the pin is configured in functional mode (RTPPC0), setting the corresponding bit to one in RTPPC4 will not affect the pin's state. Figure 27-20 and Table 27-20 illustrate this register.

Figure 27-20. RTP Pin Control 4 Register (RTPPC4) [offset = 44h]

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="13">Reserved</td><td>ENA-
SET</td><td>CLKSET</td><td>SYNC-
SET</td></tr>
<tr><td colspan="13">R-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td></tr>
</table>

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">DATA[15:0]SET</td></tr>
<tr><td colspan="16">R/W-0</td></tr>
</table>

R = Read, W = Write in all modes; -*n* = Value after reset

.

Table 27-20. RTP Pin Control 4 Register (RTPPC4) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENASET		Sets the output state of $\overline{\text{RTPENA}}$ pin to logic high. Value in the ENASET bit sets the data output control register bit to 1 regardless of the current value in the ENAOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).

Table 27-20. RTP Pin Control 4 Register (RTPPC4) Field Descriptions (Continued)

Bit	Name	Value	Description
17	CLKSET		Sets the output state of RTPCLK pin to logic high. Value in the CLKSET bit sets the data output control register bit to 1 regardless of the current value in the CLKOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).
16	SYNCSET		Sets output state of RTPSYNC pin logic high. Value in the SYNCSET bit sets the data output control register bit to 1 regardless of the current value in the SYNCOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).
15-0	DATA[15:0]SET		Sets output state of RTPDATA[15:0] pins to logic high. Value in the DATAxSET bit sets the data output control register bit to 1 regardless of the current value in the DATAxOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic high (1) (output voltage is V_{OH} or higher).

27.5.13 RTP Pin Control 5 Register (RTPPC5)

This register provides the option to set pins to a logic 0 level without influencing the state of other pins. It eliminates the read-modify-write operation necessary with RTPPC3. Once the pin is configured in functional mode (RTPPC0), setting the corresponding bit to one in RTPPC5 will not affect the pinstate. Figure 27-21 and Table 27-21 illustrate this register.

Figure 27-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENA-CLR	CLK-CLR	SYNC-CLR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]CLR															
R/W-0															

R = Read, WP = Write in privileged mode only; -*n* = Value after reset

.

Table 27-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h] Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENACLR		Sets the output state of $\overline{\text{RTPENA}}$ pin to logic high. Value in the ENASET bit sets the data output control register bit to 1 regardless of the current value in the ENAOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is V_{OL} or lower).

Table 27-21. RTP Pin Control 5 Register (RTPPC5) [offset = 48h] Field Descriptions (Continued)

Bit	Name	Value	Description
17	CLKCLR		Sets output state of RTPCLK pin to logic low. Value in the CLKCLR bit sets the data output control register bit to 0 regardless of the current value in the CLKOUT bit
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is V_{OL} or lower).
16	SYNCCLR		Sets output state of RTPSYNC pin logic low. Value in the SYNCCLR bit clears the data output control register bit to 0 regardless of the current value in the SYNCOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is V_{OL} or lower).
15-0	DATA[15:0]CLR		Sets output state of RTPDATA[15:0] pins to logic low. Value in the DATAxCLR bit clears the data output control register bit to 0 regardless of the current value in the DATAxOUT bit.
			User and privilege mode (read):
		0	The pin is configured to output a logic low (0) (output voltage is V_{OL} or lower).
		1	The pin is configured to output logic high (1) (output voltage is V_{OH} or higher).
			User and privilege mode (write):
		0	Writing a zero to this bit has no effect.
		1	Set pin to logic low (0) (output voltage is V_{OL} or lower).

27.5.14 RTP Pin Control 6 Register (RTPPC6)

These bits configure the pins in push-pull or open-drain functionality. If configured to be open-drain, the module only drives a logic low level on the pin. An external pull-up resistor needs to be connected to the pin to pull it high when the pin is in high-impedance mode. Figure 27-22 and Table 27-22 illustrate this register.

Figure 27-22. RTP Pin Control 6 Register (RTPPC6) [offset = 4Ch]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAPDR	CLK-PDR	SYN-CPDR
R-0													R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]PDR															
R/W-0															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-22. RTP Pin Control 6 Register (RTPPC6) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved		Reads return zeros and writes have no effect.
18	ENAPDR		$\overline{\text{RTPENA}}$ Open drain enable. This bit enables open drain functionality on the pin if it is configured as a GIO output (RTPPC0[18]=0; RTPPC1[18]=1). If the pin is configured as a functional pin (RTPPC0[18]=1), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain

Table 27-22. RTP Pin Control 6 Register (RTPPC6) Field Descriptions (Continued)

Bit	Name	Value	Description
17	CLKPDR		RTPCLK Open drain enable. This bit enables open drain functionality on the pin if it is configured as GIO output (RTPPC0[17]=0; RTPPC1[17]=1). If the pin is configured as functional pin (RTPPC0[17]=1), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain
16	SYNCPDR		RTPSYNC Open drain enable. This bit enables open drain functionality on the pin if it is configured as a GIO output (RTPPC0[16]=0; RTPPC1[16]=1). If pin is configured as functional pin (RTPPC0[16]=1), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain
15-0	DATA[15:0]PDR		RTPDATA[15:0] Open drain enable. These bits enable open drain functionality on the pins if they are configured as a GIO output (RTPPC0[15:0]=0; RTPPC1[15:0]=1). If the pins are configured as a functional pins (RTPPC0[15:0]=1), the open drain functionality is disabled.
			User and privilege mode (read):
		0	Pin behaves as normal push/pull pin
		1	Pin operates in open drain mode
			User and privilege mode (write):
		0	Configures the pin as push/pull
		1	Configures the pin as open drain

27.5.15 RTP Pin Control 7 Register (RTPPC7)

The bits in register control the pullup/down functionality of a pin. The internal pullup/down can be enabled or disabled by this register. The reset configuration of these bits is device implementation dependent. Please consult the device datasheet this information. Figure 27-23 and Table 27-23 illustrate this register.

Figure 27-23. RTP Pin Control 7 Register (RTPPC7) [offset = 50h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAP-DIS	CLKP-DIS	SYNCP-DIS
R-0													R/W-x	R/W-x	R/W-x

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]PDIS															
R/W-x															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-23. RTP Pin Control 7 Register (RTPPC7) Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENADIS		$\overline{\text{RTPENA}}$ Pull disable. This bit removes internal pullup/pulldown functionality from the pin when it is configured as an input pin (RTPPC1[18]=0).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
17	CLKPDIS		RTPCLK Pull disable. This bit removes the internal pullup/pulldown functionality from the pin when it is configured as an input pin (RTPPC1[17]=0).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality

Table 27-23. RTP Pin Control 7 Register (RTPPC7) Field Descriptions (Continued)

Bit	Name	Value	Description
16	SYNCPDIS		RTPSYNC Pull disable. Removes internal pullup/pulldown functionality from the pin when configured as an input pin (RTPPC1[16]=0).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality
15-0	DATA[15:0]PDIS		RTPDATA[15:0] Pull disable. Removes internal pullup/pulldown functionality from the pins when configured as input pins (RTPPC1[15:0]=0).
			User and privilege mode (read):
		0	Pullup/pulldown functionality enabled
		1	Pullup/pulldown functionality disabled
			User and privilege mode (write):
		0	Enables pullup/pulldown functionality
		1	Disables pullup/pulldown functionality

27.5.16 RTP Pin Control 8 Register (RTPPC8)

These bits control if the internal pullup or pulldown is configured on the input pin. A secondary function exists when the pull configuration is disabled and a pulldown is selected. This will disable the input buffer. Figure 27-24 and Table 27-24 illustrate this register.

Figure 27-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ENAPSEL	CLKPSEL	SYNCPSEL
R-0													R/W-1	R/W-1	R/W-1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]PSEL															
R/W-1															

R = Read, W = Write, P = Privileged, U = Undefined, C = Clear; -*n* = Value after reset

.

Table 27-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h] Field Descriptions

Bit	Name	Value	Description
31–19	Reserved		Reads return zeros and writes have no effect.
18	ENAPSEL		$\overline{\text{RTPENA}}$ Pull select. This bit configures pullup or pulldown functionality if RTPPC7[18]=0.
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality
17	CLKPSEL		RTPCLK Pull select. This bit configures pullup or pulldown functionality if RTPPC7[17]=0.
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality

Table 27-24. RTP Pin Control 8 Register (RTPPC8) [offset = 54h] Field Descriptions (Continued)

Bit	Name	Value	Description
16	SYNCPSEL		RTPSYNC Pull select. This bit configures pullup or pulldown functionality if RTPPC7[16]=0.
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality
15-0	DATA[15:0]PSEL		RTPDATA[15:0] Pull select. These bits configure pullup or pulldown functionality if RTPPC7[15:0]=0.
			User and privilege mode (read):
		0	Pulldown functionality enabled
		1	Pullup functionality enabled
			User and privilege mode (write):
		0	Enables pulldown functionality
		1	Enables pullup functionality

Note:
If the pullup/down is disabled in RTPPC7 and configured as pulldown in RTPPC8, then the input buffer is disabled

Chapter 28
SPNU489C - February 2012

Revision History

This revision history highlights the technical changes made to the device or the Technical Reference Manual (TRM).

Date	Additions, Deletions, And Modifications	Revision
March 2010		0
July 2010	Updated the TRM with the ‘Spec change’ items in errata sheet. BTS_AWM_3, No change, ADBUFFER register has already been updated. BTS_DCAN_14, Description in Table 16-4 and Figure 16-25 is correct. BTS_DCAN_19, Reset value in Figure 16-44 and Figure 16-45 is correct. Note in Page 893. BTS_DCAN_21, Add description in Table 16-16. BTS_DCAN_23, Add note in Table 16-19 bit 31. BTS_DMM_14, Updated in section 26.3.1.2, *Trace Mode Packet*, page 1829. BTS_ERAY.55 and BTS_FTU.13, Add note in page 1023. BTS_ESRAMW_35, No change, this document is correct. BTS_FTU_8, Add description to section 17.5.1.2, *Transfer Abort*, page 967. BTS_FWM_90, No change. This document is correct. BTS_FWM_135, No change. Bit DIS_PREEMPT is reserved for flash API. This document is correct. BTS_HTU_58, Added description to HTUEN bit in section 19.4.1, *Global Control Register (HTU GC)*, page 1411. BTS_LIN_39, Add note in page 477. BTS_LIN_47, Add note in page 475. BTS_LIN_54, Fixed in section 13.13, *Emulation Mode*, page 493. BTS_MIBSPI_89, Fixed in Table 14-50. BTS_MIBSPI_102, SRS (slew rate control) doesn't apply to Antares. BTS_MIBSPI_113, Add a note in page 704. BTS_MIBSPI_114, Fixed in Table 14-52. BTS_MIBSPI_117, Add a note in page 703. BTS_MIBSPI_124, Fixed in the datasheet. BTS_MIBSPI_125, Fixed in the datasheet. BTS_NHET_23 and 33, Update the pseudo code in Page 1382. BTS_RTP_30, No change, the reset value of RTPPC2 is correct. BTS_SSW_19, No change, the max VCO frequency is 500MHz in this document. BTS_SSW_20, No change, PLLCTL1.22 is reserved in this document. BTS_SSW_25, Change “$(t = 1/f_S)$” to “$(t = 1/f_S$~ $2/f_S)$” in page 245. BTS_SSW_26, No change, PLLCTL2.21 is reserved in this document. BTS_SYS_17, Add a note in Table 5-24.	A

Date	Additions, Deletions, And Modifications	Revision
July 2010	BTS_SYS_29, No change, The description of MSTCGSTAT.0 is correct. BTS_SYS_51, No change, The description of ECPCNTRL.24 is correct. BTS_SYS_59, Changed the description in Table 5-26. BTS_ANTARES_26, Not applicable to Antares. No EEPROM mode.	A
October 2010	Swapped the register at offset 0x64 and 0x68 in Figure 5-1. Added PLLCTL1, PLLCTL2 and DEVCR1 in Figure 5-1. Added section 5.1.43, *DEV Parity Control Register1 (DEVCR1)*, page 136. Changed INTVECT to TGINTVECT in MibSPIP section, consistent across the document. Changed MibSPIENA to MSPIENA in MibSPIP section, consistent across the document. Updated the offset address in Figure 14-39. Corrected title of Figure 18-82. Added timing diagram in section 15.3.5, *How to configure software or hardware trigger?*, page 728. Added section 22.3.2.4, *Data Trace Mode*, page 1641. Provided the based address for TCRAM registers in section 4.7, *Control and Status Registers*, page 54. Errata SC Update: BTS_NHET34, No change, MCMP pseudo code is correct. BTS_NHET35, Update the pseudo code in Page 1382. BTS_DCAN18, Add a note in Page 890. Add Figure 10-9 to address how to connect to 16 bit memory. Modify Figure 10.2.1 about the clock source. Modify the LBIST register base address to 0xFFFFE600 in Figure 9.9. Add SYSESR register into the Frame 1 System Control Registers Summary, Figure 5-1.	B

Date	Additions, Deletions, And Modifications	Revision
January 2011	BTS_SYS_74, Add SYSESR.0 description in Table 5-46. BTS_SYS87, No change, MPMENA description is correct. BTS_SYS90, No change, this device only has one ECLK pin. BTS_SYS92, No change, the value of SSDATA1[7:0] is correct in this document. BTS_SYS93, No change, VSWRST bit of SYSESR Register (Offset =0xE4) has already been removed. BTS_SYS94, No change, this device does not have internal VREG. BTS_SYS96, Updated section 5.1.39, *Bus Matrix Module Control Register2 (BMMCR2)*, page 129. BTS_SYS96, Updated section 5.1.23, *MSTC Global Status Register (MSTCGSTAT)*, page 107, MINIDONE bit field. Changed the reset value of LTCP to 0x3FFF in section 25.5.14, *ESM Low-Time Counter Preload Register (ESMLTCPR)*, page 1822 Replaced “to ensure a minimum low time of 100us” to “to ensure a minimum low time greater than 100us” in section 25.5.13, *ESM Low-Time Counter Register (ESMLTCR)*, page 1821, and section 25.5.14, *ESM Low-Time Counter Preload Register (ESMLTCPR)*, page 1822 BTS_ANTARES_25, Add description of IO pin reset status in the datasheet. Modified Table 6-27, removed the reserved bits. Fixed a few typos in Chapter 6 PBIST module: “groupsf“->“group“, “with out“->“without“, “Constant Loop Register“ -> “Constant Loop Count Register“, “Override“->“Over“, “Fail Status Count 0 Register“ ->“Fail Status Count Register 0“, “MSINENA“->“MSIENA“. Fixed a few typos in the Chapter 10 EMIF, section 10.2.5, *Asynchronous Controller and Interface*, page 344, “three major modes” -> “two major modes” Add base address 0xFFFFE800 in section 10.4, *Registers*, page 367. Changed CS[2:5] in EMIF to CS[0:3] to match the datasheet, add description for ASIZE=1. Add a note in page 14.	B
July 2011	Fixed the offset address of CRC_INTR and CRC_STATUS in Figure 22-9. Renamed "CRC Pattern Counter Preload Register 1" to "CRC Pattern Counter Preload Register 2" in section 22.5.22, *CRC Pattern Counter Preload Register2(CRC_PCOUNT_REG2)*, page 1711. Fixed Bit 9 description of section 8.6.3, *Flash Error Detection and Correction Control Register 1 (FEDACCTRL1 - 0xFFF87008)*, page 290. Change EM_D[7:0] to EM_D[15:0] in Table 10-1. Change register MSTFAIL to reserved in Figure 5-1. Fixed description of section 5.1.22, *Memory Self-Test/Initialization Enable Register (MSIENA)*, page 105. BTS_AWM_20, not applicable. BTS_NHET_45, not applicable. BTS_NHET_46, Change the syntax of ADM32, section 18.7.3.4, *ADM32 (Add Move 32)*, page 1332. BTS_NHET_49, Add more description about maximum frequency for input signals in section 18.4.5.7, *Edge Detection*, page 1260. BTS_SSW_17, Add note and workaround in Table 5-47. BTS_DMM_14, Add more description about packet error generation in section 26.3.1.3, *Direct Data Mode Packet*, page 1829 and section 26.3.3.2, *Packet Error*, page 1832. BTS_ERAY_53, User guide is correct. No modification is needed. BTS_HTU_53, Add note in Table 19-27 on the set conditions for Buffer Full Interrupt Flag.	C

Date	Additions, Deletions, And Modifications	Revision
Feb 2012	Minor changes to Figure 20-18. Changed 9C-A0 to 9C-B0. Fixed the register name from offset 0xEC to 0x104. Minor changes in Table 19-42. HTU register description for bits TMBB in IHADDRCT register was shown as 19:18 in previous version, but should be 17:16. Minor changes in section 4.6, *Auto Initialization Support*, page 53. Changed the ECC initialization value from 0xFC to 0x0C. Fixed the descriptions of CHx_CCITENS field in Table 22-7. Fixed the descriptions of CHx_CCITENR field in Table 22-8. Fixed the title of Table 22-8. Fixed the field name in Table 22-25. Fixed the offset address in Figure 22-9. Switch the R1 and R2 value in Figure 15-16 and Figure 15-19. Correct spelling errors. Change the title from “TMS570LS Series” to “TMS570LS20x/10x Safety MCUs”.	C

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
		TI E2E Community Home Page	e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265