SmartFusion Microcontroller Subsystem User's Guide

Microsemi

Table of Contents

1 – ARM Cortex-M3 Microcontroller

The ARM® Cortex-M3 microcontroller is a low-power processor that features low gate count, low and predictable interrupt latency, and low-cost debug. It is intended for deeply embedded applications that require fast interrupt response features. The processor implements the ARMv7-M architecture and is depicted in its entirety in Figure 1-1. Microsemi SoC Products Group SmartFusion® customizable system-on-chip (cSoC) devices use the R1P1 version of the Cortex-M3 core. The following manuals, available from the ARM Infocenter, are recommended reading:

- *Cortex-M3 Technical Reference Manual*
- *ARMv7-M Architecture Reference Manual*
- *ARMv7-M Architecture Application Level Reference Manual*

The Definitive Guide to the ARM Cortex-M3 by Joseph Yiu is recommended as additional reading (ISBN: 978-0-7506-8534-4).

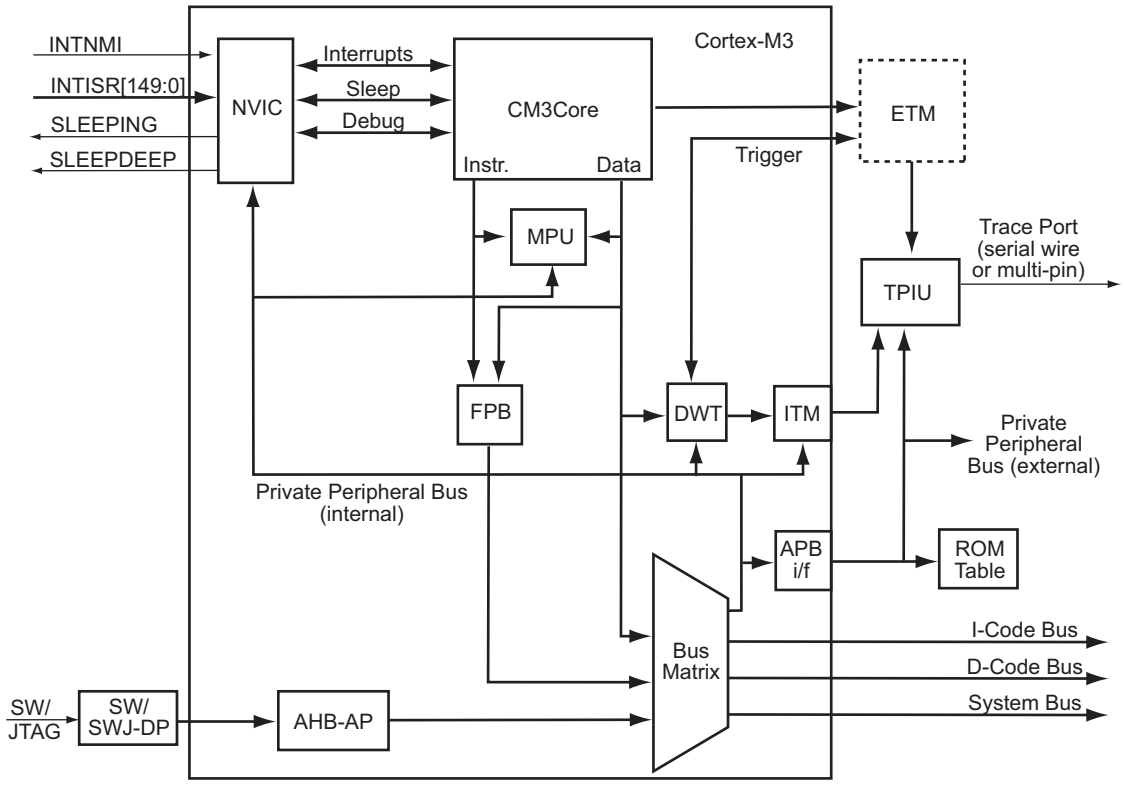


Figure 1-1 • **Cortex-M3 R1P1 Block Diagram**

Manufacturers of Cortex-M3 integrated circuits are permitted some latitude in configuring a particular implementation of the Cortex-M3 processor delivered by ARM. These are the implementation specifics in the SmartFusion device:

- Number of interrupts set to 150 (151 including NMI)
- 32 levels of interrupt priority
- Memory Protection Unit (MPU)
- The Data Watchpoint and Trace (DWT) unit is configured to include data matching.
- The Embedded Trace Macrocell (ETM) is not included.

- The debug port is implemented using a Serial Wire JTAG Debug Port (SWJ-DP) rather than a Serial Wire Debug Port (SW-DP). This enables either the JTAG or SW protocol to be used for debugging. The SWJ-DP defaults to JTAG mode at power-up and can be switched to SW by applying a specific sequence to the debug pins.

The Trace Port Interface Unit (TPIU) is configured to support Instrumentation Trace Macrocell (ITM) debug trace only, and not Embedded Trace Macrocell (ETM) debug trace. The optional ETM is not included. Also, Serial Wire mode is used for the TPIU output data and this is overlaid on the JTAG TDO port (Figure 1-2). One implication of this is that Instrumentation Trace cannot be used along with JTAG-based debugging. SW debugging and ITM can be used together. SW debugging operates at 98 KHz for the A2F200 device only. A2F060 and A2F500 allow debug at up to 10 MHz, as limited by the debugger.

- The ROM table has not been modified and matches the description given in the *Cortex-M3 Technical Reference Manual*.
- The deployment of a Cortex-M3 microcontroller in a SmartFusion cSoC combines the I-Code and D-Code buses into a single shared code bus. This multiplexing occurs within the AHB bus matrix. The Cortex-M3 microcontroller internally arbitrates between these two buses to determine which one obtains ownership of the code bus at any given time.

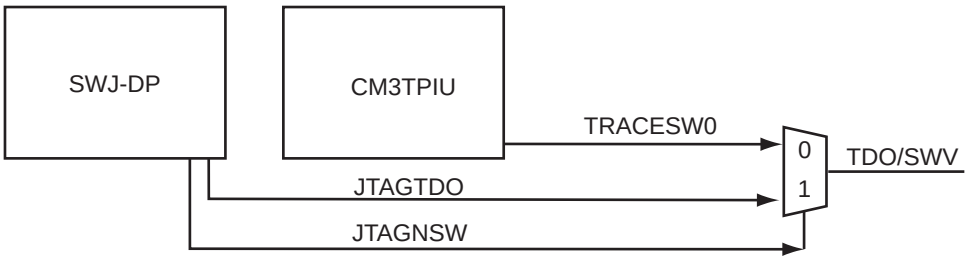


Figure 1-2 • **SWJ-DP / Single Wire Viewer**

Cortex-M3 SysTick Timer

The SysTick Timer is used to generate a periodic interrupt to the Cortex-M3 microcontroller. It is essentially a 24-bit down counter.

The Cortex-M3 microcontroller has four internal registers related to the SysTick timer, described briefly in Table 1-1.

Table 1-1 • **SysTick Control Register Embedded in NVIC Module**

Register Name	Address	R/W	Reset Value	Description
SysTick Control and Status	0xE000E010	R/W	0x0	Basic control of SysTick, including enable, clock source, interrupt, or poll
SysTick Reload Value	0xE000E014	R/W	Unpredictable	Value to load in Current Value register when 0 is reached
SysTick Current Value	0xE000E018	R/W	Unpredictable	The current value of the count down
SysTick Calibration Value	0xE000E01C	R	STCALIB	Contains the number of ticks to generate a 10 ms interval.

The SYSTICK_CR, located in the SYSREG address space at address 0xE0042038, is used in conjunction with the SysTick control registers embedded within the NVIC module to control the behavior of the SysTick timer. Individual bits of the SYSTICK_CR register are described in Table 1-3. The SysTick counter in the Cortex-M3 microcontroller is clocked by the free-running clock FCLK, and it can count either the free-running clock itself, or the cycles of the timing reference signal STCLK. The SysTick timer uses FCLK if NOREF is set to 1, and uses STCLK if NOREF is set to 0. STCLK is divided down from

FCLK based on how you program the STCLK_DIVISOR field. If you run the SysTick Timer using STCLK, the remaining fields in SYSTICK_CR must be programmed properly. FCLK must always be greater than or equal to 2.5 × STCLK, even when the Cortex-M3 microcontroller is in sleep mode.

To generate an exact 10 ms tick, for example (applicable to –1 speed device), use these steps:

1. Program the MSS_CCC to provide a 100 MHz clock to the MSS and, hence, to the Cortex-M3 microcontroller. FCLK = 100 MHz.
2. Program STCLK_DIVISOR to 0x03 to divide by 32. STCLK is now 100 MHz divided by 32, or 3.125 MHz.
3. Set NOREF to 0, indicating that a reference clock is provided. With a 3.125 MHz reference clock, the counter must be reloaded with a value of 31,250, which is 0x7A12. This value is loaded into the TENMS field of the SYSTICK_CR register.
4. Set the SKEW bit to 0, indicating there is an exact 10 ms period.
5. You can verify the settings programmed into the SYSTICK_CR register by reading the SysTick Calibration Value (STCVR) register, located at 0xE000E01C.

Refer to the ARM Infocenter for more information. An ARM Knowledge Article with further detail on STCLK is posted in the ARM Infocenter at the time of this writing.

Table 1-2 • **SYSTICK_CR Map**

Register Name	Address	R/W	Reset Value	Description
SYSTICK_CR	0xE0042038	R/W	0x32000000	Provides firmware control of the STCALIB[25:0] pins of Cortex-M3 microcontroller.

Table 1-3 • **SYSTICK_CR**

Bit Number	Name	R/W	Reset Value	Description
29:28	STCLK_DIVISOR	R/W	0b11	See Table 1-4.
27:26	Reserved			Do not use.
25	NOREF	R/W	1	1 = Reference clock is not provided.
24	SKEW	R/W	0	1 = The calibration value is not exactly 10 ms because of clock frequency.
23:0	TENMS	R/W	0	This value is the Reload value to use for 10 ms timing. Depending on the value of SKEW, this might be exactly 10 ms or might be the closest value.

The STCLK_DIVISOR field of SYSTICK_CR is used to divide the FCLK by 4, 8, 16, or 32 (Table 1-4). The resultant clock is used to provide the STCLK input to the SysTick Timer of the Cortex-M3 microcontroller. The reset state of STCLK_DIVISOR is FCLK divided by 32. FCLK must always be greater than or equal to 2.5 × STCLK.

Table 1-4 • **STCLK_DIVISOR Definition**

STCLK_DIVISOR		FCLK Divided By
Bit 29	Bit 28	
0	0	4
0	1	8
1	0	16
1	1	32

The NOREF, SKEW, and TENMS fields of the SYSTICK_CR are mapped to the STCALIB[25:0] input pins of the Cortex-M3 microcontroller. Within the NVIC module of the Cortex-M3 microcontroller, you have read access to the STCALIB pins through the SysTick Calibration Value (STCVR) register, located at address 0xE000E01C. The SYSTICK_CR at address 0xE0042038 can be read and written by user firmware. The NOREF, SKEW, and TENMS fields in SYSTICK_CR map directly to the same fields in the SysTick Calibration Value register, located at 0xE000E01C, although at different bit locations. Specifically, NOREF of SYSTICK_CR (bit 25) is mapped to NOREF of STCVR (bit 31) and SKEW of SYSTICK_CR (bit 24) is mapped to SKEW of STCVR (bit 30).

An application note describing the configuration of the SysTick Timer is available at the ARM Infocenter at the time of this writing.

Interrupts

Table 1-5 lists the interrupt numbers (corresponding to the NVIC input pins of the Cortex-M3 microcontroller), their sources, and which functions assert the interrupt for the SmartFusion family of cSoCs. Details for each specific interrupt are located in the relevant section of the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet where the interrupt is sourced. A description of exceptions 0–15 can be found in the *Cortex-M3 Technical Reference Manual*. The Watchdog Timer interrupt is mapped to the Non-Maskable interrupt of the NVIC. All other SmartFusion interrupts are mapped to the external interrupt pins of the Cortex-M3 microcontroller (NVIC), starting at INTISR[0].

***Table 1-5* • SmartFusion Interrupt Sources**

Cortex-M3 NVIC Input	IRQ Label	IRQ Source
NMI	WDOGTIMEOUT_IRQ	WATCHDOG
INTISR[0]	WDOGWAKEUP_IRQ	WATCHDOG
INTISR[1]	BROWNOUT1_5V_IRQ	VR/PSM
INTISR[2]	BROWNOUT3_3V_IRQ	VR/PSM
INTISR[3]	RTCMATCHEVENT_IRQ	RTC
INTISR[4]	PU_N_IRQ	RTC
INTISR[5]	EMAC_IRQ	Ethernet MAC
INTISR[6]	M3_IAP_IRQ	IAP
INTISR[7]	ENVM_0_IRQ	ENVM Controller
INTISR[8]	ENVM_1_IRQ	ENVM Controller
INTISR[9]	DMA_IRQ	Peripheral DMA
INTISR[10]	UART_0_IRQ	UART_0
INTISR[11]	UART_1_IRQ	UART_1
INTISR[12]	SPI_0_IRQ	SPI_0
INTISR[13]	SPI_1_IRQ	SPI_1
INTISR[14]	I2C_0_IRQ	I2C_0
INTISR[15]	I2C_0_SMBALERT_IRQ	I2C_0
INTISR[16]	I2C_0_SMBSUS_IRQ	I2C_0
INTISR[17]	I2C_1_IRQ	I2C_1
INTISR[18]	I2C_1_SMBALERT_IRQ	I2C_1
INTISR[19]	I2C_1_SMBSUS_IRQ	I2C_1
INTISR[20]	TIMER_1_IRQ	TIMER

***Table 1-5* • SmartFusion Interrupt Sources (continued)**

INTISR[21]	TIMER_2_IRQ	TIMER
INTISR[22]	PLLLOCK_IRQ	MSS_CCC
INTISR[23]	PLLLOCKLOST_IRQ	MSS_CCC
INTISR[24]	ABM_ERROR_IRQ	AHB BUS MATRIX
INTISR[25]	Reserved	Reserved
INTISR[26]	Reserved	Reserved
INTISR[27]	Reserved	Reserved
INTISR[28]	Reserved	Reserved
INTISR[29]	Reserved	Reserved
INTISR[30]	Reserved	Reserved
INTISR[31]	FAB_IRQ	FABRIC INTERFACE
INTISR[32]	GPIO_0_IRQ	GPIO
INTISR[33]	GPIO_1_IRQ	GPIO
INTISR[34]	GPIO_2_IRQ	GPIO
INTISR[35]	GPIO_3_IRQ	GPIO
INTISR[36]	GPIO_4_IRQ	GPIO
INTISR[37]	GPIO_5_IRQ	GPIO
INTISR[38]	GPIO_6_IRQ	GPIO
INTISR[39]	GPIO_7_IRQ	GPIO
INTISR[40]	GPIO_8_IRQ	GPIO
INTISR[41]	GPIO_9_IRQ	GPIO
INTISR[42]	GPIO_10_IRQ	GPIO
INTISR[43]	GPIO_11_IRQ	GPIO
INTISR[44]	GPIO_12_IRQ	GPIO
INTISR[45]	GPIO_13_IRQ	GPIO
INTISR[46]	GPIO_14_IRQ	GPIO
INTISR[47]	GPIO_15_IRQ	GPIO
INTISR[48]	GPIO_16_IRQ	GPIO
INTISR[49]	GPIO_17_IRQ	GPIO
INTISR[50]	GPIO_18_IRQ	GPIO
INTISR[51]	GPIO_19_IRQ	GPIO
INTISR[52]	GPIO_20_IRQ	GPIO
INTISR[53]	GPIO_21_IRQ	GPIO
INTISR[54]	GPIO_22_IRQ	GPIO
INTISR[55]	GPIO_23_IRQ	GPIO
INTISR[56]	GPIO_24_IRQ	GPIO
INTISR[57]	GPIO_25_IRQ	GPIO

***Table 1-5* • SmartFusion Interrupt Sources (continued)**

INTISR[58]	GPIO_26_IRQ	GPIO
INTISR[59]	GPIO_27_IRQ	GPIO
INTISR[60]	GPIO_28_IRQ	GPIO
INTISR[61]	GPIO_29_IRQ	GPIO
INTISR[62]	GPIO_30_IRQ	GPIO
INTISR[63]	GPIO_31_IRQ	GPIO
INTISR[64]	ACE_PC0_FLAG0_IRQ	ACE[0]
INTISR[65]	ACE_PC0_FLAG1_IRQ	ACE[1]
INTISR[66]	ACE_PC0_FLAG2_IRQ	ACE[2]
INTISR[67]	ACE_PC0_FLAG3_IRQ	ACE[3]
INTISR[68]	ACE_PC1_FLAG0_IRQ	ACE[4]
INTISR[69]	ACE_PC1_FLAG1_IRQ	ACE[5]
INTISR[70]	ACE_PC1_FLAG2_IRQ	ACE[6]
INTISR[71]	ACE_PC1_FLAG3_IRQ	ACE[7]
INTISR[72]	ACE_PC2_FLAG0_IRQ	ACE[8]
INTISR[73]	ACE_PC2_FLAG1_IRQ	ACE[9]
INTISR[74]	ACE_PC2_FLAG2_IRQ	ACE[10]
INTISR[75]	ACE_PC2_FLAG3_IRQ	ACE[11]
INTISR[76]	ACE_ADC0_DATAVALID_IRQ	ACE[12]
INTISR[77]	ACE_ADC1_DATAVALID_IRQ	ACE[13]
INTISR[78]	ACE_ADC2_DATAVALID_IRQ	ACE[14]
INTISR[79]	ACE_ADC0_CALDONE_IRQ	ACE[15]
INTISR[80]	ACE_ADC1_CALDONE_IRQ	ACE[16]
INTISR[81]	ACE_ADC2_CALDONE_IRQ	ACE[17]
INTISR[82]	ACE_ADC0_CALSTART_IRQ	ACE[18]
INTISR[83]	ACE_ADC1_CALSTART_IRQ	ACE[19]
INTISR[84]	ACE_ADC2_CALSTART_IRQ	ACE[20]
INTISR[85]	ACE_COMP0_FALL_IRQ	ACE[21]
INTISR[86]	ACE_COMP1_FALL_IRQ	ACE[22]
INTISR[87]	ACE_COMP2_FALL_IRQ	ACE[23]
INTISR[88]	ACE_COMP3_FALL_IRQ	ACE[24]
INTISR[89]	ACE_COMP4_FALL_IRQ	ACE[25]
INTISR[90]	ACE_COMP5_FALL_IRQ	ACE[26]
INTISR[91]	ACE_COMP6_FALL_IRQ	ACE[27]
INTISR[92]	ACE_COMP7_FALL_IRQ	ACE[28]
INTISR[93]	ACE_COMP8_FALL_IRQ	ACE[29]
INTISR[94]	ACE_COMP9_FALL_IRQ	ACE[30]

***Table 1-5* • SmartFusion Interrupt Sources (continued)**

INTISR[95]	ACE_COMP10_FALL_IRQ	ACE[31]
INTISR[96]	ACE_COMP11_FALL_IRQ	ACE[32]
INTISR[97]	ACE_COMP0_RISE_IRQ	ACE[33]
INTISR[98]	ACE_COMP1_RISE_IRQ	ACE[34]
INTISR[99]	ACE_COMP2_RISE_IRQ	ACE[35]
INTISR[100]	ACE_COMP3_RISE_IRQ	ACE[36]
INTISR[101]	ACE_COMP4_RISE_IRQ	ACE[37]
INTISR[102]	ACE_COMP5_RISE_IRQ	ACE[38]
INTISR[103]	ACE_COMP6_RISE_IRQ	ACE[39]
INTISR[104]	ACE_COMP7_RISE_IRQ	ACE[40]
INTISR[105]	ACE_COMP8_RISE_IRQ	ACE[41]
INTISR[106]	ACE_COMP9_RISE_IRQ	ACE[42]
INTISR[107]	ACE_COMP10_RISE_IRQ	ACE[43]
INTISR[108]	ACE_COMP11_RISE_IRQ	ACE[44]
INTISR[109]	ACE_ADC0_FIFOFULL_IRQ	ACE[45]
INTISR[110]	ACE_ADC0_FIFOAFULL_IRQ	ACE[46]
INTISR[111]	ACE_ADC0_FIFOEMPTY_IRQ	ACE[47]
INTISR[112]	ACE_ADC1_FIFOFULL_IRQ	ACE[48]
INTISR[113]	ACE_ADC1_FIFOAFULL_IRQ	ACE[49]
INTISR[114]	ACE_ADC1_FIFOEMPTY_IRQ	ACE[50]
INTISR[115]	ACE_ADC2_FIFOFULL_IRQ	ACE[51]
INTISR[116]	ACE_ADC2_FIFOAFULL_IRQ	ACE[52]
INTISR[117]	ACE_ADC2_FIFOEMPTY_IRQ	ACE[53]
INTISR[118]	ACE_PPE_FLAG0_IRQ	ACE[54]
INTISR[119]	ACE_PPE_FLAG1_IRQ	ACE[55]
INTISR[120]	ACE_PPE_FLAG2_IRQ	ACE[56]
INTISR[121]	ACE_PPE_FLAG3_IRQ	ACE[57]
INTISR[122]	ACE_PPE_FLAG4_IRQ	ACE[58]
INTISR[123]	ACE_PPE_FLAG5_IRQ	ACE[59]
INTISR[124]	ACE_PPE_FLAG6_IRQ	ACE[60]
INTISR[125]	ACE_PPE_FLAG7_IRQ	ACE[61]
INTISR[126]	ACE_PPE_FLAG8_IRQ	ACE[62]
INTISR[127]	ACE_PPE_FLAG9_IRQ	ACE[63]
INTISR[128]	ACE_PPE_FLAG10_IRQ	ACE[64]
INTISR[129]	ACE_PPE_FLAG11_IRQ	ACE[65]
INTISR[130]	ACE_PPE_FLAG12_IRQ	ACE[66]
INTISR[131]	ACE_PPE_FLAG13_IRQ	ACE[67]

***Table 1-5* • SmartFusion Interrupt Sources (continued)**

INTISR[132]	ACE_PPE_FLAG14_IRQ	ACE[68]
INTISR[133]	ACE_PPE_FLAG15_IRQ	ACE[69]
INTISR[134]	ACE_PPE_FLAG16_IRQ	ACE[70]
INTISR[135]	ACE_PPE_FLAG17_IRQ	ACE[71]
INTISR[136]	ACE_PPE_FLAG18_IRQ	ACE[72]
INTISR[137]	ACE_PPE_FLAG19_IRQ	ACE[73]
INTISR[138]	ACE_PPE_FLAG20_IRQ	ACE[74]
INTISR[139]	ACE_PPE_FLAG21_IRQ	ACE[75]
INTISR[140]	ACE_PPE_FLAG22_IRQ	ACE[76]
INTISR[141]	ACE_PPE_FLAG23_IRQ	ACE[77]
INTISR[142]	ACE_PPE_FLAG24_IRQ	ACE[78]
INTISR[143]	ACE_PPE_FLAG25_IRQ	ACE[79]
INTISR[144]	ACE_PPE_FLAG26_IRQ	ACE[80]
INTISR[145]	ACE_PPE_FLAG27_IRQ	ACE[81]
INTISR[146]	ACE_PPE_FLAG28_IRQ	ACE[82]
INTISR[147]	ACE_PPE_FLAG29_IRQ	ACE[83]
INTISR[148]	ACE_PPE_FLAG30_IRQ	ACE[84]
INTISR[149]	ACE_PPE_FLAG31_IRQ	ACE[85]

2 – AHB Bus Matrix

The AHB bus matrix (ABM) is a multi-layer AHB matrix. It is not a full crossbar switch, but a customized subset of a full switch. It works purely as an AHB-Lite (AHBL) matrix. The SmartFusion AHB Matrix has five masters and eight direct slaves, as depicted in Figure 2-1. One master is permitted to access a slave at the same time another master is accessing a different slave. If more than one master is attempting to access the same slave simultaneously, then arbitration for that slave is performed. Arbitration is programmable by the user and is either pure round robin or a weighted round robin where certain masters have favor over others. One master is elected as the winner, while the other masters are held up temporarily. Theoretical maximum bus bandwidth through the AHB bus matrix is 16 Gbps. This assumes that the five masters are communicating with five different slaves at the maximum clock rate of 100 MHz.

Functional Description

Figure 2-1 depicts the connectivity of masters and slaves in the ABM. Label nomenclature such as MM0 and MS0 refers to a mirrored master and mirrored slave. A mirrored master port in the matrix connects directly to an AHB master; it has the same set of signals, but the direction of the signals is described relative to the other end of the connection. A mirrored slave port in the matrix connects directly to an AHB slave.

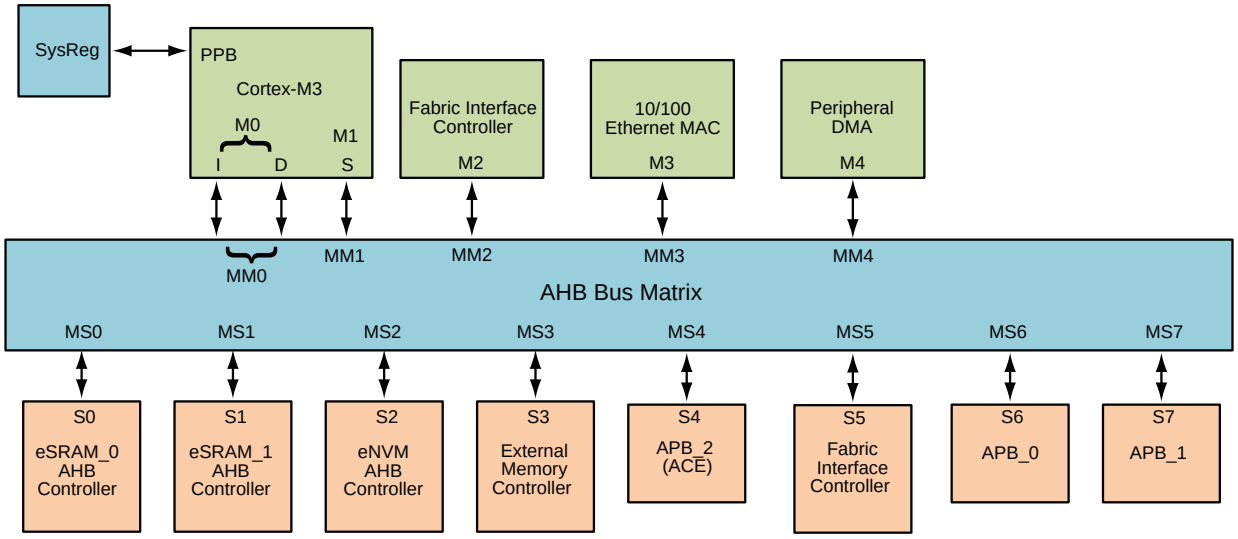


Figure 2-1 • **AHB Bus Matrix Masters and Slaves**

Only a subset of the full set of theoretical paths is implemented within the AHB bus matrix. Furthermore, the I-Code and D-Code buses of the ARM Cortex-M3 microcontroller are multiplexed within the AHB bus matrix, so they actually constitute one combined master between them. The Cortex-M3 microcontroller is configured to avoid activating both buses together.

The connections available in the AHB bus matrix are shown in Table 2-1.

***Table 2-1* • AHB Bus Matrix Connectivity**

	eSRAM_0 S0	eSRAM_1 S1	eNVM S2	EMC S3	APB_2 S4	Fabric Slave S5	APB_0 S6	APB_1 S7
Cortex-M3 I-Code/D-Code M0	R (I-Code) R/W (D-Code)	R (I-Code) R/W (D-Code)	R*					
Cortex-M3 System M1	R/W	R/W	R/W*	R/W	R/W	R/W	R/W	R/W
Fabric Master M2	R/W	R/W	R/W*	R/W	R/W	R/W	R/W	R/W
Ethernet MAC M3	R/W	R/W		R/W		R/W		
Peripheral DMA M4	R/W	R/W	R*	R/W	R/W	R/W	R/W	R/W

*Note: *Users must exercise caution when commanding the eNVM to program or erase data. Other masters in the system may not be aware that the eNVM is unavailable. Therefore users should use some form of software semaphore to control access.*

By default, non-Cortex-M3 ports are disabled on power-up. Users must enable each port by setting the appropriate bits in the AHB_MATRIX_CR register (refer to Table 2-12 on page 31. The Cortex-M3 microcontroller is the only master in the system that can enable other masters, since the control registers that enable masters reside on the Private Peripheral Bus of the Cortex-M3 microcontroller. Access errors in the AHB bus matrix set the appropriate bit in the COM_ERRORSTATUS field of the MSS_SR register. The ABM_ERROR_IRQ signal is also asserted and an error can be trapped if IRQ24 is enabled in the NVIC. IRQ24 corresponds to bit location 24 in the 32-bit word at address location 0xE000E100. The following types of errors can occur:

1. Write by an enabled master to a slave that is not R/W
2. Write by a disabled master to any location
3. A read by an enabled master to any slave that is not R or R/W
4. A read by a disabled master to any location

Reads to a non-enabled slave or unimplemented address space return undefined values. Write errors do not propagate beyond the AHB bus matrix, that is, the ABM consumes the write error.

The user has the option of restricting access to eNVM from a fabric master by programming the appropriate registers in FAB_PROT_SIZE_CR and FAB_PROT_BASE_CR. If a region of memory in the eNVM is protected and a fabric master attempts to read or write to it, the COM_ERRORSTATUS field of the MSS_SR register is updated to reflect the appropriate error and the ABM_ERROR_IRQ (IRQ24) signal is asserted.

Arbitration

Each of the slave interfaces contains an arbiter. The arbiter has two modes of operation: round robin and weighted round robin. The arbitration scheme selected is applied to all slave interfaces.

Round Robin Arbitration

This is the default arbitration mode. As depicted in Figure 2-2 in this mode, the arbitration scheme for each slave port is identical. Each master accessing a slave has equal priority on a round robin basis. However, if a locked transaction occurs, the master issuing the lock maintains ownership of the slave until the locked transaction completes. Clearing bit COM_WEIGHTEDMODE in the AHB_MATRIX_CR sets arbitration to round robin.

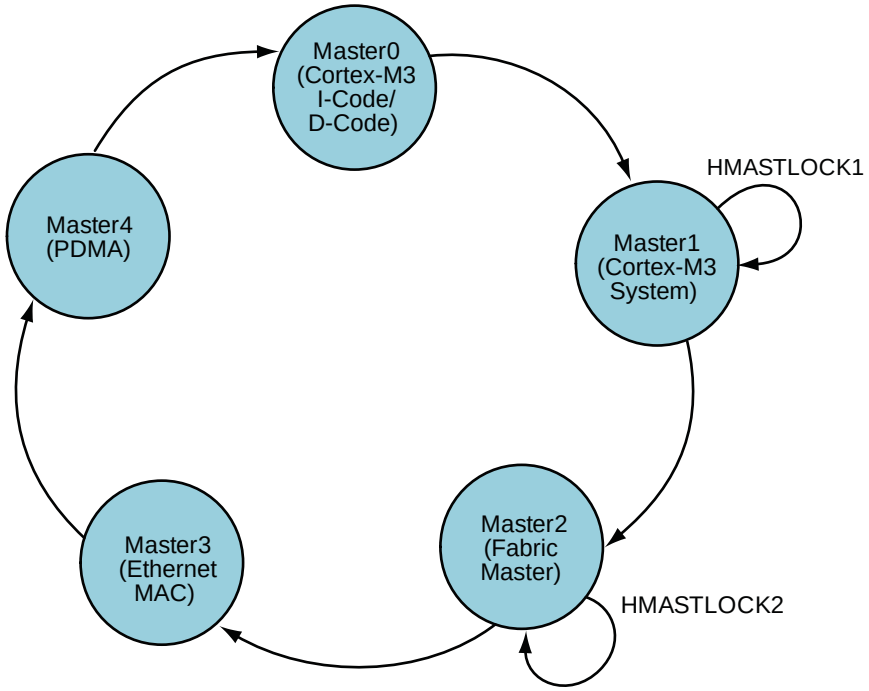


Figure 2-2 • **Round Robin Arbitration**

The pure round robin scheme has the advantage of low latency. So, for example, the Cortex-M3 microcontroller can respond quickly to service a high-priority interrupt, even if the MAC is performing a long AHB burst to the same slave required by the Cortex-M3 microcontroller. This is at the expense of not taking full advantage of the slave bandwidth achievable via burst accesses, in some cases.

Weighted Round Robin Arbitration

The user can configure arbitration by setting the bit COM_WEIGHTEDMODE in the AHB_MATRIX_CR to operate as weighted round robin. In this mode, the slave arbiter for every slave operates on a round robin basis, with three of the master interfaces (Cortex-M3 I-Code/D-Code interface, Cortex-M3 system interface, and the Ethernet MAC) having a maximum of eight consecutive access opportunities to the slave in each round of arbitration.

This scheme is illustrated in Figure 2-3.

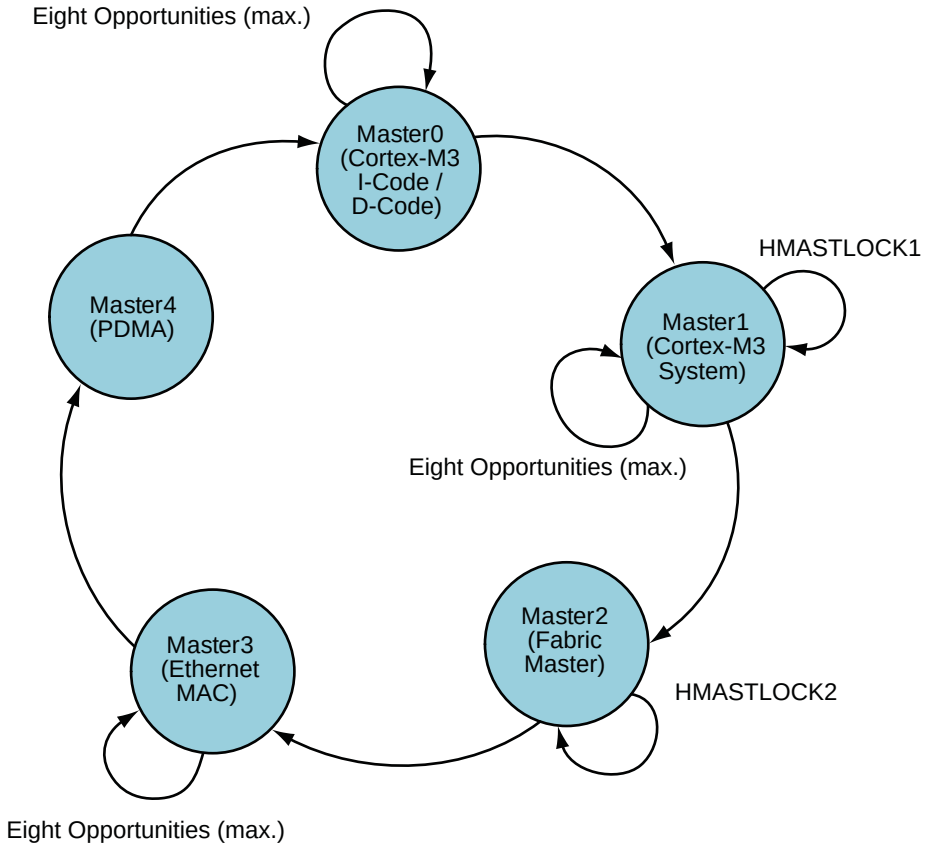


Figure 2-3 • **Weighted Round Robin Arbitration**

Weighted round robin arbitration allows more efficient usage of slave bandwidth in the cases where the slaves have a penalty when transitioning from one master to another. For example, in situations where both the Ethernet MAC and Cortex-M3 I-Code/D-Code interfaces are performing write and read AHB bursts to eSRAM, this scheme groups together a maximum of eight Ethernet MAC accesses followed by a maximum of eight Cortex-M3 accesses (even if AHB bursts of greater than eight transfers are in progress from the master's point of view). Due to the fact that the eSRAM AHB controller inserts an idle cycle every time there is a write followed by a read, enabling weighted round robin can increase the effective eSRAM bandwidth during this time from 66% to 94% of the theoretical maximum. If a sequence of locked transfers is in progress, then the locked master remains selected by the slave arbiter until the lock sequence is finished, regardless of the number of transfers (which could be more than eight).

Weighted round robin arbitration would also be useful in situations where more than one master is accessing eNVM, as it allows each master to access multiple prefetched data words in the eNVM buffer instead of repeatedly filling the buffer with one word. Refer to the "Embedded Nonvolatile Memory (eNVM) Controller" section on page 49 for details.

This arbitration mode has slightly longer potential latencies than pure round robin mode. For example, an urgent interrupt to the Cortex-M3 microcontroller could require servicing that involves accesses to a slave while the MAC is using that slave. However, by limiting the bursts to eight at the arbitration level, regardless of AHB burst size, the latency can be kept at a low value.

It is possible to switch between the two arbitration modes dynamically.

System Memory Map

The AHB bus matrix is responsible for implementing the address decoding of all masters to all slaves, so it defines the system memory map. Figure 2-4 on page 20 depicts the default system memory map for the A2F200 device.

Unimplemented Address Space

The AHB bus matrix performs address decoding based on the memory map defined in Figure 2-4 on page 20 and Figure 2-6 on page 22, to decide which slave, if any, is being addressed. Any access to memory space outside of these regions is considered unimplemented from the point of view of the AHB bus matrix and results in the assertion of a COM_ERRORSTATUS register bit and the interrupt COM_ERRORINTERRUPT to the Cortex-M3 microcontroller, as well as the assertion of HRESP by the AHB bus matrix to the master—which could be in the FPGA fabric.

If any master attempts a write access to unimplemented address space, the AHB bus matrix completes the handshake to the master, with an HRESP error indication. No write occurs to any slave.

If any master attempts a read access from unimplemented address space, the AHB bus matrix completes the handshake to the master, with an HRESP error indication. Undefined data is returned in this case.

Within individual slave memory regions, there may be further memory areas that are unimplemented. Depending on the slave, accesses may be aliased within these areas or not. Firmware should not perform writes to these locations because the aliasing may cause a write to another location within the slave. Data read from these intra-slave unimplemented regions may be undefined. In the case of the external memory controller, some of these accesses may result in HRESP assertion by the memory controller. This occurs when attempting to access a location corresponding to an external memory that is not present at that address.

Burst Support

The AHB bus matrix handshakes correctly with masters performing AHB bursts to any slave. However, it does not pass the transactions through to the slaves as bursts. Instead, the AHB bus matrix converts the burst accesses into single-cycle accesses of the type NONSEQ. This simplifies the design of the slaves (which can exist in the FPGA fabric), since they do not need to support AHB bursts. It also allows the system designer to avoid having long latencies incurred by bursts of indeterminate length (such as those from the FPGA fabric). The AHB bus matrix does not connect to the HBURST bus of any master or slave.

Locked Transactions

The AHB bus matrix supports implementation of locked transactions for accesses by the Cortex-M3 microcontroller to the memory controllers (eNVM AHB controller, eSRAM AHB controller, and external memory controller), by monitoring the HMASTLOCK signal. The only slave to which HMASTLOCK is actually passed is the fabric slave, because a circuit within the FPGA fabric may need to perform further locking. For a more detailed description of HMASTLOCK, refer to the ARM AMBA bus specification at the ARM website.

Memory Map

In the memory map shown in Figure 2-4 on page 20, the eNVM is mapped into the Cortex-M3 system space. This allows other masters in AHB bus matrix to read from and write to eNVM. The capability exists to map a physical portion of eNVM into the address space occupied at 0x0, which is the Cortex-M3 code space. This essentially creates a virtual view of the eNVM at address 0x0, allowing users the option of

storing multiple application images in eNVM and mapping the newest or desired version to address 0x0 in the Cortex-M3 code space.

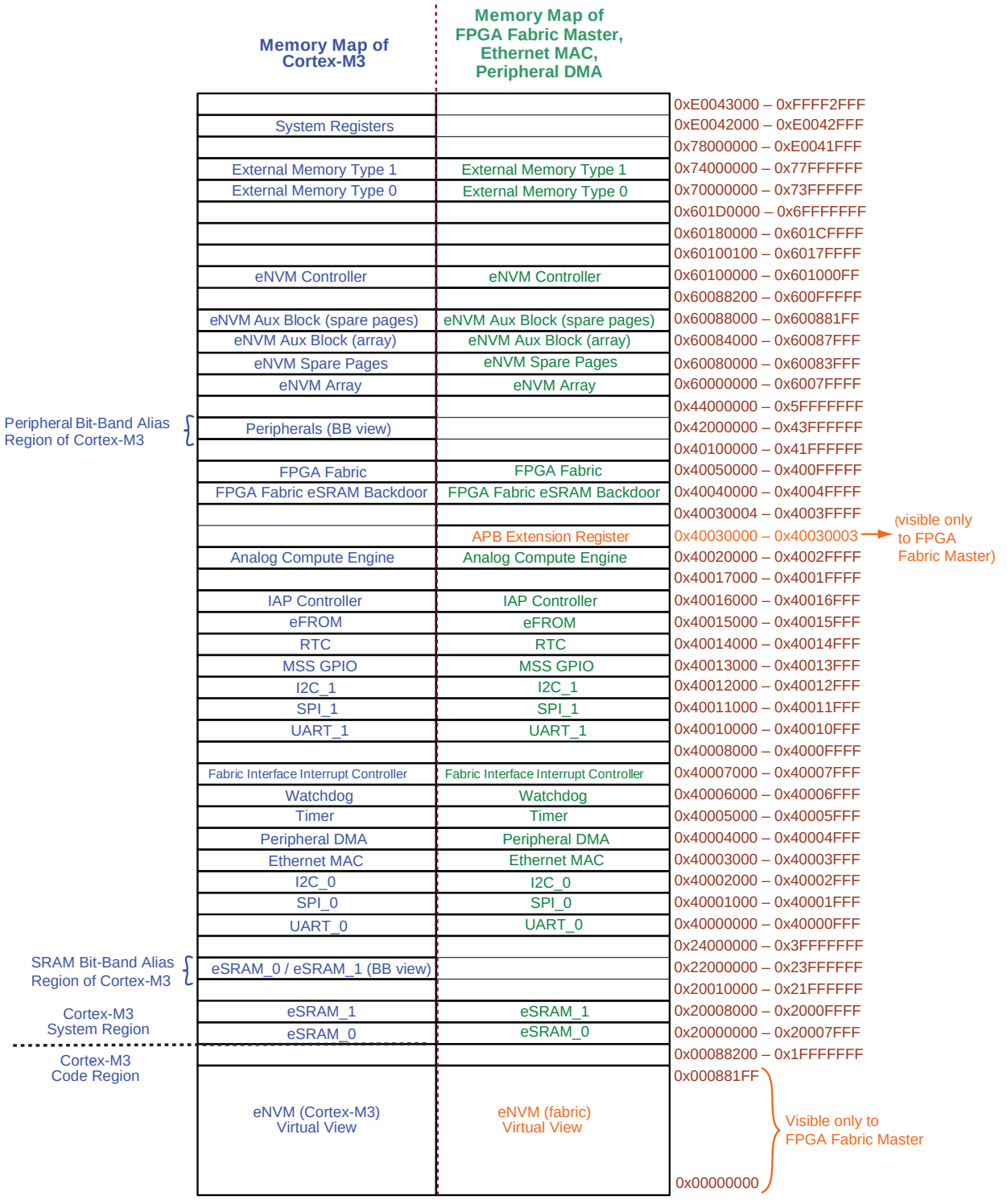


Figure 2-4 • **System Memory Map with 64 Kbytes of SRAM (A2F200)**

Remapping Embedded SRAMs

The AHB bus matrix supports the ability of remapping the eSRAM address space into code space (both eSRAM blocks are remapped). In this case, the two eSRAM blocks are remapped to appear at the bottom of Cortex-M3 code space. During this boot stage, the actual runtime firmware is copied into eSRAM and the firmware then sets the COM_ESRAMFWREMAP bit in the ESRAM_CR to 1. The resultant memory map is illustrated in Figure 2-6 on page 22.

By allowing the Cortex-M3 code bus to perform instruction fetches from the eSRAMS, performance is improved.

The eSRAM remap is actually performed by aliasing the eSRAM blocks, so that they appear in the code space, but are still accessible in system space. Therefore, the system designer must manage eSRAM accesses in such a way that a portion of eSRAM allocated in one space (the code space, for example) is left untouched in the other space (system space, for example).

In Figure 2-5, the Cortex-M3 microcontroller executes the application (including ISRs) from code space, allowing optimal performance. However, the corresponding region in system space is grayed out. Conversely, the stack (and heap, if present) as well as buffering for non-M3 masters (such as peripheral DMA or Ethernet DMA) is allocated out of system space and so must be left grayed out in code space.

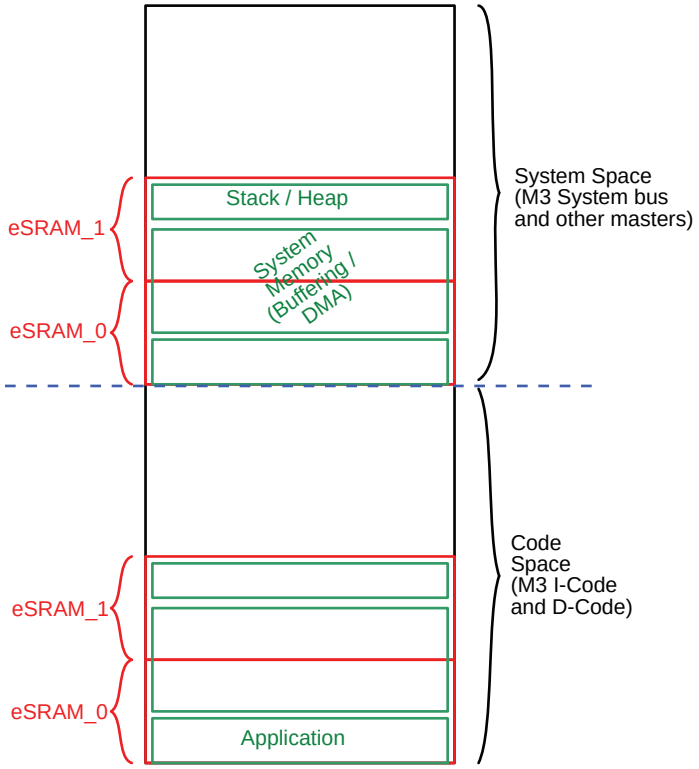


Figure 2-5 • **Remapped eSRAMs**

Figure 2-6 shows the resulting memory map when eSRAM is remapped.

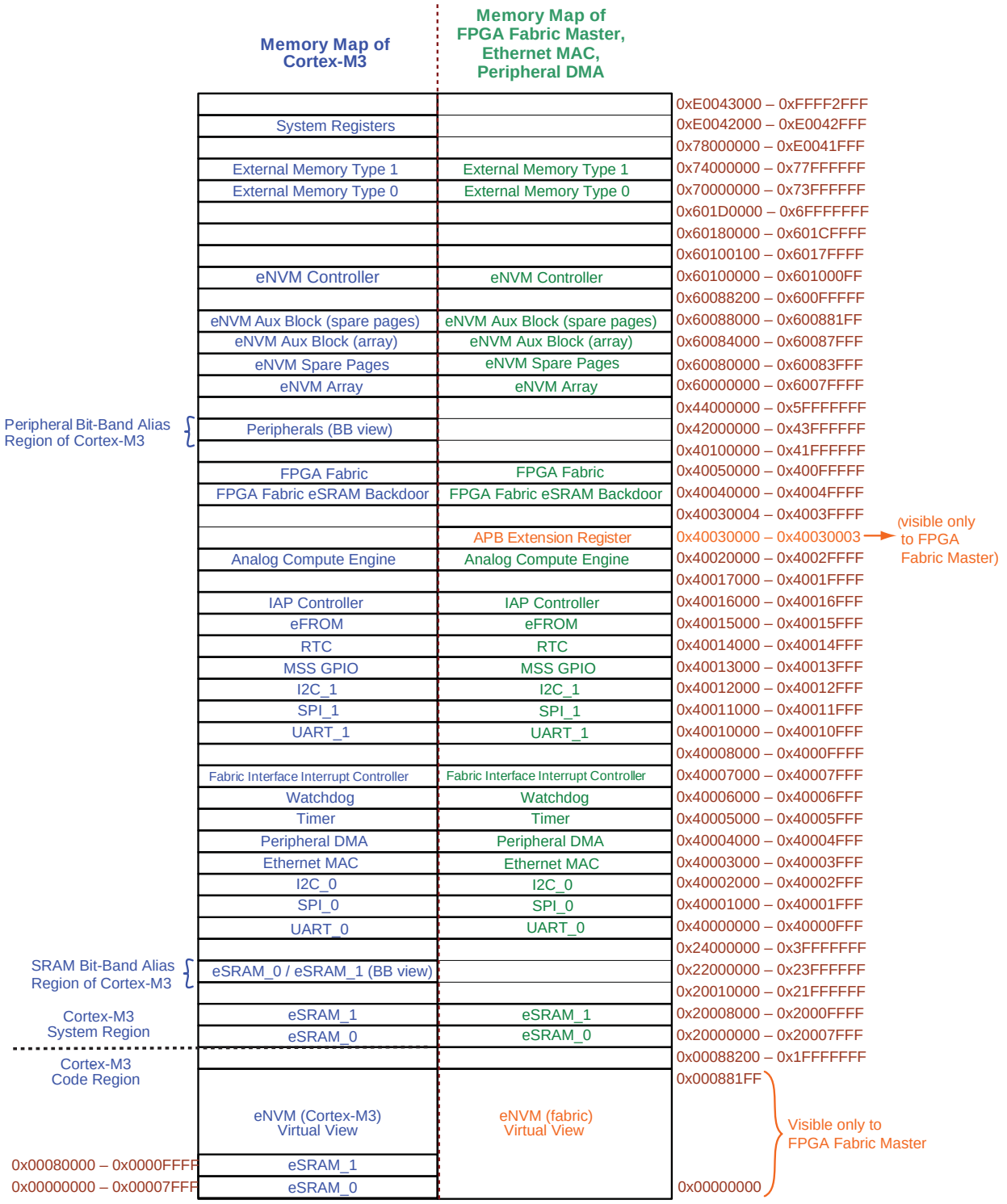


Figure 2-6 • **Memory Map with eSRAM Remapped**

This scheme allows flexibility to the system designer as to how much eSRAM is to be dedicated to each class of storage. For example, if the application, stack, and heap are small, this allows a large chunk of contiguous RAM to be allocated to buffering. If on the other hand, the system designer is more interested in optimal performance than flexibility, then eSRAM_0 could be dedicated to the application (and ISRs), while eSRAM_1 would be dedicated to stack, heap, and buffering. This would mean that the Cortex-M3 microcontroller operates in a fully Harvard fashion, since eSRAM_0 would only be accessed by the combined code bus, while eSRAM_1 would only be accessed by the system bus of M3, as well as the other (non-M3) masters.

Furthermore, if the system designer wishes to have deterministic latencies of ISR execution, the ISRs need to be located in eSRAM. However, the eSRAM must be uncontended in order to guarantee true determinism. Therefore, in such situations, the ISR and the stack should be in a separate eSRAM block from the memory being accessed for buffering by other masters, such as DMA.

The allocation of these memory classes to specific locations in eSRAM is accomplished by configuring the Cortex-M3 firmware linker script.

It is also possible for the user to execute code out of external memory (SRAM or flash). This is a slower interface, due to the latencies in accessing external memory and the fact that instruction fetches from system space are registered by the Cortex-M3 microcontroller.

The Boot Process

The boot process consists of three distinct steps: factory boot, system boot, and user boot.

Factory boot is reserved for use by Microsemi. System boot can be automated by the Libero® System-on-Chip (SoC) tool flow using the MSS configurator or can be performed by the user. User boot is generated by the user, if needed.

Factory Boot

After reset, the AHB bus matrix maps spare pages 1–17 of eNVM down into the bottom of Cortex-M3 code space at location 0x00000000. These spare pages are factory write protected. Factory boot initializes the device to a known state and passes control to system boot.

System Boot

System boot consists of the following steps:

1. C startup code.
2. Mapping of eNVM and, optionally, eSRAM to the desired address spaces.
3. Initialization of the microcontroller subsystem (MSS) to a known state.

The user can write portions of the system boot code or use the Libero MSS configurator to provide all the desired functionality of system boot. The current version of the System Boot code can be read at location 0x60080840.

User Boot

User boot would be any custom code that does not accomplish the steps outlined in the automated system boot and is optional.

AHB Bus Matrix Register Map

***Table 2-2* • AHB Bus Matrix Register Map**

Register Name	Address	R/W	Reset Value	Description
ESRAM_CR	0xE0042000	R/W	0x00000010	Controls address mapping of the eSRAMs
ENVM_CR	0xE0042004	R/W	0x00000072.	Configures eNVM parameters
ENVM_REMAP_SYS_CR	0xE0042008	R/W	0x00000001	eNVM mapping in system space
ENVM_REMAP_FAB_CR	0xE004200C	R/W	0x0	eNVM mapping in fabric master space
FAB_PROT_SIZE_CR	0xE0042010	R/W	0x0000001E	Fabric protect size
FAB_PROT_BASE_CR	0xE0042014	R/W	0x0	Fabric protect base address
AHB_MATRIX_CR	0xE0042018	R/W	0x00000007	Configures the AHB bus matrix
MSS_SR	0xE004201C	R	0x0	MSS status bits
CLR_MSS_SR	0xE0042020	W	0x0	Clear the MSS status bits

AHB Bus Matrix Register Bit Definitions

The AHB bus matrix control registers are located in the system registers address space at 0xE0042000 and extend to address 0xE004FFFF in the Cortex-M3 memory map.

eSRAM Configuration Register

***Table 2-3* • ESRAM_CR**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R/W	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	COM_ESRAMFWREMAP	R/W	0	Remap of embedded SRAMs. 0 = No remapping of the eSRAMs occurs. 1 = eSRAM_0 is mapped to location 0x00000000 and eSRAM_1 is mapped directly above it.

eNVM Configuration Register

***Table 2-4* • ENVM_CR**

Bit Number	Name	R/W	Reset Value	Description
31:8	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	ENVM_SIX_CYCLE	R/W	1	0 = No extra delay when reading from eNVM. 1 = Reads from eNVM will have one extra clock cycle of delay.
6	ENVM_PIPE_BYPASS	R/W	0	0 = Pipeline bypass disabled. 1 = Pipeline bypass enabled.
5	Reserved	R/W	0	Reserved
4:0	COM_ENVMREMAPSIZE	R/W	0b10010	COM_ENVMREMAPSIZE indicates the size of the segment in eNVM, which is to be remapped to location 0x00000000. This logically splits eNVM into a number of segments, each of which can be used to store a different firmware image. COM_ENVMREMAPSIZE is used to define the segment size for remapping of eNVM to Cortex-M3 space and for remapping a segment of eNVM for a soft processor in fabric if one so desires. SeeTable 2-8 on page 28.

***Table 2-5* • Definitions of Bit Combinations for COM_ENVMREMAPSIZE**

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Remap Size
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	Reserved
0	0	1	0	0	Reserved
0	0	1	0	1	Reserved
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	16 Kbytes
0	1	1	1	0	32 Kbytes
0	1	1	1	1	64 Kbytes
1	0	0	0	0	128 Kbytes
1	0	0	0	1	256 Kbytes
1	**0**	**0**	**1**	**0**	**512 Kbytes, reset value**

ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE are used to control access behavior to the eNVM. The latency of the initial access to a new eNVM page and the subsequent three accesses, if initiated, to the same eNVM page depends on the state of both ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE. The latencies (number of FCLK cycles) corresponding to the various combinations of ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS are as shown in Table 2-6.

***Table 2-6* • Bit Combination Definitions for ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE**

Bit 7	Bit 6	eNVM Access FCLK Cycles*
0	0	Reserved
0	1	5:1:1:1
1	0	Reserved
1	1	6:1:1:1

*Note: *Refer to the "Embedded Nonvolatile Memory Block (eNVM)" section in the SmartFusion Customizable System-on-Chip (cSoC) datasheet for useful information in determining speed.*

eNVM Remap Base Address Register

***Table 2-7* • ENVM_REMAP_SYS_CR**

Bit Number	Name	R/W	Reset Value	Description
31:20	Reserved	R/W	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19:1	COM_ENVMREMAPBASE	R/W	0x40000	Offset address of eNVM for remapping. COM_ENVMREMAPBASE indicates the offset within eNVM address space of the base address of the segment in eNVM, which is to be remapped to location 0x00000000. The base address of the remapped segment of eNVM is determined by the value of this bus. Bit 0 of this bus is defined as COM_ENVMREMAPENABLE.
0	COM_ENVMREMAPENABLE	R/W	0b1	0 = eNVM remap not enabled. Bottom of eNVM is mapped to address 0x00000000. 1 = eNVM remap enabled. eNVM visible at 0x00000000 is a remapped segment of the eNVM.

Bits [19:N] of this bus indicate the base address of the remapped segment. The value of N depends on the eNVM remap section size, so that the base address is aligned according to an even multiple of segment size. The power of 2 size specified by COM_ENVMREMAPSIZE defines how many bits of base address are used. For example, if the COM_ENVMREMAPSIZE is 0x0f, this corresponds to a segment size of 64 Kbytes, which is 2^{16}. Therefore, the value of N in this case is 16. The base address of the region, in this case, is specified by COM_ ENVMREMAPBASE [19:16]. For example:

1. COM_ENVMREMAPBASE[19:16] = 0x0. The 64 Kbytes segment located at the physical memory address of 0x60000000 is mapped into address 0x00000000.
2. COM_ENVMREMAPBASE[19:16] = 0x1. The 64 Kbytes segment located at the physical memory address of 0x60010000 is mapped into address 0x00000000.
3. COM_ENVMREMAPBASE[19:16] = 0x2. The 64 Kbytes segment located at the physical memory address of 0x60020000 is mapped into address 0x00000000.

If the user attempts to remap a segment of eNVM that does not exist, unpredictable results will occur.

eNVM FPGA Fabric Remap Base Address Register

***Table 2-8* • ENVM_REMAP_FAB_CR**

Bit Number	Name	R/W	Reset Value	Description
31:20	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19:1	COM_ENVMFABREMAPBASE	R/W	0	Offset address of eNVM for remapping. COM_ ENVMFABREMAPBASE indicates the offset within eNVM address space of the base address of the segment in eNVM, which is to be remapped to location 0x00000000 for use by a soft processor in the FPGA fabric. The base address of the remapped segment of eNVM is determined by the value of this bus. Bit 0 of this bus is defined as COM_ENVMFABREMAPENABLE.
0	COM_ENVMFABREMAPENABLE	R/W	0	0 = eNVM remap not enabled. Bottom of eNVM is mapped to address 0x00000000. 1 = eNVM remap enabled. eNVM visible at 0x00000000 is a remapped segment of the eNVM.

Bits [19:N] of this bus indicate the base address of the remapped segment. The value of N depends on the eNVM remap section size, so that the base address is aligned according to an even multiple of segment size. The power of 2 size specified by COM_ENVMREMAPSIZE defines how many bits of base address are used. For example, if the COM_ENVMREMAPSIZE is 0x0f, this corresponds to a segment size of 64 Kbytes, which is 2^{16}. Therefore the value of N in this case is 16. The base address of the region, in this case, is specified by COM_ ENVMFABREMAPBASE [19:16]. For example:

1. COM_ ENVMFABREMAPBASE [19:16] = 0x0. The 64 Kbytes segment located at the physical memory address of 0x60000000 is mapped into address 0x00000000.
2. COM_ ENVMFABREMAPBASE [19:16] = 0x1. The 64 Kbytes segment located at the physical memory address of 0x60010000 is mapped into address 0x00000000.
3. COM_ ENVMFABREMAPBASE [19:16] = 0x2. The 64 Kbytes segment located at the physical memory address of 0x60020000 is mapped into address 0x00000000.

If the user attempts to remap a segment of eNVM that does not exist, unpredictable results will occur.

FPGA Fabric Protect Size Register

***Table 2-9* • FAB_PROT_SIZE_CR**

Bit Number	Name	R/W	Reset Value	Description
31:5	Reserved	R/W	0x00000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	COM_PROTREGIONSIZE	R/W	0x1E	Size of the memory region inaccessible to the FPGA fabric master.

***Table 2-10* • Definitions of Bit Combinations for COM_PROTREGIONSIZE**

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Remap Size
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	Reserved
0	0	1	0	0	Reserved
0	0	1	0	1	Reserved
0	0	1	1	0	128 Bytes
0	0	1	1	1	Reserved
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	2 Kbytes
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	16 Kbytes
0	1	1	1	0	32 Kbytes
0	1	1	1	1	64 Kbytes
1	0	0	0	0	128 Kbytes
1	0	0	0	1	256 Kbytes
1	0	0	1	0	512 Kbytes
1	0	0	1	1	Reserved
1	0	1	0	0	Reserved
1	0	1	0	1	Reserved
1	0	1	1	0	8 Mbytes
1	0	1	1	1	Reserved
1	1	0	0	0	Reserved
1	1	0	0	1	Reserved
1	1	0	1	0	128 Mbytes
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	Reserved
1	1	1	1	0	2 Gbytes
1	1	1	1	1	Reserved

FPGA Fabric Protect Base Register

Table 2-11 • **FAB_PROT_BASE_CR**

Bit Number	Name	R/W	Reset Value	Description
31:N	COM_PROTREGIONBASE	R/W	0	Bits [31:N] of this bus indicate the absolute base address of the protected segment. The value of N depends on the protected region size, so that the base address is aligned according to an even multiple of segment size. The power of 2 size specified by COM_PROTREGIONSIZE defines how many bits of base address are used. See examples below.
0	COM_PROTREGIONENABLE	R/W	0	0 = Protection region disabled. A fabric master can access any location in the memory map as long as the fabric master port is enabled in the AHB bus matrix. 1 = Protection region enabled. Any access by a fabric master to this region of memory returns an error in the bus transaction. The COM_ERRORSTATUS field of the MSS_SR register is updated appropriately. The ABM_ERROR_IRQ signal is also asserted and a trap can be made if IRQ24 is enabled in the NVIC.

For example, if the COM_PROTREGIONSIZE is 0x0F, this corresponds to a segment size of 64 Kbytes, which is 2^{16}. Therefore the value of N in this case is 16. So the base address of the region, in this case, is specified by COM_PROTREGIONBASE [31:16]. Likewise, if COM_PROTREGIONSIZE is 0x10, this corresponds to a segment size of 128 Kbytes, which is 2^{17}. Therefore the value of N in this case is 17. So the absolute base address of the region is specified by COM_PROTREGIONBASE [31:17]. For example:

1. COM_PROTREGIONBASE [31:17] = 0x0000. The 128 Kbytes segment located at the physical memory address of 0x60000000 is protected from FPGA fabric master access.
2. COM_PROTREGIONBASE [31:17] = 0x0001. The 128 Kbytes segment located at the physical memory address of 0x60020000 is protected from FPGA fabric master access.
3. COM_PROTREGIONBASE [31:17] = 0x0002. The 128 Kbytes segment located at the physical memory address of 0x60040000 is protected from FPGA fabric master access.
4. COM_PROTREGIONBASE [31:17] = 0x0003. The 128 Kbytes segment located at the physical memory address of 0x60060000 is protected from FPGA fabric master access.
5. COM_PROTREGIONBASE [31:17] = 0x0004. The 128 Kbytes segment located at the physical memory address of 0x60080000 is protected from FPGA fabric master access.

AHB Bus Matrix Configuration Register

***Table 2-12* • AHB_MATRIX_CR**

Bit Number	Name	R/W	Reset Value	Description
31:4	Reserved	R/W	0x0000000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	COM_WEIGHTEDMODE	R/W	0	0 = Round robin slave arbitration (reset default). 1 = Weighted round robin slave arbitration.
2:0	COM_MASTERENABLE	R/W	0	Enable control for each of the non-Cortex-M3 masters connected to the AHB bus matrix. For each of these masters, if the corresponding bit is 0, then the master cannot access any of the slave ports connected to the matrix. If the bit is 1, the master can access any of the slaves connected to the matrix. In the case of the fabric master, access is further qualified with the protected region mechanism, described above. The bits have the following definitions: **Bit 2: Peripheral DMA** 0 = Peripheral DMA cannot access any AHB bus matrix slaves. 1 = Peripheral DMA has access to the AHB bus matrix slaves. **Bit 1: Ethernet MAC** 0 = Ethernet MAC cannot access any AHB bus matrix slaves. 1 = Ethernet MAC has access to the AHB bus matrix slaves. **Bit 0: FPGA fabric master** 0 = FPGA master cannot access any AHB bus matrix slaves. 1 = FPGA master has access to AHB bus matrix slaves qualified by FAB_PROT_BASE_CR and FAB_PROT_SIZE_CR values.

Microcontroller Subsystem Status Register (MSS_SR)

***Table 2-13* • MSS_SR**

Bit Number	Name	R/W	Reset Value	Description
31:11	Reserved	R	0x00000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	PLLLOCKLOSTINT	R	0	This bit indicates that a falling edge event occurred on PLLLOCK. This signal is also available to the FPGA fabric. This indicates that the PLL lost lock. This signal corresponds to IRQ23 in the Cortex-M3 NVIC. IRQ23 corresponds to bit location 23 in the 32-bit word at address location 0xE000E100. This bit is read-only and can be cleared by writing a 1 to the CLRPLLLOCKLOSTINT bit in the CLR_MSS_SR register. 0 = "Don't care." 1 = PLL lost lock.
9	PLLLOCKINT	R	0	This bit indicates that a rising edge event occurred on the PLLLOCK signal. This indicates that the PLL is locked. This signal corresponds to IRQ22 in the Cortex-M3 NVIC. IRQ22 corresponds to bit location 22 in the 32-bit word at address location 0xE000E100. This bit is read-only and can be cleared by writing a 1 to the CLRPLLLOCKINT bit in the CLR_MSS_SR register. 0 = "Don't care." 1 = PLL came into lock.
8:4	COM_ERRORSTATUS	R	0	Each bit on this bus indicates if any accesses by the corresponding master on the AHB bus matrix resulted in either HRESP assertion by the slave to the AHB bus matrix, HRESP assertion by the AHB bus matrix to that master (in the case of blocked fabric master) or was decoded by the AHB bus matrix as being unimplemented address space. These register bits are sticky and are cleared by writing one to the corresponding COM_CLEARSTATUS bit in the CLR_MSS_SR register. Bit definitions are as follows: Bit 8: Peripheral DMA master Bit 7: Ethernet MAC master Bit 6: Fabric master Bit 5: Cortex-M3 system bus master Bit 4: Cortex-M3 I-Code/D-Code bus master These signals are not used as interrupts to the Cortex-M3 microcontroller. Instead, they are ORed together in the AHB bus matrix to create a signal called ABM_ERROR_IRQ, which is used as an interrupt to the Cortex-M3 microcontroller. This signal corresponds to IRQ24 in the Cortex-M3 NVIC. IRQ24 corresponds to bit location 24 in the 32-bit word at address location 0xE000E100. ABM_ERROR_IRQ is not brought into the System Register's space as a status bit for user's firmware to read.

***Table 2-13* • MSS_SR (continued)**

Bit Number	Name	R/W	Reset Value	Description
3	BROWNOUT3_3VINT	R	0	Indicates that the 3.3 V supply has dropped below 2.5 V. This signal corresponds to IRQ 2 in the Cortex-M3 NVIC. IRQ 2 corresponds to bit location 2 in the 32-bit word at address location 0xE000E100. 0 = "Don't care." 1 = 3.3 V has fallen below 2.5 V.
2	BROWNOUT1_5VINT	R	0	Indicates that the 1.5 V supply has dropped below 1.3 V. This signal corresponds to IRQ 1 in the Cortex-M3 NVIC. IRQ 1 corresponds to bit location 1 in the 32-bit word at address location 0xE000E100. 0 = "Don't care." 1 = 1.5 V has fallen below 1.3 V.
1	WDOGTIMEOUTEVENT	R	0	This signal is a sticky version of the WDOGTIMEOUTINT signal (which is itself sticky but is cleared by MSS_SYSTEM_RESET_N). WDOGTIMEOUTEVENT is not affected by MSS_SYSTEM_RESET_N. This allows firmware to determine if a system reset occurred due to a watchdog timeout event. This signal is not used as an interrupt to the Cortex-M3 microcontroller. This bit is reset to 0 by PORESET_N only and is unaffected by MSS_SYSTEM_RESET_N. 0 = "Don't care." 1 = Watchdog has timed out.
0	RTCMATCHEVENT	R	0	This signal is a sticky version of the MATCH signal from the RTC. If a rising edge event is seen on MATCH, after synchronization to FCLK domain, then this bit is asserted. It stays asserted until cleared by CLRRTCMATCHEVENT. This signal is used as an interrupt to the Cortex-M3 microcontroller. This signal corresponds to IRQ3 in the Cortex-M3 NVIC. IRQ3 corresponds to bit location 3 in the 32-bit word at address location 0xE000E100. Reset value = 0. 0 = "Don't care." 1 = RTC has matched an event.

Clear Microcontroller Subsystem Status Register

Table 2-14 • **CLR_MSS_SR**

Bit Number	Name	R/W	Reset Value	Description
31:11	Reserved	W	0	To provide compatibility with future products, the value of a reserved bit should be preserved across a write operation by writing a zero to those bits.
10	CLRPLLLOCKLOSTINT	W	0	Writing a 1 to this bit clears the interrupt signal PLLLOCKLOSTINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the PLLLOCKLOSTINT signal.
9	CLRPLLLOCKINT	W	0	Writing a 1 to this bit clears the interrupt signal PLLLOCKTINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the PLLLOCKINT signal.
8:4	COM_CLEARSTATUS	W	0	Writing a 1 to any of the bits in COM_CLEARSTATUS clears the interrupt signal ABM_ERROR_IRQ. Writing a zero has no effect. Bit 8: Peripheral DMA master Bit 7: Ethernet MAC master Bit 6: Fabric master Bit 5: Cortex-M3 system bus master Bit 4: Cortex-M3 I-Code/D-Code bus master
3	CLRBROWNOUT3_3VINT	W	0	Writing a 1 to this bit clears the interrupt signal BROWNOUT3_3VINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the BROWNOUT3_3VINT signal.
2	CLRBROWNOUT1_5VINT	W	0	Writing a 1 to this bit clears the interrupt signal BROWNOUT1_5VINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the BROWNOUT1_5VINT signal.
1	CLRWDOGTIMEOUTEVENT	W	0	Writing a 1 to this bit clears the WDOGTIMEOUTEVENT bit in the WDOG_EVENT_REG register. Writing a zero has no effect. 0 = No effect. 1 = Clear the WDOGTIMEOUTEVENT.
0	CLRRTCMATCHEVENT	W	0	Writing a 1 to this bit clears the RTCMATCHEVENT bit in the RTC_MATCH_EVENT_REG register. Writing a zero has no effect. 0 = No effect. 1 = Clear the RTCMATCHEVENT.

3 – Peripheral DMA (PDMA)

The PDMA offloads the ARM Cortex-M3 processor from data movement tasks from peripherals to memory, memory to peripherals, and memory to memory. The block diagram of the PDMA is shown in Figure 3-1.

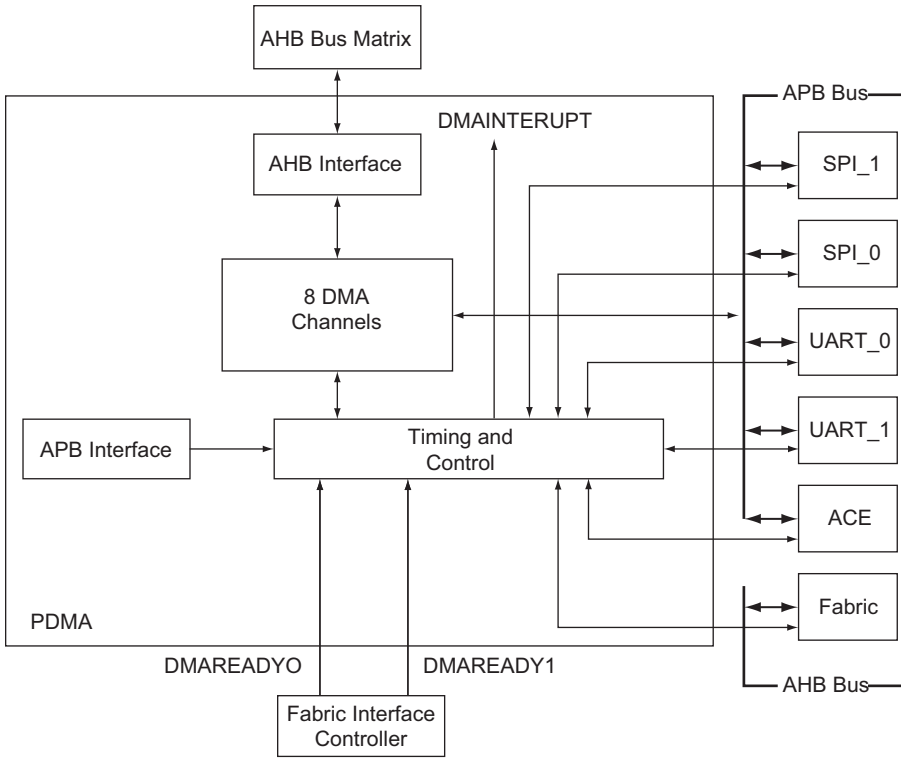


Figure 3-1 • **PDMA Block Diagram**

PDMA Features

- 8 channels
- Ping-pong mode support
- Memory to memory DMA capable
- Channels can be designated as high priority

Functional Description

The PDMA consists of eight instances of a single DMA channel design. Each channel can be configured to perform 8-bit, 16-bit, or 32-bit transfers from the peripheral to memory, memory to peripheral, or between memory and memory. Channels can be assigned to peripherals or memory arbitrarily. For example, if the user is interested in receiving only DMA data from one of the SPI ports, only one channel is required. In this case, the DIR bit in the CHx_CONTROL_REG would be set to 0 (peripheral to memory) and the PERIPHERAL_SEL field would be set to 4 (SPI_0 receive to memory). Throughout this document, a lower case x in register and signal descriptions is used as a place holder for 0 or 1, indicating PDMA_0 or PDMA_1.

If bidirectional DMA of peripheral to memory (receive) and memory to peripheral (transmit) is desired, two channels must be programmed appropriately. In particular, the TRANSFER_SIZE fields in both the CHx_CONTROL_REG registers must be programmed identically.

The PDMA performs the correct byte lane adjustments appropriate to the address being used on the AHB. Efficient use of memory storage is achieved in this manner, even if only performing byte or 16-bit accesses to or from a peripheral.

For accesses by the PDMA to peripherals, the lowest 8 or 16 bits of the data bus are always used for 8-bit or 16-bit transfers. For 32-bit transfers, the full 32 bits are used.

It is possible to configure the data width of a transfer to be independent of the address increment. The address increment at both ends of the DMA transfer can be different, which is required when reading from a peripheral holding register (single address) and writing to memory incrementally (many addresses).

DMA transfers can be paused by setting the PAUSE bit in the CHx_CONTROL_REG. The DMA will stall until the user clears this bit.

The PDMA performs single cycle accesses on the AHB interface. No DMA operations occur on the APB bus interface of the PDMA. This interface is purely an APB slave, used for configuration of the PDMA.

For each peripheral DMA channel (0 to 7), two sets of registers are maintained in the PDMA. These are set up by firmware in order to specify the start address of the DMA burst, the destination address of the DMA burst, and the transfer count of the DMA burst for each of the two buffers. This is called ping-pong mode.

Ping-Pong Mode

In order to support continuous DMA operations on each peripheral DMA channel, dual-buffering is provided, along with two sets of registers per channel. The buffers are referred to as A and B. The sequence of operations performed by firmware for ping-pong operation on DMA channel 0 is as follows: (the channel is assumed to be configured properly by writing to CHANNEL_0_CTRL first).

1. Write to CHANNEL_0_BUFFER_A_SRC_ADDR.
2. Write to CHANNEL_0_BUFFER_A_DST_ADDR.
3. Write to CHANNEL_0_BUFFER_B_SRC_ADDR.
4. Write to CHANNEL_0_BUFFER_B_DST_ADDR.
5. Write to CHANNEL_0_BUFFER_A_TRANSFER_COUNT (DMA starts using buffer A).
6. Write to CHANNEL_0_BUFFER_B_TRANSFER_COUNT (DMA will use buffer B when CHANNEL_0_BUFFER_A_TRANSFER_COUNT is 0).
7. Wait for interrupt on the DMA channel, buffer A.
8. Write to CHANNEL_0_BUFFER_A_SRC_ADDR.
9. Write to CHANNEL_0_BUFFER_A_DST_ADDR.
10. Write to CHANNEL_0_BUFFER_A_TRANSFER_COUNT (DMA will use buffer A when CHANNEL_0_BUFFER_B_TRANSFER_COUNT is 0).
11. Wait for interrupt on the DMA channel, buffer B.
12. Write to CHANNEL_0_BUFFER_B_SRC_ADDR.
13. Write to CHANNEL_0_BUFFER_B_DST_ADDR.
14. Write to CHANNEL_0_BUFFER_B_TRANSFER_COUNT (DMA will use buffer B when CHANNEL_0_BUFFER_A_TRANSFER_COUNT is 0).
15. Repeat steps 7 to 14 until finished.

This removes the real-time constraint on the firmware of having to service the DMA channel in real time, which would exist if there were only one DMA buffer per channel.

Posted APB Writes

The AHB to APB bridges in SmartFusion cSoCs implement posted writes (also called dump and run) for write accesses to peripherals. The effect of this is that if the PDMA performs a write operation to a peripheral, the data is not actually written into the peripheral until sometime after the PDMA block thinks it is written. Therefore, the PDMA block should not start another DMA on this channel based on the state of the ready signal from that peripheral until the write is complete. The time window involved is variable, depending on the ratio of FCLK to the APB clocks (PCLK0, PCLK1, or ACLK). WRITE_ADJ in CHx_CONTROL_REG is an 8-bit binary coded field used to define, for each DMA channel, how long to wait (in FCLKs) after each DMA transfer cycle before interpreting the ready signal for that DMA channel as representing a new request.

Memory to Memory Transfers

For memory to memory transfers, the DMA starts once the BUF_A_COUNT or BUF_B_COUNT is non-zero. Firmware should initialize the transfer first by writing to the source, destination, and control registers; then writing to one of the transfer count registers, BUF_A_COUNT or BUF_B_COUNT, to initiate the DMA. If the PAUSE bit in CHx_CONTROL_REG is set when the user writes a non-zero value to either BUF_A_COUNT or BUF_B_COUNT, the DMA cycle will wait until PAUSE is cleared.

Channel Priority

The arbitration algorithm used to service the channels assumes all channels are equal priority by default. However, it is possible to define a channel as being high priority. For example, the user may want to give higher priority to DMA channels corresponding to SPI peripherals than to those corresponding to UARTs, as SPI has no built-in flow-control.

As a way of prioritizing traffic within the DMA, the RATIOHILO field in RATIO_HIGH_LOW is used to indicate the ratio of high priority to low priority DMA access opportunities. This register gives the number of DMA opportunities provided by the channel arbiter to high priority channels for every one opportunity provided to a low priority channel. Table 3-1 describes valid values for RATIOHILO. All other values are reserved; RATIOHILO can only assume a value listed in the Value column of Table 3-1.

***Table 3-1* • RATIOHILO Field Definition**

Value	High:Low Ratio	Comments
0	–	Round Robin
1	1:1	Ping-pong between high and low priority requests
3	3:1	3 high to 1 low
7	7:1	7 high to 1 low
15	15:1	15 high to 1 low
31	31:1	31 high to 1 low
63	63:1	63 high to 1 low
127	127:1	127 high to 1 low
255	255:1	255 high to 1 low
All others		Reserved

For example, a RATIOHILO value of 3:1 means that if there are continuous high priority requests and low priority requests, then there will be 3 high priority requests serviced to one low priority request. There is an internal counter in the PDMA which takes its value from the RATIOHILO value. When this internal counter reaches 0, low priority requests are allowed. Each time a high priority request is serviced, the counter is decremented by 1. Each time a low priority request is serviced, the internal counter is reset to the RATIOHILO value.

When RATIOHILO is a 0, the high requests and the low requests will round robin. This is built into the arbiter. The arbiter does a high request and then asks if a low request is permitted. Similarly, a value of 1 in RATIOHILO will allow for ping-ponging between high and low requests. This is not to be confused with ping-pong mode.

System Dependencies

Clocks

The PDMA runs off the system clock, FCLK. Users must be cognizant of the clock speed of the APB bus over which DMA transfers are being exercised. The field WRITE_ADJ contains a binary value, indicating the number of FCLK periods the PDMA must wait after completion of a read or write access to a peripheral before evaluating the out-of-band status signals from that peripheral for another transfer. This is typically used to ensure that a posted write has fully completed to the peripheral in cases where the peripheral is running at a lower clock frequency than the PDMA. However, it may also be used to allow the PDMA to take account of internal latencies in the peripheral, where the ready status of a FIFO may not be available for a number of clock ticks after a read or write, due to internal synchronization delays, for example, within the peripheral. This applies particularly in the case of user-designed peripherals in the FPGA fabric.

Resets

All PDMA registers are reset to zero on power-up. Users have the option under software control to reset the PDMA by writing to the System Register located on the private peripheral bus of the Cortex-M3 processor. Specifically, this System Register is SOFT_RST_CR, located at address 0xE0042030. The PDMA_SR control bit is encoded in bit location 5 as follows:

Bit 5: Function

0: PDMA reset released

1: PDMA held in reset (reset value)

In addition to being able to reset the entire PDMA under firmware control, each individual channel can be reset by user firmware by setting the RESET bit in the CHx_CONTROL_REG to 1.

Interrupts

There is one interrupt (DMAINTERRUPT) from the PDMA to the NVIC on the Cortex-M3 processor (see Table 1-5 on page 10).

The DMAINTERRUPT signal is mapped to INTISR[9] or IRQ 9 in the Cortex-M3 NVIC controller. The interrupt enable bit for DMAINTERRUPT within the NVIC is located at address 0xE000E100; IRQ 9 corresponds to bit location 9. Users must also enable specific channel interrupts within the PDMA by setting the INTEN bit in the CHx_CONTROL_REG to 1. Users can determine which buffer of which channel caused the interrupt by reading the BUFFER_STATUS.

PDMA Register Map

***Table 3-2* • PDMA Memory Map**

Register Name	Address	R/W	Reset Value	Description
RATIO_HIGH_LOW	0x40004000	R/W	0	Ratio of high priority transfers versus low priority transfers
BUFFER_STATUS	0x40004004	R/W	0	Indicates when buffers have drained
CHANNEL_x_CONTROL (x = 0)	0x40004020	R/W	0	Channel 0 Control Register
CHANNEL_x_STATUS (x = 0)	0x40004024	R	0	Channel 0 Status Register
CHANNEL_x_BUFFER_A_SRC_ADDR (x = 0)	0x40004028	R/W	0	Channel 0 buffer A source address
CHANNEL_x_BUFFER_A_DST_ADDR (x = 0)	0x4000402C	R/W	0	Channel 0 buffer A destination address
CHANNEL_x_BUFFER_A_TRANSFER_COUNT (x = 0)	0x40004030	R/W	0	Channel 0 buffer A transfer count
CHANNEL_x_BUFFER_B_SRC_ADDR (x = 0)	0x40004034	R/W	0	Channel 0 buffer B source address
CHANNEL_x_BUFFER_B_DST_ADDR (x = 0)	0x40004038	R/W	0	Channel 0 buffer B destination address
CHANNEL_x_BUFFER_B_TRANSFER_COUNT (x = 0)	0x4000403C	R/W	0	Channel 0 buffer B transfer count
CHANNEL_1_CTRL	0x40004040	R/W	0	Channel 1 Control Register
CHANNEL_1_STATUS	0x40004044	R	0	Channel 1 Status Register
CHANNEL_1_BUFFER_A_SRC_ADDR	0x40004048	R/W	0	Channel 1 buffer A source address
CHANNEL_1_BUFFER_A_DST_ADDR	0x4000404C	R/W	0	Channel 1 buffer A destination address
CHANNEL_1_BUFFER_A_TRANSFER_COUNT	0x40004050	R/W	0	Channel 1 buffer A transfer count
CHANNEL_1_BUFFER_B_SRC_ADDR	0x40004054	R/W	0	Channel 1 buffer B source address
CHANNEL_1_BUFFER_B_DST_ADDR	0x40004058	R/W	0	Channel 1 buffer B destination address
CHANNEL_1_BUFFER_B_TRANSFER_COUNT	0x4000405C	R/W	0	Channel 1 buffer B transfer count
CHANNEL_2_CTRL	0x40004060	R/W	0	Channel 2 Control Register
CHANNEL_2_STATUS	0x40004064	R	0	Channel 2 Status Register
CHANNEL_2_BUFFER_A_SRC_ADDR	0x40004068	R/W	0	Channel 2 buffer A source address
CHANNEL_2_BUFFER_A_DST_ADDR	0x4000406C	R/W	0	Channel 2 buffer A destination address
CHANNEL_2_BUFFER_A_TRANSFER_COUNT	0x40004070	R/W	0	Channel 2 buffer A transfer count
CHANNEL_2_BUFFER_B_SRC_ADDR	0x40004074	R/W	0	Channel 2 buffer B source address
CHANNEL_2_BUFFER_B_DST_ADDR	0x40004078	R/W	0	Channel 2 buffer B destination address
CHANNEL_2_BUFFER_B_TRANSFER_COUNT	0x4000407C	R/W	0	Channel 2 buffer B transfer count
CHANNEL_3_CTRL	0x40004080	R/W	0	Channel 3 Control Register

***Table 3-2* • PDMA Memory Map (continued)**

Register Name	Address	R/W	Reset Value	Description
CHANNEL_3_STATUS	0x40004084	R	0	Channel 3 Status Register
CHANNEL_3_BUFFER_A_SRC_ADDR	0x40004088	R/W	0	Channel 3 buffer A source address
CHANNEL_3_BUFFER_A_DST_ADDR	0x4000408C	R/W	0	Channel 3 buffer A destination address
CHANNEL_3_BUFFER_A_TRANSFER_COUNT	0x40004090	R/W	0	Channel 3 buffer A transfer count
CHANNEL_3_BUFFER_B_SRC_ADDR	0x40004094	R/W	0	Channel 3 buffer B source address
CHANNEL_3_BUFFER_B_DST_ADDR	0x40004098	R/W	0	Channel 3 buffer B destination address
CHANNEL_3_BUFFER_B_TRANSFER_COUNT	0x4000409C	R/W	0	Channel 3 buffer B transfer count
CHANNEL_4_CTRL	0x400040A0	R/W	0	Channel 4 Control Register
CHANNEL_4_STATUS	0x400040A4	R	0	Channel 4 Status Register
CHANNEL_4_BUFFER_A_SRC_ADDR	0x400040A8	R/W	0	Channel 4 buffer A source address
CHANNEL_4_BUFFER_A_DST_ADDR	0x400040AC	R/W	0	Channel 4 buffer A destination address
CHANNEL_4_BUFFER_A_TRANSFER_COUNT	0x400040B0	R/W	0	Channel 4 buffer A transfer count
CHANNEL_4_BUFFER_B_SRC_ADDR	0x400040B4	R/W	0	Channel 4 buffer B source address
CHANNEL_4_BUFFER_B_DST_ADDR	0x400040B8	R/W	0	Channel 4 buffer B destination address
CHANNEL_4_BUFFER_B_TRANSFER_COUNT	0x400040BC	R/W	0	Channel 4 buffer B transfer count
CHANNEL_5_CTRL	0x400040C0	R/W	0	Channel 5 Control Register
CHANNEL_5_STATUS	0x400040C4	R	0	Channel 5 Status Register
CHANNEL_5_BUFFER_A_SRC_ADDR	0x400040C8	R/W	0	Channel 5 buffer A source address
CHANNEL_5_BUFFER_A_DST_ADDR	0x400040CC	R/W	0	Channel 5 buffer A destination address
CHANNEL_5_BUFFER_A_TRANSFER_COUNT	0x400040D0	R/W	0	Channel 5 buffer A transfer count
CHANNEL_5_BUFFER_B_SRC_ADDR	0x400040D4	R/W	0	Channel 5 buffer B source address
CHANNEL_5_BUFFER_B_DST_ADDR	0x400040D8	R/W	0	Channel 5 buffer B destination address
CHANNEL_5_BUFFER_B_TRANSFER_COUNT	0x400040DC	R/W	0	Channel 5 buffer B transfer count
CHANNEL_6_CTRL	0x400040E0	R/W	0	Channel 6 Control Register
CHANNEL_6_STATUS	0x400040E4	R	0	Channel 6 Status Register
CHANNEL_6_BUFFER_A_SRC_ADDR	0x400040E8	R/W	0	Channel 6 buffer A source address
CHANNEL_6_BUFFER_A_DST_ADDR	0x400040EC	R/W	0	Channel 6 buffer A destination address
CHANNEL_6_BUFFER_A_TRANSFER_COUNT	0x400040F0	R/W	0	Channel 6 buffer A transfer count
CHANNEL_6_BUFFER_B_SRC_ADDR	0x400040F4	R/W	0	Channel 6 buffer B source address
CHANNEL_6_BUFFER_B_DST_ADDR	0x400040F8	R/W	0	Channel 6 buffer B destination address

Table 3-2 • **PDMA Memory Map (continued)**

Register Name	Address	R/W	Reset Value	Description
CHANNEL_6_BUFFER_B_TRANSFER_COUNT	0x400040FC	R/W	0	Channel 6 buffer B transfer count
CHANNEL_7_CTRL	0x40004100	R/W	0	Channel 7 Control Register
CHANNEL_7_STATUS	0x40004104	R	0	Channel 7 Status Register
CHANNEL_7_BUFFER_A_SRC_ADDR	0x40004108	R/W	0	Channel 7 buffer A source address
CHANNEL_7_BUFFER_A_DST_ADDR	0x4000410C	R/W	0	Channel 7 buffer A destination address
CHANNEL_7_BUFFER_A_TRANSFER_COUNT	0x40004110	R/W	0	Channel 7 buffer A transfer count
CHANNEL_7_BUFFER_B_SRC_ADDR	0x40004114	R/W	0	Channel 7 buffer B source address
CHANNEL_7_BUFFER_B_DST_ADDR	0x40004118	R/W	0	Channel 7 buffer B destination address
CHANNEL_7_BUFFER_B_TRANSFER_COUNT	0x4000411C	R/W	0	Channel 7 buffer B transfer count

RATIO_HIGH_LOW Register

Table 3-3 • **RATIO_HIGH_LOW**

Bit Number	Name	R/W	Reset Value	Function
31:8	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RATIOHILO	R/W	0	This field indicates the ratio of high priority to low priority for DMA access opportunities. Only certain values are allowed, as indicated in Table 3-1 on page 37.

BUFFER_STATUS Register

Table 3-4 • **BUFFER_STATUS**

Bit Number	Name	R/W	Reset Value	Function
31:16	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	CH7BUFB	R/W	0	If CH_COMP_B for channel 7 is set and if BUF_B_SEL for channel 7 is clear, then this bit is asserted = 1.
14	CH7BUFA	R/W	0	If CH_COMP_A for channel 7 is set and if BUF_A_SEL for channel 7 is clear, then this bit is asserted = 1.
13	CH6BUFB	R/W	0	If CH_COMP_B for channel 6 is set and if BUF_B_SEL for channel 6 is clear, then this bit is asserted = 1.
12	CH6BUFA	R/W	0	If CH_COMP_A for channel 6 is set and if BUF_A_SEL for channel 6 is clear, then this bit is asserted = 1.
11	CH5BUFB	R/W	0	If CH_COMP_B for channel 5 is set and if BUF_B_SEL for channel 5 is clear, then this bit is asserted = 1.
10	CH5BUFA	R/W	0	If CH_COMP_A for channel 5 is set and if BUF_A_SEL for channel 5 is clear, then this bit is asserted = 1.
9	CH4BUFB	R/W	0	If CH_COMP_B for channel 4 is set and if BUF_B_SEL for channel 4 is clear, then this bit is asserted = 1.
8	CH4BUFA	R/W	0	If CH_COMP_A for channel 4 is set and if BUF_A_SEL for channel 4 is clear, then this bit is asserted = 1.
7	CH3BUFB	R/W	0	If CH_COMP_B for channel 3 is set and if BUF_B_SEL for channel 3 is clear, then this bit is asserted = 1.
6	CH3BUFA	R/W	0	If CH_COMP_A for channel 3 is set and if BUF_A_SEL for channel 3 is clear, then this bit is asserted = 1.
5	CH2BUFB	R/W	0	If CH_COMP_B for channel 2 is set and if BUF_B_SEL for channel 2 is clear, then this bit is asserted = 1.
4	CH2BUFA	R/W	0	If CH_COMP_A for channel 2 is set and if BUF_A_SEL for channel 2 is clear, then this bit is asserted = 1.
3	CH1BUFB	R/W	0	If CH_COMP_B for channel 1 is set and if BUF_B_SEL for channel 1 is clear, then this bit is asserted = 1.
2	CH1BUFA	R/W	0	If CH_COMP_A for channel 1 is set and if BUF_A_SEL for channel 1 is clear, then this bit is asserted = 1.
1	CH0BUFB	R/W	0	If CH_COMP_B for channel 0 is set and if BUF_B_SEL for channel 0 is clear, then this bit is asserted = 1.
0	CH0BUFA	R/W	0	If CH_COMP_A for channel 0 is set and if BUF_A_SEL for channel 0 is clear, then this bit is asserted = 1.

CHANNEL_x_CONTROL Register

Table 3-5 • **CHANNEL_x_CONTROL**

Bit Number	Name	R/W	Reset Value	Function
31:27	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:23	PERIPHERAL_SEL	R/W	0	Selects the peripheral assigned to this channel. See Table 3-6 on page 44.
22	Reserved	R/W	0	Reserved
21:14	WRITE_ADJ	R/W	0	This field contains a binary value, indicating the number of FCLK periods which the PDMA must wait after completion of a read or write access to a peripheral before evaluating the out-of-band status signals from that peripheral for another transfer. This is typically used to ensure that a posted write has fully completed to the peripheral in cases where the peripheral is running at a lower clock frequency than the PDMA. However, it can also be used to allow the PDMA to take account of internal latencies in the peripheral, where the ready status of a FIFO may not be available for a number of clock ticks after a read or write, due to internal synchronization delays, for example, within the peripheral. This applies particularly in the case of user-designed peripherals in the FPGA fabric.
13:12	DSTADDRINC	R/W	0	This field controls the destination address increment for the DMA transfer. The values have the following meanings: 0 = 0 byte, 1 = 1 byte, 2 = 2 bytes, 3 = 4 bytes.
11:10	SRCADDRINC	R/W	0	This field controls the source address increment for the DMA transfer. The values have the following meanings: 0 = 0 byte, 1 = 1 byte, 2 = 2 bytes, 3 = 4 bytes
9	HI_PRIORITY	R/W	0	When = 1, this channel is defined to be a high priority channel.
8	CLR_COMP_B	R/W	0	When asserted, clears the CH_COMP_B bit in the CHx_STATUS_REG and the BUFFER_STATUS for this buffer (B) in this channel x. This causes DMAINTERRUPT to negate if not being held asserted by another channel. This bit always reads back as zero.
7	CLR_COMP_A	R/W	0	When asserted, clears the CH_COMP_A bit in the CHx_STATUS_REG and the BUFFER_STATUS for this buffer (A) in this channel x. This causes DMAINTERRUPT to negate if not being held asserted by another channel. This bit always reads back as zero.
6	INTEN	R/W	0	When = 1, a DMA completion on this channel causes DMAINTERRUPT to assert. When = 0, DMA completions for this channel do not cause assertion of DMAINTERRUPT.
5	RESET	R/W	0	When = 1, resets this channel. Always read backs as 0.
4	PAUSE	R/W	0	When = 1, pauses the transfer for this channel until set to 0.
3:2	TRANSFER_SIZE	R/W	0	This field determines the data width of each DMA transfer cycle for this DMA channel. 0b00 = byte, 0b01 = halfword, 0b10 = word, 0b11 = reserved.

***Table 3-5* • CHANNEL_x_CONTROL (continued)**

Bit Number	Name	R/W	Reset Value	Function
1	DIR	R/W	0	If PERIPHERAL_DMA = 1, then this bit is valid. When = 0, transfers are from peripheral to memory. When = 1, transfers are from memory to peripheral.
0	PERIPHERAL_DMA	R/W	0	When = 0, this channel is configured for memory to memory DMA. When = 1, this channel is configured for peripheral DMA.

PERIPHERAL_SEL

***Table 3-6* • PERIPHERAL_SEL**

Bit 26	Bit 25	Bit 24	Bit 23	Function
0	0	0	0	From UART_0 receive to any MSS memory mapped address
0	0	0	1	From any MSS memory mapped address to UART_0 transmit
0	0	1	0	From UART_1 receive to any MSS memory mapped address
0	0	1	1	From any MSS memory mapped address to UART_1 transmit
0	1	0	0	From SPI_0 receive to any MSS memory mapped address
0	1	0	1	From any MSS memory mapped address to SPI_0 transmit
0	1	1	0	From SPI_1 receive to any MSS memory mapped address
0	1	1	1	From any MSS memory mapped address to SPI_1 transmit.
1	0	0	0	From to/from FPGA fabric peripheral DMAREADY1
1	0	0	1	From to/from FPGA fabric peripheral DMAREADY0
1	0	1	0	From any MSS memory mapped address to the ACE
1	0	1	1	From the ACE to any MSS memory mapped address

CHANNEL_x_STATUS Register

***Table 3-7* • CHANNEL_x_STATUS**

Bit Number	Name	R/W	Reset Value	Function
31:3	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	BUF_SEL	R/W	0	When = 0, buffer A is used. When = 1, buffer B is used.
1	CH_COMP_B	R/W	0	Asserts when this channel completes its DMA. Cleared by writing to CLR_COMP_B, bit 8 in CHx_CONTROL_REG for this channel. If INTEN is set for this channel, then the assertion of CH_COMP_B causes DMAINTERRUPT to assert.
0	CH_COMP_A	R/W	0	Asserts when this channel completes its DMA. Cleared by writing to CLR_COMP_A, bit 8 in CHx_CONTROL_REG for this channel. If CHx_INTEN is set for this channel, then the assertion of CH_COMP_A causes DMAINTERRUPT to assert.

CHANNEL_x_BUFFER_A_SRC_ADDR Register

***Table 3-8* • CHANNEL_x_BUFFER_A_SRC_ADDR**

Bit Number	Name	R/W	Reset Value	Function
31:0	BUF_A_SRC	R/W	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 0, then this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

CHANNEL_x_BUFFER_A_DST_ADDR Register

***Table 3-9* • CHANNEL_x_BUFFER_A_DST_ADDR**

Bit Number	Name	R/W	Reset Value	Function
31:0	BUF_A_DST	R/W	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 1, then this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

CHANNEL_x_BUFFER_A_TRANSFER_COUNT Register

***Table 3-10* • CHANNEL_x_BUFFER_A_TRANSFER_COUNT**

Bit Number	Name	R/W	Reset Value	Function
31:16	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	BUF_A_COUNT	R/W	0	Number of transfers remaining to be completed between source and destination for buffer A for this channel. This field is decremented after every DMA transfer cycle. Writing a non-zero value to this register causes the DMA to start. This must be the last register to be written by firmware when setting up a DMA transfer.

CHANNEL_x_BUFFER_B_SRC_ADDR Register

***Table 3-11* • CHANNEL_x_BUFFER_B_SRC_ADDR**

Bit Number	Name	R/W	Reset Value	Function
31:0	BUF_B_SRC	R/W	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 0, this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

CHANNEL_x_BUFFER_B_DST_ADDR Register

***Table 3-12* • CHANNEL_x_BUFFER_B_DST_ADDR**

Bit Number	Name	R/W	Reset Value	Function
31:0	BUF_B_DST	R/W	0	Start address from which data is to be read during the next DMA transfer cycle. If PERIPHERAL_DMA = 1 and DIR = 1 (peripheral to memory), then this value is not incremented from one DMA transfer cycle to the next. Otherwise, it is always incremented by an amount corresponding to the TRANSFER_SIZE for this channel.

CHANNEL_x_BUFFER_B_TRANSFER_COUNT Register

Table 3-13 • **CHANNEL_x_BUFFER_B_TRANSFER_COUNT**

Bit Number	Name	R/W	Reset Value	Function
31:16	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	BUF_B_COUNT	R/W	0	Number of transfers remaining to be completed between source and destination for buffer B for this channel. This field is decremented after every DMA transfer cycle. Writing a non-zero value to this register causes the DMA to start. This must be the last register to be written by firmware when setting up a DMA transfer.

4 – Embedded Nonvolatile Memory (eNVM) Controller

The embedded nonvolatile memories (eNVM) controller in SmartFusion devices consists of two components: the eNVMs and the eNVM controller. The eNVM controller converts logical AHB addresses to physical eNVM addresses and allows you to command the eNVM to perform specific tasks such as programming and erasing. The block diagram in Figure 4-1 shows the configuration of the eNVM controller with two eNVM blocks. Not all SmartFusion devices contain two memory blocks; for the SmartFusion device with only one eNVM, the controls signals for the nonexistent eNVM are ignored.

Note that x is used as a place holder in register names and field names within registers to indicate a particular eNVM (for example, ENVM_STATUS_x = ENVM_STATUS_0 or ENVM_STATUS_1).

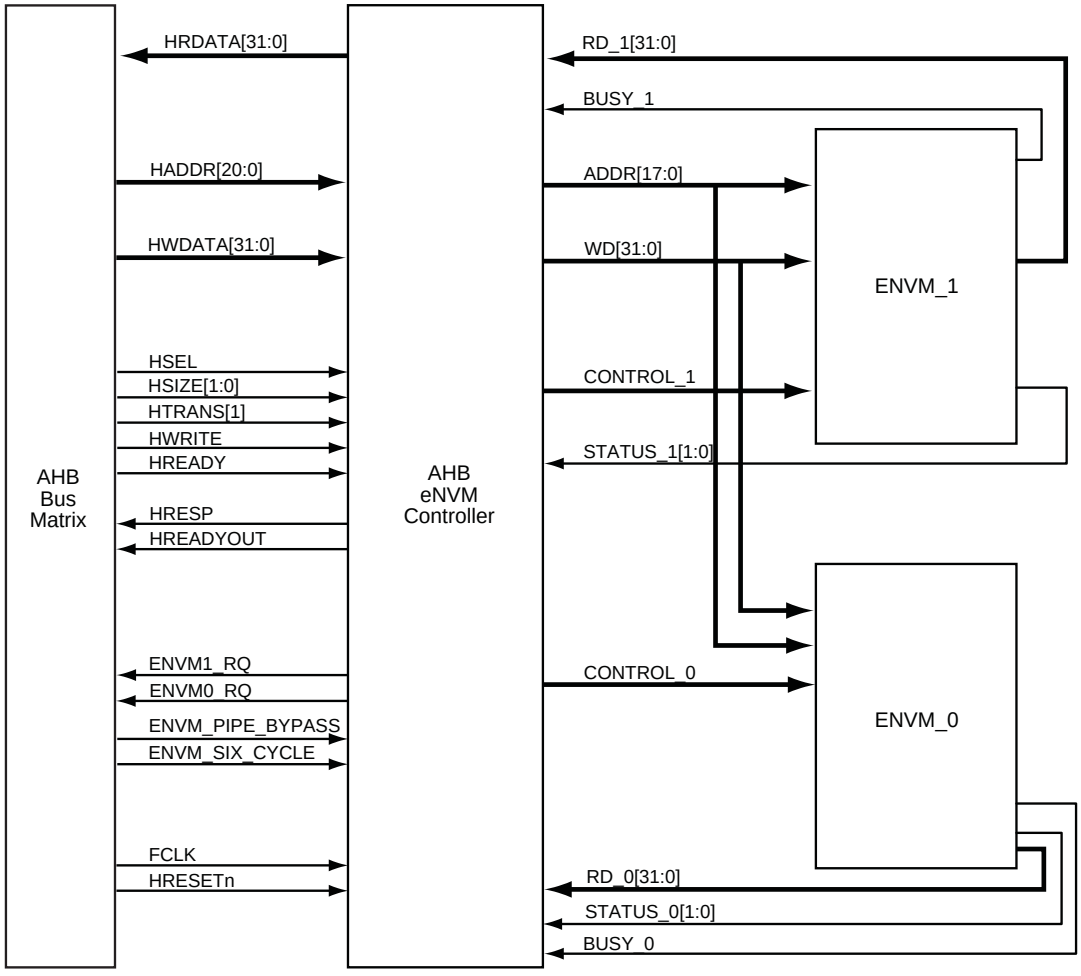


Figure 4-1 • **Block Diagram of eNVM Controller with Two eNVM Blocks**

The eNVM controller consists of the following sub-blocks:

- Flash array

  Contains all stored data. The flash array contains 64 sectors, and each sector contains 33 pages of data.
- Page buffer

  A page-wide volatile register. A page contains 8 blocks of data and an AUX block.
- Block buffer

  Contains the contents of the last block accessed. A block contains 128 data bits.
- ECC logic

  The FB stores error correction information with each block to perform single-bit error correction and double-bit error detection on all data blocks.

Figure 4-2 illustrates the block diagram of an individual eNVM and its associated control logic.

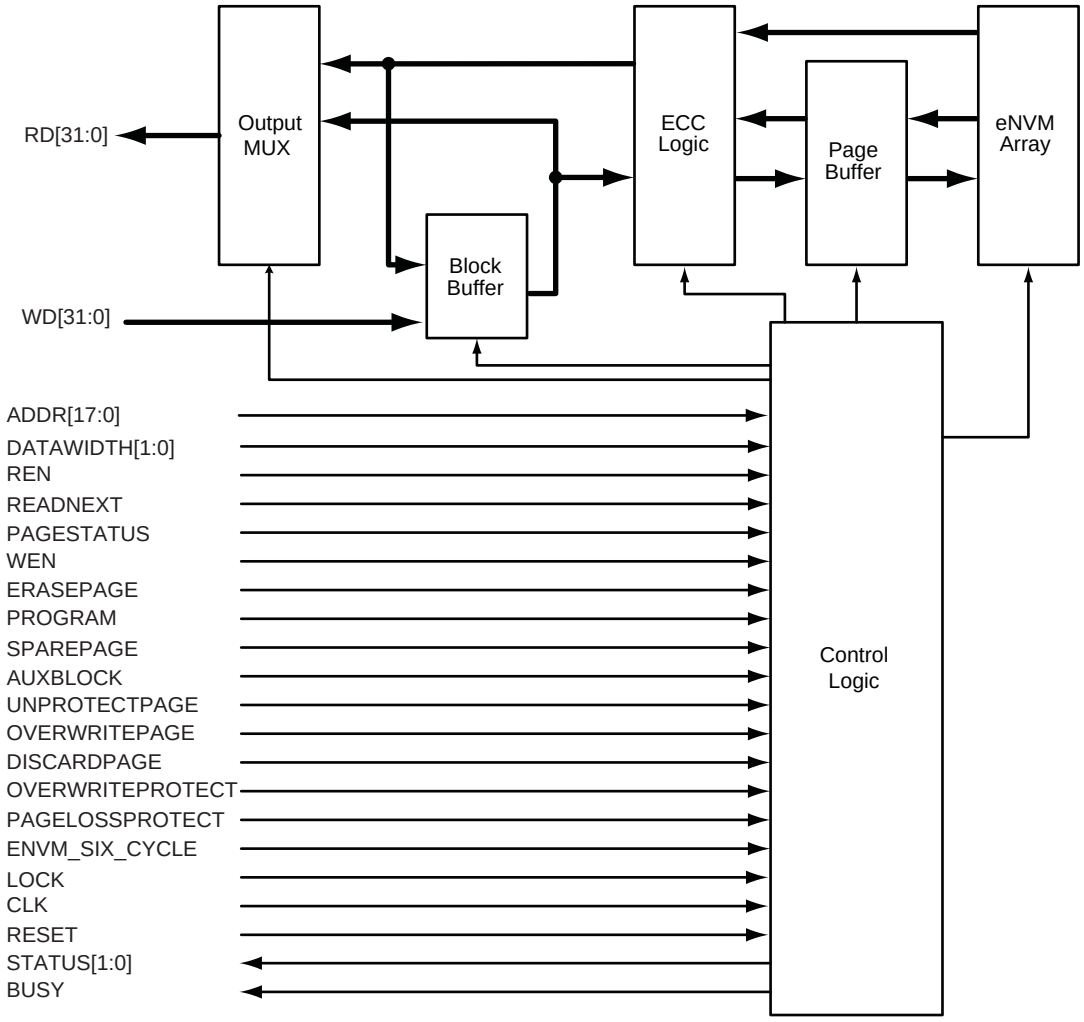


Figure 4-2 • **Block Diagram for eNVM Controller**

The eNVM controller uses a simple register-based command structure that allows all eNVM operations to be performed. All commands are initiated in a single AHB cycle (Address and Data phases) and perform a single operation to the eNVM. Reads hold the AHB busy (HREADYOUT) until they complete, but all writes are posted and completed independent of the AHB bus. If a new operation is started when the addressed eNVM is busy, HREADYOUT is deasserted for the new operation and it starts when the

eNVM is ready. This can cause the AHB bus to be busy for an extended period of time, especially if the current operation is a lengthy one, such as program or erase. To avoid this, read the ENVM_STATUS_REG register and verify that BUSY_x for the desired eNVM is clear before starting a new operation on that memory block.

Memory Organization

Figure 4-3 depicts the physical organization of a single eNVM. eNVMs are organized by sectors, pages, blocks, and bytes. Each sector contains 32 pages and 1 spare page. Each page contains 8 data blocks and 1 auxiliary block. Each data block contains 16 bytes of data, with the auxiliary block containing an additional 4 bytes of data.

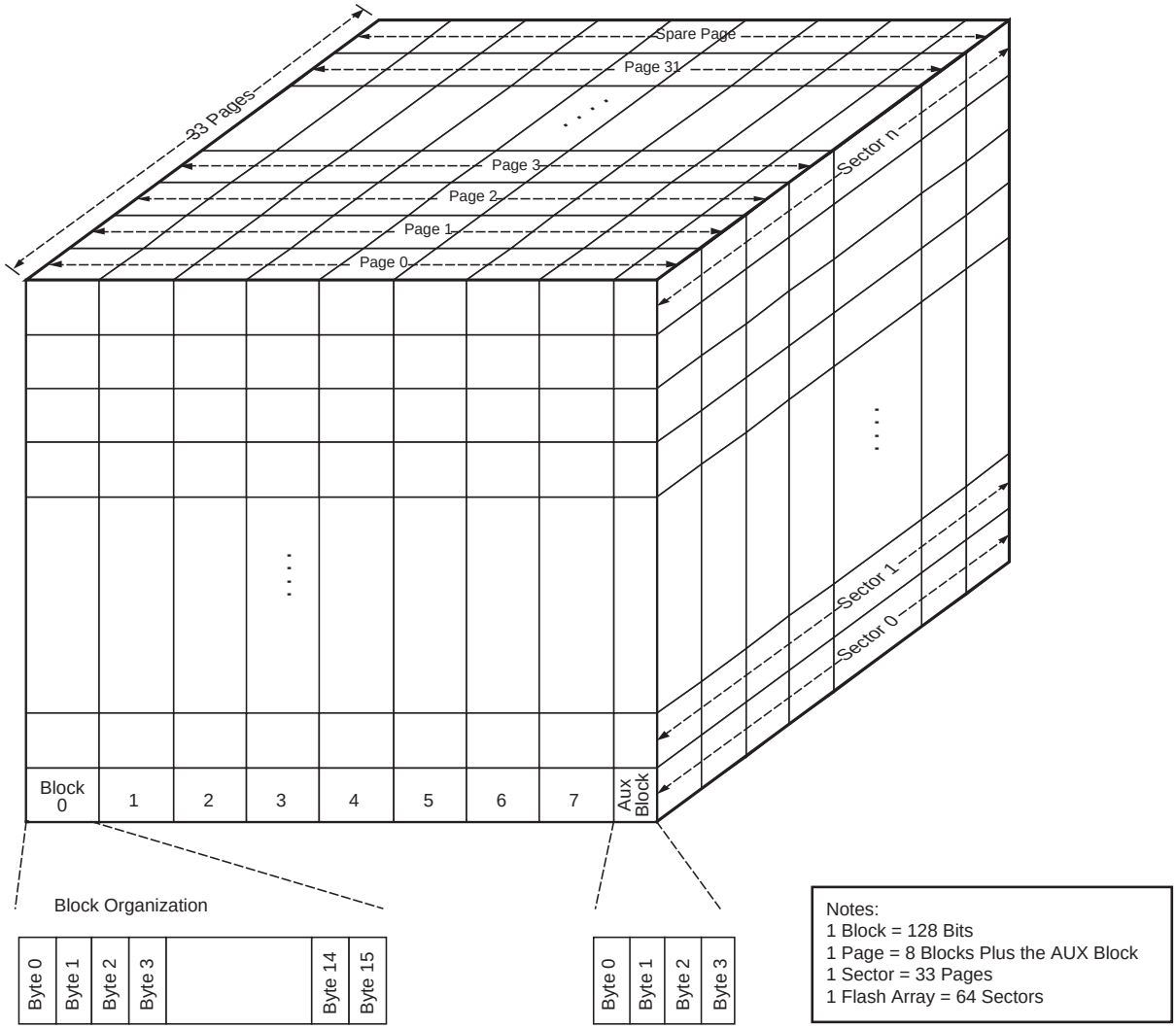


Figure 4-3 • **eNVM Organization**

From the programmer's perspective, as depicted in the memory map in Table 4-1 on page 52, the eNVMs are logically split into four address spaces: the eNVM array, eNVM spare pages, eNVM Auxiliary (Aux) block (array), and eNVM Aux block (spare pages). The spare page 63 of the eNVM is not available to the user and always reads 0. The spare pages in sectors 0 – 16 in the eNVM are used to store factory boot code and manufacturing parameters. These pages are write protected. For SmartFusion devices with two eNVM blocks (A2F500), the spare page in sector 63 of the additional eNVM is also unavailable to the user; however, the spare pages in sectors 0-16 of the additional eNVM are writable.

Table 4-2 lists the contents of a portion of the spare pages.

Table 4-1 • **eNVM Physical Memory Map**

Cortex-M3 Memory Map	Memory Map of FPGA Fabric Master, Ethernet MAC, Peripheral DMA	Address Range
		0xE0043000 – 0xFFFF2FFF
System Registers		0xE0042000 – 0xE0042FFF
		0x78000000 – 0xE0041FFF
External Memory Type 1	External Memory Type 1	0x74000000 – 0x77FFFFFF
External Memory Type 0	External Memory Type 0	0x70000000 – 0x73FFFFFF
		0x601D0000 – 0x6FFFFFFF
		0x60180000 – 0x601CFFFF
		0x60100100 – 0x6017FFFF
eNVM Controller	eNVM Controller	0x60100000 – 0x601000FF
		0x60088200 – 0x600FFFFF
eNVM Aux Block (spare pages)	eNVM Aux Block (spare pages)	0x60088000 – 0x600881FF
eNVM Aux Block (array)	eNVM Aux Block (array)	0x60084000 – 0x60087FFF
eNVM Spare Pages	eNVM Spare Pages	0x60080000 – 0x60083FFF
eNVM Array	eNVM Array	0x60000000 – 0x6007FFFF
		0x44000000 – 0x5FFFFFFF
Peripherals (bit band view)		0x42000000 – 0x43FFFFFF
		0x40100000 – 0x41FFFFFF

Table 4-2 • **Spare Page Contents**

Description	Size (bytes)	Size (spare pages)	Address Range
Generic initial data blocks and PPE RAM merge operations	Dependent on user design		0x600816CC – 0x60081F7F
MSS configuration	180	2	0x60081618 – 0x600816CB
Analog block configuration	24		0x60081600 – 0x60081617
System boot	3,072	28	0x60080800 – 0x600815FF
Factory boot	1,024	8	0x60080400 – 0x600807FF
Manufacturing parameters	576	8	0x60080000 – 0x600803FF
ARM Cortex-M3 vector table	16		

The eNVM address bus decodes sectors, pages, blocks, and bytes, as shown in Figure 4-4.

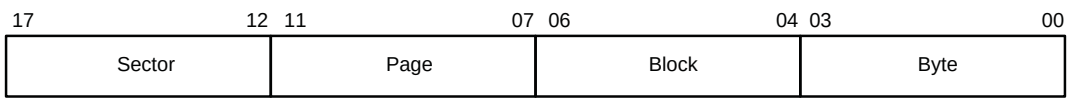


Figure 4-4 • **Sector, Page, Block, and Byte Addressing Scheme**

Table 4-3 depicts the size of the various eNVM sections in bytes for each SmartFusion family member. There are two physical eNVM blocks in the A2F500 that are logically mapped into Cortex-M3 memory space as one eNVM. The eNVM controller manages mapping the two separate memory blocks as one logical contiguous memory for all sections, eNVM array, spare pages, Aux block array, and Aux block spare pages. Note, however, when reading from ENVM_0, the spare page in sector 63 returns all zeros. For those devices with two eNVM blocks, reading from ENVM_1, the spare page in sector 63 also returns zeros. Register or bit descriptions that follow will indicate which eNVM is affected by a user operation. For example, BUSY_0 indicates ENVM_0 is busy. In devices with two eNVMs, each memory section is split logically in two, with the bottom half addressing ENVM_0 and the top half of the memory section addressing ENVM_1. For example, the main eNVM array for ENVM_0 occupies address space 0x60000000 – 0x6003FFFF and the main eNVM array for ENVM_1 occupies address space 0x60040000 – 0x6007FFFF.

The SmartFusion MSS configurator uses a certain number of user eNVM pages to store the MSS configuration. These pages are located at the top of the eNVM address space. The number of pages is variable based on the MSS configuration (ACE, GPIOs, and eNVM Init Clients). Application code should not write in these user pages as it will most likely cause a runtime failure for your design. Note also that if these pages have been inadvertently corrupted, the part will not boot again and will need to be reprogrammed.

The first reserved address can be computed as follows. After the MSS has been successfully generated, open the eNVM configurator and record the number of available pages shown in the Usage Statistics group on the main page. The first reserved address is defined as:

first_reserved_address = 0x60000000 + (available_pages * 128)

Table 4-3 • **eNVM Section Sizes**

Device	Sectors	eNVM Array Bytes	Spare Pages Bytes	Aux. Block (array) Bytes	Aux. Block (spare pages) Bytes	Total eNVM Bytes
A2F500	128	52,4288	16,384	16,384	512	557,568
A2F200	64	262,144	8,192	8,192	256	278,784
A2F060	–	–	–	–	–	–

Read Control

Read operations for the eNVM can read from the block buffer, the page buffer, or the eNVM array. Sector and page boundaries are not important when reading from the eNVM because the block address can be assumed to be contained in address bits 17:4, as shown in Figure 4-5. Read timing depends solely on block address boundaries. Instructions/data are presented to the AHB bus 32 bits at a time. Depending on compiler optimizations either one or 2 instructions are fetched at a time.

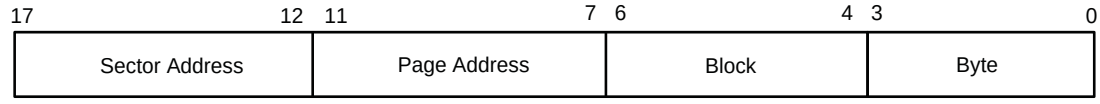


Figure 4-5 • **Address Decoding for eNVM Read Operations**

- If the block addressed by a read operation is the same as that of the previous read or write operation, the data is read from the block buffer.
- If the block addressed by a read operation has changed since the previous read or write operation but the page addressed is the same as that of the previous write operation or unprotect page operation, the read data originates from the page buffer.
- If the page addressed by a read operation is not the same as that of the previous write operation, data is fetched from the eNVM array. The page buffer is not modified.

A read always reads the latest data written to the eNVM, whether the data resides in the block buffer, page buffer, or eNVM array.

The ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE registers are used to control read access behavior to the eNVM. The latency of the initial access to an eNVM block and the subsequent three accesses, if initiated, to the same eNVM block depends on the state of both ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE. The latencies (number of FCLK cycles) corresponding to the various combinations of ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS are shown in Table 4-4. ENVM_SIX_CYCLE (bit 07) and ENVM_PIPE_BYPASS (bit 06) are controlled by the ENVM_CR located at address 0xE0042004.

Table 4-4 • **Latencies Corresponding to ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS**

ENVM_SIX_CYCLE	ENVM_PIPE_BYPASS	eNVM Access FCLK Cycles
0	1	5:1:1:1
1	1	6:1:1:1 (default)

Note: 6:1:1:1 indicates 6 cycles for the first access and 1 each for the next three accesses. 5:1:1:1 indicates 5 cycles for the first access and 1 each for the next three accesses.

In 5:1:1:1 read mode, a newly addressed block is fetched from the eNVM array, and presented to the output multiplexer and copied into the block buffer simultaneously. In 5:1:1:1 read mode (ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 1), the first read comes directly from the eNVM array. The following reads, up to three sequentially, originate from the block buffer and occur in single cycles. The read data path for this mode is illustrated in Figure 4-6 (the dotted line shows the read data path).

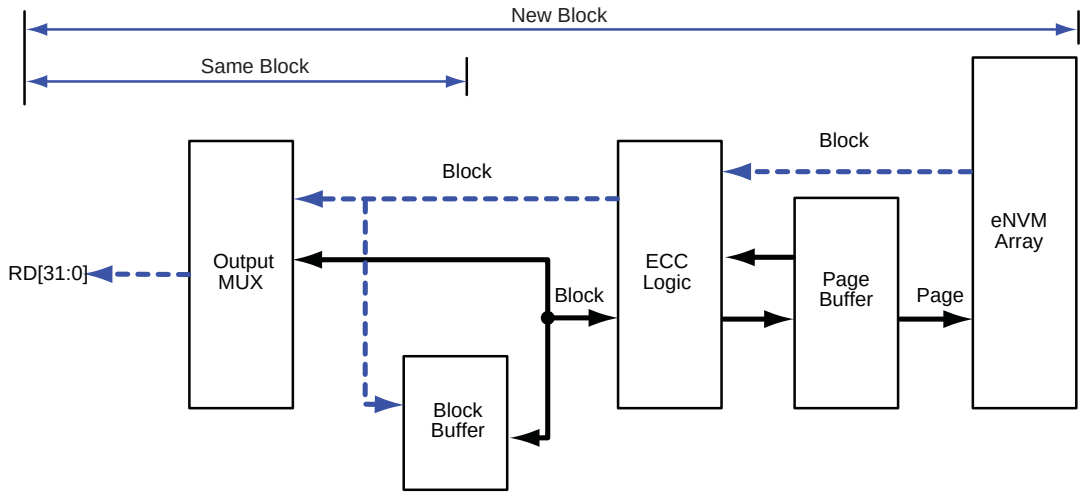


Figure 4-6 • **Five-Cycle Read Data Path, ENVM_SIX_CYCLE = 0**

The timing diagram for this mode is illustrated in Figure 4-7.

Note: AHB signals are sampled on the rising edge of FCLK.

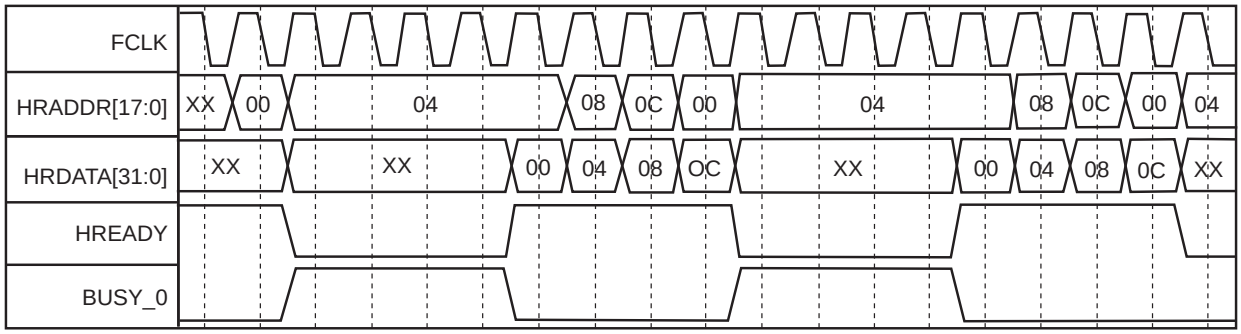


Figure 4-7 • **eNVM Read: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 1, SEQ or NONSEQ Block Address (5:1:1:1)**

When the ENVM_PIPE_BYPASS bit is cleared to 0, the first read takes 6 cycles and the subsequent three reads insert a single-cycle AHB pipeline delay in the read data path. Figure 4-8 illustrates timing when ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE are both 0.

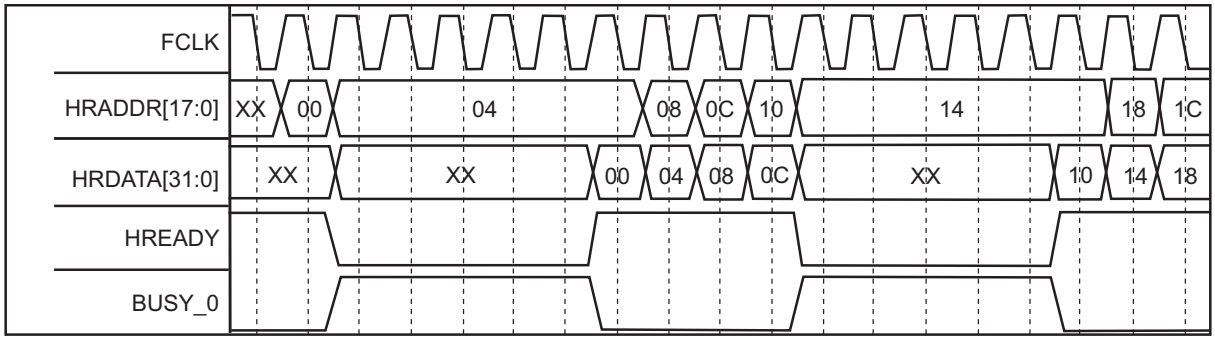


Figure 4-8 • **eNVM Read: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 0, SEQ or NONSEQ Block Address (6:1:1:1)**

Read Next Operation

The Read Next operation reads the next sequential block in the eNVM array relative to the current block in the block buffer while the block buffer is being read. The goal is to minimize wait states during consecutive sequential block read operations.

The Read Next operation is performed in a predetermined manner because it does look-ahead reads. The general look-ahead function is as follows:

- Within a page, the next block fetched will be the current block address + 1.
- When reading the last data block of a page, it will fetch the first block of the next page.
- When reading spare pages, it will read the first block of the next sector's spare page.
- Reads of the last sector will wrap around to sector 0.
- Reads of Auxiliary blocks will read the next linear page's Auxiliary block.

When a block address becomes non-sequential, the current read operation must complete. The time penalty for this access is anywhere from 9 to 12 cycles, depending on how ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS are set and whether or not the block buffer has drained completely.

If the next block to be addressed is the current block address + 1 and the block buffer is completely drained, the delay between block reads is one cycle. For example, if you read only one data/instruction

from the block buffer, there are four cycles of busy, as shown in Table 4-5. As the block buffer is being read, the eNVM controller is simultaneously reading the next block from the eNVM array. Once initiated, this transfer must complete, hence the extra delay if the block buffer is not completely drained. Read Next mode allows you access to the entire eNVM array in a high-speed pipelined sequential fashion, as indicated in Figure 4-9 on page 56 and Figure 4-10 on page 56. The same functionality pertains to the spare pages section, Aux block (array) section, and Aux block (spare pages) section when READ_NEXT is set.

Table 4-5 • **Busy Cycles Between Consecutive Block Reads When READ_NEXT = 1 and Block Buffer is Not Drained**

Number of Reads from Block Buffer	HREADY Low Cycles	
	ENVM_PIPE_BYPASS = 1	ENVM_PIPE_BYPASS = 0
1	4	4
2	3	2
3	2	1
4	1	1

Read Next operation is enabled by setting the READ_NEXT bit in the ENVM_x_CR (x = 0 or 1) register. For SmartFusion devices with two eNVM blocks, it is possible to have one eNVM in Read Next mode and the other in normal Read mode. Read Next mode can be modified dynamically.

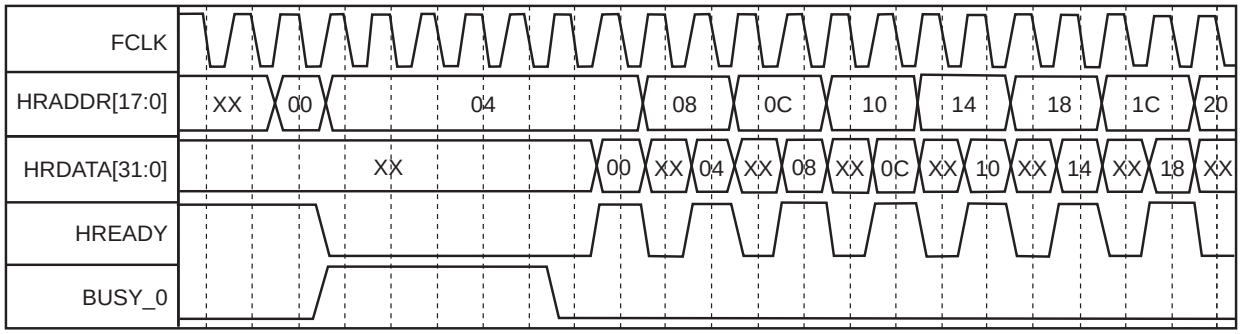


Figure 4-9 • **eNVM Read Next Enabled: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 0, SEQ Block Address**

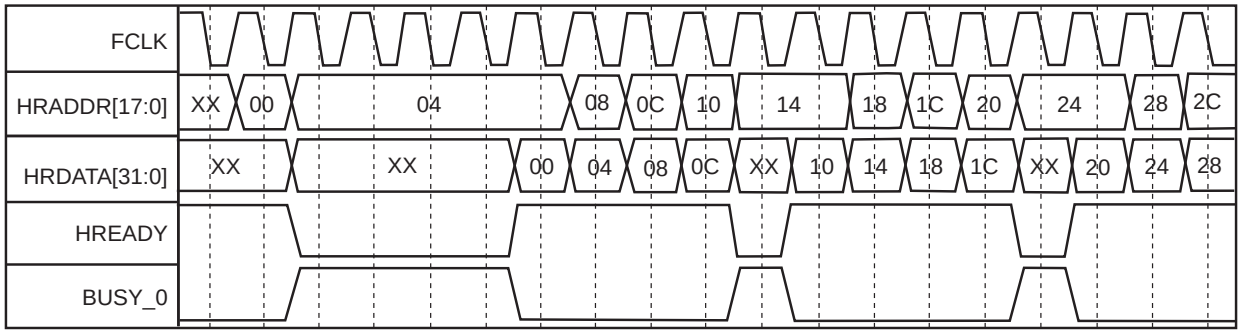


Figure 4-10 • **eNVM Read Next Enabled: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 1, SEQ Block Address**

Write Operations

All program and erase operations to the eNVM occur from the page buffer. Writes to the page buffer can be byte, halfword, or words. Writing to the eNVM's page buffer does not start a program or erase operation. Specific command sequences in the eNVM controller memory space must be issued to start a program or erase operation. These commands are listed in Table 4-8 on page 63. The eNVM Controller will capture the sector address and page address when a write to the eNVM occurs and store those addresses within the page buffer, as shown in Figure 4-11. The sector address and the page address stored in the page buffer are then used to confirm that the page being modified is the desired page to program into the eNVM when a program or erase operation is commanded.

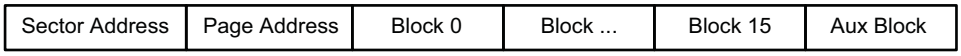


***Figure 4-11* • Page Buffer**

The page buffer is marked internally as being modified if one of the following conditions is true:

- A successful write operation to the page buffer occurs.
- A successful write operation to the Auxiliary block occurs.
- If the state of protection for the page buffer has been modified.

The internal flag indicating whether a page is modified or not is used by programming and erase commands to ensure coherency between the page buffer and the eNVM. The page buffer is marked as unmodified when it is committed to eNVM or discarded.

Programming the eNVM is accomplished using a Read-Modify-Write methodology. The first write causes the eNVM controller to copy the entire page from the eNVM array, place it into the page buffer, and write the first word into the block buffer. Figure 4-12 depicts the flow of data from eNVM to the block buffer and to the page buffer during this operation.

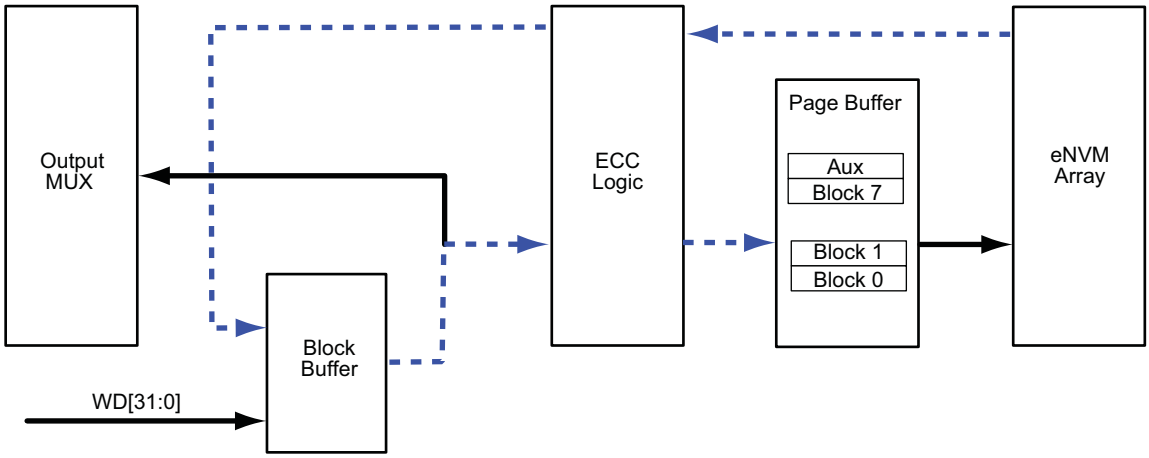


***Figure 4-12* • Copy Page Data Flow**

While the copy takes place, which could be many cycles, BUSY_x is asserted. You should check the BUSY_x status from the eNVM where the operation is occurring before continuing to write to the block buffer. Subsequent writes to the same block in the page buffer take no additional BUSY cycles. Once the block buffer is loaded, you can read from it or write to it at will. Care must be taken that sector and page addresses do not change during these read or write operations because that will cause the eNVM controller to fetch the newly addressed page from the eNVM and load the block buffer / page buffer with it. This could lead to inadvertently programming the wrong page in the eNVM, unless page loss protection is enabled.

Writes to a different block in the same page will assert BUSY_x for four cycles, since the current block buffer is synchronized to the page buffer. The data flow for this operation is shown in Figure 4-13. Once the page buffer has been updated, either a PROGRAM_PAGE command or PROGRAM_PAGE_PROTECTED command can be issued to synchronize the eNVM and the page buffer.

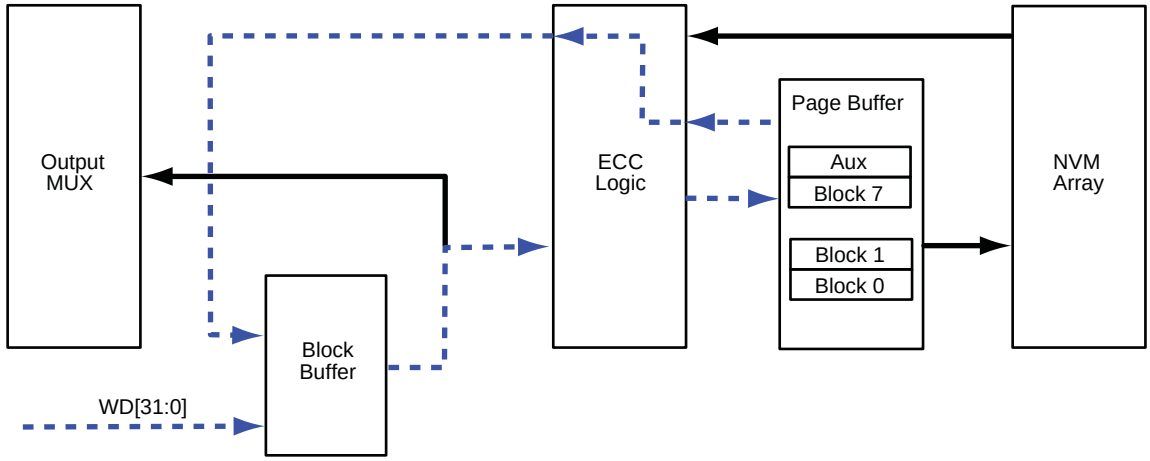


***Figure 4-13* • Modify Page Data Flow**

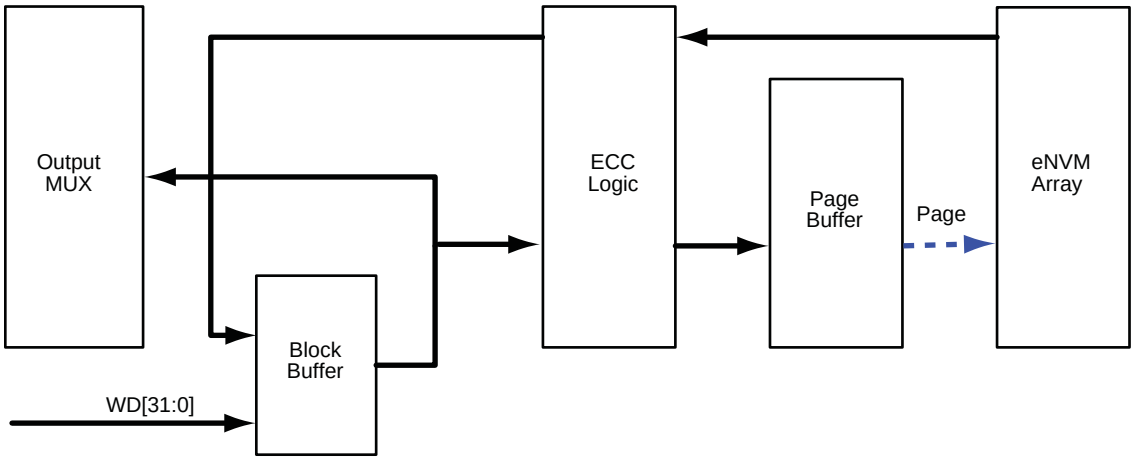


***Figure 4-14* • Program/Erase Data Flow**

There are protection mechanisms to prevent the accidental copy of a page buffer to the wrong page in eNVM. For example, if the PAGE_LOSS bit is set and the sector or page address is changed after the first write to the page buffer, and a program command is then issued, the operation will fail.

The PROT_ERROR_x bit is set in the ENVM_STATUS_REG register, indicating a protection fail has occurred. An interrupt signal, if enabled, will be asserted to the Cortex-M3 NVIC.

Reading/Writing to the Aux Block section(s)

When reading or writing to either the Aux block or Aux block spare pages section, the individual 4-byte Auxiliary blocks are mapped contiguously in Cortex-M3 space, as shown in Figure 4-15 and Figure 4-16. Reading and writing to these sections has the same functionality as reading and writing to the main eNVM array.

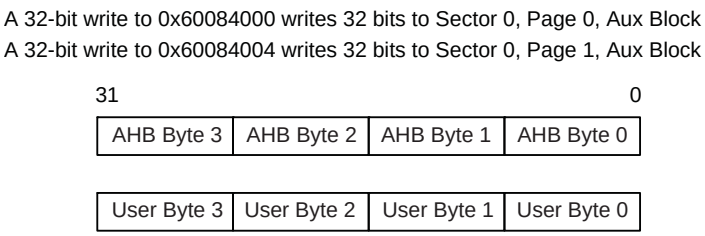


***Figure 4-15* • Aux Block Memory Mapping**

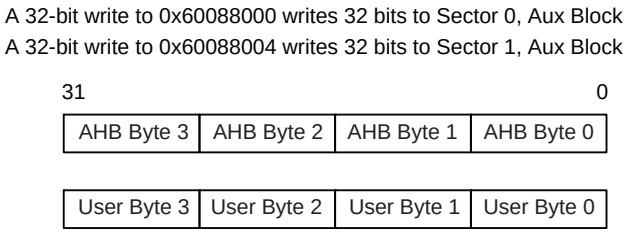


***Figure 4-16* • Aux Block Spare Pages Memory Mapping**

A program of either the Aux block or Aux block spare pages section to eNVM performs a program of the corresponding eNVM page. The contents of the page are preserved; however, the program does count against the eNVM's endurance budget. You can update the eNVM page, then the associated Aux block (or the eNVM spare page and associated spare page Aux block), before programming the page to preserve eNVM endurance.

eNVM Block Protection

Page Loss Protection

When the PAGE_LOSS bit is set to logic 1, the eNVM controller prevents writes to any page other than the current page in the page buffer until that page is either discarded or programmed in the eNVM cell array. Addressing any other page while the current page is page loss protected will return an ENVM_STATUS_x of 11, set the appropriate PROT_ERROR_x bit in the ENVM_STATUS_REG, and assert an interrupt signal to the Cortex-M3 NVIC if the PROT_ERROR_x bit in the ENVM_ENABLE_REG is set.

Page Protection

Any page that is write protected will result in the ENVM_STATUS_x being set to 01 when an attempt is made to write, program, or erase it. To write protect a page, use the PROGRAM_PAGE_PROTECTED or ERASE_PAGE_PROTECTED command. To temporarily clear the protection state for a given page, and allow modification of the page buffer, issue an UNPROTECT_PAGE command on the desired page.

LOCK

The LOCK bit is used to give you control over access to the eNVM from the JTAG interface. When LOCK is asserted, the JTAG interface will be prevented from any access attempts to the eNVM until LOCK is deasserted. For example, if a fabric master has access to the eNVM and does not want a JTAG operation to command or control the eNVM, the fabric master should set the LOCK bit. Likewise, if you only allow eNVM access via the Cortex-M3 microcontroller and want to prevent the JTAG interface from accessing the eNVM, you should set the LOCK bit.

eNVM Commands

Table 4-6 lists the various commands available for controlling the behavior of the eNVM. Program and erase operations on the eNVM occur on a page boundary.

Table 4-6 • **eNVM Commands**

Command	Address/Data Bus			
	Op.	**ADDR**	**Data[31:24]**	**Data[23:0]**
ARRAY_READ	Read	eNVM array address	eNVM array data	
ARRAY_WRITE	Write	eNVM array address	eNVM array data	
UNPROTECT_PAGE	Write	ENVM_CONTROLLER_REG	0x02	Page address
DISCARD_PAGE	Write	ENVM_CONTROLLER_REG	0x04	Page address
PROGRAM_PAGE	Write	ENVM_CONTROLLER_REG	0x10	Page address
PROGRAM_PAGE_PROTECTED	Write	ENVM_CONTROLLER_REG	0x11	Page address
ERASE_PAGE	Write	ENVM_CONTROLLER_REG	0x20	Page address
ERASE_PAGE_PROTECTED	Write	ENVM_CONTROLLER_REG	0x21	Page address
OVERWRITE_PAGE	Write	ENVM_CONTROLLER_REG	0x50	Page address
GET_PAGE_STATUS	Write	ENVM_CONTROLLER_REG	0x88	Page address
NOP	Write	ENVM_CONTROLLER_REG	0x00	Page address

UNPROTECT_PAGE Command

Writing 0x02 to the COMMAND field of the ENVM_CONTROL_REG will unprotect the page addressed. The page addressed will be copied from eNVM into the page buffer if the page is not in the page buffer and the current contents of the page buffer are not marked as modified. The page addressed will also be copied from eNVM into the page buffer if the page in the page buffer is marked as modified and PAGE_LOSS = 0. If the contents of the page buffer are marked as modified and PAGE_LOSS = 1, the copy of the page from eNVM will not occur and a protection violation error will be reported in the ENVM_STATUS_REG register by setting PROT_ERROR_x to 1. An interrupt signal is asserted to the Cortex-M3 NVIC if the PROT_ERROR_x bit is set in the ENVM_ENABLE_REG.

If the page addressed is a read protected spare page (0-15) in ENVM_0, the UNPROTECT_PAGE operation does not occur and a protection violation error will be reported in the ENVM_STATUS_REG register by setting PROT_ERROR_x to 1.

Table 4-7 on page 61 summarizes information on the UNPROTECT_PAGE command.

Table 4-7 • UNPROTECT_PAGE modes

PAGE_LOSS	Page Buffer Modified	Page Copied from eNVM to Page Buffer	OVERWRITE_PROTECTED	PROT_ERROR_x
0	No	Yes	Set to 0	0
0	Yes	Yes; overwrites page buffer.	Set to 0	0
1	No	Yes	Set to 0	0
1	Yes	No	Unchanged	1

DISCARD_PAGE Command

Writing 0x04 to the COMMAND field of the ENVM_CONTROL_REG discards the data in the current page buffer. No interrupt is generated. Check the ENVM_STATUS_x field of the ENVM_STATUS_REG to ensure the operation completed (ENVM_STATUS_x = 00).

PROGRAM_PAGE Command

Writing 0x10 to the COMMAND field of the ENVM_CONTROL_REG programs the current page buffer to the page in eNVM addressed by PAGE_ADDRESS and leaves the page unprotected. The page in eNVM is automatically erased prior to the copy from page buffer to eNVM. The following conditions apply:

- Attempting to program a protected page using the PROGRAM_PAGE command will result in a protection error. The PROT_ERROR_x bit in the ENVM_STATUS_REG is set to 1.
- Attempting to program a different page in eNVM from the page currently contained in the page buffer while PAGE_LOSS = 1 will result in a protection error. The PROT_ERROR_x bit in the ENVM_STATUS_REG set to 1.
- Performing a PROGRAM_PAGE command increments the write count for that page by one.

PROGRAM_PAGE_PROTECTED Command

Writing 0x11 to the COMMAND field of the ENVM_CONTROL_REG programs the current page buffer to the page in eNVM addressed by PAGE_ADDRESS and leaves the page write protected. This is the same command as PROGRAM_PAGE, except the page is write protected in the process. Also, if you attempt to program a different page from the one in the page buffer with PAGE_LOSS = 0, the new page will be loaded in the page buffer and programmed with its protection bit set.

Performing a PROGRAM_PAGE_PROTECTED command increments the write count for that page by one.

ERASE_PAGE Command

Writing 0x20 to the COMMAND field of the ENVM_CONTROL_REG erases the eNVM page addressed by PAGE_ADDRESS. If the page addressed by Page Address does not match the page in the page buffer and the PAGE_LOSS bit is set, the erase fails. The erase also fails if the page is protected. Performing an ERASE_PAGE command increments the write count for that page by one.

ERASE_PAGE_PROTECTED Command

Writing 0x21 to the COMMAND field of the ENVM_CONTROL_REG erases the current protected page addressed by Page Address and leaves the page write protected. The page must be in an unprotected state prior to issuing this command. Erasing a protected page will flag a protection error. This is the same command as ERASE_PAGE, except the page is write protected in the process. Performing an ERASE_PAGE_PROTECTED command increments the write count for that page by one.

OVERWRITE_PAGE Command

Writing 0x50 to the COMMAND field of the ENVM_CONTROL_REG overwrites the page addressed by Page Address with the contents of the page buffer. If the destination page is protected, no overwriting of the page occurs and the PROT_ERROR_x bit is set in the ENVM_STATUS_REG register. This command can be used to move one eNVM page from one location to another. Performing an OVERWRITE_PAGE command increments the write count for that page by one.

GET_PAGE_STATUS Command

Writing 0x88 to the COMMAND field of the ENVM_CONTROL_REG retrieves the page status of the page addressed by Page Address and stores the status in the ENVM_PAGE_STATUS_x_REG register. The status bits remain valid for the page that was just issued until another GET_PAGE_STATUS command is completed, at which time the status bits reflect the state of the newly addressed page.

NOP Command

The NOP command does nothing. This command can be used to clear the COMMAND field.

Programming Errors

Program operations that result in an ENVM_STATUS_x value of 01 do not modify the addressed page. For all other values of ENVM_STATUS_x, the addressed page is modified.

Program errors include the following:

1. Attempting to program a page that is write protected (ENVM_STATUS_x = 01)
2. Attempting to program a page that is not in the page buffer when the page buffer has entered page loss protection mode (ENVM_STATUS_x = 01)
3. Attempting to perform a program with the OVERWRITE_PAGE command when the page addressed has been write protected (ENVM_STATUS_x = 01)
4. The write count of the page programmed exceeding the write threshold defined in the part specification (ENVM_STATUS_x = 11)
5. The ECC logic determining that there is an uncorrectable error within the programmed page (ENVM_STATUS_x = 10)
6. Attempting to program a page that is **not** in the page buffer when the OVERWRITE_PAGE command has been issued and the page in the page buffer is modified (ENVM_STATUS_x = 01)
7. Attempting to program the page in the page buffer when the page buffer is **not** modified

The contents of Table 4-8 indicate which eNVM commands can set the status bits within the ENVM_STATUS_REG register. For example, if there is a protection violation when issuing a PROGRAM_PAGE command, the PROT_ERROR bit will be set and the ENVM_STATUS_0 bits will be equal to 01.

Table 4-8 • **eNVM Commands that Set the eNVM Status Bits**

	ENVM_STATUS_REG							
	PROT_ERROR_x	PROT_ERROR_x	PROG_ERROR_x	ERASE_ERROR_x	OVER_THRESHOLD_x	ECC1_ERROR_x	ECC2_ERROR_x	OP_DONE_x
ARRAY_READ	–	–	–	–	–	01	10	–
ARRAY_WRITE	01	11	–	–	–	–	–	–
UNPROTECT_PAGE	–	11	–	–	–	01	10	–
DISCARD_PAGE	–	–	–	–	–	–	–	–
PROGRAM_PAGE	01	–	10	–	11	–	–	00
PROGRAM_PAGE_PROTECTED	01	–	10	–	11	–	–	00
ERASE_PAGE	01	–	–	10	11	–	–	00
ERASE_PAGE_PROTECTED	01	–	–	10	11	–	–	00
OVERWRITE_PAGE	01	–	10	–	11	–	–	00
GET_PAGE_STATUS	–	–	–	–	–	01	01	–
NOP	–	–	–	–	–	–	–	–

Clocks

The eNVMs are driven from the AHB bus matrix FCLK. On power-up, the default clock sourcing FCLK is the 100 MHz RC oscillator divided by 4, or 25 MHz. If another clock frequency is desired, you must configure the PLL accordingly. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109.

Resets

The eNVM controller resets to zero on power-up and is released as soon as PORRESET_N deasserts. You have the option under software control to reset the eNVM controller by writing to the System Registers located on the private peripheral bus of the Cortex-M3 microcontroller. Specifically, System Register SOFT_RST_CR is located at address 0xE0042030 in the memory map. The ENVM_SOFTRESET control bit is encoded in bit location 0, as shown in Table 4-9.

Table 4-9 • **ENVM_SOFTRESET Control Bit**

Bit 6	Function
0	eNVM controller reset released (reset value).
1	eNVM controller held in reset.

Interrupts

There is one interrupt signal per eNVM block that can be asserted, based on the result of operations on the eNVM(s). IRQ7 is asserted when ENVM_0's ENVM0_INT signal is raised and IRQ8 is asserted when ENVM_1's (if it exists) ENVM1_INT signal is raised. Interrupts must be enabled for the particular response you are trying to trap within the eNVM controller by setting the appropriate bits in the ENVM_ENABLE_REG and also by setting the appropriate bits within the Cortex-M3 NVIC. Both interrupt enable bits within the NVIC are located at address 0xE000E100; IRQ7 and IRQ8 correspond to bit locations 7 and 8 respectively. Even if interrupts are disabled and the ECC2_ERROR_x status bit is set, the HRESP signal on the AHB bus matrix will assert. If the bus master accessing the eNVM is the Cortex-M3 microcontroller, the hard fault exception vector will execute.

eNVM Controller Register Map

The eNVM controller control registers are located in the System Registers address space at 0x60100000 and extend to address 0x601000FF in the Cortex-M3 memory map. Refer to Figure 2-4 on page 25.

Table 4-10 • **eNVM Controller Register Map**

Name	Address	R/W	Reset Value	Description
ENVM_STATUS_REG	0x60100000	R/W	0x0	Returns the status of the last commanded operation.
ENVM_CONTROL_REG	0x60100004	R/W	0x0	Control register used for all eNVM commands
ENVM_ENABLE_REG	0x60100008	R/W	0x0	eNVM interrupt enable register
Reserved	0x6010000C	R/W	0x0	Reserved
ENVM_0_CR	0x60100010	R/W	0x0	eNVM_0 configuration register
ENVM_1_CR	0x60100014	R/W	0x0	eNVM_1 configuration register
ENVM_PAGE_STATUS_0_REG	0x60100018	R	0x0	eNVM_0 page status register
ENVM_PAGE_STATUS_1_REG	0x6010001C	R	0x0	eNVM_1 page status register

eNVM Status Register

Table 4-11 • **ENVM_STATUS_REG**

Bit Number	Name	R/W	Reset	Description
31	ILLEGAL_CMD_1	R/W	0	0 = "Don't care." 1 = An illegal command has been issued to ENVM_1. Write a 1 to this location to clear the bit.
30:26	Reserved	R	0	Read as 0.
25:24	ENVM_STATUS_1	R	0	These bits provide status information from ENVM_1 based upon the command and/or write that was issued to the eNVM. These are read only bits; writes have no effect. See Table 4-12 on page 67.
23	OP_DONE_1	R/W	0	0 = "Don't care." 1 = ENVM_1 has completed the commanded operation. Write a 1 to this location to clear the bit.
22	ECC2_ERROR_1	R/W	0	0 = "Don't care." 1 = ENVM_1 reported an ECC2 error. Write a 1 to this location to clear the bit.
21	ECC1_ERROR_1	R/W	0	0 = "Don't care." 1 = ENVM_1 reported an ECC1 error. Write a 1 to this location to clear the bit.
20	OVER_THRESH_1	R/W	0	0 = "Don't care." 1 = ENVM_1 accessed page over threshold. Write a 1 to this location to clear the bit.
19	ERASE_ERROR_1	R/W	0	0 = "Don't care." 1 = ENVM_1 reported an erase error. Write a 1 to this location to clear the bit.
18	PROG_ERROR_1	R/W	0	0 = "Don't care." 1 = ENVM_1 reported a programming error. Write a 1 to this location to clear the bit.
17	PROT_ERROR_1	R/W	0	0 = "Don't care." 1 = ENVM_1 reported a protection error. Write a 1 to this location to clear the bit.
16	BUSY_1	R	0	0 = ENVM_1 is ready to read. 1 = ENVM_1 is busy. This is a read only bit; writes have no effect.
15	ILLEGAL_CMD_0	R/W	0	0 = "Don't care." 1 = An illegal command has been issued to ENVM_0. Write a 1 to this location to clear the bit.
14:10	Reserved	R	0	Read as 0.

Table 4-11 • **ENVM_STATUS_REG (continued)**

Bit Number	Name	R/W	Reset	Description
9:8	ENVM_STATUS_0	R	0	These bits provide status information from the eNVM based upon the command and/or write that was issued to the eNVM. These are read only bits; writes have no effect. See Table 4-12 on page 67.
7	OP_DONE_0	R/W	0	0 = "Don't care." 1 = ENVM_0 has completed the commanded operation. Write a 1 to this location to clear the bit.
6	ECC2_ERROR_0	R/W	0	0 = "Don't care." 1 = ENVM_0 reported an ECC2 error. Write a 1 to this location to clear the bit.
5	ECC1_ERROR_0	R/W	0	0 = "Don't care." 1 = ENVM_0 reported an ECC1 error. Write a 1 to this location to clear the bit.
4	OVER_THRESH_0	R/W	0	0 = "Don't care." 1 = ENVM_0 accessed page over threshold. Write a 1 to this location to clear the bit.
3	ERASE_ERROR_0	R/W	0	0 = "Don't care." 1 = ENVM_0 reported an erase error. Write a 1 to this location to clear the bit.
2	PROG_ERROR_0	R/W	0	0 = "Don't care." 1 = ENVM_0 reported a programming error. Write a 1 to this location to clear the bit.
1	PROT_ERROR_0	R/W	0	0 = "Don't care." 1 = ENVM_0 reported a protection error. Write a 1 to this location to clear the bit.
0	BUSY_0	R	0	0 = ENVM_0 is ready to read. 1 = ENVM_0 is busy. This is a read only bit; writes have no effect.

Table 4-12 • **ENVM_STATUS_x**

eNVM Command	ENVM_STATUS_x		Description
	MSB	LSB	
Any command	0	0	Command completed successfully.
ARRAY_READ	0	1	Single bit error detected and corrected.
ARRAY_WRITE	0	1	Operation addressed a write protected page.
UNPROTECT_PAGE	0	1	Single bit error detected and corrected during the copy page operation.
PROGRAM_PAGE	0	1	Page buffer is unmodified or write to a protected page.
ERASE_PAGE	0	1	Attempt to erase a protected page
OVERWRITE_PAGE	0	1	Attempt to program a protected page
ARRAY_READ	1	0	Two or more errors detected.
UNPROTECT_PAGE	1	0	Two or more errors detected during copy page operation.
PROGRAM_PAGE	1	0	Programmed eNVM page does not match the page buffer.
PROGRAM_PAGE_PROTECTED	1	0	Programmed eNVM page does not match the page buffer.
ERASE_PAGE	1	0	Programmed eNVM page does not match the page buffer.
ERASE_PAGE_PROTECTED	1	0	Programmed eNVM page does not match the page buffer.
OVERWRITE_PAGE	1	0	Programmed eNVM page does not match the page buffer.
ARRAY_WRITE	1	1	Attempt to write another page before programming current page when PAGE_LOSS = 1.
UNPROTECT_PAGE	1	1	Attempt to copy unprotected page into page buffer that contains a modified page and PAGE_LOSS = 1
PROGRAM_PAGE	1	1	Page write count has exceeded the 10-year retention threshold.
PROGRAM_PAGE_PROTECTED	1	1	Page write count has exceeded the 10-year retention threshold.
ERASE_PAGE	1	1	Page write count has exceeded the 10-year retention threshold.
ERASE_PAGE_PROTECTED	1	1	Page write count has exceeded the 10-year retention threshold.
OVERWRITE_PAGE	1	1	Page write count has exceeded the 10-year retention threshold.

eNVM Control Register

Table 4-13 • **ENVM_CONTROL_REG**

Bit Number	Name	R/W	Reset	Description
31:24	COMMAND	R/W	0	This field contains the command to be executed by the eNVM. Commands are listed in Table 4-8 on page 63.
23:20	Reserved	R	0	Read as 0. Writes have no effect. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19:0	PAGE_ADDRESS	R/W	0	This field contains the page address for the operation defined in the COMMAND field.

eNVM Interrupt Enable Register

Table 4-14 • **ENVM_ENABLE_REG**

Bit Number	Name	R/W	Reset	Description
31	ILLEGAL_CMD_1	R/W	0	0 = Interrupt disabled when an illegal command has been issued. 1 = Enable interrupts when an illegal command has been issued.
24:30	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	OP_DONE_1	R/W	0	0 = Interrupt disabled. 1 = Enable interrupts when ENVM_1 completes an operation.
22	ECC2_ERROR_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 ECC2 errors.
21	ECC1_ERROR_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 ECC1 errors.
20	OVER_THRESH_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 over threshold errors.
19	ERASE_ERROR_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 erasing errors.
18	PROG_ERROR_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 programming errors.
17	PROT_ERROR_1	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_1 protection errors.
16	Reserved	R	0	Read as 0. Writes have no effect.

Table 4-14 • **ENVM_ENABLE_REG (continued)**

Bit Number	Name	R/W	Reset	Description
15	ILLEGAL_CMD_0	R/W	0	0 = Interrupt disabled. 1 = Enable interrupts when an illegal command has been issued.
14:8	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	OP_DONE_0	R/W	0	0 = Interrupt disabled. 1 = Enable interrupts when ENVM_0 completes an operation.
6	ECC2_ERROR_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 ECC2 errors.
5	ECC1_ERROR_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 ECC1 errors.
4	OVER_THRESH_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 over threshold errors.
3	ERASE_ERROR_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 erasing errors.
2	PROG_ERROR_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 programming errors.
1	PROT_ERROR_0	R/W	0	0 = Interrupt disabled. 1 = Interrupt enabled for ENVM_0 protection errors.
0	Reserved	R	0	Read as 0. Writes have no effect.

ENVM_0 Configuration Register

Table 4-15 • **ENVM_0_CR**

Bit Number	Name	R/W	Reset	Description
31:3	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	LOCK	R/W	0	0 = Disabled. 1 = eNVM disabled from JTAG access. The LOCK bit is used to give the user control over access to the eNVM from the JTAG interface. When LOCK is asserted, the JTAG interface will be prevented from any access attempts to the eNVM until LOCK is deasserted. For example, if a fabric master has access to the eNVM and does not want a JTAG operation to command or control the eNVM, the fabric master should set the LOCK bit. Likewise, if you only allow eNVM access via the Cortex-M3 microcontroller and want to prevent the JTAG interface from accessing the eNVM, you should set the LOCK bit.
1	PAGE_LOSS	R/W	0	0 = Disabled. 1 = Page loss protection enabled. When the PAGE_LOSS bit is set to 1, it prevents writes to any page other than the current page in the page buffer until the page is either discarded or programmed. A write to another page while the current page is Page Loss Protected will set the BUSY_0 bit and the PROT_ERROR_0 if the operation was performed on ENVM_0 for example.
0	READ_NEXT	R/W	0	0 = Disabled. 1 = Read next command enabled.

ENVM_1 Configuration Register

Table 4-16 • **ENVM_1_CR**

Bit Number	Name	R/W	Reset	Description
31:3	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	LOCK	R/W	0	0 = Disabled 1 = eNVM disabled from JTAG access. The LOCK bit gives you control over access to the eNVM from the JTAG interface. When LOCK is asserted, the JTAG interface will be prevented from any access attempts to the eNVM until LOCK is deasserted. For example, if a fabric master has access to the eNVM and does not want a JTAG operation to command or control the eNVM, the fabric master should set the LOCK bit. Likewise, if you only allow eNVM access via the Cortex-M3 microcontroller and want to prevent the JTAG interface from accessing the eNVM, you should set the LOCK bit.
1	PAGE_LOSS	R/W	0	0 = Disabled. 1 = Page loss protection enabled.
0	READ_NEXT	R/W	0	0 = Disabled. 1 = Read next command enabled.

ENVM_0 Page Status Register

Table 4-17 • **ENVM_PAGE_STATUS_0_REG**

Bit Number	Name	R/W	Reset	Description
31:8	WRITE_COUNT	R	0	Page write count; the number of times the page has been written to.
7:4	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OVER_THRESHOLD	R	0	0 = Page is under threshold. 1 = Page is over threshold.
2	READ_PROTECTED	R	0	0 = Page can be read. 1 = Page is user pass key read protected. JTAG read protect bit for page. This bit indicates that the page has been read protected using the user pass key via the JTAG interface. The user pass key must be used to change this bit setting. Read protection returns all zeros when the page is read.
1	WRITE_PROTECTED	R	0	0 = Page write protect bit is not set. 1 = Page is user pass key write protected. JTAG write protect bit for page. This bit indicates that the page has been write protected using the user pass key via the JTAG interface. The user pass key must be used to change this bit setting.
0	OVERWRITE_PROTECTED	R	0	0 = Page can be written to. 1 = Page is write protected. This status bit indicates that the page was programmed or erased using either the PROGRAMMED_PAGE_PROTECTED or ERASE_PAGE_PROTECTED command.

ENVM_1 Page Status Register

Table 4-18 • **ENVM_PAGE_STATUS_1_REG**

Bit Number	Name	R/W	Reset	Description
31:8	WRITE_COUNT	R	0	Page write count. The number of times the page has been written to.
7:4	Reserved	R	0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OVER_THRESHOLD	R	0	0 = Page is under threshold. 1 = Page is over threshold.
2	READ_PROTECTED	R	0	0 = Page can be read. 1 = Page is user pass key read protected. JTAG read protect bit for page. This bit indicates that the page has been read protected using the user pass key via the JTAG interface. The user pass key must be used to change this bit setting. Read protection returns all zeros when the page is read.
1	WRITE_PROTECTED	R	0	0 = Page write protect bit is not set. 1 = Page is user pass key write protected. JTAG write protect bit for page. This bit indicates that the page has been write protected using the user pass key via the JTAG interface. The user pass key must be used to change this bit setting.
0	OVERWRITE_PROTECTED	R	0	0 = Page can be written to. 1 = Page is write protected. This status bit indicates that the page was programmed or erased using either the PROGRAMMED_PAGE_PROTECTED or ERASE_PAGE_PROTECTED command.

5 – SmartFusion Embedded FlashROM (eFROM)

SmartFusion cSoCs have 1,024 bits of on-chip nonvolatile flash memory called embedded flash read only memory (embedded FlashROM or eFROM). The eFROM can be read and written via the JTAG interface when performing external device programming. This embedded flash read only memory is directly accessible for reading during normal operation from user firmware running on the SmartFusion microcontroller subsystem (MSS).

Architecture of the Embedded FlashROM (eFROM)

The eFROM is arranged in eight banks of 128 bits (16 bytes) during programming (eFROM writing). The 16-byte bank is also referred to as a page. Figure 5-1 shows a graphical representation of the 16 bytes in each of the 8 banks, or pages, which make up the 1,024-bit eFROM. The eFROM has a 7-bit address and 8-bit data port. The upper 3 bits of the 7-bit address define the page number to be read and the 4 lower bits of the 7-bit address specify the byte number to be read.

The eFROM can be programmed externally via the JTAG port. It cannot be programmed directly from the MSS or FPGA fabric during normal operation. Each of the eight 128-bit pages of the eFROM can be selectively programmed. The eFROM can only be reprogrammed on a page boundary. Programming involves an automatic, on-chip page erase prior to reprogramming the page.

The eFROM supports synchronous reading. The eFROM can be read on byte boundaries. To enable user access to the eFROM during normal operation, the eFROM is visible to the MSS as an APB peripheral component which is mapped to the ARM Cortex-M3 memory space.

For detailed eFROM data access timing characteristics, refer to the DC and Switching Characteristics chapter of the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet. The eFROM APB Interface and Configuration Registers are presented later in this section.

		Byte Number in Bank 4 LSB of ADDR (READ)															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bank Number - 3MSB of ADDR (READ)	7																
	6																
	5																
	4																
	3																
	2																
	1																
	0																

Figure 5-1 • **Graphical Representation of eFROM Architecture**

Table 5-1 on page 76 summarizes the various eFROM read and write capabilities, sorted by access mode. For more information regarding in-application programming (IAP), refer to the "SmartFusion Programming" chapter of the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet. Similarly, refer to the appropriate datasheet section to learn more about the external JTAG programming options. The eFROM content is typically entered in Libero software by using the graphical configuration interface within the SmartFusion MSS configurator. Refer to the *Libero SoC User's Guide* or the Libero SoC Online Help for more information about using the MSS configurator and the Embedded FlashROM Configurator. The eFROM data content can be entered directly into the FlashROM configuration window, loaded from a data file (in binary, decimal, hex, or ASCII text format), or serialized when used to specify eFROM contents for a large number of devices in an incrementing or decrementing series. The FlashROM

configurator produces a UFC file, which must be used when generating the SmartFusion device programming bitstream in order to write the user eFROM data into the SmartFusion physical eFROM during device programming.

Table 5-1 • **eFROM Read/Write Capabilities by Access Mode**

Access Mode	eFROM Read	eFROM Write
JTAG Programming	Yes	Yes
In-Application Programming (IAP)	Yes	Yes (A2F060, A2F500); No (A2F200)
Directly from MSS	Yes	No
Directly from FPGA Fabric	No	No

Reading the eFROM Contents via the MSS

The SmartFusion embedded FlashROM physical memory is interfaced to the SmartFusion microcontroller subsystem (MSS) via an APB interface controller, as shown in Figure 5-2. This makes the eFROM accessible to user firmware running on the MSS through simple APB peripheral accesses between the Cortex-M3 processor and the eFROM. From the firmware programmer's point of view, this is a memory mapped peripheral access. Physically, the Cortex-M3 microcontroller accesses the eFROM APB peripheral via the AHB bus matrix and an AHB to APB Bridge. The eFROM resides on APB_1 and is clocked by the PCLK1. The eFROM occupies address range 0x40015000 - 0x40015FFF in the Cortex-M3 memory map. For more information, refer to the "AHB Bus Matrix" section on page 15.

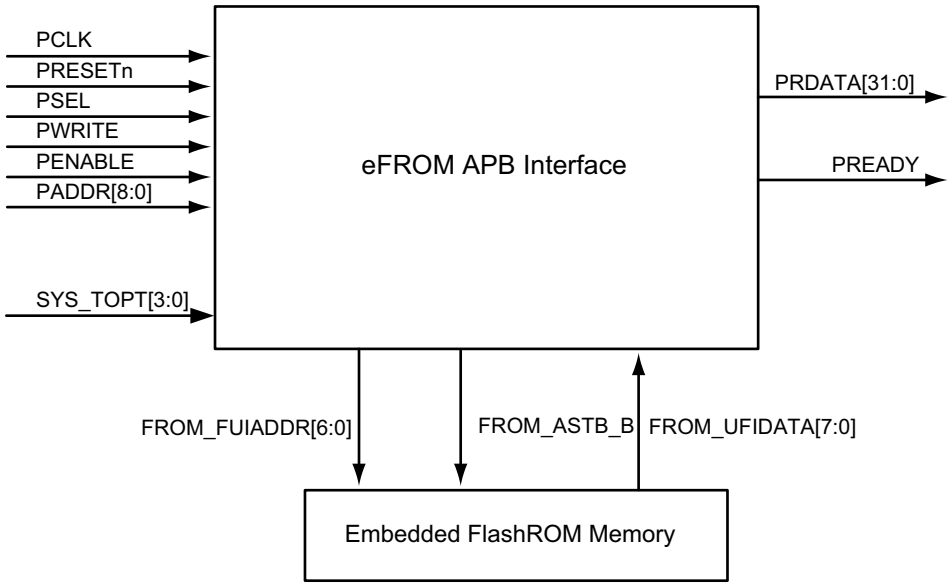


Figure 5-2 • **eFROM APB Interface Controller**

The ports shown in Figure 5-2 on page 76 are explained in Table 5-2 and Table 5-3.

***Table 5-2* • eFROM APB Controller to MSS Interface Port List**

Port	Type	Function
PADDR[8:0]	Input	APB address bus (PADDR[1:0] are not used)
PCLK	Input	APB clock
PENABLE	Input	APB enable
PRDATA[31:0]	Output	APB read data bus
PWRITE	Input	APB write
PSEL	Input	APB select
PREADY	Output	APB ready
PRESETn	Input	APB reset (active low)

***Table 5-3* • eFROM APB Controller to Physical Memory Port List**

Port	Type	Function
FROM_ASTB_B	Output	FROM address strobe (active on falling edge) can be aligned with PCLK rising or falling edge, as described in Table 5-7 on page 78
FROM_UFIADDR[6:0]	Output	FROM address bus
FROM_UFIDATA[7:0]	Input	FROM data output bus

In Figure 5-2 on page 76, the SYS_TOPT[3:0] input port represents part of the MSS System Register called the EFROM_CR register. The EFROM_CR register is used to set configurable eFROM APB interface timing options, as described in Table 5-4 on page 77, through Table 5-7 on page 78. All APB accesses to the eFROM are read only. Since the user has various MSS_CLK and PCLK1 configuration options, the eFROM timing options have been preselected to ensure that a successful eFROM data access is performed regardless of the MSS clocking configuration selected.

APB write transactions are ignored and result in no activity at the eFROM APB controller to eFROM physical memory interface. When not performing a read access, the FROM_UFIADDR[6:0] bus is forced to all 1s to limit dynamic dissipation usage in the eFROM.

***Table 5-4* • EFROM_CR Register Map**

Register Name	Address	R/W	Reset Value	Description
EFROM_CR	0xE0042024	R/W	0x00000009	Used to set eFROM APB interface controller timing options

***Table 5-5* • EFROM_CR**

Bit Number	Name	R/W	Reset Value	Description
31:4	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:1	SYS_TOPT[3:1]	R/W	100 (binary)	Controls the number of wait states inserted by the eFROM APB interface controller before asserting PREADY during an APB access. Wait states are needed since the APB clock (PCLK1) frequency can be higher than the maximum eFROM clock frequency. This can result in PCLK cycle times which are shorter than the expected eFROM data access timing and result in incorrect eFROM data being read onto the APB PRDATA bus. Table 5-6 shows the recommended wait state setting.
0	SYS_TOPT[0]	R/W	1	0 – Timing Option 1 - Reserved 1 – Timing Option 2 - Refer to Table 5-7

***Table 5-6* • Timing Options Controlled by SYS_TOPT[3:1]**

SYS_TOPT[3:1]	Wait States Added	Notes
000	0	Reserved; do not use
001	1	Reserved; do not use
010	2	Reserved; do not use
011	3	Reserved; do not use
100	4 – DEFAULT	This is the default and recommended setting selected to provide sufficient eFROM memory access time. This setting ensures that the correct eFROM data is propagated from the memory back to the MSS through the eFROM APB controller, regardless of whether the MSS_CLK to PCLK1 ratio is 1:1, 2:1, or 4:1.
101	5	Reserved
110	6	Reserved
111	7	Reserved

***Table 5-7* • Timing Options Controlled by SYS_TOPT[0]**

SYS_TOPT[0] Value	Timing Option	Description
0	Timing option 1	Reserved; do not use
1	Timing option 2 – DEFAULT	Allows one complete PCLK1 cycle for the APB address to propagate from the APB bus to the eFROM through the eFROM APB interface controller. This is the default and recommended setting.

When reading the eFROM via firmware running on the SmartFusion MSS, each byte is read from the eFROM by addressing an offset of the eFROM APB peripheral base address. For example, the first byte of the eFROM is read by addressing the eFROM APB peripheral base address (0x40015000) and returned via the MSS AHB bus matrix. The next byte of the eFROM is read by incrementing the base address by 4. Thus to read byte i of the eFROM, the firmware would access the following Cortex-M3 memory space address: (eFROM APB base address) + (4 * i).

6 – Embedded SRAM (eSRAM) Memory Controllers

The embedded SRAM (eSRAM) memory controller is an Advanced High-Performance Bus (AHB) slave that provides access to two equal-sized blocks of eSRAM. The total amount of available eSRAM ranges from 16 Kbytes to 64 Kbytes, depending on device size. Each individual eSRAM block is therefore 8 Kbytes to 32 Kbytes, organized in a 2Kx32 to 8Kx32 fashion. The eSRAM in these blocks is byte, halfword, and word addressable. The ARM Cortex-M3 microcontroller and other masters find the eSRAMs available as one contiguous area of memory.

The address of eSRAM_0 is 0x20000000 and eSRAM_1 is located at an address which depends on the total amount of eSRAM present on the device. The location of eSRAM_1 is always directly after eSRAM_0 in the memory map. Table 6-1 lists the memory locations for eSRAM in SmartFusion devices and Table 6-2 gives pertinent register definitions.

***Table 6-1* • eSRAM Address Locations**

Device	eSRAM_0	eSRAM_1	Total SRAM
A2F060	0x20000000	0x20002000	16 Kbytes
A2F200	0x20000000	0x20008000	64 Kbytes
A2F500	0x20000000	0x20008000	64 Kbytes

***Table 6-2* • ESRAM_CR Register Map**

Register Name	Address	R/W	Reset Value	Description
ESRAM_CR	0xE0042000	R/W	0x00000010	Controls address mapping of the eSRAMs
AHB_MATRIX_CR	0xE0042018	R/W	0x00000007	Configures the AHB bus matrix

Table 6-3 gives bit definitions for the eSRAM Configuration Register.

***Table 6-3* • ESRAM_CR**

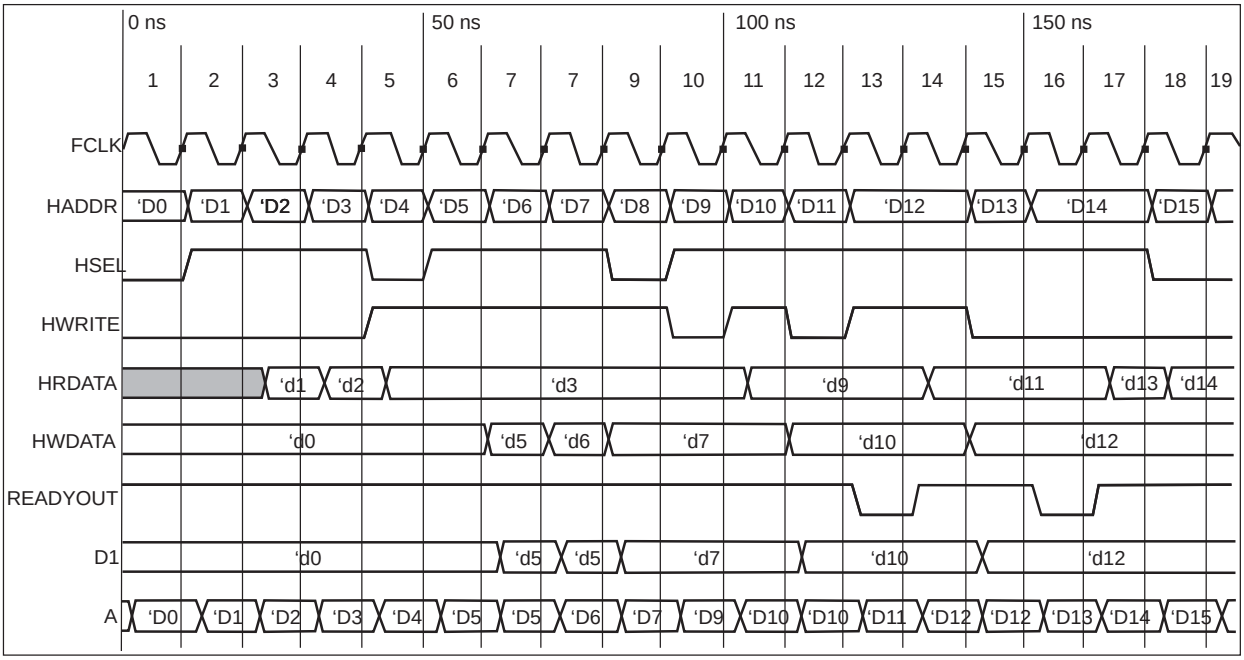
Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R/W	0x0	Read 0. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	COM_ESRAMFWREMAP	R/W	0x0	Remap of embedded SRAMs. 0 = no remapping of the eSRAMs occurs 1 = eSRAM0 is mapped to location 0x00000000 and eSRAM1 is mapped directly above it.

After power-on reset, both eSRAM blocks are mapped into the SRAM bit-banding area of Cortex-M3 system space, which is located from address 0x20000000 to address 0x20100000. In the Cortex-M3 microcontroller's view of the memory map, the eSRAM area can be remapped to 0x00000000 within Cortex-M3 code space. This is done with boot code by setting the COM_ESRAMFWREMAP bit in the ESRAM_CR register ("eSRAM Configuration Register" section of the "AHB Bus Matrix" section on page 15, address 0xE0002000) to 1.

Note: The eSRAM continues to be available to the Cortex-M3 at address 0x20000000, even when the eSRAM is remapped to address 0x00000000 in the Cortex-M3 memory map.

The eSRAM is also visible to masters other than the Cortex-M3 microcontroller, although the corresponding port on the AHB bus matrix must be enabled by boot code before the fabric master can access eSRAM. See the AHB_MATRIX_CR definition in the "AHB Bus Matrix" section on page 15. The eSRAM is always located at address 0x20000000 in the memory map seen by each of the masters other than the Cortex-M3 microcontroller. The COM_ESRAMFWREMAP bit of the ESRAM_CR ("eSRAM Configuration Register" section of the"AHB Bus Matrix" section on page 15) controls only the memory map seen by the Cortex-M3 microcontroller.

Figure 6-1 depicts read and write timing for the eSRAMs. All eSRAMs operate with zero wait states. However, if an eSRAM is busy completing a write transaction, the initiation of a read cycle will incur one wait state and be initiated in the following cycle.



Notes:

1. *The Dx numbers on HADDR are simply used to number and track transactions and are not actual addresses.*
2. *Transactions numbered D0, D4, D8, and D15 are idle transactions.*
3. *Transactions D1, D2, D3, D9, D11, D13, and D14 are eSRAM read transactions.*
4. *Transactions D5, D6, D7, D10, and D12 are eSRAM write transactions.*

Figure 6-1 • **Read and Write Timing**

Misaligned Addresses

Misaligned addresses are not allowed. The Cortex-M3 microcontroller aligns addresses as they exit the core. However, for a master residing in the FPGA fabric, you must ensure that the correct addressing (byte, halfword, or word) is applied to the correct address. The eSRAM controller will map misaligned addresses to the appropriate address by ignoring the appropriate least significant bits (LSBs) of HADDR. For example, if a fabric master attempts to read a word at address 0x20000001, 0x20000002, or 0x20000003, it will actually read the word at address 0x20000000.

7 – External Memory Controller

This section describes the external memory controller (EMC), available only on SmartFusion cSoCs A2F060, A2F200 and A2F500 (not available on A2F500 for the PQ208 package).

Main Features

The EMC provides a glueless interface to external memories. Memory types supported are asynchronous memories and synchronous SRAMs. The EMC is mapped into system address space from 0x70000000 to 0x77FFFFFF.

The EMC has the following features:

- 2 chip selects, each addressing 64 MBytes of address space
  - Programmable timing for each chip select
- 8-bit or 16-bit shared data bus
- Asynchronous memories supported
  - Static random access memory (SRAM)
  - NOR flash memory
  - PSRAM
- Synchronous memories supported
  - Synchronous static random access memory (SSRAM)
- Write enable and byte lane support
- Translates 32-bit AHB transactions into successive halfword and byte transactions.
- Automatic translation of misaligned addresses

Naming Convention

Throughout this section a lower case x is used as a placeholder in a register name or signal to signify either a 0 or 1, which corresponds to chip select 0 or 1.

Block Diagram

The EMC primarily exists to interface with off-chip memory devices that can be addressed by the MSS or user logic in the FPGA fabric. It appears as a slave on the AHB bus matrix, as shown in Figure 7-1.

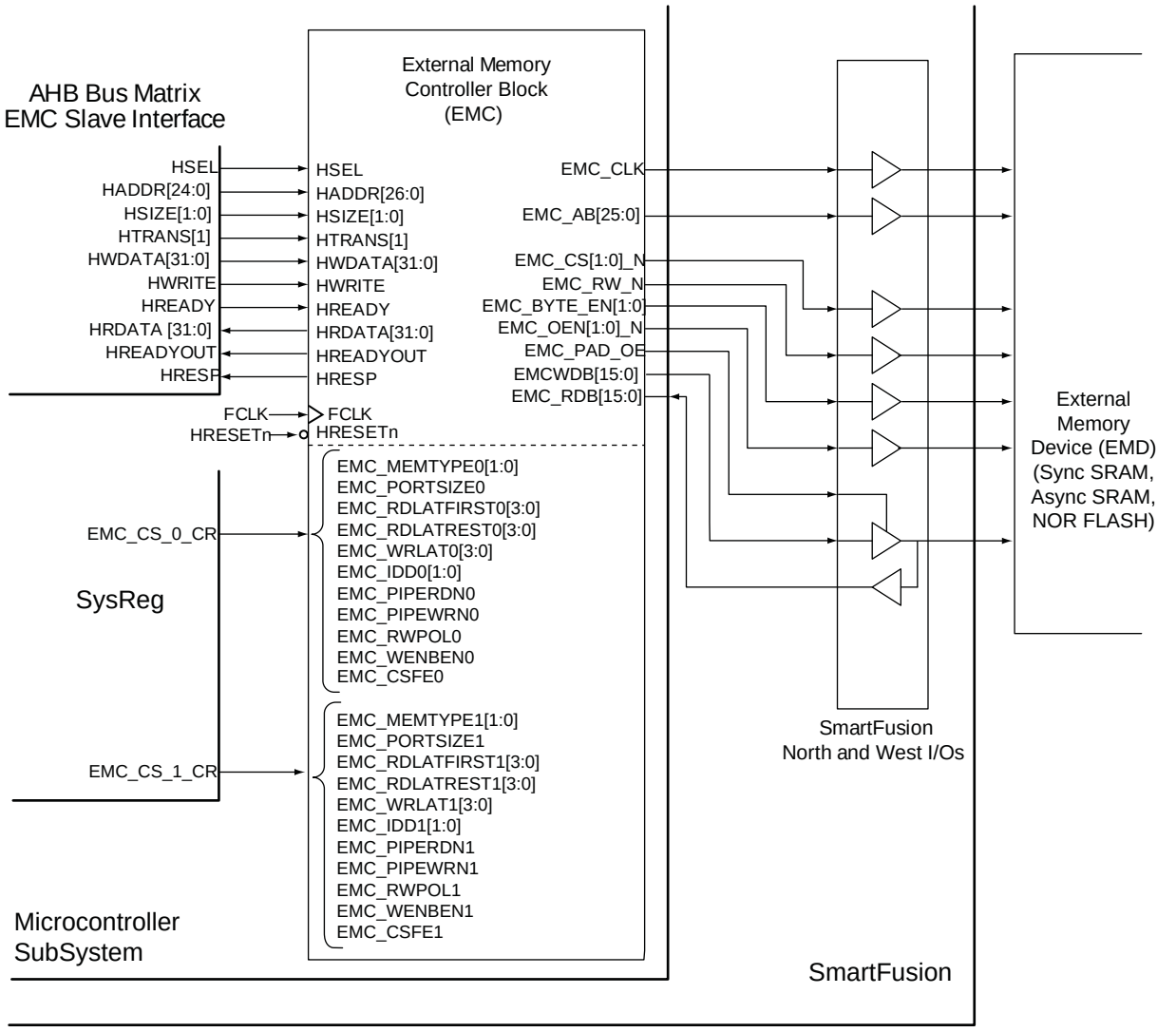


Figure 7-1 • **External Memory Controller Block Diagram**

Functional Description

The EMC accepts single AHB transactions for reading and writing to external memory devices (EMDs). The EMC reformats single AHB transactions into the format required by the external EMD. The EMC may use multiple FCLK cycles to complete an EMD access, depending on the characteristics of the EMD and on the size of the access (word, halfword, or byte) and the width of the data bus to the EMD. An AHB access consists of an address phase and a data phase, as shown in Figure 7-2.

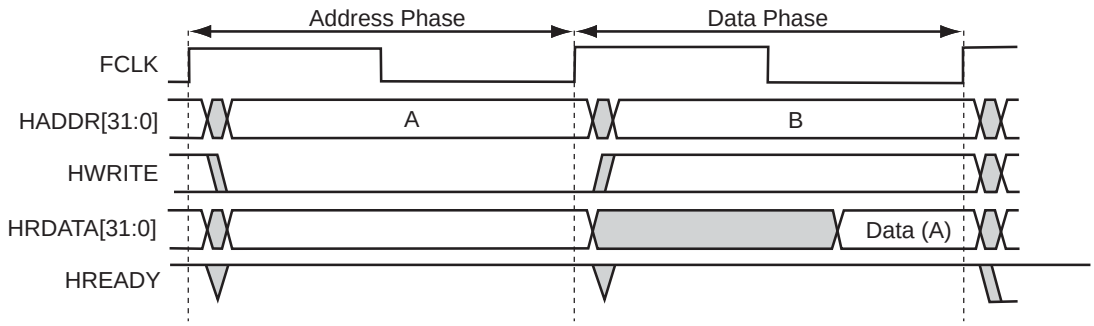


Figure 7-2 • **AHB Address/Data Phase for Read Transfer**

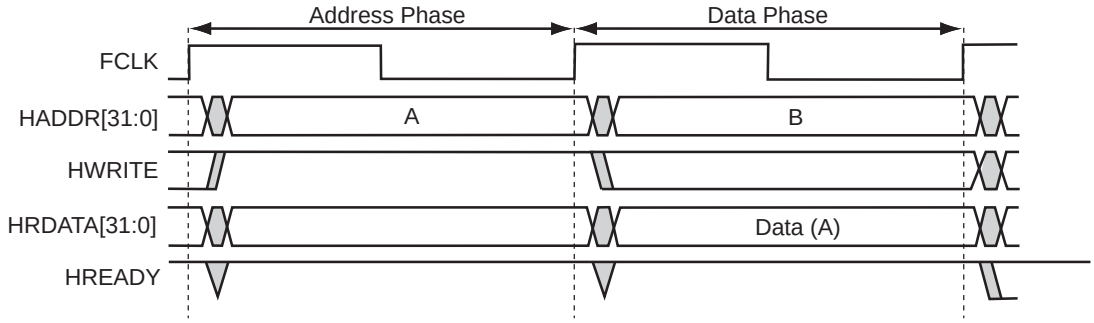


Figure 7-3 • **AHB Address/Data Phase for Write Transfer**

The EMC cannot complete EMD read and write transactions in only two FCLK cycles, so the user must configure the EMC and insert wait states in the data phase of the AHB access to complete the EMD access. Figure 7-4 shows an AHB read transaction with two wait states inserted into the data phase of the AHB transaction.

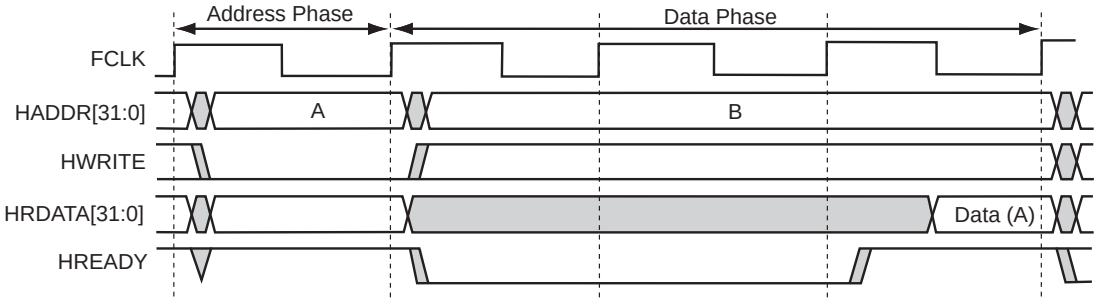


Figure 7-4 • **AHB Read Access with Two Wait States**

The EMC uses the additional clock cycles to complete the EMD transaction. Figure 7-5 illustrates an AHB transaction with three wait states and shows the EMD interface signals during the AHB data phase and how they are used to complete the EMD access. Note that the AHB address phase will always be one FCLK cycle and the wait states are inserted into the data phase. The figure illustrates a pipelined synchronous SRAM EMD requiring a pipeline clock cycle between EMC_AB valid and EMC_DB output.

As shown, the EMC requires one FCLK cycle to output the EMD address, EMC_AB. The EMD requires two FCLK cycles to fetch the data, EMC_DB. One additional FCLK cycle is required to transfer the EMD data, EMC_DB, to the AHB Bus, HRDATA. Therefore, this read is an AHB read access with a total of three wait states.

Depending on the type of access, the EMC may need to be configured with up to six wait states to complete an AHB transaction (a word read from a byte wide EMD requires a total of eight cycles, for example).

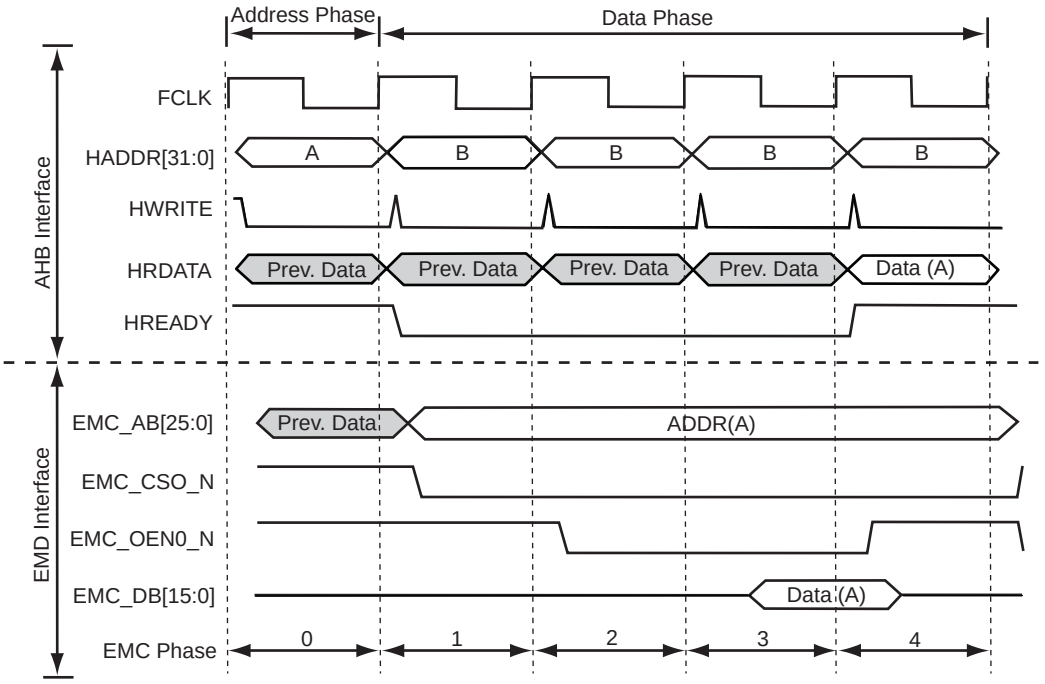


Figure 7-5 • **AHB to EMC Transaction**

FCLK Cycles and EMC Phases

For synchronous EMDs and asynchronous EMDs that operate at FCLK speeds, one FCLK cycle is the same as one EMC phase. If FCLK is 100 MHz, for example, during an EMC address phase where address is input to the EMD, and if the setup time for the address to be latched in the EMD by EMC_CLK or a control signal is less than 10 ns, then the EMC address phase can be one FCLK cycle long. For some asynchronous EMDs the timing required for each phase may be greater than one FCLK cycle. EMC_CLK is driven by FCLK. Therefore, the EMC is not designed around FCLK cycles but rather around EMC phases that correspond to EMD phases and are at least one FCLK cycle long. EMC phases are made greater than an FCLK cycle by programming the latency fields in the configuration registers, EMC_CS_0_CR and EMC_CS_1_CR.

Back-to-Back AHB Accesses

AHB accesses to EMD may be consecutive, in which case the current access may cause the pending access to wait until it completes. At the completion of the first access, the second access can start.

The last phase of an EMC access is either the transfer of read data to HRDATA or the write of EMC_DB to the EMD. Since neither of these actions involves the signals required for an AHB address phase, it is possible for the last EMC phase to overlap with the address phase of a new AHB transaction, thus saving one EMC phase per AHB transaction.

In this case, the EMC never transitions back to its 0 phase (idle), but continues to EMC phase 1 for the second transaction, after completing the first transaction. This also requires that the EMC recognize that a pending transaction is waiting and generate the EMC control signals during the last phase of the current transaction (which is also the address phase of the pending transaction). When there is no pending transaction, the EMC generates the appropriate control signals during its phase 0, which is coincident with the AHB address phase when a new transaction is received.

When the EMC is not being accessed, it remains in phase 0 (idle). Figure 7-6 shows the AHB and EMC signals when back-to-back AHB transactions are received by the EMC.

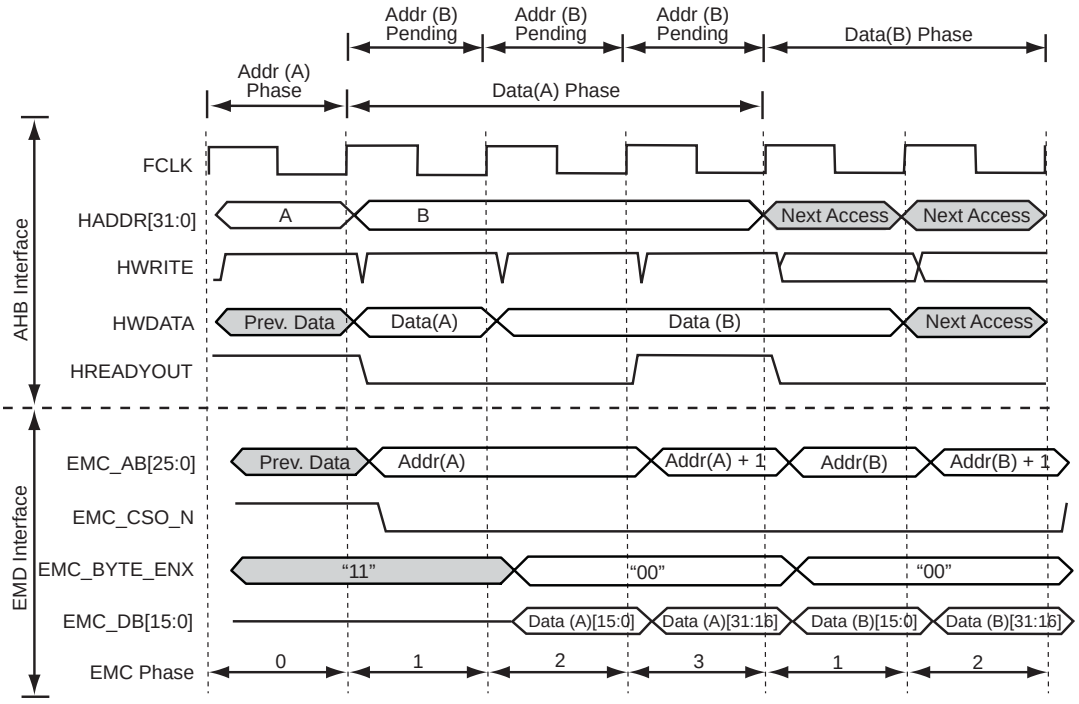


Figure 7-6 • **Back to Back AHB Bus Transactions**

EMC Memory Map

AHB addressing is byte oriented, so although the AHB buses, HRDATA and HWDATA, are 32-bit buses, individual bytes, half-words (16 bits), or the entire word can be read or written.

The width of the data bus to the EMD(s) is either 16-bit or 8-bit. Therefore, all AHB accesses must be broken into halfword or byte EMD accesses, depending on the EMD connected to the EMC. The EMC configuration bit, EMC_PORTSIZEx, is used to configure the EMC to the width of the connected EMD.

The EMC memory space is divided into an upper and lower half. Each half can be connected to a separate EMD. Addresses to the EMDs, EMC_AB, are common. Access to each memory space half is determined by the assertion of the controls EMC_CS[1:0]_N, where index 1 corresponds to the upper half and index 0 to the lower half. Each half of the EMC supports up to 64 MBytes of memory if using all 26 memory address bits. Table 7-1 illustrates the starting and ending address for each chip select.

***Table 7-1* • External Memory Controller Memory Regions**

Chip Select	Starting Address	Ending Address
EMC_CS0_N	0x70000000	0x73FFFFFF
EMC_CS1_N	0x74000000	0x77FFFFFF

EMC to EMD Memory Maps

Figure 7-7 through Table 7-10 on page 88 show the AHB to EMD memory mapping when various widths of memories are used.

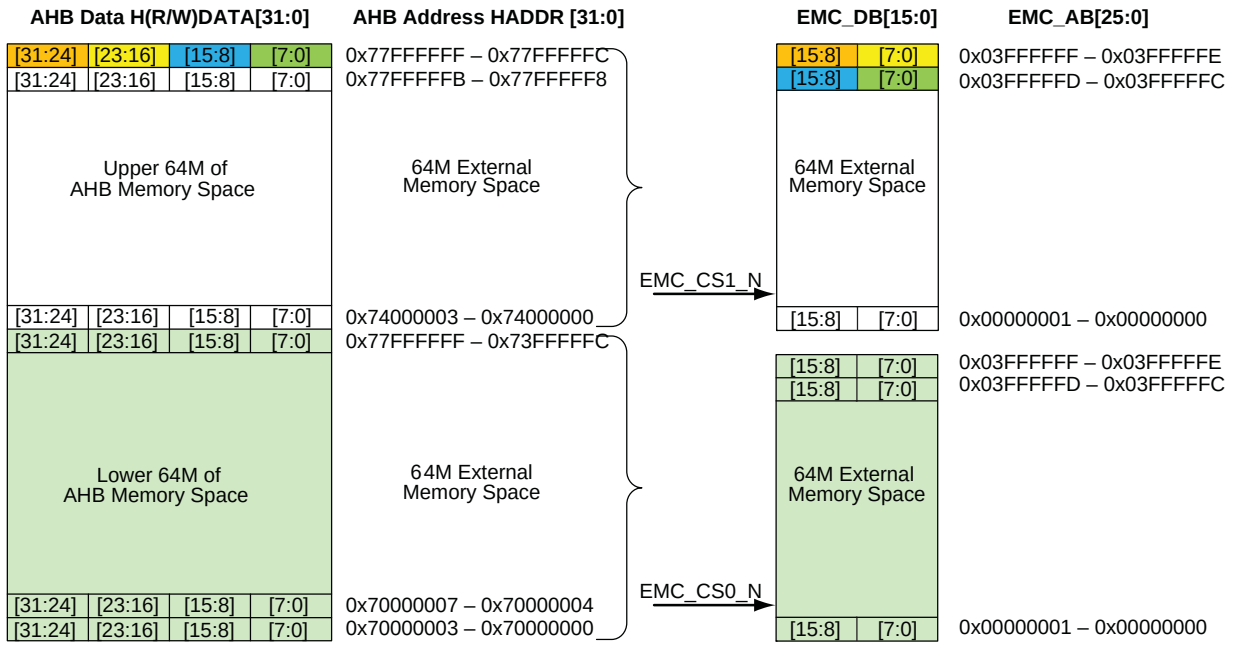


***Figure 7-7 •* Halfword Wide External Memory Device Memory Map**

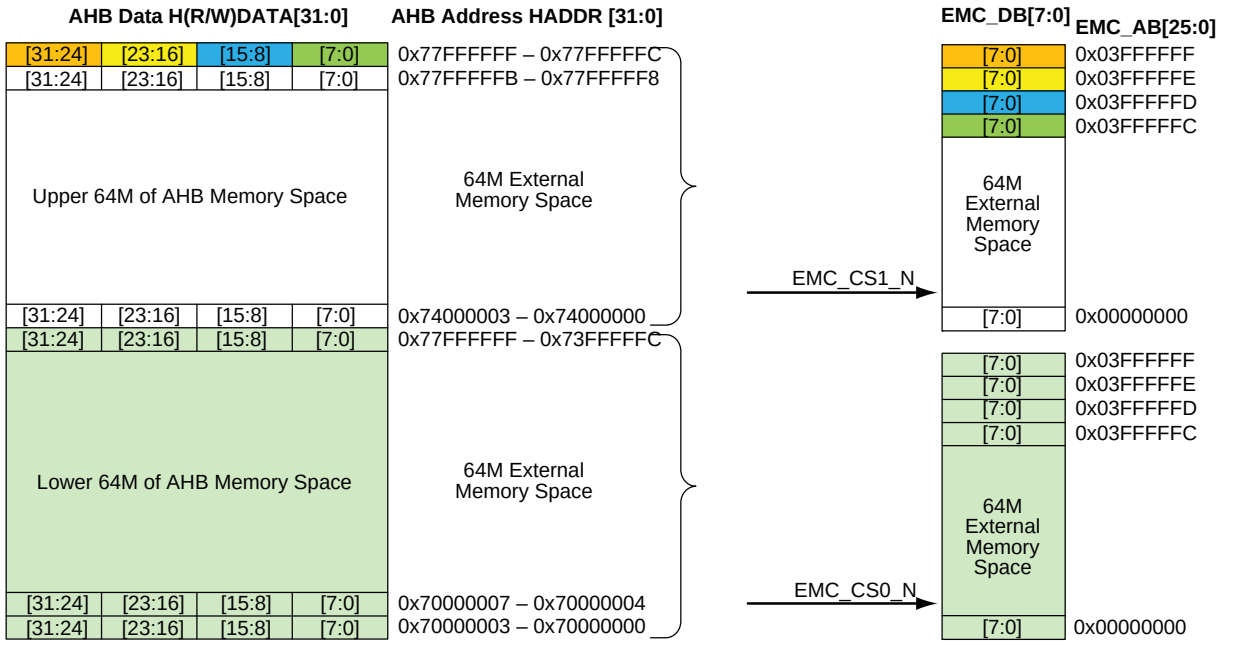


Figure 7-8 • **Byte Wide External Memory Device Memory Map**

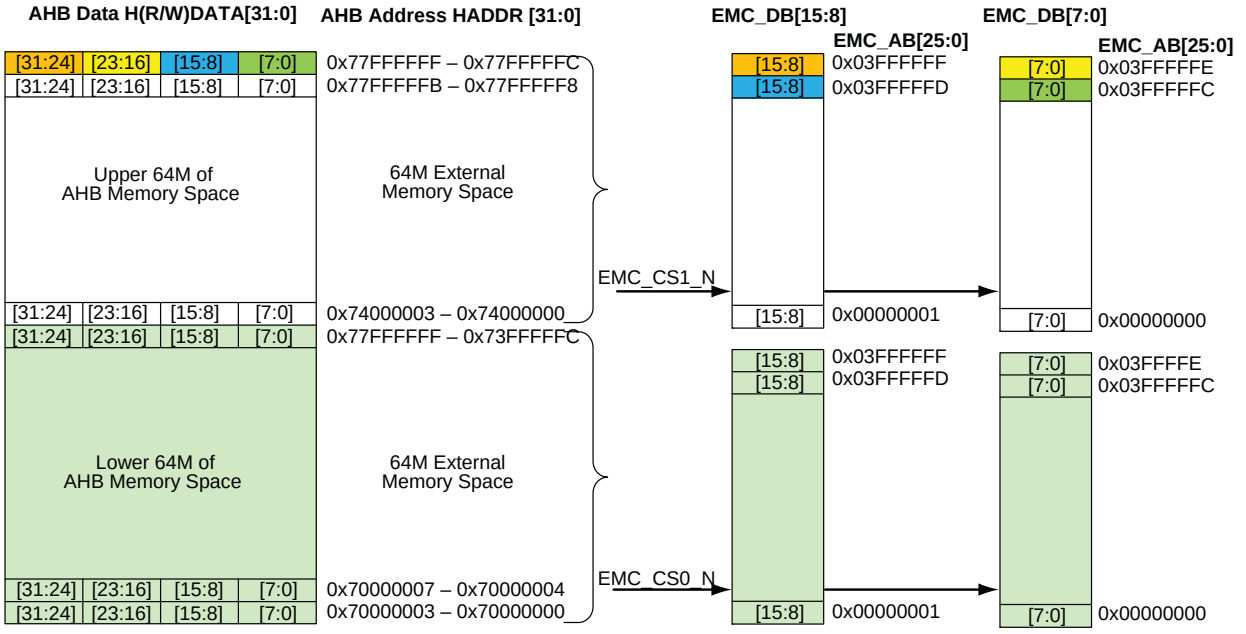


Figure 7-9 • **Halfword Wide External Memory Device Memory Map Using x8 EMDs**

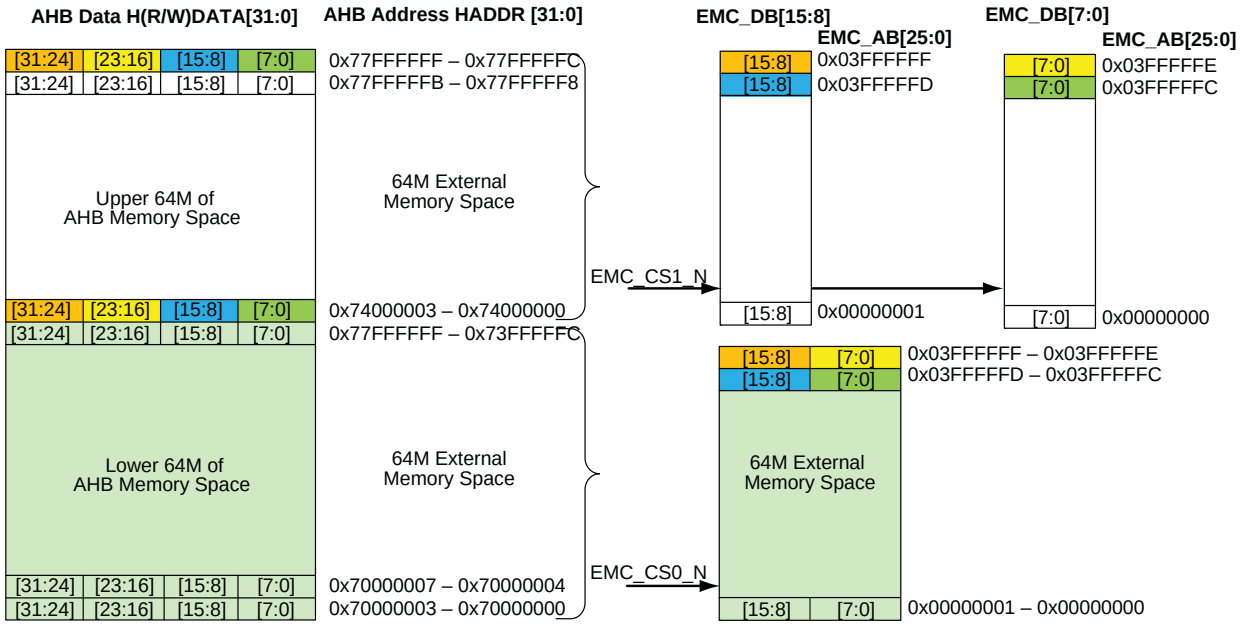


***Figure 7-10* • Halfword and Byte Wide External Memory Device Memory Map**

Misaligned Access Behavior

All halfword accesses are performed on halfword boundaries and all word accesses are performed on word boundaries. As shown in Table 7-2 and Table 7-3 on page 89, misaligned accesses are automatically aligned to an appropriate boundary. In other words, for word accesses, the EMC ignores the two LSBs of the address bus and for halfword accesses the EMC ignores the LSB of the address. No error indication is generated during this automatic alignment. The misaligned transactions are processed, but they are not trapped. Users must be aware of this when implementing masters in the FPGA fabric.

***Table 7-2* • Misaligned Write Transactions**

Write Transaction Size	AHB Address	EMD Location(s) Written
Byte	0x00	0x00
Byte	0x01	0x01
Byte	0x02	0x02
Byte	0x03	0x03
Halfword	0x00	0x01, 0x00
Halfword	0x01	0x01, 0x00
Halfword	0x02	0x03, 0x02
Halfword	0x03	0x03, 0x02
Word	0x00	0x03, 0x02, 0x01, 0x00
Word	0x01	0x03, 0x02, 0x01, 0x00
Word	0x02	0x03, 0x02, 0x01, 0x00
Word	0x03	0x03, 0x02, 0x01, 0x00

***Table 7-3* • Misaligned Read Transactions**

Read Transaction Size	AHB Address	EMD Addresses Read
Byte	0x00	0x00
Byte	0x01	0x01
Byte	0x02	0x02
Byte	0x03	0x03
Halfword	0x00	0x01, 0x00
Halfword	0x01	0x01, 0x00
Halfword	0x02	0x03, 0x02
Halfword	0x03	0x03, 0x02
Word	0x00	0x03, 0x02, 0x01, 0x00
Word	0x01	0x03, 0x02, 0x01, 0x00
Word	0x02	0x03, 0x02, 0x01, 0x00
Word	0x03	0x03, 0x02, 0x01, 0x00

FCLK Cycles Required for Memory Accesses

Based on the memory access (read or write), EMD data width (EMC_PORTSIZEx), and memory access width, the EMC will require different number of phases to complete a read or write. In the case where the AHB transaction width (HSIZE) is greater than EMC_PORTSIZEx, the EMC must perform multiple EMD accesses. Since accesses can have pipelined addresses and data, the number of EMC phases required depends on the configured programming of the EMC. Table 7-4 shows the number of phases required for each access as a function of EMC_PORTSIZEx and AHB transaction width (HSIZE) with no pipeline or latency delays.

The various possible configurations of EMC_PORTSIZEx and AHB transaction width (HSIZE) result in EMD accesses that will require either one, two, or four addresses and make one, two, or four EMD accesses (read or write). Table 7-4 identifies the number of accesses required and the maximum number of phases required for an access when a phase is equal to one FCLK cycle.

***Table 7-4* • Memory Address Generation**

Access Type	AHB Transaction Width (HSIZE)	EMC_PORTSIZEx	Number of Phases
Read	Byte	Byte	1
Read	Halfword	Byte	2
Read	Word	Byte	4
Read	Byte	Halfword	1
Read	Halfword	Halfword	1
Read	Word	Halfword	2
Write	Byte	Byte	1
Write	Halfword	Byte	2
Write	Word	Byte	4
Write	Byte	Halfword	1
Write	Halfword	Halfword	1
Write	Word	Halfword	2

From the perspective of an AHB access, Table 7-5 and Table 7-6 on page 91 can be used to calculate the number of FCLK cycles required for each type of access. The following assumptions are made:

- An access is measured from the AHB address phase to the assertion of HREADY at the end of the transaction, including the FCLK cycle where HREADYOUT is asserted. Figure 7-11 shows a transaction that requires four FCLK cycles (address plus data phases).
- Synchronous EMDs will require an additional pipeline delay of one FCLK cycle.
- A read or write to an asynchronous EMD requires two FCLK cycles for each AHB address phase.
- If N consecutive (back-to-back) accesses are done on the same EMD, the total number of FCLK cycles required is given by EQ 1:

$$\text{Total FCLK cycles} = T_{FCLK} + (N - 1) \times (T_{FCLK} - 1)$$

EQ 1

  – This occurs because for back-to-back transactions, the last FCLK cycle can overlap with the next AHB address cycle. The EMC phase does not return to the idle or 0 state between transactions, as shown in Figure 7-6.

An additional FCLK cycle is added whenever an access to a synchronous EMD is followed by an access to an asynchronous or FLASH EMD or vice versa. One additional cycle is inserted at the end of the access to ensure that PCB delays will not cause a race condition with the EMD control signals and cause an erroneous write to either memory.

***Table 7-5* • Synchronous SRAM FCLK Cycle Counts**

Access Type	EMD Data Bus Width	AHB Transaction Width (HSIZE)	Number of Accesses	FCLK Cycles (T_{FCLK})
Read	Byte	Byte	1	4
Read	Byte	Halfword	2	5
Read	Byte	Word	4	7
Read	Halfword	Byte	1	4
Read	Halfword	Halfword	1	4
Read	Halfword	Word	2	5
Write	Byte	Byte	1	3
Write	Byte	Halfword	2	4
Write	Byte	Word	4	6
Write	Halfword	Byte	1	3
Write	Halfword	Halfword	1	3
Write	Halfword	Word	2	4

***Table 7-6* • Asynchronous Memory FCLK cycle count**

Access Type	EMD Data Bus Width	AHB Transaction Width (HSIZE)	Number of Accesses	FCLK Cycles (T_{FCLK})
Read	Byte	Byte	1	3
Read	Byte	Halfword	2	5
Read	Byte	Word	4	8
Read	Halfword	Byte	1	3
Read	Halfword	Halfword	1	3
Read	Halfword	Word	2	5
Write	Byte	Byte	1	3
Write	Byte	Halfword	2	5
Write	Byte	Word	4	8
Write	Halfword	Byte	1	3
Write	Halfword	Halfword	1	3
Write	Halfword	Word	2	5

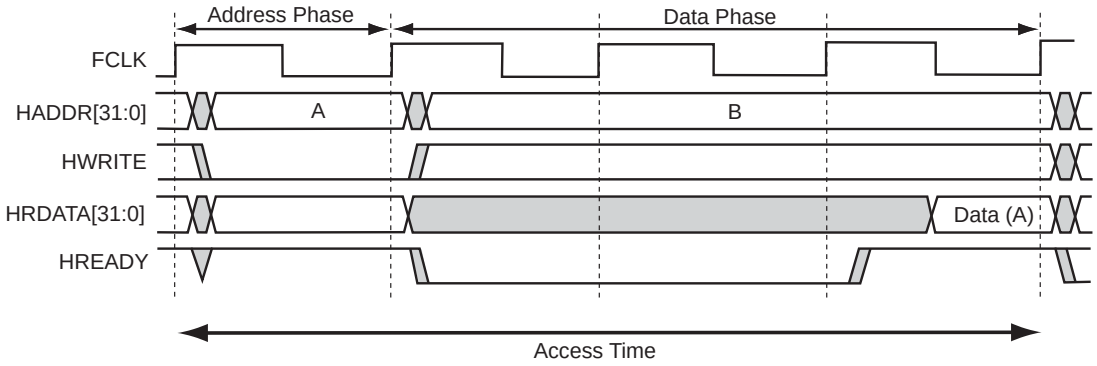


***Figure 7-11* • Access Time**

External Memory Device Examples

Figure 7-12 gives an overview of how to connect external memories to the EMC. While not exhaustive, the examples given are intended to provide the user with a sense of what the EMC is capable of.

Figure 7-12 shows a typical x16 SRAM connected to the EMC of a SmartFusion cSoC. An eight megabyte device is shown. The address bus is halfword aligned (A[17:0] = EMC_AB[18:1], since EMC_AB is a byte address). The halfword synchronous SRAM (16-bit) device uses the byte enable control pins to affect a single byte write.

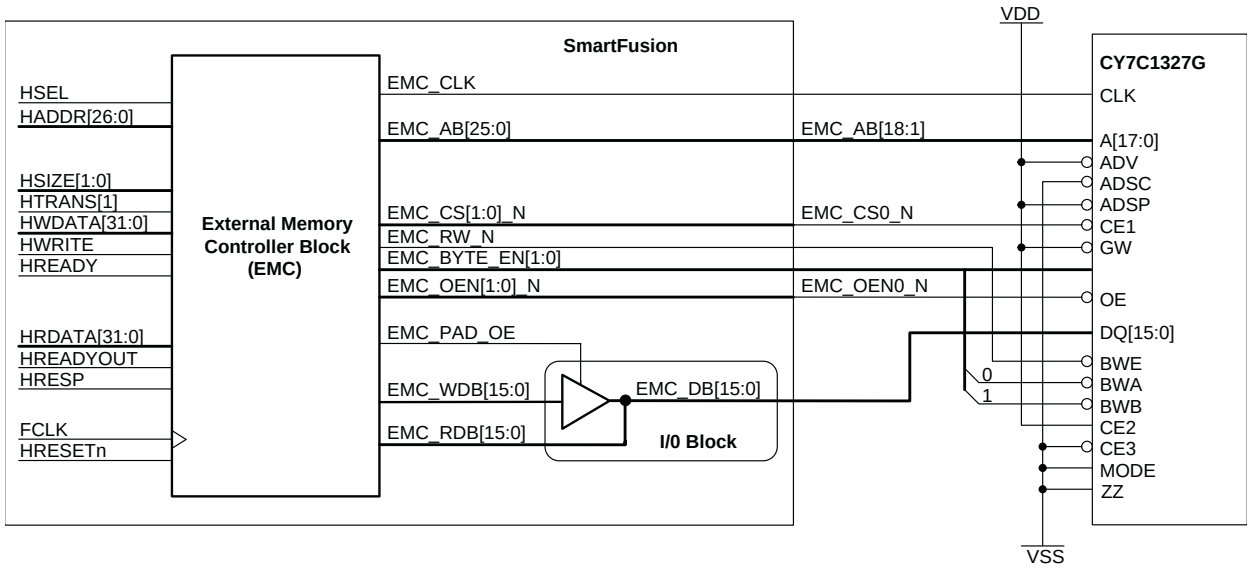


Figure 7-12 • **x16 Synchronous SRAM – 1 EMD**

The circuit of Figure 7-13 shows a representative configuration of synchronous SRAM for the SmartFusion EMC. Eight megabyte SSRAMS are shown. The address bus is again halfword aligned (A[21:0] = EMC_AB[22:1], since EMC_AB is a byte address).

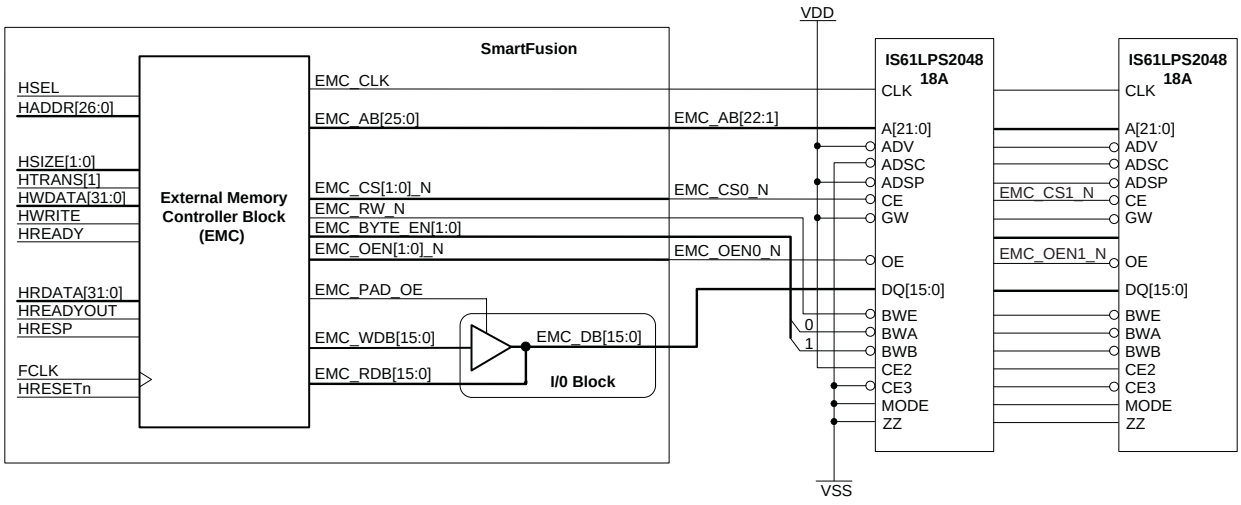


Figure 7-13 • **x16 Synchronous SRAM – 2 EMDs**

Figure 7-14 shows a circuit diagram that connects two large byte-mode NOR flash devices to the SmartFusion device. Here, the byte enables are being used as write enables (usage depends on device and configuration). Use the EMC_WENBENx bits in the EMC_CS_x_CR register to select this mode.

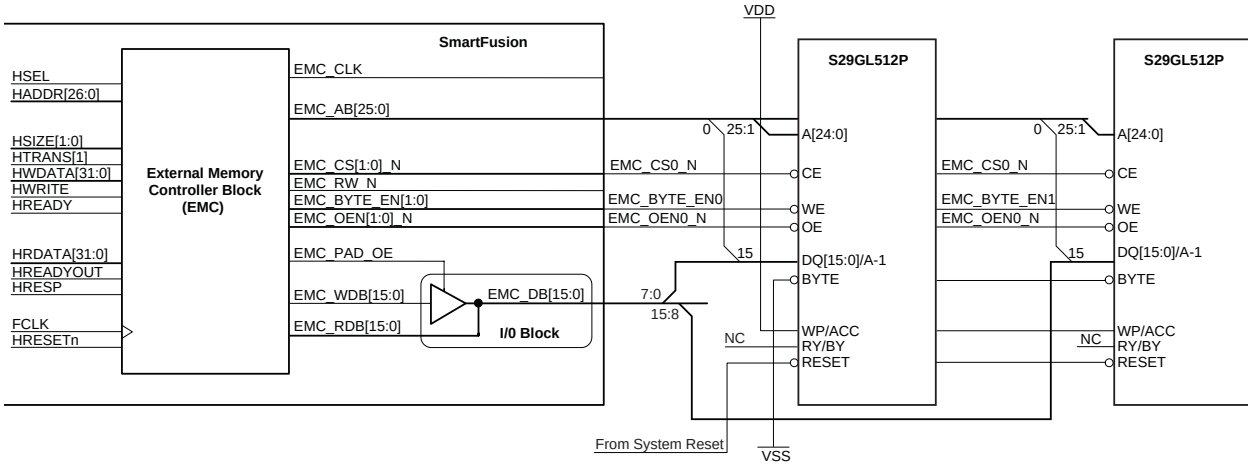


Figure 7-14 • **x16 NOR Flash – Two EMDs**

Figure 7-15 shows a circuit diagram that connects four asynchronous SRAMs to the SmartFusion device. This is another case of byte enables being used as write enables (usage depends on device and configuration).

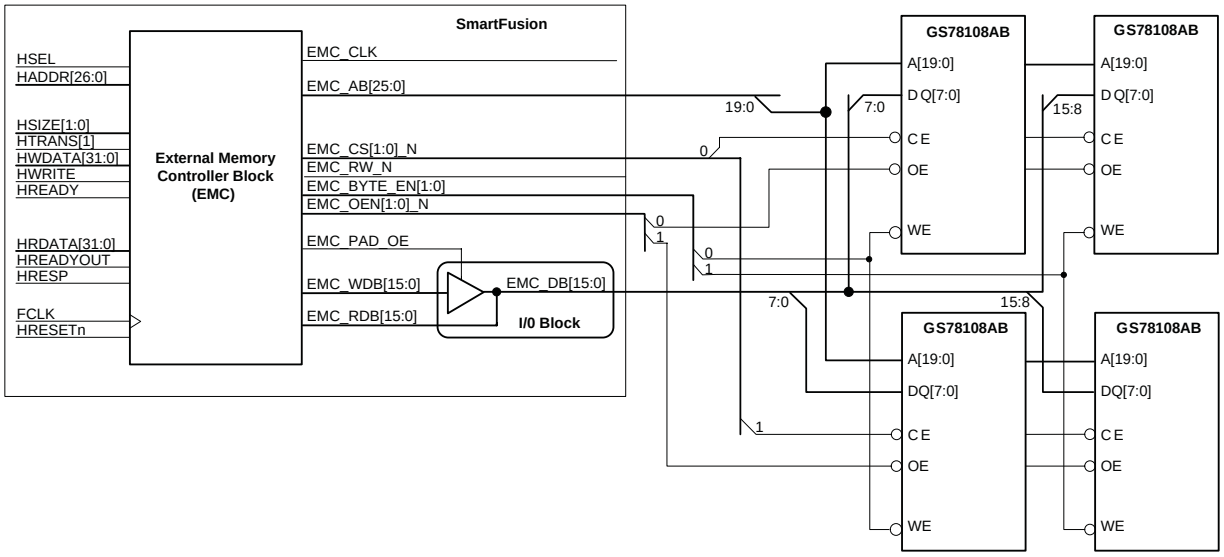


Figure 7-15 • **Asynchronous SRAMs – Four Byte-Wide EMDs**

The circuit of Figure 7-16 shows a synchronous SRAM (halfword address) alongside and sharing the data bus with two byte-wide asynchronous SRAM devices. Again, in this case the SmartFusion byte enable pins are connected to the write enable pins of the asynchronous SRAM external devices.

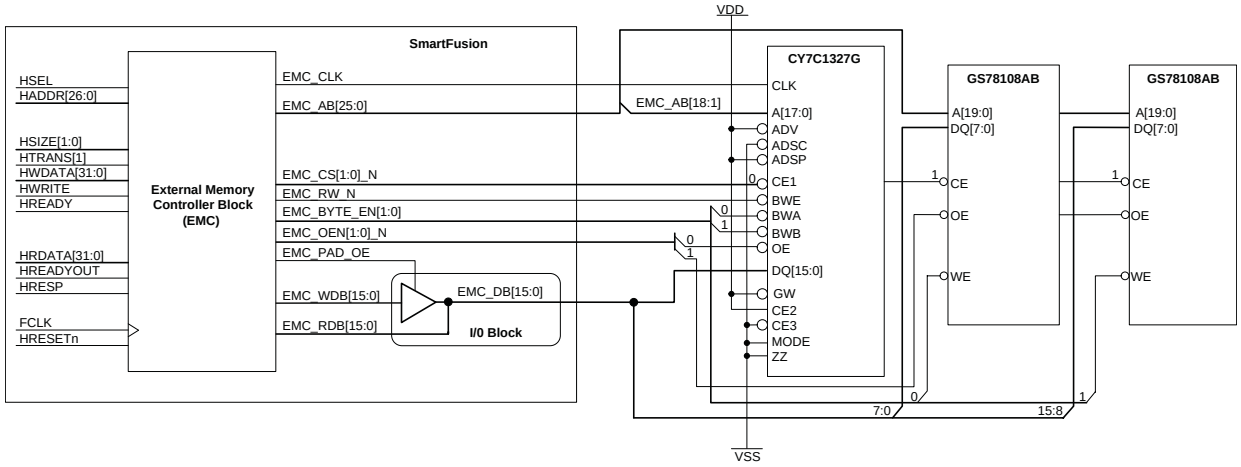


***Figure 7-16* • Synchronous SRAM – 1 x16 EMD and 2 x8 EMDs**

The circuit of Figure 7-17 shows a synchronous SRAM alongside and sharing the data bus with a NOR flash device. This drawing illustrates the connections necessary when the NOR flash device is configured in halfword (16-bit) mode. The user must not perform AHB byte accesses to the NOR flash device in this configuration (but can to the SRAM). While predictable, the results will likely be unsatisfactory.

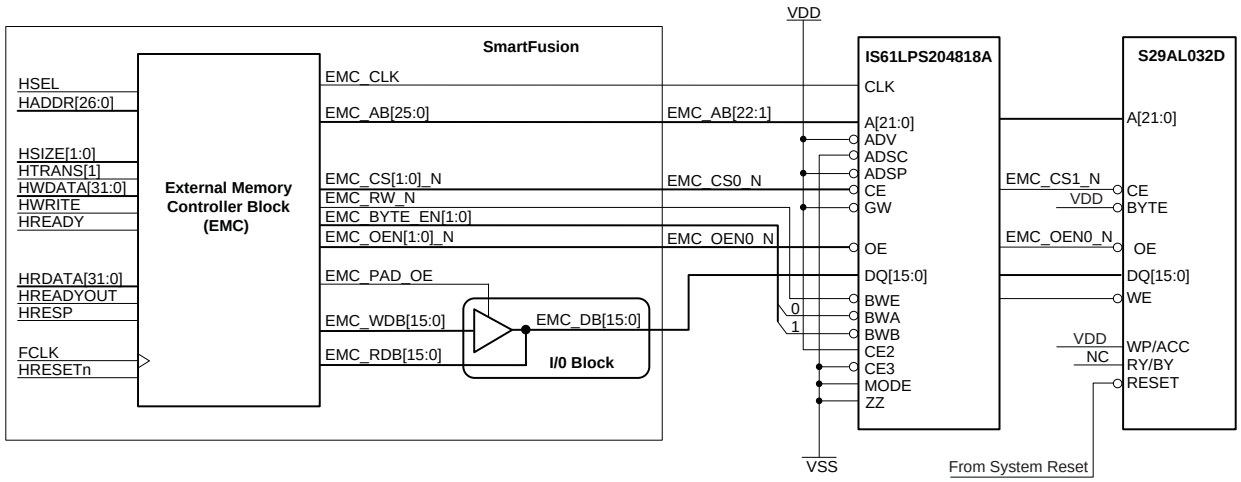


***Figure 7-17* • x16 Synchronous SRAM and x16 Flash**

External Memory Controller Configuration

Table 7-7 lists the control registers used in configuring the EMC. Table 7-8 gives the bit definition for EMC_MUX_CR and Table 7-9 lists the bit definitions for the EMC_CS_x_CR, where x is either 0 or 1, signifying chip select 0 or chip select 1.

***Table 7-7* • External Memory Controller Register Map**

Name	Address	R/W	Reset Value	Description
EMC_MUX_CR	0xE004203C	R/W	0x0	External memory controller MUX configuration
EMC_CS_0_CR	0xE0042040	R/W	0x0	EMC timing parameters for chip select 0
EMC_CS_1_CR	0xE0042044	R/W	0x0	EMC timing parameters for chip select 1

***Table 7-8* • EMC_MUX_CR**

Bit Number	Name	R/W	Reset Value	Description
0	EMC_SEL	R/W	0	Multiplexed EMC I/O control

***Table 7-9* • EMC_CS_x_CR**

Bit Number	Name	R/W	Reset Value	Description
21	EMC_CSFEx	R/W	0	Chip select falling edge
20	EMC_WENBENx	R/W	0	Write enable/byte enable
19	EMC_RWPOLx	R/W	0	Read/write polarity
18	EMC_PIPEWRNx	R/W	0	Pipelined write
17	EMC_PIPERDNx	R/W	0	Pipelined read
16:15	EMC_IDDx	R/W	0	Inter device delay
14:11	EMC_WRLATx	R/W	0	Write data latency
10:7	EMC_RDLATRESTx	R/W	0	Read data latency, next access
6:3	EMC_RDLATFIRSTx	R/W	0	Read data latency, first access
2	EMC_PORTSIZEx	R/W	0	Port size
1:0	EMC_MEMTYPEx	R/W	0	External memory type

Table 7-10 lists the different types of memory that can be seamlessly connected to the EMC. Users are free to assign any memory type to one chip select and any other memory type to the other chip select constrained only by the types listed in Table 7-10.

***Table 7-10* • EMC_MEMTYPEx Field Definition**

Bit 1	Bit 0	Function
0	0	No memory assigned to chip select x
0	1	Asynchronous/PSRAM memory assigned to chip select x
1	0	Synchronous memory assigned to chip select x
1	1	NOR flash memory assigned to chip select x

Valid configurations are illustrated in Table 7-11. If EMC_MEMTYPEx has a value of 0b00 and an AHB bus matrix master attempts to access that region of memory, the EMC will return an error and the COM_ERRORINTERRUPT (which maps to IRQ24 in the ARM Cortex-M3 interrupt controller) will be asserted. The specific master which caused the error can be determined by reading the MSS_SR located at address 0xE004201C. Specifically, the COMM_ERRORSTATUS field indicates which master generated the error condition. It should be noted that PSRAMS are only supported in asynchronous mode.

***Table 7-11* • Valid Combinations of EMD Types**

Chip Select 0	Chip Select 1
None	None
None	Asynchronous
None	Synchronous
None	NOR flash
Asynchronous	None
Asynchronous	Asynchronous
Asynchronous	Synchronous
Asynchronous	NOR flash
Synchronous	None
Synchronous	Asynchronous
Synchronous	Synchronous
Synchronous	NOR flash
NOR flash	None
NOR flash	Asynchronous
NOR flash	Synchronous
NOR flash	NOR flash

Table 7-12 defines EMC_PORTSIZEx. EMD data bus width is either 16 bits or 8 bits. AHB transactions on the AHB bus matrix can be up to 32 bits wide. Therefore, all AHB accesses wider than the configured EMD data bus width must be converted into halfword or byte EMD accesses, depending on the configured EMD bus width.

***Table 7-12* • EMC_PORTSIZEx Definition**

Bit 2	Function
0	8-bit EMD data bus
1	16-bit EMD data bus

Table 7-13 illustrates how the EMC converts AHB transactions into various EMC accesses based on EMD data bus width.

***Table 7-13* • Mapping of AHB Transactions to EMC Accesses**

EMC_PORTSIZEx	AHB Input Transaction		No. Accesses Required	EMC_BYTE_EN[1:0]	Access 1			Access 2			Access 3			Access 4		
	HSIZE[1:0]	HADDR[1:0]			EMC_AB[1:0]	HRDATA/HWDATA Bits	EMC_DB Bits	EMC_AB[1:0]	HRDATA/HWDATA Bits	EMC_DB Bits	EMC_AB[1:0]	HRDATA/HWDATA Bits	EMC_DB Bits	EMC_AB[1:0]	HRDATA/HWDATA Bits	EMC_DB Bits
8	Byte	0	1	0b10	0	[7:0]	[7:0]	–	–	–	–	–	–	–	–	–
8	Byte	1	1	0b10	1	[15:8]	[7:0]	–	–	–	–	–	–	–	–	–
8	Byte	2	1	0b10	2	[23:16]	[7:0]	–	–	–	–	–	–	–	–	–
8	Byte	3	1	0b10	3	[31:24]	[7:0]	–	–	–	–	–	–	–	–	–
8	Half	0	2	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	–	–	–	–	–	–
8	Half	1*	2	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	–	–	–	–	–	–
8	Half	2	2	0b10	2	[23:16]	[7:0]	3	[31:24]	[7:0]	–	–	–	–	–	–
8	Half	3*	2	0b10	2	[23:16]	[7:0]	3	[31:24]	[7:0]	–	–	–	–	–	–
8	Word	0	4	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	2	[23:16]	[7:0]	3	[31:24]	[7:0]
8	Word	1*	4	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	2	[23:16]	[7:0]	3	[31:24]	[7:0]
8	Word	2*	4	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	2	[23:16]	[7:0]	3	[31:24]	[7:0]
8	Word	3*	4	0b10	0	[7:0]	[7:0]	1	[15:8]	[7:0]	2	[23:16]	[7:0]	3	[31:24]	[7:0]
16	Byte	0	1	0b10	0	[7:0]	[7:0]	–	–	–	–	–	–	–	–	–
16	Byte	1	1	0b01	1	[15:8]	[15:8]	–	–	–	–	–	–	–	–	–
16	Byte	2	1	0b10	2	[23:16]	[7:0]	–	–	–	–	–	–	–	–	–
16	Byte	3	1	0b01	3	[31:24]	[15:8]	–	–	–	–	–	–	–	–	–
16	Half	0	1	0b00	0	[15:0]	[15:0]	–	–	–	–	–	–	–	–	–
16	Half	1*	1	0b00	0	[15:0]	[15:0]	–	–	–	–	–	–	–	–	–
16	Half	2	1	0b00	2	[31:16]	[15:0]	–	–	–	–	–	–	–	–	–
16	Half	3*	1	0b00	2	[31:16]	[15:0]	–	–	–	–	–	–	–	–	–
16	Word	0	2	0b00	0	[15:0]	[15:0]	2	[31:16]	[15:0]	–	–	–	–	–	–
16	Word	1*	2	0b00	0	[15:0]	[15:0]	2	[31:16]	[15:0]	–	–	–	–	–	–
16	Word	2*	2	0b00	0	[31:16]	[15:0]	2	[31:16]	[15:0]	–	–	–	–	–	–
16	Word	3*	2	0b00	0	[31:16]	[15:0]	2	[31:16]	[15:0]	–	–	–	–	–	–

*Note: *Misaligned access*

Note:

In the following sections and timing diagrams, W = EMC_WRLATx, X = EMC_RDLATFIRSTx, Y = EMC_RDLATRESTx, and Z = EMC_IDDx.

Read Latency – Initial Access (EMC_RDLATFIRSTx)

The field EMC_RDLATFIRSTx determines the number of initial latency cycles for the first read access for the respective chip select, where x defines which chip select is applicable. Each cycle is defined to be 1 FCLK period. Zero to a maximum of 15 latency cycles can be programmed by the user.

For asynchronous and flash memory types, EMC_RDLATFIRSTx read latency cycles are inserted between clock edges 2 and 2 + X, as shown in Figure 7-18 on page 101, Asynchronous Read Cycle, where X = the value programmed into EMC_RDLATFIRSTx. Note that for X = 0, the shown latency cycle is removed from the timing diagram.

For synchronous memory types with EMC_PIPERDNx = 1, initial read latency cycles are inserted between clock edges 2 and 2 + (X – 1), as shown in Figure 7-19 on page 102, Non-Pipelined Synchronous Read Cycle. Note that for X = 1, the shown latency cycle is removed from the timing diagram and for X = 0, the FCLK cycle between edges 2 + (X – 1) and 3 + (X – 1) is coincident with the FCLK cycle between edges 1 and 2.

For synchronous memory types with EMC_PIPERDNx = 0, initial read latency cycles are inserted between clock edges 2 and 2 + (X – 1) for rows 0–11 and between clock edges 5 and 5 + (X – 1) for rows 12 – 20, as shown in Figure 7-20 on page 103, Pipelined Synchronous Read Cycle. Note that for X = 1, the shown latency cycle is removed from the timing diagram.

Read Latency – Remaining Accesses (EMC_RDLATRESTx)

The field EMC_RDLATRESTx determines the number of latency cycles for the remaining read accesses for the respective chip select, where x defines which chip select is applicable. Each cycle is defined to be 1 FCLK period. Zero to a maximum of 15 latency cycles can be programmed by the user.

For asynchronous and flash memory types, the EMC_RDLATRESTx latency cycles are inserted between clock edges 4 + X and 4 + X + Y, as shown in Figure 7-18 on page 101, Asynchronous Read Cycle, where Y is the value programmed into EMC_RDLATRESTx. Read latency cycles for access 3 are inserted between edges 6 + X + Y and 6 + X + 2Y. Read latency cycles for access 4 are inserted between edges 8 + X + 2Y and 8 + X + 3Y. Note that for Y = 0, the shown latency cycle(s) are removed from the timing diagram.

For synchronous memory types with EMC_PIPERDNx = 1, EMC_RDLATRESTx latency cycles are inserted between clock edges 4 + (X – 1) and 4 + (X – 1) + (Y – 1), as shown in Figure 7-19 on page 102, Non-Pipelined Synchronous Read Cycle.

EMC_RDLATRESTx latency cycles for access 3 are inserted between edges 6 + (X – 1) + (Y – 1) and 6 + (X – 1) + 2(Y – 1).

EMC_RDLATRESTx latency cycles for access 4 are inserted between edges 8 + (X – 1)X + 2(Y – 1) and 8 + (X – 1) + 3(Y – 1). Note that for Y = 1, the shown latency cycle is removed from the timing diagram.

For synchronous memory types with EMC_PIPERDNx = 0, EMC_RDLATRESTx is ignored.

Write Latency (EMC_WRLATx)

The field EMC_WRLATx determines the number of write latency cycles for the respective chip select, where x defines which chip select is applicable. Each cycle is defined to be 1 FCLK period. From 0 to a maximum of 15 latency cycles can be programmed by the user.

For asynchronous and flash memory types, write latency cycles are inserted between clock edges that are shaded in Figure 7-21 on page 104, where W is the value programmed into EMC_WRLATx. Note that for W = 0, the shown latency cycle is removed from the timing diagram.

For synchronous memory types with EMC_PIPEWRNx = 1, write latency cycles are inserted between clock edges 2 and 2 + (W – 1), 4 + (W – 1) and 4 + 2(W – 1), 6 + 2(W – 1) and 6 + 3(W – 1), and 8 + 3(W – 1) and 8 + 4(W – 1), as shown in Figure 7-23 on page 106, Non-Pipelined Synchronous Write Cycle. Note that for X = 1, the shown latency cycle is removed from the timing diagram. For W = 0, the

FCLK cycle between edges 2 + (W – 1) and 3 + (W – 1) is coincident with the FCLK cycle between edges 1 and 2, and similarly for the other accesses.

For synchronous memory types with EMC_PIPEWRNx = 0, write latency cycles are inserted between clock edges 5 and 5 + (W – 1) for rows 0-11 and between clock edges 2 and 2 + (W – 1) for rows 12-20, as shown in Figure 7-22 on page 105, Pipelined Synchronous Write Cycle. Note that for W = 1, the shown latency cycle is removed from the timing diagram.

Pipelined Read Enable (EMC_PIPERDNx)

This bit enables pipelining of memory reads.

Note: For pipelined reads the address is incremented on each FCLK cycle. For non-pipelined reads, the address is only incremented after the AHB has latched the read data.

When EMC_PIPERDNx is asserted (Low), reads of synchronous SRAMs in the region controlled by chip select x (where x can be 0 or 1) are pipelined with timing per Figure 7-20 on page 103, Pipelined Synchronous Read Cycle.

When EMC_PIPERDNx is deasserted (High), reads of synchronous SRAMs in the region controlled by chip select x (where x can be 0 or 1) are not pipelined, with timing illustrated per Figure 7-19 on page 102, Non-Pipelined Synchronous Read Cycle.

EMC_PIPERDNx has no effect for asynchronous SRAMs and NOR flash devices.

Pipelined Write Enable (EMC_PIPEWRNx)

This bit enables pipelining of memory writes.

Note: For pipelined writes, the address is incremented on each FCLK cycle. For non-pipelined writes, the address is only incremented after the EMD has latched the write data.

When EMC_PIPEWRNx is asserted (Low), writes to synchronous SRAMs in the region controlled by chip select x (where x can be 0 or 1) are pipelined with timing per Figure 7-22 on page 105, Pipelined Synchronous Write Cycle.

When EMC_PIPEWRNx is deasserted (High), writes to synchronous SRAMs in the region controlled by chip select x (where x can be 0 or 1) are not pipelined, with timing illustrated per Figure 7-23 on page 106, Non-Pipelined Synchronous Write Cycle.

EMC_PIPEWRNx has no effect for asynchronous SRAMs and NOR flash devices.

Inter Device Delay (EMC_IDDx)

The field EMC_IDDx determines the number of latency cycles inserted between consecutive memory accesses and also defines the number of latencies when switching from synchronous to asynchronous memories and vice versa. Each cycle is defined to be 1 FCLK period. Zero to a maximum of 3 latency cycles can be programmed by the user.

The first IDD cycle occurs in the FCLK period following HREADYOUT assertion at the end of an EMC access for asynchronous reads and writes, flash reads and writes, and synchronous writes.

For synchronous reads, the first IDD cycle occurs in the FCLK period of HREADYOUT assertion at the end of an EMC access. Note that in practice this limits the IDD delay for synchronous reads to two FCLK periods, with EMC_IDDx = 0 having the same effect as EMC_IDDx = 1.

One IDD cycle is always inserted when the memory type changes from synchronous to flash or asynchronous, and one IDD cycle is also inserted when the memory type changes from flash or asynchronous to synchronous.

Alternate Chip Select Falling Edge (EMC_CSFEx)

When EMC_CSFEx = 0, EMC_CSx_N is asserted on the rising edge of FCLK. When EMC_CSFEx = 1, EMC_CSx_N is asserted on the falling edge of FCLK.

When EMC_CSFEx for the selected region is low, EMC_CSx_N is driven per the waveform shown on row 11 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

When EMC_CSFEx for the selected region is high, EMC_CSx_N is driven per the waveform shown on row 12 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

Read/Write Polarity (EMC_RWPOLx)

When EMC_RWPOLx = 0, the polarity of the EMC_RW_N is non-inverted. That is, reads are a high, writes are a low. When EMC_RWPOLx = 1, the polarity of the EMC_RW_N is inverted. That is, reads are a low and writes are a high.

When EMC_RWPOLx for the selected region is low, EMC_RW_N is driven per the waveform shown on row 13 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

When EMC_RWPOLx for the selected region is high, EMC_RW_N is driven per the waveform shown on row 14 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

Write Enable/Byte Enable (EMC_WENBENx)

EMC_WENBENx controls whether the byte lane enable signals (EMC_BYTE_ENx) serve as write enables (only asserted for writes) or byte enables (asserted for reads and writes.) When EMC_WENBENx = 0, EMC_BYTE_ENx is active for both reads and writes. When EMC_WENBENx = 1, EMC_BYTE_ENx is active for only writes.

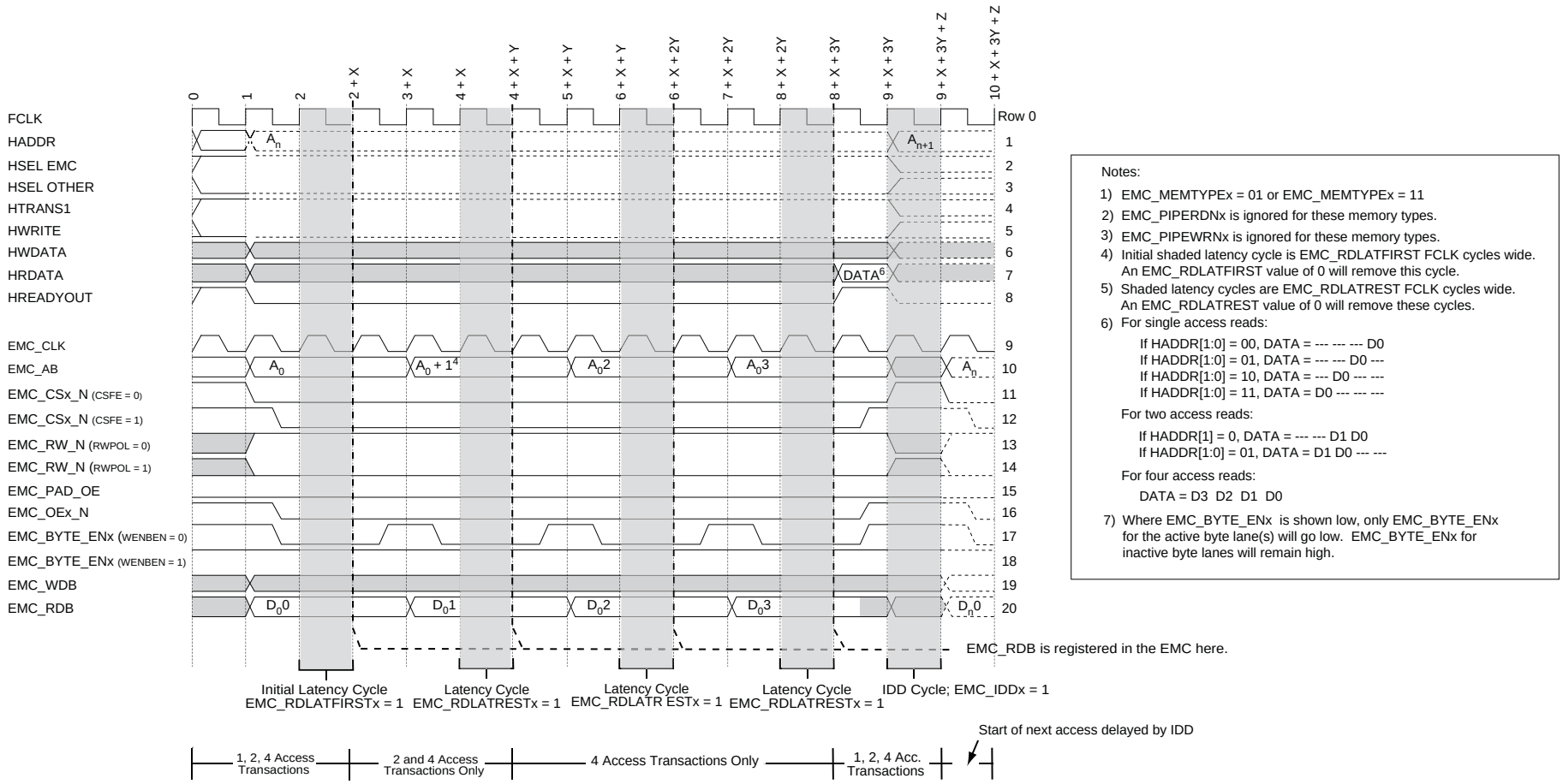
When EMC_WENBENx for the selected region is low, EMC_BYTE_ENx is driven per the waveform shown on row 17 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

When EMC_WENBENx for the selected region is high, EMC_BYTE_ENx is driven per the waveform shown on row 18 of Figure 7-18 on page 101 through Figure 7-22 on page 105 (using the figure appropriate to the access type).

Timing

The following timing diagrams in Figure 7-18 on page 101 through Figure 7-23 on page 106 show the operation of the EMC for each of the possible memory types. Note that the pipeline configuration for reads and writes is independent. A single memory can be configured to do pipelined reads and non-pipelined writes or vice versa.

Asynchronous Read Cycle



Notes:

1) EMC_MEMTYPEx = 01 or EMC_MEMTYPEx = 11
2) EMC_PIPERDNx is ignored for these memory types.
3) EMC_PIPEWRNx is ignored for these memory types.
4) Initial shaded latency cycle is EMC_RDLATFIRST FCLK cycles wide. An EMC_RDLATFIRST value of 0 will remove this cycle.
5) Shaded latency cycles are EMC_RDLATREST FCLK cycles wide. An EMC_RDLATREST value of 0 will remove these cycles.
6) For single access reads:

   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---

   For two access reads:

   If HADDR[1] = 0, DATA = --- --- D1 D0
   If HADDR[1:0] = 01, DATA = D1 D0 --- ---

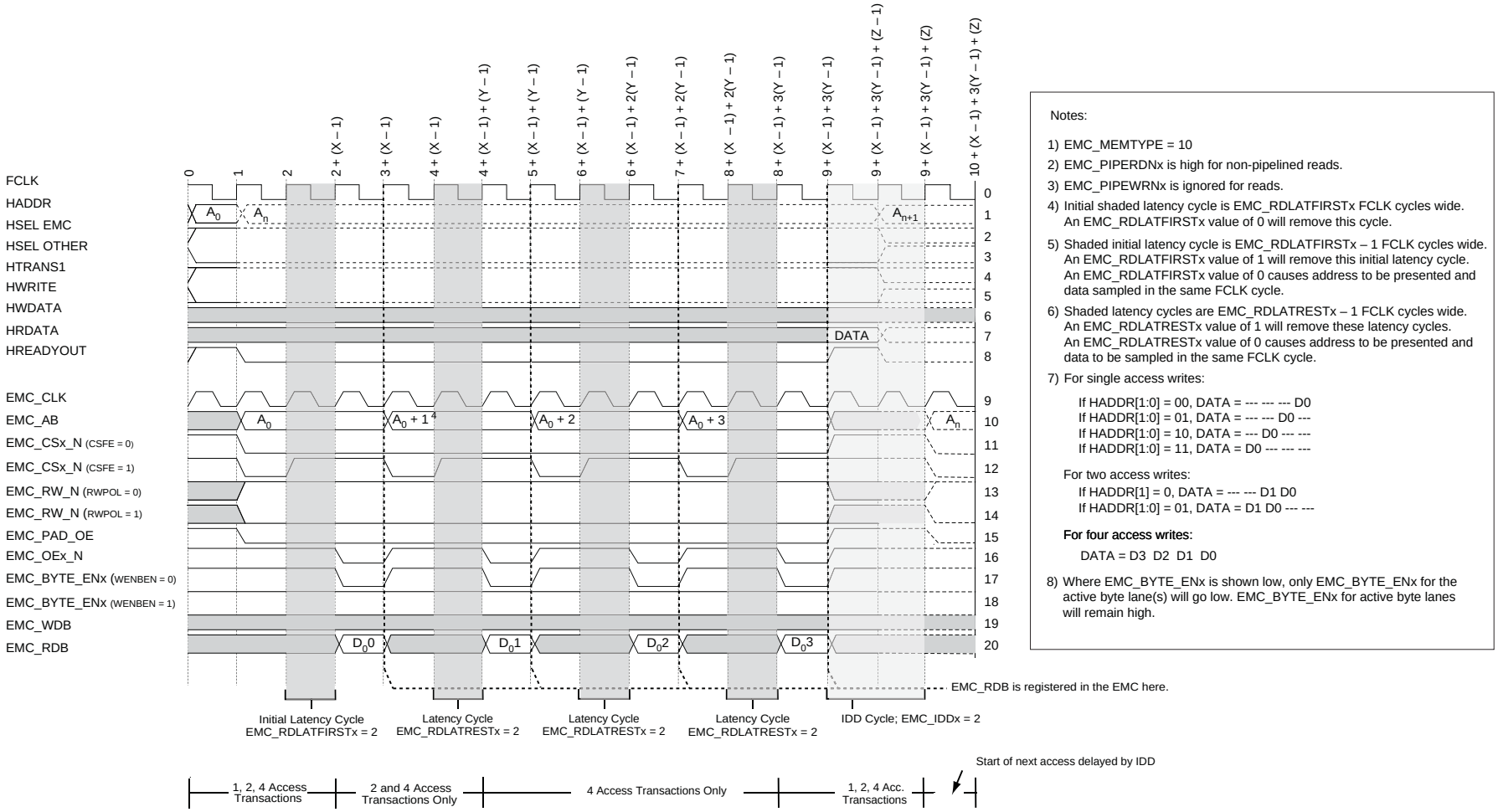
   For four access reads:

   DATA = D3 D2 D1 D0

7) Where EMC_BYTE_ENx is shown low, only EMC_BYTE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for inactive byte lanes will remain high.

Figure 7-18 • **Asynchronous Read Cycle**

Non-Pipelined Synchronous Read Cycle

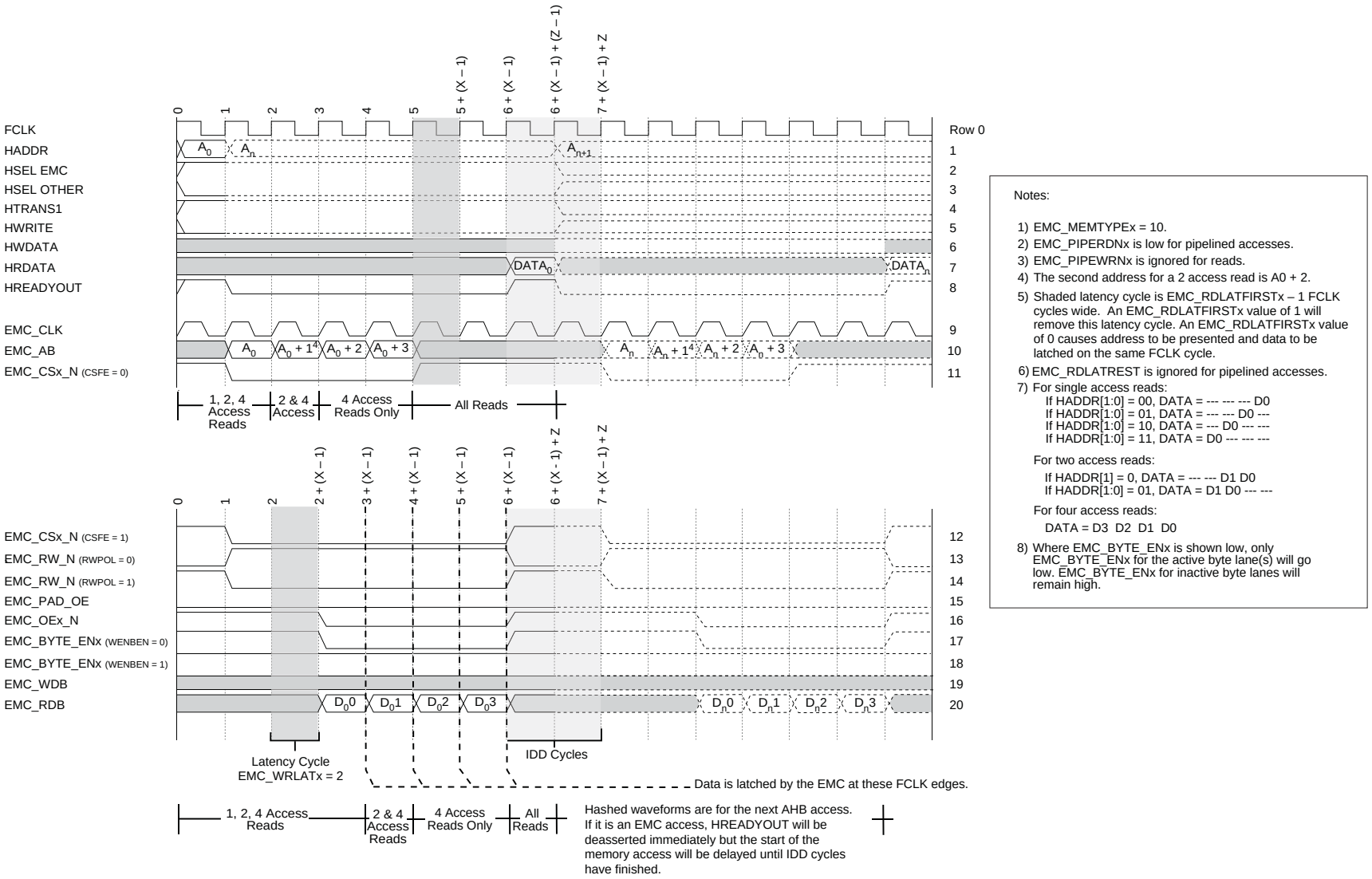


Notes:

1) EMC_MEMTYPE = 10
2) EMC_PIPERDNx is high for non-pipelined reads.
3) EMC_PIPEWRNx is ignored for reads.
4) Initial shaded latency cycle is EMC_RDLATFIRSTx FCLK cycles wide.
   An EMC_RDLATFIRSTx value of 0 will remove this cycle.
5) Shaded initial latency cycle is EMC_RDLATFIRSTx – 1 FCLK cycles wide.
   An EMC_RDLATFIRSTx value of 1 will remove this initial latency cycle.
   An EMC_RDLATFIRSTx value of 0 causes address to be presented and data sampled in the same FCLK cycle.
6) Shaded latency cycles are EMC_RDLATRESTx – 1 FCLK cycles wide.
   An EMC_RDLATRESTx value of 1 will remove these latency cycles.
   An EMC_RDLATRESTx value of 0 causes address to be presented and data to be sampled in the same FCLK cycle.
7) For single access writes:
   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---
   For two access writes:
   If HADDR[1] = 0, DATA = --- --- D1 D0
   If HADDR[1:0] = 01, DATA = D1 D0 --- ---
   For four access writes:
   DATA = D3 D2 D1 D0
8) Where EMC_BYTE_ENx is shown low, only EMC_BYTE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for active byte lanes will remain high.

Figure 7-19 • **Non-Pipelined Synchronous Read Cycle**

Pipelined Synchronous Read Cycle



Notes:

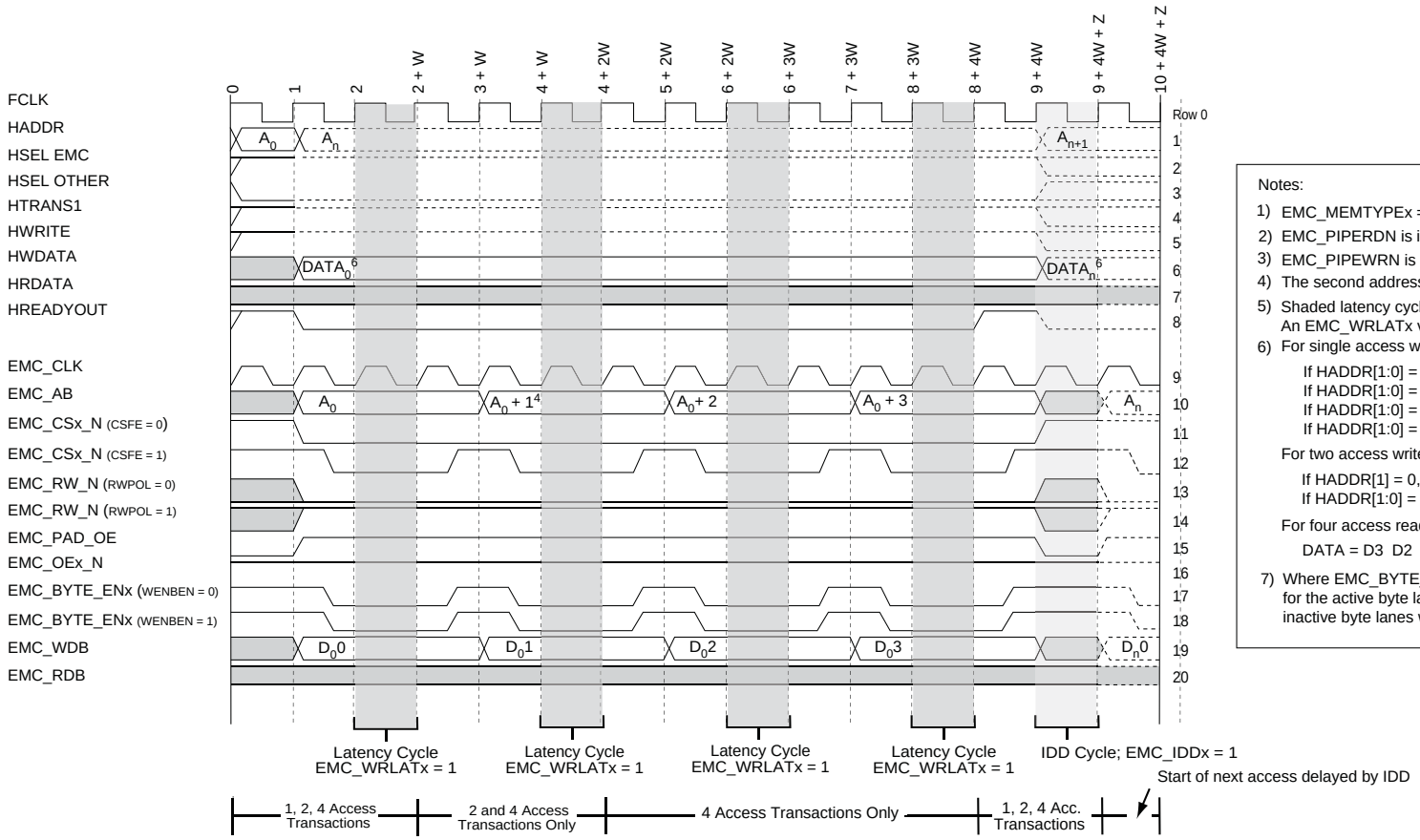
1) EMC_MEMTYPEx = 10.
2) EMC_PIPERDNx is low for pipelined accesses.
3) EMC_PIPEWRNx is ignored for reads.
4) The second address for a 2 access read is A0 + 2.
5) Shaded latency cycle is EMC_RDLATFIRSTx – 1 FCLK cycles wide. An EMC_RDLATFIRSTx value of 1 will remove this latency cycle. An EMC_RDLATFIRSTx value of 0 causes address to be presented and data to be latched on the same FCLK cycle.
6) EMC_RDLATREST is ignored for pipelined accesses.
7) For single access reads:
   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---

   For two access reads:
   If HADDR[1] = 0, DATA = --- --- D1 D0
   If HADDR[1:0] = 01, DATA = D1 D0 --- ---

   For four access reads:
   DATA = D3 D2 D1 D0
8) Where EMC_BYTE_ENx is shown low, only EMC_BYTE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for inactive byte lanes will remain high.

Figure 7-20 • **Pipelined Synchronous Read Cycle**

Asynchronous Write Cycle



Notes:

1) EMC_MEMTYPEx = 01 or EMC_MEMTYPEx = 11
2) EMC_PIPERDN is ignored for these memory types.
3) EMC_PIPEWRN is ignored for these memory types.
4) The second address for a 2 access read is A_0 + 2.
5) Shaded latency cycles are EMC_WRLATx FCLK cycles wide. An EMC_WRLATx value of 0 will remove these cycles.
6) For single access writes:

   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---

   For two access writes:

   If HADDR[1] = 0, DATA = --- --- D1 D0
   If HADDR[1:0] = 01, DATA = D1 D0 --- ---

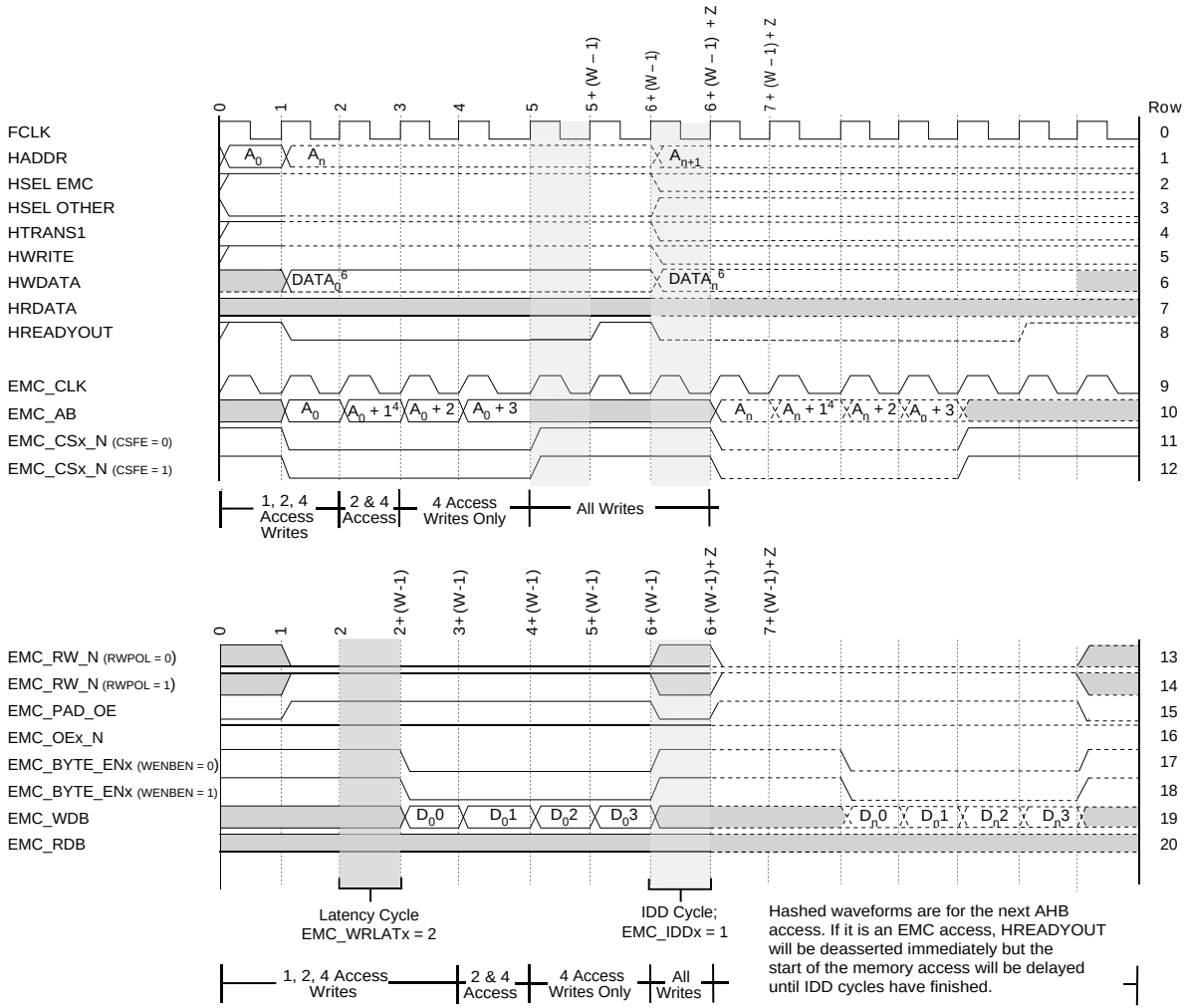
   For four access reads:

   DATA = D3 D2 D1 D0

7) Where EMC_BYTE_ENx is shown low, only EMC_BYTE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for inactive byte lanes will remain high.

Figure 7-21 • **Asynchronous Write Cycle**

Pipelined Synchronous Write Cycle

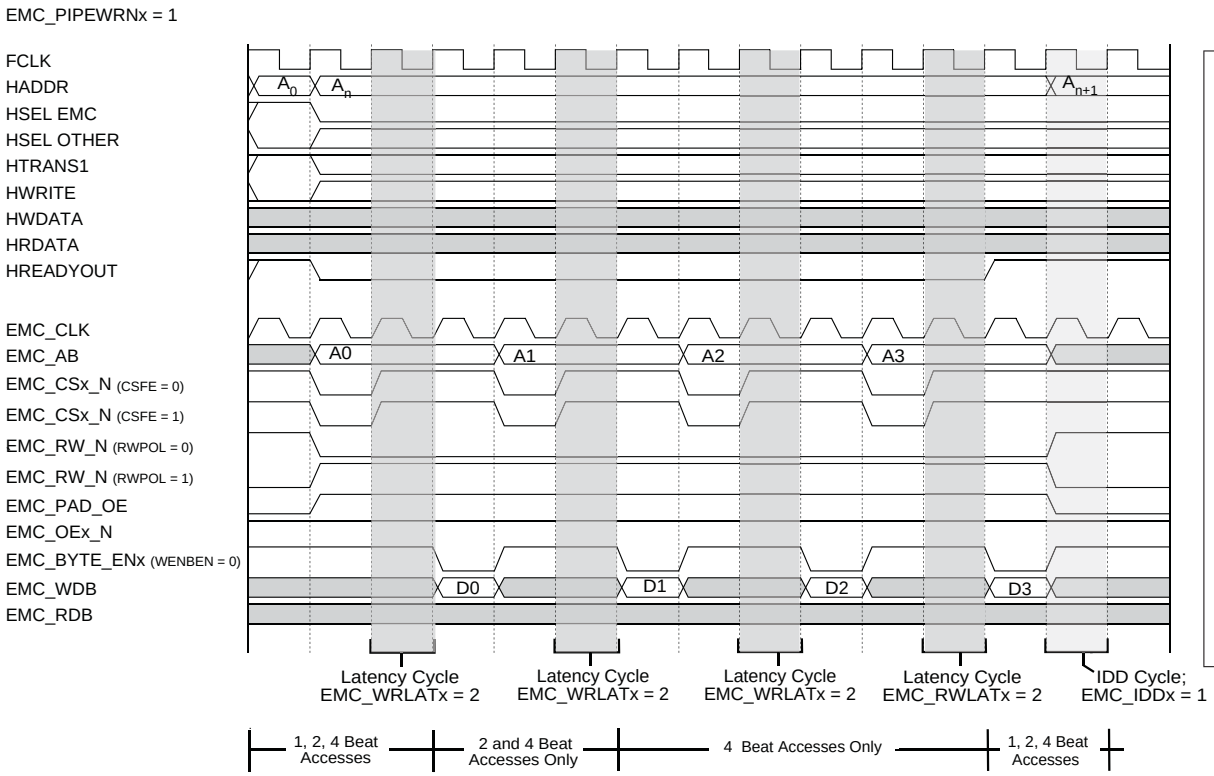


Notes:

1) EMC_MEMTYPEx =10
2) EMC_PIPERDNx is ignored for writes.
3) EMC_PIPEWRNx is low for pipelined writes.
4) The second address for a 2 access read is A_0+2.
5) Shaded latency cycles are EMC_WRLAT – 1 FCLK cycles wide. An EMC_WRLAT value of 1 will remove these latency cycles. An EMC_WRLAT value of 0 causes address and data to be presented in the same FCLK cycle.
6) For single access writes:
   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---
   For two access writes:
   If HADDR[1] = 0, DATA = --- --- D1 D0
   IF HADDR[1:0] = 0,1, DATA = D1 D0 --- ---
   For four access writes:
   DATA = D3 D2 D1 D0
7) Where EMC_BYTE_ENx is shown low, only EMC_BYTE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for inactive byte lanes will remain high.

Figure 7-22 • **Pipelined Synchronous Write Cycle**

Non-Pipelined Synchronous Write Cycle



Notes:

1) EMC_MEMTYPE = 10
2) EMC_PIPERDN is ignored for writes.
3) EMC_PIPEWRNx is high.
4) Shaded latency cycles are EMC_WRLATx – 1 FCLK cycles wide. An EMC_WRLATx value of 1 will remove these latency cycles.
5) An EMC_WRLATx value of 0 causes address and data to be presented in the same FCLK cycle.
6) For single beat writes:

   If HADDR[1:0] = 00, DATA = --- --- --- D0
   If HADDR[1:0] = 01, DATA = --- --- D0 ---
   If HADDR[1:0] = 10, DATA = --- D0 --- ---
   If HADDR[1:0] = 11, DATA = D0 --- --- ---

   For two beat writes:

   If HADDR[1] = 0, DATA = --- --- D1 D0
   If HADDR[1:0] = 01, DATA = D1 D0 --- ---

   For four beat writes:
   DATA = D3 D2 D1 D0
7) Where EMC_BYTE_ENx is shown low, only EMC_BYE_ENx for the active byte lane(s) will go low. EMC_BYTE_ENx for inactive byte lanes will remain high.

Figure 7-23 • **Non-Pipelined Synchronous Write Cycle**

External Memory Controller I/Os

I/Os used for the EMC are found on the north side and west side of the FPGA device. These I/Os are shared with user logic. That is, if the user does not need the EMC, I/Os are available for FPGA logic resources. If, however, the EMC is used, those FPGA I/Os are dedicated to the EMC. The EMC _SEL bit in the EMC_MUX_CR register is used to select either FPGA I/O or EMC I/O, as defined in Table 7-14. EMC_MUX_CR is located at address 0xE004203C in the system memory map. If the user sets the EMC_PORTSIZEx bit to 0 (8-bit memory) for both chip selects, the upper 8 bits of the data bus FPGA I/O are available to user logic.

***Table 7-14* • EMC I/O Configuration Control (EMC_MUX_CR)**

Bit 0	Function
0	The multiplexed I/Os are allocated to the FPGA logic.
1	The multiplexed I/Os are allocated to the EMC.

***Table 7-15* • EMC Pins**

Pin Name	Input/Output	Function	Count
EMC_AB[25:0]	Out	Address bus	26
EMC_DB[15:0]	Bidir	Bidirectional data bus	16
EMC_BYTE_EN[1:0]	Out	Byte lane signals	2
EMC_CS[1:0]_N	Out	Chip selects	2
EMC_OE[1:0]_N	Out	Output enables	2
EMC_RW_N	Out	Read/write	1
EMC_CLK	Out	Clock	1
			50

8 – PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators

This section describes the oscillators, phase-locked loops (PLLs), and clock conditioning circuitry (CCC) that exist in SmartFusion devices.

Functional Description

Figure 8-1 depicts the top-level SmartFusion clocking scheme. All SmartFusion devices have six CCC circuits and at least one PLL embedded in one of the CCCs, except for the A2F500 device, which has two PLLs (MSS_CCC and Fabric_CCC). The PLL in single-PLL devices is used to provide a flexible clocking scheme to the microcontroller subsystem (MSS) and the FPGA fabric. The additional PLL in other SmartFusion devices is dedicated to FPGA fabric usage. In all devices the MSS has been designed to operate at up to 100 MHz.

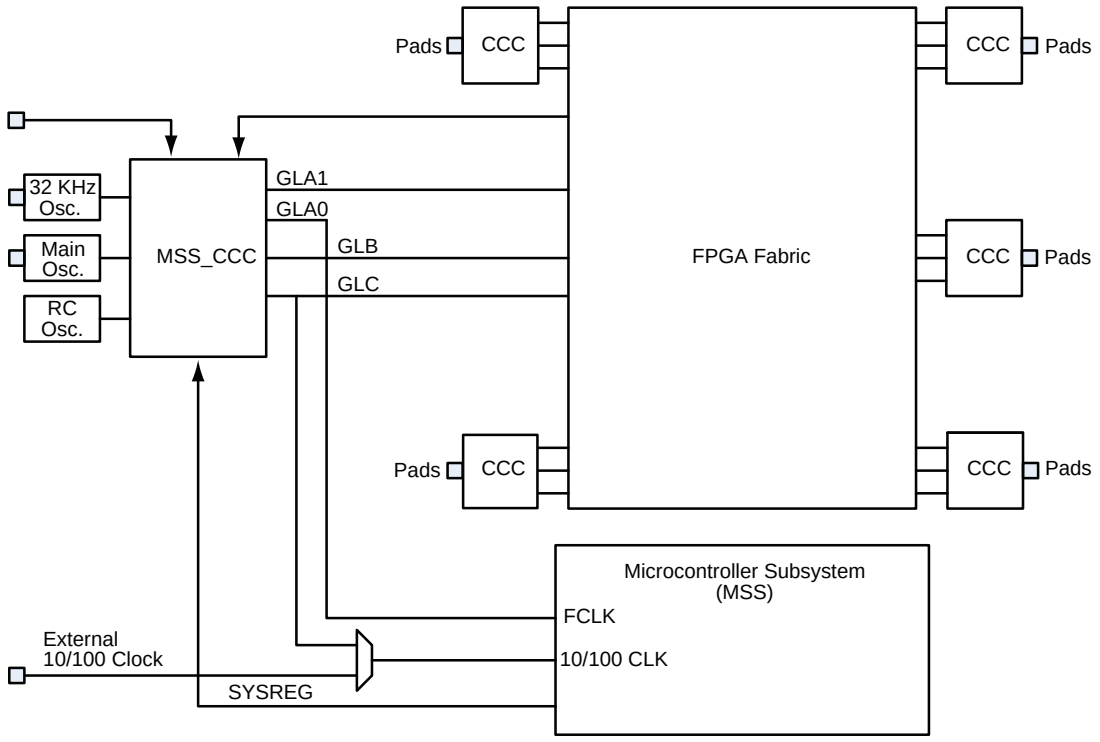


Figure 8-1 • **Top-Level SmartFusion Clock Hierarchy**

There are three internal oscillators that can be used to drive the MSS_CCC block: the 32 KHz low-power crystal oscillator, the main crystal oscillator, and the on-chip RC oscillator. The CCC blocks with an integrated PLL can only divide the input clock frequency; there are no minimum input frequency requirements when using the CCC block by itself without the PLL. Each PLL can divide/multiply its input clock to create a VCO frequency. Each PLL/CCC has three global outputs called GLA, GLB, and GLC. Each output includes a 5-bit divider that can be individually set to divide the VCO or input clock rate and create the output frequency for that connection.

In all devices, MSS_CCC drives the MSS. Once the ARM Cortex-M3 microcontroller is up and running, the firmware can choose to reconfigure the MSS_CCC to supply the MSS clock via the GLA0 output of

MSS_CCC. Within the MSS, FCLK can be further divided down via APB clock dividers and within those peripherals that will further divide down the APB clock.

The GLA0 output of the MSS_CCC block drives the input clock to the microcontroller subsystem (MSS). The clock source for the 10/100 Ethernet MAC can be sourced from an external pin or the GLC output of the MSS_CCC block, and the GLA1 and GLB outputs are dedicated to the FPGA fabric.

As depicted in Figure 8-2, the MSS_CCC block consists of the following main components: input clock multiplexers, PLL, dividers, and delays. There are three main paths through the MSS_CCC block: the CLKA, CLKB, and the CLKC paths, which output clocks onto the global buffers GLA, GLB, and GLC. As can be seen in more detail in Figure 8-3, there are actually two more outputs from the PLL/CCC block. The YB and YC outputs can drive additional local routing resources in the FPGA fabric. Figure 8-6 depicts a simplified view of the CCC blocks without a PLL.

Figure 8-2 • **Simplified View of MSS_CCC Block**

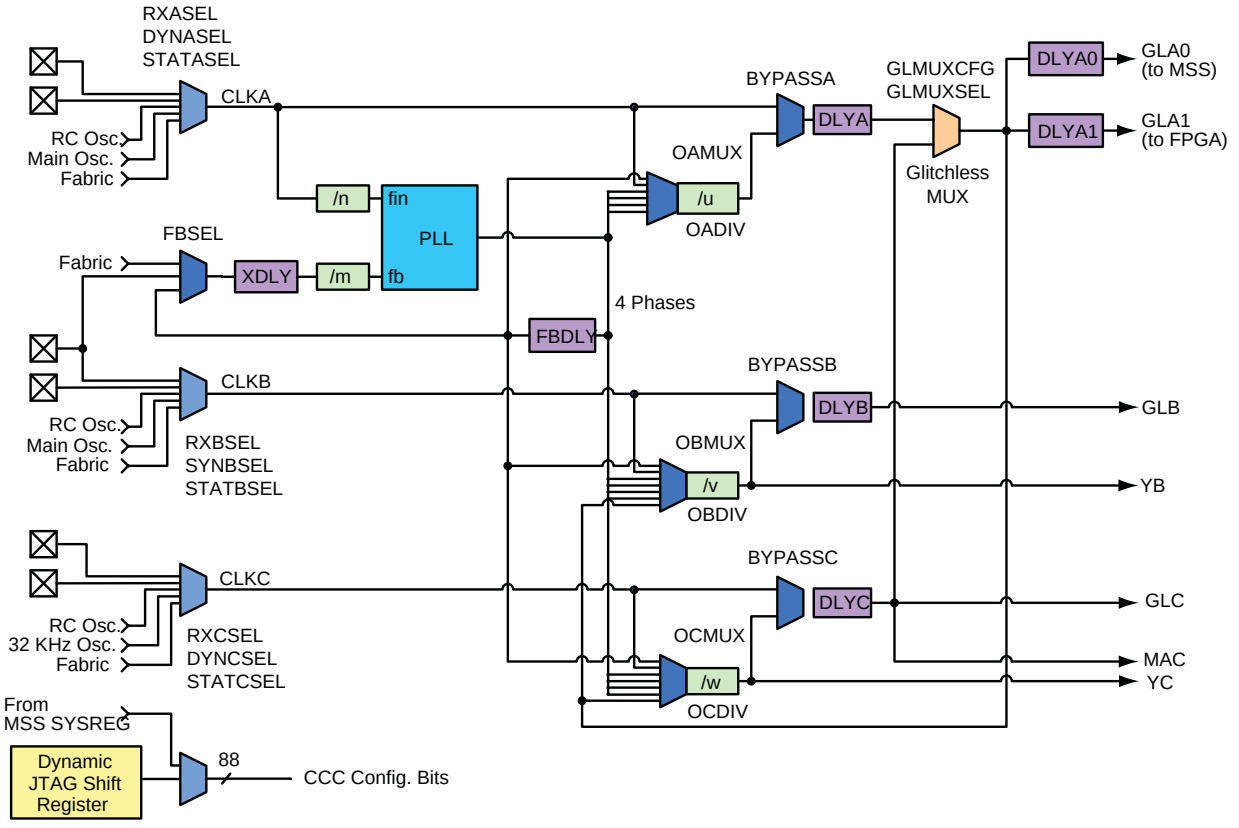


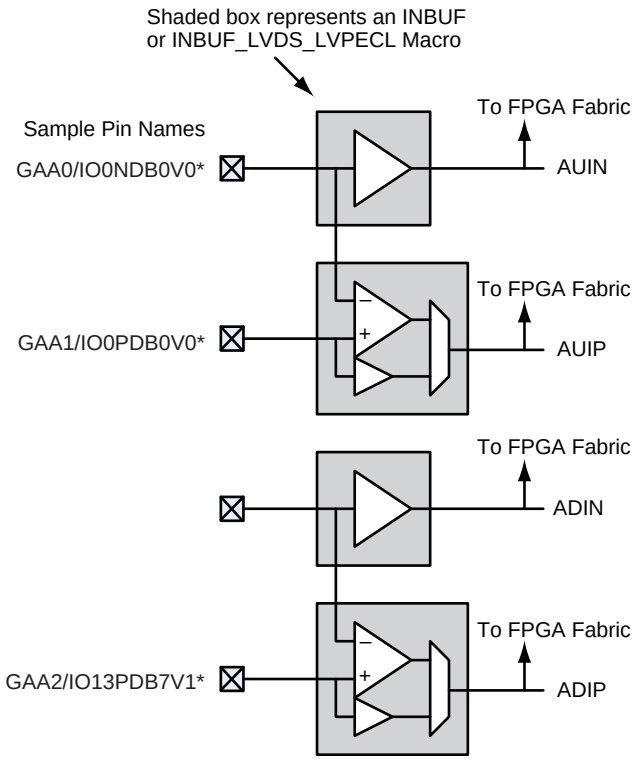
Figure 8-3 • **SmartFusion MSS_CCC Block**

Input Clock Selection

Each clock path has its own input multiplexer, allowing the user flexibility in choosing the clock source for that path. The input clock source can be changed dynamically by setting the appropriate control bits for the MSS_CCC_MUX_CR register. The control bits for CLKA are shown in Table 8-1 and the multiplexer arrangement is shown in Figure 8-5 on page 112. The input clock pads feeding the input clock multiplexers are shown in Figure 8-4. The Libero MSS Configurator configures the instantiation of the input buffer macros (depicted in Figure 8-5 on page 112) based on drop-down menu selections.

Table 8-1 • **CLKA Selection**

RXASEL	DYNASEL	STATASEL	CLKA
0	0	0	AUIN
0	1	0	AUIP
0	0	1	ADIP
0	1	1	GLAINT
1	0	X	RC oscillator
1	1	X	Main oscillator



*Note: *Represents the global input pins. Refer to the "User Pins" section of "Pin Descriptions" chapter in the SmartFusion Customizable System-on-Chip (cSoC) datasheet.*

Figure 8-4 • **Clock Input Sources for CLKA Multiplexer**

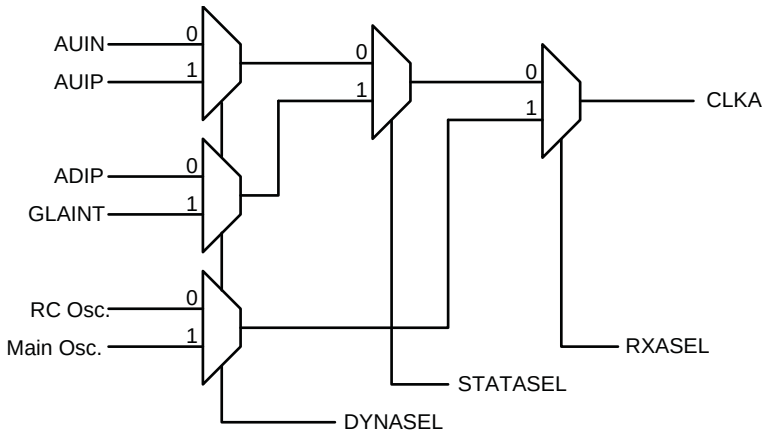


Figure 8-5 • **CLKA Multiplexers**

CLKA can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
  - AUIN, AUIP, ADIP
- Two dedicated differential I/Os using a hardwired connection
  - AUIN + AUIP pair, ADIN + ADIP pair
- The FPGA fabric
  - GLAINT
- The RC oscillator
- The Main oscillator

Similar to configuring CLKA, CLKB and CLKC have their own set of control bits to allow dynamic configuration of their clock sources.

Table 8-2 • **CLKB and CLKC Input Clock Sources**

RXBSEL	DYNBSEL	STATBSEL	CLKB	RXCSEL	DYNCSEL	STATCSEL	CLKC
0	0	0	BUIN	0	0	0	CUIN
0	1	0	BUIP	0	1	0	CUIP
0	0	1	BDIP	0	0	1	CDIP
0	1	1	GLBINT	0	1	1	GLCINT
1	0	X	RC oscillator	1	0	X	RC oscillator
1	1	X	Main oscillator	1	1	X	32 KHz oscillator

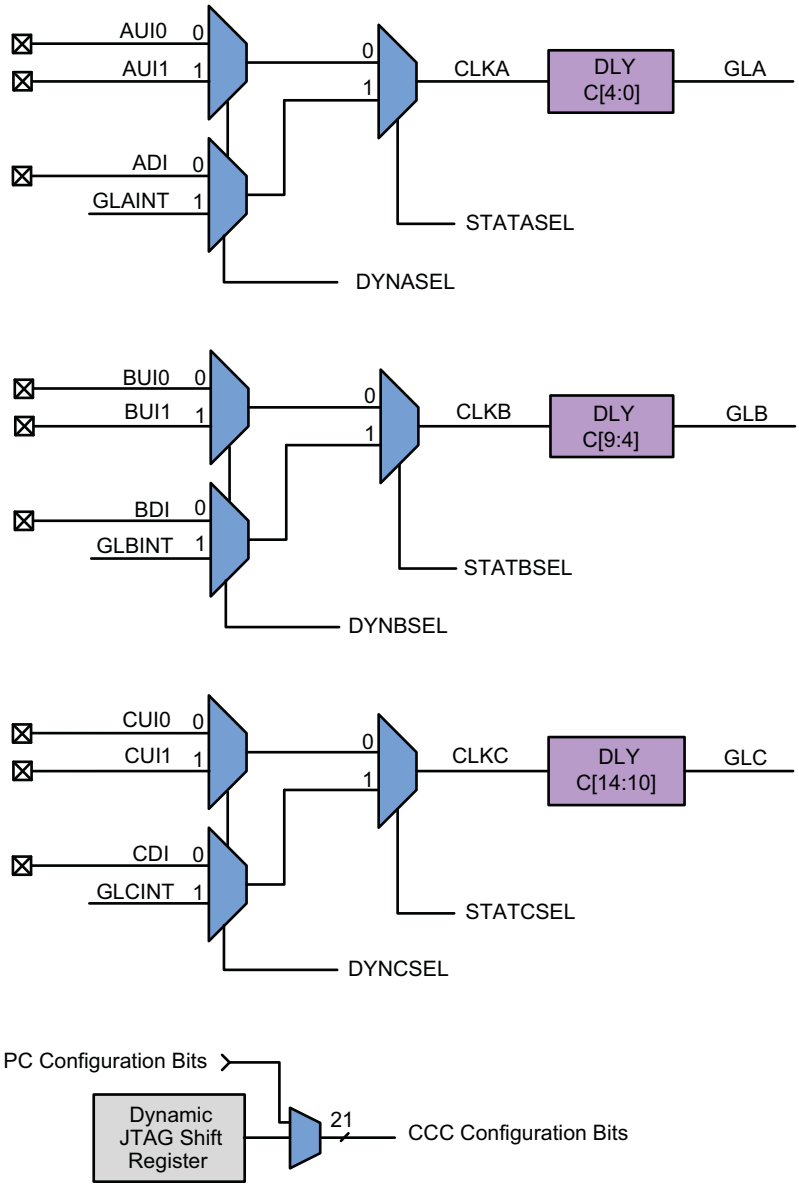


Figure 8-6 • **Simplified View of CCCs Without a PLL**

PLL Configuration

PLL Core Operating Principles

This section briefly describes the basic principles of PLL operation. The PLL core is composed of a phase detector (PD), a low-pass filter (LPF), and a four-phase voltage-controlled oscillator (VCO). Figure 8-7 illustrates a basic single-phase PLL core with a divider and delay in the feedback path.

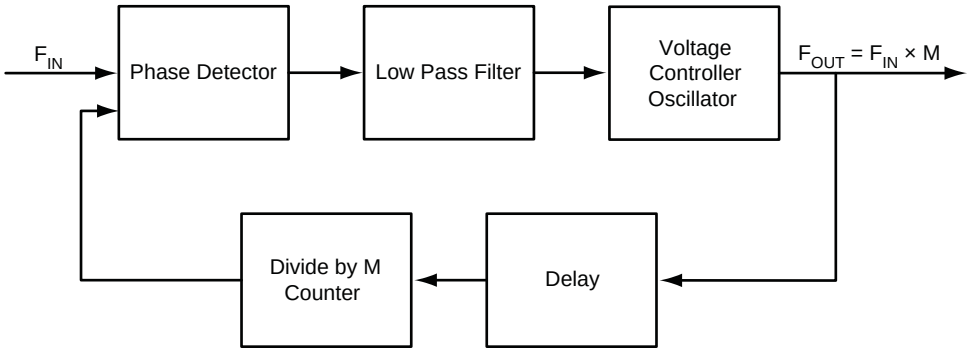


Figure 8-7 • **Simplified PLL with Feedback Divider and Delay**

The PLL is an electronic servo loop that phase-aligns the PD feedback signal with the reference input. To achieve this, the PLL dynamically adjusts the VCO output signal according to the average phase difference between the input and feedback signals. The first element is the PD, which produces a voltage proportional to the phase difference between its inputs. A simple example of a digital phase detector is an exclusive OR (XOR) gate. The second element, the LPF, extracts the average voltage from the phase detector and applies it to the VCO. This applied voltage alters the resonant frequency of the VCO, thus adjusting its output frequency.

Consider Figure 8-7 with the feedback path bypassing the divider and delay elements. If the LPF steadily applies a voltage to the VCO such that the output frequency is identical to the input frequency, this steady-state condition is known as lock. Note that the input and output phases are also identical. The PLL core sets a LOCK output signal High to indicate this condition. Should the input frequency increase slightly, the PD detects the frequency/phase difference between its reference and feedback input signals. Since the PD output is proportional to the phase difference, the change causes the output from the LPF to increase. This voltage change increases the resonant frequency of the VCO and increases the feedback frequency as a result. The PLL dynamically adjusts in this manner until the PD senses two phase-identical signals and steady-state lock is achieved. The opposite (decreasing PD output signal) occurs when the input frequency decreases.

Now suppose the feedback divider is inserted in the feedback path. As the division factor M is increased, the average phase difference increases. The average phase difference will cause the VCO to increase its frequency until the output signal is phase-identical to the input after undergoing division. In other words, lock in both frequency and phase is achieved when the output frequency is M times the input. Thus, clock division in the feedback path results in multiplication at the output. A similar argument can be made when the delay element is inserted into the feedback path. To achieve steady-state lock, the VCO output signal will be delayed by the input period *less* the feedback delay. For periodic signals, this is equivalent to time-advancing the output clock by the feedback delay. Another key parameter of a PLL system is the acquisition time. Acquisition time is the amount of time it takes for the PLL to achieve lock (phase-align the feedback signal with the input reference clock). For example, suppose there is no voltage applied to the VCO, allowing it to operate at its free-running frequency. If an input reference clock suddenly appears, a lock would be established within the maximum acquisition time.

Phase Selectors

The output from the PLL core can be phase-adjusted with respect to the reference input clock, CLKA. The user can select a 0°, 90°, 180°, or 270° phase shift independently for each of the outputs GLA, GLB/YB, and GLC/YC. Note that each of these phase-adjusted signals might also undergo further frequency division and/or time delay adjustment via the remaining dividers and delays located at the outputs of the CCC. Selecting the desired phase for each output is accomplished by writing to the OAMUX, OBMUX, and OXMUX fields of the MSS_CCC_MUX_CR control register.

Programmable Dividers

The PLL block contains five programmable dividers. Dividers *n* and *m* (the input divider and feedback divider, respectively) provide integer frequency division factors from 1 to 128. Dividers n and m correspond to the fields FINDIV and FBDIV in the MSS_CCC_PLL_CR control register.

The output dividers *u*, *v*, and *w* provide integer division factors from 1 to 32. Frequency scaling of the reference clock CLKA is performed according to the EQ 1 through EQ 3.

$$f_{GLA} = f_{CLKA} \times m / (n \times u)$$

EQ 1

$$f_{GLB} = f_{YB} = f_{CLKA} \times m / (n \times v)$$

EQ 2

$$f_{GLC} = f_{YC} = f_{CLKA} \times m / (n \times w)$$

EQ 3

Dividers u, v, and w correspond to the fields OADIV, OBDIV, and OCDIV in the MSS_CCC_DIV_CR control register. The Libero MSS Configurator provides a user-friendly method of generating the PLL settings, which includes automatically setting the division factors to achieve the closest possible match to the requested frequencies. The settings are used by the system startup code to initialize the MSS to a known state.

Since the five output clocks share the *n* and *m* dividers, the achievable output frequencies are interdependent and related according to EQ 4.

$$f_{GLA} = f_{GLB} \times (v / u) = f_{GLC} \times (w / u)$$

EQ 4

Programmable Delay Elements

There are a total of seven configurable delay elements implemented in the CCC architecture. Two of the delays are located in the feedback path: System Delay and Feedback Delay. System Delay, enabled by the XDLY control bit, provides a fixed delay of 2 ns (typical), and Feedback Delay, set by the FBDLY field in MSS_CCC_DLY_CR, provides selectable delay values from 0.535 ns to 5.56 ns in 200 ps increments (typical). For PLLs, delays in the feedback path will effectively advance the output signal from the PLL core with respect to the reference clock. Thus, the System (XDLY) and Feedback (FBDLY) delays generate negative delay on the output clock. Additionally, each of these delays can be independently bypassed if necessary.

At each global multiplexer output (GLA, GLB, and GLC) a delay element is available which is user-selectable from 0.735 ps in the first step; then to 5.56 ns with 200 ps increments after the first two steps. Setting these delays is done by writing to the DLYA, DLYB, and DLYC fields of the MSS_CCC_DLY_CR.

In addition to the above three delays there are two additional delays in series with GLA. GLA0 is a programmable delay element driving the microcontroller subsystem clock network and GLA1 is a programmable delay element driving the FPGA fabric GLA clock network. These two delay elements (DLYA0 and DLYA1) are used to allow edge alignment for hold time correction between the microcontroller subsystem and the fabric interface if GLA1 is used to clock the fabric interface controller (FIC). Setting these delays is done by writing to the DLYA0 and the DLYA1 fields of the MSS_CCC_DLY_CR. On power-up, DLYA0 and DLYA1 are set to their maximum value. System boot code provided by Microsemi, in concert with the Libero MSS Configurator, will initialize the delay values to factory-calibrated data.

Glitchless MUX (NGMUX)

The NGMUX is a 2:1 multiplexer that switches glitch-free between two different clock sources and outputs the new clock to the global network, as shown in Figure 8-8. Before switching the NGMUX, the clock source being switched to must be stable to avoid unstable clock propagation through the NGMUX. If not, the NGMUX can propagate those glitches. Table 8-3 shows the various clock sources available to the NGMUX. Switching from one clock source to another must complete before another clock source is selected. In other words, the NGMUX must propagate the clock switching before it can switch again. The output of the glitchless MUX will be undefined if the glitchless MUX is not allowed to complete the switch.

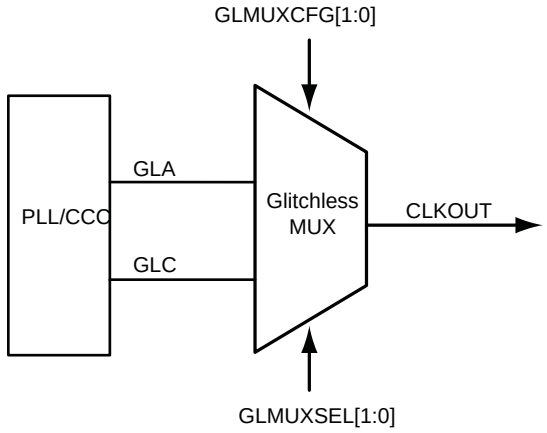


Figure 8-8 • **Glitchless Multiplexer**

Table 8-3 • **NGMUX Clock Sources**

GLMUXCFG		GLMUXSEL		
Bit 27	**Bit 26**	**Bit 25**	**Bit 24**	**Selected Input Signal**
0	0	X	0	GLA
		X	1	GLC
0	1	X	0	GLA
		X	1	Reserved
1	0	X	0	GLC
		X	1	Reserved
1	1	0	0	GLA
		0	1	GLC
		1	0	Reserved
		1	1	GND

Glitchless MUX Switching

Most users will find it is only necessary to leave GLMUXCFG at 0x03 (power-up default setting) and control the clock source by changing only GLMUXSEL. Figure 8-9 through Figure 8-11 on page 119 show the constraints that exist when switching between two clocks.

Case 1: Both Current Clock and Desired Clock Active

When both the current clock and desired clock inputs to the NGMUX are active, the switching sequence between the two clock sources (from current clock to desired clock) is as below. An example is shown in Figure 8-9.

1. A transition on S initiates the clock source switch.
2. GL drives one last complete current clock positive pulse (i.e., one rising edge followed by one falling edge).
3. GL stays Low until the second rising edge of desired clock occurs.

At the second desired clock rising edge, GL continuously delivers desired clock.

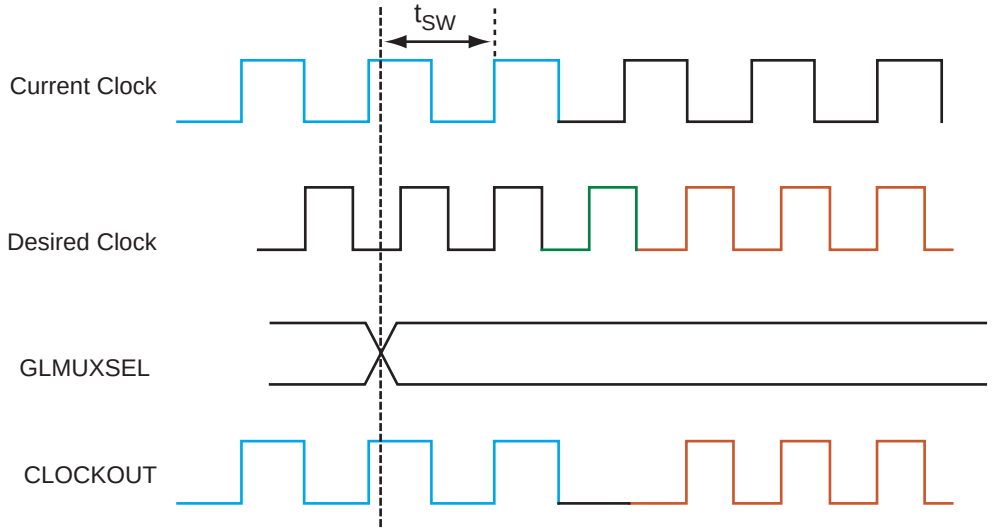


Figure 8-9 • **NGMUX Switching When Both Clocks Active**

Case 2: Current Clock Stopped or at Very Low Frequency

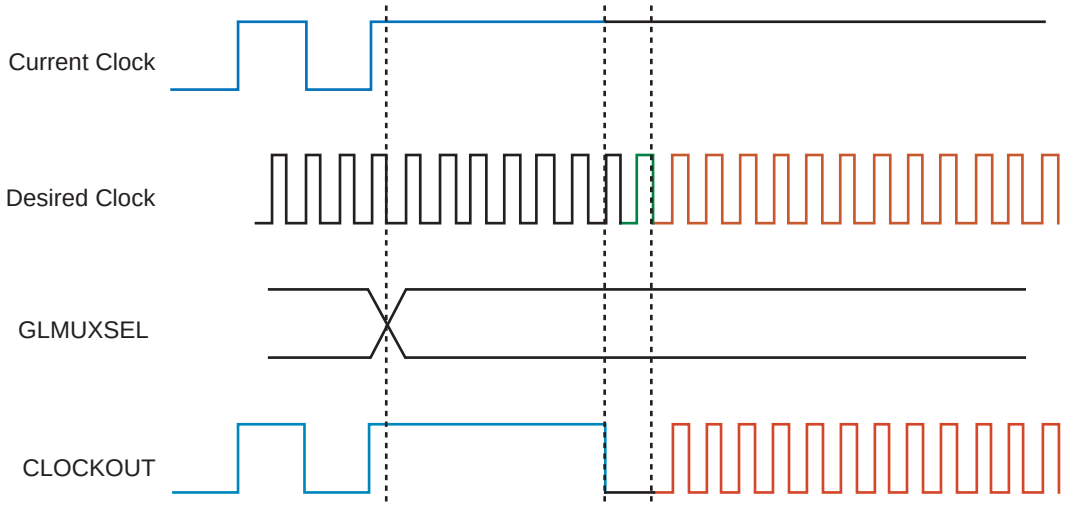
If the current clock stops or runs at a very low frequency after S transition, internal timeout circuitry will be used to complete the transition. The sequence of switching between the two clock sources (from current clock to desired clock) is described and illustrated below.

Case 2A: No Rising Current Clock Edge

If the current clock does not have a rising edge before the seventh desired clock rising edge, the switching sequence between the two clock sources (from current clock to desired clock) is as shown in Figure 8-10.

At the seventh desired clock rising edge, GL will go Low until the ninth desired clock rising edge.

At the ninth desired clock rising edge, GL will continuously deliver the desired clock signal.



Note: Min. t_{sw} = 0.05 ns at 25°C (typical conditions).

***Figure 8-10* • NGMUX Switching when No Rising Edge on Current Clock During Switching Window**

Case 2B: No Falling Current Clock Edge

If a current clock rising edge occurs before the seventh desired clock rising edge but a current clock falling edge does not occur before the fifteenth desired clock rising edge, the sequence of switching between the two clock sources (from current clock to desired clock) is as shown in Figure 8-11.

At the fifteenth desired clock rising edge, GL will go Low until the seventeenth desired clock rising edge.

At the seventeenth desired clock rising edge, GL will continuously deliver the desired clock signal.

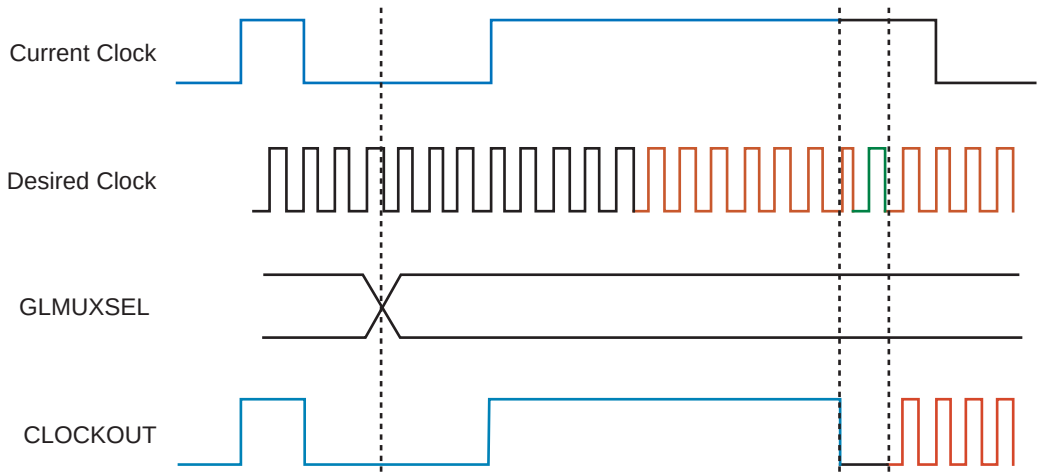


***Figure 8-11* • NGMUX Switching When No Falling Edge on Current Clock During Switching Window**

Safe Clock Switching Methods

On power-up, the clock for the MSS is sourced by the CLKC path. Specifically, the RC oscillator is selected, the *w* divider divides the 100 MHz RC by 4, and the glitchless MUX is set to select the global MUX GLC with the programmable delays feeding the GLA0 and GLA1 outputs set to their maximum. So, on power-up, GLC = GLA0 = GLA1 = 25 MHz. Microsemi-provided system boot code, in concert with the Libero MSS Configurator, provides for a safe switching methodology from power-on reset to the desired output clock frequency and source. The delay values, DLYA0 and DLYA1, are set to a factory calibration setting. It is strongly recommended that the user does not modify these delay values.

After power-up, if the user wishes to use a different clock source or change the clock frequency driving GLA0 or GLA1, the user must wait for the desired clock source to stabilize before switching the glitchless MUX from the old clock source to the new clock source.

For example, after power-up, change the clock frequency driving the MSS from a 25 MHz RC oscillator source to a 100 MHz RC oscillator source.

Step 1:

Write to MSS_CCC_MUX_CR to select the RC oscillator and bypass the PLL:

1. Set RXASEL bit to 0x1.
2. Clear DYNASEL to 0x0.
3. Set BYPASSA to 0x1.

Step 2:

Write to MSS_CCC_MUX_CR to select the GLA path:

Set GLMUXSEL bits to 0x0.

In the above example, it is not acceptable to change the divider value in OCDIV from 4 to 1 to effect the desired change from 25 MHz to 100 MHz. It is not acceptable to effect all changes targeted to the MSS_CCC_MUX_CR in one single write. The divider circuitry is not glitchless and would have passed a glitch along to the MSS. The switching of the glitchless MUX must occur last. Also, it was assumed that switching the CLKA path (input to the PLL) had no unintended consequences for the GLB output driving the FPGA fabric, if it is being used by the fabric.

In general, the frequency of the GLB, YB, GLC, and YC outputs to the FPGA fabric can be changed by the Cortex-M3 microcontroller. User logic in the FPGA fabric must be able to handle glitches from these potential changing clocks. A simple solution would be to have the Cortex-M3 microcontroller set a bit in FPGA fabric that user logic can use as a reset when the Cortex-M3 microcontroller is changing the clock sources to user logic.

On-Chip RC Oscillator

The on-chip RC oscillator (Figure 8-12) runs at a nominal frequency of 100 MHz. On power-up, the RC is used as the input clock to the microcontroller subsystem. At that time, the RC is divided by 4 through the *w* divider (OCDIV) and presented to GLA0 and GLA1 through the glitchless MUX. The RC oscillator is always turned on.

Note: The accuracy of the on-chip RC oscillator makes it unsuitable as a clock source for the Ethernet MAC.

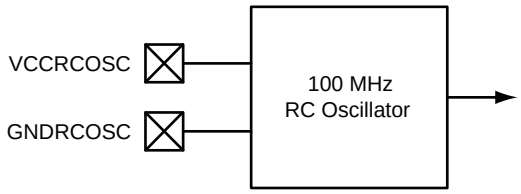


***Figure 8-12* • On-Chip RC Oscillator**

Main Crystal Oscillator

The on-chip crystal oscillator circuit works with an off-chip crystal to generate a high-precision clock and is capable of providing system clocks for peripherals and other system clock networks, both on-chip and off-chip.

The on-chip circuitry is designed to work with an external crystal, a ceramic resonator, or an RC network. It can only support one of these configurations at a time. The crystal oscillator supports four modes of operation, defined in Table 8-4. In RC Network mode, the oscillator is configured to work with an external RC network. The RC components are connected to the MAINXIN pin, with MAINXOUT left floating, as shown in Figure 8-13. The frequency generated by the circuit in RC Network mode is determined by the RC time constant of the selected components, as shown in Figure 8-15 on page 122.

In all other modes, the crystal oscillator is configured to support an external crystal or ceramic resonator. These modes correspond to low, medium, and high gain. They differ in the crystal or resonator frequency supported. The crystal or resonator is connected to the MAINXIN and MAINXOUT pins. Additionally, a capacitor is required on both MAINXIN and MAINXOUT pins to ground, as shown in Figure 8-14 on page 122. The recommended input capacitance is 22 pF.

The main crystal oscillator can be enabled and disabled by the Cortex-M3 microcontroller via the MSS_CCC_MUX_CR, bit 29 MAINOSCEN. When the main crystal oscillator is not being used, MAINXIN and MAINXOUT pins can be left floating.

***Table 8-4* • Main Oscillator Operational Modes**

MAINOSCMODE		Clock Mode	Function
Bit 31	Bit 30		
0	0	RC network	RC oscillation mode. Connect the RC network to the MAINXIN pad. The MAINXOUT pad should be disconnected.
0	1	Low gain	32 to 200 KHz low-power/-frequency mode. Oscillator consumes the least current of the three crystal modes.
1	0	Medium gain	0.20 to 2.0 MHz Standard crystal/resonator frequency
1	1	High gain	2.0 to 20.0 MHz high-frequency mode. Oscillator consumes the most current of the three modes.

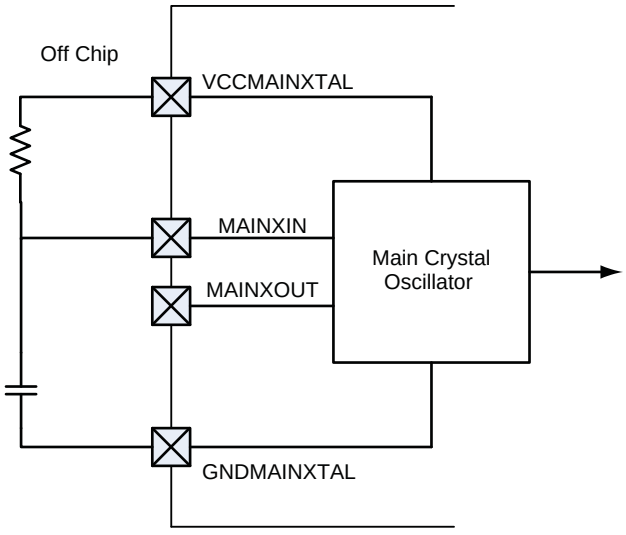


***Figure 8-13* • Main Crystal Oscillator in RC Network Mode**

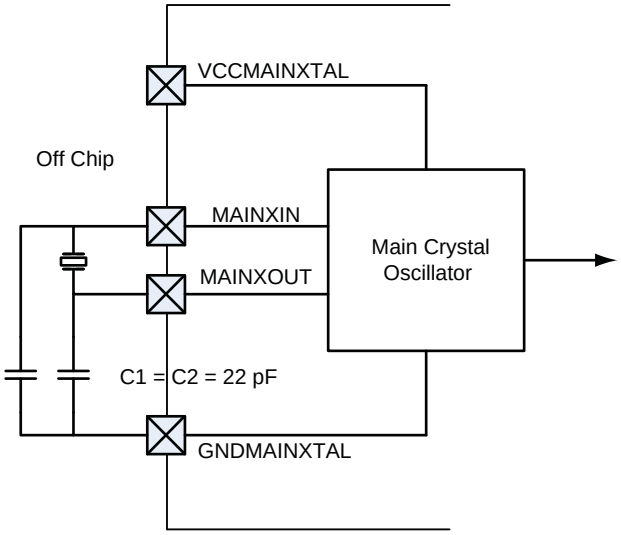


***Figure 8-14* • Main Crystal Oscillator in Ceramic Resonator or Crystal Mode**

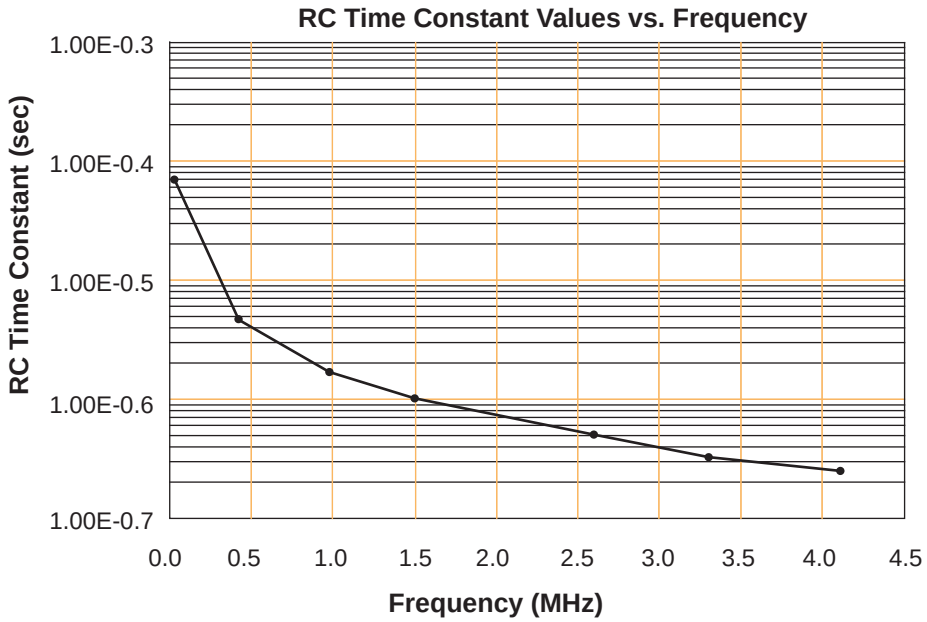


***Figure 8-15* • Main Crystal Oscillator RC Time Constant Versus Frequency**

Low-Power 32 KHz Crystal Oscillator

This oscillator is designed to work with a low-power 32 KHz watch crystal (for example, a CM519) and can be enabled and disabled by setting and clearing bit 0 of the CTRL_STAT_REG in the RTC section. If not being used in the end user application, the LPXIN and LPXOUT pins can be left floating. Additionally, a capacitor is required on both LPXIN and LPXOUT pins to ground, as shown in Figure 8-17 on page 124. The recommended input capacitance is 30 pF.

Battery Backup Circuitry

The 32 KHz low-power crystal oscillator and the real-time counter (RTC) can be powered externally by a CR2032 type of lithium coin cell. Integrated into a SmartFusion device is a battery switch-over circuit (Figure 8-16) which allows the user's application to use main power for powering the oscillator and RTC circuitry when main power is applied instead of battery power, enabling extended battery life and operation. The built-in battery switch-over circuit switches power to the RTC and low power 32 KHz oscillator between VCCLPXTAL and VDDBAT, depending on which voltage is higher (Figure 8-17 on page 124). EQ is used to determine which rail powers the low-power 32 KHz oscillator and the RTC. There is approximately 200 mV of hysteresis built into the switching from one rail to another.

If (VCCLPXTAL < (VDDBAT – 0.4 V)), then

V_{OUT} = VDDBAT

Else

Vout = VCCLPXTAL

EQ 5

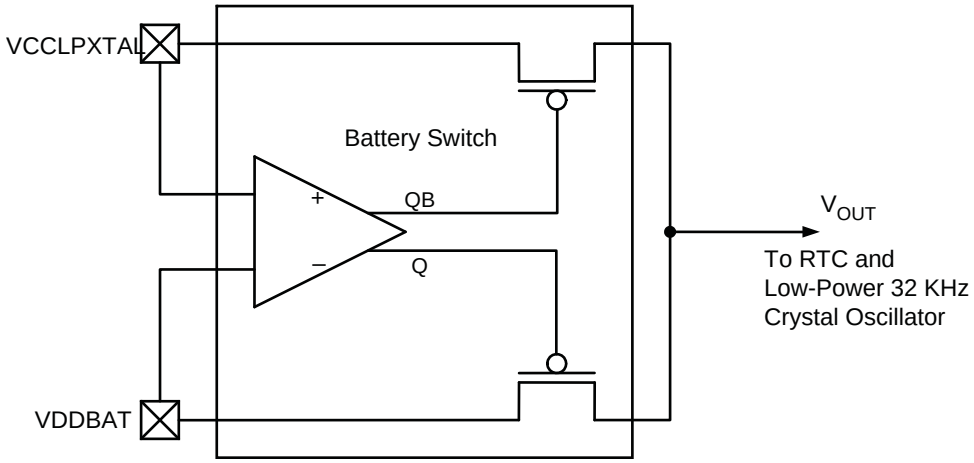


***Figure 8-16* • Battery Switch-Over Circuitry**

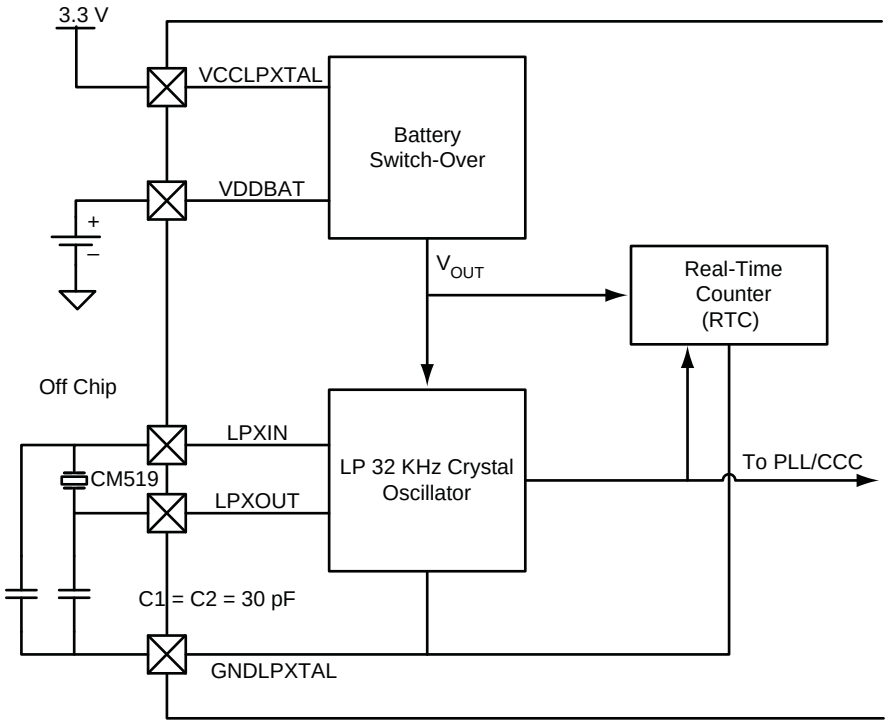


***Figure 8-17* • Low-Power 32 KHz Oscillator with Battery Switch and RTC**

PLL/CCC Register Map

The PLL/CCC control registers are located in the System Registers address space at 0xE0042000 and extend to address 0xE0042FFF in the Cortex-M3 memory map.

***Table 8-5* • PLL/CCC Register Map**

Register Name	Address	R/W	Reset Value	Description
MSS_CLK_CR	0xE0042048	R/W	0x00028A8	Clock Configuration for APB buses
MSS_CCC_DIV_CR	0xE004204C	R/W	0x00030000	Control bits for the CCC dividers
MSS_CCC_MUX_CR	0xE0042050	R/W	0x0D800000	Control bits for the CCC multiplexers
MSS_CCC_PLL_CR	0xE0042054	R/W	0x00000000	Control bits for the PLL
MSS_CCC_DLY_CR	0xE0042058	R/W	0x01FF8000	Control bits for the CCC delay elements
MSS_CCC_SR	0xE004205C	R	0x00000000	PLL Lock indication

Clock Control Register

Table 8-6 • **MSS_CLK_CR**

Bit Number	Name	R/W	Reset Value	Description
31:14	Reserved	R/W	0x00028A8	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	GLBDIVISOR	R/W	b10	Selects the clock ratio between the MSS and FPGA fabric interface. See the "GLBDIVISOR" section.
11:8	RTCIF_ACMDIVISOR	R/W	0b1000	This bit determines the divisor value to be used by the RTCIF block in the generation of ACMCLK from PCLK1. The ACMCLK must have a value of 10 MHz or less. See Table 8-8 on page 127 for allowed values.
7:6	ACLKDIVISOR	R/W	0b10	This bit determines the divisor value to be used to generate clock (ACLK) for APB bus APB_2. The Analog Compute Engine resides on this bus. A multiple of 40 MHz is required for optimal ADC conversion rates. See Table 8-9 on page 127.
5:4	PCLK1DIVISOR	R/W	0b10	This bit determines the divisor value to be used to generate the clock (PCLK1) for APB bus APB_1. See Table 8-10 on page 127.
3:2	PCLK0DIVISOR	R/W	0b10	This bit determines the divisor value to be used to generate the clock (PCLK0) for APB bus APB_0. See Table 8-11 on page 127.
1	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RMIICLKSEL	R/W	0	0 = RMII clock is sourced from an external pad. 1 = RMII clock is sourced from the GLC output of the SmartFusion Clock Control block. The source of the RMII clock for the 10/100 Ethernet MAC is determined by this bit.

GLBDIVISOR

The user has the option of selecting the clock ratio between the MSS and FPGA fabric interface. Valid clock ratios are 1:1, 2:1, and 4:1. If the MSS to FPGA fabric interface clock ratio is selected as 1:1 (whether in synchronous or fabric bypass mode), the AMBA interface logic in the FPGA fabric may use GLA1 instead of GLB as its clock. This leaves GLB free for use and it may be set at any value. However, if the MSS to FPGA fabric interface clock ratio is selected as either 2:1 or 4:1, GLB must be used as the clock source of the AMBA interface logic in the FPGA fabric and the GLB clock must be programmed to have the corresponding ratio to GLA that exists in the MSS to FPGA fabric interface. GLBDIVISOR is programmed by firmware to indicate to the fabric interface logic the actual GLA to GLB ratio.

In other words, if the clock ratio between the MSS and the FPGA fabric is anything other than 1:1, the GLB clock path must be programmed to have the same ratio to GLA that exists in the MSS to FPGA fabric interface. GLBDIVISOR is used internally to generate the appropriate timing signals in the FPGA

fabric interface logic, between the MSS and the FPGA fabric interface, when the ratio between the two is not 1:1.

For example, the MSS clock is set to 100 MHz from RC through GLA and the ratio between the MSS and the FPGA fabric interface is 4:1.

Step 1:

Write to MSS_CCC_MUX_CR to select the RC oscillator and bypass the PLL:

1. Set RXASEL bit to 0x1.
2. Clear DYNASEL to 0x0.
3. Set BYPASSA to a 0x1.

Step 2:

Write to MSS_CCC_MUX_CR to select the GLA path:

Set GLMUXSEL bits to 0x00.

Step 3:

Write to MSS_CCC_MUX_CR to select the GLB path:

1. Set RXBSEL bit to 0x1 (select RC).
2. Clear DYNBSEL to 0x0 (select RC).
3. Set BYPASSB to 0x0 (do not bypass MUX and divider).
4. Set OBMUX to 0x01 (select CLKB).

Write to MSS_CCC_DIV_CR to program the GLB path:

Set OBDIV to 0x03 (divide by 4).

Write to CLK_CONTROL_REG to program the GLA to GLB ratio:

Set GLBDIVISOR to 0x02.

An alternative is Step 3A.

Step 3A:

Write to MSS_CCC_MUX_CR to select the GLB path:

1. Set BYPASSB to 0x0 (do not bypass MUX and divider).
2. Set OBMUX to 0x03 (select GLA output).

Write to MSS_CCC_DIV_CR to program the GLB path:

Set OBDIV to 0x03 (divide by 4, divide GLA by 4 = divide 100 MHz RC / 4).

Write to CLK_CONTROL_REG to program the GLA to GLB ratio:

Set GLBDIVISOR to 0x02.

Table 8-7 • **GLBDIVISOR Bit Definitions**

GLBDIVISOR		
Bit 13	**Bit 12**	**GLB =**
0	0	FCLK
0	1	FCLK / 2
1	0	FCLK / 4
1	1	Reserved

RTCIF_ACMDIVISOR

The allowed values of RTCIF_ACMDIVISOR are shown in Table 8-8.

Table 8-8 • RTCIF_ACMDIVISOR Bit Definitions

RTCIF_ACMDIVISOR				
Bit 11	**Bit 10**	**Bit 9**	**Bit 8**	**ACMCLK =**
0	0	0	1	PCLK1
0	0	1	0	PCLK1 / 2
0	1	0	0	PCLK1 / 4
1	0	0	0	PCLK1 / 8
0	0	0	0	PCLK1 / 16

ACLKDIVISOR

ACLK is derived from FCLK, as shown in Table 8-9.

Table 8-9 • ACKLDIVISOR Bit Definitions

ACLKDIVISOR		
Bit 7	**Bit 6**	**ACLK =**
0	0	FCLK
0	1	FCLK / 2
1	0	FCLK / 4
1	1	Reserved

PCLK1DIVISOR

PCLK1 is derived from FCLK, as shown in Table 8-10.

Table 8-10 • PCLK1DIVISOR Bit Definitions

PCLK1DIVISOR		
Bit 5	**Bit 4**	**PCLK1 =**
0	0	FCLK
0	1	FCLK / 2
1	0	FCLK / 4
1	1	Reserved

PCLK0DIVISOR

PCLK0 is derived from FCLK, as shown in Table 8-11.

Table 8-11 • PCLK0DIVISOR Bit Definitions

PCLK0DIVISOR		
Bit 3	**Bit 2**	**PCLK0 =**
0	0	FCLK
0	1	FCLK / 2
1	0	FCLK / 4
1	1	Reserved

CCC Divider Configuration Register

***Table 8-12* • MSS_CCC_DIV_CR**

Bit Number	Name	R/W	Reset Value	Description
31:23	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22	OCDIVRST	R/W	0	0 = "Don't care." 1 = Reset the counter used to divide the GLC/YC output frequency. The rising edge of this bit will trigger a reset of the GLC/YC output divider. This bit is a "don't care" if the PLL is being used to drive the GLC or YC output.
21	OCDIVHALF	R/W	0	0 = OCDIV defines the GLC/YC output frequency divider. 1 = Use Table 8-13 on page 129 to determine GLC or YC output frequency divider.
20:16	OCDIV	R/W	0b00011	These bits divide the output of the global buffer GLB or YB by the contents of OBDIV + 1. See Table 8-14 on page 130.
15	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	OBDIVRST	R/W	0	0 = "Don't care." 1 = Reset the counter used to divide the GLB/YB output frequency. The rising edge of this bit will trigger a reset of the GLB/YB output divider. This bit is "don't care" if the PLL is being used to drive the GLB or YB output.
13	OBDIVHALF	R/W	0	0 = OBDIV defines the GLB/YB output frequency divider. 1 = Use Table 8-15 on page 130 to determine GLB or YB output frequency divider.
12:8	OBDIV	R/W	0	These bits divide the output of the global buffer GLB or YB by the contents of OBDIV + 1. See Table 8-16 on page 131.
7	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	OADIVRST	R/W	0	0 = "Don't care." 1 = Reset the counter used to divide the GLA output frequency. The rising edge of this bit will trigger a reset of the GLA output divider. This bit is a "don't care" if the PLL is being used to drive the GLA output.

***Table 8-12* • MSS_CCC_DIV_CR (continued)**

5	OADIVHALF	R/W	0	0 = OADIV defines the GLA output frequency divider. 1 = Use Table 8-17 on page 131 to determine GLA output frequency divider.
4:0	OADIV	R/W	0	These bits divide the output of the global buffer GLA by the contents of OADIV + 1. See Table 8-18 on page 132.

OCDIVHALF

This bit, if set to 1, divides the output frequency of the output divider defined by OCDIV by 0.5 when the PLL is bypassed with the 100 MHz RC or 32 KHz low-power oscillator. If OCDIVHALF = 1 and OCDIV = 2, the OCDIV divisor is 3, so 3 ÷ 2 = 1.5. If the GLC/YC input is sourced from the 100 MHz RC, the output of GLC/YC will be 100 ÷ 1.5 = 66.67 MHz. This bit is only valid if the input to the GLC/YC divider is not being sourced by the PLL. Table 8-13 lists the only supported values for OCDIVHALF and OCDIV. Other combinations of OCDIVHALF and OCDIV can lead to unpredictable results.

***Table 8-13* • OCDIVHALF**

OCDIVHALF	OCDIV	Divisor	Input Clock Source	Output Clock
1	0	1	100 MHz RC	100.00
1	2	1.5	100 MHz RC	66.67
1	4	2.5	100 MHz RC	40.00
1	6	3.5	100 MHz RC	28.57
1	8	4.5	100 MHz RC	22.22
1	10	5.5	100 MHz RC	18.18
1	12	6.5	100 MHz RC	15.38
1	14	7.5	100 MHz RC	13.33
1	16	8.5	100 MHz RC	11.76
1	18	9.5	100 MHz RC	10.53
1	20	10.5	100 MHz RC	9.52
1	22	11.5	100 MHz RC	8.70
1	24	12.5	100 MHz RC	8.00
1	26	13.5	100 MHz RC	7.41
1	28	14.5	100 MHz RC	6.90

OCDIV

Table 8-14 gives bit definitions for OCDIV.

***Table 8-14* • OCDIV Bit Definitions**

OCDIV					
Bit 20	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**	**DIVISOR**
0	0	0	0	0	1
0	0	0	0	1	2
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	0	31
1	1	1	1	1	32

OBDIVHALF

This bit, if set to 1, divides the output frequency of the output divider defined by OBDIV by 0.5 when the PLL is bypassed with the 100 MHz RC or 32 KHz low-power oscillator. If OBDIVHALF = 1 and OBDIV = 2, the OBDIV divisor is 3, so 3 ÷ 2 = 1.5. If the GLB/YB input is sourced from the 100 MHz RC, the output of GLB/YB will be 100 ÷ 1.5 = 66.67 MHz. This bit is only valid if the input to the GLB/YB divider is not being sourced by the PLL. Table 8-15 lists the only supported values for OBDIVHALF and OBDIV. Other combinations of OBDIVHALF and OBDIV can lead to unpredictable results.

***Table 8-15* • OBDIVHALF Bit Definitions**

OBDIVHALF	OBDIV	Divisor	Input Clock Source	Output Clock
1	0	1	100 MHz RC	100.00
1	2	1.5	100 MHz RC	66.67
1	4	2.5	100 MHz RC	40.00
1	6	3.5	100 MHz RC	28.57
1	8	4.5	100 MHz RC	22.22
1	10	5.5	100 MHz RC	18.18
1	12	6.5	100 MHz RC	15.38
1	14	7.5	100 MHz RC	13.33
1	16	8.5	100 MHz RC	11.76
1	18	9.5	100 MHz RC	10.53
1	20	10.5	100 MHz RC	9.52
1	22	11.5	100 MHz RC	8.70
1	24	12.5	100 MHz RC	8.00
1	26	13.5	100 MHz RC	7.41
1	28	14.5	100 MHz RC	6.90

OBDIV

Table 8-16 gives bit definitions for OBDIV.

***Table 8-16* • OBDIV Bit Definitions**

OBDIV					
Bit 12	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**DIVISOR**
0	0	0	0	0	1
0	0	0	0	1	2
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	0	31
1	1	1	1	1	32

OADIVHALF

This bit, if set to 1, divides the output frequency of the output divider defined by OADIV by 0.5 when the PLL is bypassed with the 100 MHz RC or 32 KHz low-power oscillator. If OADIVHALF = 1 and OADIV = 2, OADIV Divisor is 3, so 3 ÷ 2 = 1.5. If the PLL is bypassed with the 100 MHz RC, the output of GLA will be 100 ÷ 1.5 = 66.67 MHz. This bit is only valid if the PLL is bypassed and the internal 100 MHz RC oscillator is used. Table 8-17 lists the only supported values for OADIVHALF and OADIV. Other combinations of OADIVHALF and OADIV can lead to unpredictable results.

***Table 8-17* • OADIVHALF Bit Definitions**

OADIVHALF	OADIV	Divisor	Input Clock Source	Output Clock
1	0	1	100 MHz RC	100.00
1	2	1.5	100 MHz RC	66.67
1	4	2.5	100 MHz RC	40.00
1	6	3.5	100 MHz RC	28.57
1	8	4.5	100 MHz RC	22.22
1	10	5.5	100 MHz RC	18.18
1	12	6.5	100 MHz RC	15.38
1	14	7.5	100 MHz RC	13.33
1	16	8.5	100 MHz RC	11.76
1	18	9.5	100 MHz RC	10.53
1	20	10.5	100 MHz RC	9.52
1	22	11.5	100 MHz RC	8.70
1	24	12.5	100 MHz RC	8.00
1	26	13.5	100 MHz RC	7.41
1	28	14.5	100 MHz RC	6.90

OADIV

Table 8-18 gives bit definitions for OADIV.

***Table 8-18* • OADIV Bit Definitions**

OADIV					
Bit 4	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**DIVISOR**
0	0	0	0	0	1
0	0	0	0	1	2
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	0	31
1	1	1	1	1	32

CCC Multiplexer Configuration Register

***Table 8-19* • MSS_CCC_MUX_CR**

Bit Number	Name	R/W	Reset Value	Description
31:30	MAINOSCMODE	R/W	0	Sets the main RC oscillator mode.
29	MAINOSCEN	R/W	0	0 = Main crystal oscillator disabled (default). 1 = Main crystal oscillator enabled.
28	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27:26	GLMUXCFG	R/W	0b10	With GLMUXSEL, configures the glitchless multiplexer. See Table 8-21 on page 135.
25:24	GLMUXSEL	R/W	0	With GLMUXCFG, configures the glitchless multiplexer. See Table 8-21 on page 135.
23	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22	BYPASSC	R/W	0	0 = GLC = Output of PLL divider *w*. 1 = GLC = Global MUX C Path.
21:19	OCMUX	R/W	0	Clock path C output multiplexer. See Table 8-22 on page 135.
18	DYNCSEL	R/W	0	With RXCSEL and STATCSEL, selects the input clock source for clock path C. See Table 8-23 on page 136.
17	RXCSEL	R/W	0	With DYNCSEL and STATCSEL, selects the input clock source for clock path C. See Table 8-23 on page 136.
16	STATCSEL	R/W	0	With DYNCSEL and RXCSEL, selects the input clock source for clock path C. See Table 8-23 on page 136.
15	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	BYPASSB	R/W	0	0 = GLB = Output of PLL divider *v*. 1 = GLB = Global MUX B Path.
13:11	OBMUX	R/W	0	Clock Path B output multiplexer. See Table 8-24 on page 136.
10	DYNBSEL	R/W	0	With RXBSEL and STATBSEL, selects the input clock source for clock path B. See Table 8-25 on page 136.
9	RXBSEL	R/W	0	With DYNBSEL and STABSEL, configures the input clock source for clock path B. See Table 8-25 on page 136.
8	STATBSEL	R/W	0	With DYNBSEL and RXBSEL, selects the input clock source for clock path B. See Table 8-25 on page 136.

***Table 8-19* • MSS_CCC_MUX_CR (continued)**

7	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BYPASSA	R/W	0	0 = GLA = Output of PLL divider *u*. 1 = GLA = Global MUX A path.
5:3	OAMUX	R/W	0	Clock Path A output multiplexer. See Table 8-26 on page 137.
2	DYNASEL	R/W	0	With RXASEL and STATASEL, selects the input clock source for clock path A. See Table 8-27 on page 137.
1	RXASEL	R/W	0	With DYNASEL and STATASEL, selects the input clock source for clock path A. See Table 8-27 on page 137.
0	STATASEL	R/W	0	With DYNASEL and RXASEL, selects the input clock source for clock path A. See Table 8-27 on page 137.

MAINOSCMODE

Table 8-20 gives bit definitions for MAINOSCMODE.

***Table 8-20* • MAINOSCMODE Bit Definitions**

MAINOSCMODE			
Bit 31	**Bit 30**	**Clock Mode**	**Function**
0	0	RC network	RC oscillation mode. Connects the RC network to the MAINXIN pad. The MAINXOUT pad should be disconnected.
0	1	Low gain	0.32 to 0.20 MHz low-power/-frequency mode. Oscillator consumes the least current of the three crystal modes.
1	0	Medium gain	0.20 to 2.0 MHz standard crystal/resonator frequency
1	1	High gain	2.0 to 20.0 MHz high-frequency mode. Oscillator consumes the most current of the three modes.

GLMUXCFG[27:26] and GLMUXSEL[25:24]

Table 8-21 gives bit definitions for GLMUXCFG and GLMUXSEL.

***Table 8-21* • GLMUXCFG and GLMUXSEL Bit Definitions**

GLMUXCFG		GLMUXSEL		
Bit 27	**Bit 26**	**Bit 25**	**Bit 24**	**Selected Input Signal**
0	0	X	0	GLA
		X	1	GLC
0	1	X	0	GLA
		X	1	GLINT
1	0	X	0	GLC
		X	1	GLINT
1	1	0	0	GLA
		0	1	GLC
		1	0	GLINT
		1	1	GND

OCMUX

Table 8-22 gives bit definitions for OCMUX.

***Table 8-22* • OCMUX Bit Definitions**

OCMUX			
Bit 21	**Bit 20**	**Bit 19**	**Function**
0	0	0	MUX and PLL are bypassed
0	0	1	CLKC
0	1	0	PLL VCO 0° feedback delay line output (from FBDLY)
0	1	1	GLA clock source
1	0	0	PLL VCO 0° phase shift
1	0	1	PLL VCO 90° phase shift
1	1	0	PLL VCO 180° phase shift
1	1	1	PLL VCO 270° phase shift

DYNCSEL, RXCSEL, and STATCSEL

Table 8-23 gives bit definitions for DYNCSEL, RXCSEL, and STATCSEL.

***Table 8-23* • DYNCSEL, RXCSEL, and STATCSEL Bit Definitions**

RXBSEL	DYNBSEL	STATBSEL	CLKC
0	0	0	CUIN
0	1	0	CUIP
0	0	1	CDIP
0	1	1	GLCINT
1	0	X	RC oscillator
1	1	X	32 KHz oscillator

OBMUX

Table 8-24 gives bit definitions for OBMUX.

***Table 8-24* • OBMUX Bit Definitions**

OBMUX			Function
Bit 13	Bit 12	Bit 11	
0	0	0	MUX and PLL are bypassed
0	0	1	CLKB
0	1	0	PLL VCO 0° feedback delay line output (from FBDLY)
0	1	1	GLA clock source
1	0	0	PLL VCO 0° phase shift
1	0	1	PLL VCO 90° phase shift
1	1	0	PLL VCO 180° phase shift
1	1	1	PLL VCO 270° phase shift

DYNBSEL, RXBSEL, and STATBSEL

Table 8-25 gives bit definitions for RXBSEL, DYNBSEL, and STATBSEL.

***Table 8-25* • DYNBSEL, RXBSEL, and STATBSEL Bit Definitions**

RXBSEL	DYNBSEL	STATBSEL	CLKB
0	0	0	BUIN
0	1	0	BUIP
0	0	1	BDIP
0	1	1	GLBINT
1	0	X	RC oscillator
1	1	X	Main oscillator

OAMUX

Table 8-26 gives bit definitions for OAMUX.

***Table 8-26* • OAMUX Bit Definitions**

OAMUX			Function
Bit 5	**Bit 4**	**Bit 3**	
0	0	0	MUX and PLL are bypassed.
0	0	1	CLKA
0	1	0	PLL VCO 0° feedback delay line output (from FBDLY)
0	1	1	Not available
1	0	0	PLL VCO 0° phase shift
1	0	1	PLL VCO 90° phase shift
1	1	0	PLL VCO 180° phase shift
1	1	1	PLL VCO 270° phase shift

DYNASEL, RXASEL, and STATASEL

Table 8-27 gives bit definitions for DYNASEL, RXASEL, and STATASEL.

***Table 8-27* • DYNASEL, RXASEL, and STATSEL Bit Definitions**

RXASEL	DYNASEL	STATASEL	CLKA
0	0	0	AUIN
0	1	0	AUIP
0	0	1	ADIP
0	1	1	GLAINT
1	0	X	RC oscillator
1	1	X	Main oscillator

CCC PLL Configuration Register

***Table 8-28* • MSS_CCC_PLL_CR**

Bit Number	Name	R/W	Reset Value	Description
31	PLLEN	R/W	0	0 = PLL in power-down mode 1 = PLL enabled
30:25	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24:23	VCOSEL[2:1]	R/W	0	Specifies the PLL lock acquisition time and tracking jitter. See Table 8-29.
22	VCOSEL[0]	R/W	0	0 = Fast PLL lock acquisition time with high tracking jitter. 1 = Slow PLL lock acquisition time with low tracking jitter.
21	XDLYSEL	R/W	0	Setting this bit to a 1 adds an additional 2 ns (typical) delay to the output of the PLL feedback selected by the FBSEL control bits.
20:16	FBDLY	R/W	0	FBDLY sets the delay from the PLL VCO 0° phase shift output to the feedback input of the PLL. A value of 0 has a typical delay of 535 ps, every time FBDLY is incremented; 200 ps is added to the base delay. See Table 8-30 on page 139.
15:14	FBSEL	R/W	0	Selects the multiplexer input. See Table 8-31 on page 139.
13:7	FBDIV	R/W	0	FBDIV defines the feedback clock divider /m value. Divides the PLL feedback clock frequency by the value stored in FBDIV[6:0] + 1.
6:0	FINDIV	R/W	0	FINDIV defines the input clock divider /n value. Divides the input clock frequency to the PLL by the value stored in FINDIV[6:0] + 1.

VCOSEL[2:1]

Table 8-29 gives bit definitions for VCOSEL.

***Table 8-29* • VCOSEL[2:1] Bit Definitions**

VCOSEL[2:1]	VCO Output Range in MHz
00	22 – 43.75
01	43.75 – 87.5
10	87.5 – 175
11	175 – 350

FBDLY

Table 8-30 gives bit definitions for FBDLY.

***Table 8-30* • FBDLY Bit Definitions**

FBDLY					
Bit 20	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**	**Delay Value**
0	0	0	0	0	535 ps typical
0	0	0	0	1	735 ps typical
0	0	0	1	0	935 ps typical
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	1	5.56 ns typical

FBSEL

Table 8-31 gives bit definitions for FBSEL.

***Table 8-31* • FBSEL Bit Definitions**

FBSEL		
Bit 15	**Bit 14**	**Multiplexer Input Selected**
0	0	GLBINT from FPGA fabric
0	1	PLL VCO 0 degree phase shift
1	0	PLL delayed (by FBDLY) VCO 0 degree phase shift
1	1	BUIP direct input clock

CCC Delay Configuration Register

***Table 8-32* • MSS_CCC_DLY_CR**

Bit Number	Name	R/W	Reset Value	Description
31:25	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24:20	DLYA1	R/W	0b11111	DLYA1 sets the delay from the output of the glitchless MUX to the FPGA fabric. A value of 0 has a typical delay of 535 ps. Every time DLYA1 is incremented, 200 ps is added to the base delay. The default value of DLYA1 is 0x1f. See Table 8-33.
19:15	DLYA0	R/W	0b11111	DLYA0 sets the delay from the output of the glitchless MUX to the microcontroller subsystem. A value of 0 has a typical delay of 735 ps, every time DLYA0 is incremented; 200ps is added to the base delay. The default value of DLYA0 is 0x1f. See Table 8-34 on page 141.
14:10	DLYC	R/W	0	Same bit definitions as DLYA. See Table 8-35 on page 141.
9:5	DLYB	R/W	0	Same bit definitions as DLYA. See Table 8-35 on page 141.
4:0	DLYA	R/W	0	DLYA sets the delay for the Global MUX A path output prior to the glitchless MUX. A value of 0 has a typical delay of 535 ps, then every time DLYA is incremented; 200 ps is added to the previous delay value. See Table 8-35 on page 141.

DLYA1

Table 8-33 gives bit definitions for DLYA1.

***Table 8-33* • DYLA1 Bit Definitions**

DLYA1					Delay Value
Bit 24	Bit 23	Bit 22	Bit 21	Bit 20	
0	0	0	0	0	735 ps typical
0	0	0	0	1	935 ps typical
0	0	0	1	0	1135 ps typical
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	1	5.56 ns typical

DLYA0

Table 8-34 gives bit definitions for DYLA0.

Table 8-34 • DYLA0 Bit Definitions

DLYA0					
Bit 19	Bit 18	Bit 17	Bit 16	Bit 15	Delay Value
0	0	0	0	0	735 ps typical
0	0	0	0	1	935 ps typical
0	0	0	1	0	1135 ps typical
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	1	5.56 ns typical

DLYA

Table 8-35 gives bit definitions for DLYA.

Table 8-35 • DYLA Bit Definitions

DLYA					
Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Delay Value
0	0	0	0	0	535 ps typical
0	0	0	0	1	735 ps typical
0	0	0	1	0	935 ps typical
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	1	1	1	5.56 ns typical

CCC Status Register

Table 8-36 • MSS_CCC_SR

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PLL_LOCK_SYNC	R	0	This bit indicates whether the SmartFusion PLL is in a locked condition. This bit must be asserted before firmware switches the MSS clock source to the PLL. 0 = PLL is not locked (default). 1 = PLL is locked.

9 – Reset Controller

The reset controller manages the SmartFusion on-chip reset resources. On power-up, the signal PORESET_N is used to bring the SmartFusion device to a known power-up state. PORESET_N is sourced by the voltage regulator and power supply monitor (VR/PSM) block. A block diagram is shown in Figure 9-1. There are two external pads that interface to the reset controller: MSS_RESET_N and TRSTB. MSS_RESET_N can be used as an external reset and can also be used as a system level reset under control of the Cortex-M3 processor. TRSTB is used to reset the SWJ-DP logic within the Cortex-M3 processor and to reset the main JTAG TAP controller. All other inputs to and outputs from the reset controller originate on-chip. Note that the SOFT_RESETS signals sourced from Figure 9-1 place the respective peripheral in a low-power state. For example, if the user asserts I2C_0_SR (logic 1) in the SOFT_RST_CR register, all flip-flops in that block are automatically clock gated. MSS_RESET_REQ from the Cortex-M3 microcontroller is controlled by the SYSRESETREQ bit in the Application Interrupt and Reset Control register located at address 0XE000ED0C. For more information, refer to the *Cortex-M3 Technical Reference Manual* from ARM.

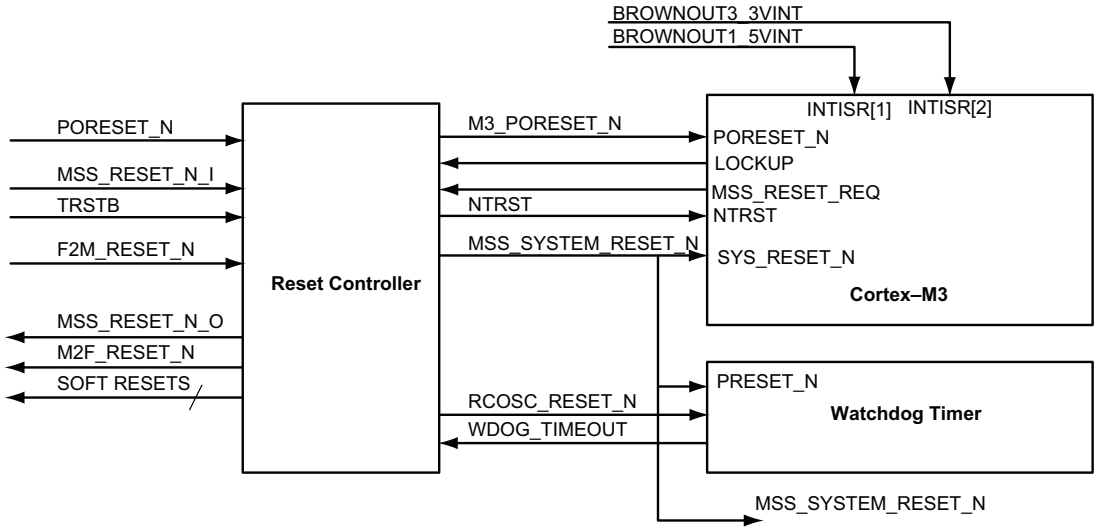


Figure 9-1 • **Reset Controller Block Diagram**

Functional Description

PORESET_N is a hard (cold) reset signal. Its assertion causes everything in the MSS except for the SWJ-DP in the Cortex-M3 microcontroller to be reset. All the other functional reset sources (those other than NTRST) are soft (warm) resets. The signals BROWNOUT3_3VINT and BROWNOUT1_5VINT are sourced from the VR/PSM block and provide interrupt capability when these supplies fall below 2.5 V and 1.3 V, respectively. These signals are also readable as status bits from the MSS_SR, located at address 0xE004201C. Note that INTISR[1] and INTISR[2] must be enabled after the analog block is turned on. The analog block can be turned on by setting the ABPOWERON bit in the ANA_COMM_CTRL register to a 1. The ANA_COMM_CTRL register is located at address 0x4002000C in the memory map.

The reset controller outputs are listed and described in Table 9-1.

***Table 9-1* • Reset Controller Outputs**

Signal	Description
M3_PORESET_N	This is a synchronized version of PORESET_N from the VR/PSM block. This signal resets all logic within the Cortex-M3 microcontroller, with the exception of the SWJ-DP block.
MSS_SYSTEM_RESET_N	This drives the SYS_RESET_N input to the Cortex-M3 microcontroller and is also the reset signal for the entire MSS. When SYS_RESET_N asserts low, the entire Cortex-M3 microcontroller is reset except for the debug logic that exists in the following blocks: • Nested vectored interrupt controller (NVIC) • Flash patch and breakpoint (FPB) • Data watchpoint and trace (DWT) • Instrumentation trace macrocell (ITM) • AHB-AP MSS_SYSTEM_RESET_N asserts asynchronously and negates synchronously to FCLK. This guarantees that it is synchronous to rising edges of FCLK, ACLK, PCLK0, and PCLK1. MSS_RESET_N is asserted if any of the following conditions is true: • PORESET_N asserted by the power supply monitor (PSM) • MSS_RESET_REQ asserted by Cortex-M3 microcontroller • F2M_RESET_N asserted from FPGA fabric, if F2MRESETENABLE asserted in SOFT_RST_CR • WDOG_TIMEOUT asserted by watchdog • LOCKUP asserted by Cortex-M3 microcontroller • MSS_RESET_N_I asserted (during allowed window, controlled by reset controller state machine)
NTRST	This drives the NTRST (debug reset) input of the Cortex-M3 microcontroller and is used to reset the SWJ-DP sub-block within the Cortex-M3 microcontroller.
M2F_RESET_N*	This reset signal is fed to the FPGA fabric. M2F_RESET_N asserts asynchronously and negates synchronously to FCLK. This guarantees that it is synchronous to rising edges of FCLK, ACLK, PCLK0, and PCLK1. M2F_RESET_N is asserted if any of the following conditions is true: • PORESET_N asserted by analog block • MSS_RESET_REQ asserted by Cortex-M3 microcontroller • WDOG_TIMEOUT asserted by watchdog • LOCKUP asserted by Cortex-M3 microcontroller • MSS_RESET_N_I asserted (during allowed window, controlled by reset controller state machine).

*Note: *M2F_RESET_N is asserted by F2M_RESET_N. Care must be taken by the user NOT to connect M2F_RESET_N to the reset input of the fabric master which, in the user's design, asserts F2M_RESET_N to reset the MSS.*

***Table 9-1* • Reset Controller Outputs (continued)**

Signal	Description
RCOSC_RESET_N	Asserts asynchronously and negates synchronously to RCOSCCLK. This signal is used to reset parts of the Watchdog block which are clocked by the RC oscillator. RCOSC_RESET_N is asserted if any of the following conditions is true: • PORESET_N asserted by analog block • MSS_RESET_REQ asserted by Cortex-M3 microcontroller • F2M_RESET_N asserted from FPGA fabric, if F2MRESETENABLE asserted in SOFT_RST_CR • LOCKUP asserted by Cortex-M3 microcontroller • MSS_RESET_N_I asserted (during allowed window, controlled by Reset Controller State Machine)
MSS_RESET_N_O	This signal is used to drive the external output enable of the I/O Buffer MSS_RESET_N, as shown in Figure 9-2 on page 146. When asserted, it causes a zero to be driven onto the MSS_RESET_N pad. This signal is asserted by the reset controller during PORESET_N assertion. When MSS_RESET_N negates, MSS_RESET_N_O remains asserted until firmware clears the EXT_SR bit in SOFT_RST_CR. This allows a SmartFusion cSoC to control the behavior of the system level reset, if the user so desires. From this point on, this signal is driven whenever one of the following reset sources asserts: • PORESET_N asserted by analog block • MSS_RESET_REQ asserted by Cortex-M3 microcontroller • F2M_RESET_N asserted from FPGA fabric, if F2MRESETENABLE asserted in SOFT_RST_CR • WDOG_TIMEOUT asserted by watchdog • LOCKUP asserted by Cortex-M3 microcontroller • MSS_RESET_N_I asserted MSS_RESET_N_O is asserted asynchronously and negates synchronously to FCLK. After a period of time, defined by user firmware, the PADRESETENABLE bit in the SOFT_RST_CR can be set. Once this bit is set, the reset controller moves to a state where it monitors the state of the MSS_RESET_N_I signal, from the MSS_RESET_N pad. From this point on, any external assertion of MSS_RESET_N_I also causes MSS_RESET_N_O to assert (as the pad is open-drain, it is okay for two sources to be driving MSS_RESET_N low together). The external signal will remain low until both sources stop driving it. See Figure 9-3 on page 147.
SOFT_RESETS	Soft resets are described in Table 9-4 on page 149.

*Note: *M2F_RESET_N is asserted by F2M_RESET_N. Care must be taken by the user NOT to connect M2F_RESET_N to the reset input of the fabric master which, in the user's design, asserts F2M_RESET_N to reset the MSS.*

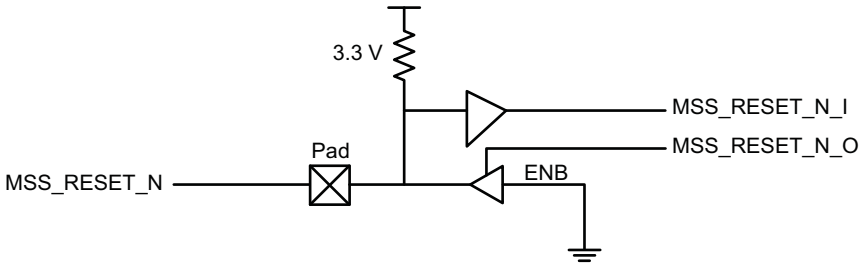


Figure 9-2 • **MSS_RESET_N Output Buffer Configuration**

The MSS_RESET_N Pin

The configuration of the MSS_RESET_N pin is controlled by the SYSREG bits BTWEST[1:0].

The MSS_RESET_N pin can be configured as any single-ended I/O (bidirectional). The output buffer is set to high slew and the drive strength depends on the I/O standard. It can be one of the following, based on user configuration:

- LVTTL – 8 mA
- LVCMOS 2.5 – 8 mA
- LVCMOS 1.8 – 4 mA
- LVCMOS 1.5 – 2 mA

Reset Controller State Machine

The reset controller state machine (Figure 9-3 on page 147) guarantees that the resets it issues are asserted for eight FCLK periods. The reset which asynchronously resets this state machine is called ASYNCRESETMAIN, which is generated if any of the following reset conditions occur:

- PORESET_N asserted
- MSS_RESET_REQ asserted (by firmware)
- LOCKUP asserted (by Cortex-M3 microcontroller)
- F2M_RESET_N asserted from FPGA fabric, provided F2MRESETENABLE is asserted in SOFT_RST_CR
- WDOG_TIMEOUT asserted

The state machine ensures that the MSS_RESET_N pad may be dual-purpose—driven out by the MSS and driven into the MSS from off-chip. The windows in which each occur are controlled by the reset controller state machine, under control of firmware writes to SOFT_RST_CR bits EXT_SR and PADRESETENABLE.

Once the state machine comes out of reset, it asserts resets for eight FCLK periods and then releases them all except for MSS_RESET_N_OE, which it continues to assert. When firmware turns off EXT_SR, the state machine negates MSS_RESET_N_OE and moves to the next state, where it waits for firmware to set PADRESETENABLE. Firmware can tune this delay by moving to the next state, to allow for any de-bouncing of MSS_RESET_N to occur. Finally, when firmware sets PADRESETENABLE, the state machine moves on to the IDLE state, where it now monitors (for the first time) the state of MSS_RESET_N_I. If it sees MSS_RESET_N_I assert, it goes to the PRESTART state, where the full reset sequencing starts again. The only time that the assertion of MSS_RESET_N by an off-chip source causes the MSS resets to assert is during this IDLE state. This avoids any asynchronous loops in situations where the reset controller state machine itself is causing the assertion of MSS_RESET_N.

The reset controller state machine is shown in Figure 9-3.

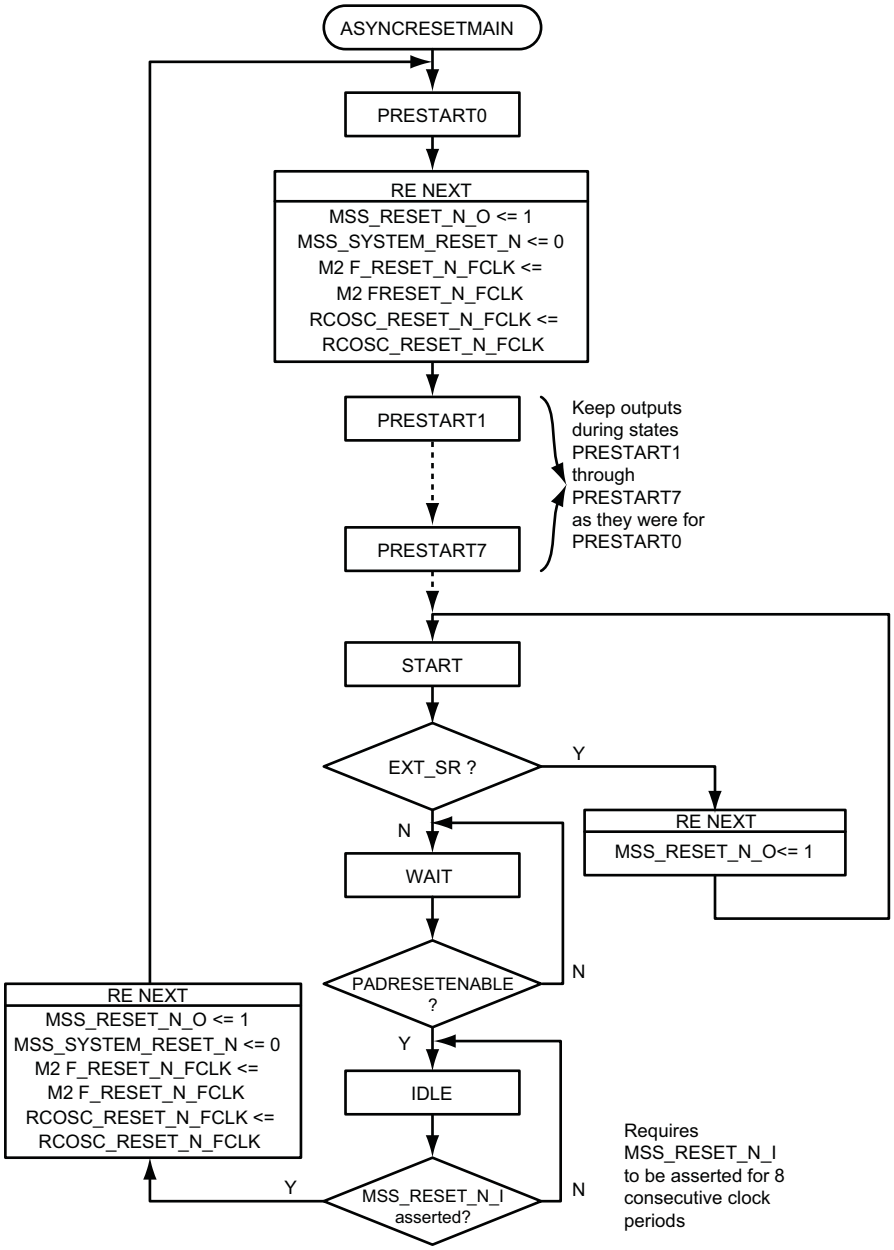


Figure 9-3 • **Reset Controller State Machine**

A requirement on the off-chip source is that it must assert MSS_RESET_N for at least eight FCLK clock periods and during the correct time window to guarantee correct operation. It must not assert MSS_RESET_N until it sees MSS_RESET_N negated. Assertion outside of the correct window could be masked (not impacting the logic). When MSS_RESET_N is asserted by the off-chip source during the correct window, the MSS_RESET_N signal remains asserted even after the off-chip source stops driving it low. This is because the MSS has taken over the pad via the EXT_SR bit in SOFT_RST_CR.

Analog Reset

An MSS reset of the analog block will force it back to its default state. The default configuration for each analog I/O depends on its type—temperature monitor (TM), current monitor (CM), direct analog-to-digital converter (ADC). Refer to the *SmartFusion Programmable Analog User's Guide* for details.

When analog inputs are reset to their default configuration, they must be redefined in the MSS system boot code. These inputs will propagate to the fabric when the analog block reset has been completed and system boot code has been executed.

If analog inputs are configured as digital LVTTL inputs, there is no impact upon system boot code execution because configuration is done in the generated netlist and stored in flash bits.

Reset Controller Register Map

Table 9-4 gives the register map for the reset controller and other registers mentioned in this document.

***Table 9-2* • Reset Controller Memory Map**

Register Name	Address	R/W	Reset Value	Description
ANA_COMM_CTRL	0x4002000C	R/W	1	The analog block can be turned on by setting the ABPOWERON bit in the ANA_COMM_CTRL register to a 1.
MSS_SR	0xE004201C	R/W	0	Signals BROWNOUT3_3VINT and BROWNOUT1_5VINT are readable as status bits from MSS_SR.
SOFT_RST_CR	0xE0042030	R/W	0x00003FFF8	Controls soft resets.

ANA_COMM_CTRL

***Table 9-3* • ANA_COMM_CTRL**

Bit Number	Name	R/W	Reset Value	Function
7:5	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	ACB_RESETN	W	0	1 – Reset analog control block 0 – No effect. This bit self clears after one ACLK cycle.
3	ABPOWERON	R/W	0	1 – Power on the entire analog block 0 – Power-down the entire analog block
2	ADCSPWRDWN	R/W	0	1 – Power-down ADC0, ADC1, and ADC2, 0 – Power-up ADC0, ADC1, and ADC2. This bit is logically ORed with the individual ADC PWRDWN bits found in the ADCx_MISC_CTRL registers.
1	ADCSRESET	R/W	0	1 – Reset ADC0, ADC1, and ADC2 0 – No effect. This bit self clears after one ACLK cycle. Note that this bit is logically ORed with bit 4 (ADCRESET) of the ADCx_MISC_CTRL registers to allow common control or individual reset control of the ADC.
0	VAREFSEL	R/W	1	1 – Select external reference voltage for ADC0, ADC1, and ADC2 0 – Select internal reference voltage for ADC0, ADC1, and ADC2

Reset Controller Register Bit Definitions

The bit definitions for SOFT_RST_CR are given in Table 9-4.

***Table 9-4* • SOFT_RST_CR**

Bit Number	Name	R/W	Reset Value	Function
31:20	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	PADRESETENABLE	R/W	0	0 = No effect. 1 = Allow reset controller to monitor MSS_RESET_N for an external reset command. This bit must be set to 1 if the user wants to allow an external reset from the MSS_RESET_N pad. Users can de-bounce the MSS_RESET_N pin by delaying the assertion of this bit from the assertion of EXT_SR.
18	F2MRESETENABLE	R/W	0	0 = F2M_RESET_N cannot reset MSS. 1 = Allow F2M_RESET_N to reset the MSS via the reset controller.
17	FPGA_SR	R/W	1	0 = Allow the M2F_RESET_N signal to be released. 1 = M2F_RESET_N is asserted. M2F_RESET_N can be used by user logic as a reset input controlled by the Cortex-M3 microcontroller. This bit can also be used as a general purpose output to the FPGA Fabric controlled by the M3.
16	EXT_SR	R/W	1	0 = Release MSS_RESET_N from reset. 1 = Keep MSS_RESET_N asserted (low). At power-up, this signal is asserted 1. This causes the MSS_RESET_N signal to remain asserted after power up. The user must set this bit to 0 to allow the MSS_RESET_N pad to deassert and release external logic from its reset state.
15	IAP_SR	R/W	1	0 = Release the IAP controller from reset. 1 = Keep the IAP controller in reset.
14	GPIO_SR	R/W	1	0 = Release the GPIOs from reset. 1 = Keep the GPIOs in reset.
13	ACE_SR	R/W	1	0 = Release the ACE from reset. 1= Keep the ACE in reset.
12	I2C_1_SR	R/W	1	0 = Release I2C_1 from reset. 1 = Keep I2C_1 in reset.
11	I2C_0_SR	R/W	1	0 = Release I2C_0 from reset. 1 = Keep I2C_0 in reset.
10	SPI_1_SR	R/W	1	0 = Release SPI_1 from reset. 1 = Keep SPI_1 in reset.
9	SPI_0_SR	R/W	1	0 = Release SPI_0 from reset. 1 = Keep SPI_0 in reset.
8	UART_1_SR	R/W	1	0 = Release UART_1 from reset. 1 = Keep UART_1 in reset.
7	UART_0_SR	R/W	1	0 = Release UART_0 from reset. 1 = Keep UART_0 in reset.

***Table 9-4* • SOFT_RST_CR (continued)**

Bit Number	Name	R/W	Reset Value	Function
6	TIMER_SR	R/W	1	0 = Release the system timer from reset. 1 = Keep the system timer in reset.
5	PDMA_SR	R/W	1	0 = Release the PDMA from reset. 1 = Keep the PDMA in reset.
4	MAC_SR	R/W	1	0 = Release the Ethernet MAC from reset. 1 = Keep the Ethernet MAC in reset.
3	EMC_SR	R/W	1	0 = Release the external memory controller from reset. 1 = Keep the external memory controller in reset.
2	ESRAM_1_SR	R/W	0	0 = Release the ESRAM_1 memory controller from reset. 1 = Keep the ESRAM_1 memory controller in reset.
1	ESRAM_0_SR	R/W	0	0 = Release the ESRAM_0 memory controller from reset. 1 = Keep the ESRAM_0 memory controller in reset.
0	ENVM_SR	R/W	0	0 = Release the ENVM memory controller from reset. 1 = Keep the ENVM memory controller in reset.

10 – Voltage Regulator (VR), Power Supply Monitor (PSM), and Power Modes

The VR and PSM provide the user with various ways to define how SmartFusion devices power up and power down. This section describes the functionality of these blocks and how they can be configured to achieve various power profiles. A high-level block diagram is shown in Figure 10-1.

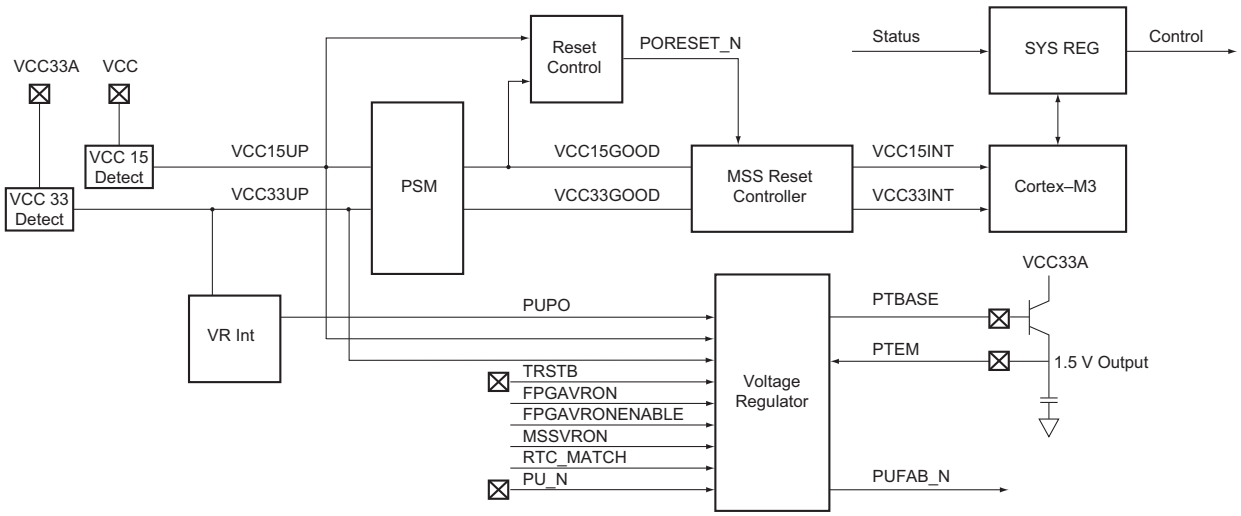


***Figure 10-1* • VR and PSM Block Diagram**

1.5 V Voltage Detector (VCC15UP)

This block (see Figure 10-4 on page 154) has a single input (VCC) and a single active high output (VCC15UP). When the VCC supply is below a threshold (approximately 0.8 V, depending on process and temperature variables), the output is low. When VCC is above this threshold the output is high. A small amount of hysteresis is included in the voltage detector to reduce the possibility of oscillation. VCC15UP is routed to the power supply monitor where it is compared against the precision band gap to create the VCC15GOOD signal. The VCC15GOOD signal becomes high at approximately 1.3 V. During a brownout condition, VCC15GOOD will deassert around 1.3 V. When the VCC15 supply drops below the specified 1.425 V minimum, the SmartFusion MSS will continue to work up until the time VCC15GOOD deasserts. When VCC15GOOD is approximately 1.3 V, the eNVM will stop functioning and assert a busy signal to hold off the AHB bus matrix. If VCC15GOOD reasserts, the eNVM will release the AHB bus matrix and normal operation will resume. The user can then trap this event with INTISR[1]: BROWNOUT1_5V_IRQ. This interrupt service routine must be executing out of internal eSRAM. Alternatively the user can monitor the VCC supply with the ADC. When it starts to drop below 1.425 V (1.5 V – (1.5 × 5%)), issue a soft reset.

3.3 V Voltage Detector (VCC33UP)

This block (see Figure 10-4 on page 154) has a single input (VCC33A) and a single active High output (VCC33UP). When the VCC33A supply is below threshold (2 to 2.5 V, depending on process and temperature variables), the output is Low (a reliable Low if VCC33A is above approximately 0.5 V). When VCC33A is above this threshold the output is High. A small amount of hysteresis is included in the voltage detector to reduce the possibility of oscillation. VCC33UP is routed to the power supply monitor (PSM), where it is compared against the precision band gap to create the VCC33GOOD signal. The VCC33GOOD signal becomes High at approximately 2.5 V. If the VCC33GOOD signal deasserts, the functionality of the analog front-end is in question. To detect a brownout condition on the VCC33A supply, the user should enable the INT[ISR2] : BROWNOUT3_3V_IRQ and place the interrupt service routine in eSRAM. Use the ADC to monitor the VCC33A supply voltage; when it drops below 2.95 V, issue a soft reset to the reset controller.

VR Init

During initial power-up, VCC33 is below the VCC33UP detection level. During this time current sources supply current to the two flash bits, as shown in Figure 10-2. Normally one of these flash bits will be programmed and the other will be erased. The resulting voltage difference is applied to a comparator whose output drives the power-up power-on signal (PUPO). When VCC33UP goes high, the PUPO logical output is held latched to its evaluated state and the current sources disabled (to conserve power). PUPO will not be reevaluated again until VCC33UP is again in a low state. The PUPO signal determines whether or not the VR is enabled when 3.3 V is first applied to the SmartFusion device. The configuration of the flash bits is done through the MSS configurator.

Note: **Changing the programmed state of the PUPO flash bits results in a change in behavior only if VCC33 and VBAT are both off (removed) following programming**.

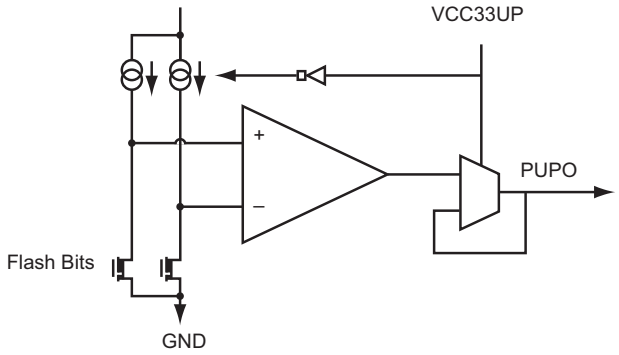


***Figure 10-2* • VR Init Block Diagram**

1.5 V Voltage Regulator

The VR consists of a high gain amplifier, resistor voltage divider sense circuit, and an external pass transistor. The regulator is powered from the 3.3 V supply and produces a 1.5 V regulated output which the user can connect to the VCC pin (which powers the FPGA and the MSS) on their PCBs. If an existing 1.5 V rail is already available in the system, the user can use it instead of the functionality provided by the internal VR. The output of the VR is the PTBASE pin and supplies the drive signal for the external NPN pass transistor. This output can source up to 20 mA in to the transistor's base. The output current of the circuit depends on the current gain of the NPN pass transistor connected externally. The PTEM pin is the sense input for the regulator and consists of a resistive voltage divider between the sense input and GND. Microsemi recommends using PN2222A or 2N2222A transistors with the VR.

The logic diagram for the VR is shown in Figure 10-3 on page 153. The VR can be enabled from several sources: the PU_N pin, RTM_MATCH signal from the RTC block, TRSTB = 1, or triggered by the PUPO signal from the VR Init block. Once triggered, the VR will remain on because of the latching functions of

RS flip-flops Q0 or Q1. Only the MSS or the FPGA fabric can reset these flip-flops and turn off the VR. The VR may also be turned off if VCC33 supply falls below the VCC33UP threshold and a reset occurs.

In summary, the VR can be turned on by the following sources:

- On power-up by the PUPO signal, which is defined by the MSS configurator
- By JTAG being active TRSTB = 1
- A low signal on the PU_N pin
- A high on RTC_MATCH

Flip-flop Q2 turns on the VR only if the VCC33UP signal is in transition from logic 0 to logic 1 and PUPO has been configured, by the MSS configurator, to turn on the VR on power-up. Once the FPGAGOOD signal is established, this particular VR enable mechanism is no longer active.

The VR may be powered off under firmware control by the ARM Cortex-M3 processor using the MSSVRON bit in the VRPSM_CR, located at address 0xE0042064, or the FPGAVRON signal sourced from the FPGA fabric. The FPGAVRON signal from the FPGA fabric is qualified by the FPGAVRONENABLE bit (must be equal to 1) in the VRPSM_CR. In either case, a low-to-high-to-low transition commands the VR to turn off. Note that the RTC_MATCH signal must be low in order to turn off the VR.

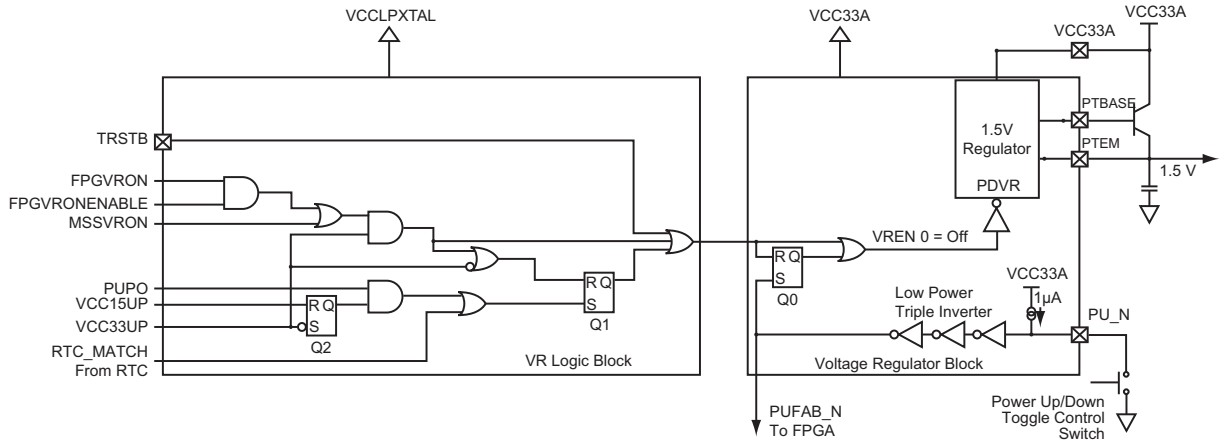


***Figure 10-3* • VR Block Diagram**

Power Supply Monitor (PSM)

The power supply monitor provides reference voltages for the analog-to-digital converter (ADC) and the eNVM. The PSM also provides separate logic outputs to indicate that certain voltage sources are valid. These sources include the band gap reference, the 1.5 V supply from the VR, and the 3.3 V supplies (VCC33A and VCC33AP).

PORESET_N is the hard power-on reset to the SmartFusion device. This signal is held low by VCC15UP until VCC15GOOD passes 1.3 V. PORESET_N is fed into the reset controller (see the "Reset Controller" section on page 143), where it is distributed to various parts of a SmartFusion device. The signals VCC15GOOD and VCC33GOOD can be used as brownout signals. When they fall below their respective thresholds (1.3 V and 2.5 V), the signals BROWNOUT3_3VINT and BROWNOUT1_5VINT assert and can cause an interrupt to the Cortex-M3 processor if they have been previously enabled. These signals are also readable as status bits in the MSS_SR.

The user can disable the PSM and VR to save power. It is assumed that the user's firmware is executing out of internal or external SRAM in this mode. When the PSM is disabled, the eNVM is also disabled. As can be seen in Figure 10-4 on page 154, the PSM_EN must be set to a 0 to turn off the PSM. The PSM_EN can be set to 0 by doing all of the following:

- Setting ENVM_SR to a 1 in the SOFT_RST_CR register
- Turning the Analog block off by clearing the ABPOWERON bit in the ANA_COMM_CTRL register

- Clearing the bit BGPSMENABLE in the VRPSM_CR

PSM Block Diagram

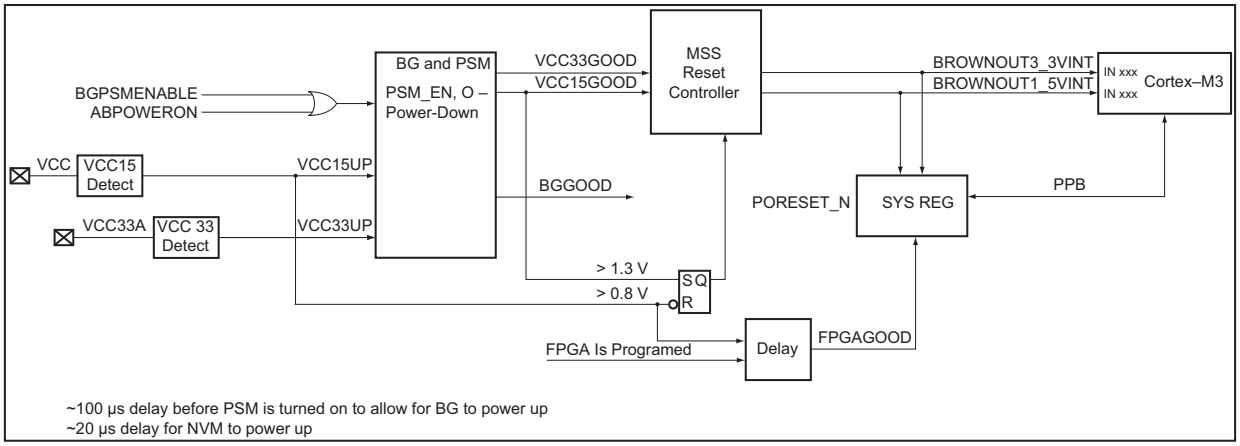


Figure 10-4 • **Power Supply Monitor**

Power-Up Sequence

1. No power applied to chip
2. Regardless of which supply comes up first, all digital logic will be held in reset state until the VCC detect circuit (VCC15UP signal) reaches the trip point (approximately 0.8 V).

   In the reset state (VCC < 0.8V):

   a. All registers are forced to their default state.

   b. The RC-Osc begins oscillating.

   c. The MSS_CCC drives RC-Osc / 4 into the MSS clock pin FCLK.

   d. PORESET_N into the MSS is held low
3. Once VCC15GOOD is high (from the PSM) at around 1.3 V, the MSS reset is removed (PORESET_N goes high). The MSS will then initiate a read from eNVM at logical address zero. The eNVM will hold off response (by deasserting HREADY) until the eNVM is functional (approximately 20 µS). The MSS starts executing factory boot code then jumps to the system boot code to continue with low-level device initialization.

Power-Down Sequence

If VCC33A drops first:

If VCC33A (3.3 V supply to PSM) falls below approximately 2 V, the BGGOOD signal will go to logic 0 and the eNVM will be reset. The MSS will stop operating, since any eNVM access will not complete.

If VCC drops first:

PORESET_N into MSS remains high until VCC15UP goes low (when VCC < ~ 0.65 V). PORESET_N goes high based on the VCC15GOOD signal (VCC > ~1.3 V) but goes low based on the VCC15UP signal (VCC < ~0.65 V). This helps prevent transient supply noise from resetting the MSS. If desired, an interrupt can be generated to the Cortex-M3 processor when the VCC15GOOD signal falls below 1.3 V. This interrupt is called BROWNOUT1_5VINT and is connected to INTISR[1] of the Cortex-M3 NVIC.

VR and PSM Interrupts

Table 10-1 lists the interrupts associated with the VR and PSM. These interrupts must be enabled in the NVIC of the Cortex-M3 processor by setting the appropriate bit to a 1.

***Table 10-1* • VR and PSM Related Interrupts**

Name	Cortex-M3 Interrupt	NVIC Address	NVIC Bit at Address	Function
BROWNOUT1_5VINT	INTISR[1]	0xE000E100	1	1.5 V below threshold
BROWNOUT3_3VINT	INTISR[2]	0xE000E100	2	3.3 V below threshold
PU_NINT	INTISR[4]	0xE000E100	4	PU_N pin asserted low.

SmartFusion Power Modes

SmartFusion devices provide various methods to control power consumption. For specific power contribution numbers, refer to the DC and Switching Characteristics section of the *SmartFusion Customizable System-on-Chip (cSoC)* datasheet.

SoC

This is the normal mode of operation where both the MSS and the FPGA fabric are operational.

FCLK is running and the Cortex-M3 processor is active. All memory controllers are enabled. This is the default mode of operation after a power-on reset when the device is configured for the 1.5 V regulator to be active on power-up (PUPO = 1). If the device is not configured for the 1.5 V regulator to be active on power-up (PUPO = 0), this state can be entered when PU_N = 0. Refer to Figure 10-5 on page 156 for entry and exit transition requirements.

Standby Mode

This mode is for applications that intend to put the device into a low-power state but be ready to respond to an interrupt that is sourced from the MSS, the FPGA, or the analog front-end. Firmware transitions into this mode after reset by executing a "wait for interrupt" (WFI) instruction in the Cortex-M3, causing FCLK to be gated off to the Cortex-M3 processor. This disables the majority of the Cortex-M3 logic. In Standby mode, the SmartFusion device is active, but running off of a lower frequency clock than what is used for normal system operation. For example, the 32 KHz oscillator can be used to clock the MSS. Peripherals not being used can be put into a low-power state by asserting their individual resets in the SOFT_RST_CR. In addition, if the analog front-end is not needed during this state, the user can turn off portions or the entire analog block. The ABPOWERDOWN bit in the ANA_COMM_CTRL register will disable the entire analog front-end. Specifically ABPOWERDOWN does the following:

- When asserted it sends the 3.3 V supply to the ADC, SCB, and SDD.
- When asserted it generates the 2.56 V voltage reference for ADC and SDD.
- When deasserted, 3.3 V is not applied to ADC, SCB, and SDD to save power and the OPAMP which generates the 2.56 V reference is disabled.

Users also have the option of turning off all the ADCs at once by setting ADCSPWRDWN in ANA_COMM_CTRL or turning them off individually by setting the PWRDWN bit in the ADCx_MISC_CTRL, where x equals 0, 1, or 2, indicating which ADC to power down.

Time Keeping Mode

In Time keeping Mode, the only supplies to the SmartFusion device that are enabled are the VBAT rails. Users can transition to Time Keeping mode from Standby mode by having all the supplies turned off except for VBAT. Typically a lithium ion coin cell is connected to VBAT. The RTC in this mode will keep track of time while the lithium coin cell is still charged. Typical current consumption in this mode is 10 µa. When VCC33A and the VCCIO supplies are again turned on to the SmartFusion device, the device will wake up on an RTC_MATCH or assertion of PU_N. This mode can be used to keep track of elapsed time in the event of a power outage or in portable devices when users swap out the main battery.

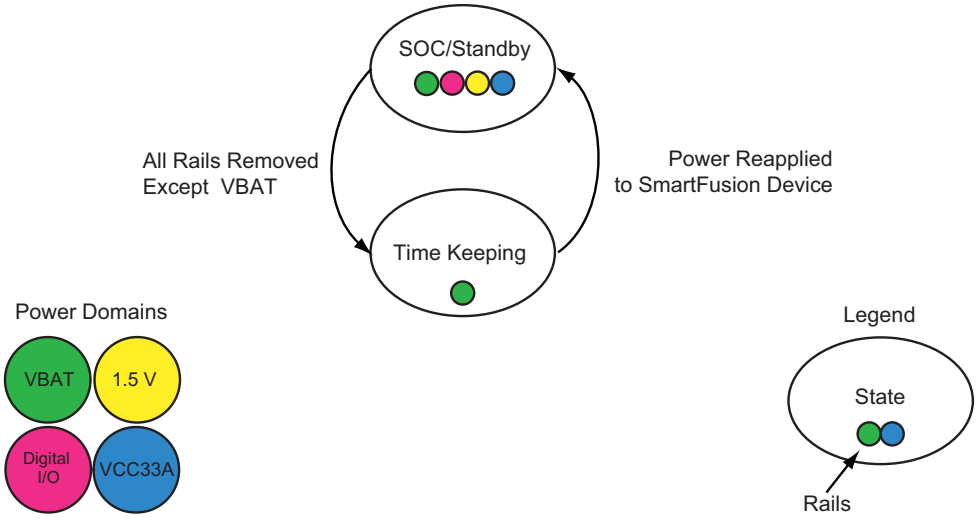


***Figure 10-5* • Power State Diagram**

Control and Status Registers

Table 10-2 through Table 10-6 on page 161 list the various control and status registers associated with the VR and PSM.

***Table 10-2* • VR and PSM Control Registers**

Register Name	Address	R/W	Reset Value	Description
CLR_MSS_SR	0xE0042020	R	0	CLR MSS status information
DEVICE_SR	0xE0042034	R	0	Provides device level status information
MSS_SR	0xE004201C	R/W	0	Provides MSS status information
VRPSM_CR	0xE0042064	R/W	0x10	Control on chip VR

***Table 10-3* • MSS_SR**

Bit Number	Name	R/W	Reset Value	Function
31:11	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	PLLLOCKLOSTINT	R	0	This bit indicates that a falling edge event occurred on PLLLOCK. This signal is also available to the FPGA fabric. This indicates that the PLL lost lock. This signal corresponds to IRQ23 in the Cortex-M3 NVIC. IRQ23 corresponds to bit location 23 in the 32-bit word at address location 0xE000E100. This bit is read only and can be cleared by writing a 1 to the CLRPLLLOCKLOSTINT bit in the CLR_MSS_SR register. 0 = "Don't care." 1 = PLL lost lock.
9	PLLLOCKINT	R	0	This bit indicates that a rising edge event occurred on PLLLOCK. This signal is also available to the FPGA fabric, indicating the PLL is locked. This signal corresponds to IRQ22 in the Cortex-M3 NVIC. IRQ22 corresponds to bit location 22 in the 32-bit word at address location 0xE000E100. This bit is read-only and can be cleared by writing a 1 to the CLRPLLLOCKLOSTINT bit in the CLR_MSS_SR register. 0 = "Don't care." 1 = PLL locked.
8:4	COM_ERROR_STATUS	R	0	Each bit on this bus indicates whether any accesses by the corresponding master on the AHB bus matrix resulted in HRESP assertion by the slave to the AHB bus matrix, HRESP assertion by the AHB bus matrix to that master (in the case of blocked fabric master), or was decoded by the AHB bus matrix as being an unimplemented address space. These register bits are sticky and are cleared by writing a 1 to the COM_CLEARSTATUS bit in the CLR_MSS_SR register. These signals are not used as interrupts to the Cortex-M3 processor Instead, they are ORed together in the AHB bus matrix to create a signal called COM_ERRORINTERRUPT, which is used as an interrupt to the Cortex-M3 processor. This signal corresponds to IRQ 24 in the Cortex-M3 NVIC. IRQ 24 corresponds to bit location 24 in the 32-bit word at address location 0xE000E100. COM_ERRORINTERRUPT is not brought into the system registers space as a status bit for the user's firmware to read. Bit definitions are as follows: Bit 8: Peripheral DMA master, Bit 7: Ethernet MAC master, Bit 6: Fabric master, Bit 5: Cortex-M3 system bus master, Bit 4: Cortex-M3 ICODE/DCODE bus master.

***Table 10-3* • MSS_SR (continued)**

Bit Number	Name	R/W	Reset Value	Function
3	BROWNOUT3_3VINT	R	0	Indicates that the 3.3 V supply has dropped below 2.5 V. This signal corresponds to IRQ 2 in the Cortex-M3 NVIC. IRQ 2 corresponds to bit location 2 in the 32-bit word at address location 0xE000E100. 0 = "Don't care." 1 = 3.3 V has fallen below 2.5 V.
2	BROWNOUT1_5VINT	R	0	Indicates that the 1.5 V supply has dropped below 1.3 V. This signal corresponds to IRQ 1 in the Cortex-M3 NVIC. IRQ 1 corresponds to bit location 1 in the 32-bit word at address location 0xE000E100. 0 = "Don't care." 1 = 1.5 V has fallen below 1.3 V.
1	WDOGTIMEOUTEVENT	R	0	This signal is a sticky version of the WDOGTIMEOUTINT signal (which is itself sticky but is cleared by MSS_SYSTEM_RESET_N). WDOGTIMEOUTEVENT is not affected by MSS_SYSTEM_RESET_N. This allows firmware to determine if a system reset occurred due to a watchdog timeout event. This signal is not used as an interrupt to the Cortex-M3 processor This bit is reset to 0 by PORESET_N only and is unaffected by MSS_SYSTEM_RESET_N. 0 = "Don't care." 1 = Watchdog has timed out.
0	RTCMATCHEVENT	R	0	This signal is a sticky version of the MATCH_SYNC signal from the RTC. If a rising edge event is seen on MATCH_SYNC, after synchronization to FCLK domain, this bit is asserted. It stays asserted until cleared by CLRRTCMATCHEVENT. This signal is used as an interrupt to the Cortex-M3 processor. This signal corresponds to IRQ 3 in the Cortex-M3 NVIC. IRQ 3 corresponds to bit location 3 in the 32-bit word at address location 0xE000E100. 0 = "Don't care." 1 = RTC has matched event.

***Table 10-4* • CLR_MSS_SR**

Bit Number	Name	R/W	Reset Value	Function
31:11	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CLRPLLLOCKLOSTINT	R	0	Writing a 1 to this bit clears the interrupt signal PLLLOCKLOSTINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the PLLLOCKLOSTINT signal.
9	CLRPLLLOCKINT	R	0	Writing a 1 to this bit clears the interrupt signal PLLLOCKINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the PLLLOCKINT signal.
8:4	COM_CLEARSTATUS	R	0	Writing a 1 to any bits in COM_CLEARSTATUS clears the interrupt signal COM_ERRORINTERRUPT. Writing a zero has no effect. Bit 8: Peripheral DMA master Bit 7: Ethernet MAC master Bit 6: Fabric master Bit 5: Cortex-M3 system bus master Bit 4: Cortex-M3 ICODE/DCODE bus master
3	CLRBROWNOUT3_3VINT	R	0	Writing a 1 to this bit clears the interrupt signal BROWNOUT3_3VINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the BROWNOUT3_3VINT signal.
2	CLRBROWNOUT1_5VINT	R	0	Writing a 1 to this bit clears the interrupt signal BROWNOUT1_5VINT. Writing a zero has no effect. 0 = No effect. 1 = Clear the BROWNOUT1_5VINT signal.
1	CLRWDOGTIMEOUTEVENT	R	0	Writing a 1 to this bit clears the interrupt signal WDOGTIMEOUTEVENT. Writing a zero has no effect. 0 = No effect. 1 = Clear the WDOGTIMEOUTEVENT signal.
0	CLRRTCMATCHEVENT	R	0	Writing a 1 to this bit clears the interrupt signal RTCMATCHEVENT. Writing a zero has no effect. 0 = No effect. 1 = Clear the RTCMATCHEVENT signal.

Table 10-5 • **DEVICE_SR**

Bit Number	Name	R/W	Reset Value	Function
31:7	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	FPGAGOOD	R	x	When 0, FPGA fabric is not programmed. In this state, all inputs from the FPGA fabric to the MSS are guaranteed to be low and outputs from MSS to FPGA fabric can be driven to any value. When 1, FPGA fabric is programmed. In this case, outputs from the MSS to the fabric may be driven normally and inputs from the fabric to the MSS may be interpreted as valid. This bit also indicates the FPGA fabric is powered up. 0 = Powered down or not programmed. 1 = Powered up and programmed. The reset value of this bit depends on whether or not the FPGA has been programmed.
5	FPGAPROGRAMMING	R	0	0 = Indicates the FPGA fabric is not in programming mode. 1 = Indicates the FPGA fabric is in programming mode.
4:3	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	Reserved	R	–	Reserved
1	BROWNOUT3_3VN	R	x	0 = 3.3 V supply has fallen below 2.5 V. 1 = 3.3 V supply okay. The reset state depends on the state of the 3.3 V power supply.
0	BROWNOUT1_5VN	R	x	0 = 1.5 V supply has fallen below 1.3 V. The reset state depends on the state of the 1.5 V power supply. 1 = 1.5 V supply okay.

***Table 10-6* • VRPSM_CR**

Bit Number	Name	R/W	Reset Value	Function
31:5	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BGPSMENABLE	R/W	1	0 = Allow the BG and PSM to be turned off (BG and PSM will then be controlled by the ABPOWERON bit in ANA_COMM_CTRL or the ENVM_SR bit in SOFT_RST_CR). 1 = Turn on the BG and PSM.
3	Reserved	R/W	0	Reserved
2	CLR_PU_NINT	W	0	Writing a one generates a pulse which clears the PU_NINT interrupt.
1	FPGAVRONENABLE	R/W	0	Fabric VR On (FPGAVRON) signal qualifier. When FPGAVRONENABLE = 1, FPGAVRON from the fabric is allowed to shut down the VR. When FPGAVRONENABLE = 0, FPGAVRON from the fabric cannot cause the VR to shut down.
0	MSSVRON	R/W	0	By pulsing this signal from low to high to low, firmware will cause the VR to turn off. This has the effect of switching off the power to the MSS and the FPGA fabric.

11 – Watchdog Timer

The Watchdog timer is an advanced peripheral bus (APB) slave that guards against system crashes by requiring that it is regularly serviced by the ARM Cortex-M3 processor or by a processor in the FPGA fabric. It is likely that the most common use model will be one where the Watchdog is serviced by the Cortex-M3 processor.

Watchdog Block Diagram

Figure 11-1 shows the block diagram for the Watchdog timer.

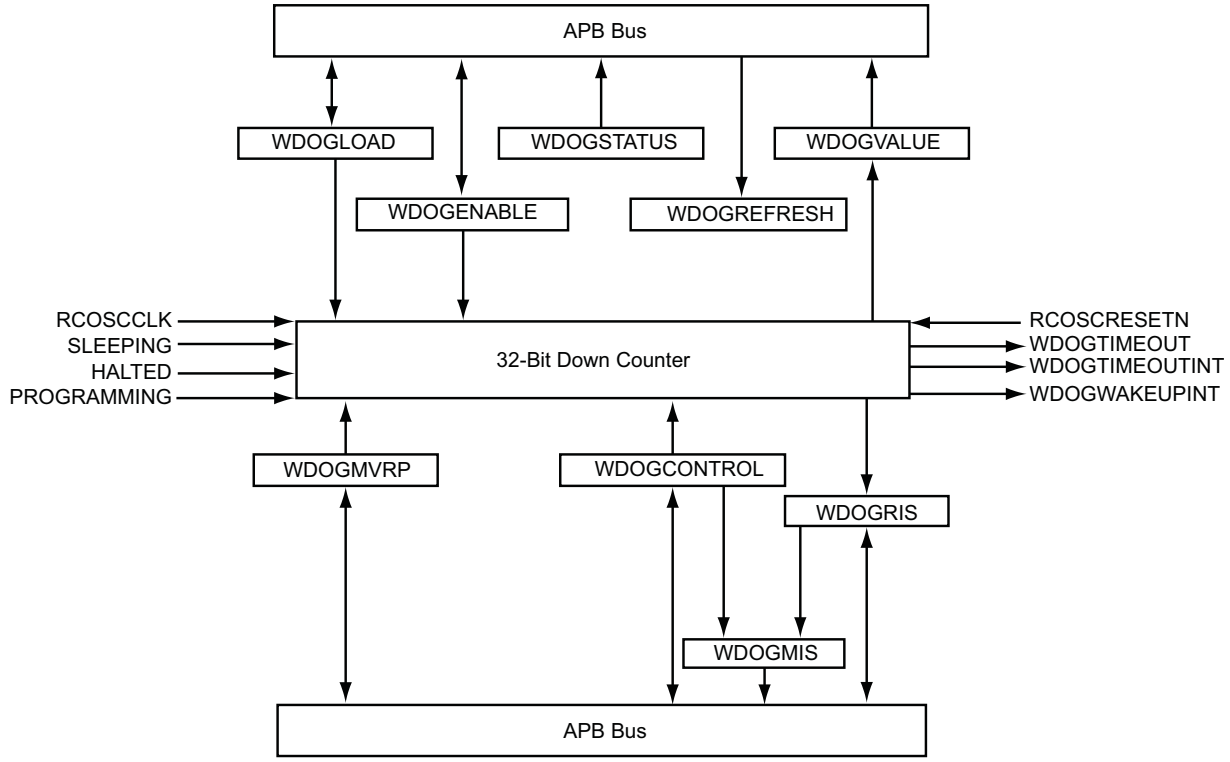


***Figure 11-1* • Watchdog Block Diagram**

Functional Description

The operation of the Watchdog is based on a 32-bit down counter that must be refreshed at regular intervals by the Cortex-M3 processor or by a fabric-based processor. If the counter is not refreshed, it will timeout and either cause a system reset or generate an interrupt to the processor, depending on the value of a control bit. In normal operation, the generation of a reset or timeout interrupt by the Watchdog does not occur because the Watchdog counter is refreshed on a regular basis.

The 32-bit counter in the Watchdog is clocked with the 100 MHz RC oscillator output.

On power-up of the device, the Watchdog is enabled with the timeout period set to approximately 5.37 seconds.

The Watchdog has an APB interface through which the processor can access various control and status registers to control and monitor the operation of the Watchdog. The APB interface is clocked by PCLK0 on APB Bus 0.

Watchdog Timeout: Reset/Interrupt

The MODE control bit in WDOGCONTROL is used to determine whether the Watchdog generates a reset or an interrupt if counter timeout occurs. The default setting is reset generation on timeout. When interrupt generation is selected, the WDOGTIMEOUTINT output is asserted on timeout and remains asserted until the interrupt is cleared. When reset generation is selected, the Watchdog does not directly generate the system reset signal. Instead, when the counter reaches zero, the Watchdog generates a pulse on the WDOGTIMEOUT output and this is routed to the reset controller to cause it to assert the necessary reset signals.

The pulse on the WDOGTIMEOUT output is generated in the RCOSCCLK domain and has a duration of 16 RCOSCCLK clock cycles.

When the counter value reaches zero, it is reloaded with the value stored in the WDOGLOAD register. This will enforce repeated interrupts should the interrupt or pulse not be serviced.

Loading and Refreshing the Watchdog

The WDOGLOAD register is used to store the value which is loaded into the counter each time the Watchdog is refreshed. When the WDOGLOAD register is updated, the least significant six bits are always set to 0x3F, or 64 clock cycles, regardless of the value written to it. In effect, this means there is a lower limit on the value that can be written to the counter. After refreshing, at least 64 RCOSCCLK clock cycles are required before the counter times out. The purpose of this feature is to prevent a Watchdog reset/interrupt from occurring immediately after or during refresh in the case where a very low value has been written to the WDOGLOAD register.

The Watchdog counter is refreshed by writing the value 0xAC15DE42 to the WDOGREFRESH register. This causes the counter to be loaded with the value in the WDOGLOAD register. An appropriate value must be written to the WDOGLOAD register before writing to the WDOGREFRESH register.

Forbidden and permitted windows in time regulate when refreshing can occur. The size of these windows is controlled by the value programmed in the WDOGMVRP control register.

When the counter value is greater than the value in WDOGMVRP, refreshing the Watchdog is forbidden. If a refresh is executed in these circumstances, the refresh is successful but a reset or interrupt (depending on the operation mode selected) is also generated. This is illustrated in Figure 11-2 on page 165.

When the counter value falls below the level programmed in WDOGMVRP, refreshing of the Watchdog is permitted. The REFRESHSTATUS status bit in the WDOGSTATUS register is set when in the permitted window and cleared when in the forbidden window.

It is possible to avoid having forbidden and permitted windows by ensuring that the value in WDOGMVRP is greater than the value in WDOGLOAD.

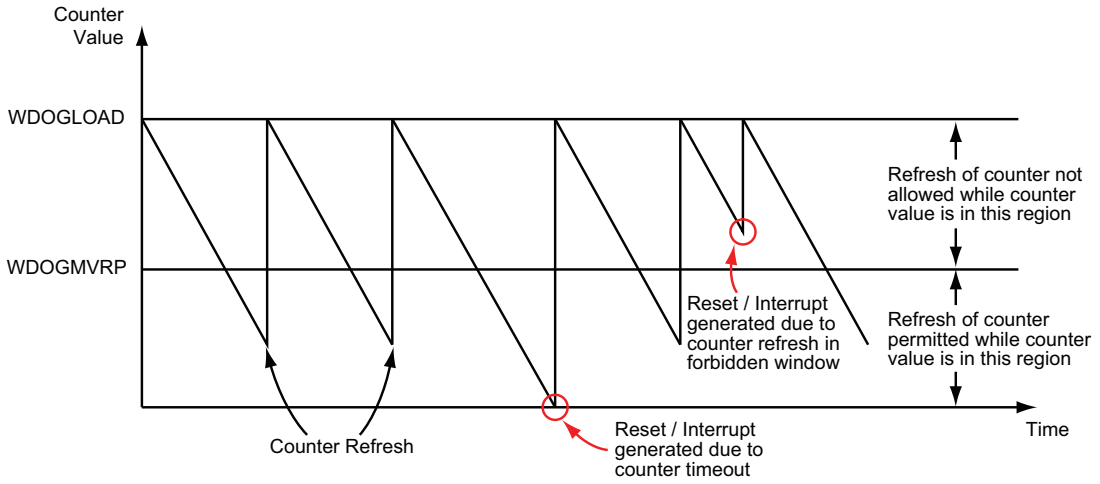


***Figure 11-2* • Watchdog Timer Windowing Example**

Watchdog Behavior with Processor Modes and Device Programming

This section describes the behavior of the Watchdog in Cortex-M3 processor modes and when the device is being programmed.

Cortex-M3 Processor in Debug State

The Watchdog counter is halted when the Cortex-M3 processor enters the Debug state. This ensures that Watchdog timeout related resets or interrupts do not occur when a system debug session is in progress.

Cortex-M3 Processor in Sleep Mode

The Cortex-M3 processor can be put into a low-power state by entering into a sleep mode. The processor exits sleep mode when an interrupt occurs. The Watchdog can be configured to generate an interrupt if its counter value moves from the forbidden to the permitted window (at the WDOGMVRP level) when the Cortex-M3 processor is in sleep mode. The processor will wake up and refresh the Watchdog and then go back into sleep mode. The WDOGWAKEUPINT output from the Watchdog is used for this interrupt. The WAKEUPINTEN control bit in the WDOGCONTROL register is used to enable/disable generation of the WDOGWAKEUPINT interrupt, with the default setting being disabled.

eNVM Being Programmed

The Watchdog is disabled when the external JTAG interface is used to program the embedded NVM on the device. When the programming cycle has finished, the Watchdog behaves as if it has just come out of reset.

Watchdog Interrupts

There are two interrupt outputs from the Watchdog: WDOGTIMEOUTINT and WDOGWAKUPINT.

WDOGTIMEOUTINT – This is asserted (if enabled) when a counter timeout occurs and interrupt rather than reset generation has been selected. This interrupt is connected to the Non Maskable Interrupt (NMI) input of the Cortex-M3 processor and also drives the WDINT interrupt to the FPGA fabric.

WDOGWAKEUPINT – This is asserted (if enabled) on crossing the WDOGMVRP level when the SLEEPING input is asserted. This interrupt is mapped to interrupt request 0 (IRQ 0) in the Cortex-M3 interrupt controller. The FPGA fabric also has visibility of the WDOGWAKEUPINT interrupt, but not as a single, independent signal. See the "Fabric Interface and IOMUX" section on page 343 for information on how WDOGWAKEUPINT is combined with a number of other MSS interrupts.

Watchdog Register Interface Summary

Table 11-1 summarizes the Watchdog register interface. Detailed descriptions of the registers are given in the "Watchdog Register Interface Details" section.

***Table 11-1* • Watchdog Register Interface**

Register Name	Address	R/W	Reset Value	Description
WDOGVALUE	0x40006000	R	0x20000000	Current value of counter
WDOGLOAD	0x40006004	R/W	0x20000000	Load value for counter
WDOGMVRP	0x40006008	R/W	0xFFFFFFFF	Maximum value for which refreshing is permitted
WDOGREFRESH	0x4000600C	W	N/A	Writing the value 0xAC15DE42 to this register causes the counter to be updated with the value in WDOGLOAD register.
WDOGENABLE	0x40006010	R/W	0x1	Watchdog enable register
WDOGCONTROL	0x40006014	R/W	0x0	Control register
WDOGSTATUS	0x40006018	R	0x1	Status register
WDOGRIS	0x4000601C	R/W	0x0	Raw interrupt status
WDOGMIS	0x40006020	R	0x0	Masked interrupt status
MSS_SR	0xE004201C	R	0x0	MSS Status register

Watchdog Register Interface Details

This section describes each of the Watchdog registers in detail.

Watchdog Value Register (WDOGVALUE)

***Table 11-2* • WDOGVALUE**

Bit Number	Name	R/W	Reset Value	Description
31:0	CURRENT_COUNT	R	0x20000000	This read only register contains the current value of the Watchdog's 32-bit counter.

Watchdog Load Register (WDOGLOAD)

***Table 11-3* • WDOGLOAD**

Bit Number	Name	R/W	Reset Value	Description
31:0	COUNT_UPDATE_VALUE	R/W	0x20000000	The value stored in this register is used to update the counter whenever the value 0xAC15DE42 is written to the WDOGREFRESH register. When the WDOGLOAD register is written to, the lower 6 bits of the register are always set to 1. This sets a lower limit of 0x3F on the value written to the counter during a refresh.

Watchdog Maximum Value Refresh Permitted Register (WDOGMVRP)

***Table 11-4* • WDOGMVRP**

Bit Number	Name	R/W	Reset Value	Description
31:0	MVRP	R/W	0xFFFFFFFF	The value stored in this register is the maximum counter value for which a refresh is permitted. If the Watchdog is refreshed (by writing 0xAC15DE42 to the WDOGREFRESH register) when the counter value is crossing the value stored in the WDOGMVRP register, then the refresh does succeed but an interrupt or reset is also generated. The Watchdog should only be refreshed when the counter value is less than the value stored in the WDOGMVRP register. If the value stored in the WDOGMVRP register is greater than the value held in the WDOGLOAD register (and also greater than the current value of the counter), then a refresh of the Watchdog can be carried out at any time without generating an interrupt or reset.

Watchdog Refresh Register (WDOGREFRESH)

***Table 11-5* • WDOGREFRESH**

Bit Number	Name	R/W	Reset Value	Description
31:0	REFRESH_KEY	W	N/A	This is a write only register which reads as zero. Writing the value 0xAC15DE42 to this register causes the counter to be refreshed with the value in the WDOGLOAD register. If this register is written to while the current value of the counter is greater than the value in the WDOGMVRP register, the counter will be refreshed and a reset or timeout interrupt will be generated (depending on the MODE bit of WDOGCONTROL). While the counter value is greater than WDOGMVRP, there is effectively a time window in which it is forbidden to refresh the Watchdog. When the counter is between the WDOGMVRP level and zero, the Watchdog is in a time window which permits it to be refreshed. It is possible to avoid having forbidden and permitted time windows for refreshing the Watchdog by setting the value of the WDOGMVRP register to a value greater than that stored in the WDOGLOAD register.

Watchdog Enable Register (WDOGENABLE)

***Table 11-6* • WDOGENABLE**

Bit Number	Name	R/W	Reset Value	Description
31:0	DISABLE_KEY	R/W	0x1	The Watchdog is enabled at power-up. After power-up, the ENABLE bit can be cleared by writing the value 0x4C6E55FA to the address of this register. Subsequent to this, the ENABLE bit can only be set again by a power-on reset.
0	ENABLE	R/W	0x1	This is the actual ENABLE bit that is used by the Watchdog. Users can read whether the Watchdog is enabled or disabled by reading this bit. Note that this bit overlays the DISABLE_KEY.

Watchdog Control Register (WDOGCONTROL)

***Table 11-7* • WDOGCONTROL**

Bit Number	Name	R/W	Reset Value	Description
31:3	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	MODE	R/W	0x0	Watchdog mode of operation. 0 = Reset generated if counter reaches zero. 1 = WDOGTIMEOUTINT interrupt generated (if enabled) if counter reaches zero.
1	WAKEUPINTEN	R/W	0x0	0 = WDOGWAKEUPINT interrupt generation disabled. 1 = WDOGWAKEUPINT interrupt generation enabled.
0	TIMEOUTINTEN	R/W	0x0	0 = WDOGTIMEOUTINT interrupt generation disabled. 1 = WDOGTIMEOUTINT interrupt generation enabled.

Watchdog Status Register (WDOGSTATUS)

***Table 11-8* • WDOGSTATUS**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	REFRESHSTATUS	R	0x1	0 = Counter in forbidden window, refresh forbidden. Refreshing the Watchdog when REFRESHSTATUS = 0 will cause an interrupt or reset to be generated.

Watchdog Raw Interrupt Status Register (WDOGRIS)

***Table 11-9* • WDOGRIS**

Bit Number	Name	R/W	Reset Value	Description
31:2	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	WAKEUPRS	R/W	0x0	Raw Status of the WDOGWAKEUPINT interrupt. Writing 1 to this bit clears the bit. Writing 0 has no effect.
0	TIMEOUTRS	R/W	0x0	Raw Status of the WDOGTIMEOUTINT interrupt. Writing 1 to this bit clears the bit. Writing 0 has no effect.

Watchdog Masked Interrupt Status Register (WDOGMIS)

***Table 11-10* • WDOGMIS**

Bit Number	Name	R/W	Reset Value	Description
31:2	Reserved	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation
1	WAKEUPMS	R	0x0	Status of the WDOGWAKEUPINT interrupt. This is the logical AND of the WAKEUPRS bit of the WDOGRIS register and WAKEUPINTEN bit of the WDOGCONTROL register.
0	TIMEOUTMS	R	0x0	Status of the WDOGTIMEOUTINT interrupt. This is the logical AND of the TIMEOUTRS bit of the WDOGRIS register and TIMEOUTINTEN bit of the WDOGCONTROL register.

MSS Status Register (MSS_SR)

The MSS Status register holds the status of system critical events, such as brownout and timeout coming from different parts of the MSS system.

***Table 11-11* • MSS_SR**

Bit Number	Name	R/W	Reset Value	Description
31:8	Reserved	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	COM_ERRORSTATUS	R	0x0	Each bit on this bus indicates whether any accesses by the corresponding master on the Communications Matrix resulted in HRESP assertion by the slave to the Communications Matrix, HRESP assertion by the Communications Matrix to that master (in the case of blocked fabric master), or decoding by the Communications Matrix as being unimplemented address space. These register bits are "sticky" and are cleared by the writing 1 to the COM_CLEARSTATUS bit in SysReg. The bit definitions are as follows: • Bit 0: Corresponds to Cortex-M3 ICODE/DCODE master. • Bit 1: Corresponds to Cortex-M3 SYSTEM master. • Bit 2: Corresponds to fabric master. • Bit 3: Corresponds to Ethernet MAC master.
3	BROWNOUT3_3VINT	R	0x0	Sticky interrupt derived from a falling edge event on the BROWNOUT3_3V_SYNCN input from the analog subsystem (via synchronization in the reset controller), indicating that the 3.3 V supply has dropped below a specified threshold voltage. This signal is used as an interrupt to the Cortex-M3 processor.
2	BROWNOUT1_5VINT	R	0x0	Sticky interrupt derived from a falling edge event on the BROWNOUT1_5V_SYNCN input from the analog subsystem (via synchronization in the reset controller), indicating that the 1.5 V supply has dropped below a specified threshold voltage. This signal is used as an interrupt to the Cortex-M3 processor.

***Table 11-11* • MSS_SR (continued)**

Bit Number	Name	R/W	Reset Value	Description
1	WDOGTIMEOUTEVENT	R	0x0	This signal is a sticky version of the WDOGTIMEOUTINT signal (which is itself sticky but is cleared by MSS_SYSTEM_RESET_N). WDOGTIMEOUTEVENT is not affected by MSS_SYSTEM_RESET_N. This allows firmware to determine if a system reset occurred due to a watchdog timeout event. This signal is not used as an interrupt to the Cortex-M3 processor. This bit is reset to 0 by PORESET_N only and is unaffected by MSS_SYSTEM_RESET_N. The MSS Status Register is only visible to the Cortex-M3 processor and cannot be accessed by a processor located in the FPGA fabric. If required, the user could implement a register bit with similar behavior to WDOGTIMEOUTEVENT in the fabric. One approach to doing this would be to create a processor readable flip-flop which is set when the FPGARESETN signal asserts and is cleared by a power-on reset. 0 = "Don't care." 1 = Watchdog has timed out.
0	RTCMATCHEVENT	R	0x0	This signal is a sticky version of the MATCH_SYNC signal from the RTC. If a rising edge event is seen on MATCH_SYNC, after synchronization to the FCLK domain, this bit is asserted. It stays asserted until cleared by CLRRTCMATCHEVENT. This signal is used as an interrupt to the Cortex-M3 processor. The reset value could be either 0 or 1, depending on the SmartFusion top-level tie-off.

12 – Ethernet MAC

Introduction

The SmartFusion Ethernet MAC is a high-speed media access control (MAC) Ethernet controller. It implements carrier sense multiple access with collision detection (CSMA/CD) algorithms defined by the IEEE 802.3 standard. The Ethernet MAC complies with the low-pin-count Reduced Media Independent Interface (RMII™) specification, as defined by the RMII Consortium, to interface to an external physical layer (PHY) device. Communication with the ARM Cortex-M3 processor is implemented via a set of Control and Status registers on an APB slave interface. The Ethernet MAC is an AHB bus master on the AHB bus matrix (see the "AHB Bus Matrix" section on page 15). The built-in DMA controller inside the MAC block, along with the AHB master interface, is used to automatically move data between external RAM and the built-in transmit FIFO and receive FIFOs with minimal CPU intervention. Linked list management enables the use of various memory allocation schemes. Internal RAMs are used as configurable FIFO memory blocks, and there are separate memory blocks for transmit and receive processes. The host interface uses little-endian byte ordering for the address space.

Ethernet MAC Block Diagram

Figure 12-1 shows the Ethernet MAC block diagram.

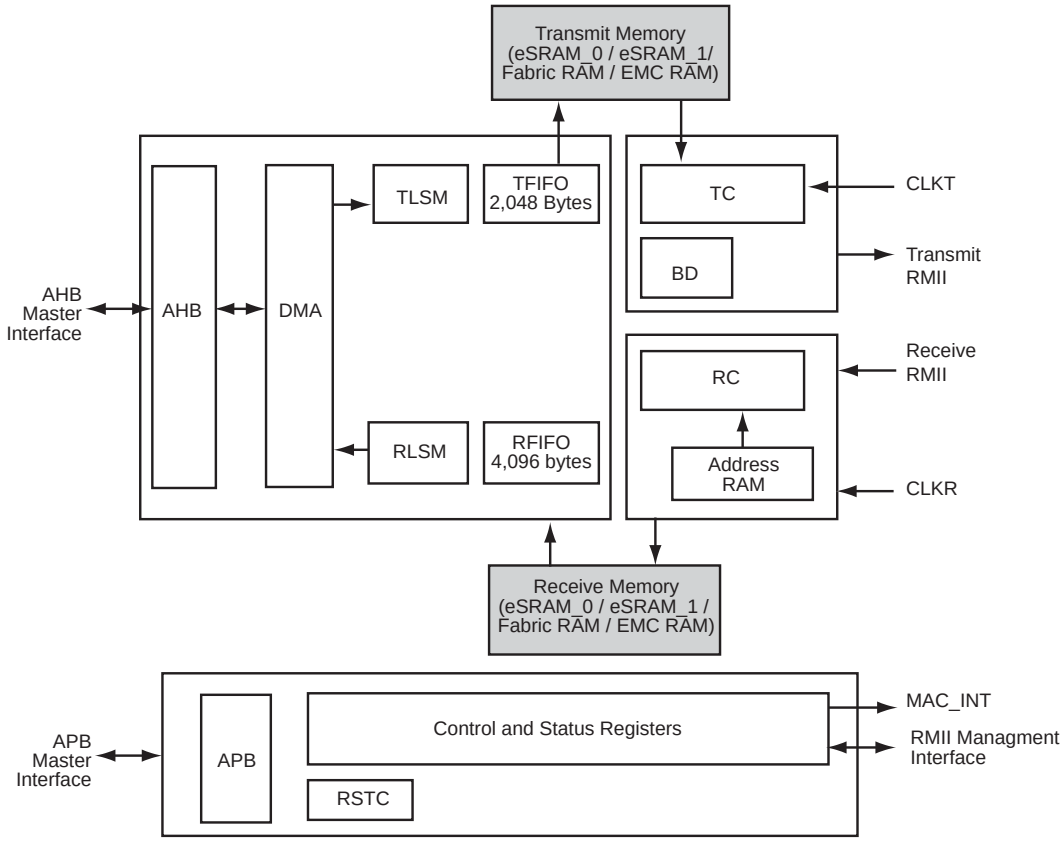


***Figure 12-1* • Ethernet MAC Block Diagram**

The RMII management interface can be used to control the external PHY device from the host side. It allows access to all of the internal PHY registers via a simple two-wire interface. There are two signals on the RMII management interface: the MDC (Management Data Clock) and the MDIO (Management Data I/O). The IEEE 802.3 indirection tristate signal defines the MDIO. Figure 12-2 shows the RMII management interface to an external RMII PHY device.

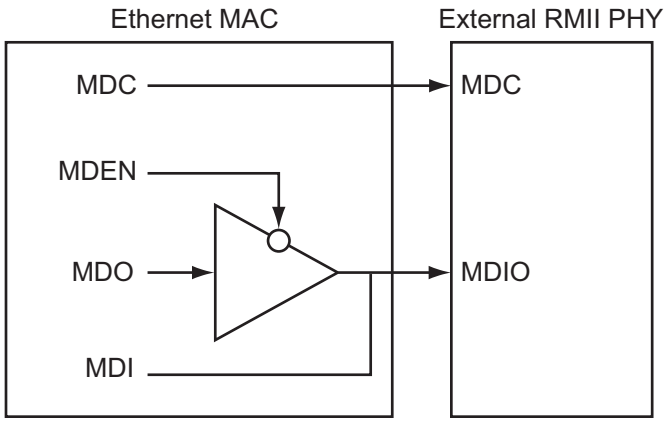


***Figure 12-2* • RMII Management Interface**

Functional Blocks of Ethernet MAC

AHB Master Interface

The AHB block implements an AHB master function, allowing the DMA controller to access memory on the AHB bus.

APB Slave Interface

This APB block implements an APB slave interface, allowing the Cortex-M3 processor to access the Control and Status Registers set (CSR).

Control/Status Register Logic (CSR)

The CSR component is used by the Cortex-M3 processor to control Ethernet MAC operation. It contains the CSR register set and the interrupt controller. The CSR also provides an RMII management interface, which the Cortex-M3 processor can access via reading and writing to the CSR registers. Refer to the "CSR Definitions" section on page 197.

Direct Memory Access Controller (DMA)

The direct memory access DMA controller implements the host data interface. It services both the receive and transmit channels. The TLSM and TFIFO have access to one DMA channel. The RLSM and RFIFO have access to the other DMA channel.

Transmit Linked List State Machine (TLSM)

The transmit linked list state machine implements the descriptor/buffer architecture of Ethernet MAC. It manages the transmit descriptor list and fetches the data prepared for transmission from the data buffers into the transmit FIFO.

Transmit FIFO (TFIFO)

The transmit FIFO is used for buffering data prepared for transmission by Ethernet MAC. It fetches the transmit data from the host via the DMA interface. The size of the transmit FIFO is 2,048 bytes (512 x 32 bits), which holds one packet up to 1,532 bytes.

Transmit Controller (TC)

The transmit controller implements the 802.3 transmit operation. From the network side, it uses the standard 802.3 RMII interface for an external PHY device. The TC unit reads transmit data from the external transmit data RAM, formats the frame, and transmits the framed data via the RMII.

Backoff/Deferring (BD)

The backoff/deferring controller implements the 802.3 half-duplex operation. It monitors the status of the Ethernet bus and decides whether to perform a transmit or backoff/deferring of the data via the RMII.

Receive Linked List State Machine (RLSM)

The receive linked list state machine implements the descriptor/buffer architecture of Ethernet MAC. It manages the receive descriptor list and moves the data from the receive FIFO into the data buffers.

Receive FIFO (RFIFO)

The receive FIFO is used for buffering data received by Ethernet MAC. The size of the FIFO is 4,096 bytes (1,024 x 32 bits), which holds two packets up to 1,532 bytes each. During reception, if the RX FIFO becomes full while receiving a partial frame, that partial frame in the RX FIFO is written into memory with a CRC error. The frames received after the RX FIFO is full are dropped. The next incoming frames are received by MAC when there is enough space in the RX FIFO to accommodate them.

Receive Controller (RC)

The receive controller implements the 802.3 receive operation. From the network side it uses the standard 802.3 RMII interface for an external PHY device. The RC block transfers data received from the RMII to the receive data RAM. It supports internal address filtering using an internal address RAM. It also supports an external address filtering interface.

Memory Blocks

There are two external memory blocks required for the proper operation of Ethernet MAC:

- Receive memory – RAM working as receive data memory
- Transmit memory – RAM working as transmit data memory

These RAMs can be implemented in the eSRAM_0, eSRAM_1, external RAMs connected through EMC, or FPGA block SRAM.

Clock and Reset Control

Clock Controls

As shown in Figure 12-3 on page 176, there are five clock domains in the design, including FCLK, PCLK0, CLKR, CLKT, and MAC_CLK. The MAC_CLK is the external 50 MHz clock and CLKT is the internal 25 MHz Transmit clock. CLKT and CLKR are the same frequency.

- The TC and BD components operate synchronously with CLKT. This is a 2.5 MHz clock for 10 Mbps operation or a 25 MHz clock for 100 Mbps operation.

- The RC operates synchronously with CLKR. This is a 2.5 MHz clock for 10 Mbps operation or a 25 MHz clock for 100 Mbps operation.
- The TFIFO, RFIFO, TLSM, RLSM, and DMA components operate synchronously with FCLK clock.
- The CSR operates synchronously with the PCLK0 clock.
- CLK_TX_RX is generated from the negative edge of RMII_CLK. For 100 Mbps operation, the RMII_CLK is divided by 2 to generate CLK_TX_RX. For 10 Mbps operation, the RMII_CLK is divided by 20 to generate CLK_TX_RX.

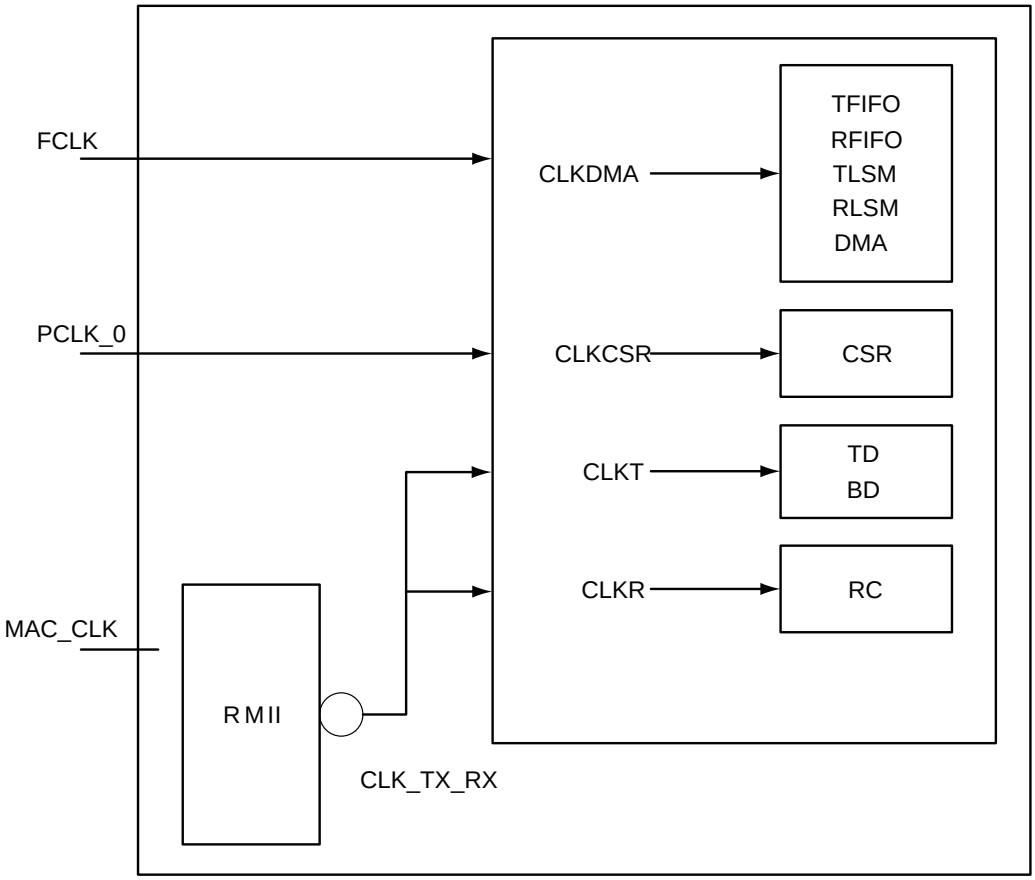


***Figure 12-3* • Ethernet MAC Clocks**

A minimum frequency of PCLK0 is required for proper operation of the transmit, receive, and general-purpose timers. The minimum frequency for PCLK0 must be at least the CLKT frequency divided by 64. For proper operation of the receive timer, the PCLK0 frequency must be at least the CLKR frequency divided by 64. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109 for details on PCLK0 settings. If the clock frequency conditions described above are not met, do not use transmit interrupt mitigation control, receive interrupt mitigation control, or the general-purpose timer. Appropriate clocks should also be supplied when the hardware reset operation is performed.

Reset Control

Reset Controller (RSTC)

The reset controller is used to reset all components of the Ethernet MAC. It generates a reset signal asynchronous to all clock domains in the design from power on reset and software reset.

Software Reset

Software reset can be performed by setting the CSR0[0] (SWR) bit. The software reset will reset all internal flip-flops. The MAC_SR bit of SOFT_RST_CR (0xE0042030) in system registers also acts as a software reset of the Ethernet MAC. Refer to the "Reset Controller" section on page 143 for more information.

Interface Signals

The signals shown in Table 12-1 are included in the Ethernet MAC.

***Table 12-1* • Signals included in Ethernet MAC**

Name	Type	Polarity / Bus Size	Description
RMII PHY Interface			
MAC_RXER	In	High	Receive error If RX_ER is asserted during Ethernet MAC reception, the frame is received and status of the frame is updated with RX_ER.
MAC_CRSDV	In	High	Carrier sense and receive data valid This signal must be asserted by the PHY when either a receive or transmit medium is non-idle. The PHY device should assert MAC_CRSDV when valid data is provided on the RXD signal.
MAC_MDIO	In/Out	1	RMII management data input and output The state of the input signal can be checked by reading the CSR9.19 bit. The output signal is driven by the CSR9.18 bit.
MAC_RXD[1:0]	In	2	Receive data recovered and decoded by PHY. The RXD[0] signal is the least significant bit.
MAC_TXEN	Out	High	Transmit enable When asserted, indicates valid data for the PHY on the TXD port.
MAC_MDC	Out	Rise	RMII management clock = 25 MHz This signal is driven by the CSR9.16 bit.
MAC_TXD[1:0]	Out	2	Transmit data The TXD[0] signal is the least significant bit.
MAC_CLK	In	Rise	50 MHz ± 50 ppm clock source shared with RMII PHY.

Frame Data and Descriptors

Descriptor / Data Buffer Architecture Overview

A data exchange between the host and Ethernet MAC is performed via the descriptor lists and data buffers, which reside in the system shared RAM (eSRAM_0, eSRAM_1, External RAMs connected through EMC, or FPGA block SRAM). The buffers hold the host data to be transmitted or received by Ethernet MAC. The descriptors act as pointers to these buffers. Each descriptor list should be constructed by the host in a shared memory area and can be of an arbitrary size. There is a separate list of descriptors for both the transmit and receive processes.

The position of the first descriptor in the descriptor list is described by CSR3 for the receive list and by CSR4 for the transmit list. The descriptors can be arranged in either a chained or a ring structure (Figure 12-4 on page 178 and Figure 12-5 on page 179). In a chained structure, every descriptor contains a pointer to the next descriptor in the list. In a ring structure, the address of the next descriptor is determined by CSR0[6:2] (DSL—descriptor skip length). Every descriptor can point to up to two data buffers. When using descriptor chaining, the address of the second buffer is used as a pointer to the next descriptor; thus, only one buffer is available. A frame can occupy one or more data descriptors and buffers, but one descriptor cannot exceed a single frame. In a ring structure, the descriptor operation may be corrupted if only one descriptor is used. Additionally, in the ring structure, at least two descriptors must be set up by the host. In a transmit process, the host can give the ownership of the first descriptor to Ethernet MAC and cause the data specified by the first descriptor to be transmitted. At the same time, the host holds the ownership of the second or last descriptor to itself. This is done to prevent Ethernet MAC from fetching the next frame until the host is ready to transmit the data specified in the second descriptor. In a receive process, the ownership of all available descriptors, unless it is pending processing by the host, must be given to Ethernet MAC.

Figure 12-4 shows descriptors in ring structure.

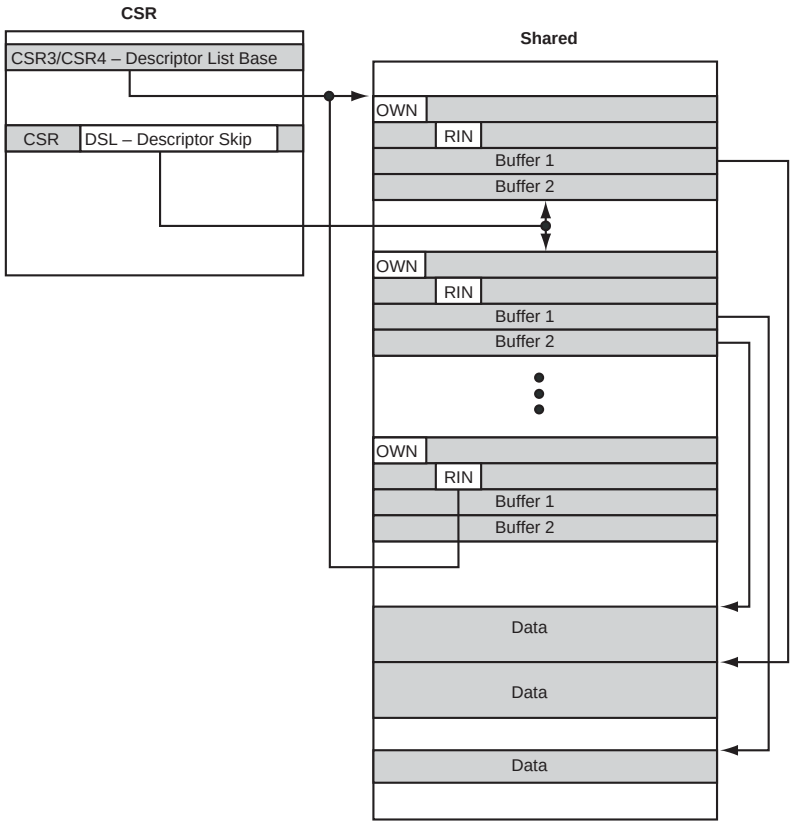


***Figure 12-4* • Descriptors in Ring Structure**

Figure 12-5 shows descriptors in chained structure.

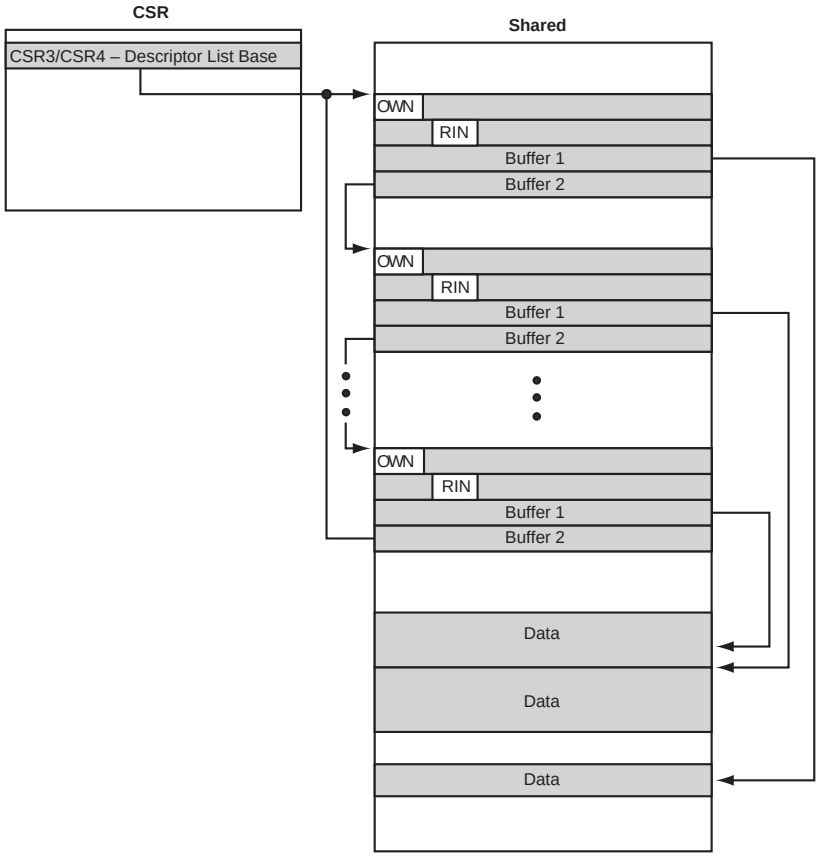


***Figure 12-5* • Descriptors in Chained Structure**

Table 12-2 through Table 12-6 on page 182 give bit descriptions and functions for the receive descriptors

***Table 12-2* • Receive Descriptors (RDESx)**

RDES0	OWN	STATUS		
RDES1		CONTROL	RBS2	RBS1
RDES2	RBA1			
RDES3	RBA2			

Note: The RDESx descriptors reside in receive data memory. They can be defined and addressed under software control.

***Table 12-3* • Receive Descriptor 0 (RDES0) Bit Functions**

Bit	Name	Function
31	OWN	Ownership bit 1 – Ethernet MAC owns the descriptor. 0 – The host owns the descriptor. Ethernet MAC will clear this bit when it completes a current frame reception or when the data buffers associated with a given descriptor are already full.
30	FF	Filtering fail When set, indicates that a received frame did not pass the address recognition process. This bit is valid only for the last descriptor of the frame (RDES0[8] set), when the CSR6[30] (receive all) bit is set and the frame is at least 64 bytes long.
29:16	FL	Frame length Indicates the length, in bytes, of the data transferred into a host memory for a given frame. This bit is valid only when RDES0[8] (last descriptor) is set and RDES0[14] (descriptor error) is cleared.
15	ES	Error summary This bit is a logical OR of the following bits: RDES0[1] – CRC error RDES0[6] – Collision seen RDES0[7]– Frame too long RDES0[11] – Runt frame RDES0[14] – Descriptor error This bit is valid only when RDES0[8] (last descriptor) is set.
14	DE	Descriptor error Set by Ethernet MAC when no receive buffer was available when trying to store the received data. This bit is valid only when RDES0[8] (last descriptor) is set.
13:12		Reserved
11	RF	Runt frame When set, indicates that the frame is damaged by a collision or by a premature termination before the end of a collision window. This bit is valid only when RDES0[8] (last descriptor) is set.
10	MF	Multicast frame When set, indicates that the frame has a multicast address. This bit is valid only when RDES0[8] (last descriptor) is set.
9	FS	First descriptor When set, indicates that this is the first descriptor of a frame.
8	LS	Last descriptor When set, indicates that this is the last descriptor of a frame.

***Table 12-3* • Receive Descriptor 0 (RDES0) Bit Functions (continued)**

Bit	Name	Function
7	TL	Frame too long When set, indicates that a current frame is longer than maximum size of 1,518 bytes, as specified by 802.3. TL (frame too long) in the receive descriptor has been set when the received frame is longer than 1,518 bytes. This flag is valid in all receive descriptors when multiple descriptors are used for one frame.
6	CS	Collision seen When set, indicates that a late collision was seen (collision after 64 bytes following SFD). This bit is valid only when RDES0[8] (last descriptor) is set.
5	FT	Frame type When set, indicates that the frame has a length field larger than 1,500 (Ethernet-type frame). When cleared, indicates an 802.3-type frame. This bit is valid only when RDES0[8] (last descriptor) is set. Additionally, FT is invalid for runt frames shorter than 14 bytes.
3	RE	Report on RMII error When set, indicates that an error has been detected by a physical layer chip connected through the RMII interface. This bit is valid only when RDES0[8] (last descriptor) is set.
2	DB	Dribbling bit When set, indicates that the frame was not byte-aligned. This bit is valid only when RDES0[8] (last descriptor) is set.
1	CE	CRC error When set, indicates that a CRC error has occurred in the received frame. This bit is valid only when RDES0[8] (last descriptor) is set. Additionally, CE is not valid when the received frame is a runt frame.
0	ZERO	This bit is reset for frames with a legal length.

***Table 12-4* • CONTROL and COUNT (RDES1) Bit Functions**

Bits	Name	Function
31:26		Reserved
25	RER	Receive end of ring When set, indicates that this is the last descriptor in the receive descriptor ring. Ethernet MAC returns to the first descriptor in the ring, as specified by CSR3 (start of receive list address).
24	RCH	Second address chained When set, indicates that the second buffer's address points to the next descriptor and not to the data buffer. *Note: RER takes precedence over RCH.*
21:11	RBS2	Buffer 2 size Indicates the size, in bytes, of memory space used by the second data buffer. This number must be a multiple of four. If it is 0, Ethernet MAC ignores the second data buffer and fetches the next data descriptor. This number is valid only when RDES1[24] (second address chained) is cleared.
10:0	RBS1	Buffer 1 size Indicates the size, in bytes, of memory space used by the first data buffer. This number must be a multiple of four. If it is 0, Ethernet MAC ignores the first data buffer and uses the second data buffer.

***Table 12-5* • RBA1 (RDES2) Bit Functions**

Bits	Name	Function
31:0	RBA1	Receive buffer 1 address Indicates the length, in bytes, of memory allocated for the first receive buffer. This number must be 32-bit word aligned (RDES2[1:0] = 0b00).

***Table 12-6* • RBA2 (RDES3) Bit Functions**

Bits	Name	Function
31:0	RBA2	Receive buffer 2 address Indicates the length, in bytes, of memory allocated for the second receive buffer. This number must be 32-bit word aligned (RDES3[1:0] = 0b00).

Table 12-7 to Table 12-11 on page 185 give bit descriptions and functions for the transmit descriptors.

***Table 12-7* • Transmit Descriptors (TDESx)**

TDES0	OWN	STATUS		
TDES1	CONTROL		TBS2	TBS1
TDES2	TBA1			
TDES3	TBA2			

Note: The TDESx descriptors reside in receive data memory. They can be defined and addressed under software control.

***Table 12-8* • Transmit Descriptor (TDES0) Bit Functions**

Bits	Name	Function
31	OWN	Ownership bit 1 – Ethernet MAC owns the descriptor. 0 – The host owns the descriptor. Ethernet MAC will clear this bit when it completes a current frame transmission or when the data buffers associated with a given descriptor are empty.
30:16		Reserved
15	ES	Error summary This bit is a logical OR of the following bits: TDES0[1] – Underflow error TDES0[8] – Excessive collision error TDES0[9] – Late collision TDES0[10] – No carrier TDES0[11] – Loss of carrier This bit is valid only when TDES1[30] (last descriptor) is set.
14:12	Reserved	
11	LO	Loss of carrier When set, indicates a loss of the carrier during a transmission. This bit is valid only when TDES1[30] (last descriptor) is set.
10	NC	No carrier When set, indicates that the carrier was not asserted by an external transceiver during the transmission. This bit is valid only when TDES1[30] (last descriptor) is set.
9	LC	Late collision When set, indicates that a collision was detected after transmitting 64 bytes. This bit is not valid when TDES0[1] (underflow error) is set. This bit is valid only when TDES1[30] (last descriptor) is set.
8	EC	Excessive collisions When set, indicates that the transmission was aborted after 16 retries. This bit is valid only when TDES1[30] (last descriptor) is set.
7		Reserved
6:3	CC	Collision count This field indicates the number of collisions that occurred before the end of a frame transmission. This value is not valid when TDES0[8] (excessive collisions bit) is set. This bit is valid only when TDES1[30] (last descriptor) is set.
1	UF	Underflow error When set, indicates that the FIFO was empty during the frame transmission. This bit is valid only when TDES1[30] (last descriptor) is set.
0	DE	Deferred When set, indicates that the frame was deferred before transmission. Deferring occurs if the carrier is detected when the transmission is ready to start. This bit is valid only when TDES1[30] (last descriptor) is set.

***Table 12-9* • Control (TDES1) Bit Functions**

Bits	Name	Function
31	IC	Interrupt on completion Setting this flag instructs Ethernet MAC to set CSR5[0] (transmit interrupt) immediately after processing a current frame. This bit is valid when TDES1[30] (last descriptor) is set or for a setup packet.
30	LS	Last descriptor When set, indicates the last descriptor of the frame.
29	FS	First descriptor When set, indicates the first descriptor of the frame.
28	FT1	Filtering type This bit, together with TDES0[22] (FT0), controls a current filtering mode. This bit is valid only for the setup frames.
27	SET	Setup packet When set, indicates that this is a setup frame descriptor.
26	AC	Add CRC disable When set, Ethernet MAC does not append the CRC value at the end of the frame. The exception is when the frame is shorter than 64 bytes and automatic byte padding is enabled. In that case, the CRC field is added, despite the state of the AC flag.
25	TER	Transmit end of ring When set, indicates the last descriptor in the descriptor ring.
24	TCH	Second address chained When set, indicates that the second descriptor's address points to the next descriptor and not to the data buffer. This bit is valid only when TDES1[25] (transmit end of ring) is reset.
23	DPD	Disabled padding When set, automatic byte padding is disabled. Ethernet MAC normally appends the PAD field after the INFO field when the size of an actual frame is less than 64 bytes. After padding bytes, the CRC field is also inserted, despite the state of the AC flag. When DPD is set, no padding bytes are appended.
22	FT0	Filtering type This bit, together with TDES0[28] (FT1), controls the current filtering mode. This bit is valid only when the TDES1[27] (SET) bit is set.
21:11	TBS2	Buffer 2 size Indicates the size, in bytes, of memory space used by the second data buffer. If it is zero, Ethernet MAC ignores the second data buffer and fetches the next data descriptor. This bit is valid only when TDES1[24] (second address chained) is cleared.
10:0	TBS1	Buffer 1 size Indicates the size, in bytes, of memory space used by the first data buffer. If it is 0, Ethernet MAC ignores the first data buffer and uses the second data buffer.

***Table 12-10* • TBA1 (TDES2) Bit Functions**

Bits	Name	Function
31:0	TBA1	Transmit buffer 1 address Contains the address of the first data buffer. For the setup frame, this address must be 32-bit word aligned (TDES3[1:0] = 0b00). In all other cases, there are no restrictions on buffer alignment.

***Table 12-11* • TBA2 (TDES3) Bit Functions**

Bits	Name	Function
31:0	TBA2	Transmit buffer 2 address Contains the address of the second data buffer. There are no restrictions on buffer alignment.

MAC Address and Setup Frames

The setup frames define addresses that are used for the receive address filtering process. These frames are never transmitted on the Ethernet connection. They are used to fill the address filtering RAM.

Following are the requirements for the setup frame:

- A valid setup frame must be exactly 192 bytes long and must be allocated in a single buffer that is 32-bit word aligned.
- TDES1[27] (setup frame indicator) must be set.
- Both TDES1[29] (first descriptor) and TDES1[30] (last descriptor) must be cleared.

The FT1 and FT0 bits of the setup frame define the current filtering mode.

Table 12-12 lists all possible combinations. Table 12-13 on page 186 shows the setup frame buffer format for perfect filtering modes. Table 12-14 on page 186 shows the setup frame buffer for imperfect filtering modes. The setup should be sent to Ethernet MAC when Ethernet MAC is in stop mode. A setup frame with more than 192 bytes can be written into the address filtering RAM to initialize its contents, but only the first 192 bytes constitute the address filtering operation. While writing the setup frame buffer in the host memory, the buffer size must be twice the size of the setup frame buffer.

***Table 12-12* • Filtering Type Selection**

FT1	FT0	Description
0	0	Perfect filtering mode. Setup frame buffer is interpreted as a set of sixteen 48-bit physical addresses.
0	1	Hash filtering mode. Setup frame buffer contains a 512-bit hash table plus a single 48-bit physical address.
1	0	Inverse filtering mode. Setup frame buffer is interpreted as a set of sixteen 48-bit physical addresses.
1	1	Hash-only filtering mode. Setup frame buffer is interpreted as a 512-bit hash table.

***Table 12-13* • Perfect Filtering Setup Frame Buffer**

Byte Number	Data Bits [31:16]	Data Bits [15:0]
1:0	{Physical address [39:32],physical address [47:40]}	
3:2	{Physical address [23:16],physical address [31:24]}	
5:4	{Physical address [7:0],physical address [15:8]}	
15:12	xxxxxxxxxxxxxxxx	Physical address 1 [15:0]
19:16	xxxxxxxxxxxxxxxx	Physical address 1 [31:16]
23:20	xxxxxxxxxxxxxxxx	Physical address 1 [47:32]
⋮	⋮	⋮
171:168	xxxxxxxxxxxxxxxx	Physical address 14 [15:0]
175:172	xxxxxxxxxxxxxxxx	Physical address 14 [31:16]
179:176	xxxxxxxxxxxxxxxx	Physical address 14 [47:32]
183:180	xxxxxxxxxxxxxxxx	Physical address 15 [15:0]
187:184	xxxxxxxxxxxxxxxx	Physical address 15 [31:16]
191:188	xxxxxxxxxxxxxxxx	Physical address 15 [47:32]

***Table 12-14* • Hash Table Setup Frame Buffer Format**

Byte Number	Data Bits [31:16]	Data Bits [15:0]
3:0	xxxxxxxxxxxxxxxx	Hash filter [15:0]
7:4	xxxxxxxxxxxxxxxx	Hash filter [31:16]
11:8	xxxxxxxxxxxxxxxx	Hash filter [47:32]
⋮	⋮	⋮
123:121	xxxxxxxxxxxxxxxx	Hash filter [495:480]
127:124	xxxxxxxxxxxxxxxx	Hash filter [511:496]
131:128	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
135:132	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
⋮	⋮	
159:156	xxxxxxxxxxxxxxxx	Physical address [15:0]
163:160	xxxxxxxxxxxxxxxx	Physical address [31:16]
167:164	xxxxxxxxxxxxxxxx	Physical address [47:32]
171:168	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
175:172	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
⋮	⋮	
183:180	xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx
187:184	xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx
191:188	xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx

Internal Operation

DMA Controller

The DMA is used to control a data flow between the host and Ethernet MAC.

The DMA services the following types of requests from the Ethernet MAC transmit and receive processes:

- Transmit request:
  - Descriptor fetch
  - Descriptor closing
  - Setup packet processing
  - Data transfer from host buffer to transmit FIFO
- Receive request:
  - Descriptor fetch
  - Descriptor closing
  - Data transfer from receive FIFO to host buffer

The key task for the DMA is to perform an arbitration between the receive and transmit processes. Two arbitration schemes are possible according to the CSR0[1] bit:

1. Value 1: Round-robin arbitration scheme in which receive and transmit processes have equal priorities
2. Value 0: The receive process has priority over the transmit process unless transmission is in progress. In this case, the following rules apply:
   - The transmit process request should be serviced by the DMA between two consecutive receive transfers.
   - The receive process request should be serviced by the DMA between two consecutive transmit transfers.

Transfers between the host and Ethernet MAC performed by the DMA component are 32-bit data transfers or burst transfers.

In the case of data buffers, the burst length is defined by CSR0[13:8] (PBL), which is set to zero, and the transfer ends when the transmit FIFOs are full or the receive FIFOs are empty.

Transmit Process

The transmit process can operate in one of three modes: running, stopped, or suspended. After a software or hardware reset, or after a stop transmit command, the transmit process is in a stopped state. The transmit process can leave a stopped state only after the start transmit command.

When in a running state, the transmit process performs descriptor/buffer processing. When operating in a suspended or stopped state, the transmit process retains the position of the next descriptor; that is, the address of the descriptor following the last descriptor being closed. After entering a running state, that position is used for the next descriptor fetch. The only exception is when the host writes the transmit descriptor base address register (CSR4). In that case, the descriptor address is reset and the fetch is directed to the first position in the list. Before writing to CSR4 the MAC must be in a stopped state.

The transmit process remains running until one of the following events occur:

- The hardware or software reset is issued. Setting the CSR0[0] (SWR) bit can perform the software reset. After the reset, all the internal registers return to their default states. The current descriptor's position in the transmit descriptor list is lost.
- A stop transmit command is issued by the host. This can be performed by writing 0 to the CSR6[13] (ST) bit. The current descriptor's position is retained.
- The descriptor owned by the host is found. The current descriptor's position is retained.
- The transmit FIFO underflow error is detected. An underflow error is generated when the transmit FIFO is empty during the transmission of the frame. When it occurs, the transmit process enters a

suspended state. Transmit automatic polling is internally disabled, even if it is enabled by the host by writing the TAP bits. The current descriptor's position is retained.

Leaving a suspended state is possible in one of the following situations:

- A transmit poll demand command is issued. This can be performed by writing CSR1 with a nonzero value. The transmit poll demand command can also be generated automatically when transmit automatic polling is enabled. Transmit automatic polling is enabled only if the CSR0[19:17] (TAP) bits are written with a nonzero value and when there was no underflow error prior to entering the suspended state.
- A stop transmit command is issued by the host. This can be performed by writing 0 to the CSR6[13] (ST) bit. The current descriptor's position is retained.

The events for the transmit process typically happen in the following order:

1. The host sets up CSR registers for the operational mode, interrupts, etc.
2. The host sets up transmit descriptors/data in the shared RAM.
3. The host sends the transmit start command.
4. Ethernet MAC starts to fetch the transmit descriptors.
5. Ethernet MAC transfers the transmit data to Transmit Data RAM from the shared RAM.
6. Ethernet MAC starts to transmit data on RMII.

A typical data flow for the transmit process is illustrated in Figure 12-6.

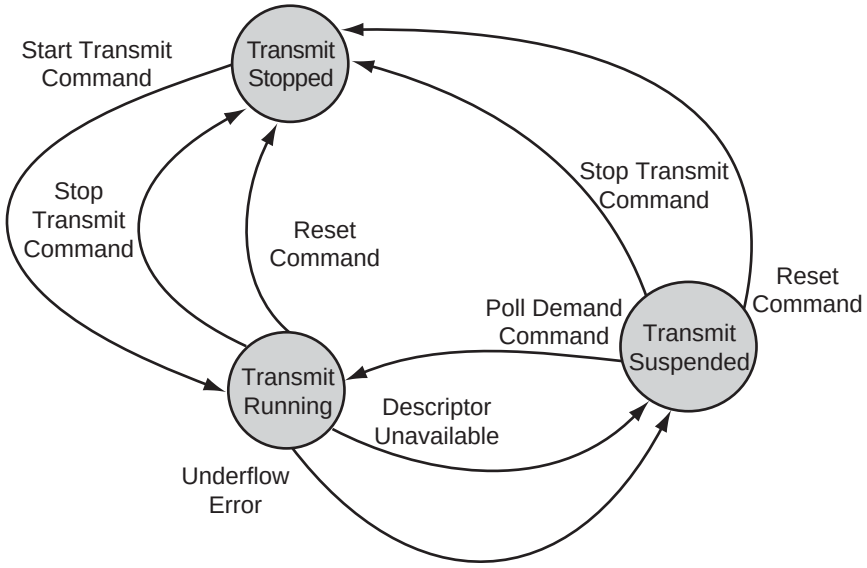


***Figure 12-6* • Transmit Process Transitions**

Receive Process

The receive process can operate in one of three modes: running, stopped, or suspended. After a software or hardware reset, or after a stop receive command, the receive process is in the stopped state. The receive process can leave a stopped state only after a start receive command.

In the running state, the receiver performs descriptor/buffer processing. In the running state, the receiver fetches from the receive descriptor list. It performs this fetch regardless of whether there is any frame on the link. When there is no frame pending, the receive process reads the descriptor and simply waits for the frames. When a valid frame is recognized, the receive process starts to fill the memory buffers pointed to by the current descriptor. When the frame ends, or when the memory buffers are completely filled, the current frame descriptor is closed (ownership bit cleared). Immediately, the next descriptor on the list is fetched in the same manner, and so on.

When operating in a suspended or stopped state, the receive process retains the position of the next descriptor (the address of the descriptor following the last descriptor that was closed). After entering a running state, the retained position is used for the next descriptor fetch. The only exception is when the host writes the receive descriptor base address register (CSR3). In that case, the descriptor address is reset and the fetch is pointed to the first position in the list. Before writing to CSR3, the MAC must be in a stopped state.

The receive process runs until one of the following events occurs:

- A hardware or software reset is issued by the host. A software reset can be performed by setting the CSR0[0] (SWR) bit. After reset, all internal registers return to their default states. The current descriptor's position in the receive descriptor list is lost.
- A stop receive command is issued by the host. This can be performed by writing 0 to the CSR6[1] (SR) bit. The current descriptor's position is retained.
- The descriptor owned by the host is found by Ethernet MAC during the descriptor fetch. The current descriptor's position is retained.

Leaving a suspended state is possible in one of the following situations:

- A receive poll command is issued by the host. This can be performed by writing CSR2 with a nonzero value.
- A new frame is detected by Ethernet MAC on a receive link.
- A stop receive command is issued by the host. This can be performed by writing 0 to the CSR6[1] (SR) bit. The current descriptor's position is retained.

The receive state machine goes into stopped state after the current frame is done if a STOP RECEIVE command is given. It does not go in to a stopped state immediately.

A typical data flow in a receive process is illustrated in Figure 12-7.

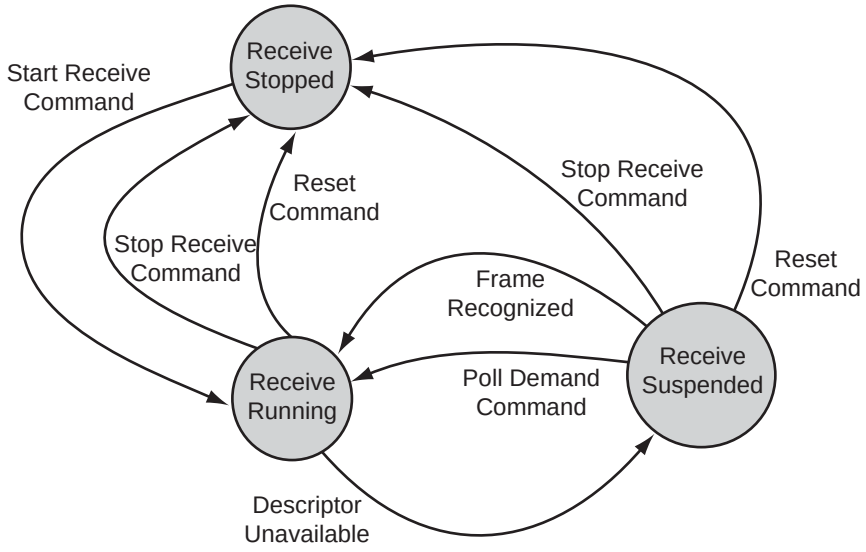


Figure 12-7 • **Receive Process Transmissions**

The events for the receive process typically happen in the following order:

1. The host sets up CSR registers for the operational mode, interrupts, etc.
2. The host sets up receive descriptors in the shared RAM.
3. The host sends the receive start command.
4. Ethernet MAC starts to fetch the transmit descriptors.
5. Ethernet MAC waits for receive data on RMII.
6. Ethernet MAC transfers received data to the Receive Data RAM.
7. Ethernet MAC transfers received data to shared RAM from Receive Data RAM.

Interrupt Controller

The interrupt controller uses three internal Control and Status registers: CSR5, CSR7, and CSR11. CSR5 contains the Ethernet MAC status information. It has 10 bits that can trigger an interrupt. These bits are collected in two groups: normal interrupts and abnormal interrupts. Each group has its own summary bit, NIS and AIS, respectively. The NIS and AIS bits directly control the MAC_INT output port of Ethernet MAC (INTISR[5] to the Cortex-M3 NVIC). Every status bit in CSR5 that can source an interrupt can be individually masked by writing an appropriate value to CSR7 (Interrupt Enable register).

Additionally, an interrupt mitigation mechanism is provided for reducing CPU usage in servicing interrupts. Interrupt mitigation is controlled via CSR11. There are separate interrupt mitigation control blocks for the transmit and receive interrupts. Both of these blocks consist of a 4-bit frame counter and a 4-bit timer. The operation of these blocks is similar for the receive and transmit processes. After the end of a successful receive or transmission operation, an appropriate counter is decremented and the timer starts to count down if it has not already started. An interrupt is triggered when either the counter or the timer reaches a zero value. This allows Ethernet MAC to generate a single interrupt for a few received/transmitted frames or after a specified time since the last successful receive/transmit operation.

It is possible to omit transmit interrupt mitigation for one particular frame by setting the Interrupt on Completion (IC) bit in the last descriptor of the frame. If the IC bit is set, Ethernet MAC sets the transmit

interrupt immediately after the frame has been transmitted. The interrupt scheme is shown in Figure 12-8.

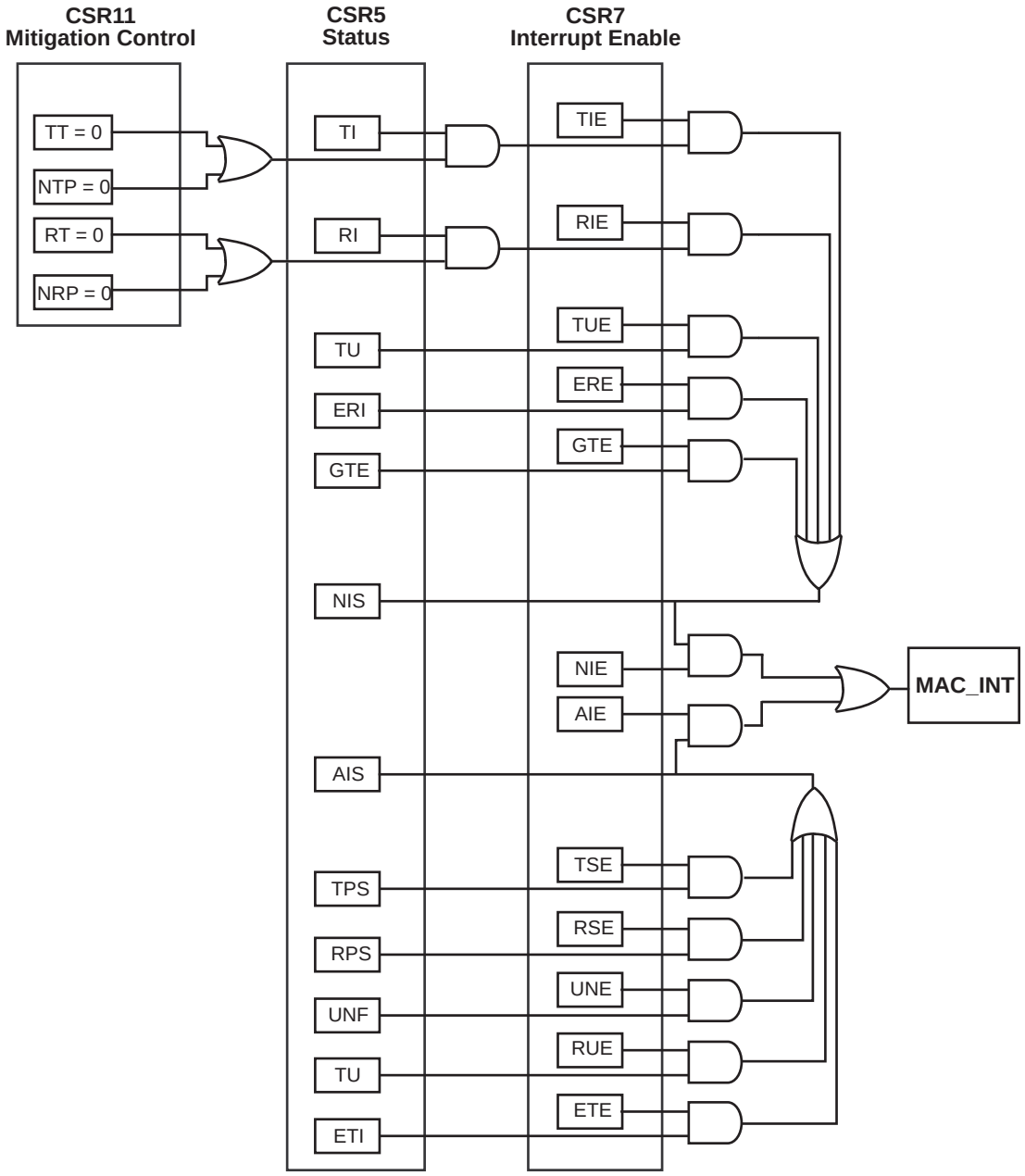


***Figure 12-8* • Interrupt Scheme**

General-Purpose Timer

Ethernet MAC includes a 16-bit general-purpose timer to simplify time interval calculation by an external host. The timer operates synchronously with the transmit clock CLKT generated by the PHY device. This gives the host the possibility of measuring time intervals based on actual Ethernet bit time.

The timer can operate in one-shot mode or continuous mode. In one-shot mode, the timer stops after reaching a zero value; in continuous mode, it is automatically reloaded and continues counting down after reaching a zero value.

The actual count value can be tested with an accuracy of ±1 bit by reading CSR11[15:0]. When writing CSR11[15:0], the data is stored in the internal reload register. The timer is immediately reloaded and starts to count down.

Frame Format

Ethernet MAC supports the Ethernet frame format shown in Table 12-15 (B indicates bytes). The standard Ethernet frames (DIX Ethernet), as well as IEEE 802.3 frames, are accepted.

***Table 12-15* • Frame Field Usage**

Field	Width (bytes)	Transmit Operation	Receive Operation
PREAMBLE	7	Generated by Ethernet MAC.	Stripped from received data. Not required for proper operation.
SFD	1	Generated by Ethernet MAC.	Stripped from received data.
DA	6	Supplied by host.	Checked by Ethernet MAC according to current address filtering mode and passed to host.
SA	6	Supplied by host.	Passed to host.
LENGTH/ TYPE	6	Supplied by host.	Passed to host.
DATA	0–1500	Supplied by host.	Passed to host.
PAD	0–46	Generated by Ethernet MAC when CSR[23] (DPD) bit is cleared and data supplied by host is less than 64 bytes.	Passed to host.
FCS	4	Generated by Ethernet MAC when CSR[26] bit is cleared.	Checked by Ethernet MAC and passed to host.

Collision Handling

If a collision is detected before the end of the PREAMBLE/ SFD, Ethernet MAC completes the PREAMBLE/SFD, transmits the JAM sequence, and initiates a backoff computation. If a collision is detected after the transmission of the PREAMBLE and SFD, but prior to 512 bits being transmitted, Ethernet MAC immediately aborts the transmission, transmits the JAM sequence, and then initiates a backoff. If a collision is detected after 512 bits have been transmitted, the collision is termed a late collision. Ethernet MAC aborts the transmission and appends the JAM sequence. The transmit message is flushed from the FIFO. Ethernet MAC does not initiate a backoff and does not attempt to retransmit the frame when a late collision is detected.

Ethernet MAC uses a truncated binary exponential backoff algorithm for backoff computing, as defined in the IEEE 802.3 standard and outlined in Figure 12-9. Backoff processing is performed only in half-duplex mode. In full-duplex mode, collision detection is disabled.

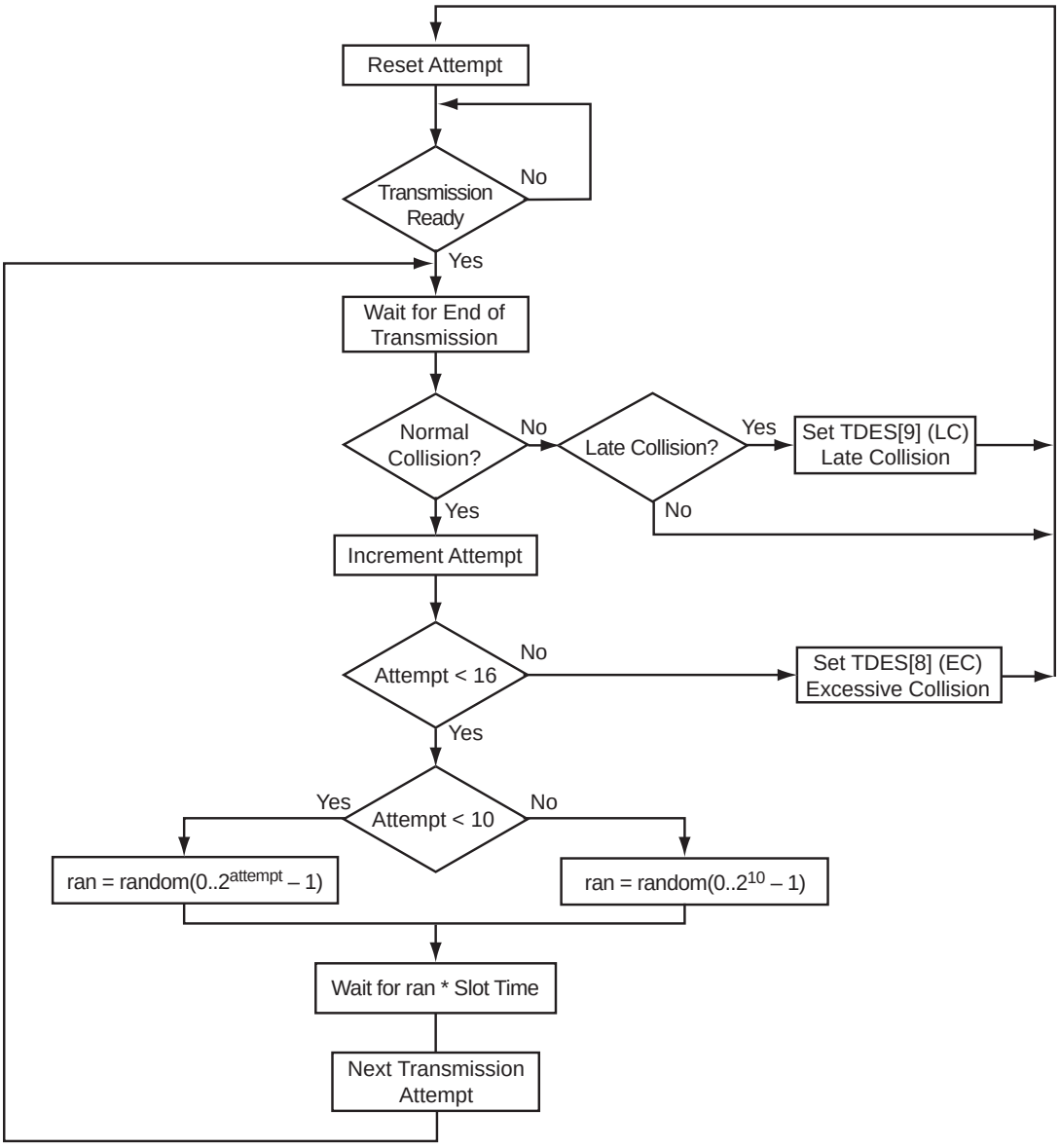


***Figure 12-9* • Backoff Process Algorithms**

Deferring

The deferment algorithm is implemented per the 802.3 specification and outlined in Figure 12-10. The inter-frame gap (IFG) timer starts to count whenever the link is not idle. If activity on the link is detected during the first 60 bit times of the IFG timer, the timer is reset and restarted once activity has stopped. During the final 36 bit times of the IFG timer, the link activity is ignored.

Carrier sensing is performed only when operating in half-duplex mode. In full-duplex mode, the state of the CRS input is ignored.

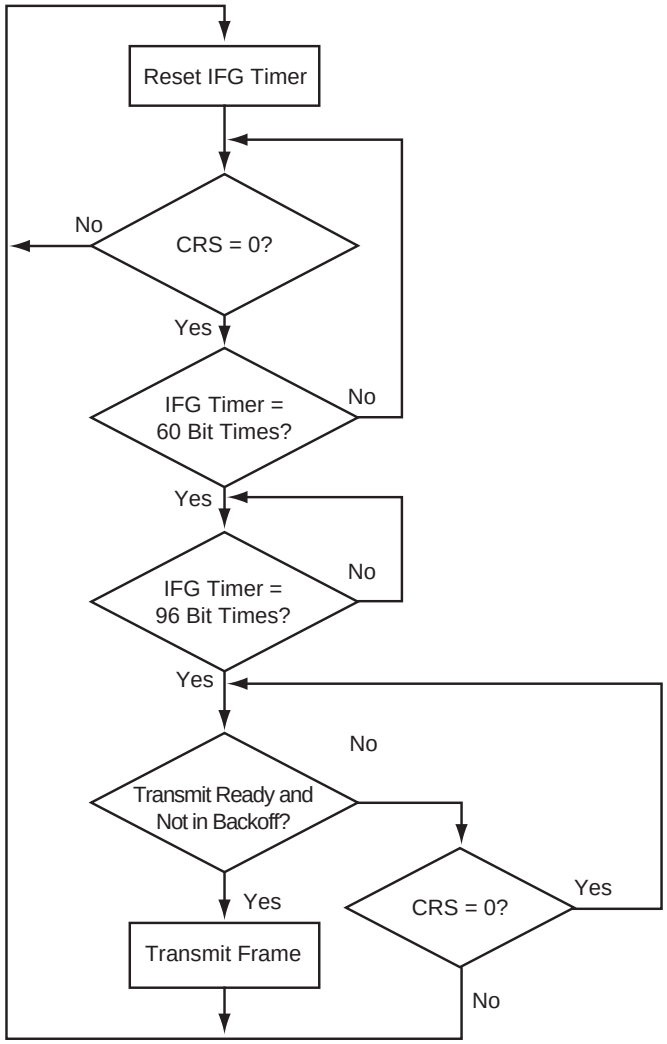


***Figure 12-10* • Deferment Process Algorithms**

Receive Address Filtering

There are three kinds of addresses on the LAN: the unicast addresses, the multicast addresses, and the broadcast addresses. If the first bit of the address (IG bit) is 0, the frame is unicast—dedicated to a single station. If the first bit is 1, the frame is multicast—destined for a group of stations. If the address field contains all ones, the frame is broadcast and is received by all stations on the LAN.

When Ethernet MAC operates in perfect filtering mode, all frames are checked against the addresses in the address filtering RAM. The unicast, multicast, and broadcast frames are treated in the same manner.

When Ethernet MAC operates in the imperfect filtering mode, the frames with the unicast addresses are checked against a single physical address. The multicast frames are checked using the 512-bit hash table. To receive the broadcast frame, the hash table bit corresponding to the broadcast address CRC value must be set. Ethernet MAC applies the standard Ethernet CRC function to the first six bytes of the frame that contains a destination address. The least significant nine bits of the CRC value are used to index the table. If the indexed bit is set, the frame is accepted. If this bit is cleared, the frame is rejected. The algorithm is shown in Figure 12-11.

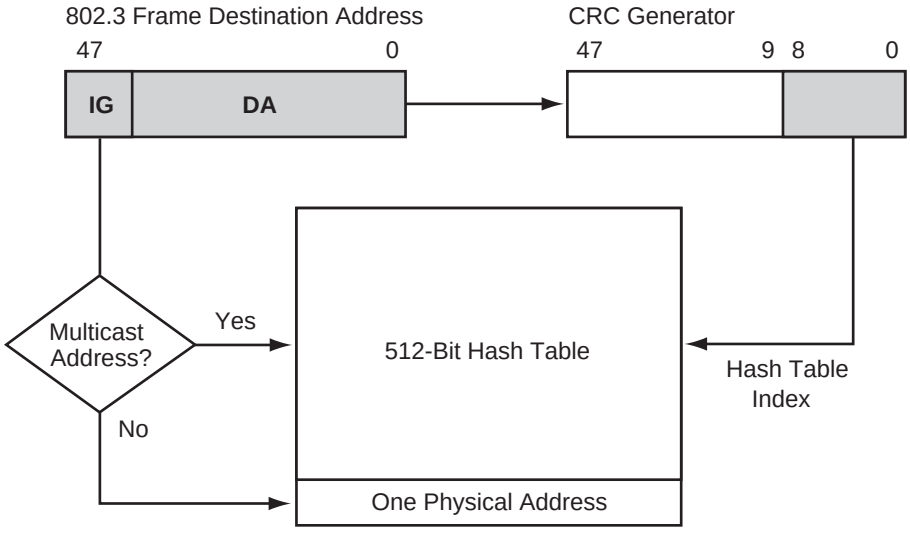


***Figure 12-11* • Filtering with One Physical Address and the Hash Table**

It is important that one bit in the hash table corresponds to many Ethernet addresses. Therefore, it is possible that some frames may be accepted by Ethernet MAC, even if they are not intended to be received. This is because some frames that should not have been received have addresses that hash to the same bit in the table as one of the proper addresses. The software should perform additional address filtering to reject all such frames.

Software Interface

Ethernet MAC Control and Status Register Addressing

The Control and Status registers are located physically inside Ethernet MAC and can be accessed directly by the Cortex-M3 processor via a 32-bit interface. All the CSRs are 32 bits long and quadword-aligned. The address bus of the CSR interface is 8 bit wide, and only bits 6:0 of the location code shown in Table 12-16 are used to decode the CSR register address.

Ethernet MAC Base Address: 0x40003000

***Table 12-16* • 12 CSR Locations**

Register Name	Address	R/W	Reset Value	Description
CSR0	0x40003000	R/W	0xFE000000	Bus mode
CSR1	0x40003008	W	0	Transmit poll demand
CSR2	0x40003010	W	0	Receive poll demand
CSR3	0x40003018	R/W	0xFFFFFFFF	Receive list base address
CSR4	0x40003020	R/W	0xFFFFFFFF	Transmit list base address
CSR5	0x40003028	R/W	0xF0000000	Status and control
CSR6	0x40003030	R/W	0x32000040	Operation mode
CSR7	0x40003038	R/W	0xF3FE0000	Interrupt enable
CSR8	0x40003040	R/W	0xE0000000	Missed frames and overflow counters
CSR9	0x40003048	R/W	0xFFF483FB	RMII management
CSR10	0x40003050	N/A	0	Reserved
CSR11	0x40003058	R/W	0xFFFE0000	Timer and interrupt mitigation control

Note: CSR9 bits 19 and 2 reset values are dependent on the MDI and SDI inputs. The above assumes MDI is high and SDI is low.

CSR Definitions

Bus Mode Register (CSR0)

Table 12-17 • **Bus Mode Register (CSR0)**

Bits 31:24								
Bits 23:16			SPD	DBO	TAP			
Bits 15:8			PBL					
Bits 7:0	BLE	DSL					BAR	SWR

Note: The CSR0 register has unimplemented bits (shaded). If these bits are read, they will return a predefined value as shown in Table 12-18. Writing to these bits has no effect.

Table 12-18 • **CSR0**

Bit	Name	R/W	Reset Value	Function
31:22		N/A	0b1111111000	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	SPD	W	0	Clock frequency selection This bit selects the clock frequency for CLKT and CLKR. When this bit is set to 0, CLKT and CLKR are 2.5 MHz. When this bit is set to 1, CLKT and CLKR are 25 MHz.
20	DBO	W	0	Descriptor byte ordering mode: 1 – Big-endian mode used for data descriptors 0 – Little-endian mode used for data descriptors
19:17	TAP	R/W	0	Transmit automatic polling If TAP is written with a nonzero value, Ethernet MAC performs an automatic transmit descriptor polling when operating in suspended state. When the descriptor is available, the transmit process goes into running state. When the descriptor is marked as owned by the host, the transmit process remains suspended. The poll is always performed at the current transmit descriptor list position. The time interval between two consecutive polls is shown in Table 12-19 on page 198.
16:14		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

***Table 12-18* • CSR0 (continued)**

Bit	Name	R/W	Reset Value	Function
13:8	PBL	R/W	0	Programmable burst length Specifies the maximum number of words (32-bit) that can be transferred within one DMA transaction. This is tied to value 0 and the bursts are limited only by the internal FIFO's threshold levels. Note that PBL is valid only for the data buffers. Store and forward operation packet size is limited to the transmit buffer size minus a space for an additional DMA burst. The DMA burst length is equivalent to 64 bytes. If store and forward operations are requested for packet sizes that are greater than this limit, the core will enter a lockup situation because it is unable to complete the store part of the store and forward operation. The maximum store and forward size is given in EQ 1. 2, 048 – (4 × 64) = 1,792 (transmit FIFO depth is 2,048 bytes) *EQ 1*
7	BLE	W	0	Big/little endian Selects the byte-ordering mode used by the data buffers. 1 – Big-endian mode used for the data buffers 0 – Little-endian mode used for the data buffers
6:2	DSL	R/W	0	Descriptor skip length Specifies the number of long words between two consecutive descriptors in a ring structure.
1	BAR	R/W	0	Bus arbitration scheme 1 – Transmit and receive processes have equal priority to access the bus. 0 – Intelligent arbitration, where the receive process has priority over the transmit process.
0	SWR	R/W	0	Software reset Setting this bit resets all internal flip-flops. The processor should write a 1 to this bit and then wait until a read returns a 0, indicating that the reset has completed. This bit will remain set for several clock cycles.

***Table 12-19* • Transmit Automatic Polling Intervals**

CSR0[19:17]	10 Mbps	100 Mbps
000	TAP disabled	TAP disabled
001	825.6 µs	82.56 µs
010	2,476.8 µs	247.68 µs
011	5,779.2 µs	577.92 µs
100	51.6 µs	5.16 µs
101	103.2 µs	10.32 µs
110	154.8 µs	15.48 µs
111	412.8 µs	41.28 µs

Transmit Poll Demand Register (CSR1)

Table 12-20 • Transmit Poll Demand Register (CSR1

Bits 31:24	TPD[31:24]
Bits 23:16	TPD[23:16]
Bits 15:8	TPD[15:8]
Bits 7:0	TPD[7:0]

Table 12-21 • CSR1

Bit	Name	R/W	Reset Value	Function
31:0	TPD	W	0	Writing this field with any value instructs Ethernet MAC to check for frames to be transmitted. This operation is valid only when the transmit process is suspended. If no descriptor is available, the transmit process remains suspended. When the descriptor is available, the transmit process goes into the running state.

Receive Poll Demand Register (CSR2)

Table 12-22 • Receive Poll Demand Register (CSR2)

Bits 31:24	RPD[31:24]
Bits 23:16	RPD[23:16]
Bits 15:8	RPD[15:8]
Bits 7:0	RPD[7:0]

Table 12-23 • CSR2

Bit	Name	R/W	Reset Value	Function
31:0	RPD	W	0	Writing this field with any value instructs Ethernet MAC to check for receive descriptors to be acquired. This operation is valid only when the receive process is suspended. If no descriptor is available, the receive process remains suspended. When the descriptor is available, the receive process goes into the running state.

Receive Descriptor List Base Address Register (CSR3)

Table 12-24 • Receive Descriptor List Base Address Register (CSR3)

Bits 31:24	RLA[31:24]
Bits 23:16	RLA[23:16]
Bits 15:8	RLA[15:8]
Bits 7:0	RLA[7:0]

Table 12-25 • CSR3

Bit	Name	R/W	Reset Value	Function
31:0	RLA	R/W	0xFFFFFFFF	Start of the receive list address Contains the address of the first descriptor in a receive descriptor list. This address must be 32-bit word aligned (RLA[1:0] = 0).

Transmit Descriptor List Base Address Register (CSR4)

Table 12-26 • Transmit Descriptor List Base Address Register (CSR4)

Bits 31:24	TLA[31:24]
Bits 23:16	TLA[23:16]
Bits 15:8	TLA[15:8]
Bits 7:0	TLA[7:0]

Table 12-27 • CSR4

Bit	Name	R/W	Reset Value	Function
31:0	TLA	R/W	0xFFFFFFFF	Start of the transmit list address Contains the address of the first descriptor in a transmit descriptor list. This address must be 32-bit word aligned (TLA[1:0] = 0).

Status and Control Register (CSR5)

Table 12-28 • Status and Control Register (CSR5)

Bits 31:24								
Bits 23:16		TS			RS			NIS
Bits 15:8	AIS	ERI			GTE	ETI		RPS
Bits 7:0	BLE	DSL	UNF			TU	TPS	TI

Note: The CSR5 register has unimplemented bits (shaded). If these bits are read, they will return a predefined value, as shown in Table 12-29. Writing to these bits has no effect.

Table 12-29 • CSR5

Bit	Name	R/W	Reset Value	Function
31:23		N/A	0b111100000	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:20	TS	R	0	Transmit process state (read-only) Indicates the current state of a transmit process: 000 – Stopped; RESET or STOP TRANSMIT command issued. 001 – Running; fetching the transmit descriptor. 010 – Running; waiting for end of transmission. 011 – Running; transferring data buffer from host memory to FIFO. 100 – Reserved 101 – Running; set up packet 110 – Suspended; FIFO underflow or unavailable descriptor. 111 – Running; closing transmit descriptor.
19:17	RS	R	0	Receive process state (read-only) Indicates the current state of a receive process: 000 – Stopped; RESET or STOP RECEIVE command issued. 001 – Running; fetching the receive descriptor. 010 – Running; waiting for the end-of-receive packet before prefetch of the next descriptor. 011 – Running; waiting for the receive packet. 100 – Suspended; unavailable receive buffer 101 – Running; closing the receive descriptor. 110 – Reserved 111 – Running; transferring data from FIFO to host memory.
16	NIS	R/W	0	Normal interrupt summary This bit is a logical OR of the following bits: CSR5[0] – Transmit interrupt CSR5[2] – Transmit buffer unavailable CSR5[6] – Receive interrupt CSR5[11] – General-purpose timer overflow CSR5[14] – Early receive interrupt Only the unmasked bits affect the normal interrupt summary bit. The user can clear this bit by writing a 1. Writing a 0 has no effect.

***Table 12-29* • CSR5 (continued)**

Bit	Name	R/W	Reset Value	Function
15	AIS	R/W	0	Abnormal interrupt summary This bit is a logical OR of the following bits: CSR5[1] – Transmit process stopped CSR5[5] – Transmit underflow CSR5[7] – Receive buffer unavailable CSR5[8] – Receive process stopped CSR5[10] – Early transmit interrupt Only the unmasked bits affect the abnormal interrupt summary bit. The user can clear this bit by writing a 1. Writing a 0 has no effect.
14	ERI	R/W	0	Early receive interrupt Set when Ethernet MAC fills the data buffers of the first descriptor. The user can clear this bit by writing a 1. Writing a 0 has no effect.
13:12		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	GTE	R/W	0	General-purpose timer expiration Gets set when the general-purpose timer reaches zero value. The user can clear this bit by writing a 1. Writing a 0 has no effect.
10	ETI	R/W	0	Early transmit interrupt Indicates that the packet to be transmitted was fully transferred into the FIFO. The user can clear this bit by writing a 1. Writing a 0 has no effect.
9		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	RPS	R/W	0	Receive process stopped RPS is set when a receive process enters a stopped state. The user can clear this bit by writing a 1. Writing a 0 has no effect.
7	RU	R/W	0	Receive buffer unavailable When set, indicates that the next receive descriptor is owned by the host and is unavailable for Ethernet MAC. When RU is set, Ethernet MAC enters a suspended state and returns to receive descriptor processing when the host changes ownership of the descriptor. Either a receive-poll demand command is issued or a new frame is recognized by Ethernet MAC. The user can clear this bit by writing a 1. Writing a 0 has no effect.
6	RI	R/W	0	Receive interrupt Indicates the end of a frame receive. The complete frame has been transferred into the receive buffers. Assertion of the RI bit can be delayed using the receive interrupt mitigation counter/timer (CSR11[19:17]–NRP /CSR11[23:20] – RT). The user can clear this bit by writing a 1. Writing a 0 has no effect.

***Table 12-29* • CSR5 (continued)**

Bit	Name	R/W	Reset Value	Function
5	UNF	R/W	0	Transmit underflow Indicates that the transmit FIFO was empty during a transmission. The transmit process goes into a suspended state. The user can clear this bit by writing a 1. Writing a 0 has no effect.
4:3	Reserved	N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation
2	TU	R/W	0	Transmit buffer unavailable When set, TU indicates that the host owns the next descriptor on the transmit descriptor list; therefore, it cannot be used by Ethernet MAC. When TU is set, the transmit process goes into a suspended state and can resume normal descriptor processing when the host changes ownership of the descriptor. Either a transmit-poll-demand command is issued or transmit automatic polling is enabled. The user can clear this bit by writing a 1. Writing a 0 has no effect.
1	TPS	R/W	0	Transmit process stopped TPS is set when the transmit process goes into a stopped state. The user can clear this bit by writing a 1. Writing a 0 has no effect.
0	TI	R/W	0	Transmit interrupt Indicates the end of a frame transmission process. Assertion of the TI bit can be delayed using the transmit interrupt mitigation counter/timer (CSR11[26:24] - NTP/CSR11[30:27] - TT). The user can clear this bit by writing a 1. Writing a 0 has no effect.

Operation Mode Register (CSR6)

Table 12-30 • Operation Mode Register (CSR6)

Bits 31:24		RA						
Bits 23:16		TTM	SF					
Bits 15:8	TR		ST				FD	
Bits 7:0	PM	PR		IF	PB	HO	SR	HP

Note: The CSR6 register has unimplemented bits (shaded). If these bits are read, they will return a predefined value, as shown in Table 12-31. Writing to these bits has no effect.

Table 12-31 • CSR6

Bit	Name	R/W	Reset Value	Function
31		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	RA	R/W	0	Receive all When set, all incoming frames are received, regardless of their destination address. An address check is performed, and the result of the check is written into the receive descriptor (RDES0[30]).
29:23		N/A	0b1100100	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22	TTM	R/W	0	Transmit threshold mode 1 – Transmit FIFO threshold set for 100 Mbps mode 0 – Transmit FIFO threshold set for 10 Mbps mode This bit is also used to select the frequency of both transmit and receive clocks between 2.5 MHz (10 Mbps operation) and 25 MHz (100 Mbps operation). This bit can be changed only when a transmit process is in a stopped state. This TTM bit is sent out of the Ethernet MAC as an output pin and connected to the SPEED port on the RMII to MII interface as an input port.
21	SF	R/W	0	Store and forward When set, the transmission starts after a full packet is written into the transmit FIFO, regardless of the current FIFO threshold level. This bit can be changed only when the transmit process is in the stopped state.
20:16		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:14	TR	R/W	0	Threshold control bits These bits, together with TTM, SF, and PS, control the threshold level for the transmit FIFO.

***Table 12-31* • CSR6 (continued)**

Bit	Name	R/W	Reset Value	Function
13	ST	R/W	0	Start/stop transmit command Setting this bit when the transmit process is in a stopped state causes a transition into a running state. In the running state, Ethernet MAC checks the transmit descriptor at a current descriptor list position. If Ethernet MAC owns the descriptor, then the data starts to transfer from memory into the internal transmit FIFO. If the host owns the descriptor, Ethernet MAC enters a suspended state. Clearing this bit when the transmit process is in a running or suspended state instructs Ethernet MAC to enter the stopped state. Ethernet MAC does not go into the stopped state immediately after clearing the ST bit. It will finish the transmission of the frame data corresponding to the current descriptor and then move to the stopped state. The status bits of the CSR5 register should be read to check the actual transmit operation state. Before giving the Stop Transmit command, the transmit state machine in CSR5 can be checked. If the transmission state machine is in SUSPENDED state, the Stop Transmit command can be given so that complete frame transmission by MAC is ensured.
12:10		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	FD	R/W	0	Full-duplex mode: 0 – Half-duplex mode 1 – Forcing full-duplex mode Changing of this bit is allowed only when both the transmitter and receiver processes are in the stopped state.
8		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PM	R/W	0	Pass all multicast When set, all frames with multicast destination addresses will be received, regardless of the address check result.
6	PR	R/W	0	Promiscuous mode When set, all frames will be received regardless of the address check result. An address check is not performed.
5		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	IF	R/W	0	Inverse filtering (read-only) If this bit is set when working in a perfect filtering mode, the receiver performs an inverse filtering during the address check process. The filtering type bits of the setup frame determine the state of this bit.

***Table 12-31* • CSR6 (continued)**

Bit	Name	R/W	Reset Value	Function
3	PB	R/W	0	Pass bad frames When set, Ethernet MAC transfers all frames into the data buffers, regardless of the receive errors. This allows the runt frames, collided fragments, and truncated frames to be received.
2	HO	R/W	0	Hash-only filtering mode (read-only) When set, Ethernet MAC performs an imperfect filtering over both the multicast and physical addresses. The filtering type bits of the setup frame determine the state of this bit.
1	SR	R/W	0	Start/stop receive command Setting this bit enables the reception of the frame by Ethernet MAC and the frame is written into the receive FIFO. If the bit is not enabled, then the frame is not written into the receive FIFO. Setting this bit when the receive process is in a stopped state causes a transition into a running state. In the running state, Ethernet MAC checks the receive descriptor at the current descriptor list position. If Ethernet MAC owns the descriptor, it can process an incoming frame. When the host owns the descriptor, the receiver enters a suspended state and also sets the CSR5[7] (receive buffer unavailable) bit. Clearing this bit when the receive process is in a running or suspended state instructs Ethernet MAC to enter a stopped state after receiving the current frame. Ethernet MAC does not go into the stopped state immediately after clearing the SR bit. Ethernet MAC will finish all pending receive operations before going into the stopped state. The status bits of the CSR5 register should be read to check the actual receive operation state.
0	HP	R/W	0	Hash/perfect receive filtering mode (read-only) 0 – Perfect filtering of the incoming frames is performed according to the physical addresses specified in a setup frame. 1 – Imperfect filtering over the frames with the multicast addresses is performed according to the hash table specified in a setup frame. A physical address check is performed according to the CSR6[2] hash-only (HO) bit. When both the HO and HP bits are set, an imperfect filtering is performed on all of the addresses. The filtering type bits of the setup frame determine the state of this bit.

Table 12-32 lists all possible combinations of the address filtering bits. The actual values of the IF, HO, and HP bits are determined by the filtering type (FT1–FT0) bits in the setup frame, as shown in Table 12-9 on page 184. The IF, HO, and HP bits are read-only.

***Table 12-32* • Receive Address Filtering Modes Summary**

PM CSR6[7]	PR CSR6[6]	IF CSR6[4]	HO CSR6[2]	HP CSR6[0]	Current Filtering Mode
0	0	0	0	0	16 physical addresses – perfect filtering mode
0	0	0	0	1	One physical address for physical addresses and 512-bit hash table for multicast addresses
0	0	0	1	1	512-bit hash table for both physical and multicast addresses
0	0	1	0	0	Inverse filtering

***Table 12-32* • Receive Address Filtering Modes Summary**

x	1	0	0	x	Promiscuous mode
0	1	0	1	1	Promiscuous mode
1	0	0	0	x	Pass all multicast frames
1	0	0	1	1	Pass all multicast frames

Table 12-33 lists the transmit FIFO threshold levels. These levels are specified in bytes.

***Table 12-33* • Transmit FIFO Threshold Levels (Bytes)**

CSR6[21]	CSR6[15:14]	CSR6[22] = 1	CSR6[22] = 0
0	00	64	128
0	01	128	256
0	10	128	512
0	11	256	1024
1	xx	Store and forward	Store and forward

Interrupt Enable Register (CSR7)

Table 12-34 • Interrupt Enable Register (CSR7)

Bits 31:24								
Bits 23:16								NIE
Bits 15:8	AIE	ERE			GTE	ETE		RSE
Bits 7:0	RUE	RIE	UNE			TUE	TSE	TIE

Note: The CSR7 register has unimplemented bits (shaded). If these bits are read, they will return a predefined value, as shown in Table 12-35. Writing to these bits has no effect.

Table 12-35 • CSR7

Bit	Name	R/W	Reset Value	Function
31:17		N/A	0b111100111111111	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation
16	NIE	R/W	0	Normal interrupt summary enable When set, normal interrupts are enabled. Normal interrupts are listed below: CSR5[0] – Transmit interrupt CSR5[2] – Transmit buffer unavailable CSR5[6] – Receive interrupt CSR5[11] – General-purpose timer expired CSR5[14] – Early receive interrupt
15	AIE	R/W	0	Abnormal interrupt summary enable When set, abnormal interrupts are enabled. Abnormal interrupts are listed below: CSR5[1] – Transmit process stopped CSR5[5] – Transmit underflow CSR5[7] – Receive buffer unavailable CSR5[8] – Receive process stopped CSR5[10] – Early transmit interrupt
14	ERE	R/W	0	Early receive interrupt enable When both the ERE and NIE bits are set, early receive interrupt is enabled.
13:12		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	GTE	R/W	0	General-purpose timer overflow enable When both the GTE and NIE bits are set, the general-purpose timer overflow interrupt is enabled.
10	ETE	R/W	0	Early transmit interrupt enable When both the ETE and AIE bits are set, the early transmit interrupt is enabled.

***Table 12-35* • CSR7 (continued)**

Bit	Name	R/W	Reset Value	Function
9		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	RSE	R/W	0	Receive stopped enable When both the RSE and AIE bits are set, the receive stopped interrupt is enabled.
7	RUE	R/W	0	Receive buffer unavailable enable When both the RUE and AIE bits are set, the receive buffer unavailable is enabled.
6	RIE	R/W	0	Receive interrupt enable When both the RIE and NIE bits are set, the receive interrupt is enabled.
5	UNE	R/W	0	Underflow interrupt enable When both the UNE and AIE bits are set, the transmit underflow interrupt is enabled.
4:3		N/A	0	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	TUE	R/W	0	Transmit buffer unavailable enable When both the TUE and NIE bits are set, the transmit buffer unavailable interrupt is enabled.
1	TSE	R/W	0	Transmit stopped enable When both the TSE and AIE bits are set, the transmit process stopped interrupt is enabled.
0	TIE	R/W	0	Transmit interrupt enable When both the TIE and NIE bits are set, the transmit interrupt is enabled.

Missed Frames and Overflow Counter Register (CSR8)

Table 12-36 • Missed Frames and Overflow Counter Register (CSR8)

Bits 31:24				OCO	FOC[10:7]
Bits 23:16	FOC[6:0]				MFO
Bits 15:8	MFC[15:8]				
Bits 7:0	MFC[7:0]				

Note: The CSR8 register has unimplemented bits (shaded). If these bits are read they will return a predefined value, as shown in Table 12-37. Writing to these bits has no effect.

Table 12-37 • CSR8

Bit	Name	R/W	Reset Value	Function
31:29	Reserved	N/A	0b111	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	OCO	R	0	Overflow counter overflow (read-only) Gets set when the FIFO overflow counter overflows. Resets when the high byte (bits [31:24]) is read.
27:17	FOC	R	0	FIFO overflow counter (read-only) Counts the number of frames not accepted due to receive FIFO overflow. The counter resets when the high byte (bits [31:24]) is read. When FIFO overflow occurs, the truncated frame is DMAed to memory with the CRC bit set.
16	MFO	R	0	Missed frame overflow Set when a missed frame counter overflows. The counter resets when the high byte (bits [31:24]) is read.
15:0	MFC	R	0	Missed frame counter (read-only) Counts the number of frames not accepted due to the unavailability of the receive descriptor. The counter resets when the high byte (bits [31:24]) is read. The missed frame counter increments when the internal frame cache is full and the descriptors are not available.

RMII Management Interface Register (CSR9)

Table 12-38 • RMII Management Interface Register (CSR9)

Bits 31:24								
Bits 23:16					MDI	MDEN	MDO	MDC
Bits 15:8								
Bits 7:0								

Note: The CSR9 register has unimplemented bits (shaded). If these bits are read they will return a predefined value, as shown in Table 12-39. Writing to these bits has no effect.

Table 12-39 • CSR9

Bit	Name	R/W	Reset Value	Function
31:20		N/A	0b111111111111	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	MDI	R	0	RMII management data in signal (read-only). This bit reflects the sample on the MDI during the read operation on the RMII management interface.
18	MDEN	R/W	0b1	RMII management operation mode 1 – Indicates that Ethernet MAC reads the RMII PHY registers. 0 – Indicates that Ethernet MAC writes to the RMII PHY registers. This bit controls the active low tristate enable for the top-level MDIO data output.
17	MDO	R/W	0	RMII management write data. The value of this bit drives the MDO signal when a write operation is performed.
16	MDC	R/W	0	RMII management clock. The value of this bit drives the MDC signal.
15:0		N/A	0b1000001111111011	Reserved Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

General-Purpose Timer and Interrupt Mitigation Control Register (CSR11)

Table 12-40 • General-Purpose Timer and Interrupt Mitigation Control Register (CSR11)

Bits 31:24	CS	TT	NTP
Bits 23:16	RT	NRP	CON
Bits 15:8	TIM[15:8]		
Bits 7:0	TIM[7:0]		

Table 12-41 • CSR11

Bit	Name	R/W	Reset Value	Function
31	CS	R/W	0b1	Cycle size Controls the time units for the transmit and receive timers according to the following: 1 – RMII 100 Mbps mode – 5.12 µs RMII 10 Mbps mode – 51.2 µs 0 – RMII 100 Mbps mode – 81.92 µs RMII 10 Mbps mode – 819.2 µs
30:27	TT	R/W	0b1111	Transmit timer Controls the maximum time that must elapse between the end of a transmit operation and the setting of the CSR5[0] (TI-transmit interrupt) bit. This time is equal to TT × (16 × CS). The transmit timer is enabled when written with a nonzero value. After each frame transmission, the timer starts to count down if it has not already started. It is reloaded after every transmitted frame. Writing 0 to this field disables the timer effect on the transmit interrupt mitigation mechanism. Reading this field gives the actual count value of the timer.
26:24	NTP	R/W	0b111	Number of transmit packets Controls the maximum number of frames transmitted before setting the CSR5[0] (TI-transmit interrupt) bit. The transmit counter is enabled when written with a nonzero value. It is decremented after every transmitted frame. It is reloaded after setting the CSR5[0] - TI bit. Writing 0 to this field disables the counter effect on the transmit interrupt mitigation mechanism. Reading this field gives the actual count value of the counter.
23:20	RT	R/W	0b1111	Receive timer Controls the maximum time that must elapse between the end of a receive operation and the setting of the CSR5[6] (RI-receive interrupt) bit. This time is equal to RT × CS. The receive timer is enabled when written with a nonzero value. After each frame reception, the timer starts to count down if it has not already started. It is reloaded after every received frame. Writing 0 to this field disables the timer effect on the receive interrupt mitigation mechanism. Reading this field gives the actual count value of the timer.

***Table 12-41* • CSR11 (continued)**

Bit	Name	R/W	Reset Value	Function
19:17	NRP	R/W	0b111	Number of receive packets Controls the maximum number of received frames before setting the CSR5[6] (RI-receive interrupt) bit. The receive counter is enabled when written with a nonzero value. It is decremented after every received frame. It is reloaded after setting the CSR5[6]-RI bit. Writing 0 to this field disables the timer effect on the receive interrupt mitigation mechanism. Reading this field gives the actual count value of the counter.
16	CON	R/W	0	Continuous mode 1 – General-purpose timer works in continuous mode 0 – General-purpose timer works in one-shot mode This bit must always be written before the timer value is written.
15:0	TIM	R/W	0	Timer value Contains the number of iterations of the general-purpose timer. Each iteration duration is as follows: RMII 100 Mbps mode – 81.92 µs RMII 10 Mbps mode – 819.2 µs

IOMUXes Associated with Ethernet MAC

IOMUXes 16, 17, 18, 19, 20, 21, 22, 23, and 24 are used to multiplex Ethernet MAC and fabric interface signals to MSSIOBUFs. These are RMII PHY interface signals except for MAC_RMII_CLK input, which is a dedicated MSSIOBUF on a SmartFusion cSoC. Refer to the "Fabric Interface and IOMUX" section on page 343 for a description of the IOMUX.

IOMUXes for MAC_TXD[1:0], MAC_RXD[1:0], MAC_TX_EN, MAC_CRSDV, MAC_RX_ER, MAC_MDIO, MAC_MDC

To use the MAC_TXD[1:0], MAC_RXD[1:0], MAC_TX_EN, MAC_CRSDV, MAC_RX_ER, MAC_MDIO, and MAC_MDC signals, an IOMUX is used to route the signal to an MSSIOBUF. This IOMUX is used to share the MSSIOBUF between the various MAC signals and fabric, when MAC is not in use.

Figure 12-12 shows the IOMUX topology for MAC_TXD[0]. A similar topology exists for the remaining MAC signals.

In this case, IOMUX_16 is configured to connect OUT_A to MSSIOBUF IO_O port. When MAC is not in use, the M2F[0], F2M[0], and F2M_OE[0] can then be routed to use MSSIOBUF.

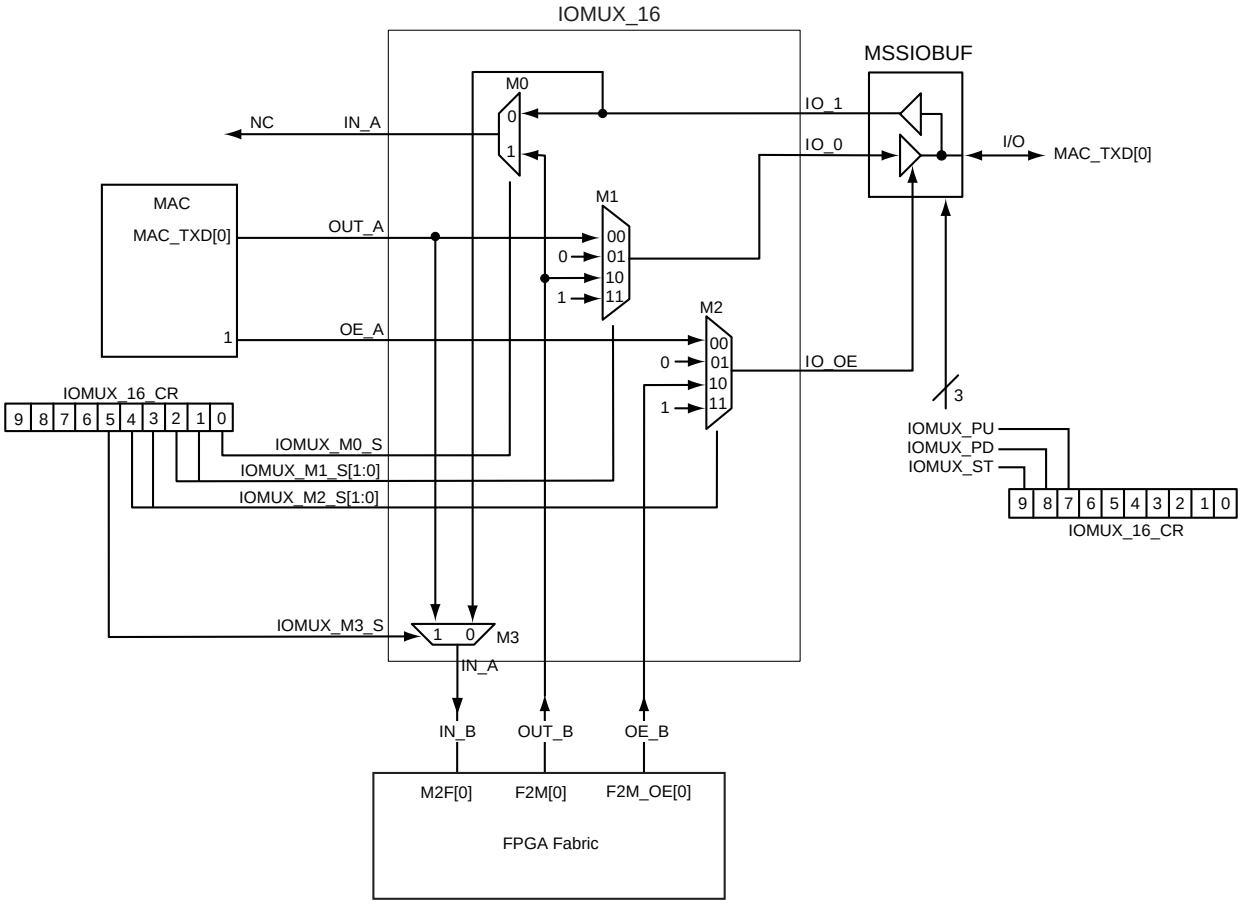


***Figure 12-12* • Example of Ethernet MAC Interaction with FPGA Fabric via an IOMUX**

Table 12-42 lists the association between MAC I/Os, IOMUXes, and fabric interface.

***Table 12-42* • MAC IO Interaction with Fabric and IOMUXes**

MAC_x_Signal	Fabric Interface	IOMUX
MAC_TXD[0]	0	16
MAC_TXD[1]	1	17
MAC_RXD[0]	2	18
MAC_RXD[1]	3	19
MAC_TXEN	4	20
MAC_CRSDV	5	21
MAC_RXER	6	22
MAC_MDIO	7	23
MAC_MDC	8	24

Table 12-43 through Table 12-51 on page 218 give the descriptions for all IOMUXes associated with the Ethernet MAC.

IOMUX 16

***Table 12-43* • IOMUX 16**

Pad Name	Pad Ports	IOMUX_16_CR	IOMUX 16 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_TXD[0]/ IOuxwByVz	I					M2F[0]		
	O			MAC_TXD[0]			F2M[0]	
	OE				VDD			F2M_OE[0]
	PU	IOMUX_16_PU						
	PD	IOMUX_16_PD						
	ST	IOMUX_16_ST						

IOMUX 17

***Table 12-44* • IOMUX 17**

Pad Name	Pad Ports	IOMUX_17_CR	IOMUX 17 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_TXD[1]/ IOuxwByVz	I					M2F[1]		
	O			MAC_TXD[1]			F2M[1]	
	OE				VDD			F2M_OE[1]
	PU	IOMUX_17_PU						
	PD	IOMUX_17_PD						
	ST	IOMUX_17_ST						

IOMUX 18

Table 12-45 • IOMUX 18

Pad Name	Pad Ports	IOMUX_18_CR	IOMUX 18 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_RXD[0]/ IOuxwByVz	I		MAC_RXD[0]			M2F[2]		
	O						F2M[2]	
	OE				GND			F2M_OE[2]
	PU	IOMUX_18_PU						
	PD	IOMUX_18_PD						
	ST	IOMUX_18_ST						

IOMUX 19

Table 12-46 • IOMUX 19

Pad Name	Pad Ports	IOMUX_19_CR	IOMUX 19 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_RXD[1]/ IOuxwByVz	I		MAC_RXD[1]			M2F[3]		
	O						F2M[3]	
	OE				GND			F2M_OE[3]
	PU	IOMUX_19_PU						
	PD	IOMUX_19_PD						
	ST	IOMUX_19_ST						

IOMUX 20

Table 12-47 • IOMUX 20

Pad Name	Pad Ports	IOMUX_20_CR	IOMUX 20 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_TXEN/ IOuxwByVz	I					M2F[4]		
	O			MAC_TXEN			F2M[4]	
	OE				VDD			F2M_OE[4]
	PU	IOMUX_20_PU						
	PD	IOMUX_20_PD						
	ST	IOMUX_20_ST						

IOMUX 21

Table 12-48 • IOMUX 21

Pad Name	Pad Ports	IOMUX_21_CR	IOMUX 21 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_CRSDV/ IOuxwByVz	I		MAC_CRSDV			M2F[5]		
	O			GND			F2M[5]	
	OE				GND			F2M_OE[5]
	PU	IOMUX_21_PU						
	PD	IOMUX_21_PD						
	ST	IOMUX_21_ST						

IOMUX 22

Table 12-49 • IOMUX 22

Pad Name	Pad Ports	IOMUX_22_CR	IOMUX 22 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_RXER/ IOuxwByVz	I		MAC_RXER			M2F[6]		
	O			GNS			F2M[6]	
	OE				GND			F2M_OE[6]
	PU	IOMUX_22_PU						
	PD	IOMUX_22_PD						
	ST	IOMUX_22_ST						

IOMUX 23

Table 12-50 • IOMUX 23

Pad Name	Pad Ports	IOMUX_23_CR	IOMUX 23 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_MDIO/ IOuxwByVz	I		MAC_MDI			M2F[7]		
	O			MAC_MDO			F2M[7]	
	OE				MAC_MDEN			F2M_OE[7]
	PU	IOMUX_23_PU						
	PD	IOMUX_23_PD						
	ST	IOMUX_23_ST						

IOMUX 24

Table 12-51 • IOMUX 24

Pad Name	Pad Ports	IOMUX_24_CR	IOMUX 24 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
MAC_MDC/ IOuxwByVz	I					M2F[8]		
	O			MAC_MDC			F2M[8]	
	OE				VDD			F2M_OE[8]
	PU	IOMUX_24_PU						
	PD	IOMUX_24_PD						
	ST	IOMUX_24_ST						

13 – Serial Peripheral Interface (SPI) Controller

The serial peripheral interface controller is an APB slave that provides a serial interface compliant with the Motorola SPI, Texas Instruments synchronous serial, and National Semiconductor MICROWIRE™ formats. In addition, the SPI supports interfacing to large SPI flash and EEPROM devices.

The SmartFusion device has two identical SPI peripherals. The letter x in register and signal descriptions is used as a placeholder for 0 or 1, indicating SPI_0 or SPI_1.

Figure 13-1 shows a block diagram for the SPI controller.

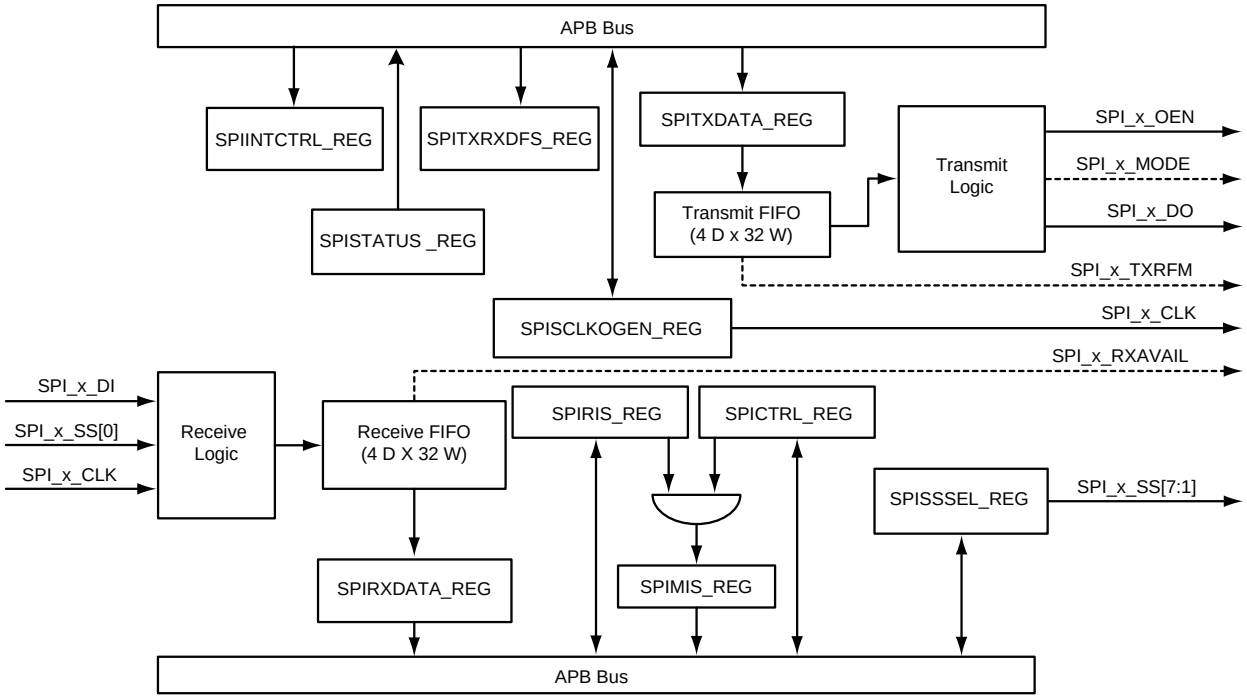


***Figure 13-1* • SPI Controller Block Diagram**

SPI Controller Functional Description

SPI controller supports both Master and Slave modes of operation.

SPI Controller Block Diagram

Figure 13-1 on page 219 shows the SPI Controller block diagram. See Table 13-1 on page 221 for SPI interface signals definitions.

- In Master mode, SPI generates SPI_x_CLK, selects a slave using SPI_x_SS, transmits data on SPI_x_DO, and receives data on SPI_x_DI.
- In Slave mode, SPI is selected by SPI_x_SS, receives clock on SPI_x_CLK, and incoming data on SPI_x_DI.

SPI controller embeds two 4 × 32 (depth × width) FIFOs for receive and transmit. These FIFOs are accessible through RX data and TX data registers. Writing to the TX data register causes the data to be written to the transmit FIFO. This is emptied by transmit logic. Similarly reading from the RX data register causes data to be read from the receive FIFO.

The not empty port of receive FIFO and not full port of transmit FIFO flags of the FIFOs are exposed as SPIRXAVAIL (SPI has data to be read) and SPITXRFM (SPI has more room to send) ports. These are connected to the PDMA engine to allow for continuous DMA streaming for large SPI transfers and thus helps to free up the ARM Cortex-M3 processor.

Interrupts can be setup to signal the following:

- The completion of a data frame transfer transmission/reception
- Overflow/under-run events when the DMA channel accesses the transmit or receive FIFOs.

SPI Modes of Transfer

SPI controller has two basic modes of transfer. It can be controlled by the Cortex-M3 microcontroller or the peripheral DMA (PDMA). In Cortex-M3 mode, the transfers are handled by firmware which can poll the status register or respond to interrupts. In PDMA mode the transfers are automatically handled by the PDMA engine.

Cortex-M3 Controlled Mode

In this mode, the size of the data frames (size of the single transfer is set in register SPITXRXDFS_REG) and the number of transfers (set in the TXRXDFCOUNT field of the CONTROL register) are specified. Upon completion of each transfer—that is, after a specified number of data frames (1 by default) are sent—an optional interrupt is generated. The SPI controller keeps track of the number of data frames so that special signals like output enable can be deactivated at the end of a transfer.

For example, consider the transmission of 64 KB of data to an external EEPROM from the Cortex-M3 controlled SPI controller. The data frame size is set to 8 and the number of data frames per transfer is set to 1. After each transfer, the software must respond to the interrupt—transmit done—and reload the FIFO until all 64 KB are sent. To improve throughput, the number of data frames per each transfer can be set to 4 to utilize the full depth of the transmit FIFO.

PDMA Mode

In this mode, interrupts are turned off and the PDMA engine uses the SPITXRFM and SPIRXAVAIL signals to control the filling and emptying of the transmit and receive FIFOs. The SPITXRFM is connected to the transmit FIFO not full flag. The SPIRXAVAIL is connected to the receive FIFO not empty flag.

In DMA mode, the TXDONE and RXRDY interrupts are masked in RIS and the interrupt capability in the PDMA engine is used to notify the application on completion. For more information on PDMA, refer to the "Peripheral DMA (PDMA)" section on page 35.

For example, consider the transmission of 64 KB data to an external EEPROM from a PDMA-controlled SPI controller. The data frame size is set to 8 and the number of data frames per transfer is set to 1. The transmit FIFO is repeatedly emptied by the PDMA engine, using the SPITXRFM signal.

SPI Interface Signals

Figure 13-1 lists SPI signals. Signals that are brought to chip-level pins are marked as external. Signals that interface with other parts of SmartFusion MSS are marked as internal.

***Table 13-1* • SPI Interface Signals**

Name	Type	Polarity/ Bus Size	Description
External Pins			
SPI_x_DI	Input	1	Serial data input
SPI_x_DO	Output	1	Serial data output
SPI_x_CLK	Input/output	1	Serial clock. Input when SPI is in Slave mode. Output when SPI is in Master mode.
SPI_x_SS[0]	Input/output	1	Slave select. Input when SPI is in Slave mode. Output when SPI is in Master mode.
Signals Routed Via IOMUXes but Not to a Pin			
SPI_x_SS[7:1]*	Output	7	Extra slave select signal. Valid only in Master mode.
Internal Signals			
SPI_x_MODE	Output	1	SPI mode. Used by the MSS IOMUXes to determine INOUT signal directions (1 = Master, 0 = Slave).
SPI_x_OEN	Output	1	Output enable
SPI_x_TXRFM	Output	1	SPI ready to transmit. Used by MSS PDMA engine.
SPI_x_RXAVAIL	Output	1	SPI received data. Used by MSS PDMA engine.

*Note: *For the A2F200 device, SPI_0 can only select four slaves. This means only SPI_0_SS[0], SPI_0_SS[3:1] outputs are valid.*

SPI Controller Operation

This section describes SPI controller operation, including FIFO, interrupt, and error handling. The SPI controller supports three types of data transfer protocols. These are selected by bits 2 and 3 of the Control Register (CONTROL [3:2]). These are the transfer protocols:

- Motorola SPI
- National Semiconductor MICROWIRE
- TI Synchronous Serial

The protocol details are explained in the"SPI Data Transfer Protocol Details" section on page 223.

SPI Transmit and Receive FIFO Flags and Interrupts

The SPI controller contains two 4 × 32 (depth x width) FIFOs: one for the receive side and the other for the transmit side. The TXFIFOFUL and TXFIFOEMP bits of the SPISTAUS_REG register indicate the full or empty status of the transmit FIFO. The RXFIFOFUL and RXFIFOEMP bits of the SPISTAUS_REG register indicate the full or empty status of the receive FIFO. The Cortex-M3 microcontroller can poll these bits to obtain the status of the corresponding FIFO.

For large data transfers under Cortex-M3 control, the full depth of transmit FIFO can be used by setting the number of data frames in a burst to a number greater than 1 (maximum is 64 K frames). When interrupts are enabled, the TXDONE bit of the RIS register is asserted after all the data frames in the burst are sent.

For example, if the data frame size is set to 32 and the count to 2, then interrupt TXDONE is generated after every two words (32 bits). The default value for the frame count is 1.

The TXUNDERRUN and RXOVERFLOW bits of the SPICNTL_REG register are conditional interrupts that are available for each channel (in DMA mode) to indicate that a FIFO under-run or FIFO overflow has occurred. If the transmit FIFO is accessed for data to transfer and there is no data in the FIFO, then a transmit under-run error (TXUNDERRUN) is generated. This can be conditionally used to generate an interrupt. In this event, the transmission is assumed to have been lost and the application must catch the error and restart the transmission from the beginning. Internally the transmit logic returns to an idle state and the entire transmission is deemed lost.

If the channel attempts to write to a receive FIFO which is already full then receive overflow error (RXOVERFLOW) is generated. This can be conditionally used to generate an interrupt. In this case, the transmission continues but the data is now corrupted because a data frame is missing. It is assumed that the software will clear the interrupt and recover (possibly by reading from the receive FIFO to clear the source of the interrupt, allowing more data to be received, or even by halting the transmission and resetting the SPI controller).

There are no interrupts available to signal to the Cortex-M3 microcontroller that the transmit FIFO has overflowed or a read operation is attempted on an empty receive FIFO. The SPITXRFM flag (room for more) and SPIRXAVAIL flag (data ready to be read) are assumed to be used only in DMA mode under the control of the DMA Engine, which has its own interrupts and control mechanism.

SPI Clock Requirements

The SPI_0 and SPI_1 peripherals are clocked by PCLK0 on APB bus 0 (APB_P0) and PCLK1 on APB bus 1 (APB_P1), respectively. PLCLK0 and PLCK1 are free running versions of FCLK (the main clock driving the entire MSS) which are derived from the MSS_CCC. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109.

In slave mode, the input clock to the SPI controller (SPICLK) can not be faster than one twelfth of PCLK0 or PCLK1. This means that for a PCLK of 100 MHz, the maximum SPI clock speed allowed is 8.33 MHz. In master mode, the SPI clock (SPI_x_CLK) can run at even divisors of PCLK, ranging from 2 to 256. This also means that for a PCLK of 100 MHz, the allowed range for SPI clock is 390 KHz to 50 MHz.

SPI Status at Reset

After reset, the slave select (SPI_x_SS[0]) pins default to a logic High. After selecting the SPI mode and enabling the SPI controller, the SPI_x_SS lines default to the correct values for each protocol (see the "SPI Control Register (CONTROL)" section on page 233). After reset, the clock out (SPI_x_CLK) is a logic Low.

At reset, the FIFOs are cleared and their respective read and write pointers are set to zero. Similarly, all the internal registers on the SPI controller are reset to their default values, as shown in the "SPI Register Interface Details" section on page 233.

SPI Error Recovery and Handling

The SPI protocol defines only the packet formats for data transmission and does not include any error recovery strategy "physical layer" protocols. Specifically, if an error occurs on a slave (for example, it fails to respond to the chip select or gets overwhelmed with incoming data) the master will not necessarily be aware of it. The master and slave must therefore have prior knowledge of each other’s capabilities before transmission can begin.

RX Overflow

An RX overflow condition arises when the receive FIFO has not been emptied in time. As a result, the last write to the receive FIFO from the channel overwrote some previously received data that had not yet been read by the host processor. An example of this scenario happens when the SPI controller is operating in master mode and the receive FIFO is not being serviced by the processor after the SPI

controller raises the RXRDY interrupt flag found in the RIS register. Eventually the FIFO will fill up and subsequent writes by the channel will cause the RX overflow to occur.

The corrective action required is for the host to read from the FIFO until the FIFO is empty. This can be checked by reading the FIFO status in the STATUS register.

TX Under-Run

A TX under-run condition arises when a channel requests to send data while no data is available in the transmit FIFO. An example of this scenario happens when the SPI Controller is operating in slave mode and gets a request to send data while no data is available in transmit FIFO.

The corrective action required is for the host to write data into the transmit FIFO. The status flags TXFIFOEMP or TXFIFOEMPNXT to indicate whether the FIFO is empty or will be after the next read operation.

SPI Data Transfer Protocol Details

This section covers the details of each of the three data transfer protocols, including timing diagrams, signal requirements, and error case scenarios.

Motorola SPI Protocol

The Motorola SPI is a full duplex, four-wire synchronous transfer protocol. It supports programmable clock polarity and phase.

The SPO (clock polarity) control bit determines the polarity of the clock. If SPO is Low, SPISCLKO is driven low when no data is transferred. If SPO is High, SPISCLK is driven high when no data is transferred.

The SPH (clock phase) control bit determines the clock edge that captures the data. When SPH is Low, data is captured on the first clock transition (rising edge if SPO = 0). When SPH is HIGH, data is captured on the second clock transition (rising edge if SPO = 1).

Table 13-2 summarizes the active edges of the various master SPI modes for A2F200. See Table 13-3 on page 231 for A2F060 and A2F500.

***Table 13-2* • Motorola SPI Transfer Modes (A2F200 only)**

Mode	SPO/SPH	Sample Edge	Shift Edge	Pulse Slave Select Between Continuous Transfers (Master mode)	Clock in Idle Period, Slave Select in Idle Period
0	0/0	Rising	Falling	Yes	0/1
1	0/1	Falling	Rising	No	0/1
2	1/0	Falling	Rising	Yes	1/1
3	1/1	Rising	Falling	No	1/1

The number of bits transferred is set in the TxRx Data Frame Register (TXRXDF_SIZE). Note that SPI_x_SS is not pulsed between frames when SPH = 1.

For completeness, the rest of the possible transfer modes are shown in Figure 13-6 through Figure 13-3 on page 224.

Note: The timing diagrams in Figure 13-2 on page 224 through Figure 13-13 on page 230 apply to all SmartFusion devices.

Single Frame Transfer – Mode 0: SPO = 0, SPH = 0

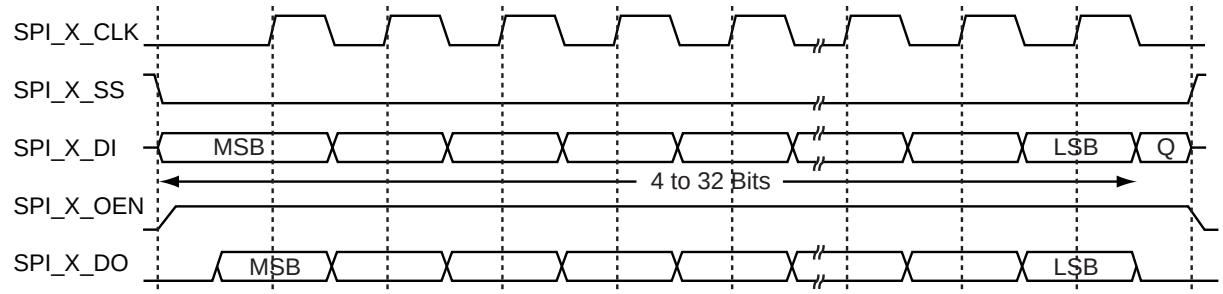
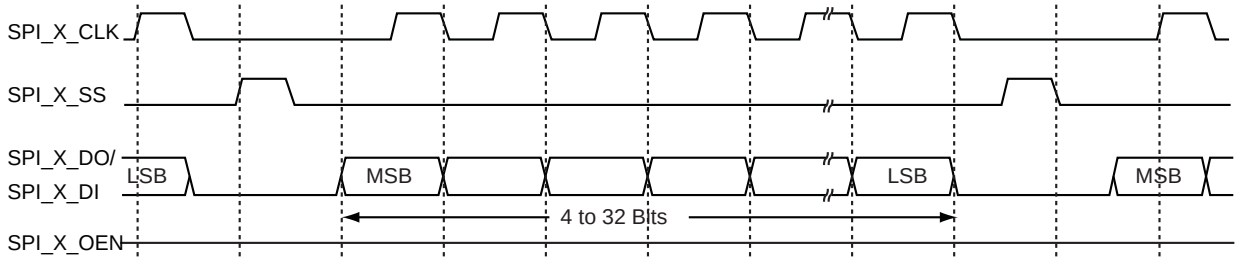


***Figure 13-2* • Motorola SPI Mode 0**

Multiple Frame Transfer – Mode 0: SPO = 0, SPH = 0



Notes:

1. *Between frames, the slave select (SPI_x_SS) is asserted for the duration of clock pulse.*
2. *Between frames, the clock (SPI_x_CLK) is low.*
3. *Data is transferred most significant bit (MSB) first.*
4. *The output enable (SPI_x_OEN) signal is asserted during transmission, deasserted at end of transfer (after the last frame is sent).*

***Figure 13-3* • Motorola SPI Mode 0 Multiple Frame Transfer**

Single Frame Transfer – Mode 1: SPO = 0, SPH = 1

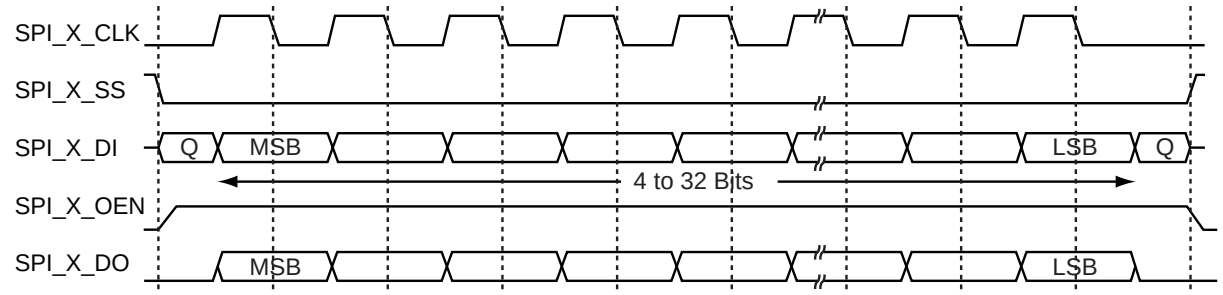


***Figure 13-4* • Motorola SPI Mode 1**

Single Frame Transfer – Mode 2: SPO = 1, SPH = 0

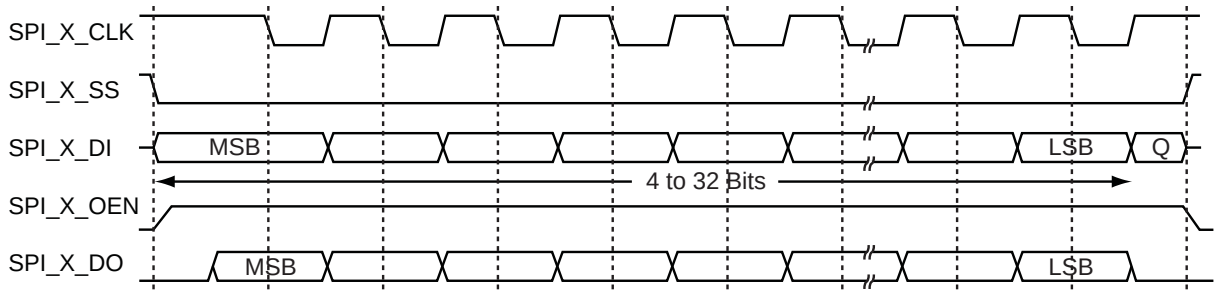


***Figure 13-5* • Motorola SPI Mode 2**

Single Frame Transfer – Mode 3: SPO = 1, SPH = 1

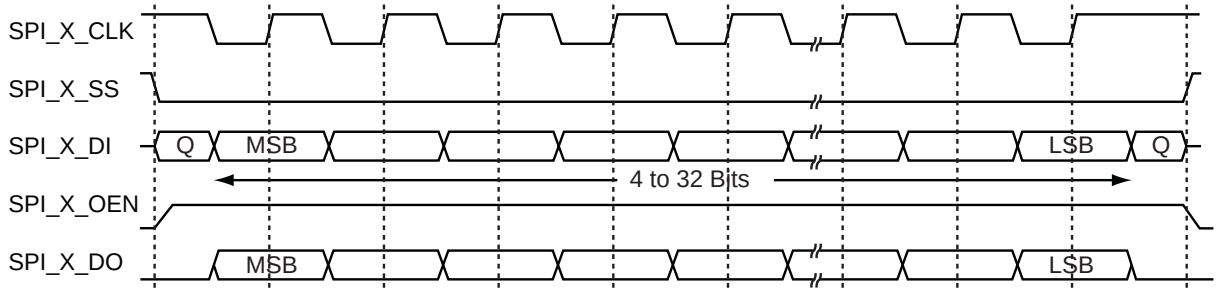


***Figure 13-6* • Motorola SPI Mode 3**

Output Enable (SPI_x_OEN) Timing

- Each SPI mode comprises two phases: a transmit (or shift out) and receive (or sample). It is a requirement that the output enable (SPI_x_OEN) line which enables the output pad should be driven so that the pad is ready to transmit when the data is available (setup).
- The pad is held on long enough for the recipient to sample the data (hold).

The minimum setup and hold time is one half SPI_x_CLK.

In slave mode, the situation is slightly complicated because the input clock is withdrawn at the end of the transfer. For example, consider the waveform for SPO = 1, SPH = 0 (Figure 13-5). In this case, data is sampled on the falling edge of the clock and shifted on the rising edge of the clock.

The data is sampled on the falling edge and must be held for one half SPI_x_CLK after the last falling edge at the end of the transmission. This means SPI_x_OEN must be held High for at least one half SPI_x_CLK after the last falling edge to satisfy the hold time requirement.

However, in the above slave case, the SPI_x_CLK input has been withdrawn so it cannot be used for timing purposes. In this SPI controller implementation the following rules are used:

- After the last active edge of SPI_x_CLK, the SPI_x_OEN signal is held active for at least one internal slave SPI_x_CLK cycle (which is derived from the input PCLK using the clock division frequency register CLK_GEN).
- The SPI_x_OEN is held active if the slave select line (SPI_x_SS) is Low, there is a transmission in progress, or there is more data to be transmitted.

Motorola SPI Error Case Scenarios

The SPI protocol does not specify any error recovery strategy. The master and slave require prior knowledge of clock rates and data-frame layouts.

However, there are built-in mechanisms in the SPI controller, by which when the Slave encounters an error, the Master can toggle the Slave clock until it gets to a known state. Here are three specific error scenarios and the behavior of the SPI controller in Motorola protocol mode:

- If the slave select signal is withdrawn in the middle of a transfer, the transfer continues until the end of the data frame.
- If the input clock is withdrawn, the SPI controller will remain paused until the clock is restarted. It will pick up where it left off.
- If the slave select signal is withdrawn before a transfer occurs, the slave remains in the IDLE state (no data transfer having been initiated).

The SPI controller has no built-in timer. For applications where there is a possibility of a slave going to sleep for a long time, or in the case of very long transfers, the application should use one of the SmartFusion on-chip timers. Refer to the "System Timer" section on page 305 for more information.

National Semiconductor MICROWIRE Protocol

The National Semiconductor MICROWIRE serial interface is a half-duplex protocol using a master/slave message passing technique. Each serial transmission begins with an 8-bit control word, during which time no incoming data is received. After the control word is sent, the external slave decodes it, and after waiting one serial clock cycle from the end of the control word, responds with the required data, which may be of a length of 4 to 16 bits.

Single Frame Transfer

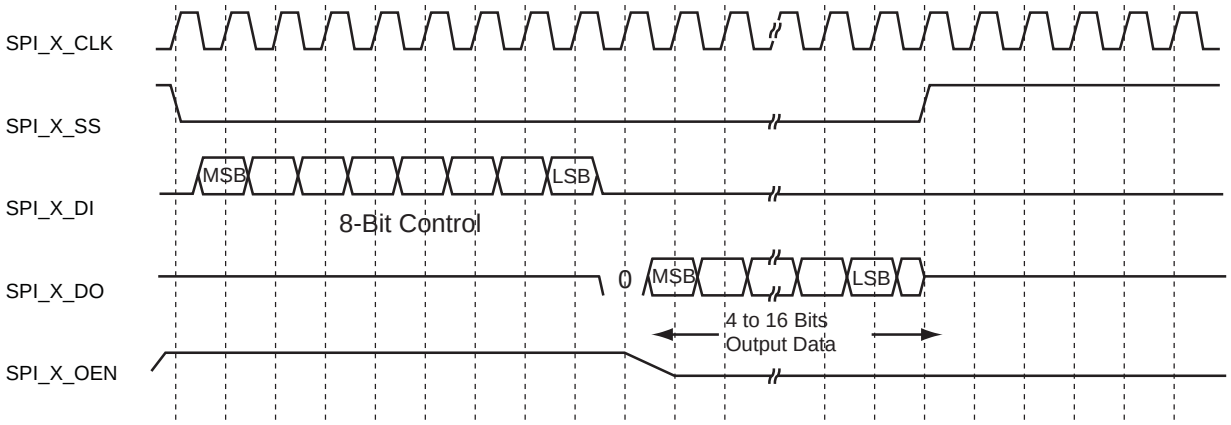


***Figure 13-7* • National Semiconductor MICROWIRE Single Frame Transfer**

In this mode, the most significant byte of the FIFO transmit word is the control byte. The total data frame size supplied must be at least 12 bits long (8 bits for the control word and a minimum of 4 bits for data payload). Only the output data is sampled and inserted in the receive FIFO.

Multiple Frame Transfer

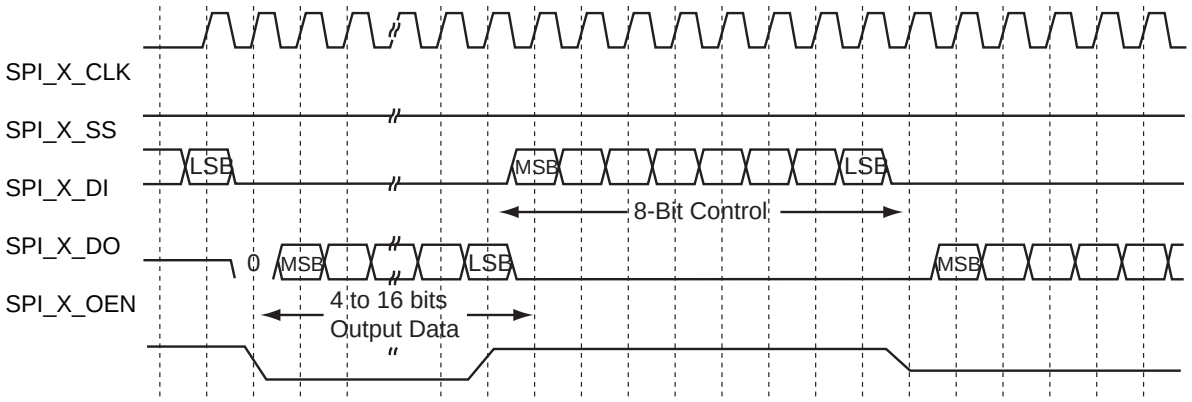


***Figure 13-8* • National Semiconductor MICROWIRE Multiple Frame Transfer**

The slave select signal (SPI_x_SS) is continuously asserted (held Low) while SPI_x_OEN is also asserted (Low) for the duration of each control byte. The other data transfers proceed in back-to-back manner.

Texas Instruments (TI) Synchronous Serial Protocol

The TI synchronous serial interface is based on a full duplex, four-wire synchronous transfer protocol. The transmit data pin is put in a high-impedance mode (tristated) when not transmitting.

- The slave select (SPI_x_SS) signal is pulsed between transfers to guarantee a High-to-Low transition between each frame.
- In an idle state, the slave select (SPI_x_SS) signal is kept Low.
- Data is available on the clock cycle immediately following the slave select (SPI_x_SS) assertion.
- Both the SPI master and the SPI slave capture each data bit into their serial shift registers on the falling edge of the clock (SPI_x_CLK). The received data is latched on the rising edge of the clock (SPI_x_CLK).
- The output enable signal (SPI_x_OEN) is asserted (active Low) throughout the transfer.

Single Frame Transfer

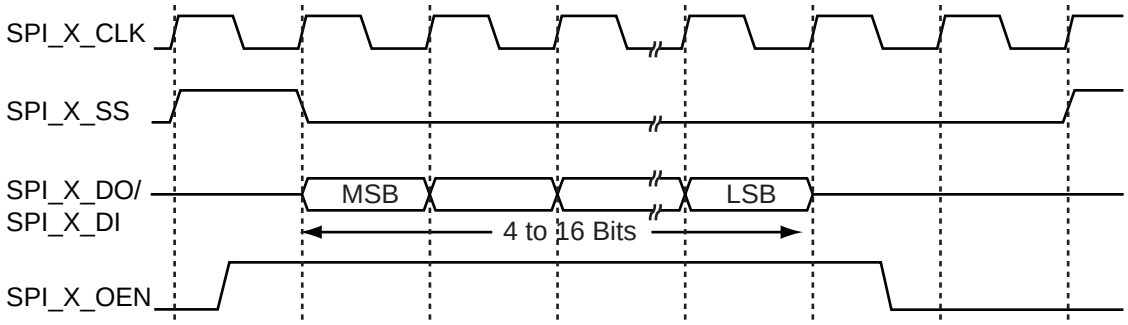


***Figure 13-9* • TI Synchronous Serial Single Frame Transfer**

Multiple Frame Transfer

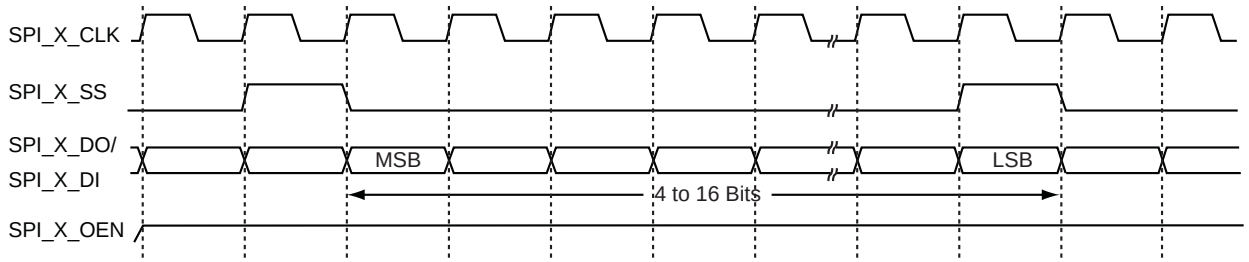


***Figure 13-10* • TI Synchronous Serial Multiple Frame Transfer**

Texas Instruments Synchronous Serial Error Case Scenarios

When the SPI controller is configured for the TI synchronous serial protocol, while in Slave mode, it responds to failure events on slave select (SPI_x_SS) and slave clock (SPI_x_CLK) in the following manner:

- Withdrawal of SPI_x_CLK: In this case the device pauses and will resume on reasserting the clock.
- Premature pulsing of slave select: If the slave select is pulsed during a data frame transmission, it will be ignored.
- Disconnection of slave select before a transfer: The transfer is not initiated unless the pulse is issued.

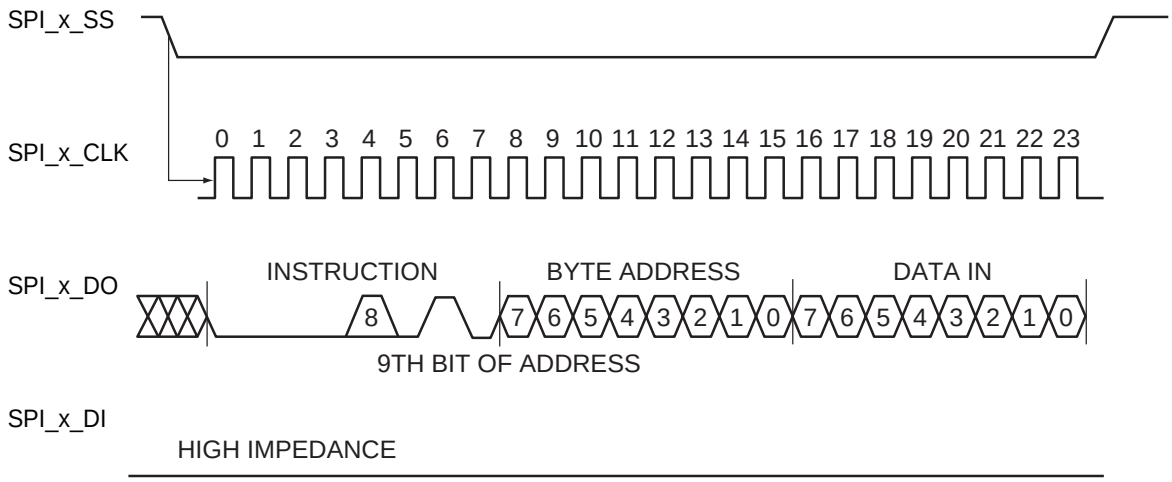
SPI Data Transfer for Large Flash/EEPROM Devices in Motorola SPI Modes

Serial flash and EEPROM devices can be driven using the Motorola SPI modes. The following outlines the interfaces to the required flash/EEPROM devices and shows how they can be driven using the Motorola SPI modes. In each of these modes, the SPI controller is configured as a master with the slave select line hooked to the signal SPICS shown on the waveform. The serial flash/EEPROM device then acts as the slave.

Devices that Require Data Frame Sizes of up to 32 Bits

Serial flash/EEPROM devices, such as the Atmel 25010/020/040, have a data frame size smaller than 32 bits and can be directly driven from the SPI mode.

Write Operation for Atmel 25010/20/40 Devices

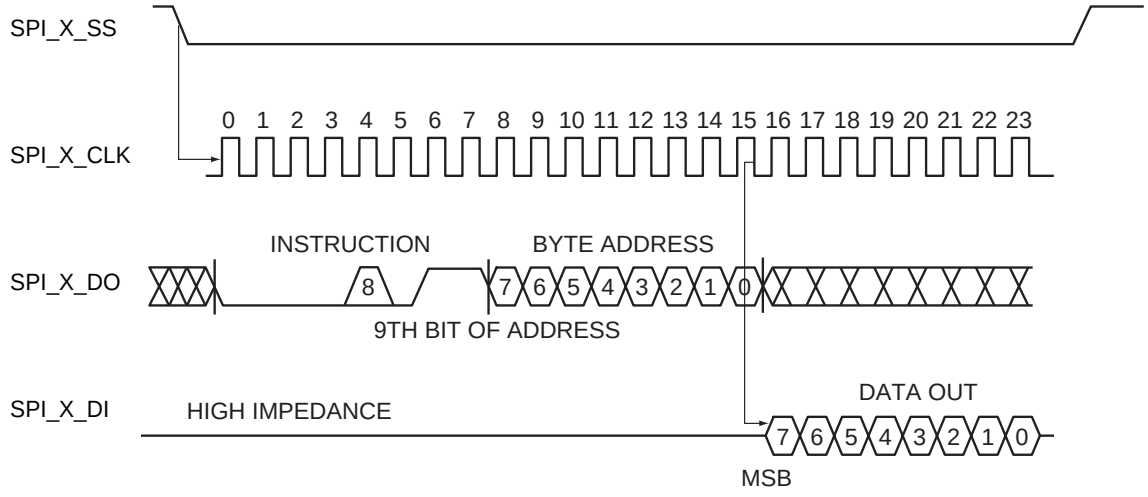


Note: This first byte contains the op-code that defines the operations to be performed. The op-code also contains address bit A8 in both the READ and WRITE instructions. This is mandated by the Atmel device.

***Figure 13-11* • Write Operation Timing**

The SPI controller selects the devices using the slave select signal. The data frame size is set to 24 bits. The SPI is configured with SPO = 0, SPH = 0. The first byte is the instruction. Bit 5 of the instruction is part of the address (the 9th bit as required by the Atmel part). Bits 8-15 form a byte address. The residual 8 bits correspond to the data to be written.

Read Operation for Atmel 25010/20/40 Devices



Note: This first byte contains the op-code that defines the operations to be performed. The op-code also contains address bit A8 in both the READ and WRITE instructions. This is mandated by the Atmel device.

***Figure 13-12* • Read Operation Timing**

For the read operation, the data frame size is set to 24 bits and the SPI controller is configured with SPO = 0, SPH = 0. Upon completion, the least significant byte of the received data frame corresponds to the data read.

Devices that Require Data Frame Sizes of More than 32 Bits

Serial flash devices, such as the Atmel AT25DF321, which support mode 3 (SPO = 1 and SPH = 1), require more than 32 bits of frame data in some modes.

To drive these devices, continuous transfers are required from the SPI interface while holding slave select Low continuously (which is connected to the chip select of the target device). This is accomplished by using the transmit FIFO from the SPI which, if it is kept not empty, will enforce continuous back-to-back transfers. The slave select will continue to be held Low (active) in SPI Mode 1 (SPO = 0 and SPH = 1) and Mode 1 (SPO = 1 and SPH = 1) and not pulsed between data frames.

For example, to send 64 bits to the AT25DF321 (8-bit op-code, 24-bit address, 4 data bytes), the data frame size (TXRXDF_SIZE) can be set to 32 and the data frame count set to 2 (field TXRXDFCOUNT of CONTROL).

Page Program for Atmel AT25DF321

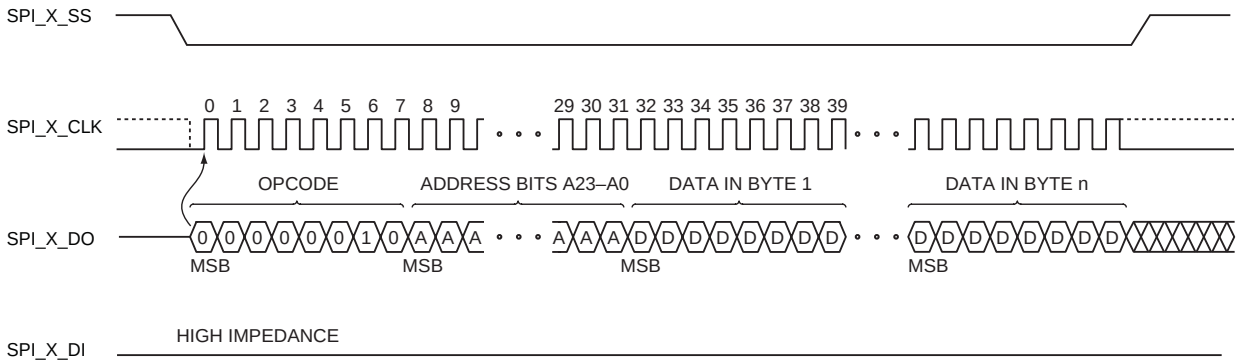


Figure 13-13 **• Page Program Timing**

In this mode, the op-code, address, and data require more than 32 clock periods. To drive this device, the chip select (CS) can be connected to the slave select signal, the data frame size set to 16, and the FIFO repeatedly filled until the target flash device is programmed. As long as there is data to be transmitted in the FIFO, the chip select signal (connected to slave select on the SPI controller) should be asserted Low.

Devices that Do Not Support Mode 1 (SPO = 0 and SPH =1) or Mode 3 (SPO = 1 and SPH = 1)

For flash devices which do not support mode 1 (SPO = 0 and SPH = 1) or mode 3 (SPO = 1 and SPH = 1), it is necessary to use a dedicated GPIO pin to drive the chip select signal.

The device driver would initially assert the chip select through the GPIO, then activate the SPI transfer—possibly comprising many individual 32-bit data frames—and finally withdraw the GPIO.

Control Bits SPS, SPO, and SPH (A2F060 and A2F500)

These bits configure the SPI clock and select for A2F060 and A2F500, as shown in Table 13-3.

***Table 13-3* • SPI Clock and Select (A2F060 and A2F500)**

Mode	SPS[1]	SPO	SPH	Clock in Idle	Sample Edge	Shift Edge	Select in Idle	Select Between Frames
MOT	0	0	0	Low	Rising	Falling	High	Pulse between all frames
	0	1	0	High	Falling	Rising	High	
	0	0	1	Low	Falling	Rising	High	Does not pulse between back-to-back frames, does if transmit FIFO empties
	0	1	1	High	Rising	Falling	High	Does not pulse between back-to-back frames, does if transmit FIFO empties
	1	0	0	Low	Rising	Falling	High	Stays active until all the frames set by frame counter have been transmitted
	1	0	1	High	Falling	Rising	High	
	1	1	0	Low	Falling	Rising	High	
	1	1	1	High	Rising	Falling	High	
TI	0	0	0	Low	Falling	Rising	Low	Normal Operation. SPICLK only generated with select and data bits
	–	–	–	Low	Falling	Rising	Low	Removes SLAVE_SELECT on consecutive frames (back-to-back) making them look like big frames
	–	1	–	Running	Falling	Rising	Low	SPI_x_CLK is free running.
NSC	0	0	0	Low	Rising	Falling	High	Normal Operation. SPI_x_CLK only generated with select and data bits
	–	–	1	Low	Rising	Falling	High	Forces IDLE cycles (SLAVE_SELECT deactivated) between back-to-back frames
	–	1	–	Running	Rising	Falling	High	SPI_x_CLK is free running.
	1	–	–	Low	Rising	Falling	High	After sending the command part of the frame, subsequent frames are concatenated to create a single large data frame (master operation only).

Notes:

1. *SPS is the additional control in A2F060 and A2F500.*
2. *The timing diagrams in Figure 13-2 on page 224 through Figure 13-13 on page 230 apply to all devices.*

SPI Register Interface Summary

Table 13-4 summarizes each of the registers covered by this document. There are two addresses for each register; one for each of the SPI controllers in a SmartFusion device.

***Table 13-4* • SPI Register Summary**

Register Name	Address (SPI_0)	Address (SPI_1)	R/W	Reset Value	Description
CONTROL	0x40001000	0x40011000	R/W	0x0000102	Control register
TXRXDF_SIZE	0x40001004	0x40011004	R/W	0x04	Transmit and receive data frame size
STATUS	0x40001008	0x40011008	R	0x2440	Status register
INT_CLEAR	0x4000100C	0x4001100C	W	0x00	Interrupt Clear register
RX_DATA	0x40001010	0x40011010	R	Unknown	Receive Data register
TX_DATA	0x40001014	0x40011014	W	0x00000000	Transmit Data register
CLK_GEN	0x40001018	0x40011018	R/W	0x07	Output Clock Generator (master mode)
SLAVE_SELECT	0x4000101C	0x4001101C	R/W	0x00	Specifies slave selected (master mode)
MIS	0x40001020	0x40011020	R	0x00	Masked interrupt status
RIS	0x40001024	0x40011024	R	0x00	Raw interrupt status
The following registers apply to A2F060 and A2F500 only					
CONTROL2	0x40001028	0x40011028	R/W	0x00	Control bits for enhanced mode
COMMAND	0x4000102C	0x4001102C	R/W	0x00	Command register
PKTSIZE	0x40001030	0x40011030	R/W	0x00	Packet size
Reserved	0x40001034	0x40011034	–	0x00	Reserved. Should not be written.
Reserved	0x40001038	0x40011038	–	0x00	Reserved. Should not be written.
STAT8	0x4000103C	0x4001103C	R	0x44	Status register (reduced width)
CTRL0 (CTRL)	0x40001040	0x40011040	R/W	0x02	Aliased control register – read and write bits 7:0
CTRL1 (CTRL)	0x40001044	0x40011044	R/W	0x01	Aliased control register – read and write bits 15:8
CTRL2 (CTRL)	0x40001048	0x40011048	R/W	0x00	Aliased control register – read and write bits 23:16
CTRL3 (CTRL)	0x4000104C	0x4001104C	R/W	0x00	Aliased control register – read and write bits 25:24

SPI Register Interface Details

This section describes each of the SPI registers in detail.

SPI Control Register (CONTROL)

***Table 13-5* • CONTROL**

Bit Number	Name	R/W	Reset Value	Description
[31:26]	Reserved	R/W	0	(**A2F200 only**) Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
31	RESET	R/W	1	(**A2F060 and A2F500**) 0: The logic clocked by the SPI clocks is enabled 1: The logic clocked by the SPI clocks is held in reset
30	OENOFF	R/W	0	(**A2F060 and A2F500**) 0: SPI output enable active as required 1: The core will not assert the SPI output enable. This allows multiple slaves to be connected to a single master sharing a single slave select and software protocol implemented that can enable the slaves transmit data when a certain broadcast address SPI command is received.
29	BIGFIFO	R/W	0	(**A2F060 and A2F500**) 0: FIFO size is 4 frames 1: FIFO sizes are Frame Size 4–8 Bits = 32 frames 9–16 bits = 16 frames 17–32 bits = 8 frames
28	CLKMODE	R/W	0	(**A2F060 and A2F500**) 0: Sets SPI_x_CLK using PCLK / (2**(CLK_GEN + 1)) where CLK_GEN = 0 to 15 1: Sets SPI_x_CLK using PCLK / (2*(CLK_GEN+ 1) where CLKRATE = 0 to 255
27	FRAMEURUN	R/W	0	(**A2F060 and A2F500**) 0: Under-runs are generated whenever a read is attempted from an empty transmit FIFO. 1: Under-run condition will be ignored for the complete frame if the first data frame read resulted in a potential overflow; that is, the slave was not ready to transmit any data. If the first data frame is read from the FIFO and transmitted, an under-run will be generated if the FIFO becomes empty for any of the remaining packet frames (while SLAVE_SELECT is active). Master operation will never create a transmit FIFO under-run condition

***Table 13-5* • CONTROL (continued)**

Bit Number	Name	R/W	Reset Value	Description
26	SPS	R/W	0	(**A2F060 and A2F500**) Controls SPI signal polarity, etc. Refer to Table 13-3 on page 231.
25	SPH	R/W	0	Clock phase (Table 13-2 on page 223 for A2F200)
24	SPO	R/W	0	Clock polarity
[23:8]	TXRXDFCOUNT	R/W	0x0001	Number of data frames to be sent/received, 1 to 65,535. This register once set will not change. An internal frame counter counts frames until it reaches the limit set by this register. When this register is written, the internal frame counter is also reset. When that counter reaches TXRXDFCOUNT, interrupts are generated and idle cycles inserted on the SPI bus (SLAVE_SELECT deactivated). The core will transmit data whenever the transmit FIFO is loaded. The counter is used to count frames, terminate bursts and generate interrupts
7	INTTXUNRRUN	R/W	0	Interrupt on transmit under-run. If set to 1, interrupt is not masked and will result in interrupt to the Cortex-M3 microcontroller. If set to 0 interrupt is masked.
6	INTRXOVRFLO	R/W	0	Interrupt on receive overflow. If set to 1, interrupt is not masked and will result in interrupt to the Cortex-M3 microcontroller. If set to 0 interrupt is masked.
5	INTTXDATA	R/W	0	Interrupt on transmit data. If set to 1, interrupt is not masked and will result in interrupt to the Cortex-M3 microcontroller. If set to 0, interrupt is masked.
4	INTRXDATA	R/W	0	Interrupt on receive data. If set to 1, interrupt is not masked and will result in interrupt to the Cortex-M3 microcontroller. If set to 0, interrupt is masked.
[3:2]	TRANSFPRTL	R/W	0	Transfer protocol Decode: 0b00 – Motorola SPI 0b01 – TI Synchronous Serial 0b10 – National Semiconductor MICROWIRE 0b11 – Reserved *Note: The transfer protocol cannot be changed while the SPI is enabled.*

Table 13-5 • **CONTROL (continued)**

Bit Number	Name	R/W	Reset Value	Description
1	MODE	R/W	1	SPI implementation 0 – Slave 1 – Master (default)
0	ENABLE	R/W	0	0: Core will not respond to external signals until this bit is enabled. SPISCLKO driven to zero and SPIOEN, SPISS (slave select) driven inactive. 1: Core is active *Note: All the registers are accessible to the Cortex-M3 microcontroller, even when the core is disabled.*

Recommended Start-Up Sequence

Bits in the Control register should be set in the following order for slave operation; this is also recommended for master operation.

1. Write with RESET =1 and ENABLE = 0 and other bits as required
2. Write with RESET = 0 and ENABLE = 0 and other bits as required
3. Write with RESET = 0 and ENABLE = 1 and other bits as required

This sequence prevents spurious clock pulses clocking the SPI slave logic as the clock source is and polarity is selected should the SPICLK be in a non-idle state

SPI TxRx Data Frame Register (TXRXDF_SIZE)

Table 13-6 • **TXRXDF_SIZE**

Bit Number	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[5:0]	TXRXDFS	R/W	0x04	Transmit and receive data size. Maximum value is 32. Number of bits shifted out and received per frame (count starts from 1). In National Semiconductor MICROWIRE mode, this is the number of shifts to be done after the control byte is sent. *Note: This Register must be set before SPI is enabled. Writes to this register are ignored after SPI is enabled.*

SPI Status Register (STATUS)

***Table 13-7* • STATUS**

Bit Number	Name	R/W	Reset Value	Description
[31:15][1]	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	ACTIVE	R	0	(**A2F060 and A2F500 only**) Core is still transmitting or receiving data
13	SSEL	R	0	(**A2F060 and A2F500 only**) Current state of SSEL signal
12	FRAMESTART	R	0	(**A2F060 and A2F500 only**) Indicates that the next frame in the receive FIFO was the first received after SSEL went active; i.e., the start of a packet of frames.
11	TXFIFOEMPNXT	R	0	Transmit FIFO empty on next read
10	TXFIFOEMP	R	1	Transmit FIFO empty
9	TXFIFOFULNXT	R	0	Transmit FIFO full on next write
8	TXFIFOFUL	R	0	Transmit FIFO full
7	RXFIFOEMPNXT	R	0	Receive FIFO empty on next read
6	RXFIFOEMP	R	1	Receive FIFO empty
5	RXFIFOFULNXT	R	0	Receive FIFO full on next write
4	RXFIFOFUL	R	0	Receive FIFO full
3	TXUNDERRUN	R	0	No data available for transmission. The channel cannot read data from the transmit FIFO because the transmit FIFO is empty. In reality this can only be raised in Slave mode because the Master will not attempt to transmit unless there is data in FIFO.
2	RXOVERFLOW	R	0	Channel is unable to write to receive FIFO as it is full. Applies to Master and Slave modes.
1	RXDATRCED	R	0	1 means the number of frames specified by TXRXDFCOUNT (CONTROL register) has been received and can be read. Applies to Master and Slave modes.
0	TXDATSENT	R	0	1 means the numbers of frames specified by TXRXDFCOUNT (CONTROL register) have been sent. Applies to Master and Slave modes.

Notes:

1. *For A2F200, the reserved bits are [31:12].*
2. *Bits [11:4] correspond to FIFO Status.*
3. *None of these status bits are sticky. That means during the run time the status of these bits reflects the current status of SPI.*
4. *To determine the cause of an interrupt, the Masked Interrupt Status register (MIS) needs to read.*

SPI Interrupt Clear Register (INT_CLEAR)

***Table 13-8* • INT_CLEAR**

Bit Number	Name	R/W	Reset Value	Description
[31:6][1]	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSEND	W	0	(**A2F060 and A2F500 only**) Slave select end
4	Reserved	W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXCHUNDRUN	W	0	Transmit channel under-run
2	RXCHOVRFLW	W	0	Receive channel over flow
1	RXRDYCLR	W	0	Clears receive ready (rx_rdy)
0	TXDONECLR	W	0	Clears transmit done (tx_done)

Notes:

1. *For A2F200, the reserved bits are [31:4].*
2. *A read to this register has no effect. It returns all zeroes.*

SPI Receive Data Register (RX_DATA)

***Table 13-9* • RX_DATA**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	RXDATA	R	0	Received data. Reading this clears the register of the received data.

SPI Transmit Data Register (TX_DATA)

***Table 13-10* • TX_DATA**

Bit Number	Name	R/W	Reset Value	Description
[31:0]	TXDATA	W	0	Data to be transmitted. Writing to this clears the last data transmitted.

SPI SCLK Generation Register (CLK_GEN)

***Table 13-11* • CLK_GEN**

Bit Number	Name	R/W	Reset Value	Description
[31:8]*	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[7:0]	SCLKOGEN	R/W	0x07	Specifies the division of incoming PCLK for generation of SPI_x_CLK. **(A2F200 only)** SPI_x_CLK = PCLK / ($2^{(SCLKOGEN +1)}$) where SCLKOGEN = 0 to 15 *Note: For A2F200, CLK_GEN[7:4] are reserved bits.* (**A2F060 and A2F500 only**) The method used to calculate the PCLK divisor is defined by CONTROL Register bit 28 (CLKMODE). CLKMODE = 0: SPI_x_CLK = PCLK / ($2^{(SCLKOGEN +1)}$) where SCLKOGEN = 0 to 15 CLKMODE = 1: SPI_x_CLK = PCLK / (2 x (SCLKOGEN +1)) where SCLKOGEN = 0 to 255

*Note: *For A2F200, the reserved bits are [31:4].*

SPI Slave Select Register (SLAVE_SELECT)

***Table 13-12* • SLAVE_SELECT**

Bit Number	Name	R/W	Reset Value	Description
[31:8]	Reserved	R/W	N/A	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
[7:0]	SLAVESELECT	R/W	0	Specifies the slave selected. Writing 1 to a bit position selects the corresponding slave. SLAVESELECT[7:1] are available at the FPGA fabric interface, while SLAVESELECT[0] is available at the SPI_x_SS[0] pin.* *Note: The slave select output signal is active low.*

*Note: *Currently for the A2F200 device, SPI_0 can only select four slaves. This means only SPI_0_SS[0], SPI_0_SS[3:1] outputs are valid.*

SPI Masked Interrupt Status Register (MIS)

These bits indicate the masked interrupt status by ANDing the interrupt enables in the CONTROL registers with the raw interrupt register. When any of these bits are set, the INTERRUPT output will be active. The bits are cleared by writing to the Interrupt clear register.

***Table 13-13* • MIS**

Bit Number	Name	R/W	Reset Value	Description
[31:6][1]	Reserved	R/W	N/A	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSENDMSKINT	R	0	(**A2F060 and A2F500 only**) Masked status of slave select end. SSENDMSKINT = SSEND and INTEN_SSEND [2]
4	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXCHUNDMSKINT	R	0	Masked status of transmit channel under-run. TXCHUNDMSKINT = TXCHUNDR and INTTXUNRRUN [3]
2	RXCHOVRFMSKINT	R	0	Masked status of receive channel overflow RXCHOVRFMSKINT = RXCHOVRF and INTRXOVRFLO [3]
1	RXRDYMSKINT	R	0	Masked status of receive data ready (data received in FIFO) RXRDYMSKINT = RXRDY and INTRXDATA [3]
0	TXDONEMSKINT	R	0	Masked status of transmit done (data shifted out) TXDONEMSKINT = TXDONE and INTTXDATA [3]

Notes:

1. *For A2F200, the reserved bits are [31:4].*
2. *MIS[5:4] = RIS[5:4] and CONTROL2[5:4].*
3. *MIS[3:0] = RIS[3:0] and CONTROL [7,6,4,5].*

SPI Raw Interrupt Status Register (RIS)

***Table 13-14* • RIS**

Bit Number	Name	R/W	Reset Value	Description
[31:6]*	Reserved	R/W	N/A	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSEND	R	0	(**A2F060 and A2F500 only**) Indicates that SPI_X_SS[0] has gone inactive.
4	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXCHUNDR	R	0	RAW interrupt status. Reading this returns raw interrupt status. Raw status of transmit channel under-run
2	RXCHOVRF	R	0	Raw status of receive channel overflow
1	RXRDY	R	0	Receive data ready (data received in FIFO)
0	TXDONE	R	0	Raw status of transmit done (data shifted out)

Note: For A2F200, the reserved bits are [31:4].

SPI Control Bits for Enhanced Mode (CONTROL2)

This register is applicable to A2F060 and A2F500 devices only.

***Table 13-15* • CONTROL2**

Bits	Name	R/W	Reset	Description
[31:6]	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	INTEN_SSEND	R/W	0	0: No effect 1: Enables the interrupt as SLAVE_SELECT goes High. SPI master and slave modes.
[4:3]	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DISFRMCNT	R/W	0	0: The internal frame counter is active. When the counter reaches the programmed limit it will pause the current SPI transfer inserting idle cycles, and generate the appropriate interrupts. 1: The internal frame counter is NOT active. The core will transmit data until the transmit FIFO empties. TXRXDFCOUNT (CONTROL register) should also be programmed to zero.
[1:0]	Reserved	–	0	Write 00 to these bits.

SPI Command Register (COMMAND)

Use of the command register allows multiple operations to be achieved without the possibility of a CPU interrupt happening.

This register is applicable to A2F060 and A2F500 devices only

Table 13-16 • **COMMAND**

Bits	Name	R/W	Reset	Description
[31:7]	Reserved	R/W		Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	TXNOW	R/W	0	Writing a 1 will clear the TXBUSY bit in slave mode immediately rather than waiting for PKTSIZE frames to be available, telling the master that data is available. This is intended for use when less than the programmed PKTSIZE data frames are being transmitted, removing the requirement to transmit PKTSIZE frames. Will stay set until the first data frame is transmitted.
5	AUTOSTALL	R/W	0	Writing a 1 will cause the master to delay transmission until the transmit FIFO contains the number of frames specified by the PKTSIZE register. This will guarantee that the frames are transmitted with no idle cycles or time gaps between them. The bit will be automatically cleared as soon as the core starts transmitting the frames.
4	CLRFRAMECNT	R/W	0	Writing a 1 will clear the internal frame counter, This bit always reads as zero. The counter is also cleared when the core is disabled, or CTRL1 or CTRL2 are written (the frame count limit is changed).
3	TXFIFORST	R/W	0	Writing a 1 will reset the TX FIFO. This bit always reads as zero.
2	RXFIFORST	R/W	0	Writing a 1 will reset the RX FIFO. This bit always reads as zero.
1	AUTOEMPTY	R/W	0	Writing a 1 will cause the SPI core to automatically discard any further received data until the number of frames requested in TXRXDFCOUNT (CONTROL register) have been received, or in slave mode SELECT_SELECT goes inactive. This bit will stay set until all the frames are complete or the CPU clears it.
0	AUTOFILL	R/W	0	Writing a 1 will cause the SPI core to automatically fill the transmit FIFO with zeros to match the number of frames requested in TXRXDFCOUNT (CONTROL register). Typically the CPU will write the five bytes to the TXDATA register and then set this bit. The CPU now simply reads data from the receive FIFO until the complete set of frames has been read. This bit will stay set until all the frames are complete or the CPU clears it.

SPI Packet Size (PKTSIZE)

This register is applicable to A2F060 and A2F500 only.

***Table 13-17* • PKTSIZE**

Bits	Name	R/W	Reset Value	Description
[31:6]	Reserved	R/W		Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	PKTSIZE	R/W	0	Sets the size of SPI CMD/DATA frame.

SPI Status8 Register (STAT8)

This register allows important status bits to read as a single 8-bit value, which reduces the overhead of checking status bits when an 8-bit processor is being used.

This register is applicable to A2F060 and A2F500 only.

***Table 13-18* • STAT8**

Bit	Name	Equivalent Status Register Bits	Description
[31:8]	Reserved		Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	ACTIVE	14	Core is still transmitting data
6	SSEL	13	Current State of SSEL
5	TXUNDERRUN	3	Transmit FIFO under flowed
4	RXOVFLOW	2	Receive FIFO overflowed
3	TXFULL	8	Transmit FIFO is full, i.e. no space for more data
2	RXEMPTY	6	Receive FIFO is empty (no data available to read
1	DONE	0 and 1	Number of requested frames have been transmitted and received.
0	FIRSTFRAME	12	Next frame in Receive FIFO was first received after SLAVE_SELECT went active (Command frame).

Note: R/W and Reset Value are the same as the corresponding Control register bits.

Control Register Alias

This directly maps the control register to 4-byte wide registers; this reduces the overhead of setting bits in the control register when an 8-bit processor is being used.

This register is applicable to A2F060 and A2F500 only**.**

***Table 13-19* • CTRL**

Address	Bits	Name	Description
0x40	7:0	CTRL0	Same as Control register bits [7:0]
0x44	7:0	CTRL1	Same as Control register bits [15:8] This sets the lower bits of TXRXDFCOUNT (CONTROL register) and clears the internal frame counter.
0x48	7:0	CTRL2	Same as Control register bits [23:16] This sets the upper bits of TXRXDFCOUNT (CONTROL register) and clears the internal frame counter
0x4C	7:0	CTRL3	Same as control register bits [31:24]

Note: R/W and Reset Value are the same as the corresponding Control register bits.

IOMUXes Associated with SPI_x

IOMUXes 0, 1, 2, 3, and 57 to 63 are used to multiplex SPI_0, GPIO, and fabric interface signals to MSSIOBUFs. IOMUXes 8, 9, 10, 11, and 70 to 76 are used to multiplex SPI_1, GPIO, and fabric interface signals to MSSIOBUFs.

IOMUX Description

The MSS contains multiplexers which are involved in the re-use of some MSS related I/O pads and in providing a number of options for multiplexing GPIO, peripheral signals, and fabric interface signals to the I/O pad. IOMUXes are associated with the GPIO block, fabric interface, and all MSS communications peripherals (UARTS 0 and 1, SPI 0 and 1, I^2C 0 and 1, and the Ethernet MAC).

For every reusable MSS I/O pad there is an IOMUX. The IOMUX is intended to provide flexibility in the allocation of MSS I/O pads, so that if the user is not using a particular interface. the corresponding I/O pads can be reallocated to another interface. Also, if certain I/O pads are not being used (or are not present in some devices), the IOMUX allows the signals to be connected within the IOMUX internally. The IOMUX is composed of four multiplexers—M0, M1, M2, and M3—as shown in the "IOMUX Functional Description" section on page 366. The register description is contained in the same document following the IOMUX functional description.

IOMUXes for SPI_x_DI, SPI_x_DO, SPI_x_CLK, and SPI_x_SS[0]

To use the SPI_x_DO, SPI_x_DI, SPI_x_CLK, and SPI_x_SS[0] signals, an IOMUX is used to route the signal to a MSSIOBUF. This IOMUX is used to share the MSSIOBUF between the SPI_x signal and a GPIO. If both the GPIO and the SPI_x signal are needed, another IOMUX can be configured to route the GPIO to the FPGA fabric interface. The GPIN source select register must be configured appropriately as well. For more information about the GPIN source select register, refer to the "General Purpose I/O Block (GPIO)" section on page 319. The IOMUXes can be configured using the MSS configurator. The GPIO signal can be routed to an FPGA I/O using the SmartDesign tool.

Figure 13-14 shows the IOMUX topology for SPI_0_DI, which applies to SPI_1_DI, SPI_x_DO, SPI_x_CLK, and SPI_x_SS[0] as well.

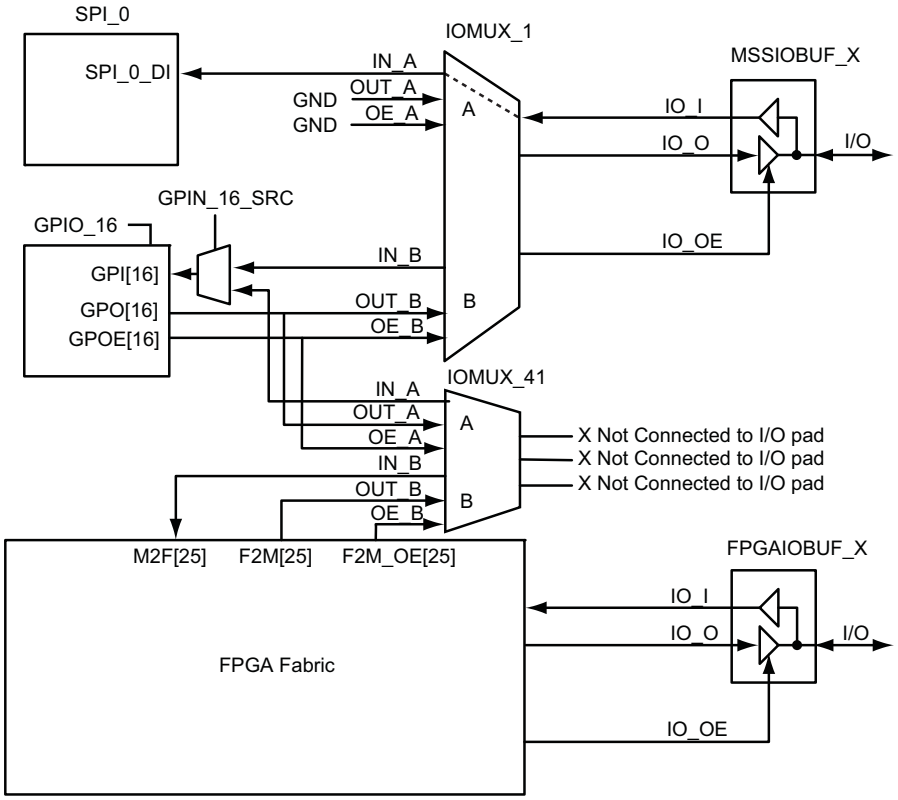


***Figure 13-14* • SPI Signal Interaction with GPIO and Fabric**

In this case, IOMUX_1 is configured to connect IN_A to MSSIOBUF IO_I port. IOMUX_41 can be configured to route the GPIO_16 signals to the fabric interface. The M2F[25], F2M[25], and F2M_OE[25] can then be routed to a FPGAIOBUF using the SmartDesign tool. Similar configuration applies to SPI_1_DI, SPI_x_DO, SPI_x_CLK, and SPI_x_SS[0] as well.

Table 13-20 shows the associated GPIO, IOMUX, and fabric interface signals for these signals.

***Table 13-20* • SPI Signal GPIO and Fabric Mapping**

SPI_x_Signal	MSS GPIO	Fabric Interface	IOMUX
SPI_0_SDO	16	25	0, 41
SPI_0_SDI	17	26	1, 42
SPI_0_CLK	18	27	2, 43
SPI_0_SS[0]	19	28	3, 44
SPI_1_SDO	24	33	8, 49
SPI_1_SDI	25	34	9, 50
SPI_1_CLK	26	35	10, 51
SPI_1_SS[0]	27	36	11, 52

IOMUXes for Extra Slave Select Signals SPI_x_SS[7:1]

To use these SPI_x signals, you must route them to an FPGA I/O through an IOMUX. The MSS configurator in the SmartDesign tool is used to route the SPI_x signal to the FPGA fabric interface (through an IOMUX). Routing the signal from the FPGA fabric interface to a FPGA I/O is performed separately using the SmartDesign tool. Figure 13-15 shows the IOMUX topology for SPI_0_SS[1].

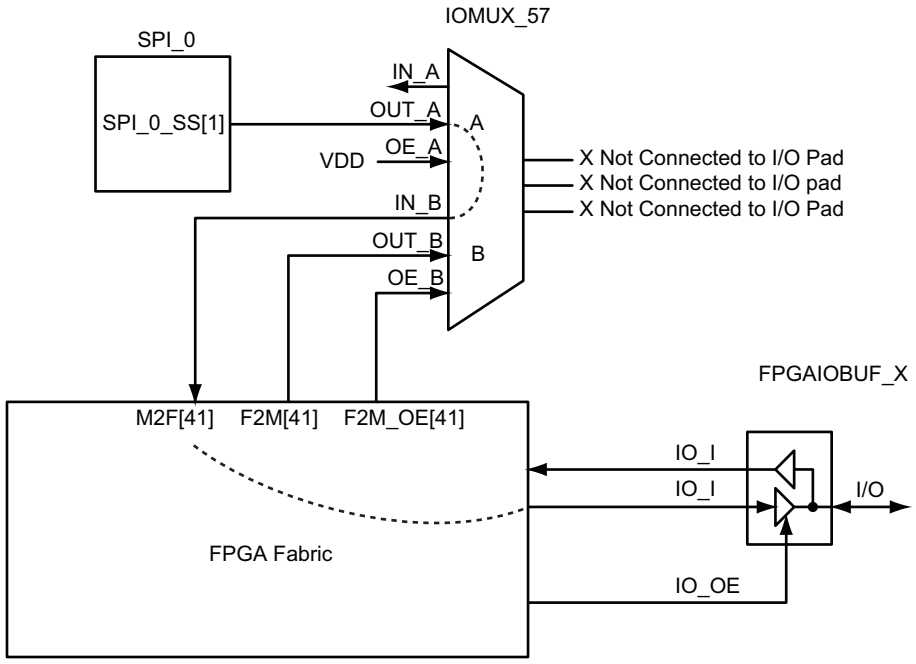


***Figure 13-15* • SPI_x_SS[7:1] Interaction with GPIO and Fabric**

For SPI_0_SS[1], IOMUX_57 is configured using the IOMUX_57_CR register to connect OUT_A to IN_B, which connects SPI_0_SS[1] to M2F[41] in the fabric. The M2F[41] signal can then be connected to a FPGAIOBUF using the SmartDesign tool. Similar configuration applies to SPI_1_SS[1], SPI_x_SS[2], SPI_x_SS[3], SPI_x_SS[4], SPI_x_SS[5], SPI_x_SS[6], and SPI_x_SS[7]. Table 13-21 shows the associated GPIO, IOMUX, and fabric interface signals for these signals.

***Table 13-21* • SPI Extra Signal GPIO and Fabric Mapping**

SPI_x_Signal	Fabric Interface Signal	IOMUX
SPI_0_SS[1]	M2F_41	57
SPI_0_SS[2]	M2F_42	58
SPI_0_SS[3]	M2F_43	59
SPI_1_SS[1]	M2F_54	70
SPI_1_SS[2]	M2F_55	71
SPI_1_SS[3]	M2F_56	72
SPI_1_SS[4]	M2F_57	73
SPI_1_SS[5]	M2F_58	74
SPI_1_SS[6]	M2F_59	75
SPI_1_SS[7]	M2F_60	76

Table 13-22 through Table 13-39 on page 252 give descriptions for all IOMUXes associated with SPI_x signals.

IOMUX 0

***Table 13-22* • IOMUX 0**

Pad Name	Pad Ports	IOMUX_0_CR	IOMUX 0 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_DO/GPIO_16	I					GPI_16		
	O			SPI_0_DO			GPO_16	
	OE				SPI_0_DEN			GPOE_16
	PU	IOMUX_0_PU						
	PD	IOMUX_0_PD						
	ST	IOMUX_0_ST						

IOMUX 1

***Table 13-23* • IOMUX 1**

Pad Name	Pad Ports	IOMUX_1_CR	IOMUX 1 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_DI/GPIO_17	I		SPI_0_DI			GPI_17		
	O			GND			GPO_17	
	OE				GND			GPOE_17
	PU	IOMUX_1_PU						
	PD	IOMUX_1_PD						
	ST	IOMUX_1_ST						

IOMUX 2

***Table 13-24* • IOMUX 2**

Pad Name	Pad Ports	IOMUX_2_CR	IOMUX 2 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_CLK/GPIO_18	I		SPI_0_CLKI			GPI_18		
	O			SPI_0_CLKO			GPO_18	
	OE				SPI_0_MODE			GPOE_18
	PU	IOMUX_2_PU						
	PD	IOMUX_2_PD						
	ST	IOMUX_2_ST						

IOMUX 3

***Table 13-25* • IOMUX 3**

Pad Name	Pad Ports	IOMUX_3_CR	IOMUX 3 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_SS[0]/GPIO_19	I		SPI_0_SSI			GPI_19		
	O			SPI_0_SSO			GPO_19	
	OE				SPI_0_MODE			GPOE_19
	PU	IOMUX_3_PU						
	PD	IOMUX_3_PD						
	ST	IOMUX_3_ST						

IOMUX 8

***Table 13-26* • IOMUX 8**

Pad Name	Pad Ports	IOMUX_8_CR	IOMUX 8 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_DO/GPIO_24	I					GPI_24		
	O			SPI_1_DO			GPO_24	
	OE				SPI_1_DEN			GPOE_24
	PU	IOMUX_8_PU						
	PD	IOMUX_8_PD						
	ST	IOMUX_8_ST						

IOMUX 9

***Table 13-27* • IOMUX 9**

Pad Name	Pad Ports	IOMUX_9_CR	IOMUX 9 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_DI/GPIO_25	I		SPI_1_DI			GPI_25		
	O			GND			GPO_25	
	OE				GND			GPOE_25
	PU	IOMUX_9_PU						
	PD	IOMUX_9_PD						
	ST	IOMUX_9_ST						

IOMUX 10

***Table 13-28* • IOMUX 10**

Pad Name	Pad Ports	IOMUX_10_CR	IOMUX 10 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_CLK/GPIO_26	I		SPI_1_CLKI			GPI_26		
	O			SPI_1_CLKO			GPO_26	
	OE				SPI_1_MODE			GPOE_26
	PU	IOMUX_10_PU						
	PD	IOMUX_10_PD						
	ST	IOMUX_10_ST						

IOMUX 11

***Table 13-29* • IOMUX 11**

Pad Name	Pad Ports	IOMUX_11_CR	IOMUX 11 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[0]/GPIO_27	I		SPI_1_SSI			GPI_27		
	O			SPI_1_SSO			GPO_27	
	OE				SPI_1_MODE			GPOE_27
	PU	IOMUX_11_PU						
	PD	IOMUX_11_PD						
	ST	IOMUX_11_ST						

IOMUX 57

***Table 13-30* • IOMUX 57**

Pad Name	Pad Ports	IOMUX_57_CR	IOMUX 57 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_SS[1]	I					M2F[41]		
	O			SPI_0_SS[1]			F2M[41]	
	OE				VDD			F2M_OE[41]
	PU	IOMUX_57_PU						
	PD	IOMUX_57_PD						
	ST	IOMUX_57_ST						

IOMUX 58

***Table 13-31* • IOMUX 58**

Pad Name	Pad Ports	IOMUX_58_CR	IOMUX 58 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_SS[2]	I					M2F[42]		
	O			SPI_0_SS[2]			F2M[42]	
	OE				VDD			F2M_OE[42]
	PU	IOMUX_58_PU						
	PD	IOMUX_58_PD						
	ST	IOMUX_58_ST						

IOMUX 59

***Table 13-32* • IOMUX 59**

Pad Name	Pad Ports	IOMUX_59_CR	IOMUX 59 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_SS[3]	I					M2F[43]		
	O			SPI_0_SS[3]			F2M[43]	
	OE				VDD			F2M_OE[43]
	PU	IOMUX_59_PU						
	PD	IOMUX_59_PD						
	ST	IOMUX_59_ST						

IOMUX 70

***Table 13-33* • IOMUX 70**

Pad Name	Pad Ports	IOMUX_70_CR	IOMUX 70 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[1]	I					M2F[54]		
	O			SPI_1_SS[1]			F2M[54]	
	OE				VDD			F2M_OE[54]
	PU	IOMUX_70_PU						
	PD	IOMUX_70_PD						
	ST	IOMUX_70_ST						

IOMUX 71

***Table 13-34* • IOMUX 71**

Pad Name	Pad Ports	IOMUX_71_CR	IOMUX 71 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[2]	I					M2F[55]		
	O			SPI_1_SS[2]			F2M[55]	
	OE				VDD			F2M_OE[55]
	PU	IOMUX_71_PU						
	PD	IOMUX_71_PD						
	ST	IOMUX_71_ST						

IOMUX 72

***Table 13-35* • IOMUX 72**

Pad Name	Pad Ports	IOMUX_72_CR	IOMUX 72 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[3]	I					M2F[56]		
	O			SPI_1_SS[3]			F2M[56]	
	OE				VDD			F2M_OE[56]
	PU	IOMUX_72_PU						
	PD	IOMUX_72_PD						
	ST	IOMUX_72_ST						

IOMUX 73

***Table 13-36* • IOMUX 73**

Pad Name	Pad Ports	IOMUX_73_CR	IOMUX 73 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[4]	I					M2F[57]		
	O			SPI_1_SS[4]			F2M[57]	
	OE				VDD			F2M_OE[57]
	PU	IOMUX_73_PU						
	PD	IOMUX_73_PD						
	ST	IOMUX_73_ST						

IOMUX 74

***Table 13-37* • IOMUX 74**

Pad Name	Pad Ports	IOMUX_74_CR	IOMUX 74 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[5]	I					M2F[58]		
	O			SPI_1_SS[5]			F2M[58]	
	OE				VDD			F2M_OE[58]
	PU	IOMUX_74_PU						
	PD	IOMUX_74_PD						
	ST	IOMUX_74_ST						

IOMUX 75

***Table 13-38* • IOMUX 75**

Pad Name	Pad Ports	IOMUX_75_CR	IOMUX 75 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[6]	I					M2F[59]		
	O			SPI_1_SS[6]			F2M[59]	
	OE				VDD			F2M_OE[59]
	PU	IOMUX_75_PU						
	PD	IOMUX_75_PD						
	ST	IOMUX_75_ST						

IOMUX 76

***Table 13-39* • IOMUX 76**

Pad Name	Pad Ports	IOMUX_78_CR	IOMUX 76 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[7]	I					M2F[60]		
	O			SPI_1_SS[7]			F2M[60]	
	OE				VDD			F2M_OE[60]
	PU	IOMUX_76_PU						
	PD	IOMUX_76_PD						
	ST	IOMUX_76_ST						

14 – Inter-Integrated Circuit (I^2C) Peripherals

SmartFusion devices contain two identical master/slave I^2C peripherals that perform serial-to-parallel conversion on data originating from serial devices, and perform parallel-to-serial conversion on data from the ARM Cortex-M3 processor to these devices. The Cortex-M3 embedded processor controls the I^2C peripherals via the APB interface.

Throughout this chapter, a lower case x in register and signal descriptions is used as a place holder for 0 or 1, indicating I2C_0 (on the APB_0 bus) or I2C_1 (on the APB_1 bus).

The I^2C peripherals support I^2C, SMBus, and PMBus data transfers, which conform to the Philips Inter-Integrated Circuit (I^2C) v2.1 specification and support the SMBus v2.0 and PMBus v1.1 specifications. The I^2C peripherals use a 7-bit addressing format and run up to 400 Kbps data rates nominally. Faster rates can be achieved depending on the external load. The I^2C peripherals include the following features:

- Data transfer in multiples of bytes
- Multi-master collision detection and arbitration
- Own slave address and general call address detection
- SMBus timeout and real-time idle condition counters
- Optional SMBus signals, I2C_x_SMBSUS_N and I2C_x_SMBALERT_N, controlled via APB interface

Block Diagram

Figure 14-1 shows the block diagram for the I^2C peripherals.

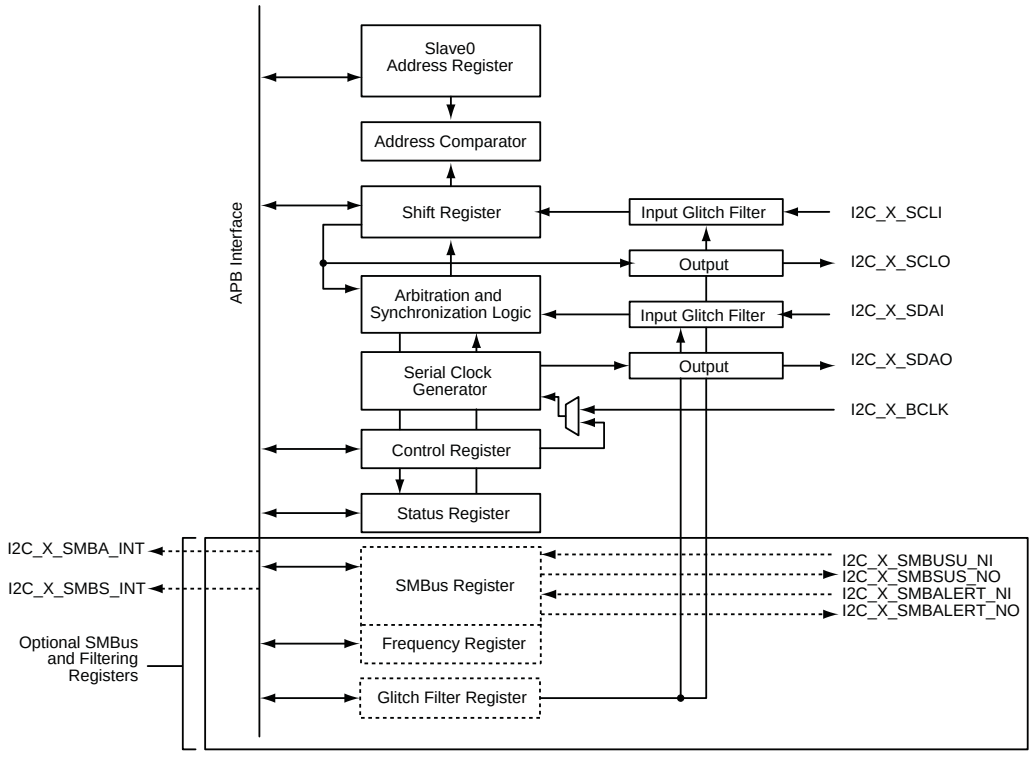


Figure 14-1 • **I^2C Block Diagram**

Functional Description

The Cortex-M3 processor accesses the I^2C peripherals registers via the advanced peripheral bus (APB) interface. The APB registers are defined in the "I2C_x Register Map" on page 259.

Input signals are synchronized with the internal clock, PCLK0 for I2C_0 and PCLK1 for I2C_1. Glitches shorter than the glitch register length are filtered out. The filter length is configurable from 3 to 6 clock periods. Note that the I^2C Fast Mode (400 kbps) specification states that glitches 50 ns or less should be filtered out of the incoming clock and data lines. Refer to the "GLITCHREG Register" section on page 272 for more details.

The address comparator checks the received 7-bit slave address with its own slave address. It also compares the first received 8-bit byte with the general call address (0x00). If a match is found, the STATUS register is updated and an interrupt is requested.

The I^2C peripherals can operate in the following four modes:

1. Master Transmitter mode: Serial data transmitted via SDA; serial clock transmitted via SCL.
2. Master Receiver mode: Serial data received via SDA; serial clock transmitted via SCL.
3. Slave Receiver mode: Serial data and the serial clock received via SDA and SCL.
4. Slave Transmitter mode: Serial data transmitted via SDA; serial clock received via SCL.

The programmable clock pulse generator provides the serial bus clock pulses when the I^2C peripheral is in a master mode. The clock generator is switched off when the I^2C peripheral is in a slave mode. Refer to the "Clocks" section on page 256 for details about the baud rate clock (BCLK).

Slave Mode

After setting the ENS1 bit in the CTRL register, the I^2C peripheral is in the "not addressed slave mode," which means the I^2C peripheral looks for its own slave address and the general call address. If one of these addresses is detected, the peripheral switches to addressed slave mode and an interrupt is requested. Then the peripheral can operate as a slave transmitter or a slave receiver.

Transfer Example

- Cortex-M3 processor sets ENS1 and AA bits.
- I^2C peripheral receives own address and 0.
- I^2C peripheral generates interrupt request; STATUS register = 0x60 (Table 14-8 on page 265).
- Cortex-M3 processor prepares for receiving data and then clears the SI bit.
- I^2C peripheral receives next data byte and then generates interrupt request. The STATUS register contains a value of 0x80 or 0x88, depending on the AA bit (Table 14-8 on page 265).
- Transfer is continued according to Table 14-8 on page 265.

Master Mode

When the Cortex-M3 processor wishes to become the bus master, the I^2C peripheral waits until the serial bus is free. When the serial bus is free, the peripheral generates a start condition, sends the slave address and transfers the direction bit. The peripheral may operate as a master transmitter or as a master receiver depending on the transfer direction bit.

Transfer Example

- Cortex-M3 processor sets ENS1 and STA bits.
- I^2C peripheral sends START condition and then generates interrupt request. STATUS register = 0x80 (Table 14-6 on page 261).
- Cortex-M3 processor writes the DATA register (7-bit slave address and 0) and then clears SI bit.
- I^2C peripheral sends DATA register contents and then generates interrupt request. The STATUS register contains 0x18 or 0x20 value depending on received ACK bit. (Table 14-6 on page 261).
- Transfer is continued according to Table 14-6 on page 261.

While in a master mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the bus. While in Master Transmitter mode, if another device on the bus overrules a logic 1 and

pulls the data line low, arbitration is lost and the I2C peripheral immediately changes from Master Transmitter mode to Slave Receiver mode. The synchronization logic synchronizes the serial clock generator block with the transmitted clock pulses coming from another master device. The arbitration and synchronization logic also utilizes timeout requirements set forth in the SMBus Specification Version 2.0.

Figure 14-2 depicts a PMBus example using the Cortex-M3 processor, I2C, and MSS GPIO. PMBus protocols are run through the serial bus, and the additional PMBus control signal is routed through the MSS GPIO. The firmware to control the PMBus control signals must be written by the user. SMBus devices may also be linked to the same bus as shown.

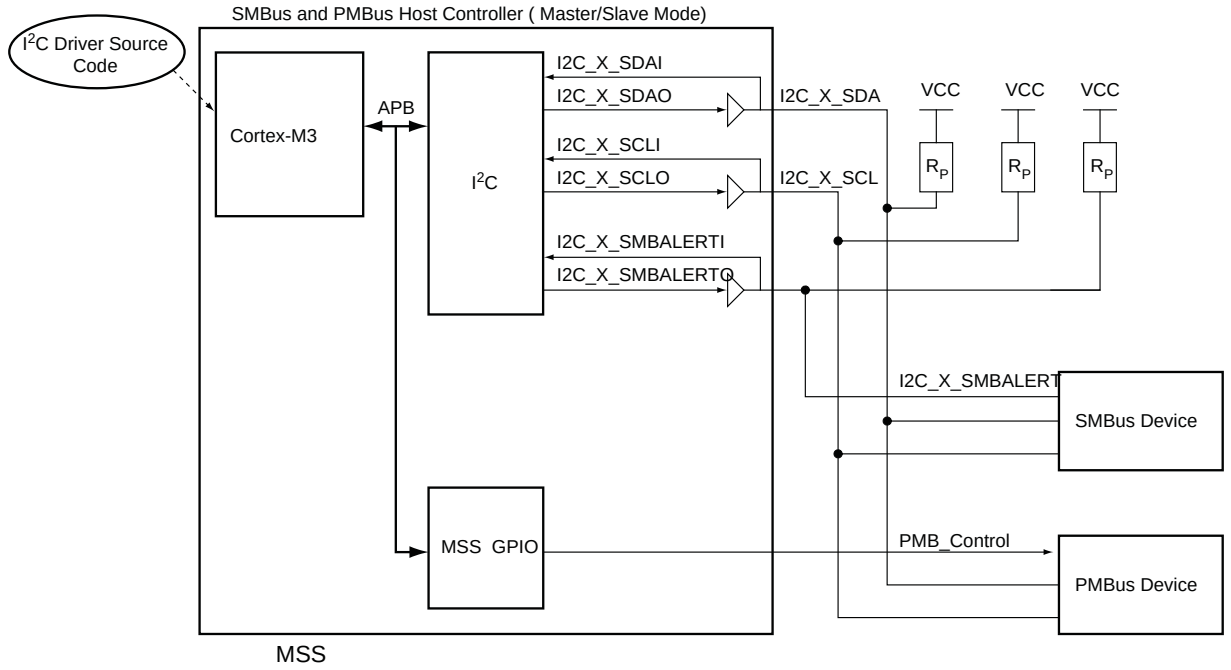


Figure 14-2 • **I2C Application Example**

Clockstretching

The I2C peripheral supports the clock stretching feature as defined in Philips I2C v2.1 Specifications. This is to address situations where an I2C slave is not able to meet the clock speed provide by the I2C master and needs to slow down slightly. An I2C slave is allowed to hold down the clock if it needs to reduce the bus speed. The I2C master reads back the clock signal after releasing it to a High state and waits until the line has actually gone High.

Clockstretching sounds a bit odd but is a common practice. However, the total bandwidth of the shared bus might be significantly decreased. So, especially for I2C buses shared by multiple devices, it is important to estimate the impacts of clockstretching. Do not make the slowest I2C device dominate your bus performance.

Note: When an I2C peripheral is used in SMBus or IPMI modes, there are built-in timeout counters that limit the amount of time the clock can be stretched by a slave. For the SMBus protocol, the clock stretching limit is 35 ms (there is a minimum limit of 10 KHz on the SMBus clock). The SMBus protocol assumes that if something takes too long, there is a problem on the bus and that all devices must reset in order to clear this mode. Similarly, for IPMI there is a 3 ms SCL low timeout. Thus in these modes, slave devices are not allowed to hold the clock Low for too long.

System Dependencies

Clocks

The I2C_0 and I2C_1 peripherals are clocked by PCLK0 on APB Bus 0 and PCLK1 on APB Bus 1, respectively. PLCLK0 and PLCK1 are free running versions of FCLK (the main clock driving the entire MSS) which are derived from the MSS_CCC. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109 for details.

Baud Rate Clock (BCLK)

The baud rate clock (BCLK) is a pulse-for-transmission speed control signal and is internally synchronized with the clock input. BCLK may be used to set the serial clock frequency from a clock sourced within the FPGA fabric when the CR[2:0] bits in the CTRL register are set to 0b111. Otherwise, PCLK0 or PCLK1 is used to determine the serial clock frequency. The actual non-stretched serial bus clock frequency can be calculated based on the setting in the CR[2:0] fields of the CTRL register and the frequencies of PCLK0 or PCLK1 and BCLK. Refer to the "CTRL Register" section on page 260 for the bit settings.

Resets

The I^2C peripherals reset to zero on power-up and are held in reset until the user enables them. The user has the option under software control to reset the I^2C peripherals by writing to bit 11 or bit 12 in the System Register, SOFT_RST_CR, located at address 0xE0042030 in the memory map. The soft resets are encoded in Table 14-1.

***Table 14-1* • Soft Reset Bit Definitions for the I^2C peripherals**

Bit Number	Name	R/W	Reset Value	Description
11	I2C_0_SR	R/W	0x1	Controls reset input to I2C_0 0 – Release I2C_0 from reset 1 – Keep I2C_0 in reset (reset value)
12	I2C_1_SR	R/W	0x1	Controls reset input to I2C_1 0 – Release I2C_1 from reset 1 – Keep I2C_1 in reset (reset value)

At power-up, the reset signals are asserted 1. This keeps the I^2C peripherals in a reset state. If the user sets this bit to 0, the I^2C peripheral is allowed to become active. If I2C_x_SR is 0, the I2C_x peripheral could still be held in reset by other system reset sources. See the "Reset Controller" section on page 143 for more details.

SMBus Clock Low Reset

If the SCL clock line is held low by a master that has initiated a bus reset with the SMBUS register, the following sequence should occur. Refer to Figure 14-3.

SMBus Bus Reset Sequence:

- The master device sets the SMBUS RESET bit, forcing the SCL clock line low.
- The master device enters the reset state, D0, and an interrupt is generated after 35 ms.
- A slave device will enter the reset state, D8, after 25 ms and an interrupt will be generated.
- Once the interrupt is asserted, the APB controller of the slave device needs to clear the interrupt within 10 ms per the SMBus Specification v.2.0, and the slave device enters the idle state, F8.
- After 35 ms, the master device's interrupt will be asserted, and the APB controller of the master device needs to clear the interrupt, forcing the master device into the idle state, F8.

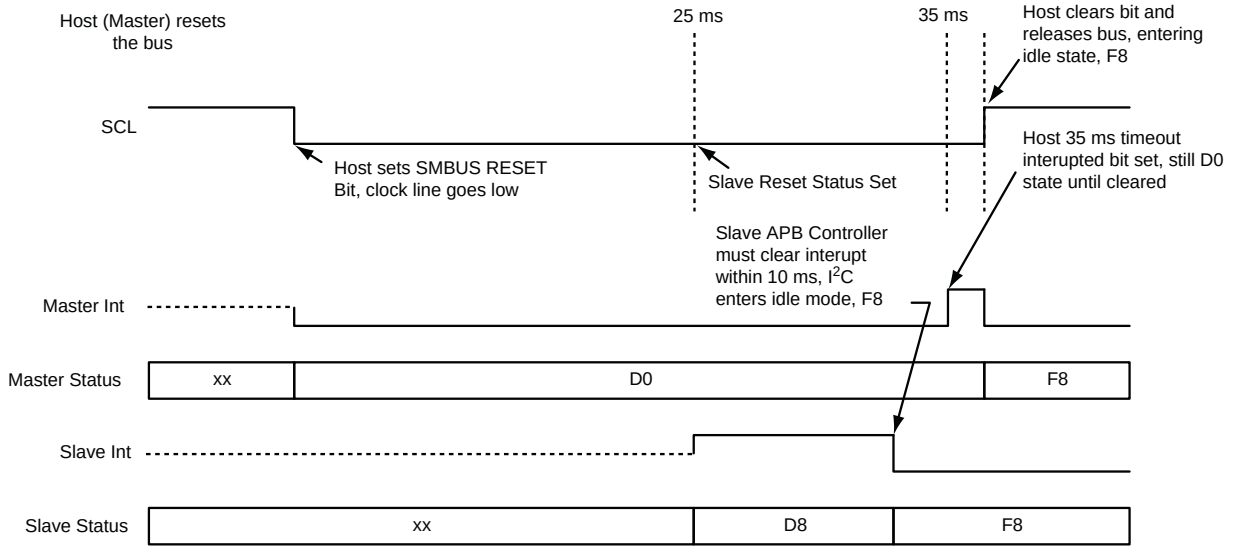


Figure 14-3 • **SMBus Bus Reset Sequence**

Interrupts

There are three interrupt signals from each I^2C peripheral. The I2C_0_INT, I2C_0_SMBALERT, and I2C_0_SMBSUS signals are generated by I2C_0 and are mapped to IRQ 14, IRQ 15, and IRQ 16 in the Cortex-M3 NVIC controller. The I2C_1_INT, I2C_1_SMBALERT, and I2C_1_SMBSUS signals are generated by I2C_1 and are mapped to IRQ 17, IRQ 18, and IRQ 19 in the Cortex-M3 NVIC controller. All interrupt enable bits within the NVIC are located at address 0xE000E100; IRQ 14 through IRQ 19 correspond to bit locations 14 through 19, respectively. The user must also enable SMBus interrupts (I2C_x_SMBALERT and I2C_x_SMBSUS) in the I^2C peripheral by setting the appropriate bits in the SMBUS register. The user must clear the appropriate bit in the SMBus Register in the respective interrupt service routine to prevent a reassertion of the interrupt.

The I2C_x_INT, I2C_x_SMBALERT, and I2C_x_SMBSUS I^2C interrupt signals can be monitored by FPGA logic via the fabric interface interrupt controller (FIIC). These six signals (three for each I^2C) are combined with other interrupt sources and connected to the MSSINT[0] bit. One exception while in non-ACE mode is that the I2C_1_INT interrupt signal is connected to MSSINT[5] bit. Refer to the "Fabric Interface and IOMUX" section on page 343 for more details.

Fabric Interface Signals Associated with I2C_0 and I2C_1

Some I^2C signals are connected directly to the fabric interface so they can be routed to FPGA logic or FPGA I/Os. Table 14-2 lists the I^2C signals available on the fabric interface. Refer to the "Fabric Interface and IOMUX" section on page 343 for more information.

Table 14-2 • **I^2C Fabric Interface Signals**

I2C_0 and I2C_1 Signals		
Name	**Input to / Output from FPGA Fabric**	**Function**
I2C0SMBSUSNO	Input	Output suspend mode signal; used if I^2C is the master/host. *Note: Not a wired-AND signal.*
I2C0SMBALERTNO	Input	Output wired-AND interrupt signal; used in slave/device mode if the I^2C wants to force communication with a host.
I2C0SMBSUSNI	Output	Input suspend mode signal; used if I^2C is slave/device. *Note: Not a wired-AND signal.*
I2C0SMBALERTNI	Output	Input wired-AND interrupt signal; used in master/host mode to monitor whether slave/devices want to force communication with the host.
I2C0BCLK0	Output	Pulse for SCL speed control. Used only if the configuration bits CR[2:0] = 111; otherwise various divisions of PCLK are used.
I2C1SMBSUSNO	Input	Output suspend mode signal; used if I^2C is the master/host. *Note: Not a wired-AND signal.*
I2C1SMBALERTNO	Input	Output wired-AND interrupt signal; used in slave/device mode if the I^2C wants to force communication with a host.
I2C1SMBSUSNI	Output	Input suspend mode signal; used if I^2C is slave/device. *Note: Not a wired-AND signal.*
I2C1SMBALERTNI	Output	Input wired-AND interrupt signal; used in master/host mode to monitor if slave/devices want to force communication with the host.
I2C1BCLK	Output	Pulse for SCL speed control. Used only if the configuration bits cr[2:0] = 111; otherwise various divisions of PCLK are used.

I2C_x Register Map

The I2C_0 base address resides at 0x40002000 and extends to address 0x40002FFF in the Cortex-M3 memory map. The I2C_1 base address resides at 0x40012000 and extends to address 0x40012FFF in the Cortex-M3 memory map.

***Table 14-3* • I2C_x Register Definitions**

Register Name	I2C_0 Address	I2C_1 Address	R/W	Reset Value	Description
CTRL	0x40002000	0x40012000	R/W	0	Used to configure the I^2C peripheral.
STATUS	0x40002004	0x40012004	R	0xF8	Read-only value which indicates the current state of the I^2C peripheral
DATA	0x40002008	0x40012008	R/W	0	Read/write data to/from the serial interface
ADDR	0x4000200C	0x4001200C	R/W	0	Contains the primary programmable address of the I^2C peripheral
SMBUS	0x40002010	0x40012010	R/W	0b01X1X000	Configuration register for SMBus timeout reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N
FREQ	0x40002014	0x40012014	R/W	0x08	Necessary for configuring real-time timeout logic. Can be set to the PCLK frequency for 25 ms SMBus timeouts, or may be changed to increase/decrease the timeout value.
GLITCHREG	0x40002018	0x40012018	R/W	0x03	Number of registers in the glitch filter. Can be set to value from 3 to 6. Correct value to meet I^2C fast mode 50 ns spike suppression will depend on the PCLK frequency.

CTRL Register

***Table 14-4* • CTRL**

Bit Number	Name	R/W	Reset Value	Description
7	CR2	R/W	0	Clock rate bit 2; refer to bit 0.
6	ENS1	R/W	0	Enable bit. When ENS1 = 0, the sda and scl outputs are in a high impedance and sda and scl input signals are ignored. When ENS1 = 1, the I^2C is enabled.
5	STA	R/W	0	The START flag. When STA = 1, the I^2C checks the status of the serial bus and generates a START condition if the bus is free.
4	STO	R/W	0	The STOP flag. When STO = 1 and the I^2C is in a master mode, a STOP condition is transmitted to the serial bus.
3	SI	R/W	0	The Serial Interrupt flag. The SI flag is set by the I^2C whenever there is a serviceable change in the STATUS register. After the register has been updated, the SI bit must be cleared by software. *Note: The SI bit is directly readable via the APB INTERRUPT signal.*
2	AA	R/W	0	The Assert Acknowledge flag. When AA = 1, an acknowledge will be returned when: • The own slave address has been received • The general call address has been received while the gc bit in the Address register is set. • A data byte has been received while the I^2C is in the master receiver mode. • A data byte has been received while the I^2C is in the slave receiver mode. When AA = 0, a not acknowledge will be returned when: • A data byte has been received while the I^2C is in the master receiver mode. • A data byte has been received while I^2C is in the slave receiver mode.
1	CR1	R/W	0	Serial clock rate bit 1. Refer to bit 0.
0	CR0	R/W	0	Serial clock rate bit 0. Clock rate is defined as follows: **CR2 CR1 CR0 SCL Frequency** 0 0 0 PCLK frequency/256 0 0 1 PCLK frequency/224 0 1 0 PCLK frequency/192 0 1 1 PCLK frequency/160 1 0 0 PCLK frequency/960 1 0 1 PCLK frequency/120 1 1 0 PCLK frequency/60 1 1 1 ***BCLK*** frequency/8 *Note: BCLK is synchronized to PCLK and hence must be PCLKFREQ/2 or less.*

STATUS Register

***Table 14-5* • STATUS**

Bit Number	Name	R/W	Reset Value	Description
7:0	STATUS register	R	0xF8	The STATUS register is read-only. The status values depend on the mode of operation and are listed in Table 14-6 through Table 14-10 on page 269. Whenever there is a change of state, INTERRUPT is requested. After updating any registers, the APB interface control must clear the INTERRUPT by clearing the SI bit of the CTRL register.

STATUS Register – Master Transmitter Mode

***Table 14-6* • STATUS Register – Master Transmitter Mode**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W	X	0	0	X	SLA+W will be transmitted; ACK will be received.
		or Load SLA+R	X	0	0	X	SLA+W will be transmitted; Core will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; not ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

***Table 14-6* • STATUS Register – Master Transmitter Mode (continued)**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x28	Data byte in DATA register has been transmitted; ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in DATA register has been transmitted; NACK has been received.	Data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or data bytes.	No action	0	0	0	X	The bus will be released; not-addressed slave mode will be entered.
		or no action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0xD0	SMBus Master Reset has been activated.	No Action	X	X	X	X	Wait 35 ms for interrupt to be set, clear interrupt, and proceed to F8H state.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

STATUS Register – Master Receiver Mode

Table 14-7 • **STATUS Register – Master Receiver Mode**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK will be received.
		or Load SLA+W	X	0	0	X	SLA+W will be transmitted; F2DSS_I2C will be switched to MST/TRX mode.
0x38	Arbitration lost in not ACK bit.	No action	0	0	0	X	The bus will be released; F2DSS_I2C will enter slave mode.
		or no action	1	0	0	X	A start condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No action	0	0	0	0	Data byte will be received; not ACK will be returned.
		or no action	0	0	0	1	Data byte will be received; ACK will be returned.
0x48	SLA+R has been transmitted; not ACK has been received.	No action	1	0	0	X	Repeated START condition will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte	0	0	0	0	Data byte will be received; not ACK will be returned.
		or read data byte	0	0	0	1	Data byte will be received; ACK will be returned.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

Table 14-7 • **STATUS Register – Master Receiver Mode (continued)**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x58	Data byte has been received; not ACK has been returned.	Read data byte	1	0	0	X	Repeated START condition will be transmitted.
		or read data byte	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		or read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0xD0	SMBus Master Reset has been activated.	No Action	X	X	0	X	Wait 35 ms for interrupt to be set, clear interrupt, and proceed to F8H state.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

STATUS Register – Slave Receiver Mode

***Table 14-8* • STATUS Register – Slave Receiver Mode**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No action	X	0	0	0	Data byte will be received and not ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; own SLA+W has been received; ACK returned.	No action	X	0	0	0	Data byte will be received and not ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (00H) has been received; ACK has been returned.	No action	X	0	0	0	Data byte will be received and not ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; general call address has been received; ACK returned.	No action	X	0	0	0	Data byte will be received and not ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV address; DATA has been received; ACK returned.	Read data byte	X	0	0	0	Data byte will be received and not ACK will be returned.
		or read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

***Table 14-8* • STATUS Register – Slave Receiver Mode (continued)**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x88	Previously addressed with own SLA; DATA byte has been received; not ACK returned.	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with general call address; DATA has been received; ACK returned.	Read data byte	X	0	0	0	Data byte will be received and not ACK will be returned.
		or read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x98	Previously addressed with general call address; DATA has been received; NACK returned.	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

***Table 14-8* • STATUS Register – Slave Receiver Mode (continued)**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xD8	25 ms SCL low time has been reached; device must be reset	no action	X	X	0	X	Slave must proceed to reset state by clearing the interrupt within 10ms, according to SMBus Specification 2.0.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

STATUS Register – Slave Transmitter Mode

***Table 14-9* • Status Register – Slave Transmitter Mode**

Status Code	Status	DATA Register Action	CTRL register bits				Next Action Taken by Core
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB8	Data byte has been transmitted; ACK has been received.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xC0	Data byte has been transmitted; not ACK has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

***Table 14-9* • Status Register – Slave Transmitter Mode (continued)**

Status Code	Status	DATA Register Action	CTRL register bits				Next Action Taken by Core
			STA	STO	SI	AA	
0xC8	Last data byte has transmitted; ACK has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xD8	25 ms SCL low time has been reached; device must be reset.	no action	X	X	0	X	Slave must proceed to reset state by clearing the interrupt within 10ms, according to SMBus Specification 2.0.

Notes:

1. *SLA = Slave address*
2. *SLV = Slave*
3. *REC = Receiver*
4. *TRX = Transmitter*
5. *SLA+W = Master sends slave address then writes data to slave*
6. *SLA+R = Master sends slave address then reads data from slave*

STATUS Register – Miscellaneous States

***Table 14-10* • STATUS Register – Miscellaneous States**

Status Code	Status	DATA Register Action	CTRL Register Bits				Next Action Taken by Core
			STA	STO	SI	AA	
0x38	Arbitration lost	No action	0	0	0	X	Bus will be released
		or no action	1	0	0	X	A start condition will be transmitted when the bus becomes free.
0xF8	No relevant state information available; SI = 0.	No Action	No Action				Idle
0x00	Bus error during MST or selected slave modes.	No action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the state switched in non-addressed slave mode. Stop Flag is reset.

DATA Register

The DATA register contains a byte of serial data to be transmitted or a byte that has just been received. The Cortex-M3 processor can read from and write to this 8-bit, directly addressable register while it is not in the process of shifting a byte (after an interrupt has been generated).

The bit descriptions are listed below in both data and addressing context. Data context is the 8-bit data format from MSB to LSB. Addressing context is based on a master sending an address call to a slave on the bus, along with a direction bit (master transmit data or receive data from a slave).

***Table 14-11* • DATA**

Bit Number	Name	R/W	Reset Value	Description
7	sd7	R/W	0	Data context: serial data bit 7 (MSB) Addressing context: serial address bit 6 (MSB)
6	sd6	R/W	0	Data context: serial data bit 6 Addressing context: serial address bit 5
5	sd5	R/W	0	Data context: serial data bit 5 Addressing context: serial address bit 4
4	sd4	R/W	0	Data context: serial data bit 4 Addressing context: serial address bit 3
3	sd3	R/W	0	Data context: serial data bit 3 Addressing context: serial address bit 2
2	sd2	R/W	0	Data context: serial data bit 2 Addressing context: serial address bit 1
1	sd1	R/W	0	Data context: serial data bit 1 Addressing context: serial address bit 0 (LSB)
0	DIR	R/W	0	Data context: serial data bit 0 (LSB) Addressing context: direction bit. 0 = Write; 1 = Read

ADDR Register

***Table 14-12* • ADDR**

Bit Number	Name	R/W	Reset Value	Description
7	adr6	R/W	0	Own slave address bit 6
6	adr5	R/W	0	Own slave address bit 5
5	adr4	R/W	0	Own slave address bit 4
4	adr3	R/W	0	Own slave address bit 3
3	adr2	R/W	0	Own slave address bit 2
2	adr1	R/W	0	Own slave address bit 1
1	adr0	R/W	0	Own slave address bit 0
0	GC	R/W	0	General call address acknowledge. If the gc bit is set, the general call address is recognized; otherwise it is ignored.

SMBUS Register

***Table 14-13* • SMBUS**

Bit Number	Name	R/W	Reset Value	Description
7	SMBus Reset	R/W	0	Writing a one to this bit will force the clock line low until 35 ms has been exceeded, thus resetting the entire bus as per the SMBus Specification Version 2.0. Usage: When the I^2C is used as a host controller (master), the user can decide to reset the bus by holding the clock line low 35 ms. Slaves must react to this event and reset themselves.
6	SMBSUS_NO Control	R/W	0b1	SMBSUS_NO control; used in master/host mode to force other devices into power-down/suspend mode. Active low. *Note: SMBSUS_NO and SMBSUS_NI are separate signals (not wired-AND). If the I^2C is part of a host-controller, SMBSUS_NO could be used as an output; if I^2C is a slave to a host-controller that has implemented SMBSUS_N, then only SMBSUS_NI's status would be relevant.*
5	SMBSUS_NI Status	R	0bX	Status of SMBSUS_NI signal. *Note: SMBSUS_NO and SMBSUS_NI are separate signals (not wired-AND). If the I^2C is part of a host-controller, SMBSUS_NO could be used as an output; if I^2C is a slave to a host-controller that has implemented SMBSUS_N, then only SMBSUS_NI's status would be relevant.*
4	SMBALERT_NO Control	R/W	0b1	SMBALERT_NO control; used in slave/device mode to force communication with the master/host. Wired-AND. Status of SMBALERT_NI signal. Wired-AND.
3	SMBALERT_NI Status	R	0bX	Status of SMBALERT_NI signal. Wired-AND.
2	SMBus Enable	R/W	0	0 = SMBus timeouts and status logic disabled (standard I^2C bus operation) 1 = SMBus timeouts and status logic enabled.
1	SMBUS Interrupt Enable	R/W	0	0 = SMBUS Interrupt signal (SMBS) disabled. 1 = SMBUS Interrupt signal (SMBS) enabled.
0	SMBALERT Interrupt Enable	R/W	0	0 = SMBALERT Interrupt signal (SMBA) disabled. 1 = SMBALERT Interrupt signal (SMBA) enabled.

FREQ Register

***Table 14-14* • FREQ**

Bit Number	Name	R/W	Reset Value	Description
7:0	FREQ	R/W	0x08	PCLKx frequency in MHz from 1 to 255. If the PCLKx frequency is used, and SMBus is enabled, the SMBus timeouts will be configured per the SMBus specification. If another timeout value is desired, scale the Frequency value per the following formula: Timeout scale = Fscale/Factual If the actual PCLKx frequency is 100 Mhz, and a scale down is desired that results in a 3 ms timeout rather than 25 ms timeout, then: Fscale = 3/25 x Factual = 0.12 x 100 = 12 Mhz Writing 12 into the FREQ register will have the effect of reducing the maximum timeout count value and thus reducing the real-time timeout from 25 ms to 3 ms.

GLITCHREG Register

***Table 14-15* • GLITCHREG**

Bit Number	Name	R/W	Reset Value	Description
7:0	GLITCHREG	R/W	0x03	The number of registers in the glitch filter can be set to a value from 3 to 6. Correct value to meet I^2C fast mode 50 ns spike suppression will depend on the PCLK frequency. Guideline: **PCLK Freq. (MHz)** — **GlitchReg Value for 50 ns Spike Suppression** Freq. ≤ 40 — 3 40 < Freq. ≤ 60 — 4 60 < Freq. ≤ 80 — 5 80 < Freq. ≤ 100 — 6

IOMUXes Associated with I2C_0 and I2C_1

IOMUXes 6, 7, 47, and 48 are used to multiplex I2C_0, GPIO, and fabric interface signals to MSSIOBUFs. IOMUXes 14, 15, 55, and 56 are used to multiplex I2C_1, GPIO, and fabric interface signals to MSSIOBUFs. Refer to the "Fabric Interface and IOMUX" section on page 343 for a more thorough description of how IOMUXes operate.

IOMUXes for I2C_x_SDAI, I2C_x_SDAO, I2C_x_SCLI, and I2C_x_SCLO

To use the I2C_x_SDAI, I2C_x_SDAO, I2C_x_SCLI, and I2C_x_SCLO signals, an IOMUX is used to route the signals to an MSSIOBUF. This IOMUX is used to share the MSSIOBUF between the I^2C signals and a GPIO.

Figure 14-4 shows the IOMUX topology for I2C_0_SDAI and I2C_0_SDAO, which applies to I2C_0_SCLI and I2C_0_SCLO, I2C_1_SDAI and I2C_1_SDAO, and I2C_1_SCLI and I2C_1_SCLO as well.

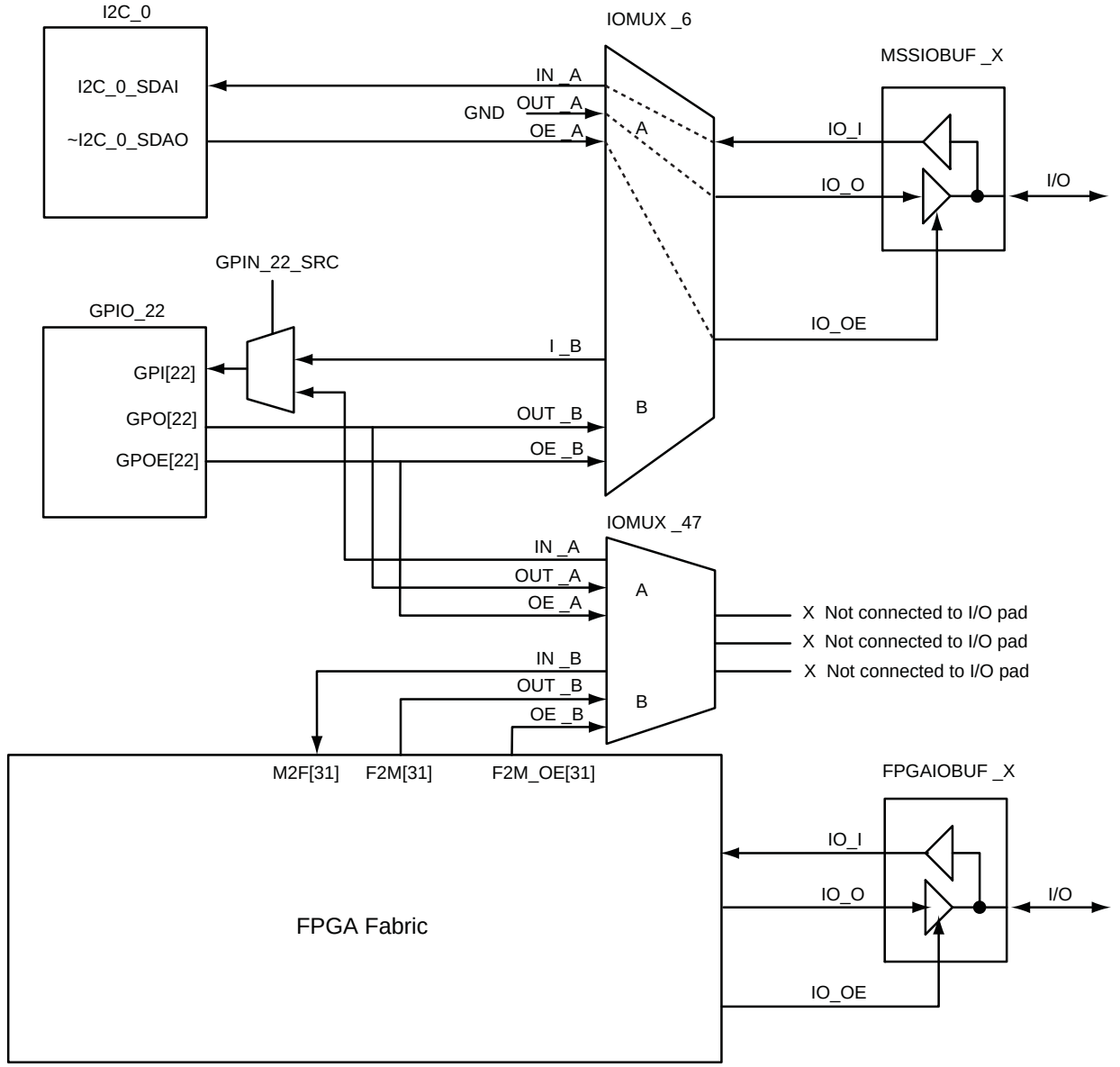


***Figure 14-4* • I2C_0_SDAI and I2C_0_SDAO IOMUX Topology**

In this case, IOMUX_6 is configured to connect all three interface A ports (IN_A, OUT_A, and OE_A) to the MSSIOBUF. IOMUX_47 can be configured to route the GPIO_22 signals to the fabric interface. The M2F[31], F2M[31], and F2M_OE[31] can then be routed to an FPGAIOBUF using the Libero tool. Similar configuration applies to I2C_0_SCLI and I2C_0_SCLO, I2C_1_SDAI and I2C_1_SDAO, and I2C_1_SCLI and I2C_1_SCLO.

When utilizing the MSSIOBUF for I^2C SDA and SCL signals (instead of MSS GPIO) the IOMUXes are configured for open-drain operation. To achieve this, the inverse of the I^2C output signal (I2C_x_SDAO or I2C_x_SCLO) is fed into the IOMUX OE_A port. The OUT_A port of the IOMUX (which is an input port into the IOMUX) is tied to GND. The I^2C input signal (I2C_x_SDAI or I2C_x_SCLI) is driven by the IOMUX IN_A port (which is an output port on the IOMUX).

Table 14-16 shows I^2C signal connections

***Table 14-16* • I2C_x_SDAI, I2C_x_SDAO, I2C_x_SCLI, and I2C_x_SCLO Signal Connections**

I^2C Signals	MSS GPIO	Fabric Interface	IOMUX
I2C_0_SDAI I2C_0_SDAO	22	31	6, 47
I2C_0_SCLI I2C_0_SCLO	23	32	7, 48
I2C_1_SDAI I2C_1_SDAO	30	39	14, 55
I2C_1_SCLI I2C_1_SCLO	31	40	15, 56

Table 14-17 through Table 14-24 on page 276 provide descriptions for all IOMUXes associated with I2C_0 and I2C_1 signals.

IOMUX 6

***Table 14-17* • IOMUX 6**

Pad Name	Pad Ports	IOMUX_6_CR	IOMUX 6 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_0_SDA/GPIO_22	I		I2C_0_SDAI			GPI_22		
	O			GND			GPO_22	
	OE				~(I2C_0_SDAO)			GPOE_22
	PU	IOMUX_6_PU						
	PD	IOMUX_6_PD						
	ST	IOMUX_6_ST						

IOMUX 7

***Table 14-18* • IOMUX 7**

Pad Name	Pad Ports	IOMUX_7_CR	IOMUX 7 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_0_SCL/GPIO_23	I		I2C_0_SCLI			GPI_23		
	O			GND			GPO_23	
	OE				~(I2C_0_SCLO)			GPOE_23
	PU	IOMUX_7_PU						
	PD	IOMUX_7_PD						
	ST	IOMUX_7_ST						

IOMUX 47

Table 14-19 • IOMUX 47

Pad Name	Pad Ports	IOMUX_47_CR	IOMUX 47 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_22/ioUXWbYvZ	I		GPI_22			M2F[31]		
	O			GPO_22			F2M[31]	
	OE				GPOE_22			F2M_OE[31]
	PU	IOMUX_47_PU						
	PD	IOMUX_47_PD						
	ST	IOMUX_47_ST						

IOMUX 48

Table 14-20 • IOMUX 48

Pad Name	Pad Ports	IOMUX_48_CR	IOMUX 48 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_23/ioUXWbYvZ	I		GPI_23			M2F[32]		
	O			GPO_23			F2M[32]	
	OE				GPOE_23			F2M_OE[32]
	PU	IOMUX_48_PU						
	PD	IOMUX_48_PD						
	ST	IOMUX_48_ST						

IOMUX 14

Table 14-21 • IOMUX 14

Pad Name	Pad Ports	IOMUX_14_CR	IOMUX 14 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_1_SDA/GPIO_30	I		I2C_1_SDAI			GPI_30		
	O			GND			GPO_30	
	OE				~(I2C_1_SDAO)			GPOE_30
	PU	IOMUX_14_PU						
	PD	IOMUX_14_PD						
	ST	IOMUX_14_ST						

IOMUX 15

Table 14-22 • IOMUX 15

Pad Name	Pad Ports	IOMUX_15_CR	IOMUX 15 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_1_SCL/GPIO_31	I		I2C_1_SCLI			GPI_31		
	O			GND			GPO_31	
	OE				~(I2C_1_SCLO)			GPOE_31
	PU	IOMUX_15_PU						
	PD	IOMUX_15_PD						
	ST	IOMUX_15_ST						

IOMUX 55

Table 14-23 • IOMUX 55

Pad Name	Pad Ports	IOMUX_55_CR	IOMUX 55 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_30/ioUXWbYvZ	I		GPI_30			M2F[39]		
	O			GPO_30			F2M[39]	
	OE				GPOE_30			F2M_OE[39]
	PU	IOMUX_55_PU						
	PD	IOMUX_55_PD						
	ST	IOMUX_55_ST						

IOMUX 56

Table 14-24 • IOMUX 56

Pad Name	Pad Ports	IOMUX_56_CR	IOMUX 56 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_31/ioUXWbYvZ	I		GPI_31			M2F[40]		
	O			GPO_31			F2M[40]	
	OE				GPOE_31			F2M_OE[40]
	PU	IOMUX_56_PU						
	PD	IOMUX_56_PD						
	ST	IOMUX_56_ST						

15 – Universal Asynchronous Receiver/Transmitter (UART) Peripherals

SmartFusion devices contain two identical universal asynchronous receiver/transmitter peripherals that provide software compatibility with the popular 16550 device. They perform serial to-parallel conversion on data originating from modems or other serial devices, and perform parallel-to-serial conversion on data from the ARM Cortex-M3 processor to these devices.

Throughout this chapter, a lower case x in register and signal descriptions is used as a place holder for 0 or 1, indicating UART_0 or UART_1.

Block Diagram

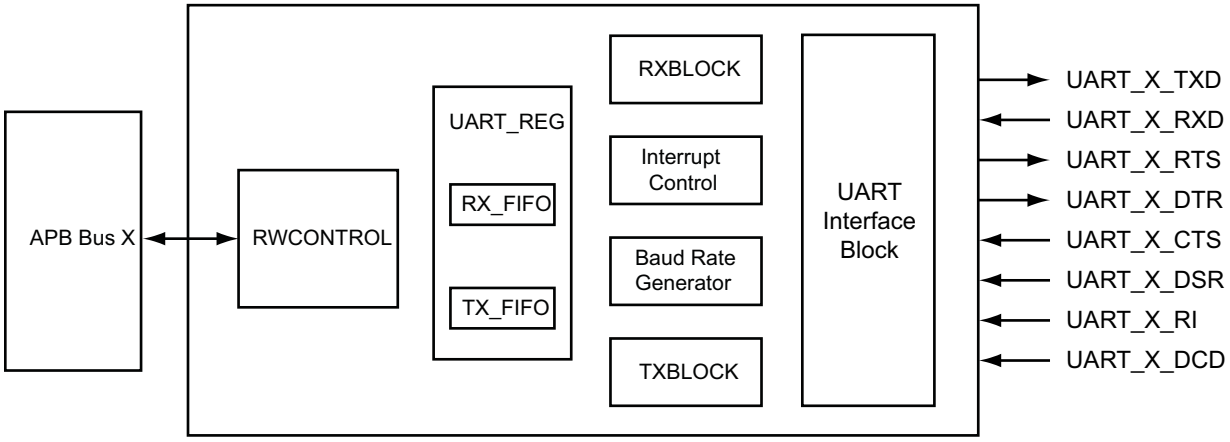


***Figure 15-1* • UART Block Diagram**

Functional Description

When transmitting, data is written in parallel into the transmit FIFO of the UART. The data is then transmitted in serial form. When receiving data, the UART transforms the serial input data into a parallel form to facilitate reading by the Cortex-M3 processor.

The data width is programmable to 5, 6, 7, or 8 bits. The UART also supports various parity settings including even, odd, and no-parity as well as different stop bits including 1, 1½, and 2 bits. If the incoming word is error free, it is placed in the receiver RX_FIFO.

The interrupt control block sends an interrupt signal back to the Cortex-M3 processor, depending on the state of the FIFO and its received and transmitted data. The Interrupt Identification Register (IIR) provides the level of the interrupt. Interrupts are sent for empty transmission/receipt buffers (or FIFOs), an error in receiving a character, or other conditions requiring the attention of the processor.

The UART_0 and UART_1 peripherals are clocked by PCLK0 on APB Bus 0 and PCLK1 on APB Bus 1, respectively. PLCLK0 and PLCK1 are free running versions of FCLK (the main clock driving the entire MSS) which are derived from the MSS_CCC. See the "Clocking Resources Available to the SmartFusion FPGA Fabric" section in the *SmartFusion FPGA Fabric User's Guide* for additional information.

The baud rate generator block takes the input PCLK (PCLK0 on APB_0 and PCLK1 on APB_1) and divides it by a programmed value (from 1 to 2^{16} – 1). The result is divided by 16 to create the transmission clock (BAUDOUT).

System Dependencies

Resets

UART_x resets to zero on power-up and is held in reset until the user enables it. The user has the option under software control to reset the UART_x by writing to bit 7 or bit 8 of the SOFT_RST_CR located at address 0xE0042030 in the system memory map, as shown in Table 15-1.

***Table 15-1* • Soft Reset Bit Definition for the UARTs**

Bit Number	Name	R/W	Reset Value	Description
8	UART_1_SR	R/W	0x1	Controls reset input to UART_1 0 = Release UART_1 from reset 1 = Keep UART_1 in reset (reset value)
7	UART_0_SR	R/W	0x1	Controls reset input to UART_0 0 = Release UART_0 from reset 1 = Keep UART_0 in reset (reset value)

At power-up this signal is asserted as 1. This keeps UART_x in a reset state. If the user sets this bit to 0, UART_x is allowed to become active. If UART_x_SR is 0, UART_x could still be held in reset by other system reset sources. See the "Reset Controller" section on page 143 for more details.

Interrupts

There is one interrupt signal from each UART peripheral. The UART_0_INTR signal is generated by UART_0 and is mapped to IRQ10 in the Cortex-M3 NVIC Controller. The UART_1_INTR signal is generated by UART_1 and is mapped to IRQ11 in the Cortex-M3 NVIC controller. Both interrupt enable bits within the NVIC are located at address 0xE000E100; IRQ10 and IRQ11 correspond to bit locations 10 and 11 respectively. The user must also enable interrupts in the UART_x by setting the appropriate bits in the IER register while the divisor latch access bit (DLAB), bit 7 of LCR, is 0.

The user must clear the appropriate bit in the IER in the respective interrupt service routine to prevent a re-assertion of the interrupt.

UART_x Register Map

The UART_0 base address resides at 0x40000000 and extends to address 0x40000FFF in the Cortex-M3 memory map. The UART_1 base address resides at 0x40010000 and extends to address 0x40010FFF in the Cortex-M3 memory map. Table 15-2 defines the control and status registers for the UARTs, and Table 15-3 on page 280 through Table 15-15 on page 287 give bit definitions for each of the registers.

***Table 15-2* • UART_x Register Definitions**

Register Name	Divisor Latch Access Bit	UART_0 Address	UART_1 Address	R/W	Reset Value	Description
RBR	0	0x40000000	0x40010000	R	–	Receive Buffer Register
THR	0	0x40000000	0x40010000	W	–	Transmit Holding Register
DLR	1	0x40000000	0x40010000	R/W	0x01	Divisor Latch (LSB) Register
DMR	1	0x40000004	0x40010004	R/W	0	Divisor Latch (MSB) Register
IER	0	0x40000004	0x40010004	R/W	0	Interrupt Enable Register
IIR	–	0x40000008	0x40010008	R	0xC1	Interrupt Identification Register
FCR	–	0x40000008	0x40010008	W	0	FIFO Control Register
LCR	–	0x4000000C	0x4001000C	R/W	0	Line Control Register
MCR	–	0x40000010	0x40010010	R/W	0	Modem Control Register
LSR	–	0x40000014	0x40010014	R	0x60	Line Status Register
MSR	–	0x40000018	0x40010018	R	0	Modem Status Register
SR	–	0x4000001C	0x4001001C	R/W	0	Scratch Register

Receive Buffer Register (RBR)

***Table 15-3* • RBR**

Bit Number	Name	R/W	Reset Value	Description
7:0	RBR	R	–	This register holds the receive data bits for UART_x. The reset value is unknown since the register is loaded with data in the Receive FIFO. Bit 0 is the LSB and is the first bit received. The divisor latch access bit (DLAB), bit 7 of LCR, must be 0 to read this register. This register is read only; writing to this register with the DLAB 0 changes the THR register value.

Transmit Holding Register (THR)

***Table 15-4* • THR**

Bit Number	Name	R/W	Reset Value	Description
7:0	THR	W	–	This register holds the data bits to be transmitted. Bit 0 is the LSB and is transmitted first. The reset value is unknown since the register is loaded with data in the Transmit FIFO. The divisor latch access bit (DLAB), bit 7 of LCR, must be 0 to write to this register. This register is write only; reading from this register with the DLAB 0 reads the RBR register value.

Divisor Latch (LSB) Register (DLR)

***Table 15-5* • DLR**

Bit Number	Name	R/W	Reset Value	Description
7:0	DLR	R/W	0x01	This register holds the LSB of the divisor value used to calculate the baud rate. The baud rate (BR) clock is generated by dividing the input reference clock (PCLK_0 for UART_0 and PCLK_1 for UART_1) by 16 and the divisor value, as shown in EQ 1. $$BR = \frac{PCLK\_x}{16 \times \text{divisor value}}$$ *EQ 1* The divisor latch access bit (DLAB) (bit 7 of LCR) must be 1 to access this register.

Divisor Latch (MSB) Register (DMR)

***Table 15-6* • DMR**

Bit Number	Name	R/W	Reset Value	Description
7:0	DMR	R/W	0	This register holds the MSB of the divisor value used to calculate the baud rate. The baud rate (BR) clock is generated by dividing the input reference clock (PCLK_0 for UART_0 and PCLK_1 for UART_1) by 16 and the divisor value, as shown in EQ 2. $BR = \frac{PCLK\_x}{16 \times \text{divisor value}}$ *EQ 2* The divisor latch access bit (DLAB) (bit 7 of LCR) must be 1 to access this register.

Interrupt Enable Register (IER)

***Table 15-7* • IER**

Bit Number	Name	R/W	Reset Value	Description
7:4	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EDSSI	R/W	0	Modem status interrupt enable 0 = Disabled (default) 1 = Enabled
2	ELSI	R/W	0	Receiver line status interrupt enable 0 = Disabled (default) 1 = Enabled
1	ETBEI	R/W	0	Transmit holding register empty interrupt enable 0 = Disabled (default) 1 = Enabled
0	ERBFI	R/W	0	Receive data available interrupt enable 0 = Disabled (default) 1 = Enabled The divisor latch access bit (DLAB) (bit 7 of LCR) must be 0 to access this register.

Interrupt Identification Register (IIR)

***Table 15-8* • IIR**

Bit Number	Name	R/W	Reset Value	Description
7:6	Mode	R	0b11	Always 0b11. Enables FIFO mode.
5:4	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	Interrupt identification bits	R	0b0001	0b0110 = Highest priority. Receiver line status interrupt due to overrun error, parity error, framing error, or break interrupt. Reading the Line Status Register resets this interrupt. 0b0100 = Second priority. Receive data available interrupt modem status interrupt. Reading the Receiver Buffer Register (RBR) or the FIFO drops below the trigger level resets this interrupt. 0b1100 = Second priority. Character timeout indication interrupt occurs when no characters have been read from the RX FIFO during the last four character times and there was at least one character in it during this time. Reading the Receive Buffer Register (RBR) resets this interrupt. 0b0010 = Third priority. Transmit Holding Register Empty interrupt. Reading the IIR or writing to the Transmit Holding Register (THR) resets the interrupt. 0b0000 = Fourth priority. Modem status interrupt due to Clear to Send, Data Set Ready, Ring Indicator, or Data Carrier Detect being asserted. Reading the Modem Status Register resets this interrupt. This register is read only; writing has no effect. Also see Table 15-9.

***Table 15-9* • Interrupt Identification Bit Values**

IIR Value	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0b0110	Highest	Receiver line status	Overrun error, parity error, or break interrupt	Reading the Line Status Register
0b0100	Second	Received data available	Receiver data available	Reading the Receiver Buffer register or the FIFO drops below the trigger level
0b1100	Second	Character timeout indication	No characters have been read from the RX FIFO during the last four character times and there was at least one character in it during this time.	Reading the Receiver Buffer register
0b0010	Third	Transmitter Holding register empty	Transmitter Holding Register empty	Reading the IRR or writing into the Transmitter Holding register
0b0000	Fourth	Modem status	Clear to Send, Data Set Ready, Ring Indicator, or Data Carrier Detect	Reading the Modem Status register

FIFO Control Register (FCR)

***Table 15-10* • FCR**

Bit Number	Name	R/W	Reset Value	Description
7:6	RX_TRIG	W	0b11	These bits are used to set the trigger level for the RX FIFO interrupt. 0b00 = 1 byte 0b01 = 4 bytes 0b10 = 8 bytes 0b11 = 14 bytes (default)
5:4	Reserved	W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ENABLE_TXRDY_RXRDY	W	1	Software must always set this bit to 1 for efficient data transfer from transmit FIFO to PDMA.
2	CLEAR_TX_FIFO	W	0	Clears all bytes in TX FIFO and resets counter logic. The transmit shift register is not cleared. 0 = Disabled (default) 1 = Enabled
1	CLEAR_RX_FIFO	W	0	Clears all bytes in RX FIFO and resets counter logic. The receive shift register is not cleared. 0 = Disabled (default) 1 = Enabled
0	Reserved	W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Line Control Register (LCR)

***Table 15-11* • LCR**

Bit Number	Name	R/W	Reset Value	Description
7	DLAB	R/W	0	Divisor latch access bit. Enables access to the divisor latch registers during read or write operation to address 0 and 1. 0 = Disabled (default) 1 = Enabled
6	SB	R/W	0	Set break. Enabling this bit sets SOUT to 0. This does not have any effect on the transmitter logic. 0 = Disabled (default) 1 = Enabled
5	SP	R/W	0	Stick parity 0 = Disabled (default) 1 = Enabled When stick parity is enabled, the parity bit is set according to bits [4:3] as follows: 11 = 0 is sent as parity bit and checked when receiving. 01 = 1 is sent as parity bit and checked when receiving.
4	EPS	R/W	0	Even parity select 0 = Odd parity (default) 1 = Even parity
3	PEN	R/W	0	Parity enable. When enabled, parity is added to transmission and checked when receiving. 0 = Disabled (default) 1 = Enabled
2	STB	R/W	0	Number of stop bits (STB). 0 = 1 stop bit (default) 1 = 1 ½ stop bits when WLS = 00 The number of stop bits is 2 for all other cases not described above (STB = 1 and WLS = 01, 10 or 11)
1:0	WLS	R/W	0	Word length select 0b00 = 5 bits (default) 0b01 = 6 bits 0b10 = 7 bits 0b11 = 8 bits

Modem Control Register (MCR)

***Table 15-12* • MCR**

Bit Number	Name	R/W	Reset Value	Description
7:5	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation
4	LOOP	R/W	0	Loop enable bit. In Loop mode, SOUT is set to 1 and the SIN, DSRn, CTSn, RIn, and DCDn inputs are disconnected. The output of the Transmitter is looped back into the Receiver. The modem control outputs (DTRn, RTSn, OUT1n, and OUT2n) are connected internally to the modem control inputs, and the modem control output pins are set at 1. In loopback mode, the transmitted data is immediately received, allowing the CPU to check the operation of the UART. The interrupts are operating in loop mode. 0 = Disabled (default) 1 = Enabled
3	OUT2	R/W	0	Controls the Output2 (OUT2n) signal. Active low. 0 = OUT2n is 1 (default) 1 = OUT2n is 0
2	OUT1	R/W	0	Controls the Output1 (OUT1n) signal. Active low. 0 = OUT1n is 1 (default) 1 = OUT1n is 0
1	RTS	R/W	0	Controls the Request to Send (RTSn) signal. Active low. 0 = RTSn is 1 (default) 1 = RTSn is 0
0	DTR	R/W	0	Data Terminal Ready (DTRn) signal. Active low. 0 = DTRn is 1 (default) 1 = DTRn is 0

Line Status Register (LSR)

***Table 15-13* • LSR**

Bit Number	Name	R/W	Reset Value	Description
7	FIER	R	0	This bit is set when there is at least one parity error, framing error, or break indication in the FIFO. FIER is cleared when the Cortex-M3 processor reads the LSR if there are no subsequent errors in the FIFO.
6	TEMT	R	1	Transmit empty. This bit is set to 1 when both the transmitter FIFO and shift registers are empty.
5	THRE	R	1	Transmitter holding register empty. THRE causes an interrupt to the Cortex-M3 processor when bit 1 (ETBEI) in the Interrupt Enable Register is 1. This bit is set when the TX FIFO is empty. It is cleared when at least one byte is written to the TX FIFO.
4	BI	R	0	Break interrupt. Indicates that the receive data is at 0 longer than a full word transmission time (start bit + data bits + parity + stop bits). BI is cleared when the CPU reads the Line Status Register. This error is revealed to the Cortex-M3 processor when its associated character is at the top of the FIFO. When break occurs, only one zero character is loaded into the FIFO.
3	FE	R	0	Framing error. Indicates that the receive byte did not have a valid stop bit. FE is cleared when the CPU reads the Line Status Register. The UART will try to resynchronize after a framing error. To do this, it assumes that the framing error was due to the next start bit, so it samples this start bit twice, and then starts receiving the data. This error is revealed to the CPU when its associated character is at the top of the FIFO.
2	PE	R	0	Parity error. Indicates that the receive byte had a parity error. PE is cleared when the CPU reads the Line Status Register. This error is revealed to the CPU when its associated character is at the top of the FIFO.
1	OE	R	0	Overrun error. Indicates that the new byte was received before the CPU read the byte from the receive buffer, and that the earlier data byte was destroyed. OE is cleared when the CPU reads the Line Status Register. If the data continues to fill the FIFO beyond the trigger level, an overrun error will occur once the FIFO is full and the next character has been completely received in the shift register. The character in the shift register is overwritten, but it is not transferred to the FIFO.
0	DR	R	0	Data ready. Indicates when a data byte has been received and stored in the receive buffer or the FIFO. DR is cleared to 0 when the CPU reads the data from the receive buffer or the FIFO.

Modem Status Register (MSR)

***Table 15-14* • MSR**

Bit Number	Name	R/W	Reset Value	Description
7	DCD	R	0	Data carrier detect. The complement of DCDn input. When bit 4 of the MCR is set to 1 (LOOP), this bit is equivalent to OUT2 in the MCR.
6	RI	R	0	Ring indicator. The complement of the RIn input. When bit 4 of the MCR is set to 1 (LOOP), this bit is equivalent to OUT1 in the MCR.
5	DSR	R	0	Data set ready. The complement of the DSR input. When bit 4 of the MCR is set to 1 (LOOP), this bit is equivalent to RTSn in the MCR.
4	CTS	R	0	Clear to send. The complement of the CTSn input. When bit 4 of the Modem Control Register (MCR) is set to 1 (LOOP), this bit is equivalent to DTR in the MCR.
3	DDCD	R	0	Delta data carrier detect. Indicates that DCD input has changed state. Whenever bit 0, 1, 2, or 3 is set to 1, a modem status interrupt is generated.
2	TERI	R	0	Trailing edge of ring indicator. Indicates that RI input has changed from 0 to 1. Whenever bit 0, 1, 2, or 3 is set to 1, a modem status interrupt is generated.
1	DDSR	R	0	Delta data set ready. Indicates that the DSRn input has changed state since the last time it was read by the CPU. Whenever bit 0, 1, 2, or 3 is set to 1, a modem status interrupt is generated.
0	DCTS	R	0	Delta clear to send. Indicates that the CTSn input has changed state since the last time it was read by the CPU. Whenever bit 0, 1, 2, or 3 is set to 1, a modem status interrupt is generated.

Scratch Register (SR)

***Table 15-15* • SR**

Bit Number	Name	R/W	Reset Value	Description
7:0	SR	R/W	0	Scratch Register. This register has no effect on UART operation.

IOMUXes Associated with UART_x

IOMUXes 4, 5, 12, and 13 are used to multiplex UART_x transmit and receive signals with GPIOs to MSSIOBUFs. IOMUXes 64-69 and 77-82 are used to multiplex modem control signals to fabric interface signals for further connection to FPGA IO. Refer to the "Fabric Interface and IOMUX" section on page 343 for a more thorough description of how IOMUXes operate.

IOMUXes for UART_x_TXD and UART_x_RXD

To use the UART_x_TXD and UART_x_RXD signals, an IOMUX is used to route the signal to an MSSIOBUF. This IOMUX is used to share the MSSIOBUF between the UART_x signal and a GPIO. As an example, Figure 15-2 shows the IOMUX topology for UART_0_TXD. Similar topologies apply to UART_0_RXD, UART_1_RXD, and UART_1_TXD. For UART_0_TXD, IOMUX_4 would be configured to connect OUT_A to MSSIOBUF IO_O port. IOMUX_45 can be configured to route the GPIO_20 signals to the fabric interface. The M2F[29], F2M[29], and F2M_OE[29] signals can then be routed to an FPGAIOBUF using the Libero tool.

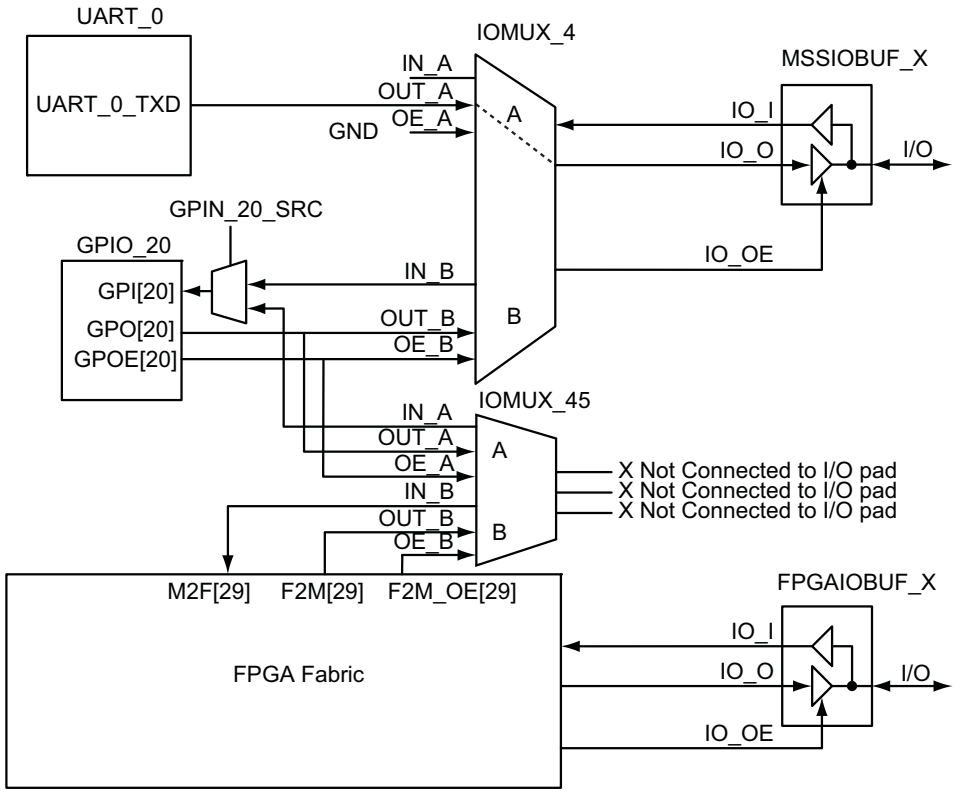


Figure 15-2 • **UART_0_TXD IOMUX Topology**

IOMUXes for Modem Control Signals

To use these UART_x signals, they must be routed to an FPGA IO through an IOMUX. The MSS configurator in SmartDesign is used to route the UART_x signal to the FPGA fabric interface (through an IOMUX) by initializing the contents of the IOMUX_n_CR registers. Routing the signal from the FPGA fabric interface to an FPGA I/O is performed separately using the Libero tool.

As an example, in Figure 15-3 on page 289, for UART_0_RTS, IOMUX_64 is configured using the IOMUX_64_CR register to connect OUT_A to IN_B, which connects UART_0_RTS to M2F[48] in the fabric. The M2F[48] signal can then be connected to a FPGAIOBUF using the Libero tool. Similar IOMUX

arrangements apply to UART_1_RTS, UART_x_DTR, UART_x_CTS, UART_x_DSR, UART_x_RI, and UART_x_DCD as well.

Table 15-16 shows the associated GPIO, IOMUX, and fabric interface signals for each UART_x signal. The MSS configurator tool manages the assignment and allocation of IOMUXes in a graphical user friendly way. The output of the MSS configurator is used by the system boot code to initialize the MSS to a known state.

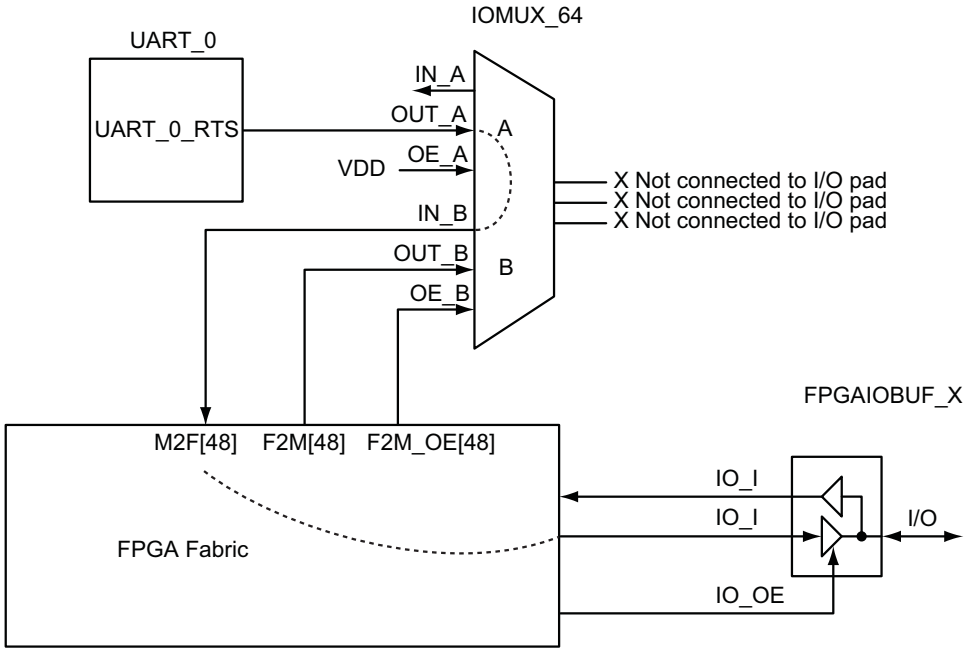


***Figure 15-3* • UART_0_RTS Signal IOMUX Topology**

***Table 15-16* • UART_x Fabric Interface Signal Connections**

UART_x Signal	Fabric Interface Signal	IOMUX
UART_0_RTS	M2F_48	64
UART_0_DTR	M2F_49	65
UART_0_CTS	F2M_50	66
UART_0_DSR	F2M_51	67
UART_0_RI	F2M_52	68
UART_0_DCD	F2M_53	69
UART_1_RTS	M2F_61	77
UART_1_DTR	M2F_62	78
UART_1_CTS	F2M_63	79
UART_1_DSR	F2M_64	80
UART_1_RI	F2M_65	81
UART_1_DCD	F2M_66	82

Table 15-17 through Table 15-32 on page 295 describe the IOMUXes associated with UART_x signals.

IOMUX 4

***Table 15-17* • IOMUX 4**

Pad Name	Pad Ports	IOMUX_4_CR	IOMUX 4 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_TXD/GPIO_20	I					GPI_20		
	O			UART_0_TXD			GPO_20	
	OE				GND			GPOE_20
	PU	IOMUX_4_PU						
	PD	IOMUX_4_PD						
	ST	IOMUX_4_ST						

IOMUX 5

***Table 15-18* • IOMUX 5**

Pad Name	Pad Ports	IOMUX_5_CR	IOMUX 5 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_RXD/GPIO_21	I		UART_0_RXD			GPI_21		
	O			GND			GPO_21	
	OE				GND			GPOE_21
	PU	IOMUX[5]_PU						
	PD	IOMUX[5]_PD						
	ST	IOMUX[5]_ST						

IOMUX 64

***Table 15-19* • IOMUX 64**

Pad Name	Pad Ports	IOMUX_64_CR	IOMUX 64 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_RTS	I					M2F[48]		
	O			UART_0_RTS			F2M[48]	
	OE				VDD			F2M_OE[48]
	PU	IOMUX_64_PU						
	PD	IOMUX_64_PD						
	ST	IOMUX_64_ST						

IOMUX 65

***Table 15-20* • IOMUX 65**

Pad Name	Pad Ports	IOMUX_65_CR	IOMUX 65 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_DTR	I					M2F[49]		
	O			UART_0_DTR			F2M[49]	
	OE				VDD			F2M_OE[49]
	PU	IOMUX_65_PU						
	PD	IOMUX_65_PD						
	ST	IOMUX_65_ST						

IOMUX 66

***Table 15-21* • IOMUX 66**

Pad Name	Pad Ports	IOMUX_66_CR	IOMUX 66 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_CTS	I		UART_0_CTS			M2F[50]		
	O			GND			F2M[50]	
	OE				GND			F2M_OE[50]
	PU	IOMUX_66_PU						
	PD	IOMUX_66_PD						
	ST	IOMUX_66_ST						

IOMUX 67

***Table 15-22* • IOMUX 67**

Pad Name	Pad Ports	IOMUX_67_CR	IOMUX 67 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_DSR	I		UART_0_DSR			M2F[51]		
	O			GND			F2M[51]	
	OE				GND			F2M_OE[51]
	PU	IOMUX_67_PU						
	PD	IOMUX_67_PD						
	ST	IOMUX_67_ST						

IOMUX 68

***Table 15-23* • IOMUX 68**

Pad Name	Pad Ports	IOMUX_68_CR	IOMUX 68 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_RI	I		UART_0_RI			M2F[52]		
	O			GND			F2M[52]	
	OE				GND			F2M_OE[52]
	PU	IOMUX_68_PU						
	PD	IOMUX_68_PD						
	ST	IOMUX_68_ST						

IOMUX 69

***Table 15-24* • IOMUX 69**

Pad Name	Pad Ports	IOMUX_69_CR	IOMUX 69 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_DCD	I		UART_0_DCD			M2F[53]		
	O			GND			F2M[53]	
	OE				GND			F2M_OE[53]
	PU	IOMUX_69_PU						
	PD	IOMUX_69_PD						
	ST	IOMUX_69_ST						

IOMUX 12

***Table 15-25* • IOMUX 12**

Pad Name	Pad Ports	IOMUX_12_CR	IOMUX 12 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_TXD/GPIO_28	I					GPI_28		
	O			UART_1_TXD			GPO_28	
	OE				GND			GPOE_28
	PU	IOMUX_12_PU						
	PD	IOMUX_12_PD						
	ST	IOMUX_12_ST						

IOMUX 13

***Table 15-26* • IOMUX 13**

Pad Name	Pad Ports	IOMUX_13_CR	IOMUX 13 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_RXD/GPIO_29	I		UART_1_RXD			GPI_29		
	O			GND			GPO_29	
	OE				GND			GPOE_29
	PU	IOMUX_13_PU						
	PD	IOMUX_13_PD						
	ST	IOMUX_13_ST						

IOMUX 77

***Table 15-27* • IOMUX 77**

Pad Name	Pad Ports	IOMUX_77_CR	IOMUX 77 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_RTS	I					M2F[61]		
	O			UART_1_RTS			F2M[61]	
	OE				VDD			F2M_OE[61]
	PU	IOMUX_77_PU						
	PD	IOMUX_77_PD						
	ST	IOMUX_77_ST						

IOMUX 78

***Table 15-28* • IOMUX 78**

Pad Name	Pad Ports	IOMUX_78_CR	IOMUX 78 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_DTR	I					M2F[62]		
	O			UART_1_DTR			F2M[62]	
	OE				VDD			F2M_OE[62]
	PU	IOMUX_78_PU						
	PD	IOMUX_78_PD						
	ST	IOMUX_78_ST						

IOMUX 79

***Table 15-29* • IOMUX 79**

Pad Name	Pad Ports	IOMUX_79_CR	IOMUX 79 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_CTS	I		UART_1_CTS			M2F[63]		
	O			GND			F2M[63]	
	OE				GND			F2M_OE[63]
	PU	IOMUX_79_PU						
	PD	IOMUX_79_PD						
	ST	IOMUX_79_ST						

IOMUX 80

***Table 15-30* • IOMUX 80**

Pad Name	Pad Ports	IOMUX_80_CR	IOMUX 80 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_DSR	I		UART_1_DSR			M2F[64]		
	O			GND			F2M[64]	
	OE				GND			F2M_OE[64]
	PU	IOMUX_80_PU						
	PD	IOMUX_80_PD						
	ST	IOMUX_80_ST						

IOMUX 81

***Table 15-31* • IOMUX 81**

Pad Name	Pad Ports	IOMUX_81_CR	IOMUX 81 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_RI	I		UART_1_RI			M2F[65]		
	O			GND			F2M[65]	
	OE				GND			F2M_OE[65]
	PU	IOMUX_81_PU						
	PD	IOMUX_81_PD						
	ST	IOMUX_81_ST						

IOMUX 82

***Table 15-32* • IOMUX 82**

Pad Name	Pad Ports	IOMUX_82_CR	IOMUX 82 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_DCD	I		UART_1_DCD			M2F[66]		
	O			GND			F2M[66]	
	OE				GND			F2M_OE[66]
	PU	IOMUX_82_PU						
	PD	IOMUX_82_PD						
	ST	IOMUX_82_ST						

SmartFusion MSS UART Application Development

This section provides overview of the design flow to facilitate seamless UART application development using SmartFusion devices. Using the MSS UART peripherals does not require any hardware configuration in the MSS configurator. Configuration of UART peripherals using the Microsemi UART drivers is required. The user application would then use predefined functions in the UART drivers to perform application-specific serial transmit and receive tasks.

User application code can be developed and debugged using any of the three supported embedded software development tools: SoftConsole, Keil Microcontroller Development Kit (MDK) µVision,® and IAR Embedded Workbench.® Microsemi provides a set of UART drivers that can be generated from the MSS configurator or from the Firmware Catalog. These drivers are common for all three tool flows. However, the Cortex Microcontroller Software Interface Standard (CMSIS) access layer is dependent on the tool flow selected and should be chosen based on the specific tool flow the user is implementing.

The MSS UART drivers allow rapid application code development using the SmartFusion MSS UART without having to manually read and write the MSS System Registers to send and receive serial data. MSS UART drivers are efficient and flexible, allowing UART to be used in either polled mode or in interrupt driven mode. For specific details on drivers, refer to the SmartFusion MSS Configurators and Drivers User's Guides.

Application Development Using MSS UART Drivers

As a first step, tool-specific CMSIS files must be imported into the project along with MSS UART drivers.The *mss_uart.h* header file, which defines UART function prototypes, must be included in the application code to get access to the UART functions, as shown below:

Example

```
#include "mss_uart.h"
```

The drivers define structure type mss_uart_instance_t to hold UART data. Predefined instance names for UART_0 and UART_1 are *g_mss_uart_0* and *g_mss_uart_1,* respectively. These names should be used in all the functions to refer to UART instances.

Before using an instance of MSS UART, it must be initialized and configured to match the serial line communication parameters required by the system. These parameters include the baud rate, the number of data bits, the parity setting, and the number of stop bits. The function MSS_UART_init() must be used, as shown in the example below.

Example

```
MSS_UART_init (
&g_mss_uart0, // Instantiating UART_0
MSS_UART_57600_BAUD,
MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY |
MSS_UART_ONE_STOP_BIT);
```

Transmission and reception of serial data using MSS UART can be done in either polled method or interrupt driven method. Applications in which transmit or receipt of serial data is a primary function can typically use the polled method. In contrast, applications in which the Cortex-M3 microcontroller performs various tasks will benefit from using the interrupt driven method. This prevents the processor from idling cycles, polling for transmit ready or received serial data.

Polled Transmission Method

The function MSS_UART_polled_tx() is used for polled transmission. This function returns when the full content of the transmit data buffer, defined as one of the arguments, has been transferred to the UART's transmit FIFO. The following example shows usage of this function:

Example

```
MSS_UART_polled_tx // UART in polled mode
( &g_mss_uart0, // UART instance being addressed
greeting_msg, // pointer to transmit data buffer
sizeof(greeting_msg)); // size of the data to be transmitted
```

Interrupt Driven Transmission Method

The function MSS_UART_irq_tx() is used for interrupt driven transmission. This function returns immediately after transmit data buffer location (address) is stored and transmit interrupts are enabled on both the MSS UART peripheral and the Cortex-M3 Interrupt Controller (NVIC).

It is important to note that before the transmit data buffer can be cleared or overwritten, the user must confirm that the UART data has been moved from the data buffer to the UART's transmit FIFO by calling the function MSS_UART_tx_complete(). This function returns zero when the transfer is complete. The following example shows usage of these two companion functions:

Example

```
MSS_UART_irq_tx(// UART in Interrupt Driven Mode
&g_mss_uart0, // UART Instance being addressed
tx_buff, // Pointer to transmit data buffer
sizeof(tx_buff) ); // Size of the data to be transmitted.

// Check whether UART_0 data has been moved to TX FIFO
while(0 == MSS_UART_tx_complete(&g_mss_uart0 )
{;}
```

Polled Receive Method

In the polled receive mode, the MSS UART driver function MSS_UART_get_rx() is called in a loop or at a regular interval to check for data received in the RX FIFO of the MSS UART. The function returns the number of bytes that were copied into the receive data buffer and returns 0 if no data has been received. The following example shows usage of this function:

Example

```
While(1){
rx_size = MSS_UART_get_rx (
&g_mss_uart0, // UART Instance
rx_buff,  // Pointer to receive data buffer
sizeof(rx_buff) // Size of the data receive buffer
);}
```

Interrupt Driven Receive Method

In interrupt driven receive mode, the user must register the RX Interrupt Service Routine (ISR), also called known as Interrupt Handler Function, so that the appropriate function can be called when the UART Receive Data Available (RDA) interrupt occurs. Within the user defined ISR, the MSS_UART_get_rx() function is called to access received data. The following section explains the process.

Registering ISR (receive handler)

The function MSS_UART_set_rx_handler() is used to register the user defined ISR function. This function also enables the UART Received Data Available (RDA) interrupt and the corresponding UART interrupt in the Cortex-M3 NVIC as part of its implementation. The trigger level argument sets the number of bytes that must be received before UART issues the RDA Interrupt. The example below shows usage of the handler function:

Example

```
MSS_UART_set_rx_handler(
&g_mss_uart0, //UART Instance being addressed
uart0_rx_handler, // Pointer to the user defined receive handler function
MSS_UART_FIFO_SINGLE_BYTE); // Trigger level for RDA interrupt firing
```

Example

The following example shows a typical user defined ISR function:

```
void uart0_rx_handler( void )
{
uint8_t rx_buff[RX_BUFF_SIZE];
uint32_t rx_size = 0;

rx_size = MSS_UART_get_rx( &g_mss_uart0, rx_buff, sizeof(rx_buff) );

process_rx_data( rx_buff, rx_size );//User Defined Function
}
```

The SmartFusion MSS UART application development overview provided in this section should enable the user to rapidly and easily adopt the SmartFusion MSS UART peripherals in their embedded application. For additional details on the three tool flows and UART examples, refer to the SmartFusion UART tutorials available at http://www.microsemi.com/soc/products/smartfusion/docs.aspx.

16 – Real-Time Counter (RTC) System

The real-time counter (RTC) system enables SmartFusion devices to support both Standby and Time Keeping modes of operation, greatly reducing power consumption in many applications. The RTC system comprises the following four blocks that work together to provide this increased functionality and reduced power consumption:

- RTC
- Low-power 32 KHz crystal oscillator
- Battery switching circuit
- MSS interface

Figure 16-1 shows these blocks and how they are connected.

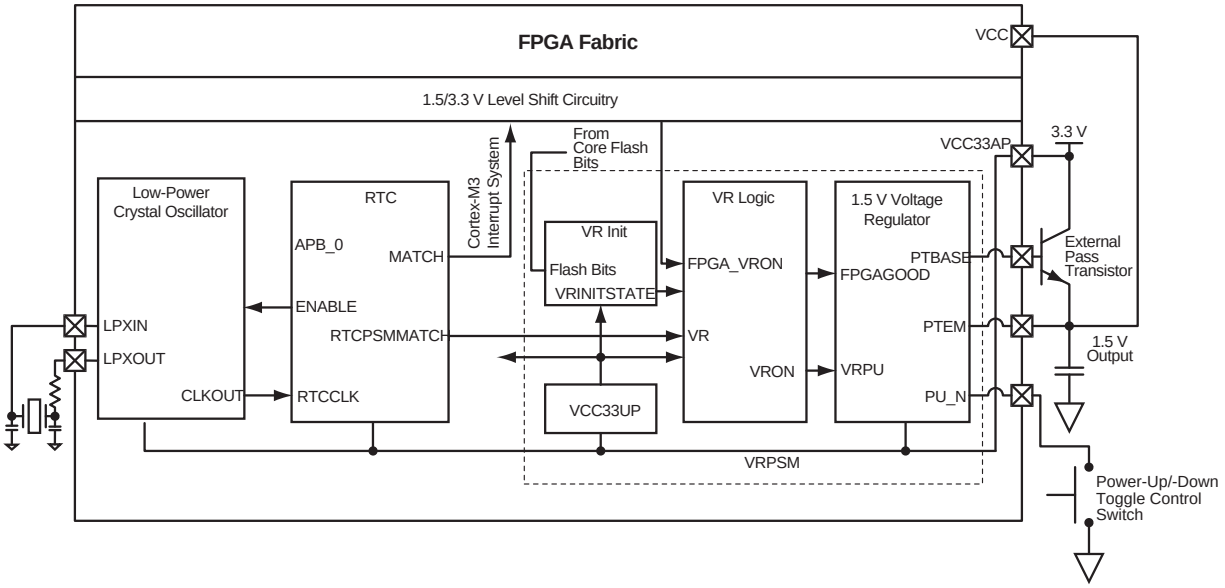


***Figure 16-1* • Real-Time Counter System Block Diagram**

Low-Power Crystal Oscillator Functional Description

The low-power crystal oscillator generates a 32.768 KHz clock for the RTC. It consists of an inverting amplifier, an external ceramic or quartz resonator, and two load capacitors. The generated clock is also connected to the microcontroller subsystem clock conditioning circuit (MSS_CCC) so that this clock can be used by the FPGA fabric. This clock may also be used by the MSS.

This oscillator is enabled/disabled by the XTAL_EN bit (bit 0) of the RTC's control/status register (CTRL_STAT_REG).

Battery Switching Circuit Functional Description

SmartFusion devices have an input for an external battery source that allows both the RTC and the low-power crystal oscillator to function when the 3.3 V VCC supply has been removed. This V_{DDBAT} pin is intended to be connected to 3.0 V lithium cells and should not exceed 3.5 V. This pin may be used with higher voltage cells, such as 3.7 V lithium-ion, provided a suitable method (such as a diode) is used to keep the V_{DDBAT} at or below 3.5 V.

The battery switching circuit continuously compares the battery voltage with the voltage on the VCCLPXTAL pin. This circuit will automatically power the RTC and the low-power crystal oscillator from the battery whenever the battery voltage is approximately 0.4 V or more above the VCCLPXTAL pin voltage. The combined load on the battery (switching circuit, RTC, and low-power crystal oscillator) is expected to be less than 10 µA.

The comparator hysteresis for this battery switching circuit has hysteresis.

RTC Functional Description

The RTC is an APB_0 slave which provides a counter as well as a MATCH output signal that can be used to interrupt the Cortex-M3 processor and to power-up the on-chip 1.5 V voltage regulator. An on-chip 32 KHz oscillator provides the clock source for the RTC (Figure 16-2).

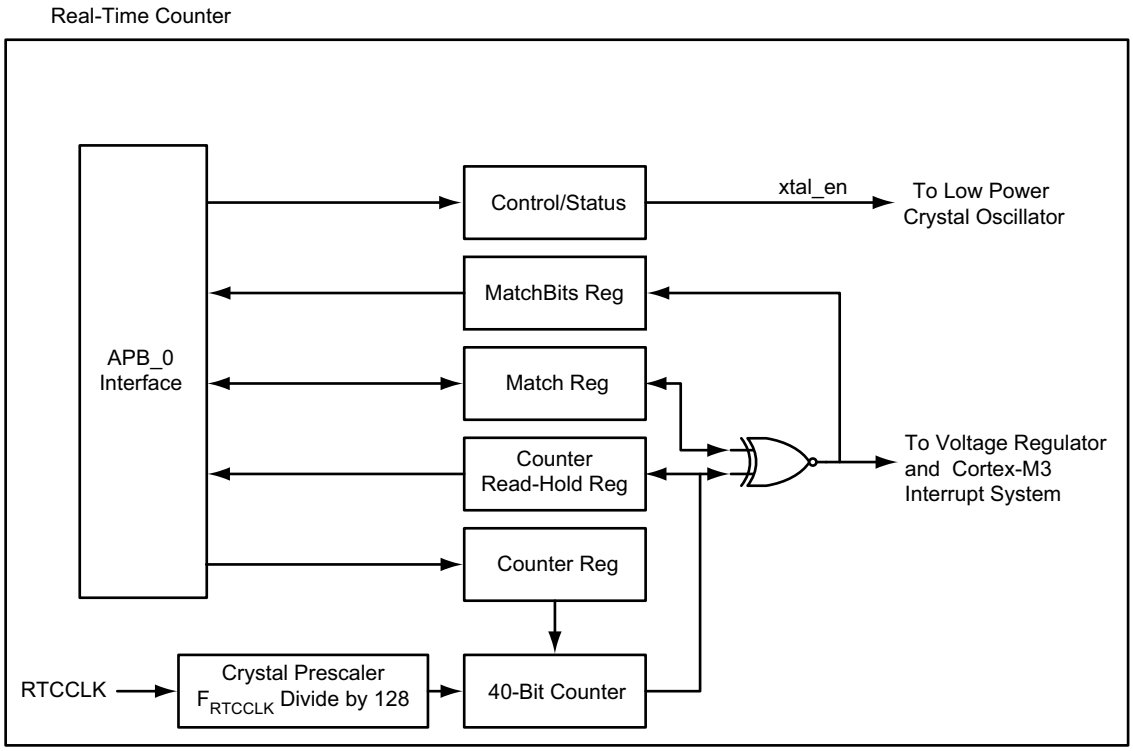


***Figure 16-2* • RTC Block Diagram**

A 40-bit loadable counter is used as the primary timekeeping element within the RTC. This counter can be configured to reset itself when a count value is reached that matches the value set within a 40-bit match register. Note that the only exception to this self-clearing mechanism occurs when the 40-bit counter is equal to zero, since the counter would never increment from zero. When the device is first powered up (when the 3.3 V supply becomes valid), the 40-bit counter and 40-bit match register are cleared to logic 0, and the MATCH output signal is active (logic 1). At any time when the 40-bit counter value does not match the value in the 40-bit match register, the MATCH output signal will become inactive (logic 0).

When the MATCH signal is active, the RTCMATCHEVENT bit (bit 0) in the DSS_STATUS register will be set. This bit is used as an interrupt to the Cortex-M3 processor. This bit will remain set until cleared by writing to the CLRRTCMATCHEVENT bit (bit 0) in the CLR_DSS_STATUS Register.

Both the counter and match registers are addressable (read/write) from the APB_0 bus interface. The counter action can be suspended/resumed by clearing/setting the CNTR_EN bit in the CTRL_STAT_REG register. This allows the RTC to be used for measuring intervals in time.

If a 32.768 KHz external resonator is connected to the low-power crystal oscillator pins, the 40-bit counter will have a maximum count of 4,294,967,296 seconds, which equates to just over 136 years of elapsed timekeeping with a minimum period of 1/256 of a second, which will be the toggle rate of the LSB of the 40-bit counter.

Frequencies other than 32.768 KHz can be used as a clock source with the appropriate scaling of the LSB time interval.

A 7-bit prescaler is used to divide the source clock (from the external crystal) by 128. This prescaled 50% duty-cycle clock is then used by the counter logic as its reference clock. Given an external crystal frequency of 32.768 KHz, the prescaler output clock will toggle at a rate of 32.768 KHz / 128 = 256 Hz.

The 40-bit counter and match registers are each divided into five bytes. Each byte is directly addressable. APB reads and writes must be word aligned. The address map of registers is shown below.

Real-Time Counter Register Interface Summary

Table 16-1 describes the Real-Time Counter Register interface.

***Table 16-1* • Real-Time Counter Register Map**

Register Name	Address	R/W	Reset Value	Description
COUNTER0_REG	0x40014100	R/W	0	Counter bits 7:0
COUNTER1_REG	0x40014104	R/W	0	Counter bits 15:8
COUNTER2_REG	0x40014108	R/W	0	Counter bits 23:16
COUNTER3_REG	0x4001410C	R/W	0	Counter bits 31:24
COUNTER4_REG	0x40014110	R/W	0	Counter bits 39:32
MATCHREG0_REG	0x40014120	R/W	0	Match register bits 7:0
MATCHREG1_REG	0x40014124	R/W	0	Match register bits 15:8
MATCHREG2_REG	0x40014128	R/W	0	Match register bits 23:16
MATCHREG3_REG	0x4001412C	R/W	0	Match register bits 31:24
MATCHREG4_REG	0x40014130	R/W	0	Match register bits 39:32
MATCHBITS0_REG	0x40014140	R/W	0	Individual Match bits 7:0
MATCHBITS1_REG	0x40014144	R/W	0	Individual Match bits 15:8
MATCHBITS2_REG	0x40014148	R/W	0	Individual Match bits 23:16
MATCHBITS3_REG	0x4001414C	R/W	0	Individual Match bits 31:24
MATCHBITS4_REG	0x40014150	R/W	0	Individual Match bits 39:32
CTRL_STAT_REG	0x40014160	R/W	0	Control (write) / Status (read) register bits 7:0

Note: Accessing RTC Registers: When reading the RTC count or match register, which operates in the XTLCLK domain, the appropriate 40-bit value is first copied to a capture register through clock synchronization circuitry, if and only if the least significant byte of that set of register is addressed. Higher-order bytes of the same set of registers captured with the LSB can then be read on immediately later read cycles. Higher order bytes of that set of registers can be read in any order but must be read before switching to a different set of registers to ensure data consistency. For example, when using the RTC counter address ranges from 0x40014100 to 0x40014110, register 0x40014100 must be accessed first before accessing addresses 0x40014104, 0x40014108, 0x4001410C, and 0x40014110 to get the full 40-bit value.

Control/Status Register (CTRL_STAT_REG)

The Control/Status register (CTRL_STAT_REG) is an 8-bit register that defines the operation of the RTC. The Control register can reset the RTC, enabling operation to begin with all zeroes in the counter. The RTC can be configured to clear when it is matched with the Match register, or it can continue to count while still setting the match signal. To enable the SmartFusion device to power up at a specific time or at periodic intervals, the RTC can be configured to turn on the 1.5 V voltage regulator.

Table 16-2 • **CTRL_STAT_REG**

Bit Number	Name	R/W	Reset Value	Description
7	RTC_RST	R/W	0	RTC Reset 1: Writing a logic 1 to this bit causes an RTC reset. 0: Writing a logic 0 to this bit will allow synchronous de-assertion of reset after two clock cycles if VCC33UP = 1.
6	CNTR_EN	R/W	0	Counter Enable 1: Enables the counter if the RTC is not in reset. It takes 64 RTCCLK positive edges (one-half of the prescaler division factor), after reset is removed and CNTR_EN = 1, before the counter is incremented. 0: A logic 0 in this bit resets the prescaler and therefore suspends incrementing the counter, but the counter is not reset. Before writing to the counter registers, the counter must be disabled.
5	VR_EN_MAT	R/W	0	Voltage Regulator Enable on Match 1: Allows the MATCH output port to go to logic 1 when a match occurs between the 40-bit counter and 40-bit match register. 0: Forces the MATCH to logic 0, which prevents the RTC from enabling the voltage regulator.
4:3	Not used			Reserved
2	RST_CNT_OMAT	R/W	0	Reset Counter on Match 1: Allows the counter to clear itself when a match occurs. In this situation, the 40-bit counter clears on the next rising edge of the prescaled clock, approximately 4 ms after the match occurs (the prescaled clock toggles at a rate of 256 Hz, given a 32.768 KHz external crystal). 0: Allows the counter to increment indefinitely while still enabling match events to occur.
1	RSTB_CNT	R/W	0	Counter Reset 1: Allows the counter to count. 0: Resets the 40-bit counter value to zero.
0	XTAL_EN	R/W	0	Crystal Oscillator Enable: This bit enables the low-power crystal oscillator. 1: If a logic 1 is written to this bit, the low-power crystal oscillator is turned on. 0: If a logic 0 is written to this bit, the low-power crystal oscillator is turned off.

Counter 0 Register

All 40 bits of the count are transferred to an internal capture register when the COUNTER0_REG register is read. The second byte of the count (COUNTER1_REG) must be read prior to the next RTC clock.

Therefore, software routines to read the current RTC count should disable interrupts prior to reading the count and re-enable interrupts (if needed), after all five count registers have been read.

***Table 16-3* • COUNTER0_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	CNT_7_0	R/W	0	Counter bits 7:0

Counter 1 Register

***Table 16-4* • COUNTER1_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	CNT_15_8	R/W	0	Counter bits 15:8

Counter 2 Register

***Table 16-5* • COUNTER2_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	CNT_23_16	R/W	0	Counter bits 23:16

Counter 3 Register

***Table 16-6* • COUNTER3_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	CNT_31_24	R/W	0	Counter bits 31:24

Counter 4 Register

***Table 16-7* • COUNTER4_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	CNT_39_32	R/W	0	Counter bits 39:32

Match Register 0 Register

***Table 16-8* • MATCHREG0_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	MATCH_7_0	R/W	0	Match bits 7:0

Match Register 1 Register

***Table 16-9* • MATCHREG1_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	MATCH_15_8	R/W	0	Match bits 15:8

Match Register 2 Register

***Table 16-10* • MATCHREG2_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	MATCH_23_16	R/W	0x000000	Match bits 23:16

Match Register 3 Register

***Table 16-11* • MATCHREG3_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	MATCH_31_24	R/W	0x000000	Match bits 31:24

Match Register 4 Register

***Table 16-12* • MATCHREG4_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	MATCH_39_32	R/W	0	Match bits 39:32

Individual Match Bits 0 Register

***Table 16-13* • MATCHBITS0_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	IND_MATCH_7_0	R/W	0	Individual match bits 7:0

Individual Match Bits 1 Register

***Table 16-14* • MATCHBITS1_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	IND_MATCH_15_8	R/W	0	Individual match bits 15:8

Individual Match Bits 2 Register

***Table 16-15* • MATCHBITS2_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	IND_MATCH_23_16	R/W	0	Individual match bits 23:16

Individual Match Bits 3 Register

***Table 16-16* • MATCHBITS3_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	IND_MATCH_31_24	R/W	0	Individual match bits 31:24

Individual Match Bits 4 Register

***Table 16-17* • MATCHBITS4_REG**

Bit Number	Name	R/W	Reset Value	Description
7:0	IND_MATCH_39_32	R/W	00	Individual match bits 39:32

17 – System Timer

Introduction

The System Timer consists of two programmable 32-bit decrementing counters that generate interrupts to the ARM Cortex-M3 microcontroller and FPGA fabric or can be used in polled mode.

Each counter has two possible modes of operation for generating interrupts: Periodic mode or One-Shot mode. The two timers can be concatenated to create a 64-bit timer with Periodic and One-Shot modes. The two 32-bit timers are identical. The letter "x" in register descriptions is used as a placeholder for 1 or 2, indicating Timer 1 or Timer 2.

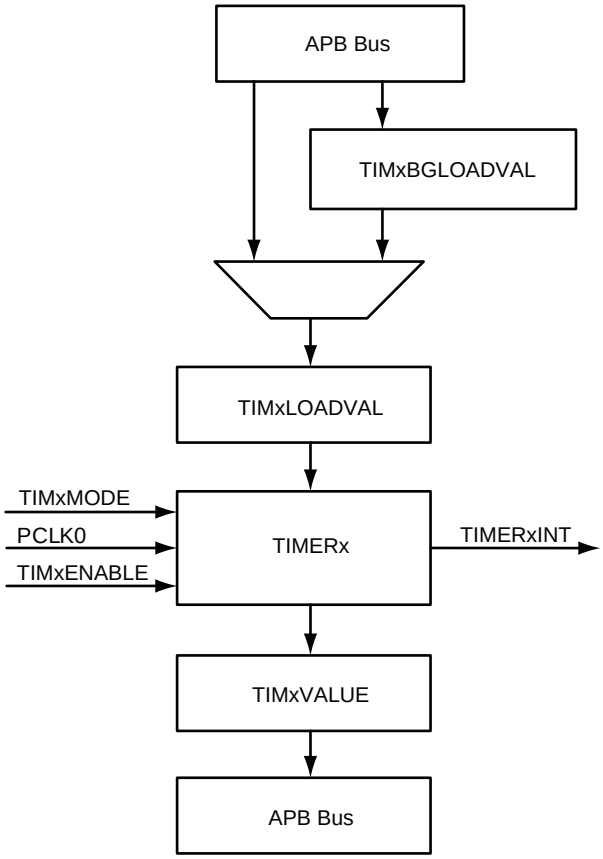


Figure 17-1 **• Block Diagram 32-Bit Mode**

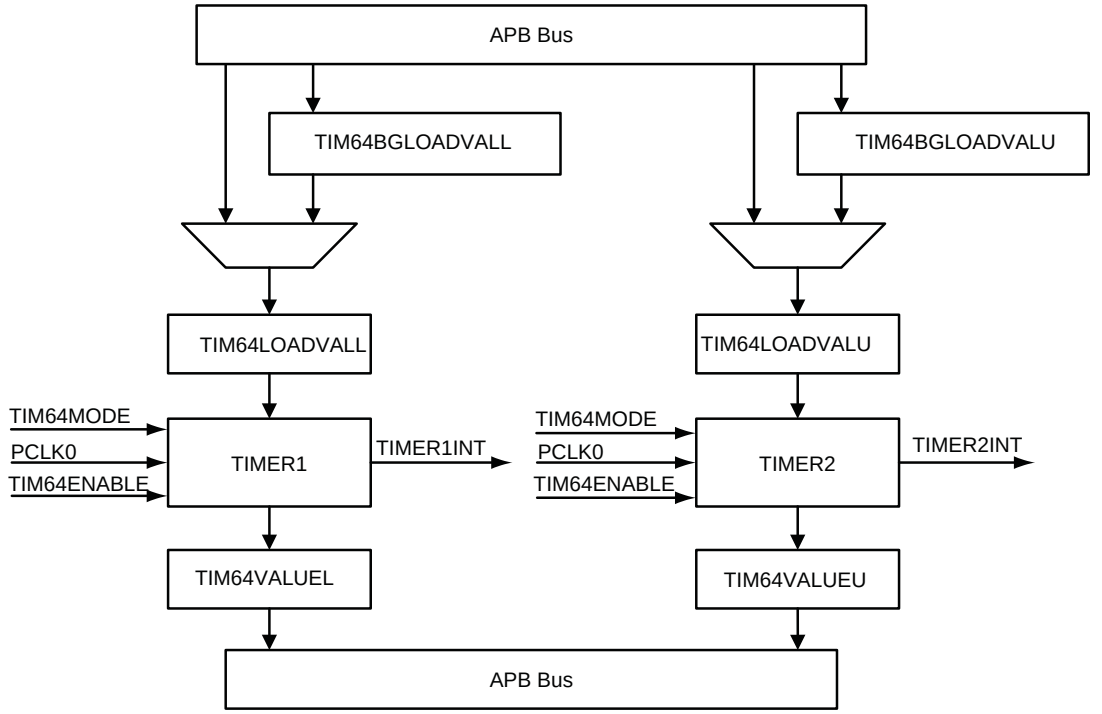


***Figure 17-2* • Block Diagram 64-Bit Mode**

The System Timer is an APB_0 slave module that provides two programmable, interrupt-generating, 32-bit decrementing counters. The two 32-bit timers can be configured to behave as a single 64-bit timer in which Timer 1 contains the lower 32 bits and Timer 2 contains the upper 32 bits of the 64-bit count. The System Timer in dual 32-bit mode or 64-bit mode has two modes of operation.

1. Periodic mode: In this mode the counter generates interrupts at constant intervals. On reaching zero, the counter is reloaded with a value held in a register and begins counting down again.
2. One-Shot mode: The counter generates a single interrupt in this mode. On reaching zero, the counter halts until reprogrammed by the user.

Each 32-bit counter in the System Timer is clocked with the PCLK0 input. With a PCLK frequency of 100 MHz, the maximum timeout period is approximately 42.9 seconds in 32-bit mode and 1.8×10^{11} seconds in 64-bit mode or 5,845.5 years.

The 64-bit and 32-bit modes are mutually exclusive.

Periodic Mode

Periodic mode is selected by setting the TIMxMODE bit in the TIMx_CTRL register to 0.

In Periodic mode, the counter continually counts down to zero when enabled. On reaching zero, an interrupt is generated and the counter is reloaded with the value stored in the TIMxLOADVAL register. The counter then continues to count down towards zero again, without waiting for the interrupt to be cleared. The interrupt remains asserted until cleared by the processor. If the counter reaches zero again without the previous interrupt having been cleared, the counter behaves as if it had just timed out (reached zero). In effect, an interrupt has been lost. This situation can continue indefinitely as long as the counter is enabled in Periodic mode and interrupts are not being cleared.

Writing to the TIMxLOADVAL register at any time causes the counter to be immediately loaded with the value written and to continue counting down from the new value (if enabled). If the TIMxBGLOADVAL register is written to, the value written is used to overwrite the TIMxLOADVAL register without affecting the counter. When the counter next reaches zero, the new value in the TIMxLOADVAL register (which was loaded via the TIMxBGLOADVAL register) will be used to reinitialize the counter.

One-Shot Mode

One-Shot mode is selected by setting the TIMxMODE bit in the TIMx_CTRL register to 1.

In One-Shot mode, the counter will stop on reaching zero and a single interrupt will be generated. When the counter is stopped in One-Shot mode, it can be restarted by writing a non-zero value to the TIMxLOADVAL register. Alternatively, the counter can be restarted by clearing the TIMxMODE bit. This will cause the counter to be loaded with the value held in the TIMxLOADVAL register and to begin operating in Periodic mode.

While the counter is counting down, it is possible to change the value of the TIMxMODE bit at any time without immediately affecting the operation. For example, if the counter is decrementing in One-Shot mode and the TIMxMODE bit is cleared before the counter reaches zero, the counter will begin to operate in Periodic mode on reaching zero.

Writing to the TIMxLOADVAL register at any time causes the counter to be loaded immediately with the value written and to continue counting down from the new value (if enabled).

Writing to the TIMxBGLOADVAL register in One-Shot mode has no real effect unless you intend to switch to Periodic mode when (or before) the next interrupt occurs. When in One-Shot mode, the value written to TIMxBGLOADVAL will be loaded into the TIMxLOADVAL register as normal, but when the counter reaches zero, it will generate a single interrupt and stop, without making use of the value written to TIMxBGLOADVAL. Only a subsequent write to the TIMxLOADVAL register will initiate another One-Shot count-down sequence. However, if the counter is restarted by changing the operating mode to Periodic (by clearing the TIMxMODE bit), then the value previously written to the TIMxBGLOADVAL register is relevant because this will be the start value (taken from the TIMxLOADVAL register) used to initialize the counter in Periodic mode.

64-Bit Mode

Timers 1 and 2 can be concatenated into a single 64-bit timer that operates either in Periodic mode or One-Shot mode. Writing a 1 to the TIM64_MODE register bit location 0 sets the timers in 64-bit mode. Whenever the TIM64MODE bit changes state, the timers are re-initialized to their default reset values. Timer 1 contains the lower 32-bit count of the 64-bit count value. Consequently, when updating or initializing the state of the counter, the upper 32 bits of the 64-bit counter must be written to first, followed by the lower 32 bits. You must ensure that when updating the background load value registers, TIM64_BGLOAD_VAL_U is followed by a write to TIM64_BGLOAD_VAL_L; and when updating the load value registers, TIM64_LOADVAL_U is followed by a write to TIM64_LOADVAL_L. When the lower 32-bit write occurs, the 64-bit counter is updated as one 64-bit value. There are temporary holding registers in the System Timer block that are used to facilitate proper loading of the System Timer in 64-bit mode. These registers are not readable by the user.

System Dependencies

Clocks

The System Timer is clocked by PCLK0 on APB Bus 0. PLCLK0 is a free running version of FCLK (the main clock driving the entire microcontroller subsystem) that is derived from the MSS_CCC output. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109 for more information.

Resets

The System Timer resets to zero on power-up and is held in reset until you enable it. You have the option under software control to reset the System Timer by writing to the System Register, located on the Private Peripheral Bus of the Cortex-M3 microcontroller. Specifically, this System Register is SOFT_RST_CR, located at address 0xE0042030. The TIMER_SOFTRESET control bit is encoded in bit location 6 as follows:

b6: Function

0: System Timer reset released

1: System Timer held in reset (reset value)

Note that setting bit 6 to 0 allows the System Timer to count, but does not cause it to count. You must enable the System Timer by setting the appropriate TIMxENABLE bits in the TIM1_CTRL, TIM2_CTRL, or TIM64_CTRL registers.

Interrupts

There are two interrupt signals from the System Timer Block: the TIMER1INT and TIMER2INT signals. The TIMER1INT signal is mapped to IRQ20 and the TIMER2INT signal is mapped to IRQ21 in the Cortex-M3 NVIC controller. Both interrupt enable bits within the NVIC are located at address 0xE000E100; IRQ20 and IRQ21 correspond to bit locations 20 and 21, respectively. You must also enable interrupts in the System Timer by setting the appropriate TIMxINTEN bits in TIM1_CTRL, TIM2_CTRL, or TIM64_CTRL registers.

When the System Timer is in 64-bit mode and the counter counts down to zero, TIMER1INT and TIMER2INT signals assert. Consequently IRQ20 and IRQ21 interrupts are asserted. Both interrupt signals are identical in that they both represent the 64-bit timer interrupt. In other words, TIMER1INT is not the interrupt signal asserted when Timer 1 counts down to zero while in 64-bit mode. You must disable one of the interrupts in the NVIC controller or you will get two interrupts for the same event.

In 32-bit mode you must clear the TIMxRIS bit, in the respective interrupt service routine, to prevent a reassertion of the interrupt. Likewise, in 64-bit mode you must clear the TIM64RIS bit in the respective interrupt service routine to prevent a reassertion of the interrupt.

GPIO

There are no requirements to configure a GPIO for the System Timer to operate properly.

System Timer Register Map

The System Timer base address resides at 0x40005000 and extends to address 0x40005FFF in the Cortex-M3 memory map.

***Table 17-1* • System Timer Register Map**

Register Name	Address	R/W	Reset Value	Description
TIM1_VAL (TIMx_VAL)	0x40005000	R	0x0	Current value of Timer 1
TIM1_LOADVAL (TIMx_LOADVAL)	0x40005004	R/W	0x0	Load value for Timer 1
TIM1_BGLOADVAL (TIMx_BGLOADVAL)	0x40005008	R/W	0x0	Background load value for Timer 1
TIM1_CTRL (TIMx_CTRL)	0x4000500C	R/W	0x0	Timer 1 Control register
TIM1_RIS (TIMx_RIS)	0x40005010	R/W	0x0	Timer 1 raw interrupt status
TIM1_MIS (TIMx_MIS)	0x40005014	R	0x0	Timer 1 masked interrupt status
TIM2_VAL (TIMx_VAL)	0x40005018	R	0x0	Current value of Timer 2
TIM2_LOADVAL (TIMx_LOADVAL)	0x4000501C	R/W	0x0	Load value for Timer 2
TIM2_BGLOADVAL (TIMx_BGLOADVAL)	0x40005020	R/W	0x0	Background load value for Timer 2
TIM2_CTRL (TIMx_CTRL)	0x40005024	R/W	0x0	Timer 2 Control register
TIM2_RIS (TIMx_RIS)	0x40005028	R/W	0x0	Timer 2 raw interrupt status
TIM2_MIS (TIMx_MIS)	0x4000502C	R	0x0	Timer 2 masked interrupt status
TIM64_VAL_U	0x40005030	R	0x0	Upper 32-bit word in 64-bit mode
TIM64_VAL_L	0x40005034	R	0x0	Lower 32-bit word in 64-bit mode
TIM64_LOADVAL_U	0x40005038	R/W	0x0	Upper 32-bit load value word in 64-bit mode
TITM64_LOADVAL_L	0x4000503C	R/W	0x0	Lower 32-bit load value word in 64-bit mode
TIM64_BGLOADVAL_U	0x40005040	R/W	0x0	Upper 32-bit background load value in 64-bit mode
TIM64_BGLOADVAL_L	0x40005044	R/W	0x0	Lower 32-bit background load value in 64-bit mode
TIM64_CTRL	0x40005048	R/W	0x0	Control register in 64-bit mode
TIM64_RIS	0x4000504C	R/W	0x0	Raw interrupt status in 64-bit mode
TIM64_MIS	0x40005050	R	0x0	Masked interrupt status in 64-bit mode
TIM64_MODE	0x40005054	R/W	0x0	System Timer dual 32-bit or 64-bit mode

Timer x Value Register

***Table 17-2* • TIMx_VAL**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIMx_VAL	R	0x0	This register holds the current value of the counter for Timer x. Reading this register while the System Timer is set to 64-bit mode returns the reset value.

Timer x Load Value Register

***Table 17-3* • TIMx_LOADVAL**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIMx_LOADVAL	R/W	0x0	This register holds the value to load into the counter for Timer x. When this register is written to, the value written is immediately loaded into the counter, regardless of which mode Timer x is in (Periodic or One-Shot). If Timer x is enabled, the counter starts decrementing from this value. When operating in Periodic mode, the value in this register is used to reload the counter when the counter decrements to zero. This register is overwritten with the value in TIMxBGLOADVAL when you write to TIMxBGLOADVAL. In Periodic mode, TIMxLOADVAL always stores the value which is loaded into the counter. Writing or reading this register while the System Timer is set to 64-bit mode has no effect.

Timer x Background Load Value Register

***Table 17-4* • TIMx_BGLOADVAL**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIMx_BGLOADVAL	R/W	0x0	When this register is written to, the value written is loaded into the TIMxLOADVAL register without updating the counter itself. This allows a new value to be loaded into the respective counter without interrupting the current count cycle. The counter is updated with the new value in TIMxLOADVAL when the counter decrements to 0. Writing/reading this register while the System Timer is set to 64-bit mode has no effect.

Timer x Control Register

Table 17-5 • **TIMx_CTRL**

Bit Number	Name	R/W	Reset Value	Description
31:3	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	TIMxINTEN	R/W	0x0	Timer x Interrupt Enable. When the counter reaches zero, an interrupt is signaled to the Cortex-M3 Nested Vectored Interrupt Controller; IRQ20 for Timer x, IRQ21 for Timer 2. 0 = Timer x interrupt disabled 1 = Timer x interrupt enabled Writing this register while the System Timer is set to 64-bit mode has no effect. Reading this register while the System Timer is set to 64-bit mode returns the reset value.
1	TIMxMODE	R/W	0x0	Timer x Mode. 0 = Timer x in Periodic Mode. If TIMxENABLE = 1 when the counter reaches zero the counter is reloaded from the value in the TIMxLOADVAL register and begins counting down immediately. 1 = Timer x in One-Shot mode. If TIMxENABLE = 1 when the counter reaches zero the counter stops counting. To start the counter again, the user must load TIMxLOADVAL with a non-zero value or set the Timer to Periodic mode by clearing TIMxMODE to 0. Writing this register while the System Timer is set to 64-bit mode has no effect. Reading this register while the System Timer is set to 64-bit mode returns the reset value.
0	TIMxENABLE	R/W	0x0	Timer x Enable 0 = Timer x disabled 1 = Timer x enabled Setting to 1 enables the timer and starts it counting from the current value in TIMx_VAL unless TIMx_VAL is 0, in which case TIMx_VAL is loaded from TIMx_LOADVAL. Writing this register while the System Timer is set to 64-bit mode has no effect. Reading this register while the System Timer is set to 64-bit mode returns the reset value.

Timer x Raw Interrupt Status Register

***Table 17-6* • TIMx_RIS**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TIMx_RIS	R/W	0x0	Timer x Raw Interrupt Status 0 = Timer x has not reached zero 1 = Timer x has reached zero at least once since this bit was last cleared (by a reset or by writing 1 to this bit). Writing a 1 to this bit clears the bit and the interrupt, writing a zero has no effect.

Timer x Masked Interrupt Status Register

***Table 17-7* • TIMx_MIS**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TIMx_MIS	R	0x0	Timer x masked interrupt status This read only bit is a logical AND of the TIMxRIS and TIMxINTEN bits. The TIMERxINT output from the timer has the same value as this bit. Writing to this bit has no effect.

Timer 64 Value Upper Register

***Table 17-8* • TIM64_VAL_U**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIM64_VAL_U	R	0x0	This register holds the current value of the upper 32 bits of the 64-bit count value for the System Timer. This register is read only; writes have no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Value Lower Register

***Table 17-9* • TIM64_VAL_L**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIM64_VAL_L	R	0x0	This register holds the current value of the lower 32 bits of the 64-bit count value for the System Timer. This register is read only; writes have no effect. When reading from this register, the upper 32 bits of the 64-bit counter is stored into TIM64_VAL_U. To properly read the 64-bit counter value, the user must read from this register first; then the TIM64_VAL_U. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Load Value Upper Register

***Table 17-10* • TIM64_LOADVAL_U**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIM64_LOADVAL_U	R/W	0x0	This register holds the upper 32-bit value to load into the System Timer when in 64-bit mode. When this register is written to, the value written is immediately loaded into a temporary register. The value in the temporary register is only written to the System Timer when the lower 32-bit word TIM64_LOADVAL_L is written. Writing this register while the System Timer is set to 32-bit mode has no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Load Value Lower Register

***Table 17-11* • TITM64_LOADVAL_L**

Number	Name	R/W	Reset Value	Description
31:0	TIM64_LOADVAL_L	R/W	0x0	When this register is written to, the value written is immediately loaded into the lower 32 bits of the 64-bit counter along with the value previously written in register TIM64_LOADVAL_U. This applies in both Periodic and One-Shot mode. The value stored in this register is also used to reload the counter when the count reaches zero and the counter is operating in Periodic mode. This register will be overwritten if the TIM64BGLOADVAL register is written to, but the counter will not be updated with the new value. The TIM64BGLOADVAL register is an internal register to the System Timer, used to concatenate the two 32-bit values from TIM64_BGLOAD_VAL_U and TIM64_BGLOAD_VAL_L. If Periodic mode is selected, the values in the TIM64_LOADVAL_L and the TIM64_LOADVAL_U are loaded into the counter when the counter decrements to zero. Writing this register while the System Timer is set to 32-bit mode has no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Background Load Value Upper Register

***Table 17-12* • TIM64_BGLOADVAL_U**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIM64_BGLOADVAL_U	R/W	0x0	This register holds the upper 32-bit background value to load into the System Timer when in 64-bit mode. The value in this register is written to the internal TIM64LOADVAL register when the lower 32-bit word TIM64_BGLOADVAL_L is written. Writing this register while the System Timer is set to 32-bit mode has no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Background Load Value Lower Register

***Table 17-13* • TIM64_BGLOADVAL_L**

Bit Number	Name	R/W	Reset Value	Description
31:0	TIM64_BGLOADVAL_L	R/W	0x0	Background load value for the lower 32 bits of 64-bit System Timer. When this register is written to, both the upper and lower words are written into an internal 64-bit TIM64LOADVAL register without updating the counter itself. The TIM64LOADVAL register is an internal register to the System Timer used in 64-bit mode for concatenating the two 32-bit registers, TIM64_LOADVAL_U and TIM64_LOADVAL_L. The new 64-bit load value is loaded into the counter when the counter reaches zero. Writing this register while the System Timer is set to 32-bit mode has no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Control Register

***Table 17-14* • TIM64_CTRL**

Bit Number	Name	R/W	Reset Value	Description
31:3	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	TIM64INTEN	R/W	0x0	Timer 64 Interrupt Enable. When the counter reaches zero, an interrupt is signaled to the Cortex-M3 Nested Vectored Interrupt Controller. 0 = Timer 64 interrupt disabled 1 = Timer 64 interrupt enabled
1	TIM64MODE	R/W	0x0	Timer 64 Mode 0 = Timer 64 in Periodic mode If TIM64ENABLE = 1 when the counter reaches zero, the counter is reloaded from the value in the TIM64_LOADVAL_U and TIM64_LOADVAL_L registers and begins counting down immediately. 1 = Timer 64 in One-Shot mode If TIM64ENABLE = 1 when the counter reaches zero, the counter stops counting. To start the counter again, load TIM64_LOADVAL_U and TIM64_LOADVAL_L with a non-zero value or set the Timer to Periodic mode by clearing TIM64MODE to 0.
0	TIM64ENABLE	R/W	0x0	Timer 64 Enable 0 = Timer 64 disabled 1 = Timer 64 enabled Writing this register while the System Timer is set to 32-bit mode has no effect. Reading this register while the System Timer is set to 32-bit mode returns the reset value.

Timer 64 Raw Interrupt Status Register

***Table 17-15* • TIM64_RIS**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TIM64_RIS	R/W	0x0	Timer x raw interrupt status 0 = Timer 64 has not reached zero 1 = Timer 64 has reached zero at least once since this bit was last cleared (by a reset or by writing 1 to this bit) Writing a 1 to this bit clears the bit and the interrupt; writing a zero has no effect.

Timer 64 Masked Interrupt Status Register

***Table 17-16* • TIM64_MIS**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TIM64_MIS	R	0x0	Timer 64 masked interrupt status This read only bit is a logical AND of the TIM64RIS and TIM64INTEN bits. The TIMER1INT and TIMER2INT outputs from the timer both have the same value as this bit when the timer is set to 64-bit mode. Writing to this bit has no effect.

Timer 64 Mode Register

***Table 17-17* • TIM64_MODE**

Bit Number	Name	R/W	Reset Value	Description
31:1	Reserved	R/W	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TIM64_MODE	R/W	0x0	Timer 64 mode 0 = Timer 64 disabled; two separate 32-bit timers 1 = Timer 64 enabled; one 64-bit timer Changing the state of this bit has the effect of reinitializing the System Timer register map to its default power-up state.

SmartFusion MSS Timer Application Development

The SmartFusion MSS System Timer peripherals provide two 32-bit decrementing counters which can be used to implement two 32-bit timers or one 64-bit timer. This section provides a high level overview of the design flow in order to facilitate fast and straightforward MSS Timer application development. The timers generate an interrupt that can be used by the MSS or the FPGA fabric to trigger a user-defined action. Using the MSS Timer peripherals primarily requires configuring the MSS peripherals in use and developing user firmware. Using the MSS Timer does not require the FPGA fabric or the Libero design flow if there are no FPGA fabric resources required by the user application. The SmartFusion MSS must be configured to setup the MSS clocking scheme and configure any additional MSS peripherals desired to process a Timer interrupt. The user firmware will then control these active peripherals so that they can perform the tasks required by the user application.

User application code can be developed and debugged using any of the three supported embedded software development tools: SoftConsole, Keil Microcontroller Development Kit (MDK) µVision,® and IAR Embedded Workbench®. Microsemi provides a set of MSS Timer drivers that can be generated from the MSS configurator or from the Firmware Catalog. These drivers are common for all three tool flows. However, the Cortex Microcontroller Software Interface Standard (CMSIS) access layer is dependent on the tool flow selected and should be chosen based on the specific tool flow being utilized.

The MSS Timer drivers enable application code development without having to manually read and write the MSS System Registers to initialize, configure, and operate the timers. The MSS configurator and Firmware Catalog also provide sample projects depicting the MSS Timer usage. The MSS Timer drivers contain functions that allow the user to set up Timer1 and Timer2 independently or as a single 64-bit timer. For additional driver-specific details, refer to the SmartFusion MSS Configurators and Drivers User's Guides.

Developing User Application Code with MSS Timer Drivers

When creating Timer application code using the SmartFusion MSS Timer drivers, the mss_timer.h header file must be included in the C code, as shown below.

Example

```
#include "mss_timer.h"
```

Before an MSS Timer can be used, it must first be initialized to operate in a specific mode. As described in the "Introduction" section on page 305, each timer can operate in periodic mode or one-shot mode. In the firmware these are MSS_TIMER_PERIODIC_MODE and MSS_TIMER_ONE_SHOT_MODE.

The functions MSS_TIM1_init(), MSS_TIM2_init(), and MSS_TIM64_init() are used to initialize 32-bit Timer1, 32-bit Timer2, or the 64-bit combination of Timer1 and Timer2. It is important to note that the SmartFusion MSS Timer cannot be used as both a 64-bit and 32-bit timer. Calling the MSS_TIM64_init() function will overwrite any previous configuration of the MSS Timer1 or Timer2 as a 32-bit timer. Similarly, calling MSS_TIM1_init() or MSS_TIM2_init()will overwrite any previous configuration of the MSS Timer as a 64-bit timer. For example, to initialize Timer1 as a 32-bit periodic timer, the code below would be used:

```
MSS_TIM1_init( MSS_TIMER_PERIODIC_MODE );
```

Once the desired timer is initialized, the 32-bit or 64-bit count value must be loaded into the timer down counter. There are two ways to load the timer value: the immediate load and the background load.

In the immediate load method, the load_value is the value from which the timer will start counting down to zero immediately after being enabled. If the timer was already enabled when an immediate load occurred, the counter value will be set to the new load value and immediately start counting down. The functions used to perform an immediate timer load are called MSS_TIM1_load_immediate(), MSS_TIM2_load_immediate(), and MSS_TIM64_load_immediate() for each respective timer (Timer1, Timer2 or 64-bit Timer). From this point on, functions will be shown for Timer1, since the Timer2 and 64-bit functions are similar in usage.

For example, to load a timer start value equal to 1 second, the user can use the CMSIS-PAL global variable, g_FrequencyPCLK0, which provides the timer input frequency. Thus, counting down from this value to zero will take 1 second after the timer is enabled.

```
…
SystemCoreClockUpdate(); //CMSIS Func - updates global freq variables
MSS_TIM1_load_immediate( g_FrequencyPCLK0 );
…
```

In contrast, the background load method loads the timer with the value that will be reloaded into the timer down-counter the next time the counter reaches zero. When the timer is operating in periodic mode, background loading is typically used to change the delay period between the timer interrupts without stopping the timer.

Note that the MSS_TIM64_load_immediate function takes two 32-bit load values for the upper and lower 32-bit values which comprise the 64-bit down-counter start value.

After the timer value has been loaded, the timer can be started, as shown in the example below:

```
…
MSS_TIM1_start();
…
```

In order to generate timer interrupts to the Cortex-M3 processor, the timer interrupts must be enabled, as shown in the example below:

```
…
MSS_TIM1_enable_irq();
…
```

Processing a Timer Interrupt

Once the timer is initialized, loaded with the down-counter start value, started and interrupts are enabled, the Timer will generate a Timer1, Timer2, or Timer64 interrupt. In order to process the timer interrupt, the corresponding Timer Interrupt Service Routine (ISR) or Interrupt Handler function must be defined.

The SmartFusion MSS Timer Interrupt Handler function prototypes have been defined in the SmartFusion CMSIS-PAL; thus, the user timer interrupt handlers must follow the pre-defined function prototypes listed below:

```
void Timer1_IRQHandler( void );
void Timer2_IRQHandler( void );
```

To add a Timer1 interrupt handler, the Timer1_IRQHandler() function must be defined by the user application code. Similarly, for the Timer2 interrupt handler, the Timer2_IRQHandler() function must be defined by the user. Lastly, when using the MSS Timer as a 64-bit timer, only the Timer1_IRQHandler() function must be defined by the user application code because the Timer2 interrupt is not used when the MSS Timer is configured as a 64-bit timer.

The example below shows a Timer1 interrupt handler function that sends a message from the MSS UART when the Timer1 interrupt occurs. Notice that the last operation in the interrupt handler is the call to MSS_TIM1_clear_irq() to clear the Timer1 interrupt.

```
void Timer1_IRQHandler( void )
{
  /* Print a message */
  MSS_UART_polled_tx( &g_mss_uart0, tx_buff, sizeof(tx_buff) );

  /* Clear TIM1 interrupt */
  MSS_TIM1_clear_irq();
}
```

The SmartFusion MSS Timer application development overview provided in this section should enable the user to rapidly and easily utilize the SmartFusion MSS Timer peripherals within the embedded application. For additional details on the three tool flows and application development examples, refer to the SmartFusion tutorials available at http://www.microsemi.com/soc/products/smartfusion/docs.aspx.

18 – General Purpose I/O Block (GPIO)

The microcontroller subsystem (MSS) general purpose I/O (GPIO) block is an APB slave to 32 general purpose I/Os. Each GPIO bit is configurable as an input or an output with configurable interrupt generation. GPIOs can be routed to dedicated I/O buffers (MSSIOBUF) or in some cases to the FPGA fabric interface through an IOMUX. The number of GPIOs available to package pins varies with device size and package type. **The MSS peripherals are not multiplexed with each other; they are multiplexed only with the GPIO block.**

A block diagram showing the arrangement of the GPIO, the IOMUX, and the I/O buffer is shown in Figure 18-1. This particular example shows the UART_0 transmit signal multiplexed with GPIO bit 20.

GPIOs 31 through 16 are shared with peripheral signals via an IOMUX and through an alternate sourcing input, these GPIOs can be shared with the fabric Interface, while GPIOs 15 through 0 are shared with I/O interface tiles from the FPGA fabric. The GPIO block is mapped to address 0x40013000 in the memory map.

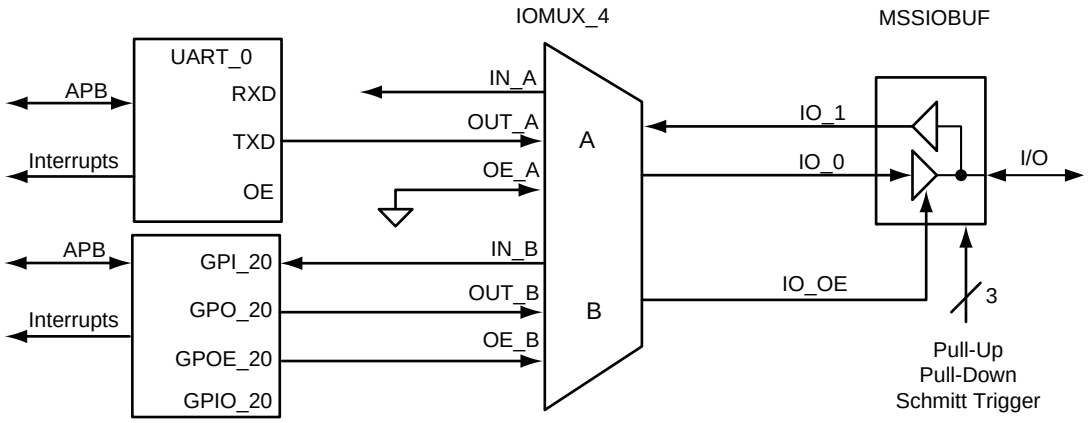


***Figure 18-1* • GPIO, IOMUX, and I/O Buffer Arrangement**

Features of MSS General Purpose I/Os

The features of MSS I/Os are described in Table 18-1 and Table 18-2 on page 320.

***Table 18-1* • MSS I/O Voltage Standards and Drive Strengths**

I/O Bank	Direction	Single-Ended I/O Standards	Drive Strength
Bank 4	West	LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V	8 mA
		LVCMOS 1.8 V	4 mA
		LVCMOS 1.5 V	2 mA
Bank 2	East	LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V	8 mA
		LVCMOS 1.8 V	4 mA
		LVCMOS 1.5 V	2 mA

***Table 18-2* • MSS I/O Features**

I/O Bank	Direction	Hot Insertion	Cold Sparing	Output Enable Skew	Pull-Up/ Pull-Down*	Schmitt Trigger*
Bank 4	West	Yes	Yes	No	Yes	Yes
Bank 2	East	Yes	Yes	No	Yes	Yes

*Note: * Selection of these occurs in the I/O editor in the MSS Configurator within SmartDesign.*

MSS GPIO Functional Description

Figure 18-2 depicts the GPIO block diagram. The GPIO block consists of one 32-bit input register, GPI; one 32-bit output register, GPO; one 32-bit interrupt register, GPIO_INTR; and 32 configuration registers (one register for each GPIO bit), GPIOCFG_x_REG (where x can range from 0 to 31). The GPIOCFG_x_REG register, per bit, sets the input/output direction, enables the relevant bit of each 32-bit register (GPINEN for GPI, GPOUTEN for GPO), and sets the interrupt mask of each individual bit in the GPIO_INTR register. Interrupts can be level-sensitive or edge-triggered.

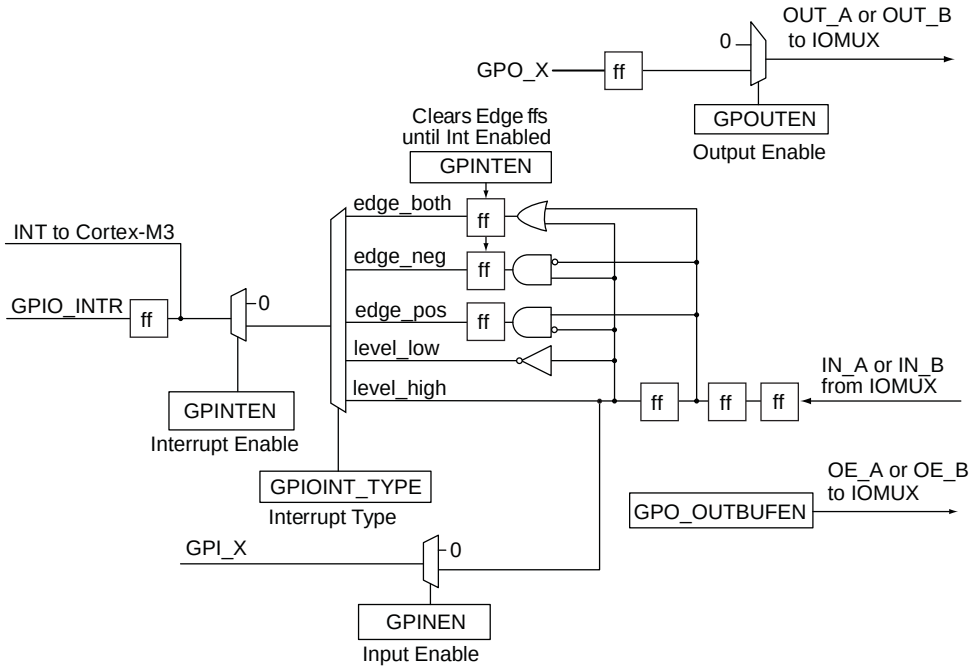


***Figure 18-2* • GPIO Block Diagram**

GPIO Register Map

The GPIO block is mapped to address 0x40013000 in the memory map. Registers referred to in this document are defined in Table 18-3.

***Table 18-3* • MSS GPIO Register Map**

Register Name	Address	R/W	Reset Value	Description
GPIO_x_CFG (x = 0)	0x40013000	R/W	0x0	GPIO Configuration register for bit 0
GPIO_x_CFG (x = 1)	0x40013004	R/W	0x0	GPIO Configuration register for bit 1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
GPIO_x_CFG (x = 31)	0x4001307C	R/W	0x0	GPIO Configuration register for bit 31
GPIO_IRQ	0x40013080	R/W	0x0	Interrupt Status register
GPIO_IN	0x40013084	R	0x0	Read only bits for ports configured as inputs
GPIO_OUT	0x40013088	R/W	0x0	Read/write bits for ports configured as outputs
MSS_IO_BANK_CR	0xE0042078	R/W	0x0	Configures logic thresholds for the MSS I/O banks
GPIN_SOURCE_CR	0xE004207C	R/W	0x0	Provides for alternate sourcing of input to GPIOs 16–31
IOMUX_n_CR	0xE0042100	R/W	0x0	Configures the various IOMUXes

MSS GPIO Register

Table 18-4 gives bit definitions for the GPIOCFG_x_REG registers.

***Table 18-4* • GPIO_x_CFG**

Bit Number	Name	R/W	Reset Value	Description
7:5	GPIOINT_TYPE	R/W	0b000	See Table 18-5 on page 322.
4	Reserved	R/W	0b0	Reserved
3	GPINTEN	R/W	0b0	0 = Interrupt disabled. 1= Interrupt enabled.
2	GPO_OUTBUFEN	R/W	0b0	0 = Disable output buffer if signal routed through IOMUX to IOBUF (dependent on IOMUX setting). 1 = Enable output buffer if signal routed through IOMUX to IOBUF (dependent on IOMUX setting).
1	GPINEN	R/W	0b0	0 = Input register disabled. 1 = Input register enabled.
0	GPOUTEN	R/W	0b0	0 = Output register disabled. 1 = Output register enabled.

To enable the GPIO as an input, set the GPINEN bit to a 1. This allows the synchronized input from the MSSIOBUF to be available in the GPI register, as shown in Figure 18-2 on page 320. When the user disables the input path by setting the GPINEN bit to a 0, the value read at the respective bit location of the GPI input register will be 0.

To enable the GPIO as an output, set the GPOUTEN bit to a 1. This allows the output from the GPO register to pass to the MSSIOBUF, as shown in Figure 18-2 on page 320. If the user wishes to disable the output path by setting the GPOUTEN to a 0, a logical 0 will drive the MSSIOBUF and the respective bit in the GPO register will retain its state.

To use a GPIO as a general purpose interrupt, the GPINTEN bit must be set to a 1 for that bit. Level interrupts clear automatically, but edge interrupts must be cleared manually by writing a 1 to the respective bit location in the GPIO_INTR register. If attempting to clear an edge interrupt at the same time as an edge occurs, the edge wins. Table 18-5 depicts the bit settings needed to generate a particular type of interrupt. GPIO interrupts 0 to 31 are mapped to ARM Cortex-M3 NVIC interrupts 32 to 63. The 32 GPIO interrupts are ORed into a single interrupt signal, which is available to the FPGA fabric interface as MSSINT[1].

***Table 18-5* • Input Interrupt Type Configuration**

GPIOINT_TYPE			
Bit 7	**Bit 6**	**Bit 5**	**Definition**
0	0	0	Level High
0	0	1	Level Low
0	1	0	Positive edge
0	1	1	Negative edge
1	0	0	Both edges

The output buffer can be tristated by setting the MSSIOBUFOE bit to a 0. Setting MSSIOBUFOE to a 1 enables the output driver in the MSSIOBUF. Pull-ups, pull-downs, and Schmitt Trigger controls are associated with the respective IOMUX and are described in the "Fabric Interface and IOMUX" section on page 343.

MSS GPIO Logic Thresholds

For the east bank and west bank of MSS GPIO pads, the logic threshold of these MSS I/Os are programmable on a per bank basis. The MSS_IO_BANK_CR is located within the SYSREG block at address 0xE0042078. This register controls the east and west bank I/O thresholds according to Table 18-7.

***Table 18-6* • MSS_IO_BANK_CR**

Bit Number	Name	R/W	Reset Value	Function
3:2	BTWEST	R/W	0	Logic threshold selection for MSS I/O west bank. See Table 18-7.
1:0	BTEAST	R/W	0	Logic threshold selection for MSS I/O east bank. See Table 18-7.

***Table 18-7* • MSS GPIO Logic Threshold**

BTWEST		BTEAST		
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Logic Threshold for Each Bank, Set Individually**
0	0	0	0	I/O threshold = LVTTL/LVCMOS 3.3 V – default state
0	1	0	1	I/O threshold = LVCMOS 2.5 V
1	0	1	0	I/O threshold = LVCMOS 1.8 V
1	1	1	1	I/O threshold = LVCMOS 1.5 V output threshold configured as 1.8 V LVCMOS

GPIN Source Select Register

For the A2F200, GPIOs 16 through 31 are multiplexed between two IOMUXes. This allows these GPIOs access to user signals from the fabric interface. The GPIN_SOURCE_CR steers the user's signal from the fabric interface through the secondary IOMUX, as shown in Figure 18-3 on page 324.

For example, for GPIN_16_SRC, the input can be sourced from IOMUXCELL 0 or 41. Figure 18-3 on page 324 shows how two IOMUXes are configured in relation to bit GPIN_16_SRC. This topology allows a user's logic signal from the fabric, F2M[25] for instance, to be read by the Cortex-M3 processor via the GPIO16 input. Table 18-8 lists the bit definitions for the GPIN_SOURCE_CR register. F2M in this case means fabric-to-MSS signal direction and M2F describes a signal sourced from the MSS driving into the FPGA fabric.

***Table 18-8* • GPIN_SOURCE_CR**

Bit Number	Name	R/W	Reset Value	Function
0	GPIN_16_SRC	R/W	0	0 = IOMUX 0; 1 = IOMUX 41
1	GPIN_17_SRC	R/W	0	0 = IOMUX 1; 1 = IOMUX 42
2	GPIN_18_SRC	R/W	0	0 = IOMUX 2; 1 = IOMUX 43
3	GPIN_19_SRC	R/W	0	0 = IOMUX 3; 1 = IOMUX 44
4	GPIN_20_SRC	R/W	0	0 = IOMUX 4; 1 = IOMUX 45
5	GPIN_21_SRC	R/W	0	0 = IOMUX 5; 1 = IOMUX 46
6	GPIN_22_SRC	R/W	0	0 = IOMUX 6; 1 = IOMUX 47
7	GPIN_23_SRC	R/W	0	0 = IOMUX 7; 1 = IOMUX 48
8	GPIN_24_SRC	R/W	0	0 = IOMUX 8; 1 = IOMUX 49
9	GPIN_25_SRC	R/W	0	0 = IOMUX 9; 1 = IOMUX 50
10	GPIN_26_SRC	R/W	0	0 = IOMUX 10; 1 = IOMUX 51
11	GPIN_27_SRC	R/W	0	0 = IOMUX 11; 1 = IOMUX 52
12	GPIN_28_SRC	R/W	0	0 = IOMUX 12; 1 = IOMUX 53
13	GPIN_29_SRC	R/W	0	0 = IOMUX 13; 1 = IOMUX 54
14	GPIN_30_SRC	R/W	0	0 = IOMUX 14; 1 = IOMUX 55
15	GPIN_31_SRC	R/W	0	0 = IOMUX 15; 1 = IOMUX 56

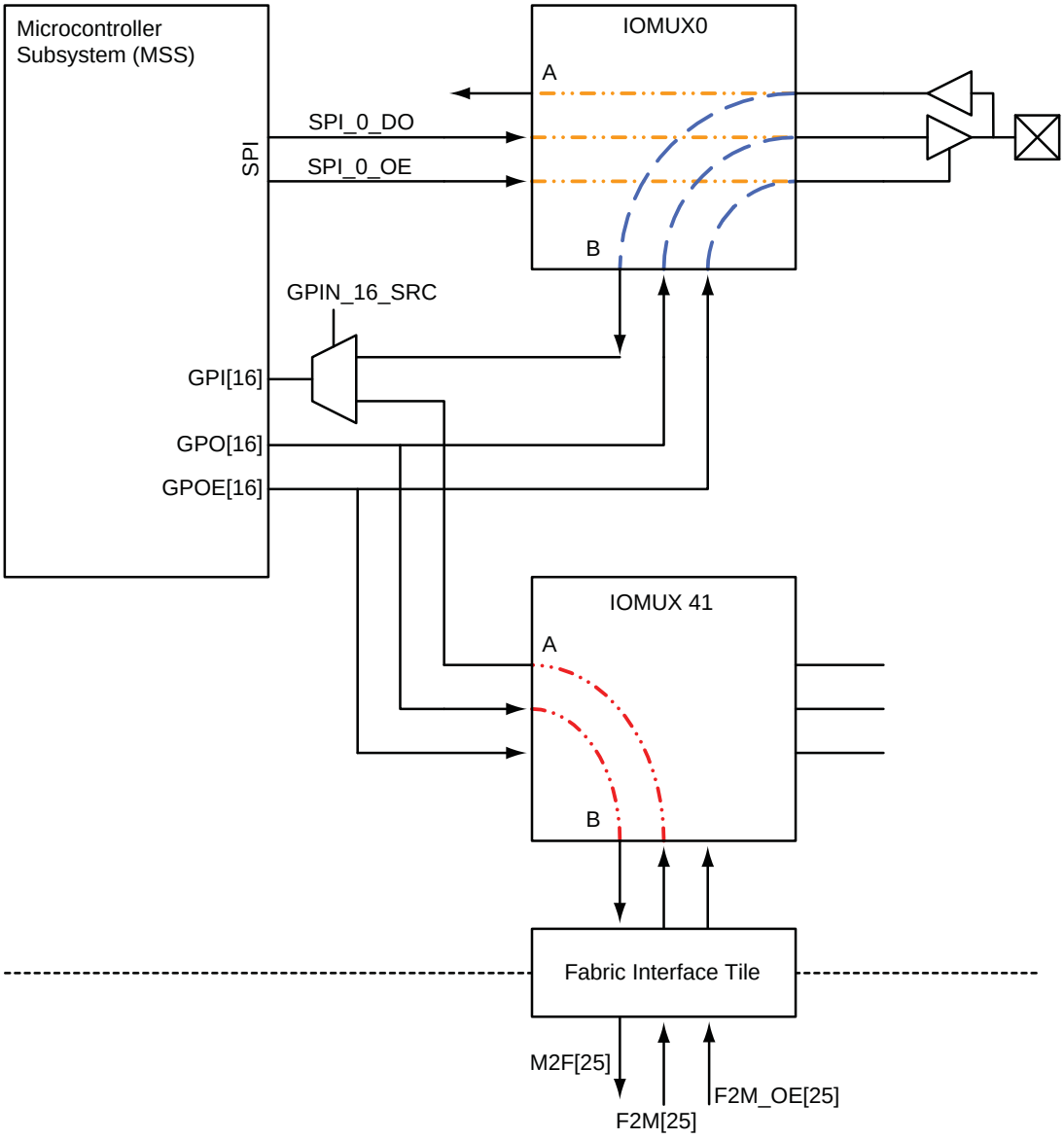


Figure 18-3 **• Example of GPIN Source Selection from Two IOMUXes**

IOMUXes Associated with GPIOs

IOMUXes 0 – 15 and 25 – 56 are used to multiplex MSS peripherals, GPIOs, and fabric interface signals to MSSIOBUFs. The individual signal mapping for each IOMUX is listed in Table 18-9 through Table 18-56 on page 340.

IOMUX 0

Table 18-9 • **IOMUX 0**

Pad Name	Pad Ports	IOMUX_0_CR	IOMUX 0 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_DO/GPIO_16	I					GPI_16		
	O			SPI_0_DO			GPO_16	
	OE				SPI_0_DEN			GPOE_16
	PU	IOMUX_0_PU						
	PD	IOMUX_0_PD						
	ST	IOMUX_0_ST						

IOMUX 1

Table 18-10 • **IOMUX 1**

Pad Name	Pad Ports	IOMUX_1_CR	IOMUX 1 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_DI/GPIO_17	I		SPI_0_DI			GPI_17		
	O			GND			GPO_17	
	OE				GND			GPOE_17
	PU	IOMUX_1_PU						
	PD	IOMUX_1_PD						
	ST	IOMUX_1_ST						

IOMUX 2

Table 18-11 • **IOMUX 2**

Pad Name	Pad Ports	IOMUX_2_CR	IOMUX 2 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_CLK/GPIO_18	I		SPI_0_CLKI			GPI_18		
	O			SPI_0_CLKO			GPO_18	
	OE				SPI_0_MODE			GPOE_18
	PU	IOMUX_2_PU						
	PD	IOMUX_2_PD						
	ST	IOMUX_2_ST						

IOMUX 3

***Table 18-12* • IOMUX 3**

Pad Name	Pad Ports	IOMUX_3_CR	IOMUX 3 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_0_SS[0]/GPIO_19	I		SPI_0_SSI			GPI_19		
	O			SPI_0_SSO			GPO_19	
	OE				SPI_0_MODE			GPOE_19
	PU	IOMUX_3_PU						
	PD	IOMUX_3_PD						
	ST	IOMUX_3_ST						

IOMUX 4

***Table 18-13* • IOMUX 4**

Pad Name	Pad Ports	IOMUX_4_CR	IOMUX 4 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_TXD/GPIO_20	I					GPI_20		
	O			UART_0_TXD			GPO_20	
	OE				High			GPOE_20
	PU	IOMUX_4_PU						
	PD	IOMUX_4_PD						
	ST	IOMUX_4_ST						

IOMUX 5

***Table 18-14* • IOMUX 5**

Pad Name	Pad Ports	IOMUX_5_CR	IOMUX 5 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_0_RXD/GPIO_21	I		UART_0_RXD			GPI_21		
	O			GND			GPO_21	
	OE				GND			GPOE_21
	PU	IOMUX[5]_PU						
	PD	IOMUX[5]_PD						
	ST	IOMUX[5]_ST						

IOMUX 6

***Table 18-15* • IOMUX 6**

Pad Name	Pad Ports	IOMUX_6_CR	IOMUX 6 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_0_SDA/GPIO_22	I		I2C_0_SDAI			GPI_22		
	O			GND			GPO_22	
	OE				~(I2C_0_SDAO)			GPOE_22
	PU	IOMUX_6_PU						
	PD	IOMUX_6_PD						
	ST	IOMUX_6_ST						

IOMUX 7

***Table 18-16* • IOMUX 7**

Pad Name	Pad Ports	IOMUX_7_CR	IOMUX 7 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_0_SCL/GPIO_23	I		I2C_0_SCLI			GPI_23		
	O			GND			GPO_23	
	OE				~(I2C_0_SCLO)			GPOE_23
	PU	IOMUX_7_PU						
	PD	IOMUX_7_PD						
	ST	IOMUX_7_ST						

IOMUX 8

***Table 18-17* • IOMUX 8**

Pad Name	Pad Ports	IOMUX_8_CR	IOMUX 8 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_DO/GPIO_24	I					GPI_24		
	O			SPI_1_DO			GPO_24	
	OE				SPI_1_DEN			GPOE_24
	PU	IOMUX_8_PU						
	PD	IOMUX_8_PD						
	ST	IOMUX_8_ST						

IOMUX 9

***Table 18-18* • IOMUX 9**

Pad Name	Pad Ports	IOMUX_9_CR	IOMUX 9 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_DI/GPIO_25	I		SPI_1_DI			GPI_25		
	O			GND			GPO_25	
	OE				GND			GPOE_25
	PU	IOMUX_9_PU						
	PD	IOMUX_9_PD						
	ST	IOMUX_9_ST						

IOMUX 10

***Table 18-19* • IOMUX 10**

Pad Name	Pad Ports	IOMUX_10_CR	IOMUX 10 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_CLK/GPIO_26	I		SPI_1_CLKI			GPI_26		
	O			SPI_1_CLKO			GPO_26	
	OE				SPI_1_MODE			GPOE_26
	PU	IOMUX_10_PU						
	PD	IOMUX_10_PD						
	ST	IOMUX_10_ST						

IOMUX 11

***Table 18-20* • IOMUX 11**

Pad Name	Pad Ports	IOMUX_11_CR	IOMUX 11 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
SPI_1_SS[0]/GPIO_27	I		SPI_1_SSI			GPI_27		
	O			SPI_1_SSO			GPO_27	
	OE				SPI_1_MODE			GPOE_27
	PU	IOMUX_11_PU						
	PD	IOMUX_11_PD						
	ST	IOMUX_11_ST						

IOMUX 12

***Table 18-21* • IOMUX 12**

Pad Name	Pad Ports	IOMUX_12_CR	IOMUX 12 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_TXD/GPIO_28	I					GPI_28		
	O			UART_1_TXD			GPO_28	
	OE				GND			GPOE_28
	PU	IOMUX_12_PU						
	PD	IOMUX_12_PD						
	ST	IOMUX_12_ST						

IOMUX 13

***Table 18-22* • IOMUX 13**

Pad Name	Pad Ports	IOMUX_13_CR	IOMUX 13 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
UART_1_RXD/GPIO_29	I		UART_1_RXD			GPI_29		
	O			GND			GPO_29	
	OE				GND			GPOE_29
	PU	IOMUX_13_PU						
	PD	IOMUX_13_PD						
	ST	IOMUX_13_ST						

IOMUX 14

***Table 18-23* • IOMUX 14**

Pad Name	Pad Ports	IOMUX_14_CR	IOMUX 14 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_1_SDA/GPIO_30	I		I2C_1_SDAI			GPI_30		
	O			GND			GPO_30	
	OE				~(I2C_1_SDAO)			GPOE_30
	PU	IOMUX_14_PU						
	PD	IOMUX_14_PD						
	ST	IOMUX_14_ST						

IOMUX 15

***Table 18-24* • IOMUX 15**

Pad Name	Pad Ports	IOMUX_15_CR	IOMUX 15 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
I2C_1_SCL/GPIO_31	I		I2C_1_SCLI			GPI_31		
	O			GND			GPO_31	
	OE				~(I2C_1_SCLO)			GPOE_31
	PU	IOMUX_15_PU						
	PD	IOMUX_15_PD						
	ST	IOMUX_15_ST						

IOMUX 25

***Table 18-25* • IOMUX 25**

Name	Pad Ports	IOMUX_25_CR	IOMUX 25 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_0/ioUXWbYvZ	I		GPI_0			M2F[9]		
	O			GPO_0			F2M[9]	
	OE				GPOE_0			F2M_OE[9]
	PU	IOMUX_25_PU						
	PD	IOMUX_25_PD						
	ST	IOMUX_25_ST						

IOMUX 26

***Table 18-26* • IOMUX 26**

Name	Pad Ports	IOMUX_26_CR	IOMUX 26 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_1/ioUXWbYvZ	I		GPI_1			M2F[10]		
	O			GPO_1			F2M[10]	
	OE				GPOE_1			F2M_OE[10]
	PU	IOMUX_26_PU						
	PD	IOMUX_26_PD						
	ST	IOMUX_26_ST						

IOMUX 27

***Table 18-27* • IOMUX 27**

Name	Pad Ports	IOMUX_27_CR	IOMUX 27 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_2/ioUXWbYvZ	I		GPI_2			M2F[11]		
	O			GPO_2			F2M[11]	
	OE				GPOE_2			F2M_OE[11]
	PU	IOMUX_27_PU						
	PD	IOMUX_27_PD						
	ST	IOMUX_27_ST						

IOMUX 28

***Table 18-28* • IOMUX 28**

Name	Pad Ports	IOMUX_28_CR	IOMUX 28 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_3/ioUXWbYvZ	I		GPI_3			M2F[12]		
	O			GPO_3			F2M[12]	
	OE				GPOE_3			F2M_OE[12]
	PU	IOMUX_28_PU						
	PD	IOMUX_28_PD						
	ST	IOMUX_28_ST						

IOMUX 29

***Table 18-29* • IOMUX 29**

Name	Pad Ports	IOMUX_29_CR	IOMUX 29 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_4/ioUXWbYvZ	I		GPI_4			M2F[13]		
	O			GPO_4			F2M[13]	
	OE				GPOE_4			F2M_OE[13]
	PU	IOMUX_29_PU						
	PD	IOMUX_29_PD						
	ST	IOMUX_29_ST						

IOMUX 30

***Table 18-30* • IOMUX 30**

Name	Pad Ports	IOMUX_30_CR	IOMUX 30 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_5/ioUXWbYvZ	I		GPI_5			M2F[14]		
	O			GPO_5			F2M[14]	
	OE				GPOE_5			F2M_OE[14]
	PU	IOMUX_30_PU						
	PD	IOMUX_30_PD						
	ST	IOMUX_30_ST						

IOMUX 31

***Table 18-31* • IOMUX 31**

Name	Pad Ports	IOMUX_31_CR	IOMUX 31 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_6/ioUXWbYvZ	I		GPI_6			M2F[15]		
	O			GPO_6			F2M[15]	
	OE				GPOE_6			F2M_OE[15]
	PU	IOMUX_31_PU						
	PD	IOMUX_31_PD						
	ST	IOMUX_31_ST						

IOMUX 32

***Table 18-32* • IOMUX 32**

Name	Pad Ports	IOMUX_32_CR	IOMUX 32 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_7/ioUXWbYvZ	I		GPI_7			M2F[16]		
	O			GPO_7			F2M[16]	
	OE				GPOE_7			F2M_OE[16]
	PU	IOMUX_32_PU						
	PD	IOMUX_32_PD						
	ST	IOMUX_32_ST						

IOMUX 33

***Table 18-33* • IOMUX 33**

Name	Pad Ports	IOMUX_33_CR	IOMUX 33 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_8/ioUXWbYvZ	I		GPI_8			M2F[17]		
	O			GPO_8			F2M[17]	
	OE				GPOE_8			F2M_OE[17]
	PU	IOMUX_33_PU						
	PD	IOMUX_33_PD						
	ST	IOMUX_33_ST						

IOMUX 34

***Table 18-34* • IOMUX 34**

Name	Pad Ports	IOMUX_34_CR	IOMUX 34 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_9/ioUXWbYvZ	I		GPI_9			M2F[18]		
	O			GPO_9			F2M[18]	
	OE				GPOE_9			F2M_OE[18]
	PU	IOMUX_34_PU						
	PD	IOMUX_34_PD						
	ST	IOMUX_34_ST						

IOMUX 35

***Table 18-35* • IOMUX 35**

Name	Pad Ports	IOMUX_35_CR	IOMUX 35 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_10/ioUXWbYvZ	I		GPI_10			M2F[19]		
	O			GPO_10			F2M[19]	
	OE				GPOE_10			F2M_OE[19]
	PU	IOMUX_35_PU						
	PD	IOMUX_35_PD						
	ST	IOMUX_35_ST						

IOMUX 36

***Table 18-36* • IOMUX 36**

Name	Pad Ports	IOMUX_36_CR	IOMUX 36 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_11/ioUXWbYvZ	I		GPI_11			M2F[20]		
	O			GPO_11			F2M[20]	
	OE				GPOE_11			F2M_OE[20]
	PU	IOMUX_36_PU						
	PD	IOMUX_36_PD						
	ST	IOMUX_36_ST						

IOMUX 37

***Table 18-37* • IOMUX 37**

Name	Pad Ports	IOMUX_37_CR	IOMUX 37 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_12/ioUXWbYvZ	I		GPI_12			M2F[21]		
	O			GPO_12			F2M[21]	
	OE				GPOE_12			F2M_OE[21]
	PU	IOMUX_37_PU						
	PD	IOMUX_37_PD						
	ST	IOMUX_37_ST						

IOMUX 38

***Table 18-38* • IOMUX 38**

Name	Pad Ports	IOMUX_38_CR	IOMUX 38 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_13/ioUXWbYvZ	I		GPI_13			M2F[22]		
	O			GPO_13			F2M[22]	
	OE				GPOE_13			F2M_OE[22]
	PU	IOMUX_38_PU						
	PD	IOMUX_38_PD						
	ST	IOMUX_38_ST						

IOMUX 39

***Table 18-39* • IOMUX 39**

Name	Pad Ports	IOMUX_39_CR	IOMUX 39 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_14/ioUXWbYvZ	I		GPI_14			M2F[23]		
	O			GPO_14			F2M[23]	
	OE				GPOE_14			F2M_OE[23]
	PU	IOMUX_39_PU						
	PD	IOMUX_39_PD						
	ST	IOMUX_39_ST						

IOMUX 40

***Table 18-40* • IOMUX 40**

Name	Pad Ports	IOMUX_40_CR	IOMUX 40 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_15/ioUXWbYvZ	I		GPI_15			M2F[24]		
	O			GPO_15			F2M[24]	
	OE				GPOE_15			F2M_OE[24]
	PU	IOMUX_40_PU						
	PD	IOMUX_40_PD						
	ST	IOMUX_40_ST						

IOMUX 41

***Table 18-41* • IOMUX 41**

Name	Pad Ports	IOMUX_41_CR	IOMUX 41 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_16/ioUXWbYvZ	I		GPI_16			M2F[25]		
	O			GPO_16			F2M[25]	
	OE				GPOE_16			F2M_OE[25]
	PU	IOMUX_41_PU						
	PD	IOMUX_41_PD						
	ST	IOMUX_41_ST						

IOMUX 42

***Table 18-42* • IOMUX 42**

Name	Pad Ports	IOMUX_42_CR	IOMUX 42 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_17/ioUXWbYvZ	I		GPI_17			M2F[26]		
	O			GPO_17			F2M[26]	
	OE				GPOE_17			F2M_OE[26]
	PU	IOMUX_42_PU						
	PD	IOMUX_42_PD						
	ST	IOMUX_42_ST						

IOMUX 43

***Table 18-43* • IOMUX 43**

Name	Pad Ports	IOMUX_43_CR	IOMUX 43 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_18/ioUXWbYvZ	I		GPI_18			M2F[27]		
	O			GPO_18			F2M[27]	
	OE				GPOE_18			F2M_OE[27]
	PU	IOMUX_43_PU						
	PD	IOMUX_43_PD						
	ST	IOMUX_43_ST						

IOMUX 44

***Table 18-44* • IOMUX 44**

Name	Pad Ports	IOMUX_44_CR	IOMUX 44 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_19/ioUXWbYvZ	I		GPI_19			M2F[28]		
	O			GPO_19			F2M[28]	
	OE				GPOE_19			F2M_OE[28]
	PU	IOMUX_44_PU						
	PD	IOMUX_44_PD						
	ST	IOMUX_44_ST						

IOMUX 45

***Table 18-45* • IOMUX 45**

Name	Pad Ports	IOMUX_45_CR	IOMUX 45 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_20/ioUXWbYvZ	I		GPI_20			M2F[29]		
	O			GPO_20			F2M[29]	
	OE				GPOE_20			F2M_OE[29]
	PU	IOMUX_45_PU						
	PD	IOMUX_45_PD						
	ST	IOMUX_45_ST						

IOMUX 46

***Table 18-46* • IOMUX 46**

Name	Pad Ports	IOMUX_46_CR	IOMUX 46 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_21/ioUXWbYvZ	I		GPI_21			M2F[30]		
	O			GPO_21			F2M[30]	
	OE				GPOE_21			F2M_OE[30]
	PU	IOMUX_46_PU						
	PD	IOMUX_46_PD						
	ST	IOMUX_46_ST						

IOMUX 47

***Table 18-47* • IOMUX 47**

Name	Pad Ports	IOMUX_47_CR	IOMUX 47 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_22/ioUXWbYvZ	I		GPI_22			M2F[31]		
	O			GPO_22			F2M[31]	
	OE				GPOE_22			F2M_OE[31]
	PU	IOMUX_47_PU						
	PD	IOMUX_47_PD						
	ST	IOMUX_47_ST						

IOMUX 48

***Table 18-48* • IOMUX 48**

Name	Pad Ports	IOMUX_48_CR	IOMUX 48 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_23/ioUXWbYvZ	I		GPI_23			M2F[32]		
	O			GPO_23			F2M[32]	
	OE				GPOE_23			F2M_OE[32]
	PU	IOMUX_48_PU						
	PD	IOMUX_48_PD						
	ST	IOMUX_48_ST						

IOMUX 49

***Table 18-49* • IOMUX 49**

Name	Pad Ports	IOMUX_49_CR	IOMUX 49 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_24/ioUXWbYvZ	I		GPI_24			M2F[33]		
	O			GPO_24			F2M[33]	
	OE				GPOE_24			F2M_OE[33]
	PU	IOMUX_49_PU						
	PD	IOMUX_49_PD						
	ST	IOMUX_49_ST						

IOMUX 50

***Table 18-50* • IOMUX 50**

Name	Pad Ports	IOMUX_50_CR	IOMUX 50 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_25/ioUXWbYvZ	I		GPI_25			M2F[34]		
	O			GPO_25			F2M[34]	
	OE				GPOE_25			F2M_OE[34]
	PU	IOMUX_50_PU						
	PD	IOMUX_50_PD						
	ST	IOMUX_50_ST						

IOMUX 51

***Table 18-51* • IOMUX 51**

Name	Pad Ports	IOMUX_51_CR	IOMUX 51 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_26/ioUXWbYvZ	I		GPI_26			M2F[35]		
	O			GPO_26			F2M[35]	
	OE				GPOE_26			F2M_OE[35]
	PU	IOMUX_51_PU						
	PD	IOMUX_51_PD						
	ST	IOMUX_51_ST						

IOMUX 52

***Table 18-52* • IOMUX 52**

Name	Pad Ports	IOMUX_52_CR	IOMUX 52 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_27/ioUXWbYvZ	I		GPI_27			M2F[36]		
	O			GPO_27			F2M[36]	
	OE				GPOE_27			F2M_OE[36]
	PU	IOMUX_52_PU						
	PD	IOMUX_52_PD						
	ST	IOMUX_52_ST						

IOMUX 53

***Table 18-53* • IOMUX 53**

Name	Pad Ports	IOMUX_53_CR	IOMUX 53 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_28/ioUXWbYvZ	I		GPI_28			M2F[37]		
	O			GPO_28			F2M[37]	
	OE				GPOE_28			F2M_OE[37]
	PU	IOMUX_53_PU						
	PD	IOMUX_53_PD						
	ST	IOMUX_53_ST						

IOMUX 54

***Table 18-54* • IOMUX 54**

Name	Pad Ports	IOMUX_54_CR	IOMUX 54 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_29/ioUXWbYvZ	I		GPI_29			M2F[38]		
	O			GPO_29			F2M[38]	
	OE				GPOE_29			F2M_OE[38]
	PU	IOMUX_54_PU						
	PD	IOMUX_54_PD						
	ST	IOMUX_54_ST						

IOMUX 55

***Table 18-55* • IOMUX 55**

Name	Pad Ports	IOMUX_55_CR	IOMUX 55 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_30/ioUXWbYvZ	I		GPI_30			M2F[39]		
	O			GPO_30			F2M[39]	
	OE				GPOE_30			F2M_OE[39]
	PU	IOMUX_55_PU						
	PD	IOMUX_55_PD						
	ST	IOMUX_55_ST						

IOMUX 56

***Table 18-56* • OMUX 56**

Name	Pad Ports	IOMUX_56_CR	IOMUX 56 Ports					
			IN_A	OUT_A	OE_A	IN_B	OUT_B	OE_B
GPIO_31/ioUXWbYvZ	I		GPI_31			M2F[40]		
	O			GPO_31			F2M[40]	
	OE				GPOE_31			F2M_OE[40]
	PU	IOMUX_56_PU						
	PD	IOMUX_56_PD						
	ST	IOMUX_56_ST						

SmartFusion MSS GPIO Application Development

This section provides an overview of the design flow for SmartFusion devices to facilitate application development using GPIOs. The 32 bits of the GPIO block are shared among MSS peripherals and MSS user I/Os as summarized in Table 18-57. Using the MSS GPIO peripheral requires minimal configuration of the GPIO and MSS I/O blocks in MSS configurator and I/O assignment in Libero SoC. The user application would then use predefined functions in GPIO drivers to perform the general purpose I/O functions with the configured I/Os.

***Table 18-57* • GPIO Block Multiplexing**

GPIO[0:15]	Multiplexed with MSS User I/Os
GPIO[16:19]	Multiplexed with SPI0
GPIO[20:21]	Multiplexed with UART0
GPIO[22:23]	Multiplexed with I2C0
GPIO[24:25]	Multiplexed with SPI1
GPIO[26:29]	Multiplexed with UART1
GPIO[30:31]	Multiplexed with I2C1

Developing User Application Code with GPIO Drivers

User application code can be developed and debugged using any of the three supported embedded software development tools, SoftConsole, Keil Microcontroller Development Kit (MDK) µVision, and IAR Embedded Workbench. Microsemi provides a set of GPIO Drivers that can be generated from the MSS configurator or from the Firmware Catalog. These drivers are common for all three tool flows. However, the Cortex Microcontroller Software Interface Standard (CMSIS) access layer is dependent on the tool flow selected and should be chosen based on the specific tool flow user is implementing.

The MSS GPIO drivers allow quick application code development using the SmartFusion MSS GPIO without having to manually read and write the MSS System Registers. MSS GPIO Drivers are efficient and flexible. For specific details on drivers, refer to the SmartFusion MSS Configurators and Drivers User's Guides.

Steps for Application Development Using MSS GPIO Drivers

As a first step, tool-specific CMSIS files need to be imported into the project along with GPIO drivers. The mss_gpio.h header file, which defines GPIO function prototypes, must be included in the application code to obtain access to the GPIO functions, as shown below:

Example

```
#include "mss_gpio.h"
```

Using the GPIO driver functions involves four distinct stages: initialization, configuration, reading and setting GPIO state, and Interrupt control.

Initializing GPIOs

The MSS GPIO driver is initialized through a call to the function MSS_GPIO_init(). This function must be called before any other GPIO driver functions can be called.

Example

```
MSS_GPIO_init()
```

Configuring GPIOs

Each GPIO port is individually configured through a call to the MSS_GPIO_config() function. This function sets the direction of the I/O as input, output, or inout.

Example

```
MSS_GPIO_config(MSS_GPIO_25 , MSS_GPIO_OUTPUT_MODE);
```

If the GPIO is configured as input, then the associated interrupt can also be configured as either level or edge sensitive. For level sensitive the option is to be high or low. For edge it is either rising, falling or both.

Example

```
MSS_GPIO_config( MSS_GPIO_16 , MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_LEVEL_LOW );
```

Example

```
MSS_GPIO_config( MSS_GPIO_22 , MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_NEGATIVE);
```

Reading GPIO State

The state of the GPIO ports configured as inputs or outputs can be read using the functions MSS_GPIO_get_inputs() and MSS_GPIO_get_outputs().

Example

```
uint32_t gpio_inputs;
gpio_inputs = MSS_GPIO_get_inputs();
```

Note: The ability to read the current state of outputs is useful in control applications where the value driven on output need to be modulated.

Setting GPIO State

The state of GPIO ports configured as outputs or inout can be set using the functions MSS_GPIO_set_output(), MSS_GPIO_set_outputs(), and MSS_GPIO_drive_inout().The inout port can be driving high or low or can be in high impedance states.

Example

```
MSS_GPIO_set_output(MSS_GPIO_26,0); // Sets GPIO26 Low
```

Example

```
uint32_t gpio_outputs;
gpio_outputs = MSS_GPIO_get_outputs(); // Get current output state
gpio_outputs &= ~( MSS_GPIO_2_MASK | MSS_GPIO_4_MASK ); // Flip states of GPIO2 & 4
MSS_GPIO_set_outputs(  gpio_outputs ); // Write new value to GPIO
```

Example

```
MSS_GPIO_drive_inout( MSS_GPIO_7, MSS_GPIO_HIGH_Z );
```

GPIO Generated Interrupt Control

For GPIOs that are configured as inputs with interrupts defined, functions MSS_GPIO_enable_irq(), MSS_GPIO_disable_irq(), and MSS_GPIO_clear_irq() help manage the interrupts.

Example

```
MSS_GPIO_enable_irq( MSS_GPIO_8 );//Would enable the interrupt generation for input
GPIO_8
```

Example

```
MSS_GPIO_disable_irq() // To disable interrupt generation for the specified GPIO
input
```

Example

```
MSS_GPIO_clear_irq() //The function clears the GPIO interrupt as well as the
Cortex-M3 interrupt controller
```

The SmartFusion MSS GPIO application development overview provided in this section should enable the user to rapidly and easily adopt the SmartFusion MSS GPIO peripheral in their embedded application. For additional details on the three tool flows, refer to the SmartFusion tutorials available at http://www.microsemi.com/soc/products/smartfusion/docs.aspx.

19 – Fabric Interface and IOMUX

The fabric interface consists of the fabric interface controller (FIC), the fabric interface interrupt controller (FIIC), IOMUXes, and signals from the analog compute engine (ACE), the signal conditioning blocks (SCBs), the voltage regulator power supply monitor (VR/PSM), and a handful of miscellaneous signals. The blocks and signals of the fabric interface are shown in Figure 19-1.

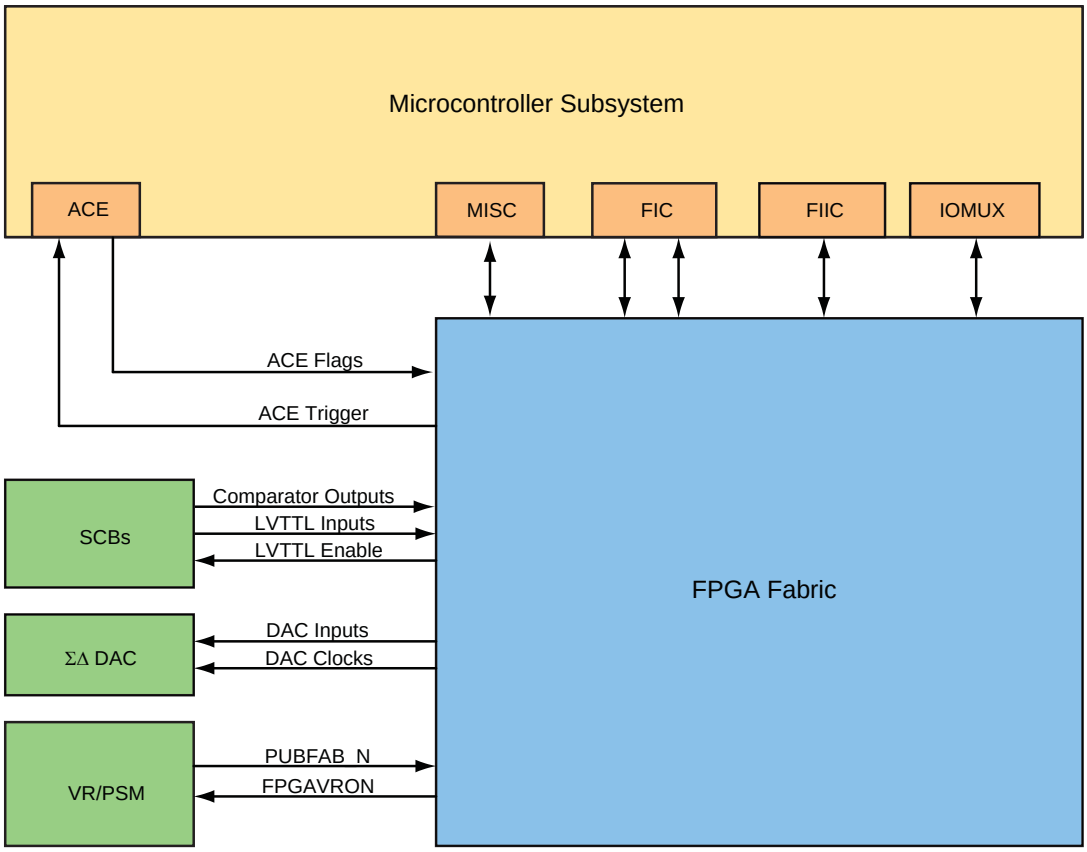


***Figure 19-1* • Fabric Interfaces**

Fabric Interface Controller

The fabric interface controller (FIC) is part of the microcontroller subsystem (MSS) and performs an AHB to AHB or AHB to APB bridging function between the AHB bus matrix and an AHB or APB bus in the FPGA fabric. The FIC consumes no FPGA resources. It provides two bus interfaces between the (MSS) and the fabric. The first is mastered by the MSS and has slaves in the fabric and the second has a master in the fabric and slaves in the MSS, as depicted in Figure 19-2.

The interfaces to the fabric can be 32-bit AHB, 32-bit APB, or 16-bit APB. The address and data buses between the FIC and the FPGA fabric are common to both the AHB and APB interfaces; hence only one type of interface can be enabled at any time. However, separate groups of signals are used for the AHB and APB control signals. The type of interface chosen is determined at design time.

In addition to the choice of AHB or APB interfaces between the MSS and the fabric, a number of options related to relative clock frequencies and pipelining of transactions are available. In pipelined mode, the ratio between the MSS FCLK frequency and the frequency of the AHB/APB circuitry in the FPGA fabric can be 1:1, 2:1, or 4:1. If the interfaces are configured as AHB and the clock ratio is 1:1, it is possible to select a bypass mode in which signals to and from the fabric are not registered. In bypass mode, fewer clock cycles are required to complete each transaction but the overall system frequency may be lower than is possible in pipelined mode.

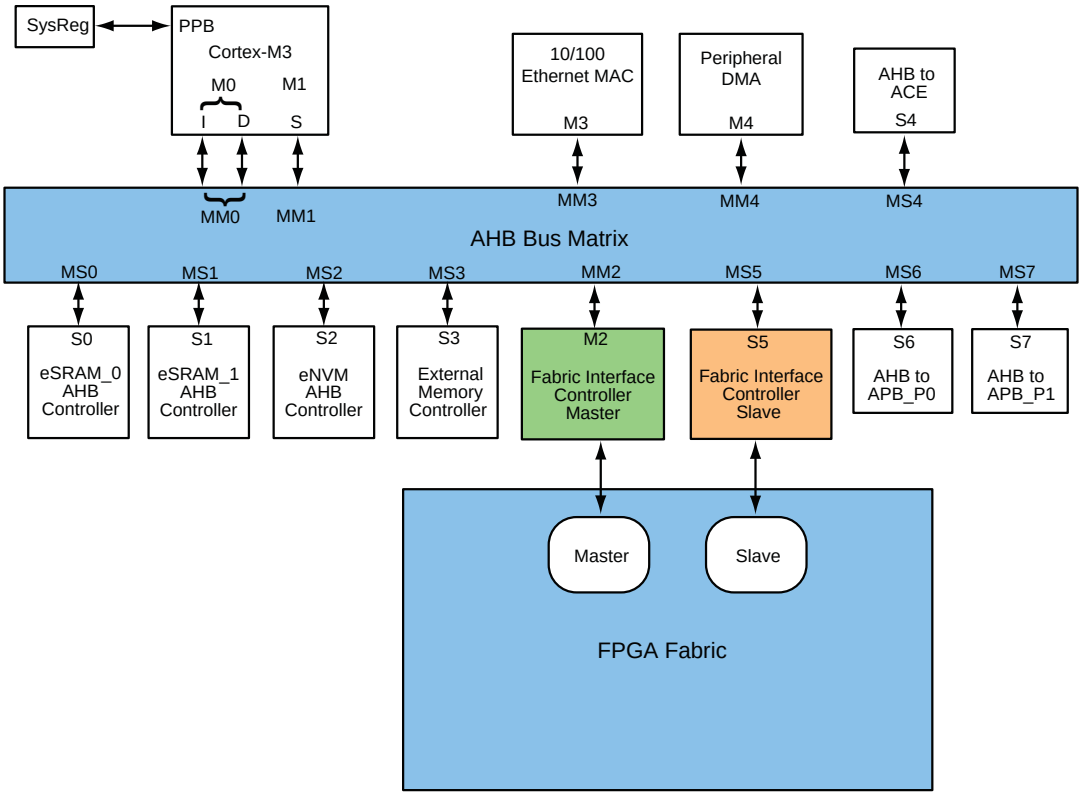


***Figure 19-2* • Fabric Interface Controller System Overview**

As shown in Figure 19-4 on page 346, on the MSS side of the FIC there are two (master and slave) AHB interfaces to the AHB bus matrix. On the FPGA side of the FIC, the master and slave interfaces can be

either AHB or APB. Both interfaces on the fabric side always use the same protocol. In other words, one cannot be AHB while the other is APB, as shown in Figure 19-3.

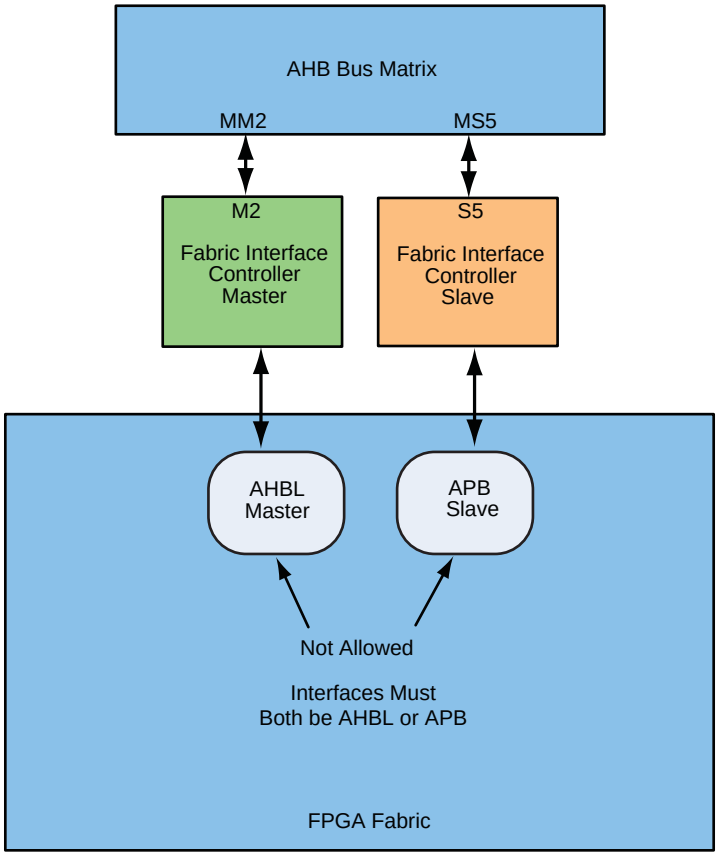


***Figure 19-3* • Mismatched FIC Interfaces**

When implementing peripherals in the fabric, these are the choices available, but the first case is the most common.

1. If you are using APB (or APB3) peripherals—for example, CoreUARTapb—use CoreAPB3 connected to the fabric interface.
2. If you are using AHB peripherals, use CoreAHBLite connected to the fabric interface.
3. If you are using mixed AHB and APB peripherals, use CoreAHBLite – CoreAHB2APB3 – CoreAPB3.

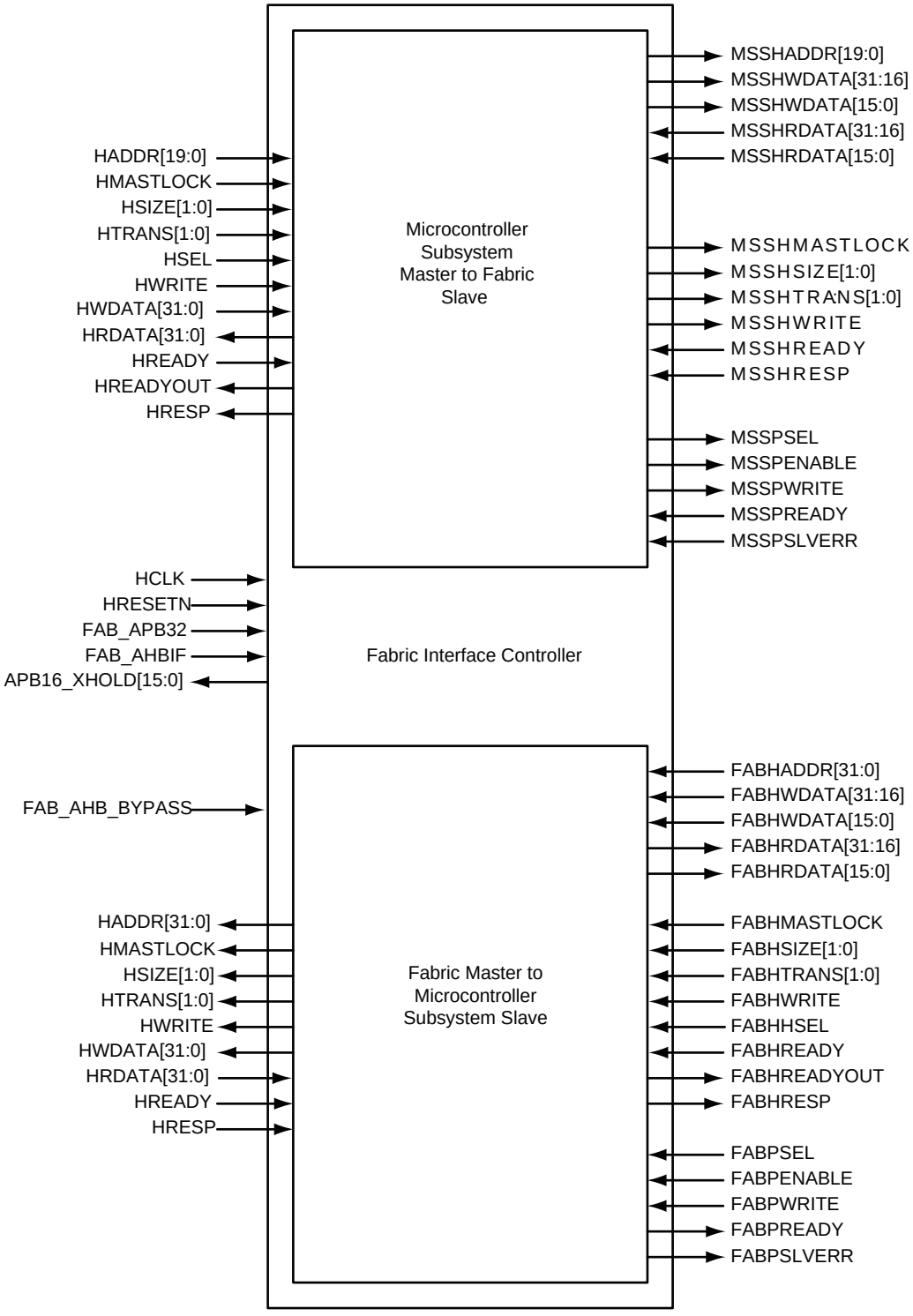


Figure 19-4 **• Fabric Interface Control Signal List**

Fabric Interface and IOMUX Register Map

Table 19-1 gives descriptions for the registers mentioned throughout this document.

***Table 19-1* • Fabric Interface and IOMUX Register Map**

Register Name	Address	R/W	Reset Value	Description
MSSIRQ_EN_0	0x40007000	R/W	0	Enables/disables interrupt sources for MSSINT[0]
MSSIRQ_EN_1	0x40007004	R/W	0	Enables/disables interrupt sources for MSSINT[1]
MSSIRQ_EN_2	0x40007008	R/W	0	Enables/disables interrupt sources for MSSINT[2]
MSSIRQ_EN_3	0x4000700C	R/W	0	Enables/disables interrupt sources for MSSINT[3]
MSSIRQ_EN_4	0x40007010	R/W	0	Enables/disables interrupt sources for MSSINT[4]
MSSIRQ_EN_5	0x40007014	R/W	0	Enables/disables interrupt sources for MSSINT[5]
MSSIRQ_EN_6	0x40007018	R/W	0	Enables/disables interrupt sources for MSSINT[6]
MSSIRQ_EN_7	0x4000701C	R/W	0	Enables/disables interrupt sources for MSSINT[7]
MSSIRQ_SRC_0	0x40007020	R/W	0	Source of interrupt for MSSINT[0]
MSSIRQ_SRC_1	0x40007024	R/W	0	Source of interrupt for MSSINT[1]
MSSIRQ_SRC_2	0x40007028	R/W	0	Source of interrupt for MSSINT[2]
MSSIRQ_SRC_3	0x4000702C	R/W	0	Source of interrupt for MSSINT[3]
MSSIRQ_SRC_4	0x40007030	R/W	0	Source of interrupt for MSSINT[4]
MSSIRQ_SRC_5	0x40007034	R/W	0	Source of interrupt for MSSINT[5]
MSSIRQ_SRC_6	0x40007038	R/W	0	Source of interrupt for MSSINT[6]
MSSIRQ_SRC_7	0x4000703C	R/W	0	Source of interrupt for MSSINT[7]
FIIC_MR	0x40007040	R/W	0	Fabric interface interrupt controller mode register
FAB_IF_CR	0xE004206C	R/W	0xC	Controls the functionality of the fabric interface controller
FAB_APB_HIWORD_DR	0xE0042070	R	0x0	16-bit APB holding register within the FIC
IOMUX _n_CR (n = 0)	0xE0042100	R/W	0x0	Configures IOMUX 0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
IOMUX _n_CR (n = 82)	0xE0042248	R/W	0x0	Configures IOMUX 82

Fabric Interface Controller Register

The FAB_IF_CR is used to select the type of the interfaces on the fabric side of the FIC, and to select bypass mode or not. This register is located in the SYSREG block at address 0xE004206C. Individual bit descriptions are given in Table 19-2. Table 19-3 lists the truth table for selecting the appropriate bus type on the FPGA side of the FIC.

***Table 19-2* • FAB_IF_CR**

Bit Number	Name	R/W	Reset Value	Function
31:4	Reserved	R/W	–	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAB_APB32	R/W	1	See Table 19-3.
2	FAB_AHBIF	R	1	See Table 19-3.
1	Reserved	R/W	–	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FAB_AHB_BYPASS	R/W	0	0 = FIC is in pipeline mode. 1 = FIC is in bypass mode.

***Table 19-3* • Bus Type Selection**

FAB_IF_CR		
FAB_APB32	**FAB_AHBIF**	
Bit 3	**Bit 2**	**Interface**
0	0	16-bit APB
0	1	32-bit AHB
1	0	32-bit APB
1	1	32-bit AHB (default state)

If the MSS FCLK to FPGA fabric clock ratio is 1:1 and the FIC is configured to operate in 32-bit AHB mode, as described in Table 19-3, the FIC can be configured to operate in bypass mode. Signals to and from the fabric are not registered in bypass mode. If, however, the FIC is configured in APB mode (16- or 32-bit) or the clock ratio between the MSS and the fabric is not 1:1, the user must set the FIC to pipeline mode, as outlined in Table 19-4. The configuration of FAB_IF_CR and FAB_AHB_BYPASS should not be performed dynamically. The MSS configurator creates the system boot initialization code that initializes the state of these fields to the user's desired configuration.

***Table 19-4* • FIC Bypass Mode Selection**

FAB_IF_CR	
FAB_AHB_BYPASS	
Bit 0	**Interface**
0	The fabric interface is configured in pipeline mode; that is, registered mode (default state).
1	The fabric interface is configured for bypass mode.

Fabric Interface Clocks

All logic within the FIC is clocked by the main MSS system clock, FCLK. The AHB/APB logic within the FPGA fabric can be clocked by GLA or GLB, which are generated by the MSS_CCC. Refer to the "PLLs, Clock Conditioning Circuitry, and On-Chip Crystal Oscillators" section on page 109.

The rising edges of GLB must occur at the same time as a rising edge of FCLK. This lining up of the rising edges of GLB and FCLK is controlled by configuring a delay element within the MSS_CCC. The delay value is factory calibrated.

MSS Master Interface

The MSS master interface side of the FIC can communicate with either an FPGA fabric AHB slave or an APB v3.0 compliant slave, as described below. Microsemi provides numerous AHB and APB v3.0 compliant cores for easy instantiation into the FPGA fabric. Users must first instantiate either the CoreAHBLite or CoreAPB3 soft IP into the fabric to allow further instantiation of soft FPGA AHB/APB masters or slaves.

MSS Master to FPGA AHB Slave Interface

The fabric interface controller allows the AHB masters in the MSS to communicate with AHB compliant slaves in the FPGA fabric, as shown in Figure 19-5. The MSS Master AHB interface passes all incoming AHB transactions to the fabric with no error checking. FPGA AHB slaves must handle any error conditions. The least significant 20 bits of the MSS address bus are passed to the FPGA fabric, along with the 32-bit read and write data buses. MSS AHB masters can perform byte, halfword, and word accesses to an FPGA fabric AHB slave.

Misaligned accesses are not supported and result in invalid data transfers in registered mode. The data transfer to the fabric proceeds as normal, but HRESP is asserted to the MSS AHB master at the end of the transfer. HRESP is not asserted if in bypass mode and the transfer proceeds as normal.

When no AHB slaves are programmed in the fabric, MSSHREADY must be tied high. SmartDesign will automatically tie this high if needed.

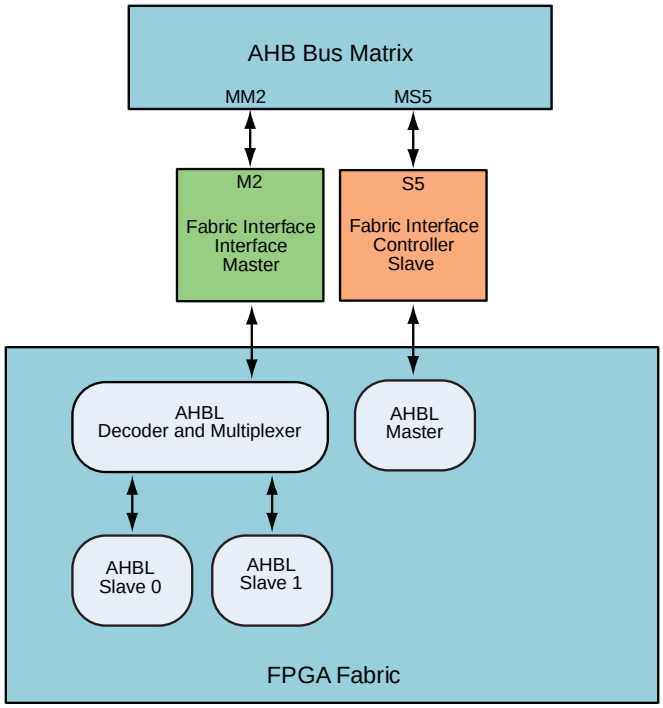


***Figure 19-5* • MSS Master to AHB Fabric Slave**

MSS Master to FPGA APB Slave Interface

The fabric interface allows AHB masters in the MSS to communicate with the fabric as an APB v3.0 compliant slave, as shown in Figure 19-6.

The MSS master APB interface provides an interface to the fabric which, on its own, is suitable for connection to a single APB slave. Users that require multiple APB slaves in the fabric must use the single MSSPSEL select signal provided to enable an additional address decoder to generate additional PSEL select signals, one for each slave. In addition, the user is required to perform multiplexing of the APB PRDATA, PSLVERR, and PREADY signals from each instantiated slave to drive the single MSSPRDATA, MSSPSLVERR, and MSSPREADY input ports of the MSS master APB interface. Microsemi provides CoreAPB3 to perform these functions.

The least significant 20 bits of the MSS address bus are passed to the FPGA fabric, along with either 16-bit or 32-bit read and write data buses (depending on the setting of the FAB_AHB32 and FAB_AHBIF bits in FAB_IF_CR).

In the case where the interfaces to the fabric are configured as 16-bit APB, an MSS master can perform byte or halfword accesses to 16-bit APB slaves in the fabric. These accesses can be performed only on word-aligned addresses. Byte or halfword accesses to 16-bit APB slaves on non-word aligned addresses are not supported and result in invalid data transfers. Word accesses to 16-bit APB slaves are not supported and result in invalid data transfers.

In the case where the interfaces to the fabric are configured as 32-bit APB, an MSS master can perform byte, halfword, or word accesses to 32-bit APB slaves in the fabric. These accesses can be performed only on word-aligned addresses. Byte, halfword, or word accesses to 32-bit APB slaves on non-word aligned addresses are not supported and result in invalid data transfers.

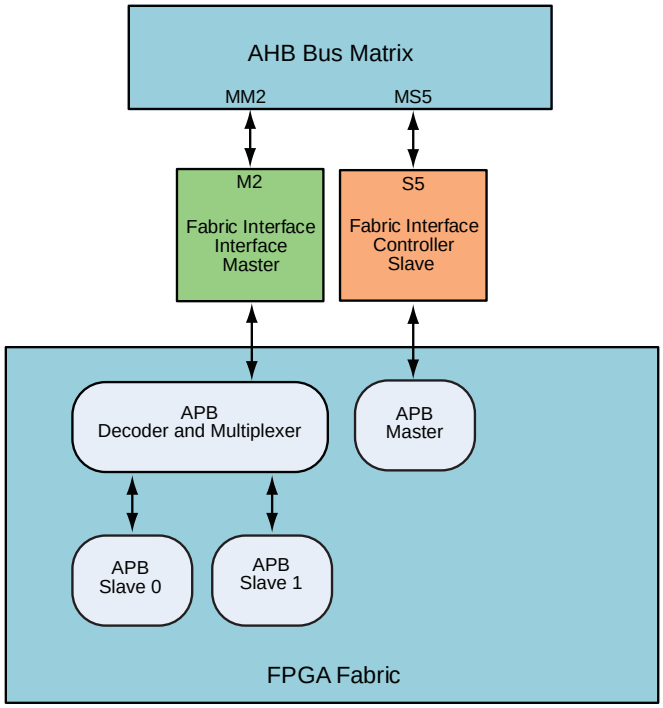


***Figure 19-6* • MSS AHB Master to APB Slave**

Fabric Master Interface

The fabric master interface of the FIC allows either an AHB or an APB v3.0 compliant master in the FPGA fabric to communicate with AHB and APB slaves in the MSS, as described below.

The AHB_MATRIX_CR, FAB_PROT_BASE_CR, and FAB_PROT_SIZE_CR system registers, which are only accessible by the ARM Cortex-M3 processor, can be used to limit access by a fabric-based master to some or all of the slaves in the MSS. See the "AHB Bus Matrix" section on page 15 for more details.

Fabric Master AHB Interface

The fabric master AHB interface allows a user-instantiated AHB compliant master in the fabric to communicate with slaves in the MSS, as shown in Figure 19-7. The fabric master AHB interface passes all incoming AHB transactions to the MSS; all transactions are passed with no error checking performed. The fabric master AHB interface provides for a 32-bit address, a 32-bit read, and a 32-bit write data bus into the MSS.

An AHB master in the fabric can perform byte, halfword, and word accesses to MSS AHB slaves. Misaligned accesses are not supported and result in invalid data transfers when in pipeline mode. In bypass mode, there is no indication of an invalid transfer.

If accessing an MSS APB peripheral, a fabric-based AHB master should use only word-aligned addresses, because all locations in the APB peripherals are at word-aligned offsets. Non-word-aligned addresses are not supported and result in invalid data transfers.

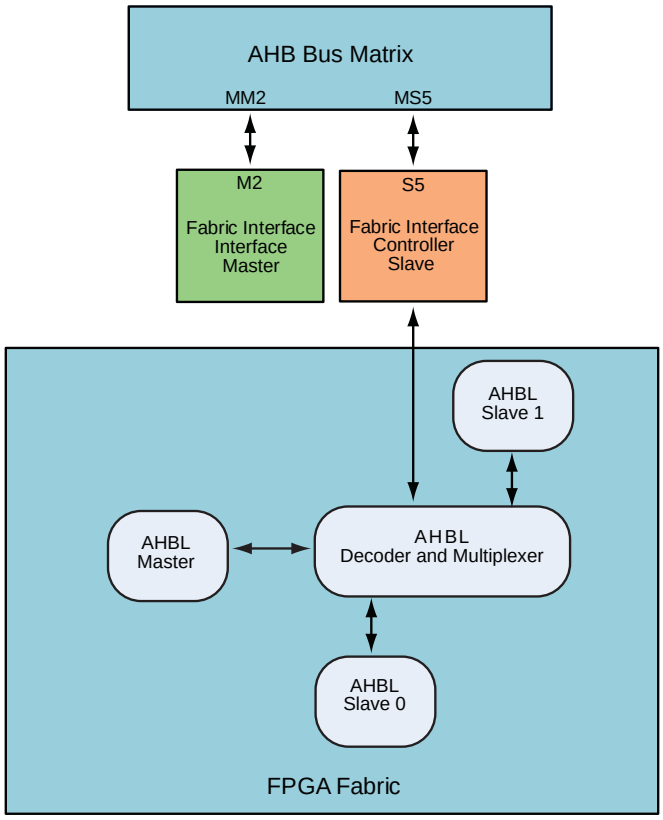


***Figure 19-7* • Fabric Master to MSS Slave**

Fabric Master APB Interface

The fabric master APB interface allows a user-instantiated APB v3.0 compliant master in the FPGA fabric to communicate with AHB slaves in the MSS, as shown in Figure 19-8 on page 353. The fabric master APB interface passes all incoming APB transactions that are transfer size aligned to the MSS; all transactions are passed with no error checking performed. The fabric master APB interface provides for a 32-bit address, and a 32-bit or 16-bit read/write data bus into the MSS. Data bus width is dependent on FAB_APB32 and FAB_AHBIF bit settings, as described in Table 19-3 on page 348.

When the data width of the fabric master APB interface is set to 16 bits, a register named APB16_XHOLD will be used to hold the upper halfword of the 32-bit AHB transactions to/from the AHB bus matrix. All transfers initiated by a fabric APB master result in 32-bit AHB transactions on the MSS side of the FIC. The APB16_XHOLD register is located at address 0x40030000 and is accessible only by a fabric master.

When issuing a write from a 16-bit APB fabric master, the master must first ensure that the APB16_XHOLD register is loaded with a valid upper halfword before writing the lower halfword to the actual (word aligned) address which is the destination of the write.

When issuing a read from a 16-bit APB fabric master, the master should read from the (word-aligned) address that is the source of the read. The lower halfword will be read by the fabric master APB interface during this transfer and the upper halfword will be stored in the APB16_XHOLD register. The fabric master APB interface can then subsequently read the upper halfword from the APB16_XHOLD register in a following transfer, if required.

To aid debugging, the Cortex-M3 processor can observe the value stored in the APB16_XHOLD register via the FAB_APB_HIWORD_DR system register located at address 0xE0042070, as shown in Table 19-1 on page 347 and Table 19-5. FAB_APB_HIWORD_DR contains the state of the APB16_XHOLD register. It is read only from the Cortex-M3 processor. The contents of this register depends on whether the last operation from the fabric master APB interface was a read or a write. If it was a read, this register contains the upper 16 bits of the read data. If it was a write, this register contains the upper 16 bits of the write data.

***Table 19-5* • FAB_APB_HIWORD_DR**

Bit Number	Name	R/W	Reset Value	Function
31:16	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	APB16_XHOLD	R/W	0	This signal contains the state of the APB16 holding register within the fabric interface block. It is read only from the Cortex-M3 processor. The contents of this register depend on whether the last operation from the APB fabric master was a read or a write. If it was a read, then this register contains the upper 16 bits of the read data. If it was a write, then this register contains the upper 16 bits of the write data.

All transfers initiated by a fabric APB master, other than those addressed to the APB16_XHOLD register, result in 32-bit AHB transactions to the AHB bus matrix. When addressing locations in the MSS other than the APB16_XHOLD register, a fabric APB master must always provide a word-aligned address. Non-word-aligned addresses are not supported and result in invalid data transfers.

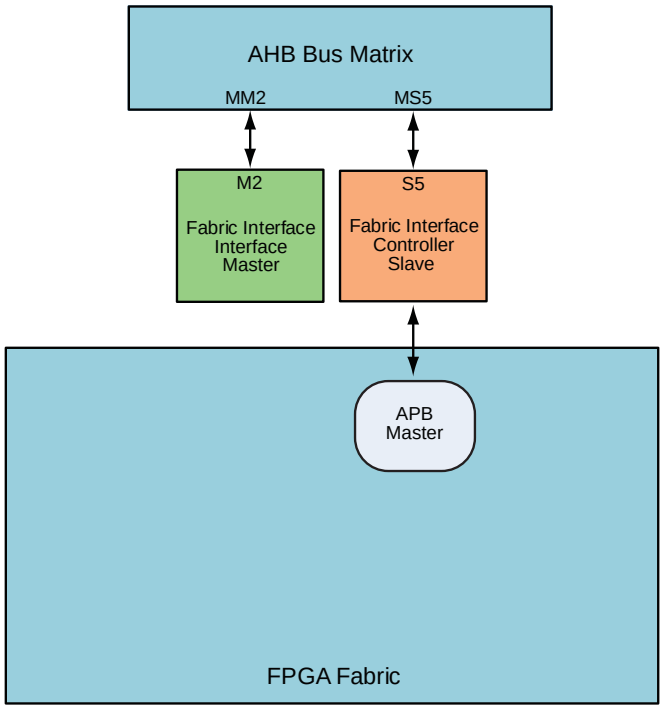


***Figure 19-8* • Fabric APB Master**

Locked Transactions

The fabric master must always perform a non-locked read of an MSS slave (whether internal slave or fabric slave) immediately after completing a locked sequence. Failure to do so will cause the debugger to hang.

16-Bit Fabric Master

Masters that can provide only a 16-bit address (such as Core8051s and CoreABC) need some means of creating the 32-bit address required to access the SmartFusion MSS peripherals. CoreAPB3 has an indirect addressing mode that can be used for this purpose. Refer to the "Memory Map (programmer's view)" section of the *CoreAPB3 Handbook* for a description of the indirect addressing mode with some examples.

Fabric Interface Interrupt Controller

There are 128 interrupt sources within the microcontroller subsystem (MSS). There are a finite number of signal resources that exist at the boundary between the MSS and the FPGA fabric. The fabric interface interrupt controller (FIIC) manages a subset of the total available MSS interrupts and maps those to a finite number of input ports along the FPGA fabric edge, as shown in Figure 19-9.

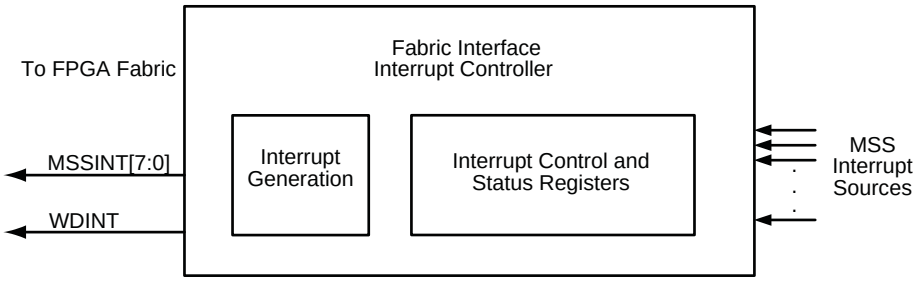


***Figure 19-9* • Fabric Interface Interrupt Controller Block Diagram**

FIIC Functional Description

The FIIC receives 128 interrupt sources from the MSS as inputs. These interrupt source inputs are level-sensitive active high inputs. These interrupt sources are combined, in a predetermined fashion, into the 8 MSSINT interrupts. There is also a pass through of the watchdog timer interrupt to WDINT. As shown in Figure 19-10, all interrupt processing is combinatorial; that is, the paths from the interrupt source inputs to the MSSINT[7:0] and WDINT outputs contain no flip-flops. Peripherals which drive the interrupt source inputs must ensure that their interrupts remain asserted until they are serviced. The FIIC performs no synchronization of source inputs nor does it synchronize the output interrupt signals to the fabric.

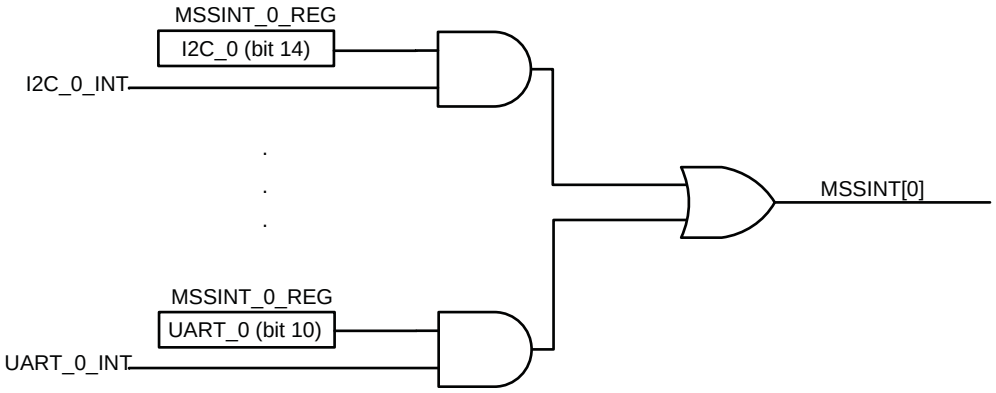


***Figure 19-10* • FICC Combinatorial Interrupt Processing**

In addition to having enabling/masking capability at each interrupt source peripheral, the FIIC also contains enable registers, to provide another level of masking, because some interrupts may need to be active in the MSS, but not in the FPGA fabric. All interrupt inputs to the FIIC are active high. Once asserted, they are guaranteed to be held asserted until cleared by firmware (via a write of a 1 to clear the peripheral). The exceptions to this are the SMBALERT and SMBUS interrupts from the I^2C peripherals. While these are held asserted, they are cleared by the sending I2C device, after a remote firmware-initiated sequence of operations (rather than clearing the interrupt in the local I^2C itself). WDOGTIMEOUTINT is always passed straight through the block as WDINT.

There are two modes of operation in the FIIC: ACE mode and non-ACE mode. In ACE mode, ACE interrupt sources along with some MSS interrupt sources are processed by the FIIC. In non-ACE mode, only MSS interrupts are processed by the FIIC. ACE mode interrupt mapping is depicted in Table 19-6 and non-ACE mode is depicted in Table 19-7. ACE mode is selected by setting the MODE bit in the FIIC_MR to a 0.

To assert the SOFTINTERRUPT (MSSINT[2]), the user must write a 1 to the SOFT_IRQ_CR in bit location 0 at address 0xE004202C. SOFTINTERRUPT stays asserted (1) as long as bit 0 of SOFT_IRQ_CR is a 1. Clearing bit 0 of SOFT_IRQ_CR deasserts the SOFTINTERRUPT (MSSINT[2]) signal.

***Table 19-6* • ACE Mode Interrupt Mapping**

Sources	Interrupt
WDOGWAKEUPINT, BROWNOUT1_5VINT, BROWNOUT3_3VINT, RTCMATCHEVENT, RTCIF_PUBINT, MAC_INT, IAP_INT, ENVM_[1:0]_INT, DMAINTERRUPT, UART_0_INT, UART_1_INT, SPI_0_INT, SPI_1_INT, I2C_0_INT, I2C_0_SMBALERT, I2C_0_SMBSUS, I2C_1_INT, I2C_1_SMBALERT, I2C_1_SMBSUS, TIMER1INT, TIMER2INT, PLLLOCKINT, PLLLOCKLOSTINT, SOFTINTERRUPT	MSSINT[0]
MSS_GPIO[31:0]	MSSINT[1]
ACE comparators	MSSINT[2]
ACE sample sequence engine events	MSSINT[3]
ACE post processing engine threshold events	MSSINT[4]
ACE ADC events	MSSINT[5]
ACE ADC FIFO Full / Almost Full events	MSSINT[6]
ACE ADC FIFO Not Empty events	MSSINT[7]

***Table 19-7* • Non-ACE Mode Interrupt Mapping**

Sources	Interrupt
WDOGWAKEUPINT, BROWNOUT1_5VINT, BROWNOUT3_3VINT, RTCMATCHEVENT, RTCIF_PUBINT, IAP_INT, ENVM_[1:0]_INT, DMAINTERRUPT, UART_0_INT, SPI_0_INT, I2C_0_INT, I2C_0_SMBALERT, I2C_0_SMBSUS, I2C_1_SMBALERT, I2C_1_SMBSUS, TIMER1INT, PLLLOCKINT, PLLLOCKLOSTINT	MSSINT[0]
MSS_GPIO[31:0]	MSSINT[1]
SOFTINTERRUPT	MSSINT[2]
TIMER2INT	MSSINT[3]
MAC_INT	MSSINT[4]
UART_1_INT	MSSINT[5]
I2C_1_INT	MSSINT[6]
SPI_1_INT	MSSINT[7]

FIICMSSIRQ_EN_0

***Table 19-8* • MSSIRQ_EN_0**

Bit Number	Name	R/W	Reset Value	Function
31:25	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	SOFTINT	R/W	0	1 = Enable; 0 = Mask
23	PLLLOCKLOST	R/W	0	1 = Enable; 0 = Mask
22	PLLLOCK	R/W	0	1 = Enable; 0 = Mask
21	TIMER2	R/W	0	1 = Enable; 0 = Mask
20	TIMER1	R/W	0	1 = Enable; 0 = Mask
19	I2C_1_SMBSUS	R/W	0	1 = Enable; 0 = Mask
18	I2C_1_SMBALERT	R/W	0	1 = Enable; 0 = Mask
17	I2C_1	R/W	0	1 = Enable; 0 = Mask
16	I2C_0_SMBSUS	R/W	0	1 = Enable; 0 = Mask
15	I2C_0_SMBALERT	R/W	0	1 = Enable; 0 = Mask
14	I2C_0	R/W	0	1 = Enable; 0 = Mask
13	SPI_1	R/W	0	1 = Enable; 0 = Mask
12	SPI_0	R/W	0	1 = Enable; 0 = Mask
11	UART_1	R/W	0	1 = Enable; 0 = Mask
10	UART_0	R/W	0	1 = Enable; 0 = Mask
9	DMA	R/W	0	1 = Enable; 0 = Mask
8	ENVM_1	R/W	0	1 = Enable; 0 = Mask
7	ENVM_0	R/W	0	1 = Enable; 0 = Mask
6	IAP	R/W	0	1 = Enable; 0 = Mask
5	MAC	R/W	0	1 = Enable; 0 = Mask
4	RTCIF_PUB	R/W	0	1 = Enable; 0 = Mask
3	RTCMATCHEVENT	R/W	0	1 = Enable; 0 = Mask
2	BROWNOUT3_3V	R/W	0	1 = Enable; 0 = Mask
1	BROWNOUT1_5V	R/W	0	1 = Enable; 0 = Mask
0	WDOGWAKEUP	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_1

***Table 19-9* • MSSIRQ_EN_1**

Bit Number	Name	R/W	Reset Value	Function
31:0	MSS_GPIO	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_2

***Table 19-10* • MSSIRQ_EN_2**

Bit Number	Name	R/W	Reset Value	Function
31:24	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	CMP_11_R	R/W	0	1 = Enable; 0 = Mask
22	CMP_10_R	R/W	0	1 = Enable; 0 = Mask
21	CMP_9_R	R/W	0	1 = Enable; 0 = Mask
20	CMP_8_R	R/W	0	1 = Enable; 0 = Mask
19	CMP_7_R	R/W	0	1 = Enable; 0 = Mask
18	CMP_6_R	R/W	0	1 = Enable; 0 = Mask
17	CMP_5_R	R/W	0	1 = Enable; 0 = Mask
16	CMP_4_R	R/W	0	1 = Enable; 0 = Mask
15	CMP_3_R	R/W	0	1 = Enable; 0 = Mask
14	CMP_2_R	R/W	0	1 = Enable; 0 = Mask
13	CMP_1_R	R/W	0	1 = Enable; 0 = Mask
12	CMP_0_R	R/W	0	1 = Enable; 0 = Mask
11	CMP_11_F	R/W	0	1 = Enable; 0 = Mask
10	CMP_10_F	R/W	0	1 = Enable; 0 = Mask
9	CMP_9_F	R/W	0	1 = Enable; 0 = Mask
8	CMP_8_F	R/W	0	1 = Enable; 0 = Mask
7	CMP_7_F	R/W	0	1 = Enable; 0 = Mask
6	CMP_6_F	R/W	0	1 = Enable; 0 = Mask
5	CMP_5_F	R/W	0	1 = Enable; 0 = Mask
4	CMP_4_F	R/W	0	1 = Enable; 0 = Mask
3	CMP_3_F	R/W	0	1 = Enable; 0 = Mask
2	CMP_2_F	R/W	0	1 = Enable; 0 = Mask
1	CMP_1_F	R/W	0	1 = Enable; 0 = Mask
0	CMP_0_F	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_3

***Table 19-11* • MSSIRQ_EN_3**

Bit Number	Name	R/W	Reset Value	Function
31:12	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	PC2_FLAG_3	R/W	0	1 = Enable; 0 = Mask
10	PC2_FLAG_2	R/W	0	1 = Enable; 0 = Mask
9	PC2_FLAG_1	R/W	0	1 = Enable; 0 = Mask
8	PC2_FLAG_0	R/W	0	1 = Enable; 0 = Mask
7	PC1_FLAG_3	R/W	0	1 = Enable; 0 = Mask
6	PC1_FLAG_2	R/W	0	1 = Enable; 0 = Mask
5	PC1_FLAG_1	R/W	0	1 = Enable; 0 = Mask
4	PC1_FLAG_0	R/W	0	1 = Enable; 0 = Mask
3	PC0_FLAG_3	R/W	0	1 = Enable; 0 = Mask
2	PC0_FLAG_2	R/W	0	1 = Enable; 0 = Mask
1	PC0_FLAG_1	R/W	0	1 = Enable; 0 = Mask
0	PC0_FLAG_0	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_4

***Table 19-12* • MSSIRQ_EN_4**

Bit Number	Name	R/W	Reset Value	Function
31:0	PPE_THRESH	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_5

***Table 19-13* • MSSIRQ_EN_5**

Bit Number	Name	R/W	Reset Value	Function
31:9	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	ADC_2_CAL_R	R/W	0	1 = Enable; 0 = Mask
7	ADC_1_CAL_R	R/W	0	1 = Enable; 0 = Mask
6	ADC_0_CAL_R	R/W	0	1 = Enable; 0 = Mask
5	ADC_2_CAL_F	R/W	0	1 = Enable; 0 = Mask
4	ADC_1_CAL_F	R/W	0	1 = Enable; 0 = Mask
3	ADC_0_CAL_F	R/W	0	1 = Enable; 0 = Mask
2	ADC_2_DV_R	R/W	0	1 = Enable; 0 = Mask
1	ADC_1_DV_R	R/W	0	1 = Enable; 0 = Mask
0	ADC_0_DV_R	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_6

***Table 19-14* • MSSIRQ_EN_6**

Bit Number	Name	R/W	Reset Value	Function
31:6	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	ADC2_AFULL	R/W	0	1 = Enable; 0 = Mask
4	ADC2_FULL	R/W	0	1 = Enable; 0 = Mask
3	ADC1_AFULL	R/W	0	1 = Enable; 0 = Mask
2	ADC1_FULL	R/W	0	1 = Enable; 0 = Mask
1	ADC0_AFULL	R/W	0	1 = Enable; 0 = Mask
0	ADC0_FULL	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_EN_7

***Table 19-15* • MSSIRQ_EN_7**

Bit Number	Name	R/W	Reset Value	Function
31:3	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	ADC2_NOTEMPTY	R/W	0	1 = Enable; 0 = Mask
1	ADC1_NOTEMPTY	R/W	0	1 = Enable; 0 = Mask
0	ADC0_NOTEMPTY	R/W	0	1 = Enable; 0 = Mask

FIIC MSSIRQ_SRC_0

***Table 19-16* • MSSIRQ_SRC_0**

Bit Number	Name	R/W	Reset Value	Function
31:25	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	SOFTINT	R	0	1 = Interrupt asserted and enabled
23	PLLLOCKLOST	R	0	1 = Interrupt asserted and enabled
22	PLLLOCK	R	0	1 = Interrupt asserted and enabled
21	TIMER2	R	0	1 = Interrupt asserted and enabled
20	TIMER1	R	0	1 = Interrupt asserted and enabled
19	I2C_1_SMBSUS	R	0	1 = Interrupt asserted and enabled
18	I2C_1_SMBALERT	R	0	1 = Interrupt asserted and enabled
17	I2C_1	R	0	1 = Interrupt asserted and enabled
16	I2C_0_SMBSUS	R	0	1 = Interrupt asserted and enabled
15	I2C_0_SMBALERT	R	0	1 = Interrupt asserted and enabled
14	I2C_0	R	0	1 = Interrupt asserted and enabled
13	SPI_1	R	0	1 = Interrupt asserted and enabled
12	SPI_0	R	0	1 = Interrupt asserted and enabled
11	UART_1	R	0	1 = Interrupt asserted and enabled
10	UART_0	R	0	1 = Interrupt asserted and enabled
9	DMA	R	0	1 = Interrupt asserted and enabled
8	ENVM_1	R	0	1 = Interrupt asserted and enabled
7	ENVM_0	R	0	1 = Interrupt asserted and enabled
6	IAP	R	0	1 = Interrupt asserted and enabled
5	MAC	R	0	1 = Interrupt asserted and enabled
4	RTCIF_PUB	R	0	1 = Interrupt asserted and enabled
3	RTCMATCHEVENT	R	0	1 = Interrupt asserted and enabled
2	BROWNOUT3_3V	R	0	1 = Interrupt asserted and enabled
1	BROWNOUT1_5V	R	0	1 = Interrupt asserted and enabled
0	WDOGWAKEUP	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_1

***Table 19-17* • MSSIRQ_SRC_1**

Bit Number	Name	R/W	Reset Value	Function
31:0	MSS_GPIO	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_2

Table 19-18 • **MSSIRQ_SRC_2**

Bit Number	Name	R/W	Reset Value	Function
31:24	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	CMP_11_R	R	0	1 = Interrupt asserted and enabled
22	CMP_10_R	R	0	1 = Interrupt asserted and enabled
21	CMP_9_R	R	0	1 = Interrupt asserted and enabled
20	CMP_8_R	R	0	1 = Interrupt asserted and enabled
19	CMP_7_R	R	0	1 = Interrupt asserted and enabled
18	CMP_6_R	R	0	1 = Interrupt asserted and enabled
17	CMP_5_R	R	0	1 = Interrupt asserted and enabled
16	CMP_4_R	R	0	1 = Interrupt asserted and enabled
15	CMP_3_R	R	0	1 = Interrupt asserted and enabled
14	CMP_2_R	R	0	1 = Interrupt asserted and enabled
13	CMP_1_R	R	0	1 = Interrupt asserted and enabled
12	CMP_0_R	R	0	1 = Interrupt asserted and enabled
11	CMP_11_F	R	0	1 = Interrupt asserted and enabled
10	CMP_10_F	R	0	1 = Interrupt asserted and enabled
9	CMP_9_F	R	0	1 = Interrupt asserted and enabled
8	CMP_8_F	R	0	1 = Interrupt asserted and enabled
7	CMP_7_F	R	0	1 = Interrupt asserted and enabled
6	CMP_6_F	R	0	1 = Interrupt asserted and enabled
5	CMP_5_F	R	0	1 = Interrupt asserted and enabled
4	CMP_4_F	R	0	1 = Interrupt asserted and enabled
3	CMP_3_F	R	0	1 = Interrupt asserted and enabled
2	CMP_2_F	R	0	1 = Interrupt asserted and enabled
1	CMP_1_F	R	0	1 = Interrupt asserted and enabled
0	CMP_0_F	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_3

***Table 19-19* • MSSIRQ_SRC_3**

Bit Number	Name	R/W	Reset Value	Function
31:12	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	PC2_FLAG_3	R	0	1 = Interrupt asserted and enabled
10	PC2_FLAG_2	R	0	1 = Interrupt asserted and enabled
9	PC2_FLAG_1	R	0	1 = Interrupt asserted and enabled
8	PC2_FLAG_0	R	0	1 = Interrupt asserted and enabled
7	PC1_FLAG_3	R	0	1 = Interrupt asserted and enabled
6	PC1_FLAG_2	R	0	1 = Interrupt asserted and enabled
5	PC1_FLAG_1	R	0	1 = Interrupt asserted and enabled
4	PC1_FLAG_0	R	0	1 = Interrupt asserted and enabled
3	PC0_FLAG_3	R	0	1 = Interrupt asserted and enabled
2	PC0_FLAG_2	R	0	1 = Interrupt asserted and enabled
1	PC0_FLAG_1	R	0	1 = Interrupt asserted and enabled
0	PC0_FLAG_0	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_4

***Table 19-20* • MSSIRQ_SRC_4**

Bit Number	Name	R/W	Reset Value	Function
31:0	PPE_THRESH	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_5

***Table 19-21* • MSSIRQ_SRC_5**

Bit Number	Name	R/W	Reset Value	Function
31:9	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	ADC_2_CAL_R	R	0	1 = Interrupt asserted and enabled
8	ADC_1_CAL_R	R	0	1 = Interrupt asserted and enabled
7	ADC_0_CAL_R	R	0	1 = Interrupt asserted and enabled
6	ADC_2_CAL_F	R	0	1 = Interrupt asserted and enabled
5	ADC_1_CAL_F	R	0	1 = Interrupt asserted and enabled
4	ADC_0_CAL_F	R	0	1 = Interrupt asserted and enabled
3	ADC_2_DV_R	R	0	1 = Interrupt asserted and enabled
2	ADC_1_DV_R	R	0	1 = Interrupt asserted and enabled
1	ADC_0_DV_R	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_6

***Table 19-22* • MSSIRQ_SRC_6**

Bit Number	Name	R/W	Reset Value	Function
31:6	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	ADC2_AFULL	R	0	1 = Interrupt asserted and enabled
4	ADC2_FULL	R	0	1 = Interrupt asserted and enabled
3	ADC1_AFULL	R	0	1 = Interrupt asserted and enabled
2	ADC1_FULL	R	0	1 = Interrupt asserted and enabled
1	ADC0_AFULL	R	0	1 = Interrupt asserted and enabled
0	ADC0_FULL	R	0	1 = Interrupt asserted and enabled

FIIC MSSIRQ_SRC_7

***Table 19-23* • MSSIRQ_SRC_7**

Bit Number	Name	R/W	Reset Value	Function
31:3	Reserved	R	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	ADC2_NOTEMPTY	R	0	1 = Interrupt asserted and enabled
1	ADC1_NOTEMPTY	R	0	1 = Interrupt asserted and enabled
0	ADC0_NOTEMPTY	R	0	1 = Interrupt asserted and enabled

FIIC_MR

***Table 19-24* • FIIC_MR**

Bit Number	Name	R/W	Reset Value	Function
31:1	Reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MODE	R/W	0	0 = ACE mode 1 = Non-ACE mode

IOMUX Functional Description

The MSS contains I/O multiplexers that are involved in the reuse of some MSS-related I/O pads and in providing a number of options for multiplexing GPIO, peripheral signals, and fabric interface signals to the I/O pad. IOMUXes are associated with the GPIO block, fabric interface, and all MSS communications peripherals (UARTS 0 and 1, SPI 0 and 1, I2C 0 and 1, and the Ethernet MAC). It is important to note that all available hard peripherals in the MSS are available to MSS I/O pads; the peripherals are not multiplexed with other peripherals at the I/O pad. Table 19-28 on page 369 lists the IOMUXes associated with each peripheral. Refer to each peripheral's section of the user's guide to determine how those individual IOMUXes are associated to the peripheral.

For every reusable MSS I/O pad there is an IOMUX. The IOMUX is intended to provide flexibility in the allocation of MSS I/O pads. If the user is not using a particular interface, the corresponding I/O pads can be reallocated to another interface. Also, if certain I/O pads are not being used (or are not present in some devices) then the IOMUX allows the signals to be connected within the IOMUX internally, as shown in Figure 19-12 on page 367. The IOMUX is composed of 4 multiplexers: M0, M1, M2, and M3, as shown in Figure 19-11.

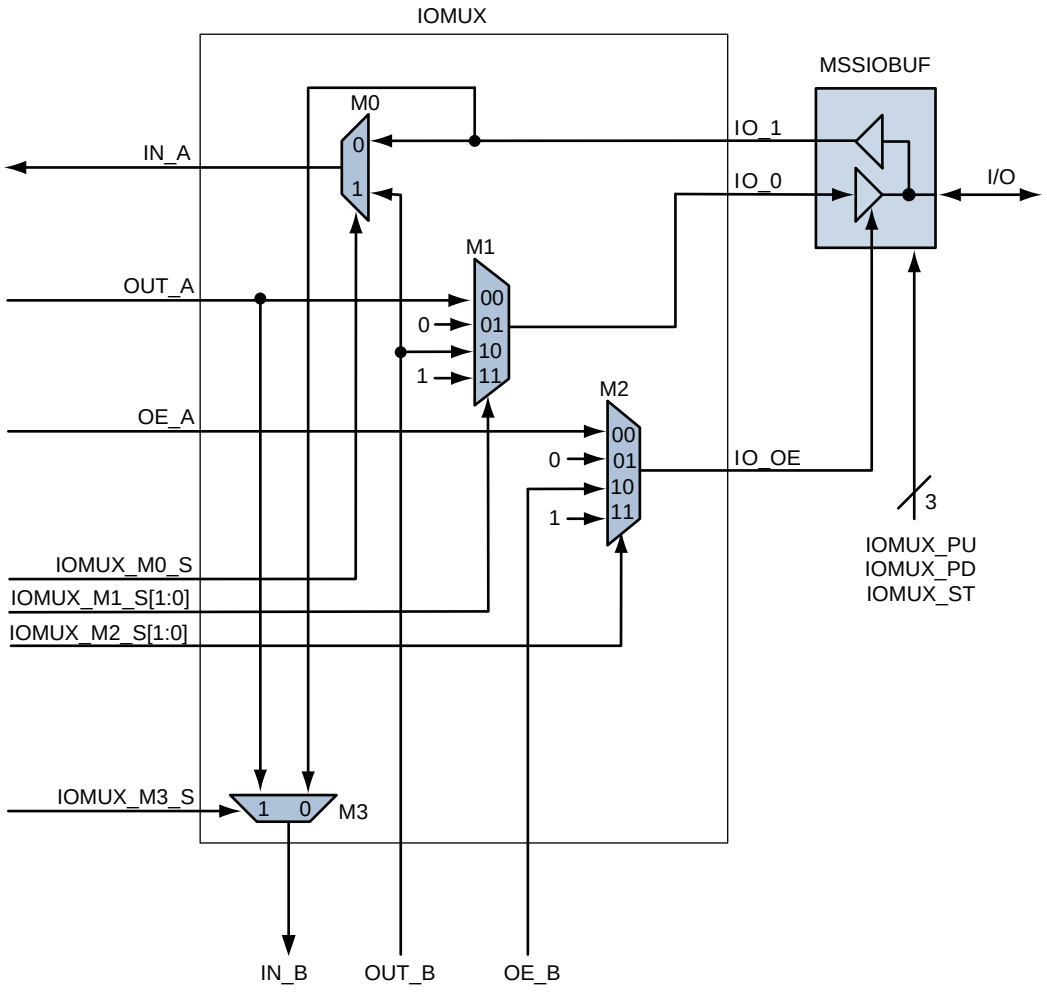


***Figure 19-11* • IOMUX Block Diagram**

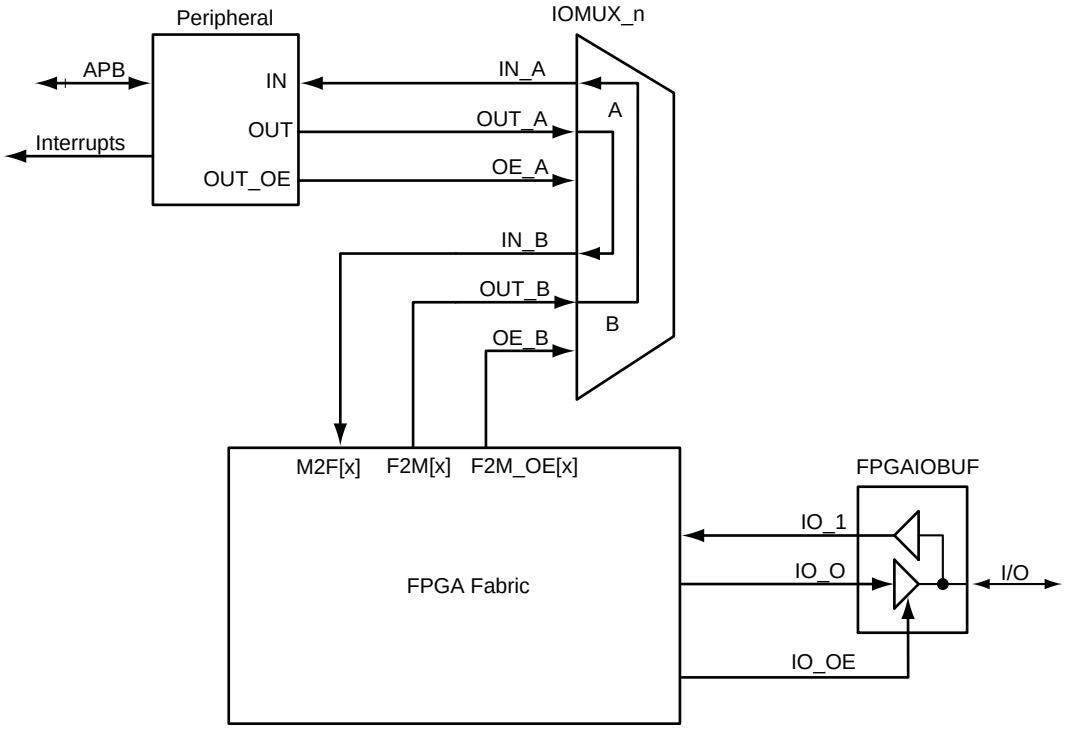


***Figure 19-12* • Example of IOMUX Signals Routed Internal to the IOMUX**

IOMUX Register Map

The IOMUX_n_CR (where n can range from 0 to 82) registers are located within the SYSREG block at address 0xE0042100 and continuing for 83 32-bit register locations or 332 bytes. There are 83 IOMUX_n_CR control registers; one for each IOMUX.

Figure 19-25 on page 368 lists the bit definitions of these control registers.

Table 19-25 • **IOMUX _n_CR**

Bit Number	Name	R/W	Reset Value	Function
9	IOMUX_n_ST	R/W	0	0 = Schmitt Trigger of IOMUX n is disabled. 1 = Enabled
8	IOMUX_n_PD	R/W	0	0 = Weak pull-down of IOMUX n is disabled. 1 = Enabled
7	IOMUX_n_PU	R/W	0	0 = Weak pull-up of IOMUX n is disabled. 1 = Enabled
6	Reserved	R/W	0	Reserved
5	IOMUX_n_M3_S	R/W	0	MUX M3 select 0 = IN_B = IO_I 1 = IN_B = OUT_A
4:3	IOMUX_n_M2_S[1:0]	R/W	0	See Table 19-26.
2:1	IOMUX_n_M1_S[1:0]	R/W	0	See Table 19-27.
0	IOMUX_n_M0_S	R/W	0	MUX M0 select 0 = IN_A = IO_I. 1 = IN_A = OUT_B.

Table 19-26 • **IOMUX MUX M2 Configuration**

IOMUX_n_M2_S[1]	IOMUX_n_M2_S[0]	Function
0	0	OE_A drives IO_OE
0	1	IO_OE driven to 0
1	0	OE_B drives IO_OE
1	1	IO_OE driven to 1

Table 19-27 • **IOMUX MUX M1 Configuration**

IOMUX_n_M1_S[1]	IOMUX_n_M1_S[0]	Function
0	0	OUT_A drives IO_O
0	1	IO_O driven to 0
1	0	OUT_B drives IO_O
1	1	IO_O driven to 1

***Table 19-28* • IOMUX to Peripheral Association**

Peripheral	Associated IOMUXes
SPI_0	0, 1, 2, 3
UART_0	4, 5, 64-69
I2C_0	6, 7
SPI_1	8, 9, 10, 11, 70-76
UART_1	12, 13, 77-82
I2C_1	14, 15
EMAC	16-24
GPIO_0	25
.	.
.	.
.	.
GPIO_15	40
GPIO_16	0, 41
.	.
.	.
.	.
GPIO_31	15, 56

ACE Thresholds

There are 32 threshold signals (ACEFLAGS[31:0]) output directly from the ACE post processing engine and one signal (FABACETRIG) sourced from the FPGA fabric that can be used to trigger the start of the sample sequence engine (SSE) in the ACE. These are described in more detail in the *SmartFusion Programmable Analog User's Guide*.

SCB Signals

Signals coming from and going to the signal conditioning block (SCB) are described in Table 19-29. These signals and functionality are available for instantiation by the Libero SoC design tool.

***Table 19-29* • 32 SCB to FPGA Fabric Interface Signals**

Name	Input To / Output From FPGA Fabric	Function
LVTTL0	Input	Direct ADC input used as LVTTL 0 input
LVTTL1	Input	Direct ADC input used as LVTTL 1 input
LVTTL2	Input	Direct ADC input used as LVTTL 2 input
LVTTL3	Input	Direct ADC input used as LVTTL 3 input
LVTTL4	Input	Direct ADC input used as LVTTL 4 input
LVTTL5	Input	Direct ADC input used as LVTTL 5 input
LVTTL6	Input	Direct ADC input used as LVTTL 6 input
LVTTL7	Input	Direct ADC input used as LVTTL 7 input
LVTTL0EN	Output	Enable for LVTTL0 input
LVTTL1EN	Output	Enable for LVTTL1 input
LVTTL2EN	Output	Enable for LVTTL2 input
LVTTL3EN	Output	Enable for LVTTL3 input
LVTTL4EN	Output	Enable for LVTTL4 input
LVTTL5EN	Output	Enable for LVTTL5 input
LVTTL6EN	Output	Enable for LVTTL6 input
LVTTL7EN	Output	Enable for LVTTL7 input
CMP0	Input	Comparator 0 output
CMP1	Input	Comparator 1 output
CMP2	Input	Comparator 2 output
CMP3	Input	Comparator 3 output
CMP4	Input	Comparator 4 output
CMP5	Input	Comparator 5 output
CMP6	Input	Comparator 6 output
CMP7	Input	Comparator 7 output

DAC Signals

As shown in Table 19-30, for each DAC there is a single-bit data input and its associated clock. Users can provide a custom input waveform to the first order SDD based on user logic. The A2F200 device has two DACs and the A2F500 device has three DACs. The DAC inputs are also capable of being driven from within the ACE as well. Refer to the *SmartFusion Programmable Analog User's Guide* for more details.

***Table 19-30* • DAC Interface Signals**

Name	Input To/ Output From FPGA Fabric	Function
FABSDD0D	Output	Data out from FPGA fabric driving the input to the SDD_0
FABSDD0CLK	Output	Clock for FABSDD0D
FABSDD1D	Output	Data out from FPGA fabric driving the input to the SDD_1
FABSDD1CLK	Output	Clock for FABSDD1D
FABSDD2D	Output	Data out from FPGA fabric driving the input to the SDD_2
FABSDD2CLK	Output	Clock for FABSDD2D

VR/PSM Signals

Table 19-31 lists the interface signals between the VR/PSM and the FPGA fabric. The functionality of these signals is described in the "Voltage Regulator (VR), Power Supply Monitor (PSM), and Power Modes" section on page 151.

***Table 19-31* • Voltage Regulator / Power Supply Monitor Signals**

Name	Input To/ Output From FPGA Fabric	Function
PUFAB_N	Input	Push-button, inverted version of PU_N pin
FPGAVRON	Output	FPGA signal to turn on and off the VREG. This bit is qualified by the FPGAVRONENABLE field in the VRPSM_CR, which is controlled by the Cortex-M3 processor. This prevents false triggering of FPGAVRON when the FPGA fabric is unprogrammed.

Miscellaneous Signals

Table 19-32 lists miscellaneous control and status signals connecting the MSS and the FPGA fabric. These signals are available for instantiation by the Libero SoC design tool.

Table 19-32 • **Miscellaneous Interface Signals**

Name	Input To/ Output From FPGA Fabric	Function
TXEV	Input	Event transmitted as a result of Cortex-M3 SEV (send event) instruction. This is a single-cycle pulse equal to 1 FCLK period.
RXEV	Output	Causes the Cortex-M3 processor to wake up from a WFE instruction. The event input, RXEV, is registered even when not waiting for an event, and so affects the next WFE.
SLEEPING	Input	This signal is asserted when the Cortex-M3 processor is in sleep-now or sleep-on-exit mode, and indicates that the clock to the processor can be stopped.
DEEPSLEEP	Input	This signal is asserted when the Cortex-M3 processor is in sleep-now or sleep-on-exit mode when the SLEEPDEEP bit of the System Control Register is set.
M2F_RESET_N	Input	MSS reset signal driven into the FPGA fabric. Controlled by the reset manager.
F2M_RESET_N	Output	Fabric reset signal driven into the MSS reset manager. Controlled by the user.
FABINT	Output	Interrupt signal sourced by user logic to the NVIC on the Cortex-M3 processor, which is INTISR[31]. This interrupt goes straight into the Cortex-M3 NVIC block without any synchronization; thus its timing must be synchronous to the clock being used in the fabric for the FIC32 AHB/APB interface, which in turn is aligned, but possibly at a lower frequency, to the processor clock, FCLK. The NVIC supports configuring interrupts as either level sensitive or edge sensitive. If level sensitive, then FABINT must be held asserted until cleared by some means via the ISR. If configured as edge sensitive, then FABINT may be just a pulse, as short as one FCLK period. This is all that the NVIC needs to detect the interrupt. It may be reasserted during the ISR and will be interpreted as a new interrupt.
DMAREADY0	Output	Indicates that a soft IP is ready to be serviced. This signal is logic active High. The minimum pulse width of this signal generated by soft IP is one clock cycle of FAB_CLK. To make sure the PDMA services soft IP running with the slowest possible FAB_CLK (i.e., FCLK/4), the user must set the Write_Adj parameter in the PDMA driver to a value of greater than 4.
DMAREADY1	Output	Indicates that a soft IP is ready to be serviced. This signal is logic active High. The minimum pulse width of this signal generated by soft IP is one clock cycle of FAB_CLK. To make sure the PDMA services soft IP running with the slowest possible FAB_CLK (i.e., FCLK/4), the user must set the Write_Adj parameter in the PDMA driver to a value of greater than 4.

***Table 19-32* • Miscellaneous Interface Signals (continued) (continued)**

Name	Input To/ Output From FPGA Fabric	Function
I2C0SMBSUSNO	Input	Output Suspend Mode signal. This signal is used if I2C is the master/host. *Note: Not a Wired-AND signal.*
I2C0SMBALERTNO	Input	Output Wired-AND interrupt signal. This signal is used in slave/device mode if the I2C wants to force communication with a host.
I2C0SMBSUSNI	Output	Input Suspend Mode signal. This signal is used if I2C is slave/device. *Note: Not a Wired-AND signal.*
I2C0SMBALERTNI	Output	Input Wired-AND interrupt signal. This signal is used in master/host mode to monitor if slave/devices want to force communication with the host.
I2C0BCLK	Output	Alternate clock for I2C_0.
I2C1SMBSUSNO	Input	Output Suspend Mode signal. This signal is used if I2C is the master/host. *Note: Not a Wired-AND signal.*
I2C1SMBALERTNO	Input	Output Wired-AND interrupt signal. This signal is used in slave/device mode if the I2C wants to force communication with a host.
I2C1SMBSUSNI	Output	Input Suspend Mode signal. This signal is used if I2C is slave/device. NOTE: Not a Wired-AND signal.
I2C1SMBALERTNI	Output	Input Wired-AND interrupt signal. This signal is used in master/host mode to monitor if slave/devices want to force communication with the host.
I2C1BCLK	Output	Alternate clock for I2C_1.

20 – SmartFusion Master Register Map

Table 20-1 lists all registers in the SYSREG space.

***Table 20-1* • Registers in the SYSREG Space**

Register Name	Address	R/W	Width	Description
ESRAM_CR	0xE0042000	R/W	1	Controls address mapping of the eSRAMs
ENVM_CR	0xE0042004	R/W	7	Configures eNVM parameters
ENVM_REMAP_SYS_CR	0xE0042008	R/W	20	Configures where eNVM is mapped in system space
ENVM_REMAP_FAB_CR	0xE004200C	R/W	20	Configures where eNVM is mapped in fabric master space
FAB_PROT_SIZE_CR	0xE0042010	R/W	5	Defines the size of the memory that is inaccessible by a fabric Master
FAB_PROT_BASE_CR	0xE0042014	R/W	32	The absolute memory address of the memory region protected from Fabric Access
AHB_MATRIX_CR	0xE0042018	R/W	4	AHB Bus matrix Control Register
MSS_SR	0xE004201C	R	11	MSS Status registers
CLR_MSS_SR	0xE0042020	W	11	Clear the DSS status bits
EFROM_CR	0xE0042024	R/W	4	Configures FROM bus timing
IAP_CR	0xE0042028	R/W	3	Configures IAP bus timing
SOFT_IRQ_CR	0xE004202C	R/W	1	SOFTINTERRUPT in FIIC Control (MSSINT2 mapping)
SOFT_RST_CR	0xE0042030	R/W	20	Generates software control interrupts to the DSS peripherals
DEVICE_SR	0xE0042034	R	7	Various Device Status bits
SYSTICK_CR	0xE0042038	R/W	29	Configures SysTick Timer STCALIB inputs
EMC_MUX_CR	0xE004203C	R/W	1	External Memory Controller MUX Configuration
EMC_CS_x_CR (x = 0)	0xE0042040	R/W	22	EMC Timing Parameters for Chip Select 0
EMC_CS_x_CR (x = 1)	0xE0042044	R/W	22	EMC Timing Parameters for Chip Select 1
MSS_CLK_CR	0xE0042048	R/W	13	Clock Configuration for MSS Clock Dividers
MSS_CCC_DIV_CR	0xE004204C	R/W	32	Control bits for the MSS_CCC Dividers
MSS_CCC_MUX_CR	0xE0042050	R/W	32	Control bits for the MSS_CCC Multiplexors
MSS_CCC_PLL_CR	0xE0042054	R/W	32	Control bits for the MSS_CCC PLL
MSS_CCC_DLY_CR	0xE0042058	R/W	32	Control bits for the MSS_CCC Delay Elements
MSS_CCC_SR	0xE004205C	R	1	Status of MSS_CCC
Reserved Register	0xE0042060			Do not write to this register.
VRPSM_CR	0xE0042064	R/W	5	Controls Band Gap Enable, Fabric Vreg qualifier, clears PU_N interrupts
Reserved	0xE0042068	–	–	Reserved
FAB_IF_CR	0xE004206C	R/W	7	Fabric Interface Control register

***Table 20-1* • Registers in the SYSREG Space (continued)**

Register Name	Address	R/W	Width	Description
FAB_APB_HIWORD_DR	0xE0042070	R	16	Configures fabric interface as either APB or AHB.
LOOPBACK_CR	0xE0042074	R/W	5	Loopback control for MSS peripherals
MSS_IO_BANK_CR	0xE0042078	R/W	4	Set I/O standard for MSS I/O Banks
GPIN_SOURCE_CR	0xE004207C	R/W	16	Alternate GPIN source select
Reserved	0xE0042080	–	–	Reserved
⋮		–	–	⋮
Reserved	0xE00420FC	–	–	Reserved
IOMUX _n_CR (n = 0)	0xE0042100	R/W	10	I/O MUX Cell 0 control register
⋮	⋮	⋮	–	⋮
IOMUX _n_CR (n = 82)	0xE0042248	R/W	10	I/O MUX Cell 82 control register

Table 20-2 provides a listing of all registers referred to in the *SmartFusion Microcontroller Subsystem User's Guide*, including cross-references to the sections where each register is defined in detail.

***Table 20-2* • SmartFusion Master Register Map**

Register Name	Address	R/W	Reset Value	Description
"Cortex-M3 SysTick Timer"				
SYSTICK_CR	0xE0042038	R/W	0x32000000	Provides firmware control of the STCALIB[25:0] pins of Cortex-M3 microcontroller.
SysTick Reload Value	0xE000E014	R/W	Unpredictable	Value to load in Current Value register when 0 is reached
SysTick Current Value	0xE000E018	R/W	Unpredictable	The current value of the count down
SysTick Calibration Value	0xE000E01C	R	STCALIB	Contains the number of ticks to generate a 10 ms interval.
SysTick Control and Status	0xE000E010	R/W	0x0	Basic control of SysTick, including enable, clock source, interrupt, or poll
			Interrupts	The interrupt numbers (corresponding to the NVIC input pins of the Cortex-M3 processor), their sources, and which functions assert the interrupt for the SmartFusion family of custommizable system-on-chip devices.

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"PDMA Register Map"				
RATIO_HIGH_LOW	0x40004000	R/W	0	Ratio of high priority transfers versus low priority transfers
BUFFER_STATUS (x = 0	0x40004004	R/W	0	Indicates when buffers have drained
CHANNEL_x_CONTROL (x = 0)	0x40004020	R/W	0	Channel 0 control register
CHANNEL_x_STATUS (x = 0)	0x40004024	R	0	Channel 0 status register
CHANNEL_x_BUFFER_A_SRC_ADDR (x = 0)	0x40004028	R/W	0	Channel 0 buffer A source address
CHANNEL_x_BUFFER_A_DST_ADDR (x = 0)	0x4000402C	R/W	0	Channel 0 buffer A destination address
CHANNEL_x_BUFFER_A_TRANSFER_COUNT (x = 0)	0x40004030	R/W	0	Channel 0 buffer A transfer count
CHANNEL_x_BUFFER_B_SRC_ADDR (x = 0)	0x40004034	R/W	0	Channel 0 buffer B source address
CHANNEL_x_BUFFER_B_DST_ADDR (x = 0)	0x40004038	R/W	0	Channel 0 buffer B destination address
CHANNEL_x_BUFFER_B_TRANSFER_COUNT (x = 0)	0x4000403C	R/W	0	Channel 0 buffer B transfer count
CHANNEL_1_CONTROL	0x40004040	R/W	0	Channel 1 control register
⋮	⋮	R/W	⋮	⋮
CHANNEL_1_BUFFER_B_TRANSFER_COUNT	0x4000405C	R/W	0	Channel 1 buffer B transfer count
⋮	⋮	R/W	⋮	⋮
CHANNEL_7_BUFFER_B_TRANSFER_COUNT	0x4000411C	R/W	0	Channel 7 buffer B transfer count
"eNVM Controller Register Map"				
ENVM_STATUS_REG	0x60100000	R/W	0x0	Returns the status of the last commanded operation.
ENVM_CONTROL_REG	0x60100004	R/W	0x0	Control register used for all eNVM commands
ENVM_ENABLE_REG	0x60100008	R/W	0x0	eNVM interrupt enable register
Reserved	0x6010000C	R/W	0x0	Reserved
ENVM_0_CR	0x6010000C	R/W	0x0	eNVM 0 configuration register
ENVM_1_CR	0x60100010	R/W	0x0	eNVM 1 configuration register
ENVM_PAGE_STATUS_0_REG	0x60100014	R	0x0	eNVM 0 page status register
ENVM_PAGE_STATUS_1_REG	0x60100018	R	0x0	eNVM 1 page status register
"EFROM_CR Register Map" on page 77				
EFROM_CR	0xE0042024	R/W	0x00000009	Used to set eFROM APB interface controller timing options

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"External Memory Controller Register Map"				
EMC_MUX_CR	0xE004203C	R/W	0x0	External memory controller MUX configuration
EMC_CS_x_CR (x = 0)	0xE0042040	R/W	0x0	EMC timing parameters for chip select 0
EMC_CS_x_CR (x = 1)	0xE0042044	R/W	0x0	EMC timing parameters for chip select 1
"Watchdog Register Interface Summary"				
WDOGVALUE	0x40006000	R	0x20000000	Current value of counter
WDOGLOAD	0x40006004	R/W	0x20000000	Load value for counter
WDOGMVRP	0x40006008	R/W	0xFFFFFFFF	Maximum value for which refreshing is permitted
WDOGREFRESH	0x4000600C	W	N/A	Writing the value 0xAC15DE42 to this register causes the counter to be updated with the value in WDOGLOAD register.
WDOGENABLE	0x40006010	R/W	0x1	Watchdog enable register
WDOGCONTROL	0x40006014	R/W	0x0	Control register
WDOGSTATUS	0x40006018	R	0x1	Status register
WDOGRIS	0x4000601C	R/W	0x0	Raw interrupt status
WDOGMIS	0x40006020	R	0x0	Masked interrupt status
MSS_SR	0xE004201C	R	0x0	MSS Status register
"Ethernet MAC Control and Status Register Addressing"				
CSR0	0x40003000	R/W	0xFE000000	Bus mode
CSR1	0x40003008	W	0	Transmit poll demand
CSR2	0x40003010	W	0	Receive poll demand
CSR3	0x40003018	R/W	0xFFFFFFFF	Receive list base address
CSR4	0x40003020	R/W	0xFFFFFFFF	Transmit list base address
CSR5	0x40003028	R/W	0xF0000000	Status and control
CSR6	0x40003030	R/W	0x32000040	Operation mode
CSR7	0x40003038	R/W	0xF3FE0000	Interrupt enable
CSR8	0x40003040	R/W	0xE0000000	Missed frames and overflow counters
CSR9	0x40003048	R/W	0xFFF483FB	RMII management
CSR10	0x40003050	N/A	0	Reserved
CSR11	0x40003058	R/W	0xFFFE0000	Timer and interrupt mitigation control

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"SPI Register Interface Summary" (addresses for SPI_0 shown; SPI_1 begins at 0x40011x00)				
CONTROL	0x40001000	R/W	0x0000102	Control register
TXRXDF_SIZE	0x40001004	R/W	0x04	Transmit and receive data frame size
STATUS	0x40001008	R	0x2440	Status register
INT_CLEAR	0x4000100C	W	0x00	Interrupt Clear register
RX_DATA	0x40001010	R	Unknown	Receive Data register
TX_DATA	0x40001014	W	0x00000000	Transmit Data register
CLK_GEN	0x40001018	R/W	0x07	Output Clock Generator (master mode)
SLAVE_SELECT	0x4000101C	R/W	0x00	Specifies slave selected (master mode)
MIS	0x40001020	R	0x00	Masked interrupt status
RIS	0x40001024	R	0x00	Raw interrupt status
The following registers apply only to A2F060 and A2F500.				
CONTROL2	0x40001028	R/W	0x00	Control bits for enhanced mode
COMMAND	0x4000102C	R/W	0x00	Command register
PKTSIZE	0x40001030	R/W	0x00	Packet size
Reserved	0x40001034	R/W	0x00	Reserved
Reserved	0x40001038	R/W	0x00	Reserved
STAT8	0x4000103C	R	0x44	Status register (reduced width)
CTRL0 (CTRL)	0x40001040	R/W	0x02	Aliased control register – read and write bits 7:0
CTRL1 (CTRL)	0x40001044	R/W	0x01	Aliased control register – read and write bits 15:8
CTRL2 (CTRL)	0x40001048	R/W	0x00	Aliased control register – read and write bits 23:16
CTRL3 (CTRL)	0x4000104C	R/W	0x00	Aliased control register – read and write bits 25:24

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"I2C_x Register Map" (addresses for I2C_0 shown; I2C_1 begins at 0x40012000)				
CTRL	0x40002000	R/W	0	Used to configure the I^2C peripheral.
STATUS	0x40002004	R	0xF8	Read-only value which indicates the current state of the I^2C peripheral
DATA	0x40002008	R/W	0	Read/write data to/from the serial interface
ADDR	0x4000200C	R/W	0	Contains the primary programmable address of the I^2C peripheral
SMBUS	0x40002010	R/W	0b01X1X000	Configuration register for SMBus timeout reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N
FREQ	0x40002014	R/W	0x08	Necessary for configuring real-time timeout logic. Can be set to the PCLK frequency for 25 ms SMBus timeouts, or may be changed to increase/decrease the timeout value.
GLITCHREG	0x40002018	R/W	0x03	Number of registers in the glitch filter. Can be set to value from 3 to 6. Correct value to meet I^2C fast mode 50 ns spike suppression will depend on the PCLK frequency.

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"UART_x Register Map" (UART_0 addresses are shown; UART_1 begins at 0x40010000)				
RBR	0x40000000	R	–	Buffer Register
THR	0x40000000	W	–	Transmit Holding Register
DLR	0x40000000	R/W	0x01	Divisor Latch (LSB) Register
DMR	0x40000004	R/W	0	Divisor Latch (MSB) Register
IER	0x40000004	R/W	0	Interrupt Enable Register
IIR	0x40000008	R	0xC1	Interrupt Identification Register
FCR	0x40000008	W	0	FIFO Control Register
LCR	0x4000000C	R/W	0	Line Control Register
MCR	0x40000010	R/W	0	Modem Control Register
LSR	0x40000014	R	0x60	Line Status Register
MSR	0x40000018	R	0	Modem Status Register
SR	0x4000001C	R/W	0	Scratch Register
"Real-Time Counter Register Map" on page 301				
COUNTER0_REG	0x40014100	R/W	0	Counter bits 7:0
COUNTER1_REG	0x40014104	R/W	0	Counter bits 15:8
COUNTER2_REG	0x40014108	R/W	0	Counter bits 23:16
COUNTER3_REG	0x4001410C	R/W	0	Counter bits 31:24
COUNTER4_REG	0x40014110	R/W	0	Counter bits 39:32
MATCHREG0_REG	0x40014120	R/W	0	Match Register bits 7:0
MATCHREG1_REG	0x40014124	R/W	0	Match Register bits 15:8
MATCHREG2_REG	0x40014128	R/W	0	Match Register bits 23:16
MATCHREG3_REG	0x4001412C	R/W	0	Match Register bits 31:24
MATCHREG4_REG	0x40014130	R/W	0	Match Register bits 39:32
MATCHBITS0_REG	0x40014140	R/W	0	Individual Match bits 7:0
MATCHBITS1_REG	0x40014144	R/W	0	Individual Match bits 15:8
MATCHBITS2_REG	0x40014148	R/W	0	Individual Match bits 23:16
MATCHBITS3_REG	0x4001414C	R/W	0	Individual Match bits 31:24
MATCHBITS4_REG	0x40014150	R/W	0	Individual Match bits 39:32
CTRL_STAT_REG	0x40014160	R/W	0	Control (write) / status (read) bits 7:0
"System Timer Register Map" (addresses for TIM1 are shown; TIM2 begins at 0x40005018)				
TIMx_VAL (x = 1)	0x40005000	R	0x0	Current value of Timer 1
TIMx_LOADVAL (x = 1)	0x40005004	R/W	0x0	Load value for Timer 1
TIMx_BGLOADVAL (x = 1)	0x40005008	R/W	0x0	Background load value for Timer 1
TIMx_CTRL (x = 1)	0x4000500C	R/W	0x0	Timer 1 Control Register

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
TIMx_RIS (x = 1)	0x40005010	R/W	0x0	Timer 1 raw interrupt status
TIMx_MIS (x = 1)	0x40005014	R	0x0	Timer 1 masked interrupt status
TIMx_VAL (x = 2)	0x40005018	R	0x0	Current value of Timer 2
TIMx_LOADVAL (x = 2)	0x4000501C	R/W	0x0	Load value for Timer 2
TIMx_BGLOADVAL (x = 2)	0x40005020	R/W	0x0	Background load value for Timer 2
TIMx_CTRL (x = 2)	0x40005024	R/W	0x0	Timer 2 Control Register
TIMx_RIS (x = 2)	0x40005028	R/W	0x0	Timer 2 raw interrupt status
TIMx_MIS (x = 2)	0x4000502C	R	0x0	Timer 2 masked interrupt status
TIM64_VAL_U	0x40005030	R	0x0	Upper 32-bit word in 64-bit mode
TIM64_VAL_L	0x40005034	R	0x0	Lower 32-bit word in 64-bit mode
TIM64_LOADVAL_U	0x40005038	R/W	0x0	Upper 32-bit load value word in 64-bit mode
TITM64_LOADVAL_L	0x4000503C	R/W	0x0	Lower 32-bit load value word in 64-bit mode
TIM64_BGLOADVAL_U	0x40005040	R/W	0x0	Upper 32-bit background load value in 64-bit mode
TIM64_BGLOADVAL_L	0x40005044	R/W	0x0	Lower 32-bit background load value in 64-bit mode
TIM64_CTRL	0x40005048	R/W	0x0	Control Register in 64-bit mode
TIM64_RIS	0x4000504C	R/W	0x0	Raw interrupt status in 64-bit mode
TIM64_MIS	0x40005050	R	0x0	Masked interrupt status in 64-bit mode
TIM64_MODE	0x40005054	R/W	0x0	System Timer dual 32-bit or 64-bit mode

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
"GPIO Register Map"				
GPIO_x_CFG (x = 0)	0x40013000	R/W	0x0	GPIO Configuration register for bit 0
⋮	⋮	R/W	⋮	⋮
GPIO_x_CFG (x = 31)	0x4001307C	R/W	0x0	GPIO Configuration register for bit 31
GPIO_IRQ	0x40013080	R/W	0x0	Interrupt Status Register
GPIO_IN	0x40013084	R	0x0	Read only bits for ports configured as inputs
GPIO_OUT	0x40013088	R/W	0x0	Read/write bits for ports configured as outputs
"Fabric Interface and IOMUX Register Map"				
MSSIRQ_EN_0	0x40007000	R/W	0	Enables/disables interrupt sources for MSSINT[0]
MSSIRQ_EN_1	0x40007004	R/W	0	Enables/disables interrupt sources for MSSINT[1]
MSSIRQ_EN_2	0x40007008	R/W	0	Enables/disables interrupt sources for MSSINT[2]
MSSIRQ_EN_3	0x4000700C	R/W	0	Enables/disables interrupt sources for MSSINT[3]
MSSIRQ_EN_4	0x40007010	R/W	0	Enables/disables interrupt sources for MSSINT[4]
MSSIRQ_EN_5	0x40007014	R/W	0	Enables/disables interrupt sources for MSSINT[5]
MSSIRQ_EN_6	0x40007018	R/W	0	Enables/disables interrupt sources for MSSINT[6]
MSSIRQ_EN_7	0x4000701C	R/W	0	Enables/disables interrupt sources for MSSINT[7]
MSSIRQ_SRC_0	0x40007020	R/W	0	Source of interrupt for MSSINT[0]
MSSIRQ_SRC_1	0x40007024	R/W	0	Source of interrupt for MSSINT[1]
MSSIRQ_SRC_2	0x40007028	R/W	0	Source of interrupt for MSSINT[2]
MSSIRQ_SRC_3	0x4000702C	R/W	0	Source of interrupt for MSSINT[3]
MSSIRQ_SRC_4	0x40007030	R/W	0	Source of interrupt for MSSINT[4]
MSSIRQ_SRC_5	0x40007034	R/W	0	Source of interrupt for MSSINT[5]
MSSIRQ_SRC_6	0x40007038	R/W	0	Source of interrupt for MSSINT[6]

***Table 20-2* • SmartFusion Master Register Map (continued)**

Register Name	Address	R/W	Reset Value	Description
MSSIRQ_SRC_7	0x4000703C	R/W	0	Source of interrupt for MSSINT[7]
FIIC_MR	0x40007040	R/W	0	Fabric interface interrupt controller mode register
ADC Register Map				
SSE_TS_CTRL	0x40020004	R/W	0	Sample sequence engine time slot control
ADC_SYNC_CONV	0x40020008	R/W	0	Synchronized ADC control
ANA_COMM_CTRL	0x4002000C	R/W	1	Common analog block control
ADC0_CONV_CTRL	0x40020050	R/W	0	ADC 0 conversion control
ADC0_STC	0x40020054	R/W	0	ADC 0 sample time control
ADC0_TVC	0x40020058	R/W	0	ADC 0 time division control
ADC0_MISC_CTRL)	0x4002005C	R/W	0	ADC 0 control register
ADC1_CONV_CTRL	0x40020090	R/W	0	ADC 1 conversion control
ADC1_STC	0x40020094	R/W	0	ADC 1 sample time control
ADC1_TVC	0x40020098	R/W	0	ADC 1 time division control
ADC1_MISC_CTRL	0x4002009C	R/W	0	ADC 1 control register
ADC2_CONV_CTRL	0x400200D0	R/W	0	ADC 2 conversion control
ADC2_STC	0x400200D4	R/W	0	ADC 2 sample time control
ADC2_TVC	0x400200D8	R/W	0	ADC 2 time division control
ADC2_MISC_CTRL	0x400200DC	R/W	0	ADC 2 control register
ADC0_STATUS	0x40021000	R	0	Status of ADC 0
ADC1_STATUS	0x40021004	R	0	Status of ADC 1
ADC2_STATUS	0x40021008	R	0	Status of ADC 2
PPE_CTRL	0x40021404	R/W	0	Post processing engine control

A – List of Changes

The following table lists critical changes that were made in each revision of the SmartFusion MSS User's Guide.

Revision	Changes	Page
Revision 3 (September 2012)	The following information was added to the "ARM Cortex-M3 Microcontroller" section (SAR 35987): "A2F060 and A2F500 allow debug at up to 10 MHz, as limited by the debugger."	8
	Table 5-1 • eFROM Read/Write Capabilities by Access Mode was revised to clarify that IAP can program eFROM in A2F060 and A2F500 only (SAR 38827).	76
	Values for A2F060 were added to Table 6-1 • eSRAM Address Locations (SAR 36912).	79
	The second instance of LPXIN was corrected to LPXOUT in Figure 16-1 • Real-Time Counter System Block Diagram (SAR 38421).	299
	Figure 9-2 • MSS_RESET_N Output Buffer Configuration was modified to show the MSS_RESET_N is pulled up to 3.3 V (SAR 38750).	146
	The "The MSS_RESET_N Pin" section is new (SAR 37407).	146
	The reference to Sleep mode was changed to Standby mode in the "Time Keeping Mode" section (SAR 36365).	156
	The names and descriptions for bits 1 and 0 were exchanged in Table 11-9 • WDOGRIS (SAR 29407).	169
	The "SPD", "DBO", and "BLE" bits in Table 12-18 • CSR0 were corrected from R/W to W because they are not readable (SAR 30561).	197
	The register addresses were corrected in Table 13-4 • SPI Register Summary. The CMDSIZE and HWSTATUS registers were changed to Reserved. (SAR 31898).	232
	The description of the CLKMODE bit (bit 28) in Table 13-5 • CONTROL was revised (SAR 31898).	233
	The following bit definitions were added to tables in the "SPI Register Interface Details" section (SAR 31898): Table 13-7 • STATUS, [14:12] Table 13-8 • INT_CLEAR, [5:4] Table 13-11 • CLK_GEN, [7:4] Table 13-13 • MIS, [5:4] Table 13-14 • RIS, [5:4]	233
	The CMDINT bit in Table 13-8 • INT_CLEAR was changed to Reserved. It is used for the command system (SAR 31898).	237
	Bit descriptions in Table 13-11 • CLK_GEN, Table 13-13 • MIS, and Table 13-14 • RIS were updated for clarity and accuracy. The reset value of bits [6:0] in Table 13-16 • COMMAND were changed from 0x00 to 0 (SAR 31898).	238 – 240
	Table 13-18 • CMDSIZE and Table 13-19 • HWSTATUS were deleted (SAR 31898).	N/A
	Notes were added to Table 13-18 • STAT8 and Table 13-19 • CTRL to indicate that R/W and Reset Value are the same as the corresponding Control register bits (SAR 31898).	243

Revision	Changes	Page
Revision 3 (continued)	The reference to Standby and Sleep modes was changed to Standby and Time Keeping modes in the introduction to the "Real-Time Counter (RTC) System" section (SAR 36365).	299
	Table 18-1 • MSS I/O Voltage Standards and Drive Strengths was expanded to clarify the drive strengths for different I/O standards (SAR 37523).	319
	The "Locked Transactions" section is new (SAR 28771).	353
	The "16-Bit Fabric Master" section is new (SAR 25759).	353
	The reset value and description for "SYSTICK_CR" were corrected in Table 20-2 • SmartFusion Master Register Map (SAR 30564).	376
	The fax number has been changed to 408.643.6913 in the "Customer Service" section.	393
Revision 2 (December 2011)	The "ARM Cortex-M3 Microcontroller" section was revised to add "for the A2F200 device only" to this sentence: "SWV operates at 98 KHz for the A2F200 device only" (SAR 30680, SAR 27097). The example for generating a 10 ms tick in the "Cortex-M3 SysTick Timer" section was qualified as applicable to a –1 speed device (SAR 30845).	8
	ACE interrupt numbers were added to Table 1-5 • SmartFusion Interrupt Sources (SAR 24908).	10
	The register addresses in Table 2-2 • AHB Bus Matrix Register Map were changed from the form 0xE000XXXX to 0xE004XXXX (SAR 24522, 32736).	24
	The "AHB Bus Matrix Register Bit Definitions" section was revised to change the register address range to "0xE0042000 to 0xE004FFFF" for the AHB bus matrix control registers (SAR 26760).	24
	In the "AHB Bus Matrix Register Map" section, several reset values were corrected: The reset value for ESRAM_CR was changed from 0x0 to 0x00000010. The reset value for ENVM_CR was changed from 0x00000092 to 0x00000072. The reset value for ENVM_REMAP_SYS_CR was changed from 0x00080001 to 0x00000001. The reset value for AHB_MATRIX_CR was changed from 0x0 to 0x00000007 (SAR 35089).	24
	Table 2-6 • Bit Combination Definitions for ENVM_PIPE_BYPASS and ENVM_SIX_CYCLE was corrected. References to frequencies were removed; a reference to the eNVM section of the SmartFusion cSoC datasheet was included to assist in determining frequency (SAR 33219).	26
	The function for WRITE_ADJ was revised in Table 3-5 • CHANNEL_x_CONTROL (SAR 26076).	43
	The "Memory Organization" section was revised to add information about user eNVM pages. The information was added just before Table 4-3 • eNVM Section Sizes (SAR 24302).	53
	Table 4-4 • Latencies Corresponding to ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS and Figure 4-8 • eNVM Read: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 0, SEQ or NONSEQ Block Address (6:1:1:1) were significantly revised (SARs 26238, 27584).	54, 55
	The titles of Figure 4-7 • eNVM Read: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 1, SEQ or NONSEQ Block Address (5:1:1:1) and Figure 4-8 • eNVM Read: ENVM_SIX_CYCLE = 0 and ENVM_PIPE_BYPASS = 0, SEQ or NONSEQ Block Address (6:1:1:1) were revised to correct the block address in each (SAR 25341).	55

Revision	Changes	Page
Revision 2 (continued)	In Table 6-2 • ESRAM_CR Register Map, the ESRAM_CR address was changed to 0xE0042000 and the AHB_MATRIX_CR address was changed to 0xE0042018 (SAR 26760). The reset value for ESRAM_CR was changed from 0x0 to 0x00000010. The reset value for AHB_MATRIX_CR was changed from 0x0 to 0x00000007 (SAR 35089).	79
	The statement, 'In all devices the MSS has been designed to operate at up to 100 MHz" was added to the "Functional Description" section. Clarification of how the FCLK divides down within the MSS was added: "Within the MSS, FCLK can be further divided down via APB clock dividers and within those peripherals that will further divide down the APB clock" (SAR 25925).	109
	In Table 8-4 • Main Oscillator Operational Modes, the function for Low Gain was revised, changing "0.32 to 0.20 MHz" to "32 to 200 KHz" (SAR 26033).	121
	The following note was added to the "On-Chip RC Oscillator" section: "The accuracy of the on-chip RC oscillator makes it unsuitable as a clock source for the Ethernet MAC" (SAR 28099).	120
	The opening section in the "Reset Controller" chapter was revised, adding information that the MSS_RESET_REQ signal from the Cortex-M3 processor is controlled by the SYSRESETREQ bit in the Application Interrupt and Reset Control register. In the same section, Figure 9-1 • Reset Controller Block Diagram was revised to move the location of M3_PORESET_N to the signal above its previous location.	143
	The "Analog Reset" section is new and explains what occurs when the analog block is reset by the MSS (SAR 30664).	148
	Table 9-3 • ANA_COMM_CTRL is now included in the "Reset Controller" chapter. Previously it existed only in the "Analog-to-Digital Converter" chapter of the *SmartFusion Programmable Analog User's Guide* (SAR 23724).	148
	RTM_MATCH was corrected to RTC_MATCH in Figure 10-1 • VR and PSM Block Diagram (SAR 27179). A 1.5 V output label was placed above the capacitor.	151
	The following note was added to the "VR Init" section: "Changing the programmed state of the PUPO flash bits results in a change in behavior only if VCC33 and VBAT are both off (removed) following programming" (SAR 28441).	152
	Figure 10-3 • VR Block Diagram was revised to show the power supplies that generate the control signals (SAR 31446).	153
	The "Sleep Mode" and "Power-Down" sections were deleted from the "SmartFusion Power Modes" section and Figure 10-5 • Power State Diagram was modified accordingly (SAR 29479).	155
	In Table 10-2 • VR and PSM Control Registers, the address for DEVICE_SR was changed to 0xE0042034 (SAR 26760).	156
	The names and descriptions for bits 1 and 0 were exchanged in Table 11-7 • WDOGCONTROL and Table 11-10 • WDOGMIS (SAR 29407).	168, 169
	The second occurrence of RDES2 was changed to RDES3 in Table 12-2 • Receive Descriptors (RDESx) (SAR 28179).	179
	Figure 12-12 • MAC I/O Interaction with Fabric was replaced by Figure 12-12 • Example of Ethernet MAC Interaction with FPGA Fabric via an IOMUX (SAR 26540).	214
	Additional SPI registers and CONTROL register information for A2F060 and A2F500 was added to the "SPI Register Interface Summary" section and "SPI Register Interface Details" section. The "Control Bits SPS, SPO, and SPH (A2F060 and A2F500)" section is new (SARs 33040, 31898).	232, 233

Revision	Changes	Page
Revision 2 (continued)	The third bullet in the Slave mode "Transfer Example" was changed from "I2C peripheral generates interrupt request; STATUS register = 0x00 (Table 14-6 on page 261)" to "I2C peripheral generates interrupt request; STATUS register = 0x60 (Table 14-8 on page 265" (SAR 35352).	254
	The "Clockstretching" section is new (SAR 35833).	255
	Table 15-10 • FCR was modified to change the descriptions for bits 1 and 2. Previously these stated, "This shift register is not cleared." This sentence was changed to "The transmit shift register is not cleared" and "The receive shift register is not cleared." In Table 15-12 • MCR, the sentence reading, "The output of the Transmitter Shift Register is looped back into the Receiver Shift Register" was revised to read, "The output of the Transmitter is looped back into the Receiver" (SAR 24349).	283, 285
	The following sentence was deleted from the description of bit 5 in Table 15-13 • LSR: "Indicates that the UART is ready to transmit a new data byte" (SAR 24350).	286
	The "System Timer" chapter was revised to remove inconsistencies of register naming (some registers were formerly referred to with _REG at the end of the name) and to note that the timers can be used in polled mode as well as in generating interrupts. The description of TIMxENABLE in Table 17-5 • TIMx_CTRL was corrected and augmented (SAR 30846).	305
	The "Features of MSS General Purpose I/Os" section is new (SAR 24700).	319
	Figure 19-2 • Fabric Interface Controller System Overview was corrected by exchanging the Master and Slave labels in the FPGA fabric portion of the figure. A similar correction was made to Figure 19-3 • Mismatched FIC Interfaces (SAR 26375, SAR 25911).	344, 345
	The "FIIC Functional Description" section was revised to state, "ACE mode is selected by setting the MODE bit in the FIIC_MR to a 0." Previously the sentence stated to set the MODE bit to a 1 (SAR 25652). The bit definitions for FIIC_MIR in Table 19-24 • FIIC_MR were revised to match so that 0 is ACE mode and 1 is non-ACE mode (SAR 25653).	354, 365
	The FABINT description in Table 19-32 • Miscellaneous Interface Signals was revised to add information needed for generating this signal with user logic (SAR 33937). Additional data was included in the DMAREADY0 and DMAREADY1 descriptions to indicate logic level and duration (SAR 31331).	372
	The address for SYSTICK_CR in Table 20-2 • SmartFusion Master Register Map was corrected from 0xE000E010 to 0xE0042038 (SAR 30564).	376
	Additional SPI registers and CONTROL register information for A2F060 and A2F500 was added to the Table 20-2 • SmartFusion Master Register Map (SARs 33040, 31898).	376
	The "SmartFusion Programming" chapter was moved to the SmartFusion datasheet (also resolves SAR 29572, 31390).	N/A
Revision 1 (April 2010)	The "SmartFusion MSS UART Application Development" section is new (SAR 25331).	295
	The "SmartFusion MSS Timer Application Development" section is new (SAR 25331).	316
	The "SmartFusion MSS GPIO Application Development" section is new (SAR 25331).	341
	References to the SmartFusion MSS Firmware Drivers v2.0 User's Guide have been changed to the website reference for SmartFusion MSS Configurators and Drivers User's Guides.	N/A

Revision	Changes	Page
Revision 0 (March 2010)	The "ARM Cortex-M3 Microcontroller" section was revised to include the fact that SWV operates at 98 KHz.	7
	Figure 3-1 • PDMA Block Diagram was revised to change the APB Bus Matrix and APB Interface blocks to AHB Bus Matrix and AHB Interface	35
	Two bit-field names were corrected: WR_ADJUST was changed to WRITE_ADJ in the "Clocks" section and PDMA_SOFTRESET was changed to PDMA_SR in the "Resets" section.	38
	Table 3-6 • PERIPHERAL_SEL was revised to exchange the definitions of the 1000 and 1001 bit combinations.	44
	Table 3-7 • CHANNEL_x_STATUS was revised. The definitions of the bit numbers changed.	45
	The ENVM_CONFIG_0_REG and ENVM_CONFIG_1_REG registers were renamed to ENVM_0_CR and ENVM_1_CR. References to them throughout the chapter and datasheet were revised.	N/A
	Captions were revised in Figure 4-1 • Block Diagram of eNVM Controller with Two eNVM Blocks to make signal names consistent in format.	49
	Figure 4-5 • Address Decoding for eNVM Read Operations was revised.	53
	Bit 9 was changed to MSB and Bit 8 was changed to LSB in the headings for Table 4-12 • ENVM_STATUS_x.	67
	EMC_CONFIG_x_REG was replaced with EMC_CS_x_CR (x = 0 or 1). This was a terminology change throughout the document.	N/A
	The HWRITE signal was revised in Figure 7-3 • AHB Address/Data Phase for Write Transfer.	83
	The following sentence was removed from the "FCLK Cycles Required for Memory Accesses" section: "When latencies are zero, the number of phases listed in Table 14-4 represents the minimum number of FCLK cycles required for an EMD access."	89
	Table 7-8 • EMC_MUX_CR is new.	95
	Table 7-9 • EMC_CS_x_CR was revised to change the EMC_CSFEx field to EMC_MEMTYPEx.	95
	Table 7-13 • Mapping of AHB Transactions to EMC Accesses was revised. Values for EMC_DB Bits were added for port size 8 and HSIZE Half. Values for EMC_AB were added for port size 16 and HSIZE Word, and the other values in these rows adjusted to their proper position in the table.	97
	EMC_MRW_N was corrected to EMC_RW_N in figures where it occurred throughout the document.	N/A
	The "On-Chip RC Oscillator" section was revised to state that the RC oscillator is always on. Previously this section stated that the on-chip RC oscillator could be disabled by setting the RCOSCDISABLE bit in the MSS_RCOSC_CR. Reference to MSS_ROCOSC_CR was removed.	120
	The "Main Crystal Oscillator" section was revised to state that The main crystal oscillator can be enabled and disabled by the Cortex-M3 processor via the MSS_CCC_MUX_CR, bit 29 MAINOSCEN.	121
	The "Low-Power 32 KHz Crystal Oscillator" section was revised to remove reference to MSS_RCOSC_CR. CTRL_STAT_REG is referenced instead.	123
	OSC_CTRL was removed from Table 8-5 • PLL/CCC Register Map, and the OSC Control Register Bit Definitions was removed.	124

Revision	Changes	Page
Revision 0 (continued)	The address for MSS_CCC_PLL_CR was changed from 0xE0040054 to 0xE0042054.	124
	The "Functional Description" section was revised to change the reference to the MMS_STATUS_REG to its correct name: MSS_SR.	143
	Table 9-2 • Reset Controller Memory Map was revised to change the register address for SOFT_RST_CR from 0X0002030 to 0xE0042030.	148
	The "1.5 V Voltage Detector (VCC15UP)" section and "3.3 V Voltage Detector (VCC33UP)" section were revised.	151, 152
	Captions in Figure 10-1 • VR and PSM Block Diagram and Figure 10-3 • VR Block Diagram were revised to correct them from VCC3A and VDD33 to VCC33A.	151, 153
	The "SmartFusion Power Modes" section was replaced. FPGA mode was deleted. Table 10-2 • SmartFusion Power Modes, Table 10-3 • SmartFusion Wake-up Transition Events, and Table 10-4 • SmartFusion Power-Down Transition Events were replaced by Figure 10-5 • Power State Diagram.	155 through 156
	Table 10-1 • VR and PSM Related Interrupts was revised to change the addresses for CLR_MSS_SR and DEVICE_SR.	155
	Table 10-2 • VR and PSM Control Registers was revised to remove MSS_RCOSC_CR. The MSS_RCOSC_CR Bit Definitions table was removed.	156
	The description for MSSVRON in Table 10-6 • VRPSM_CR was revised.	161
	The AND gate above the WDOGRIS register was deleted in Figure 11-1 • Watchdog Block Diagram.	163
	The definitions and descriptions in Table 11-6 • WDOGENABLE were revised.	168
	The R/W field values were changed to R in Table 11-8 • WDOGSTATUS.	168
	Table 11-9 • WDOGRIS was corrected. The reserved bit numbers were changed from 1:2 to 31:2.	169
	Table 11-11 • MSS_SR was revised to change SYSRESETn to MSS_SYSTEM_RESET_N.	170
	Instances in signal names of H2F and F2H were changed to M2F and F2M to indicate "microcontroller subsystem to fabric" and "fabric to microcontroller subsystem."	N/A
	In the "Receive FIFO (RFIFO)" section, the size of the receive FIFO was corrected to 1,024 × 32 bits.	175
	The last two paragraphs were deleted from the "Descriptor / Data Buffer Architecture Overview" section.	178
	"Pull demand command" was changed to "Poll demand command" in Figure 12-6 • Transmit Process Transitions and Figure 12-7 • Receive Process Transmissions.	188, 189
	Figure 12-12 • Example of Ethernet MAC Interaction with FPGA Fabric via an IOMUX was revised to correct the internal signals in the MUX.	214
	GND was changed to VDD in Table 12-43 • IOMUX 16 and Table 12-44 • IOMUX 17.	215
	Reserved bits were added to bit definition tables in the "Serial Peripheral Interface (SPI) Controller" section.	219
	Figure 13-1 • SPI Controller Block Diagram was revised to change SPI_SS[0] to SPI_x_SS[0].	219
	The DMA Mode section was renamed to the "PDMA Mode" section.	220
	A note regarding SPI_x_SS[7:1] was added to Table 13-1 • SPI Interface Signals.	221

Revision	Changes	Page
Revision 0 (continued)	SPI_x_SS was changed to SPI_x_SS[0] in the "SPI Status at Reset" section.	222
	The "SPI Clock Requirements" section was revised to state that the input clock to the SPI controller (SPICLK) can not be faster than one twelfth of PCLK0 or PCLK1.	222
	The definitions of the modes for timing diagrams in the "SPI Data Transfer Protocol Details" section were revised. Table 13-2 • Motorola SPI Transfer Modes (A2F200 only) was revised accordingly.	223
	The description was revised in Table 13-12 • SLAVE_SELECT and a note was added to the table.	238
	Table 13-20 • SPI Signal GPIO and Fabric Mapping was revised.	245
	Signals were deleted for IOMUX 60 through IOMUX 63 in Table 13-21 • SPI Extra Signal GPIO and Fabric Mapping. The signal names for IOMUX 70 through IOMUX 76 were changed from SPI_0_SS[x] to SPI_1_SS[x].	246
	"IOMUX 60" through "IOMUX 63" were deleted. These buffers are not connected to the MSS or fabric.	18-25–18-26
	Bit 3 in Table 15-10 • FCR was changed from reserved to ENABLE_TXRDY_RXRDY.	283
	The GPIOCFG_x register was renamed to GPIO_x_CFG throughout the document.	N/A
	OE_A was changed from GND to High in Table 18-13 • IOMUX 4.	326
	The "FIIC Functional Description" section was revised to include information about the MSS to fabric soft interrupt assertion.	354
	Table 19-28 • IOMUX to Peripheral Association was revised to remove IOMUXes 57-63 from the SPI_0 row, as they were incorrectly included.	369
	The SYSREG register table was added and duplicate information was removed from Table 20-2 • SmartFusion Master Register Map. The format was changed to include more information about each register. Some of the descriptions were updated.	N/A
	The PDMA Register names were corrected and the real-time counter register map was added in Table 20-2 • SmartFusion Master Register Map.	376
Draft B (December 2009)	Nomenclature was revised for SmartFusion registers and interrupts. References to AHB-Lite and AHBL were changed to AHB.	N/A
	Figure 1-1 • Cortex-M3 R1P1 Block Diagram was revised. The implementation specifics were also revised.	7
	The reset values in Table 1-3 • SYSTICK_CR were revised.	9
	Table 1-5 • SmartFusion Interrupt Sources was replaced.	10
	Figure 2-1 • AHB Bus Matrix Masters and Slaves and Table 2-1 • AHB Bus Matrix Connectivity were revised to change some of the terminology. The "Functional Description" section was revised with terminology changes.	15, 16
	The "Remapping Embedded SRAMs" section was revised significantly.	21
	The "System Boot" section was revised to include the register address for the system boot code.	23
	Table 2-2 • AHB Bus Matrix Register Map was revised to change addresses from E004xxxx to E000xxxx. The location of the AHB bus matrix control registers in the system register space was changed to 0xE0004000 to 0xE0004FFF.	24
	Table 2-4 • ENVM_CR was revised to change the reset value for ENVM_SIX_CYCLE to 1.	25

Revision	Changes	Page
Draft B (continued)	Several of the labels in Figure 4-1 • Block Diagram of eNVM Controller with Two eNVM Blocks were revised.	49
	A brief explanation and definition of terms was added to better introduce Figure 4-2 • Block Diagram for eNVM Controller.	50
	Figure 4-3 • eNVM Organization was clarified by including notes that define block, page, sector, and array.	51
	The "Read Control" section and "Read Next Operation" section were modified.	53
	Table 4-4 • Latencies Corresponding to ENVM_SIX_CYCLE and ENVM_PIPE_BYPASS was revised.	54
	Figure 4-6 • Five-Cycle Read Data Path, ENVM_SIX_CYCLE = 0 was revised.	54
	Table 6-2 • ESRAM_CR Register Map was revised to change addresses from E004xxxx to E000xxxx.	79
	Table 9-2 • Reset Controller Memory Map was revised to change the address of ANA_COMM_CTRL to 4002000C.	148
	The address was changed for MSS_CR in Table 10-2 • VR and PSM Control Registers.	156
	Table 11-1 • Watchdog Register Interface was revised to change the address for MSS_SR to E004201C from E0042000.	166
	Figure 12-2 • RMII Management Interface and its explanatory text was moved toward the beginning of the document.	174
	The "SPI Controller Block Diagram" section was revised.	220
	Table 15-9 • Interrupt Identification Bit Values is new.	282
	Table 15-14 • MSR was revised to add more information to the descriptions.	287
	The "Low-Power Crystal Oscillator Functional Description" section was revised to remove reference to the OSC32KHZDISABLE bit.	299
	Reference to the VCC3AP pin was changed to VCCLPXTAL in the "Battery Switching Circuit Functional Description" section.	300
	The description for RSTB_CNT in Table 16-2 • CTRL_STAT_REG was revised.	302
	The "Fabric Interface Controller" section was revised to include information about which interface(s) to use when implementing peripherals in the fabric.	344

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

F

G

I

L

M

N

O

P

R

S

T

U

V

W

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

50200250-3/9.12