SEC2410/SEC4410

HS Endpoint Processor with USB 2.0, Smart Card, & FMC for Secure Token & Storage

PRODUCT FEATURES

Datasheet

General Description

The SMSC SEC2410/SEC4410 are USB 2.0 compliant, hi-speed bulk-only mass storage class peripheral controllers. They are intended to be used to read and write to popular flash media, including Secure Digital (SD), and MultiMediaCard™ (MMC) families.

The SMSC SEC2410/SEC4410 are fully integrated, single-chip solutions capable of ultra-high performance operation. Average sustained transfer rates exceeding 35 MB/s are possible if the media and host can support those rates. The SMSC SEC2410/SEC4410 includes provisions to read/write to secure media formats, as well as support AES encryption, without performance impact.

General Features

- The SEC2410/SEC4410 is available in two lead-free RoHS compliant packages:
  - 64-pin QFN (9x9 mm) package
  - 72-pin QFN (10x10 mm) package that includes debug pins to interface to standard ARM debug tools
- Hardware-controlled data flow architecture for all self-mapped media
- Pipelined hardware support for access to non-self-mapped media
- Order number (see next page) with *i* denote the products that support the industrial temperature range of -40°C to 85°C
- Support included for secure media format on a licensed, customized basis
  - SD Secure

Hardware Features

- Single-chip flash media controller containing:
  - A multiplexed interface for use with combo card sockets
  - SD/MMC flash media reader/writer
- SDIO and MMC streaming mode support
- Extended configuration options
- Media Activity LED
- GPIO configuration and polarity
  - Up to 32 GPIOs for special function use
  - One GPIO with up to 200 mA drive
- On board 24 MHz crystal driver circuit
- Optional external 24 MHz clock input
- Internal card power FET
  - 200 mA
  - "Fold-back" short circuit protection
- ARM M3 32-bit microprocessor
  - 60 MHz execution speed at 1 cycle per instruction (minimum)
  - 32 KBytes of internal SRAM for a general purpose scratchpad
  - 96 KByte SRAM available for code execution
  - 32 KByte internal code ROM
  - JTAG interface
- Supports a single external 3.3 V supply source; internal regulators provide 1.2 V internal core voltage for additional bill of materials and power savings
- Optimized pinout improves signal routing, easing implementation for improved signal integrity
- 1.2 V reference voltage for HSIC (SEC4410 only)

Flash Media Specification Compliance

- Secure Digital 2.0
  - HS-SD, SDHC, SDXC
  - TransFlash™ and reduced form factor media
- MultiMediaCard
  - MMC version 4.2: 1/4/8-bit
  - eMMC version 4.4

Software Features

- Customizable vendor-specific data
- Reduced memory footprint

Applications

- Secure dongles and storage
- Flash media card reader/writers
- Desktop and mobile PCs
- Consumer A/V and media players/viewers
- Compatible with
  - Microsoft® Vista™ and Vista ReadyBoost™
  - Windows® 7, XP, ME, 2K SP4
  - Apple Mac OSx®
  - Linux Mass Storage Class Drivers

Order Numbers:

ORDER NUMBERS	LEAD-FREE ROHS COMPLIANT PACKAGE	PACKAGE SIZE (mm)	TEMPERATURE RANGE
SEC2410/SEC2410-JZX	64QFN	9x9	0ºC to 85ºC
SEC4410/SEC4410i-JZX			-40ºC to 85ºC
SEC2410/SEC2410-AKZE	72QFN	10x10	0ºC to 85ºC
SEC4410/SEC4410i-AKZE			-40ºC to 85ºC

This product meets the halogen maximum concentration values per IEC61249-2-21.

For RoHS compliance and environmental information, please visit www.smsc.com/rohs

Conventions

Within this manual, the following abbreviations and symbols are used to improve readability.

Example	Description
BIT	Name of a single bit within a field
FIELD.BIT	Name of a single bit (BIT) in FIELD
x…y	Range from x to y, inclusive
BITS[m:n]	Groups of bits from m to n, inclusive
PIN	Pin Name
zzzzb	Binary number (value zzzz)
0xzzz	Hexadecimal number (value zzz)
zzh	Hexadecimal number (value zz)
rsvd	Reserved memory location. Must write 0, read value indeterminate
`code`	Instruction code, or API function or parameter
Section Name	Section or Document name
$\overline{\text{VAL}}$	Over-bar indicates active low pin or register bit
x	Don’t care
<Parameter>	<> indicate a Parameter is optional or is only used under some conditions
{,Parameter}	Braces indicate Parameter(s) that repeat one or more times
[Parameter]	Brackets indicate a nested Parameter. This Parameter is not real and actually decodes into one or more real parameters.

Chapter 1 General Description

SEC2410/SEC4410 is a flash media card reader solution intended to provide a flexible means of providing embedded Audio/video systems (TVs, DVD players, STBs, Portable Media Copiers or Players, etc.) access to Media files stored on Flash Media Cards such as Secure Digital/MultiMediaCard (SD/MMC), and NAND Flash.

SEC2410/SEC4410 is fully compliant with the *USB 2.0 Specification*. All required transivers and resistors of the USB ports are integrated into the device. This includes all series termination resistors on D+ and D- pins and all required pull-down and pull-up resistors. The over-current sense inputs for the downstream facing ports have internal pull-up resistors. One Control, One interrupt Pair, and Two Bulk Pair Endpoints are provided with reconfigurable Endpoint Buffers.

SEC2410/SEC4410 incorporates a powerfull ARM M3 32-bit microprocessor with 60 MHz execution speed at 1 cycles per instruction (minimum).

Following memories are embedded:

- 32 KBytes of internal SRAM for general purpose scratchpad
- 96 KByte SRAM available for code execution
- 32 KByte Internal Code ROM
- 10 KBytes of reconfigurable Endpoint Buffers
- 2 KByte OTP

It also supports optional 4 MByte External Code Space using SPI Flash memory.

SEC2410/SEC4410 has on-chip SD/MMC Controller. It supports:

- High-Speed MMC version 4.2: 1/4/8 bit MMC
- eMMC version 4.4
- High-Speed SD card, SDHC
- SDXC in SDR25 Mode (no support for SD card UHS25, SDR50, or SDR100).
- TransFlash™ and reduced form factor media.
- Hardware support for Secure Digital(SD) Pass-Through
- Hardware support for SD Security Command Extensions
- Hardware support for SDIO (SD Input/Output)

It has on-chip power FET's for supplying flash media card power with minimum board components.

SEC2410/SEC4410 supports SmartCard interface, ISO/IEC 7816 compliant and has Integrated 3/1.8 Volt regulator.

Integrated cryptographical module offers AES encryption with AES 128, AES 192, AES 256 key sizes and ECB, CBC or CTR implementation.

SEC2410/SEC4410 offers up to 32 GPIOs with diverse configuration and polarity options for special function such as LED indicators, button inputs, and power control to memory devices. The number of actual GPIOs depends on the implemented configuration. One GPIO available with up to 200 mA drive and “fold-back” short circuit protection

SEC2410/SEC4410 has a 24 MHz Crystal Driver Circuit and internal PLL for 480 MHz USB 2.0 Sampling on board.

It supports a single external 3.3 V supply source. Internal regulators provide 1.2 V internal core voltage for additional bill of materials and power savings.

SEC2410/SEC4410 is offered as a single chip flash media controller in 64-pin and 72-pin QFN, lead-free RoHS compliant packages in either SEC2410/SEC4410 commercial temperature range from 0°C to +70°C or industri alfrom -40°C to +85°C.

It supports USB Mass Storage Compliant Bootable BIOS, firmware upgrade via USB bus for SPI Flash and SD/MMC cards (“boot block flash” not required).

Compatible with Microsoft Vista; Windows 7, XP, and 2K SP3&4; Mac OS X 10; and Linux Multi-LUN Mass Storage Class Drivers

1.1 SEC2410 Block Diagram

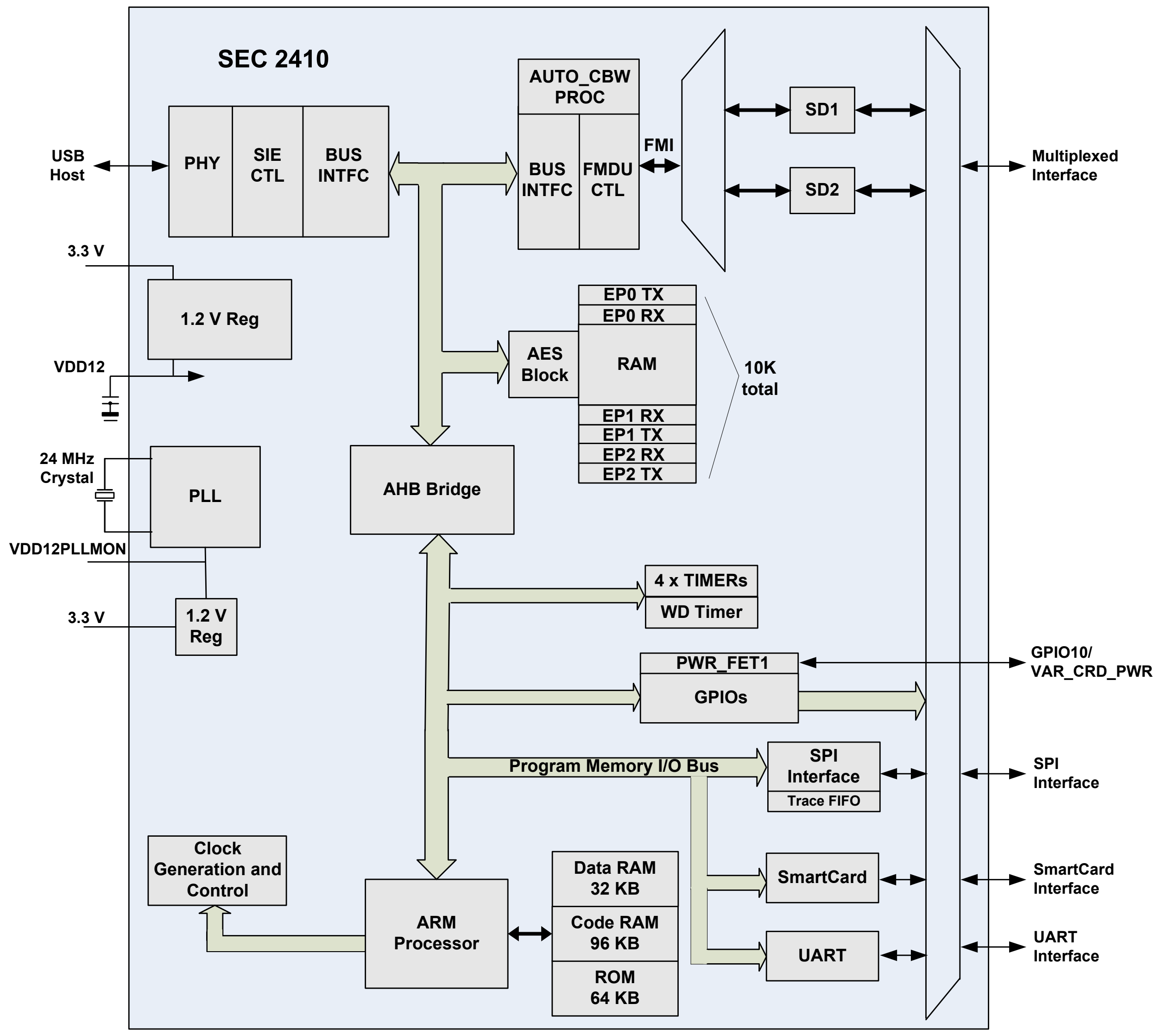


Figure 1.1 SEC2410 Block Diagram

1.2 SEC4410 Block Diagram

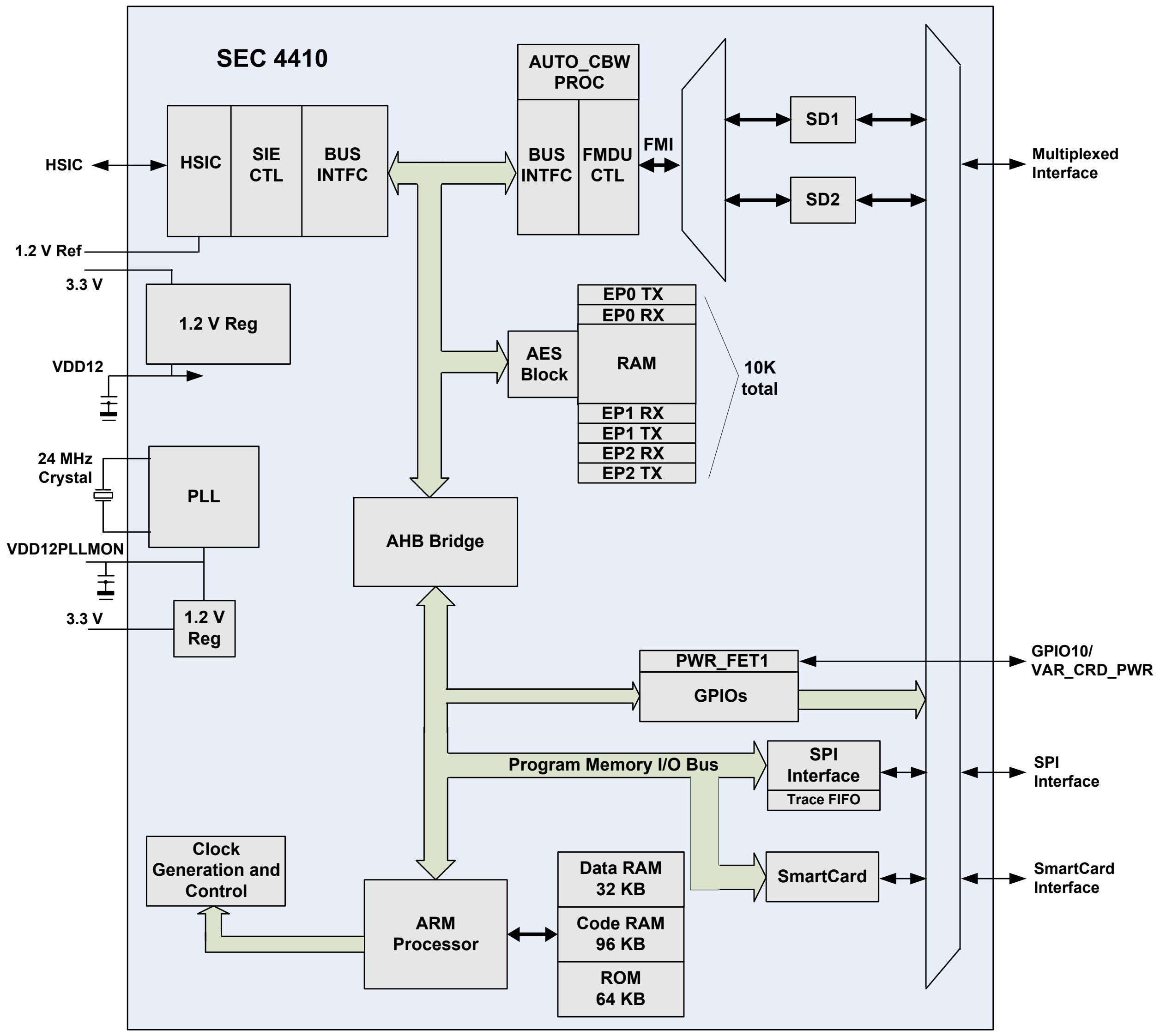


Figure 1.2 SEC4410 Block Diagram

Chapter 2 Configuration Options

The SEC2410/SEC4410 can be configured to support the desired interfaces as outlined in the sections below. The SEC2410/SEC4410 can be programmed using one of the configured interfaces as outlined in Chapter 22: *Bootloader* on page 290. Programming the SEC2410/SEC4410 using any of the described interfaces is outlined in the *Software Development Reference Guide* [6].

2.1 SPI ROM

The SPI ROM must be 1 Mbit or larger and support either 30 MHz or 60 MHz. The frequency used is set using **SPD_SEL**. For 30 MHz operation, this pin must be pulled to ground through a 100 kΩ resistor. For 60 MHz operation, this pin must pulled up through a 100 kΩ resistor. During **RESET_N** assertion, the **SPD_SEL** pin is tri-stated. When **RESET_N** is negated, the value on the pin is internally latched, and the pin reverts to **SPI_DO** functionality.

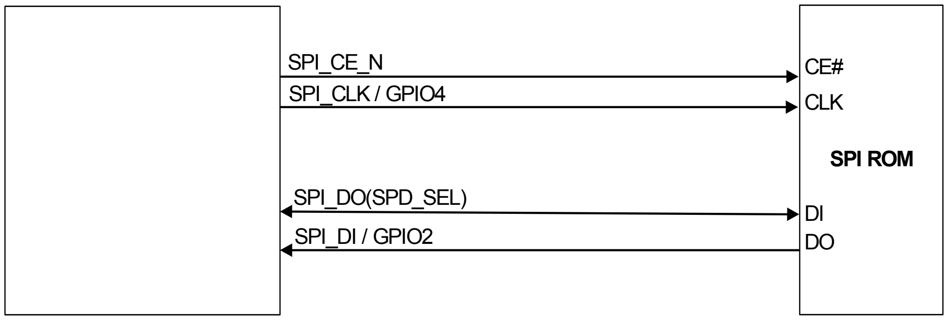


Figure 2.1 SPI ROM Connection

2.2 Supported System Configurations

This chapter illustrates some possible configurations available for SEC2410/SEC4410.

2.2.1 SD Plus SC Configuration

This one one SD interface, Smartcard and rest are 3.3 Volt GPIOs.

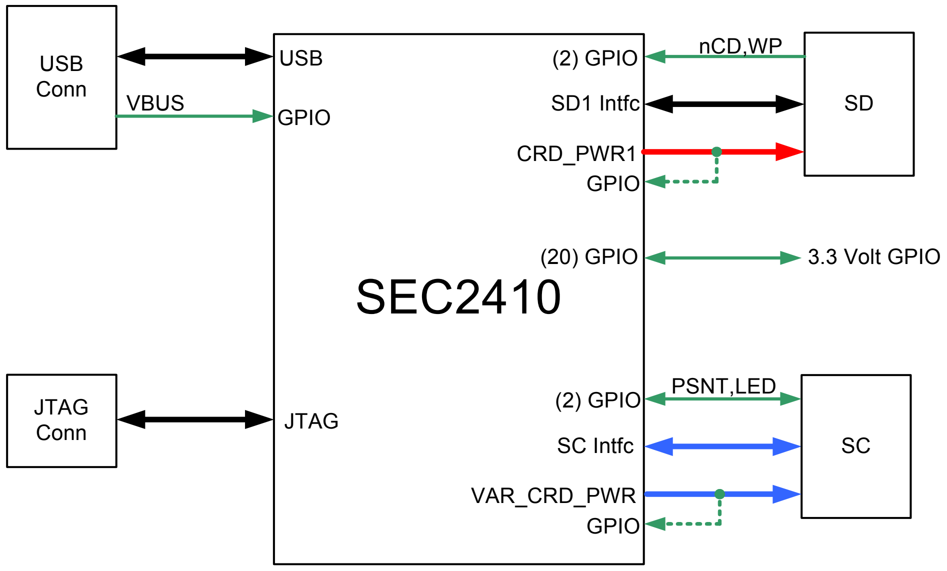


Figure 2.2 SD Plus SC Configuration

2.2.2 Dual SD Configuration

In this configuration two SD/MMC cards are supported. There are 5 variable voltage GPIOs available and 6 fixed voltage GPIOs.

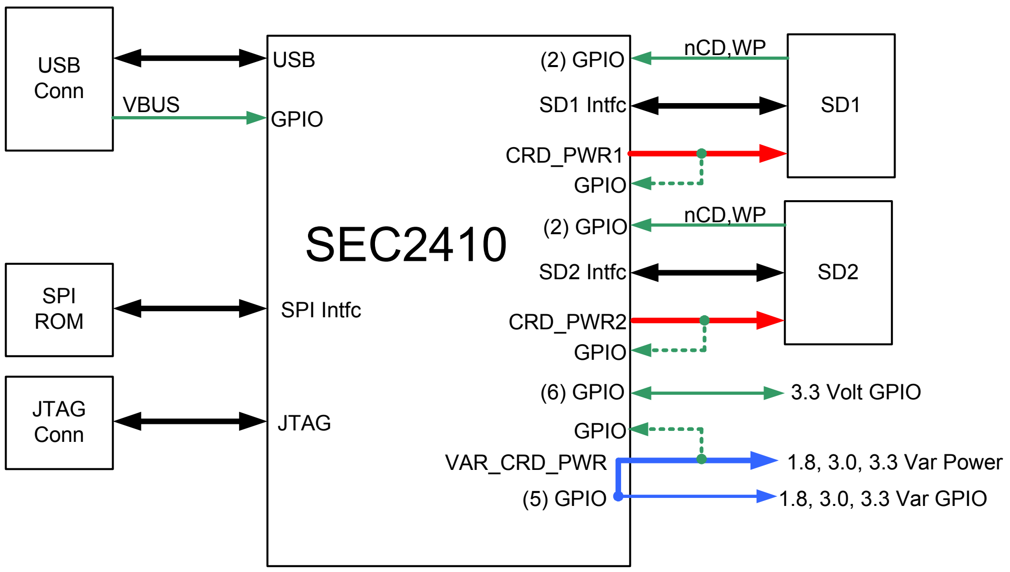


Figure 2.3 Dual SD Configuration

2.2.3 GPIO Only Configuration

In this configuration, the device would have to boot from USB and would be used as a GPIO controller.

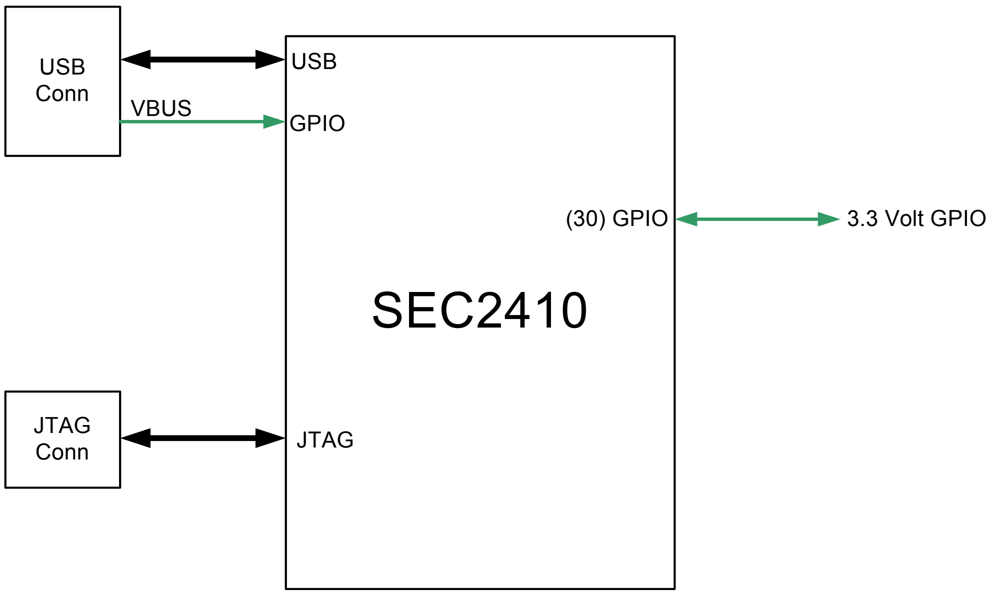


Figure 2.4 GPIO Only Configuration

Chapter 3 Pin Information

This chapter outlines the pin configurations for each package type available, followed by a corresponding pin list organized by group. The detailed pin descriptions are then outlined in Section 3.3: *Pin Descriptions* on page 16.

3.1 Pin Configurations

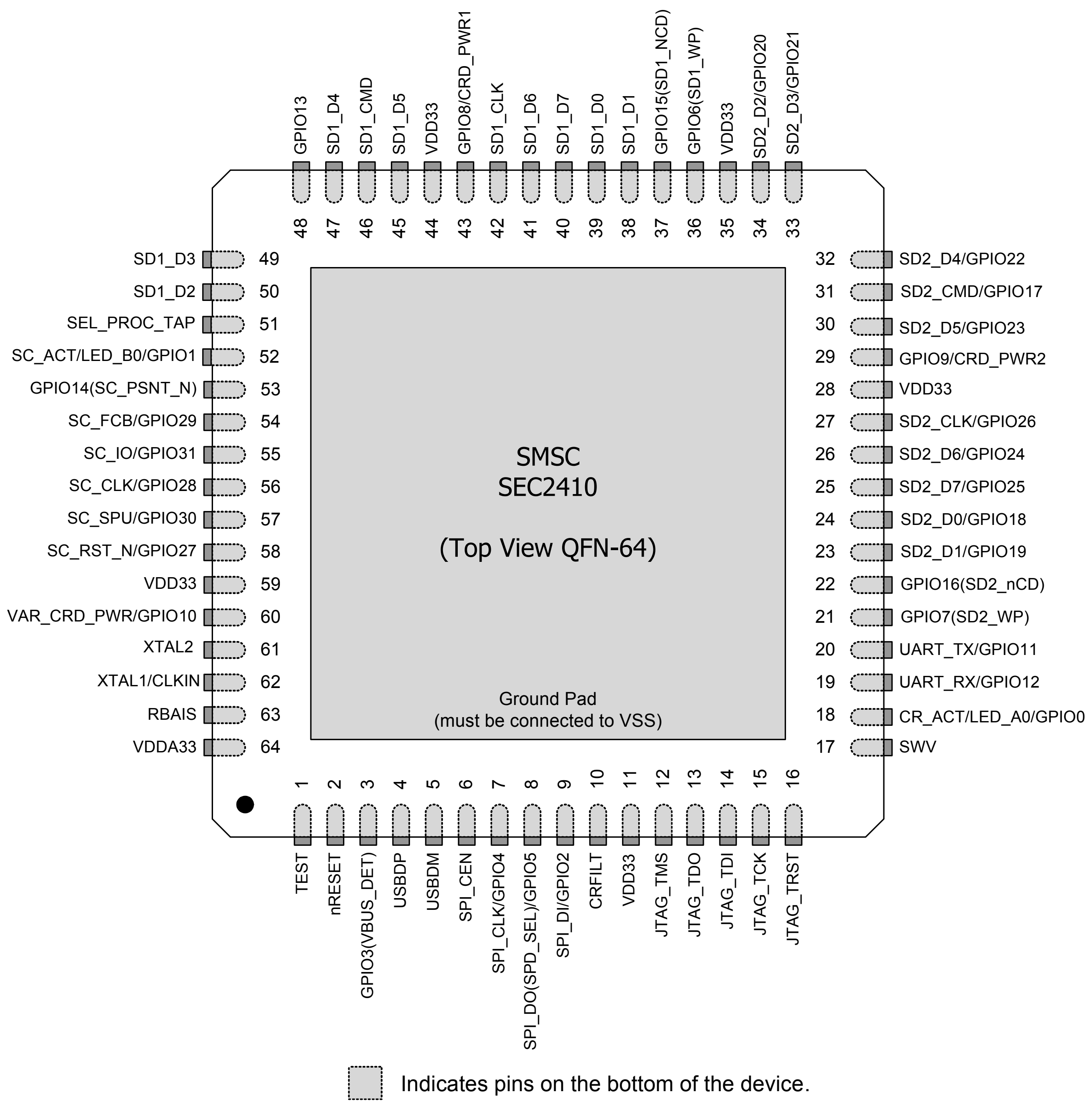


Figure 3.1 SEC2410 64-Pin QFN Package

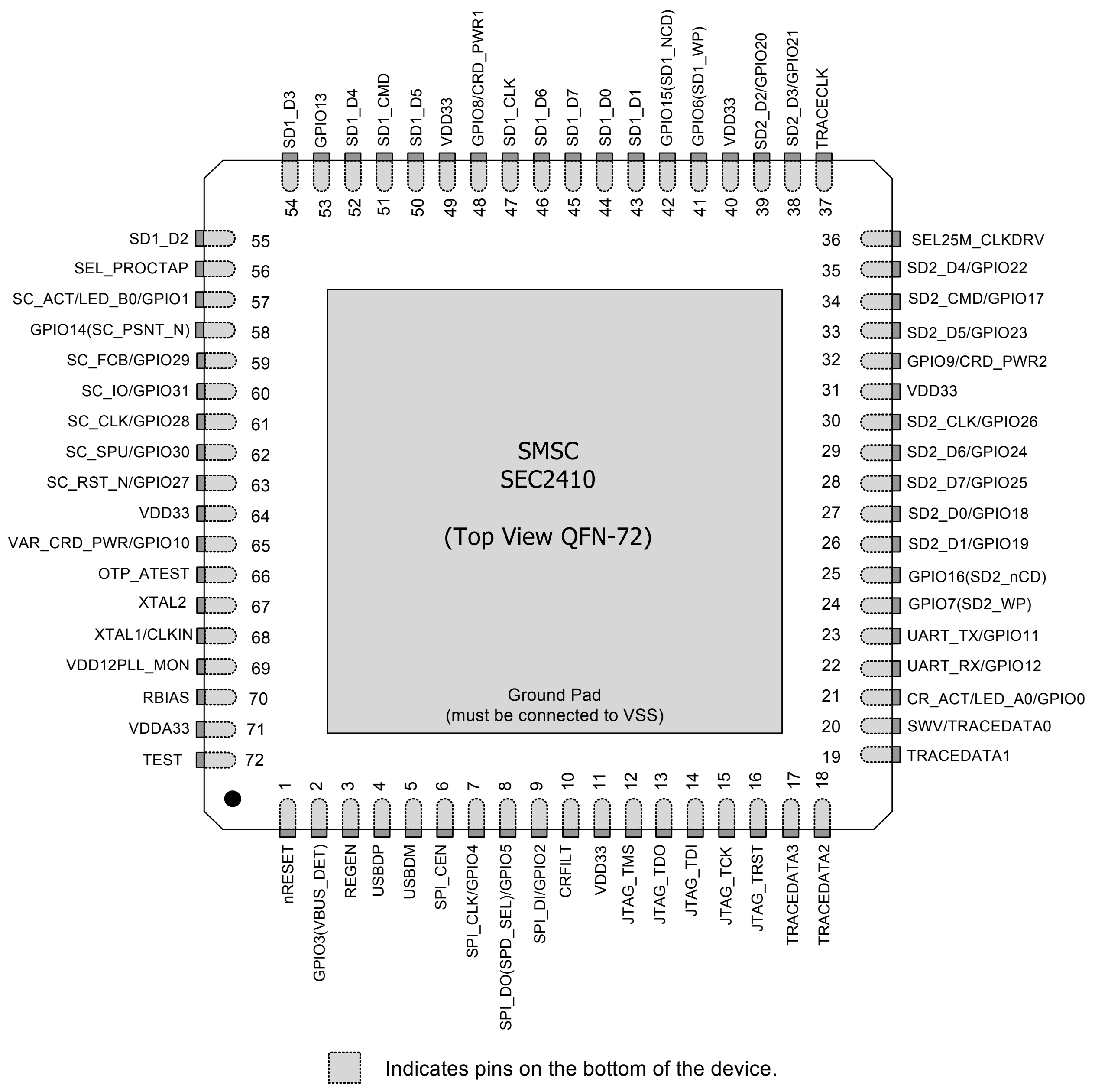
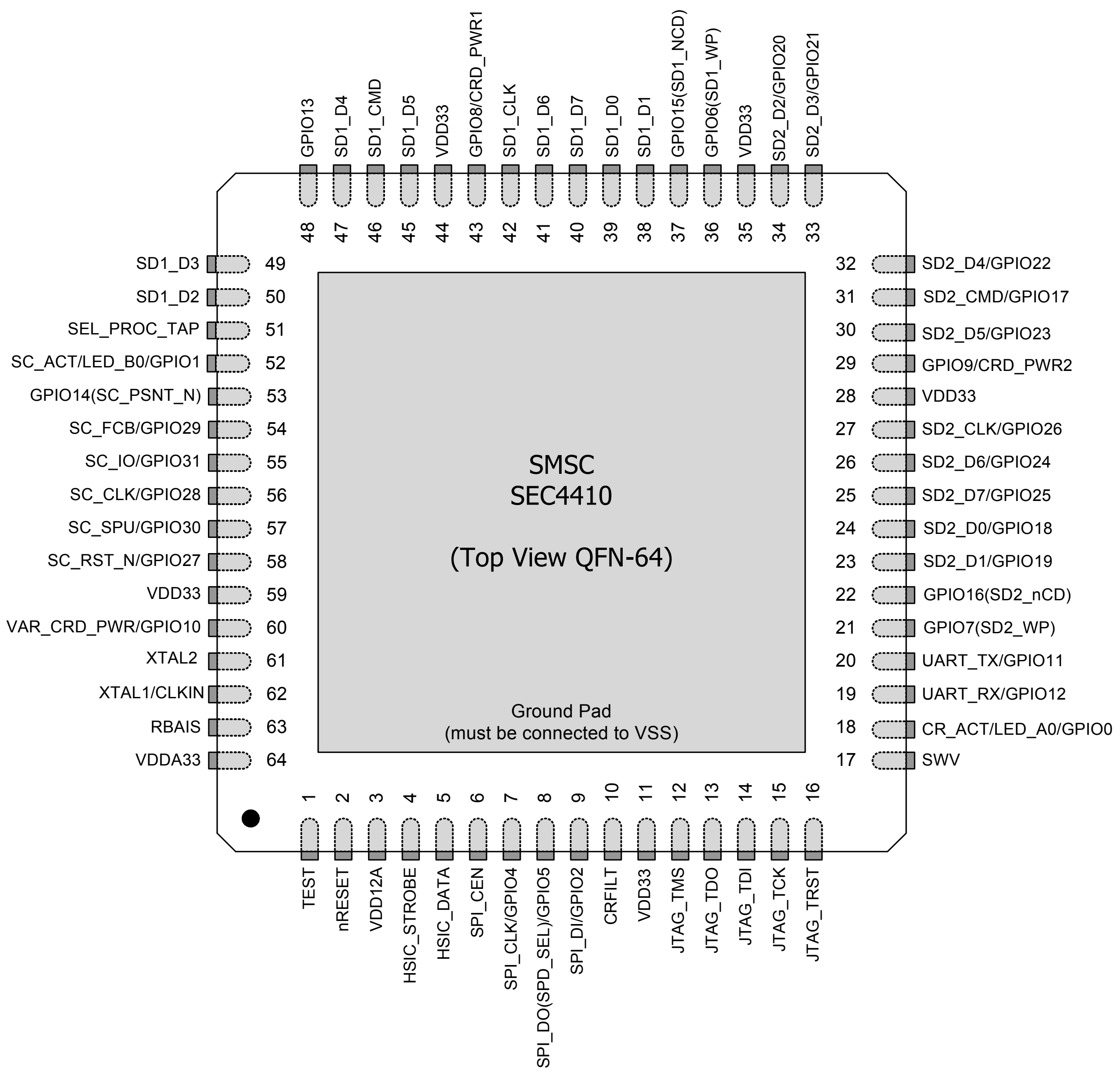


Figure 3.2 SEC2410 72-Pin QFN Package



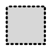
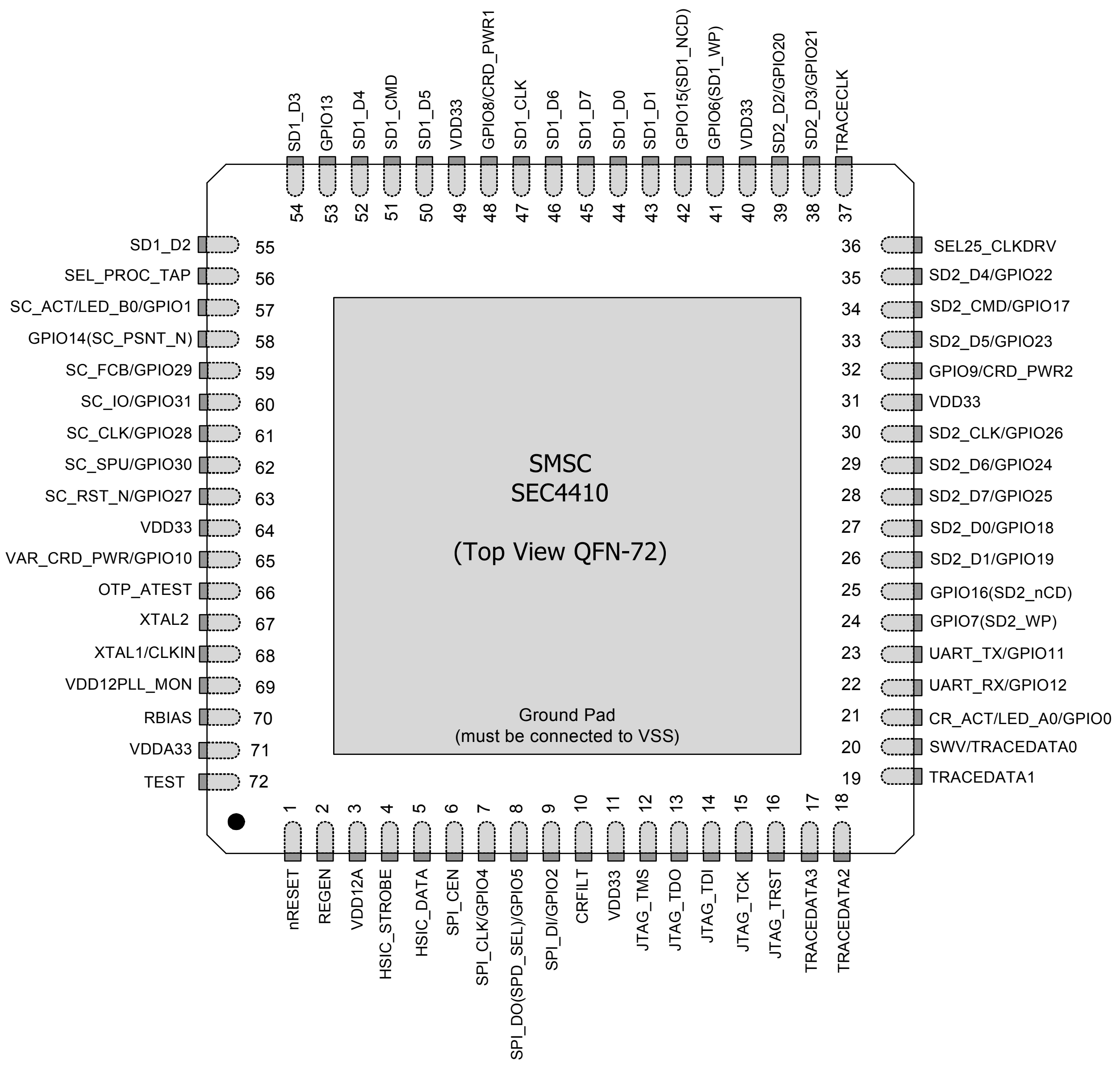

Indicates pins on the bottom of the device.

Figure 3.3 SEC4410 64-Pin QFN Package



▨ Indicates pins on the bottom of the device.

Figure 3.4 SEC4410 72-Pin QFN Package

3.2 Pin List (Grouped by Function)

Table 3.1 SEC2410 Pins Grouped by Function

SECURE DIGITAL/MULTIMEDIACARD INTERFACE (12 PINS)			
SD1_D0	SD1_D1	SD1_D2	SD1_D3
SD1_D4	SD1_D5	SD1_D6	SD1_D7
SD1_CMD	SD1_CLK	GPIO6 (SD1_WP)	GPIO15 (SD1_nCD)
SECOND SECURE DIGITAL INTERFACE (12 PINS)			
SD2_D0/ GPIO18	SD2_D1/ GPIO19	SD2_D2/ GPIO20	SD2_D3/ GPIO21
SD2_D4/ GPIO22	SD2_D5/ GPIO23	SD2_D6/ GPIO24	SD2_D7/ GPIO25
SD2_CLK/ GPIO26	SD2_CMD/ GPIO17	GPIO7 (SD2_WP)	GPIO16 (SD2_nCD)
SPI MASTER/I^{2}C INTERFACE (4 PINS)			
SPI_CEN	SPI_CLK/ GPIO4	SPI_DI/ GPIO2	SPI_DO (SPD_SEL)/ GPIO5
SMARTCARD (8 PINS)			
SC_RST_N/ GPIO27	SC_CLK/ GPIO28	SC_IO/ GPIO31	VAR_CRD_PWR/ GPIO10
SC_SPU/ GPIO30	GPIO14 (SC_PSNT_N)	SC_ACT/ LED_B0 GPIO1	SC_FCB/ GPIO29
USB INTERFACE (7 PINS)			
USBDP	USBDM	RBIAS	VDDA33
GPIO3 (VBUS_DET)	XTAL1/CLKIN	XTAL2	
JTAG INTERFACE (5 PINS)			
JTAG_TMS	JTAG_TDO	JTAG_TDI	JTAG_TCK
JTAG_TRST			
MISC (9 PINS)			
RESET_N	CR_ACT/ LED_A0/ GPIO0	TEST	GPIO9/ CRD_PWR
UART_TX/ GPIO11	UART_RX/ GPIO12	GPIO13	SEL_PROC_TAP
NC			

Table 3.1 SEC2410 Pins Grouped by Function (continued)

DIGITAL, POWER (6 PINS)			
(5) VDD33	(1) VDD12PLL_MON		
TOTAL 64			
EXTRA PINS FOR 72-PIN PACKAGE			
SWV/TRACEDATA0	TRACEDATA1	TRACEDATA2	TRACEDATA3
TRACECLK	SEL25M_CLKDRIVE		

Table 3.2 SEC4410 Pins Grouped by Function

SECURE DIGITAL/MULTIMEDIACARD INTERFACE (12 PINS)			
SD1_D0	SD1_D1	SD1_D2	SD1_D3
SD1_D4	SD1_D5	SD1_D6	SD1_D7
SD1_CMD	SD1_CLK	GPIO6 (SD1_WP)	GPIO15 (SD1_nCD)
SECOND SECURE DIGITAL INTERFACE (12 PINS)			
SD2_D0/ GPIO18	SD2_D1/ GPIO19	SD2_D2/ GPIO20	SD2_D3/ GPIO21
SD2_D4/ GPIO22	SD2_D5/ GPIO23	SD2_D6/ GPIO24	SD2_D7/ GPIO25
SD2_CLK/ GPIO26	SD2_CMD/ GPIO17	GPIO7 (SD2_WP)	GPIO16 (SD2_nCD)
SPI MASTER/I2C INTERFACE (4 PINS)			
SPI_CEN	SPI_CLK/ GPIO4	SPI_DI/ GPIO2	SPI_DO (SPD_SEL)/ GPIO5
SMARTCARD (8 PINS)			
SC_RST_N/ GPIO27	SC_CLK/ GPIO28	SC_IO/ GPIO31	VAR_CRD_PWR/ GPIO10
SC_SPU/ GPIO30	GPIO14 (SC_PSNT_N)	SC_ACT/ LED_B0 GPIO1	SC_FCB/ GPIO29
USB INTERFACE (8 PINS)			
HSIC_DATA	HSIC_STROBE	VDD12A	VDDA33
GPIO3 (VBUS_DET)	XTAL1/CLKIN	XTAL2	RBIAS

Table 3.2 SEC4410 Pins Grouped by Function (continued)

JTAG INTERFACE (5 PINS)			
JTAG_TMS	JTAG_TDO	JTAG_TDI	JTAG_TCK
JTAG_TRST			
MISC (9 PINS)			
RESET_N	CR_ACT/ LED_A0/ GPIO0	TEST	GPIO9/ CRD_PWR
UART_TX/ GPIO11	UART_RX/ GPIO12	GPIO13	SEL_PROC_TAP
NC			
DIGITAL, POWER (6 PINS)			
(5) VDD33	(1) VDD12PLL_MON		
TOTAL 64			
EXTRA PINS FOR 72-PIN PACKAGE			
SWV/TRACEDATA0	TRACEDATA1	TRACEDATA2	TRACEDATA3
TRACECLK	SEL25M_CLKDRIVE		

Note: Pads required for bond options are not listed in the above tables.

3.3 Pin Descriptions

Table 3.3 SEC2410 Pin Descriptions

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	**72-QFN**			
1	72	TEST	I	TEST Input This pin should be tied to ground for normal operation.
2	1	RESET_N	IS	RESET Input This active low signal is used by the system to reset the chip, where the active low pulse should be at least 1 µs wide.
3	2	GPIO3 (VBUS_DET)	I/O12	Detect Upstream VBUS Power Detects the state of the upstream VBUS power. The SMSC hub monitors **VBUS_DET** to determine when to assert the internal D+ pull-up resistor (signaling a connect event). When designing a detachable hub, this pin should be connected to VBUS on the upstream port via a 2:1 voltage divider. Two 100 kΩ resistors are suggested. For self-powered applications with a permanently attached host, this pin must be connected to 3.3 V (typically **VDD33**).
-	3	REG_EN	IPU	Regulator Enable This pin is internally pulled up to enable the internal 1.2 V regulators, and should be treated as a no-connect. In order to disable the regulators, this pin will need to be externally connected to ground. When the internal regulator is enabled, the 1.2 V power pins must be left unconnected, except for the required bypass capacitors.
4		USBDP	I/O-U	USB Data Plus Connect to the upstream USB bus data signals (host, port, or upstream hub).
5		USBDM	I/O-U	USB Data Minus Connect to the upstream USB bus data signals (host, port, or upstream hub).
6		SPI_CEN	I/O12	SPI Chip Enable Active low chip enable output. If the SPI interface is enabled, this pin must be driven high in power down states.
7		SPI_CLK/ GPIO4	I/O12	SPI Clock Out Clock signal out to the serial ROM. **Note:** During reset, this pin must be driven low.

Table 3.3 SEC2410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
8		SPI_DO/	I/O12	SPI Serial Data Out The output for the SPI port.
		(SPD_SEL)/ GPIO5		SPI Speed Select Selects the speed of the SPI interface. During **RESET_N** assertion, this pin will be tri-stated with the weak pull-down resistor enabled. When **RESET_N** is negated, the value on the pin will be internally latched, and the pin will revert to **SPI_DO** functionality, where the internal pull-down will be disabled. `0` : 30 MHz (no external resistor should be applied) `1` : 60 MHz (a 10 kΩ external pull-up resistor must be applied) If the latched value is 1, then the pin is tri-stated when the chip is in the suspend state. If the latched value is 0, then the pin is driven low during a suspend state.
9		SPI_DI/ GPIO2	I/O12PD	SPI Serial Data In The SPI data in to the controller from the ROM. This pin has a weak internal pull-down applied at all times to prevent floating.
10		CRFILT		+1.2 V Core Power Bypass This pin must have a 1.0 μF (or greater) ± 20% (ESR < 0.1 Ω) capacitor to VSS.
11		VDD33		3.3 V Power
12		JTAG_TMS	I	JTAG Mode Select The JTAG mode select to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.
13		JTAG_TDO	O12	JTAG Data Out The JTAG data out from the internal debug/test controller, which must be disabled during reset.
14		JTAG_TDI	I	JTAG Data In The JTAG data in to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.
15		JTAG_TCK	I	JTAG Clock The JTAG clock input to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.
16		JTAG_TRST	I	JTAG Reset The JTAG reset input to the internal debug/test controller, which must be tied low on the PCB when no debugger is used.
-	17	TRACEDATA3	O12	Trace Output Data Trace output data bit 3 from internal trace module when enabled.
-	18	TRACEDATA2	O12	Trace Output Data Trace output data bit 2 from internal trace module when enabled.

Table 3.3 SEC2410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
-	19	TRACEDATA1	O12	Trace Output Data Trace output data bit 1 from internal trace module when enabled.
17	20	SWV/	I/O12	Serial Wire Viewer Single wire output of the internal trace module, when enabled for single wire operation.
-		TRACEDATA0	O12	Trace Output Data Trace output data bit 0 from internal trace module when enabled.
18	21	CR_ACT/	I/O12	Card Reader Activity LED This pin can be configured to indicate card reader activity.
		LED_A0/ GPIO0		LED A0 This pin can be configured as a general purpose LED.
19	22	UART_RX/ GPIO12	I	UART Receive This is a 3.3 V receive signal for the internal UART. For RS232 operation, an external 12 V translator is required.
20	23	UART_TX/ GPIO11	O12	UART Transmit This is a 3.3 V transmit signal for the internal UART. For RS232 operation, an external 12 V driver is required.
21	24	GPIO7 (SD2_WP)	I/O12	SD2 Write Protect Detection The secure digital card mechanical write protect detection pin.
22	25	GPIO16 (SD2_nCD)	IPU	SD2 Card Detect The secure digital card detection pin.
23	26	SD2_D1/ GPIO19	I/O12PU	SD2 Data 1 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
24	27	SD2_D0/ GPIO18	I/O12PU	SD2 Data 0 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
25	28	SD2_D7/ GPIO25	I/O12PU	SD2 Data 7 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
26	29	SD2_D6/ GPIO24	I/O12PU	SD2 Data 6 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
27	30	SD2_CLK/ GPIO26	O12	SD2 Clock An output clock signal to SD2/MMC device, where the clock frequency is software configurable.
28	31	VDD33		3.3 V Power
29	32	GPIO9/ CRD_PWR	I/O200	Card Power 2 Drive The card power drive of 3.3 V at 200 mA.
30	33	SD2_D5/ GPIO23	I/O12PU	SD2 Data 5 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.

Table 3.3 SEC2410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
31	34	SD2_CMD/ GPIO17	I/O12PU	SD2 Command A bi-directional signal that connects to the CMD signal of the SD2/MMC device and has a weak internal pull-up resistor.
32	35	SD2_D4/ GPIO22	I/O12PU	SD2 Data 4 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
-	36	SEL25M_CLKDRIVE	I	Select 25 MHz Clock This pin selects the 25 MHz OSC as the clock source. `0` : clock source is 24 Hz XTAL `1` : clock source is 25MHz OSC on **XTAL1/CLKIN** (**XTAL2** is not used) When present on a package this pin MUST be tied high or tied low (direct connection 3.3 V or ground is acceptable). There is no internal pull-up or pull-down.
-	37	TRACECLK	O12	Debug Trace Clock The clock out of the ETM trace module when debugging is enabled.
33	38	SD2_D3/ GPIO21	I/O12PU	SD2 Data 3 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
34	39	SD2_D2/ GPIO20	I/O12PU	SD2 Data 2 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
35	40	VDD33		3.3 V Power
36	41	GPIO6 (SD1_WP)	I/O12	SD1 Write Protect Detection The secure digital card mechanical write protect detection pin.
37	42	GPIO15 (SD1_nCD)	IPU	SD1 Card Detect This secure digital card detection pin.
38	43	SD1_D1	I/O12PU	SD1 Data 1 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
39	44	SD1_D0	I/O12PU	SD1 Data 0 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
40	45	SD1_D7	I/O12PU	SD1 Data 7 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
41	46	SD1_D6	I/O12PU	SD1 Data 6 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
42	47	SD1_CLK	O12	SD1 Clock An output clock signal to SD1/MMC device, where the clock frequency is software configurable.
43	48	NC		No Connect
44	49	VDD33		3.3 V Power

Table 3.3 SEC2410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
45	50	SD1_D5	I/O12PU	SD1 Data 5 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
46	51	SD1_CMD	I/O12PU	SD1 Command A bi-directional signal that connects to the CMD signal of the SD1/MMC device and has a weak internal pull-up resistor.
47	52	SD1_D4	I/O12PU	SD1 Data 4 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
48	53	GPIO13	IPU	General Purpose I/O Pin
49	54	SD1_D3	I/O12PU	SD1 Data 3 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
50	55	SD1_D2	I/O12PU	SD1 Data 2 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
51	56	SEL_PROC_TAP	I	Processor Test Controller Select This pin must be tied low (direct connection to ground is acceptable) for normal operation. This pin should only be pulled high when debugging.
52	57	SC_ACT/	O12	SmartCard Active This signal indicates that the SmartCard is active and selects the 25 MHz OSC as the clock source. `0` : clock source is 24 Hz XTAL `1` : clock source is 25MHz OSC on **XTAL1/CLKIN** (**XTAL2** is not used) When present on a package this pin MUST be tied high or tied low (direct connection to ground or 3.3 V is acceptable). **Note:** There is no internal pull-up or pull-down.
		LED_B0/ GPIO1		LED B0 This pin can be configured as a general purpose LED.
53	58	GPIO14 (SC_PSNT_N)	IPU	SmartCard Insertion This pin is designated as the SmartCard card detection pin and can be left unconnected if the socket is not used.
54	59	SC_FCB/ GPIO29	O12	SmartCard Function Code This pin is used in conjuction with **SC_RST_N** for type 2 synchronous cards to indicate the type of command to be executed. This pin is held low while **SC_PSNT_N** is low and card power has not been applied, or while SmartCard block is deactivated.
55	60	SC_IO/ GPIO31	VIO12	SmartCard Bidirectional Serial Data The SmartCard bidirectional serial data pin should be held low when the interface is not active.

Table 3.3 SEC2410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
56	61	SC_CLK/ GPIO28	VIO12	SmartCard Clock Output This pin is the clock reference for communication with the flash media card. This pin should be held low when the interface is not active.
57	62	SC_SPU/ GPIO30	VIO12	SmartCard Standard or Proprietary Use This pin is held low while **SC_PSNT_N** is low and card power has not been applied, or while SmartCard block is deactivated.
58	63	SC_RST_N/ GPIO27	VIO12	SmartCard Reset Output A low pulse resets the card and triggers an ATR response message. This pin should be held low when the interface is not active.
59	64	VDD33		3.3 V Power
60	65	VAR_CRD_PWR/ GPIO10	I/O200	Variable Voltage Card Power: 1.8 V, 3.0 V, 3.3 V (200 mA) This pin should have a 1.0 μF Ceramic low ESR Capacitor when configured for output.
-	66	OTP_ATEST		This is a test pin and should be left un-connected for normal operation.
61	67	XTAL2	OCLKx	24 MHz Crystal This is the other terminal of the crystal, or can be left open when an external clock source is used to drive **XTAL1/CLKIN**.
62	68	XTAL1/ CLKIN	ICLKx	24 MHz Crystal (External Clock Input) This pin can be connected to one terminal of the crystal or can be connected to an external 24 MHz clock when a crystal is not used.
-	69	VDD12PLL_MON		This pin should be left unconnected - for testing only.
63	70	RBIAS	I-R	USB Transceiver Bias A 12.0 kΩ, ± 1.0% resistor is attached from VSS to this pin in order to set the transceiver's internal bias currents.
64	71	VDDA33		3.3 V Analog Power

Table 3.4 SEC4410 Pin Descriptions

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
1	72	TEST	I	TEST Input This pin should be tied to ground for normal operation.
2	1	RESET_N	IS	RESET Input This active low signal is used by the system to reset the chip, where the active low pulse should be at least 1 μs wide.

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
-	2	REG_EN	IPU	Regulator Enable This pin is internally pulled up to enable the internal 1.2 V regulators, and should be treated as a no-connect. In order to disable the regulators, this pin will need to be externally connected to ground. When the internal regulator is enabled, the 1.2 V power pins must be left unconnected, except for the required bypass capacitors.
3		VDD12A		1.2 V HSIC Reference Voltage In
4		HSIC_STROBE	I/O	HSIC Strobe Bi-directional data strobe signal defined in the *High-Speed Inter-Chip USB Specification, Version 1.0*.
5		HSIC_DATA	I/O	HSIC Data Bi-directional double data rate (DDR) data signal that is synchronous to the **HSIC_STROBE** signal as defined in the *High-Speed Inter-Chip USB Specification, Version 1.0*.
6		SPI_CEN	I/O12	SPI Chip Enable Active low chip enable output. If the SPI interface is enabled, this pin must be driven high in power down states.
7		SPI_CLK/ GPIO4	I/O12	SPI Clock Out Clock signal out to the serial ROM. **Note:** During reset, this pin must be driven low.
8		SPI_DO	I/O12	SPI Serial Data Out The output for the SPI port.
		(SPD_SEL)/ GPIO5		SPI Speed Select Selects the speed of the SPI interface. During **RESET_N** assertion, this pin will be tri-stated with the weak pull-down resistor enabled. When **RESET_N** is negated, the value on the pin will be internally latched, and the pin will revert to **SPI_DO** functionality, where the internal pull-down will be disabled. `0` : 30 MHz (no external resistor should be applied) `1` : 60 MHz (a 10 kΩ external pull-up resistor must be applied) If the latched value is 1, then the pin is tri-stated when the chip is in the suspend state. If the latched value is 0, then the pin is driven low during a suspend state.
9		SPI_DI/ GPIO2	I/O12PD	SPI Serial Data In The SPI data in to the controller from the ROM. This pin has a weak internal pull-down applied at all times to prevent floating.
10		CRFILT		VDD Core Regulator Filter Capacitor: this pin must have a 1.0 µF (or greater) ± 20% (ESR < 0.1 Ω) capacitor to VSS.
11		VDD33		3.3 V Power
12		JTAG_TMS	I	JTAG Mode Select The JTAG mode select to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
13		JTAG_TDO	O12	JTAG Data Out The JTAG data out from the internal debug/test controller, which must be disabled during reset.
14		JTAG_TDI	I	JTAG Data In The JTAG data in to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.
15		JTAG_TCK	I	JTAG Clock The JTAG clock input to the internal debug/test controller, which must have a pull-up resistor enabled during normal operation. The pull-up and the input must be disabled during reset.
16		JTAG_TRST	I	JTAG Reset The JTAG reset input to the internal debug/test controller, which must be tied low on the PCB when no debugger is used.
-	17	TRACEDATA3	O12	Trace Output Data Trace output data bit 3 from internal trace module when enabled.
-	18	TRACEDATA2	O12	Trace Output Data Trace output data bit 2 from internal trace module when enabled.
-	19	TRACEDATA1	O12	Trace Output Data Trace output data bit 1 from internal trace module when enabled.
17	20	SWV/	I/O12	Serial Wire Viewer Single wire output of the internal trace module, when enabled for single wire operation.
-		TRACEDATA0	O12	Trace Output Data Trace output data bit 0 from internal trace module when enabled.
18	21	CR_ACT/	I/O12	Card Reader Activity LED This pin can be configured to indicate card reader activity.
		LED_A0/ GPIO0		LED A0 This pin can be configured as a general purpose LED.
19	22	UART_RX/ GPIO12	I	UART Receive This is a 3.3 V receive signal for the internal UART. For RS232 operation, an external 12 V translator is required.
20	23	UART_TX/ GPIO11	O12	UART Transmit This is a 3.3 V transmit signal for the internal UART. For RS232 operation, an external 12 V driver is required.
21	24	GPIO7 (SD2_WP)	I/O12	SD2 Write Protect Detection The secure digital card mechanical write protect detection pin.
22	25	GPIO6 (SD2_nCD)	IPU	SD2 Card Detect The secure digital card detection pin.

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
23	26	SD2_D1/ GPIO19	I/O12PU	SD2 Data 1 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-up resistors.
24	27	SD2_D0/ GPIO18	I/O12PU	SD2 Data 0 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
25	28	SD2_D7/ GPIO25	I/O12PU	SD2 Data 7 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
26	29	SD2_D6/ GPIO24	I/O12PU	SD2 Data 6 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
27	30	SD2_CLK/ GPIO26	O12	SD2 Clock An output clock signal to SD2/MMC device, where the clock frequency is software configurable.
28	31	VDD33		3.3 V Power
29	32	GPIO9/ CRD_PWR	I/O200	Card Power 2 Drive The card power drive of 3.3 V at 200 mA.
30	33	SD2_D5/ GPIO23	I/O12PU	SD2 Data 5 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
31	34	SD2_CMD/ GPIO17	I/O12PU	SD2 Command A bi-directional signal that connects to the CMD signal of the SD2/MMC device and has a weak internal pull-up resistor.
32	35	SD2_D4/ GPIO22	I/O12PU	SD2 Data 4 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
-	36	SEL25M_CLKDRIVE	I	Select 25 MHz Clock This pin selects the 25 MHz OSC as the clock source. `0` : clock source is 24 Hz XTAL `1` : clock source is 25MHz OSC on **XTAL1/CLKIN** (**XTAL2** is not used) When present on a package this pin MUST be tied high or tied low (direct connection 3.3 V or ground is acceptable). There is no internal pull-up or pull-down.
-	37	TRACECLK	O12	Debug Trace Clock The clock out of the ETM trace module when debugging is enabled.
33	38	SD2_D3/ GPIO21	I/O12PU	SD2 Data 3 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
34	39	SD2_D2/ GPIO20	I/O12PU	SD2 Data 2 The **SD2_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
35	40	VDD33		3.3 V Power

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	**72-QFN**			
36	41	GPIO6 (SD1_WP)	I/O12	SD1 Write Protect Detection The secure digital card mechanical write protect detection pin.
37	42	GPIO15 (SD1_nCD)	IPU	SD1 Card Detect This secure digital card detection pin.
38	43	SD1_D1	I/O12PU	SD1 Data 1 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
39	44	SD1_D0	I/O12PU	SD1 Data 0 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
40	45	SD1_D7	I/O12PU	SD1 Data 7 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
41	46	SD1_D6	I/O12PU	SD1 Data 6 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
42	47	SD1_CLK	O12	SD1 Clock An output clock signal to SD1/MMC device, where the clock frequency is software configurable.
43	48	NC		No Connect
44	49	VDD33		3.3 V Power
45	50	SD1_D5/	I/O12PU	SD1 Data 5 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
46	51	SD1_CMD/	I/O12PU	SD1 Command A bi-directional signal that connects to the CMD signal of the SD1/MMC device and has a weak internal pull-up resistor.
47	52	SD1_D4/	I/O12PU	SD1 Data 4 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
48	53	GPIO13	IPU	General Purpose I/O Pin
49	54	SD1_D3	I/O12PU	SD1 Data 3 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
50	55	SD1_D2/	I/O12PU	SD1 Data 2 The **SD1_D[7:0]** pins are bi-directional data signals that have weak pull-down resistors.
51	56	SEL_PROC_TAP	I	Processor Test Controller Select This pin must be tied low (direct connection to ground is acceptable) for normal operation. This pin should only be pulled high when debugging.

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
52	57	SC_ACT/	O12	SmartCard Active This signal indicates that the SmartCard is active and selects the 25 MHz OSC as the clock source. `0` : clock source is 24 Hz XTAL `1` : clock source is 25MHz OSC on **XTAL1/CLKIN** (**XTAL2** is not used) When present on a package this pin MUST be tied high or tied low (direct connection to ground or 3.3 V is acceptable). **Note:** There is no internal pull-up or pull-down.
		LED_B0/ GPIO1		LED B0 This pin can be configured as a general purpose LED.
53	58	GPIO14 (SC_PSNT_N)	I/O12	SmartCard Insertion This pin is designated as the SmartCard card detection pin and can be left unconnected if the socket is not used.
54	59	SC_FCB/ GPIO29	VIO12	SmartCard Function Code This pin is used in conjuction with **SC_RST_N** for type 2 synchronous cards to indicate the type of command to be executed. This pin is held low while **SC_PSNT_N** is low and card power has not been applied, or while SmartCard block is deactivated.
55	60	SC_IO/ GPIO31	VIO12	SmartCard Bidirectional Serial Data The SmartCard bidirectional serial data pin should be held low when the interface is not active.
56	61	SC_CLK/ GPIO28	VIO12	SmartCard Clock Output This pin is the clock reference for communication with the flash media card. This pin should be held low when the interface is not active.
57	62	SC_SPU/ GPIO30	VIO12	SmartCard Standard or Proprietary Use This pin is held low while **SC_PSNT_N** is low and card power has not been applied, or while SmartCard block is deactivated.
58	63	SC_RST_N/ GPIO27	VIO12	SmartCard Reset Output A low pulse resets the card and triggers an ATR response message. This pin should be held low when the interface is not active.
59	64	VDD33		3.3 V Power
60	65	VAR_CRD_PWR/ GPIO10	I/O200	Variable voltage card power: 1.8V, 3.0V, 3.3V (100mA or 200mA) This pin should have a 1.0uF Ceramic low ESR Capacitor.
-	66	OTP_ATEST		This is a test pin and should be left un-connected for normal operation.
61	67	XTAL2	OCLKx	24 MHz Crystal This is the other terminal of the crystal, or can be left open when an external clock source is used to drive **XTAL1/CLKIN**.

Table 3.4 SEC4410 Pin Descriptions (continued)

PIN		NAME	BUFFER TYPE	DESCRIPTION
64-QFN	72-QFN			
62	68	XTAL1/ CLKIN	ICLKx	24 MHz Crystal (External Clock Input) This pin can be connected to one terminal of the crystal or can be connected to an external 24 MHz clock when a crystal is not used.
-	69	VDD12PLL_MON		This pin should be left unconnected - for testing only.
63	70	RBIAS	I-R	USB Transceiver Bias A 12.0 kΩ, ±1.0% resistor is attached from VSS to this pin in order to set the transceiver’s internal bias currents.
64	71	VDDA33		3.3 V Analog Power

3.4 Buffer Type Descriptions

Table 3.5 Buffer Type Descriptions

BUFFER	DESCRIPTION
I	Input
IPU	Input with internal weak pull-up resistor
I/O12	Input/output buffer with 12 mA sink and 12 mA source
VIO12	Variable voltage (1.8 V, 3.0 V, 3.3 V) input/output buffer with 12 mA sink and 12 mA source.
I/O200	Input/Output buffer 12 mA with FET disabled, 200 mA source only when the FET is enabled
I/O12PD	Input/output buffer with 12 mA sink and 12 mA source with an internal weak pull-down resistor
I/O12PU	Input/output buffer with 12 mA sink and 12 mA source with a pull-up resistor
O12	Output buffer with 12 mA source
ICLKx	XTAL clock input
OCLKx	XTAL clock output
I/O-U	Analog input/output as defined in the *USB 2.0 Specification* [1]
I-R	RBIAS

The DC characteristics are outlined in Section 25.3 on page 313.

Chapter 4 ARM M3 Embedded Controller

4.1 General Description

The Embedded Controller in SEC2410/SEC4410 is an ARM Cortex-M3 Processor by ARM Limited. The ARM M3 is a full-featured 32-bit embedded processor.

60 MHz clock is used for internal operation as well as all interfaces.

Processor core is a low gate count core, with low latency interrupt processing that features:

- A Thumb instruction set subset
- Banked *Stack Pointer* (SP) only.
- Hardware divide instructions, SDIV and UDIV (Thumb 32-bit instructions).
- Single cycle 32-bit multiply
- Handler and Thread modes.
- Three-stage pipeline

Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing. Features include:

- External interrupts of 1 to 240 configurable size.
- Bits of priority of 3 to 8 configurable size.
- Dynamic reprioritization of interrupts.
- Priority grouping. This enables selection of pre-empting interrupt levels and non pre-empting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
- Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.

Low cost debug solutions:

- Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is asserted.
- *Serial Wire Debug Port* (SW-DP) or *Serial Wire JTAG Debug Port* (SWJ-DP) debug access, or both.
- *Flash Patch and Breakpoint* (FPB) unit for implementing breakpoints and code patches.
- *Data Watchpoint and Trace* (DWT) unit for implementing watchpoints, data tracing, and system profiling.

Bus interfaces:

- *Advanced High-performance Bus-Lite* (AHB-Lite) ICode, DCode and System bus interfaces.
- *Private Peripheral Bus* (PPB) based on *Advanced Peripheral Bus* (APB) interface.
- Bit band support that includes atomic bit band write and read operations.
- Memory access alignment.
- Write buffer for buffering of write data.

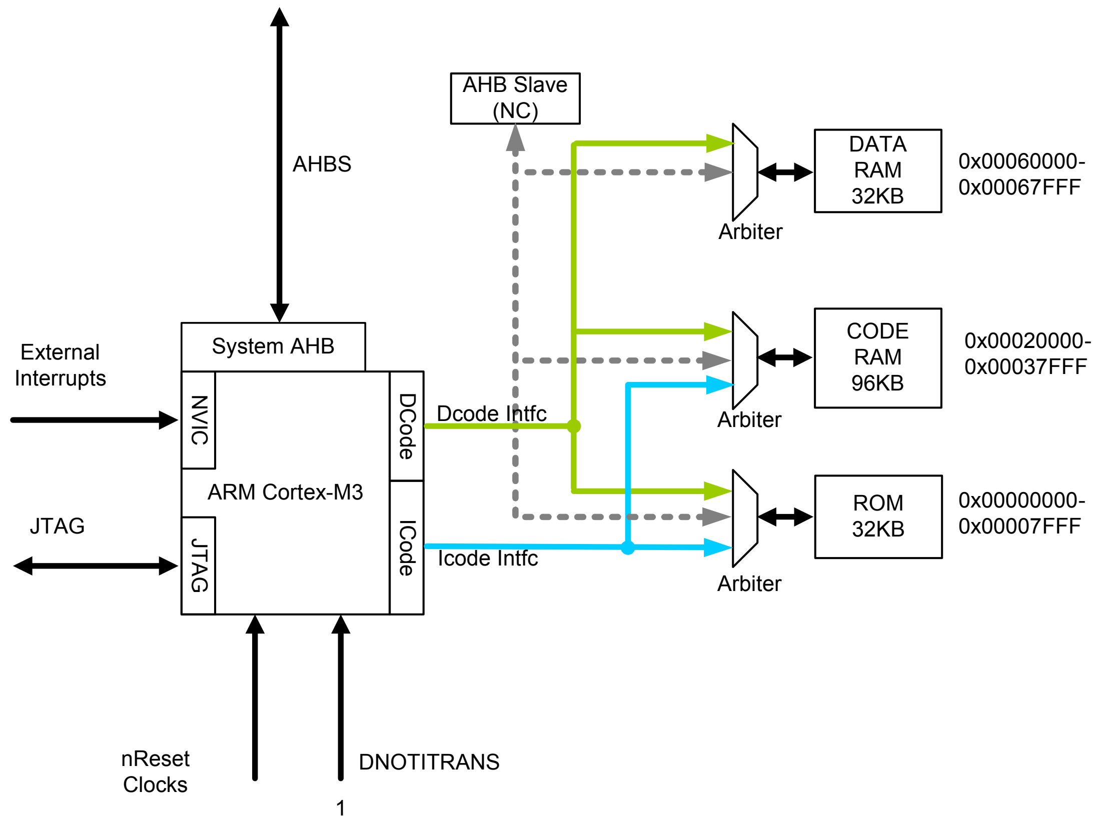


Figure 4.1 ARM Block Diagram

4.2 Sleep/Power Management

The ARM M3 supports a low power SLEEP mode in which internal pipelines do not change state and RAM is disabled which reduces power consumption. The ARM M3 is configured with clock gating support which can further limit power usage during SLEEP mode. Any non-masked interrupt or reset will bring the processor out of SLEEP mode.

4.3 Memory Controller

The Processor Memory Controller (PMC) has three buses, ICode, DCode, System AHB bus. The first two are AMBA 3.0, AHB-Lite bus compliant, and the system AHB bus is AMBA 2.0 compliant. All the three buses are configured in Little endian memory configuration, and use a single clock domain hclk.

The ICode bus is used for instruction fetches in the range of 0x00000000 to 0x1FFFFFFF, and is designed for compatibility to the Cortex-M3 processor DCode bus. But the valid Instruction addresses are limited by the ranges between MEMSUBSYS_ROM_ADDRH to MEMSUBSYS_ROM_ADDRL, and MEMSUBSYS_IRAM_ADDRH to MEMSUBSYS_IRAM_ADDRL parameters. The default in SEC2410/SEC4410 has 64KB of ROM. Any acceses to non-existent memory would cause a ERROR response.

The DCode bus is used for data and debug access in the same range from 0x00000000 to 0x1FFFFFFF, and is designed for compatibility to the Cortex-M3 processor DCode bus. But the valid data addresses are limited by the ranges between MEMSUBSYS_ROM_ADDRL to MEMSUBSYS_ROM_ADDRH, MEMSUBSYS_IRAM_ADDRL to MEMSUBSYS_IRAM_ADDRH and MEMSUBSYS_DRAM_ADDRL to MEMSUBSYS_DRAM_ADDRH parameters. The default in SEC2410/SEC4410 has 96KB of IRAM and 32KB SRAM. Any acceses to non-existent memory would cause a ERROR response.

The memory sub-system has an arbiter for each ROM or RAM. The typical use is most ICode instruction access occur to ROM or IRAM, and most data accesses are to DRAM. When code is downloaded to IRAM, or data such as literals are fetched from ROM or IRAM, there could be contention for access to these memories. The arbiter for each memory prioritizes the DCode accesses higher than ICode acceses and AHB bus accesses.

The optional AHB slave port may be connected in a system to CR_AHB/SYS_AHB bus (if this is bus matrix). This enables other AHB masters such as DMA controllers in the system to access processor memory complex. This access is controllable through programmable memory ranges for security.

The default memory map is shown below. All addresses here are from the processor perspective.

4.4 ARM-M3 AHB System Bus Interface

The ARM-M3 has a AHB-Lite System Bus Master interface (0x20000000 to 0xDFFFFFFF and 0xE0100000 - 0xFFFFFFFF; The ARM can perform 8-bit, 16-bit and 32-bit loads and stores on the System Bus. Instruction fetches over the System Bus are also supported

4.5 ARM-M3 Registers

4.6 Illegal Address Access

An access on the system AHB bus can generate an illegal address, if any of the following conditions are met:

1. The device or memory being accessed does not exist.
2. The device or memory being accessed is turned off.
3. The access takes more than 511 clock cycles.

If the processor accesses an illegal address the hardware must generate a complete bus error response on the system AHB bus.

4.7 ARM-M3 Interrupts

After RESET, the interrupt vector table will be located at 00000000h, which is in the Internal ROM.

Table 4.1 ARM-M3 Interrupt

NAME	ISR	VECTOR	PRIORITY	DESCRIPTION
Stack Pointer Top	0	00000000h		Stack Pointer loaded on reset
Reset	1	00000004h	-3	ARM-M3 Reset
NMI	2	00000008h	-2	Non maskable interrupt
Hard Fault	3	0000000Ch	-1	All classes of Fault, when the fault cannot activate because of priority or the Configurable Fault handler has been disabled. This is synchronous
Memory Management	4	00000010h	C	Unused
Bus Fault	5	00000014h	C	Pre-fetch fault, memory access fault, and other address/memory related. This is synchronous when precise and asynchronous when imprecise.
Usage Fault	6	00000018h	C	Usage fault, such as Undefined instruction executed or illegal state transition attempt. This is synchronous.
Reserved	7-10	0000001Ch-00000028H		Reserved
SVCall	11	0000002Ch	C	System service call with SVC instruction. This is synchronous.
Debug Monitor		00000030h	C	Debug monitor, when not halting. This is synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
Reserved		00000034h		
PendSV		00000038h	C	Pendable request for system service. This is asynchronous and only pended by software.
SysTick		0000003Ch	C	System tick timer has fired. This is asynchronous.
External Interrupt		00000040h-000003FCh	C	NVIC- Up to 240 interrupts

4.8 Processor Memory Controller (PMC)

The Processor Memory Controller (PMC) has three buses, ICode, DCode, System AHB bus. The first two are AMBA 3.0, AHB-Lite bus compliant, and the system AHB bus is AMBA 2.0 compliant. All the three buses are configured in Little endian memory configuration, and use a single clock domain hclk.

The ICode bus is used for instruction fetches in the range of 0x00000000 to 0x1FFFFFFF, and is designed for compatibility to the Cortex-M3 processor DCode bus. But the valid Instruction addresses are limited by the ranges between MEMSUBSYS_ROM_ADDRH to MEMSUBSYS_ROM_ADDRL, and MEMSUBSYS_IRAM_ADDRH to MEMSUBSYS_IRAM_ADDRL parameters. SEC2410/SEC4410 has 64KB of ROM. Any acceses to non-existent memory would cause a ERROR response.

The DCode bus is used for data and debug access in the same range from 0x00000000 to 0x1FFFFFFF, and is designed for compatibility to the Cortex-M3 processor DCode bus. But the valid data addresses are limited by the ranges between MEMSUBSYS_ROM_ADDRL to MEMSUBSYS_ROM_ADDRH, MEMSUBSYS_IRAM_ADDRL to MEMSUBSYS_IRAM_ADDRH and MEMSUBSYS_DRAM_ADDRL to MEMSUBSYS_DRAM_ADDRH parameters. SEC2410/SEC4410 has 96KB of IRAM and 32KB DRAM. Any acceses to non-existent memory would cause a ERROR response.

The memory sub-system has an arbiter for each ROM or RAM. The typical use is most ICode instruction access occur to ROM or IRAM, and most data accesses are to DRAM. When code is downloaded to IRAM, or data such as literals are fetched from ROM or IRAM, there could be contention for access to these memories. The arbiter for each memory prioritizes the DCode accesses higher than ICode acceses and AHB bus accesses.

The optional AHB slave port may be connected in a system to CR_AHB/SYS_AHB bus (if this is bus matrix). This enables other AHB masters such as DMA controllers in the system to access processor memory complex. This access is controllable through programmable memory ranges for security.

The memory map is shown below. All addresses here are from the processor perspective.

Table 4.2 PMC Memory Map

ADDRESS	DATA SPACE	BUS NAME
	ICODE/DCODE	
0x00000000~0x0001FFFF	Instruction ROM (128 KB, only 64KB used) ICode/DCode	MCU-ICODE, MCU-DCODE
0x00020000~0x0005FFFF	Instruction SRAM (256KB only 96KB used) ICode/DCode	MCU-ICODE, MCU-DCODE
0x00060000~0x0007FFFF	Data SRAM (128 KB, only 32KB used) DCode	MCU-DCODE
0x00080000~0x1FFFFFEF	Unused. Default slave returns zero.	MCU-ICODE, MCU-DCODE
0x1FFFFFE0~0x1FFFFFFF	Memory Sub-system Configuration	MCU-DCODE

4.9 Arbitration

The Processor Memory Complex (PMC) consists of the three memories, namely the ROM, IRAM, DRAM. The ICode bus is a read only bus which can access ROM, IROM. The DCode and AHB bus have read access to ROM, IRAM, DRAM, and write access to IRAM, DRAM.

Table 4.3 Memory Access Type

MEMORY	ICODE ACCESS	DCODE ACCESS	AHB ACCESS
ROM	R	R	R
IRAM	R	RW	RW
DRAM	None	RW	RW

The arbitration priority is fixed and is shown in table below. For each bus, the accesses are always in order of commands received. This module can be modifed for other arbitration schemes. Currently only fixed arbitration is supported. The highest priority is the DCode bus, since any wait states in the data bus access stall the processor pipeline. The next highest priority is the ICode bus. The last priority is the AHB bus.

Table 4.4 Memory Access Priority

PRIORITY	BUS ACCESS
ROM	
1	DCode (R)
2	ICode (R)
3	AHB (R)
IRAM	
1	DCode (RW)
2	ICode (R)
3	AHB (RW)
DRAM	
1	DCode (RW)
2	AHB (RW)

4.10 ICode Bus Interface

The ICode bus supports only IDLE/NSEQ type read accesses which are SINGLE and 32-bit wide. There are no bursts, and master BSY operation is not supported. The hproti protection information is not used.

Typically, the address from the ICode bus is registered by the ROM/IRAM at the rising edge of hclk, and the read data from the ROM/IRAM in the next clock is routed combinatorially to the ICode bus.

If there is no contention of access to IROM/IRAM, then back to back reads are executed in zero wait state.

The ARM Cortex-M3 processor matches the AMBA 3 specification except for maintaining control information during waited transfers. The processor might change the access type from SEQ or NONSEQ to IDLE during a waited transfer (hreadyi is low). In effect this cancels the outstanding transfer that has not yet occurred because the previous access is wait-stated and awaiting completion. This enables the processor to have a lower interrupt latency and higher performance in wait-stated systems. The ICode bus interface logic ignores such aborted outstanding reads.

4.11 DCode Bus Interface

The DCode bus supports IDLE/NSEQ/SEQ type accesses, which may be SINGLE or INCR bursts. The data width may be byte/Half-word/Word accesses. Master BSY operation and Locked accesses are not supported. The hprotd protection information is not used.

Typically, for reads, the address from the ICode bus is registered by the ROM/IRAM at the rising edge of hclk, and the read data from the ROM/IRAM in the next clock is routed combinatorially to the ICode bus. For writes, the address and control signals are registered and in the next clock, when write data is available, are routed to the IRAM/IROM.

If there is no contention of access to IROM/IRAM/DRAM, then back-to-back reads and back-to-back writes, irrespective of NSEQ or SEQ types, are executed in zero wait state. In case of writes after a read, both may be executed in zero wait state. In case of a read transaction after a write transaction, the read transaction would have one wait state.

The ARM Cortex-M3 processor matches the AMBA 3 specification except for maintaining control information during waited transfers. The processor might change the access type from SEQ or NONSEQ to IDLE during a waited transfer (hreadyi is low). In effect this cancels the outstanding transfer that has not yet occurred because the previous access is wait-stated and awaiting completion. This enables the processor to have a lower interrupt latency and higher performance in wait-stated systems. The DCode bus interface logic ignores such aborted outstanding reads/writes.

4.12 AHB Bus Interface

The AHB interace into the memory subsystem is stubbed off in SEC2410/SEC4410.

Chapter 5 ARM-M3 External Interrupts

5.1 General Description

The ARM M3 has a closely coupled *Nested Vectored Interrupt Controller* (NVIC) that supports up to 240 dynamically reprioritizable interrupts each with up to 256 levels of priority. The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts. The following is a list of the external interrupts used in SEC2410/SEC4410. The interrupt vectors for the external interrupts start at position 16 in the vector table

5.2 Interrupt Summary

Table 5.1 Interrupt List

IRQ NUMBER	NAME	DESCRIPTION
0	USB_RESET	USB Reset from UDC20
1	CLR_STALL	USB Clear Stall from UDC20
2	SET_CONF	USB Set Config from UDC20
3	SUSPEND	USB Suspend from UDC20
4	SET_INTF	USB Set Interface from UDC20
5	SETUP	USB Setup Packet from UDC20
6	RES_INTR_06	Reserved for future use
7	WAKEUP	System Wakeup
8	WQ_FIFO_INTR	WQ fifo interrupt
9	TIMER_3	Timer 3 overflow
10	TIMER_2	Timer 2 overflow
11	TIMER_1	Timer 1 overflow
12	TIMER_0	Timer 0 overflow
13	SMART_CARD_INT	Smart Card Interrupt
14	UART_INT	UART Interrupt
15	AES_UNALIGNED_ACCESS	AES Error Interrupt
16	AES_UNDER_RUN	AES Error Interrupt
17	SIE_EP3_WR	SIE Write Endpoint
18	SIE_EP3_RD	SIE Read Endpoint
19	SIE_EP2_WR	SIE Write Endpoint
20	SIE_EP2_RD	SIE Read Endpoint

Table 5.1 Interrupt List

IRQ NUMBER	NAME	DESCRIPTION
21	SIE_EP1_WR	SIE Write Endpoint
22	SIE_EP1_RD	SIE Read Endpoint
23	SIE_EP0_WR	SIE Write Endpoint
24	SIE_EP0_RD	SIE Read Endpoint
25	LUN_TIMEOUT	FMDU LUN Timeout interrupt
26	AUTO_CBW_INTR	Auto CBW processor interrupt
27	FMDU_EP0_WR_INTR	FMDU EP0 Write interrupt
28	FMDU_EP0_RD_INTR	FMDU EP0 Read interrupt
29	AES_FIFO_OVERFLOW	AES FIFO Overflow Interrupt
30	OTP_READY	OTP Ready
31	AES_RESIDUAL	AES Residual Interrupt
32	RES_INTR_32	Reserved for future use
39	SD1_CMD_ERR	From SDC1
40	SD1_DATA_ERR	From SDC1
41	SD1_CMD_TIMEOUT_ERR	From SDC1
42	SD1_SDC_CMD_RDY	From SDC1
43	SD1_SDIO_WAIT_INT	From SDC1
44	SD1_SDIO_INTR	From SDC1
45	SD1_SDC_RDY	From SDC1
46	SD2_CMD_ERR	From SDC2
47	SD2_DATA_ERR	From SDC2
48	SD2_CMD_TIMEOUT_ERR	From SDC2
49	SD2_SDC_CMD_RDY	From SDC2
50	SD2_SDIO_WAIT_INT	From SDC2
51	SD2_SDIO_INTR	From SDC2
52	SD2_SDC_RDY	From SDC2
53	RES_INTR_53	Reserved for future use
54	RES_INTR_54	Reserved for future use
55	GPIO_31	Shared with SC_IO
56	GPIO_30	Shared with SC_SPU
57	GPIO_29	Shared with SC_FSB
58	GPIO_28	Shared with SC_CLK

Table 5.1 Interrupt List

IRQ NUMBER	NAME	DESCRIPTION
59	GPIO_27	Shared with SD2_RST
60	GPIO_26	Shared with SD2_CLK
61	GPIO_25	Shared with SD2_D7
62	GPIO_24	Shared with SD2_D6
63	GPIO_23	Shared with SD2_D5
64	GPIO_22	Shared with SD2_D4
65	GPIO_21	Shared with SD2_D3
66	GPIO_20	Shared with SD2_D2
67	GPIO_19	Shared with SD2_D1
68	GPIO_18	Shared with SD2_D0
69	GPIO_17	Shared with SD2_CMD
70	GPIO_16	SD2_nCD
71	GPIO_15	SD1_nCD
72	GPIO_14	SC_PSNT_N
73	GPIO_13	Reserved
74	GPIO_12	Shared with UART_RX
75	GPIO_11	Shared with UART_TX
76	GPIO_10	Shared with VAR_CRD_PWR
77	GPIO_9	Shared with CRD_PWR2
78	GPIO_8	Shared with CRD_PWR1
79	GPIO_7	Shared with SD2_WP
80	GPIO_6	Shared with SD1_WP
81	GPIO_5	Shared with SPI_CLK
82	GPIO_4	Shared with SPI_DO
83	GPIO_3	Shared with VBUS Detect
84	GPIO_2	Shared with SPI_DI
85	GPIO_1	Shared with LED1
86	GPIO_0	Shared with LED0
87	RES_INTR_87	Reserved for future use
88	PHASE_ERROR	From FMDU
89	DATA_ERROR	From FMDU
90	SCSI_ERROR	From FMDU

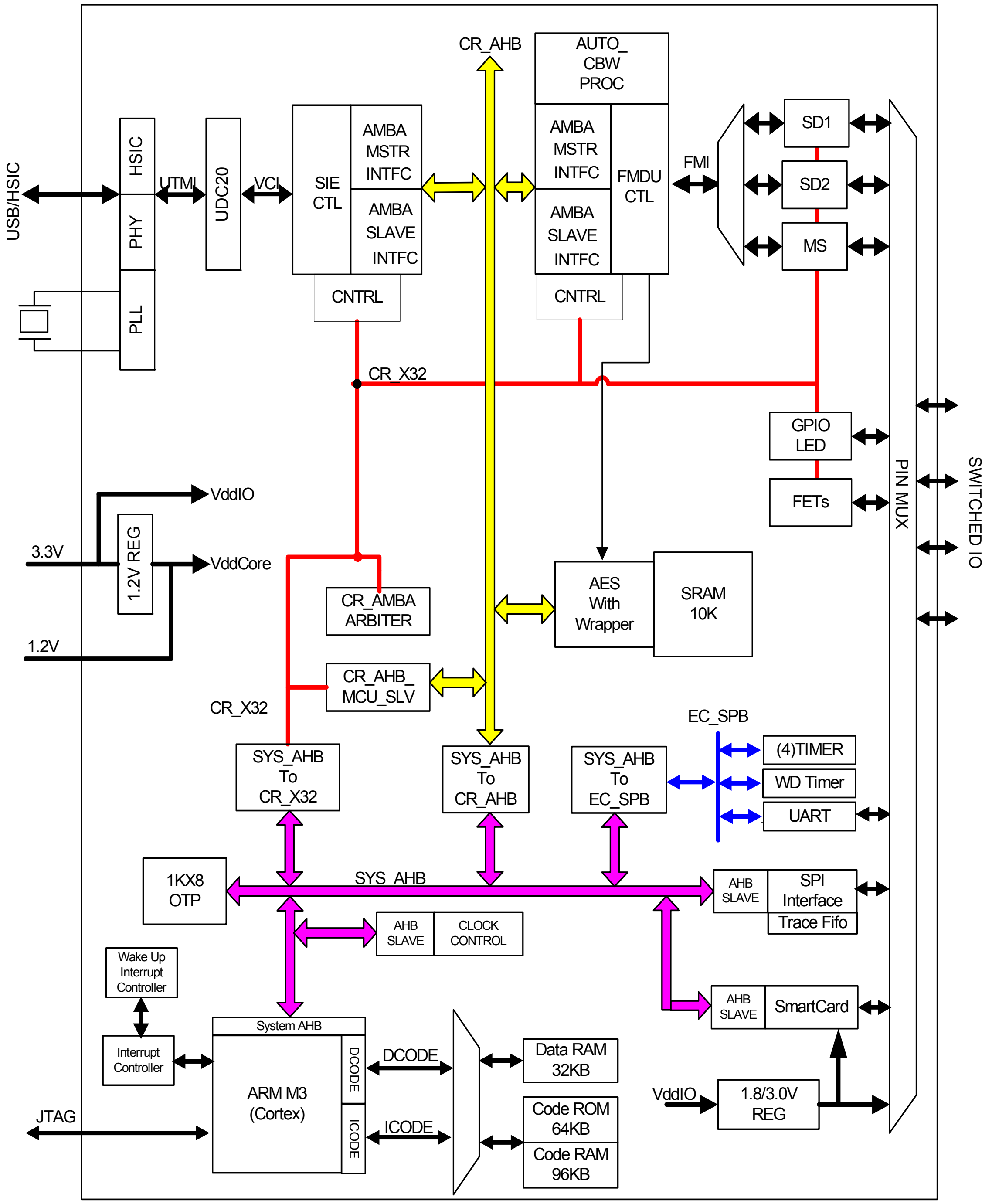
Table 5.1 Interrupt List

IRQ NUMBER	NAME	DESCRIPTION
91	MAX_LUN_FAIL	From FMDU
92	RSRV0_FAIL	From FMDU
93	CB_LEN_FAIL	From FMDU
94	WRITE_PROT	From FMDU
95	SIGNATURE_FAIL	From FMDU
96	RES_INTR_96	Reserved for future use
97	TAIL_REQUEST	From FMDU
98	UNSUPPORTED_LUN	From FMDU
99	CBWCB_LEN_FAIL	From FMDU
100	DEV_CAP_FAIL	From FMDU
101	CBW_IDLE	From FMDU
102	CBW_PKTLEN_FAIL	From FMDU
103	CBW_ONCE_INTR	From FMDU
104	RES_INTR_104	Reserved for future use
105	BLP_RECEIVE	From FMDU
106	XFER_DONE	From FMDU
107	RES_INTR_107	Reserved for future use
108	RES_INTR_108	Reserved for future use

Chapter 6 System AHB Bridges

6.1 Block Diagram

Diagram below illustrates the internal busses of the SEC2410/SEC4410.



6.2 System AHB to EC_SPB Bus Bridge

6.2.1 Overview

Timers, WD Timer and UART interface are connected to the EC_SPB bus. To access these peripherals, there is a System AHB to EC_SPB bus bridge. The bridge is one directional, and there is no way for devices on the EC_SPB bus to access devices on the system AHB bus.

6.2.2 EC_SPB Accesses

When the Processor does an access in the range of 0x040020000 to 0x4002FFFF, the access is forwarded to the EC_SPB bus. The Processor is wait stated appropriately until the AHB cycle completes. On the EC_SPB side the processor is the only master. The processor is wait stated appropriately till the AHB cycle completes. The processor is allowed to use 8, 16 and 32 bit accesses.

The EC_SPB address bus is 16 bits wide, and the upper address bits get truncated locally. The EC_SPB bus runs at 60Mhz

6.3 System AHB to Card Reader CR_X32 Bus Bridge

6.3.1 Overview

The Card Reader control registers are connected to the CR_X32 bus. To access these registers, there is a System AHB to Card Reader CR_X32 bus bridge. The bridge is one directional, and there is no way for devices on the CR_X32 bus to access devices on the system AHB bus directly.

6.3.2 CR_X32 Accesses

When the Processor does an access in the range of 0x040000000 to 0x4000FFFF, the access is forwarded to the CR_X32 bus. The Processor is wait stated appropriately until the AHB cycle completes. On the CR_X32 side the processor is the only master. The processor is wait stated appropriately till the AHB cycle completes. The processor is allowed to use 8, 16 and 32 bit accesses.

The CR_X32 address bus is 24 bits wide, and the upper address bits get truncated locally The data bus is 32 bits wide. The frequency is 60Mhz..

The CR_X32 bridge translates the little endian system AHB to the big endian card reader AHB. From the processor perspective, the card reader AHB bus appears as a little enidan bus.

6.4 System AHB to Card Reader AHB Bridge

6.4.1 Overview

The System AHB to Card Reader AHB bridge is a transparent bridge for the Processor to access the card reader AHB bus. There is also a slave interface to allow Card Reader AHB bus devices to send completion notifications back to the Processor. (CR_SHB_MCU_SLV)

6.4.2 CR_AHB Accesses

When the Processor does an access in the range of 0x20000000 to 0x200FFFFF, the access is forwarded to the Card Reader AHB bus. The processor is one of three bus masters on this bus, and has to wait for the arbiter to grant the bus. The Processor is wait stated appropriately until the AHB cycle completes. The processor is allowed to use 8, 16 and 32 bit accesses.

.The CR_AHB address bus is 20 bits wide, and the upper address bits get truncated locally. The data bus is 32 bits, and the operating frequency is 60Mhz.

6.4.3 Card Reader AHB Slave interface.

The System AHB to Card Reader AHB bridge is one directional. There is no way for devices on the CR_AHB bus to directly access devices on the system AHB bus. To facilitate communication in that direction, There is a AHB slave device, CR_AHB_MCU Slave, that sits on the CR_AHB bus. Anything written to that slave can be read on the System AHB bus.

The base address of the AHB slave interface is from the CR_AHB side is 0x00ED00. This is the address of the processor Work Queue. It is at this address that all devices on the CR_AHB write their Work Queue elements and Completion notification..

The Work Queues for the Processor is different from all other work queues. There is really only a single work queue at 0x00ED00. The Work Queue is 32 bits wide. The address of the WQ is aliased from 0x00ED00 to 0x00EDFF. AHB addresses A7 through A3 are mapped to D31 through D27. The format of the data written to the FIFO is given below.

If the processor wants to write to its own work queue, 0x50 must be pre-pended to the address.

Table 6.1 WQE/CN FIFO Format

BITS	NAME	DESCRIPTION
D31~D27	WQ ID	AHB Address A[7:3]
D27~D24	Status	Not Used Currently
D23~D8	Length	This length overrides the length in the work request. For SEC2410/SEC4410, only the lower 12 bits will be used.
D7	Last_buffer	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	Context[6:0]	This field contains the reference to the Work Request being used.

Table 6.2 Work Queue FIFO

AHB_WQ_FIFO_SLV (CR_AHB+ED00 RESET=0X00000000)			WQ FIFO SLAVE ADDRESS
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:0	CN_DATA	R	This is the Work Queue FIFO that is written to by the AMBA master on the CR_AHB bus when it wants to post a completion notification.

6.4.4 Work Queue FIFO Example:

Completion Notification to EP Read (CR_AHB+ED00), with Context 0x5, last = 1, Length = 0x777
Actual Data written into FIFO: 0x00077785

Given this value, the Processor can determine the Endpoint number. This way, the Processor can have 16 Read and 16 Write endpoints in a single FIFO. The FIFO is 16 deep. If the FIFO is full when another notification comes in, the write is suppressed, and the overflow bit is set.

6.5 Bridge Control Registers

These control registers appear on the CR_X32 bus.

Table 6.3 Work Queue FIFO

WQ_FIFO (CR_X32 + 0X0C30 RESET=0X00000000)			**PROCESSOR WORK QUEUE FIFO**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:0	WQ_DATA	R	This is the Work Queue FIFO that is read by the Processor to read the completion notifications posted to it. The FIFO read pointer is advance every time the most least byte (address 0) is read. To avoid endianess issues, read this fifo 32bits at a time only

Table 6.4 Work Queue FIFO Control

WQ_FIFO_CTL (CR_X32 + 0X0C34 - RESET=0X00)			**PROCESSOR WORK QUEUE FIFO CONTROL**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	OVERFLOW	R/W	This bit is set by the HW when a completion notification comes in, and the is full.
6	INT_ENABLE	R/W	If this bit is set, then a WQ_FIFO_INTR is generated to the processor if the FIFO_CNT is not zero.
5	Reserved	R	Always read ‘0’
4:0	FIFO_CNT	R	This is a count of the elements in the Work Queue FIFO.

Table 6.5 Bus Error Timer

BET_TIMER (CR_X32 + 0X0C44 - RESET=0XFF)			**BUS ERROR TIMER REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMEOUT	R/W	This register holds timeout value to generate a bus error. The value is bus clocks. The minimum value is 15. If a value of less than 15 is programmed, the value is automatically changed to 15.

Chapter 7 Memory Map

7.1 Bus Map Overview

The ARM has three external AHB-Lite buses. The ICode bus is used for instruction fetches in the range of 0x00000000 to 0x1FFFFFFF. The DCode bus is used for data and debug access in the same range from 0x00000000 to 0x1FFFFFFF. The System bus allows instruction fetch, data, and debug access to the ranges 0x20000000-0xDFFFFFFF and 0xE0100000-0xFFFFFFFF

The System bus coming out of the ARM is divided into 5 regions. The first is the CR_X32 bus which is a 32 bit version of the Xdata bus. This is done through a bridge which occupies a 64Kbyte address space 0x40000000~0x40000FFFF. All the card reader control registers reside in this space. The second region is CR_AHB which is the AHB bus in the card readers. This is done through another bridge which occupies a 16Mbyte address space (between 0x20000000~0x20FFFFFF. The third region is the EC_SPB which is also a 64Kbyte address space from 0x40020000-0x4002FFFF. The forth is the SPI region 0x60000000 to 0x60FFFFFF. Any access in this region results in access from the external SPI rom if present. The remaining region is used to decode for devices such as SmartCard, clock control, Data SRAM etc.

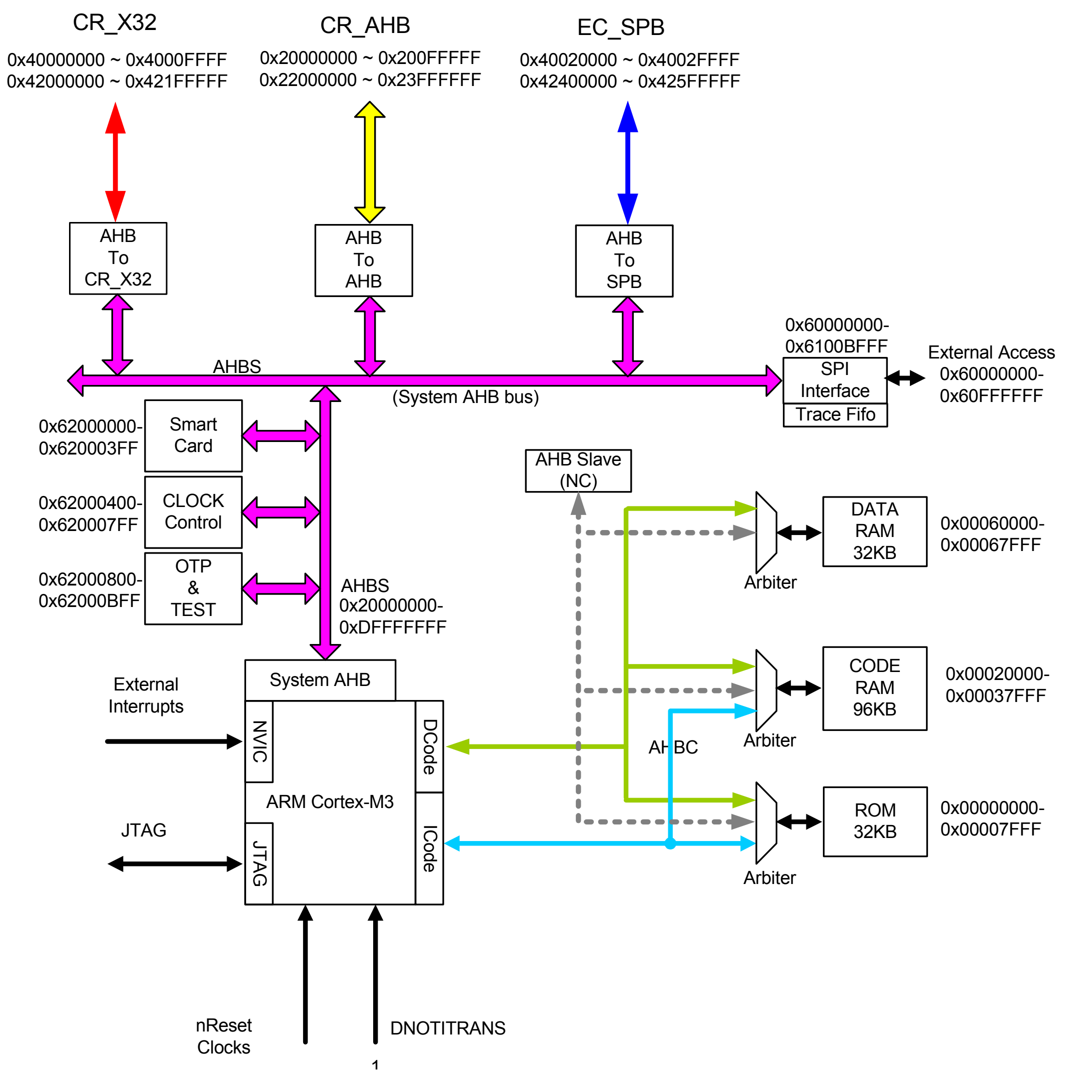


Figure 7.1 Bus Addresses

7.2 MCU Memory Map

All addresses here are from the processor perspective

Table 7.1 MCU Memory Map

ADDRESS	DATA SPACE	BUS NAME
	ICODE/DCODE	
0x00000000-0x0001FFFF	Instruction ROM (128 K, only 64K used) ICode/DCode	MCU-CODE
0x00020000-0x0004FFFF	Instruction SRAM (192K only 96K used) ICode/DCode	MCU-CODE
0x00060000-0x00067FFF	Data SRAM (32 KB)	MCU-CODE
0x00068000-0x1FFFFFFF	Unused	MCU-CODE
	CR_X32 (16 BIT BUS) BIT ALIASED AT 0X42000000	
0x40000800-0x40000BFF	MCU/GPIO/LED Registers	CR_X32
0x40000C00-0x40000FFF	CR_AHB_MCU Slave Registers	CR_X32
0x40001000-0x400013FF	FMDU Register	CR_X32
0x40001800-0x40001BFF	SD/MMC Registers	CR X32
0x40002800-0x40002BFF	SD2/MMC2 Registers	CR_X32
0x40002C00-0x40002FFF	SIE Registers	CR_X32
0x40003000-0x400033FF	AES Registers	CR X32
0x40003400-0x400037FF	AES Key Descriptors	CR X32
0x40007C00-0x40007FFF	PHY Registers	CR X32
	EC_SPB (16 BIT BUS)BIT ALIASED AT 0X42400000	
0x40020000-0x4002FFFF	EC SPB Peripherals (64K)	EC SPB
0x40020400-0x400207FF	WDT	EC SPB
0x40020C00-0x40020FFF	16 Bit Timers (4)	EC SPB
0x40024000-0x400243FF	UART (2)	EC SPB
	CR_AHB (24 BIT BUS) BIT ALIASED AT 0X22000000	
0x20004000-0x20007FFF 0x20008000-0x2000BFFF	CR AHB 10K SRAM (16K space) Passthru (R/W) CR AHB 10K Alias - Passthru (R), Encrypt/Decrypt (W)	CR AHB
0x2000F400~0x2000F7FF	CR AHB FMDU slave devices.	CR AHB
0x2000F000~0x2000F3FF	CR AHB SIE slave devices.	CR AHB
0x2000EC00~0x2000EFFF	CR AHB MCU slave devices.	CR AHB
	SYS_AHB (32 BIT BUS)	
0x60000000~0x60FFFFFF	External SPI ROM access	SYS_AHB
0x61000000~0x610003FF	SPI Control registers	SYS_AHB
0x6100BFFE~0x6100BFFF	Trace FIFO	SYS_AHB
0x62000000~0x620003FF	SmartCard	SYS_AHB
0x62000400~0x620007FF	Clock Control	SYS_AHB
0x62000800~0x62000AFF	Test Registers	SYS_AHB
0x62000B00~0x62000BFF	OTP	SYS_AHB

7.3 Bit-banding

The processor memory map includes two bit-band regions. These occupy the lowest 1MB of the SRAM and Peripheral memory regions respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

The memory map has two 32-MB alias regions that map to two 1-MB bit-band regions:

- Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region.
- Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region.

A mapping formula shows how to reference each word in the alias region to a corresponding bit, or target bit, in the bit-band region. The mapping formula is:

bit_word_offset = (byte_offset x 32) + (bit_number ´ 4)

bit_word_addr = bit_band_base + bit_word_offset

where:

- Bit_word_offset is the position of the target bit in the bit-band memory region.
- Bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- Bit_band_base is the starting address of the alias region.
- Byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- Bit_number is the bit position (0-7) of the targeted bit.

7.3.1 Directly accessing an alias region

Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

Bit [0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit [0] set writes a 1 to the bit-band bit, and writing a value with bit [0] cleared writes a 0 to the bit-band bit.

Bits [31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region returns either 0x01 or 0x00. A value of 0x01 indicates that the targeted bit in the bit-band region is set. A value of 0x00 indicates that the targeted bit is clear. Bits [31:1] are zero.

7.3.2 Directly accessing a bit-band region

You can directly access the bit-band region with normal reads and writes, and writes to that region.

7.3.3 Rom Table

A table of entries providing a mechanism to identify the debug infrastructure supported by the implemenation.

Table 7.2 ROM table

OFFSET	VALUE	NAME	DESCRIPTION
0xFD8	0x0	PID6	-
0xFDC	0x0	PID7	-
0xFE0	0x0	PID0	-
0xFE4	0x0	PID1	-
0xFE8	0x0	PID2	-
0xFEC	0x0	PID3	-
0xFF0	0x0D	CID0	-
0xFF4	0x10	CID1	-
0xFF8	0x05	CID2	-
0xFFC	0xB1	CID3	-

Table 7.3 ROM Table

OFFSET	VALUE	NAME	DESCRIPTION
0x000	0xFFF0F003	NVIC	Points to the NVIC at 0xE000E000.
0x004	0xFFF02002 or 003 if present	DWT	Points to the Data Watchpoint and Trace block at 0xE0001000. Value has bit [0] set if DWT is present.
0x008	0xFFF03002 or 003 if present	FPB	Points to the Flash Patch and Breakpoint block at 0xE0002000. Value has bit [0] set to 1 if FPB is present.
0x00C	0xFFF01002 or 003 if present	ITM	Points to the Instrumentation Trace block at 0xE0000000. Value has bit [0] set if ITM is present.
0x010	0xFFF41002 or 003 if present	TPIU	Points to the TPIU. Value has bit [0] set to 1 if TPIU is present. TPIU is at 0xE0040000.
0x014	0xFFF42002 or 003 if present	ETM	Points to the ETM. Value has bit [0] set to 1 if ETM is present. ETM is at 0xE0041000.
0x018	0	End	Marks the end of the ROM table. If CoreSight components are added, they are added starting from this location and the End marker is moved to the next location after the additional components.
0xFCC	0x1	MEMTYPE	Bits [31:1] RAZ. Bit [0] is set when the system memory map is accessible using the DAP. Bit [0] is clear when only debug resources are accessible using the DAP.
0xFD0	0x0	PID4	-
0xFD4	0x0	PID5	-
0xFD8	0x0	PID6	-
0xFDC	0x0	PID7	-
0xFE0	0x0	PID0	-
0xFE4	0x0	PID1	-
0xFE8	0x0	PID2	-
0xFEC	0x0	PID3	-
0xFF0	0x0D	CID0	-
0xFF4	0x10	CID1	-
0xFF8	0x05	CID2	-
0xFFC	0xB1	CID3	-

Chapter 8 GPIO and LED Interface

8.1 General Description

The SEC2410/SEC4410 GPIO Interface provides general purpose input monitoring and output control, as well as managing many aspects of pin functionality; including, multi-function Pin Multiplexing Control, Output Buffer Type control, PU/PD resistors, asynchronous wakeup and synchronous Interrupt Detection, GPIO Direction, pad current control, and Polarity control.

Features of the GPIO Interface include:

- Inputs:
  Asynchronous rising and falling edge wakeup detection
  Interrupt High or Low Level
  Can disable input (always reads ‘1’) to disable wakeup detection
- Pull up or pull down resistor control
- Interrupt and wake capability available for all GPIOs
- Multiplexing of all multi-function pins are controlled by the GPIO interface
- Debounce filter with individual programmable timer (10msec - 2.5sec)
- PAD Current Control

The registers in this block are accessable through CR_X32 bus.

8.2 Registers

8.2.1 LED Registers

The LEDs run off a common 50 mSec clock. This accuracy of the time setting is one clock time. The minimum and maximum times for each of the settings are as follows:

1. Blink Rate Minimum = (BLINK_RATE – 1) X 50 mSec
2. Blink Rate Maximum = BLINK_RATE X 50 mSec
3. Trail Time Minimum = (TRAIL_TIME – 1) X 50 mSec
4. Trail Time Minimum = TRAIL_TIME X 50 mSec

Setting the Blink rate to ‘0’ will cause the output to Oscillate and not be usable.

Table 8.1 LED0/GPIO0 Register

LED0_GPIO0_CTL (0X0806~0807- RESET=0X0000)			**LED0_GPIO0 CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15	XNOR	R/W	This bit toggles the polarity of the LED output. It can be used to invert the polarity, or used for the COM_MEDIA function.
14	XFER_TRIG	R/W	If this bit is set, XFER_ACTIVE in the Auto CBW processor will cause the LED to blink.
13:8	BLINK_RATE	R/W	Blink Rate of LED in 50 ms increments. Duty cycle of 50%. Rate range is 50 ms to 3.15 seconds
7:2	TRAIL_TIME	R/W	Time the LED must continue blinking after LED_ON is turned off. TRAIL_TIME is in 50ms increments. Range is 50 ms to 3.15 seconds
1	LED_ON	R/W	If LED then start blinking when this bit is one. Blink timer starts when this bit is enabled. No short blinks permitted. When this bit is disabled, blinking stops when TRAIL_TIME expires.
0	LED_GPIO	R/W	‘0’ = GPIO ‘1’ = LED

Table 8.2 LED1/GPIO1 Register

LED1_GPIO1_CTL (0X0808~0809- RESET=0X0000)			**LED1_GPIO1 CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15	XNOR	R/W	This bit toggles the polarity of the LED output. It can be used to invert the polarity, or used for the COM_MEDIA function.
14	XFER_TRIG	R/W	If this bit is set, XFER_ACTIVE in the Auto CBW processor will cause the LED to blink.
13:8	BLINK_RATE	R/W	Blink Rate of LED in 50 ms increments. Duty cycle of 50%. Rate range is 50 ms to 3.15 seconds
7:2	TRAIL_TIME	R/W	Time the LED must continue blinking after LED_ON is turned off. TRAIL_TIME is in 50ms increments. Range is 50 ms to 3.2 seconds
1	LED_ON	R/W	If LED then start blinking when this bit is one. Blink timer starts when this bit is enabled. No short blinks permitted. When this bit is disabled, blinking stops when TRAIL_TIME expires.
0	LED_GPIO	R/W	0’ = GPIO ‘1’ = LED.

Table 8.3 GPIO Output Enable Register

GPIO_OUT_EN (0X0830~0833- RESET= 0X00000000)			**GPIO OUTPUT ENABLE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO[31:0]	R/W	1= Output Enable 0 = Output Diable Note: While in Reset all GPIO outputs must be disabled.

Note: GPIO10 is input only in this device.

Table 8.4 GPIO Input Enable Register

GPIO_INP EN (0X0850~0853- RESET= 0X00000000)			**GPIO INPUT ENABLE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO[31:0]	R/W	1= Input Enable 0 = Input Diable Note: While in Reset all GPIO inputs must be disabled. That is the ENB_IN must be off on the GPIO PADs

Table 8.5 GPIO Output Register

GPIO_OUT (0X0834~0X0837 - RESET=0X00000000)			**GPIO DATA OUTPUT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO[31:0]	R/W	GPIO[31:2] Output Buffer Data

Note: GPIO10 is input only in this device.

Table 8.6 GPIO Input Register

GPIO_IN (0X0838~0X083B- RESET=0X00000000)			**GPIO INPUT 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO[31:0]	R	GPIO[31:0] Input Buffer Data Note: While in Reset all GPIO inputs must be disabled. That is the ENB_IN must be off on the GPIO PADs

Table 8.7 GPIO Pull-up Resistor Register

GPIO_PU (0X083C~0X083F- RESET=0X00000000)			**GPIO PULL UP REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO_PU[31:0]	R/W	“0” = Disables the pull-up resistor on the GPIO pad. “1” = Enables the pull-up resistor on the GPIO pad.

Table 8.8 GPIO Pull-down Register

GPIO_PD (0X082C~0X082F- RESET=0X00000000)			**GPIO PULL DOWN REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO_PD[31:0]	R/W	“0” = Disables the pull-down resistor on the GPIO pad. “1” = Enables the pull-down resistor on the GPIO pad.

8.2.2 GPIO interrupts for NVIC

An interrupt is generated in the NVIC if a GPIO transitions from a low to a high and the corresponding bit in the GPIO_INTR_HI_MSK is cleared, or if the GPIO transitions from a high to a low and the corresponding bit in the GPIO_INTR_LO_MSK bit is cleared. The NVIC may additionally mask the interrupt internally.

Table 8.9 GPIO Interrupt High Mask Register

GPIO_INTR_HI_MSK (0X0824~0X0827- RESET=0XFFFFFFFF)			**GPIO INTERRUPT HIGH MASK REGISTER**
BIT	NAME	R/W	DESCRIPTION
31:0	GPIO_HI_MSK[31:0]	R/W	“1” = Prevents an interrupt from being generated on a low to high transition of the corresponding GPIO line at the NVIC.

Table 8.10 GPIO Interrupt Low Mask Register

GPIO_INTR_LO_MSK (0X0828~0X082B- RESET=0XFFFFFFFF)			**GPIO INTERRUPT LOW MASK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	GPIO_LO_MSK[31:0]	R/W	“1” = Prevents an interrupt from being generated on a high to low transition of the corresponding GPIO line at the NVIC

Table 8.11 Debounce Control Register

GPIO_DEBOUNCE_EN (0X0840- RESET=0X00)			**SPECIAL FUNCTION REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	GPIO15_DEBOUNCE	R/W	If set, and if GPIO15 is configured as an input, then GPIO15 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
6	GPIO14_DEBOUNCE	R/W	If set, and if GPIO14 is configured as an input, then GPIO14 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
5	GPIO13_DEBOUNCE	R/W	If set, and if GPIO13 is configured as an input, then GPIO13 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
4	GPIO12_DEBOUNCE	R/W	If set, and if GPIO12 is configured as an input, then GPIO12 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
3	GPIO5_DEBOUNCE	R/W	If set, and if GPIO5 is configured as an input, then GPIO5 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
2	GPIO3_DEBOUNCE	R/W	If set, and if GPIO3 is configured as an input, then GPIO3 input is debounced by the amount specified in the GPIO_DEBOUNCE register.
1:0	Reserved	R/W	Always read ‘0’

Table 8.12 GPIO Debounce Registe2

GPIO_DEBOUNCE2 _EN (0X0841- RESET=0X00)			**GPIO DEBOUNCE REGISTER2**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	GPIO16_DEBOUNCE	R/W	If set, and if GPIO16 is configured as an input, then GPIO16 input is debounced by the amount specified in the GPIO_DEBOUNCE register.

Table 8.13 GPIO Debounce Register

GPIO_DEBOUNCE (0X080D – RESET=0X0A)			**DEBOUNCE REGISTER FOR GPIO**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DEBOUNCE	R/W	This register holds the debounce timer for the card detect signal. The first transition is allowed through, any transition within the debounce period is suppressed. Each count corresponds to 10 mS, with the default value being 100 mS.

Table 8.14 Utility Configuration Register 1

UTIL_CONFIG1 (0X080A RESET=0X00)			**UTILITY CONFIGURATION REGISTER 1**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always read ‘0’
2	SPI_DISABLE	R/W	After reset the SPI interface is always enabled. If the firmware does not detect a SPI rom externally, it sets this bit. This disable the SPI interface, and enables GPIOs.
1	UART_ENABLE	R/W	This bit enables the UART pins. 0 - GPIOs go to GPIO pins 1 - The UART TX and RX go the GPIO11,12.
0	Reserved	R	Always read ‘0’

Table 8.15 PIN MUX select Register

PIN_MUX_SEL (0X080C RESET=0X00)			**PIN MUX SELECT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	SC_LED_SEL	R/W	Pin LED0 / SC_LED_ACT_N function selected with this bit: ‘0’ = LED controlled from GPIO0 / LED0 ‘1’ = LED controlled by Smart Card interface Note: Please see LED0_GPIO0_CTL for details of using gpio versus LED
6	SD2_GPIO_EN	R/W	Setting this bit enables the GPIOs that are multiplexed with the second SecureDigital interface: GPIO17, GPIO18, GPIO19, GPIO20, GPIO21, GPIO22, GPIO23, GPIO24, GPIO25, GPIO26 Note: GPIO16, is always a GPIO, and is not affected by this bit.
5	SD2_HI_GPIO EN	R/W	Setting this bit enables the four GPIOs that are multiplexed with the upper four data bits of the second SecureDigital interface: GPIO22, GPIO23, GPIO24, GPIO25 This bit has no effect if SD2_GPIO_EN is set.
4	SC_GPIO_EN	R/W	Setting this bit enables the five GPIOs that are multiplexed with the SmartCard interface: GPIO27, GPIO28, GPIO29, GPIO30, GPIO31
3:0	MUX_SEL	R/W	The single Flash Media is selected with the following values: 0001 Reserved 0010 Reserved 0100 Reserved 1000 Secure Digital

8.2.3 GPIO Muxing Table

Table 8.16 GPIO Muxing Table

NAME	HARDWARE MUX	MUX CONTROL BIT	SW GPIO FUNCTION	COMMENT
GPIO0	CR_ACTIVTY/ LED0	LED0_GPIO_CTL	Unassigned	
GPIO1	SC_LED_ACT_N/ LED1	LED1_GPIO_CTL & LED_CTL	Unassigned	
GPIO2	SPI_DI	SPI_DISABLE	Unassigned	
GPIO3	-	NA	VBUS	Debounced
GPIO4	SPI_DO	SPI_DISABLE	I2C Data	
GPIO5	SPI_CLK		I2C Clock	
GPIO6	-	N/A	SD1_WP	
GPIO7	-	N/A	SD2_WP	
GPIO8	CRD_PWR1	FET_CTL	FET Overcurrent	FET supplies power, GPIO is used as over current sense.
GPIO9	CRD_PWR2	FET_CTL	FET Overcurrent	FET supplies power, GPIO is used as over current sense.
GPIO10	VAR_CRD_PWR	VREG_CTL	SC Overcurrent	Regulator supplies power, GPIO is used as over current sense.
GPIO11	UART_TX	UART_ENABLE	Unassigned	
GPIO12	UART-RX	UART_ENABLE	Unassigned	Debounced
GPIO13	-	N/A	Unassigned	Debounced
GPIO14	-	N/A	SC_PSNT	Debounced
GPIO15	-	N/A	SD1_nCD	Debounced
GPIO16	-	N/A	SD2_NCD	Debounced
GPIO17	SD2_CMD	SD2_GPIO_EN	Unassigned	
GPIO18	SD2_D0		Unassigned	
GPIO19	SD2_D1		Unassigned	
GPIO20	SD2_D2		Unassigned	
GPIO21	SD2_D3		Unassigned	
GPIO22	SD2_D4	SD2_HI_GPIO EN or SD2_GPIO_EN	Unassigned	
GPIO23	SD2_D5		Unassigned	
GPIO24	SD2_D6		Unassigned	
GPIO25	SD2_D7		Unassigned	

Table 8.16 GPIO Muxing Table

NAME	HARDWARE MUX	MUX CONTROL BIT	SW GPIO FUNCTION	COMMENT
GPIO26	SD2_CLK	SD2_GPIO_EN	Unassigned	
GPIO27	SC_RST	SC_GPIO_EN	Unassigned	
GPIO28	SC_CLK		Unassigned	
GPIO29	SC_FSB		Unassigned	
GPIO30	SC_SPU		Unassigned	
GPIO31	SC_IO		Unassigned	

8.2.4 Identification Registers

To read the BOND_OPT register:

1. Write a 0x80 to the register to enable the pull-ups.
2. Wait (at least) 1µSec for the pull-ups to take effect.
3. Read the register.

Write a 0x00 to the register to disable the pull-ups.

Table 8.17 : Device Revision Register

DEV_REV (0X0800- RESET=0X00)			DEVICE REVISION REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
[7:0]	REV	R	This register defines additional revision information used internally by SMSC. The value is silicon revision dependent.

Table 8.18 Device Identification Register

DEV_ID (0X0801- RESET=0X80)			DEVICE IDENTIFICATION REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
[7:0]	ID	R	This register defines additional revision information used internally by SMSC

Table 8.19 Bond Option Register

BOND_OPT (0X0802- RESET=0B0XXX_XXX)			BOND OPTION REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	OPT_PU_EN	R/W	Enables the pull-up resistors for the Bond Options. This bit must be set to a ‘1’ before reading the value in other register bits (PKG_TYPE, OPT,). This bit must be set back to ‘0’ after the read is completed to avoid unnecessary power dissipation.
6	HSIC_EN	R	This bit indicates that the HSIC interface has been enabled.
5:4	PKG_TYPE	R	The value is an encoding of the package type: 00: Reserved 01: 64/72 pin QFN 10: Reserved 11: Reserved

3:1	OPT	R	OPT defines bond option information used internally by SMSC. The value is defined as: 000: Opt 0 001: Opt 1 010: Opt 2 011: Opt 3 100: Opt 4 101: Opt 5 110: Opt 6 111: Opt 7
0	OPT0	R	TBD

8.2.5 Card Power operation Fixed voltage

On reset, all FET control registers default to CRD_PWR mode.

When operating in CRD_PWR mode:

1. The corresponding GPIO will be set to IN.
2. If the FET is off, the GPIO input is disabled.
3. If the FET is on, the GPIO input is enabled.
4. There is no pull-up or pull-down applied (unless expicitly enabled by FW)

When using an external FET, it is a requirement that a P-channel FET only be used, with an external pull-up.

When using the internal FET the associated GPIO is automatically turned to an input. The sequence for turning on a FET is the following:

1. Turn on the FET
2. Wait the power on time
3. Check that the GPIO is a high:
4. If the GPIO is a high, turn on interface, if low, turn off. – error.
5. Enable the GPIO to interrupt on high to low transition – This will cause an interrupt if the FET is shorted.

Table 8.20 FET to GPIO mapping

Control Register	Reg Address	Pin Name	Corresponding GPIO
FET_CTL1	0x080E	CRD_PWR1	GPIO8
FET_CTL2	0x080F	CRD_PWR2	GPIO9
VREG_CTL	0x0811	VAR_CRD_PWR	GPIO10

Table 8.21 FET control Register 1

FET_CTL1 (0X080E RESET=0X20)			**FET CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5	GPIO/CRD_PWR	R/W	GPIO/CRD_PWR select mux: 0 = GPIO functionality is enabled, the CRD_PWR_EN bit register is forced to ‘0’ 1 = CRD_PWR functionality is enabled, the corresponging GPIO bit is forced output disable and input enabled..
4	CRD_PWR_EN	R/W	Card Power Enable for FET 0 = Current Source is OFF/Disabled 1 = Current Source is ON/Enabled. If the GPIO/CRD_PWR bit = ‘0’, then this register is forced to zero and all writes will be ignored.
3:0	FET_MODE	R/W	0000 = 200 mA Operation (default) All other values reserved

Table 8.22 FET control Register 2

FET_CTL2 (0X080F RESET=0X20)			**FET CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5	GPIO/CRD_PWR	R/W	GPIO/CRD_PWR select mux: 0 = GPIO functionality is enabled, the CRD_PWR_EN bit register is forced to ‘0’ 1 = CRD_PWR functionality is enabled, the corresponging GPIO bit is to forced output disable, input enable.
4	CRD_PWR_EN	R/W	Card Power Enable for FET 0 = Current Source is OFF/Disabled 1 = Current Source is ON/Enabled. If the GPIO/CRD_PWR bit = ‘0’, then this register is forced to zero and all writes will be ignored.
3:0	FET_MODE	R/W	0000 = 200 mA Operation (default) All other values reserved

8.2.6 Card Power operation Variable voltage

On reset, the regulator control registers default to CRD_PWR mode.

When operating in CRD_PWR mode:

1. The corresponding GPIO will be set to IN.
2. If the Regulator is off, the GPIO input is disabled.
3. If the Regulator is on, the GPIO input is enabled.
4. There is no pull-up or pull-down applied (unless expicitly enabled by FW)

When using the internal Regulator the associated GPIO is automatically turned to an input. The sequence for turning on a Regulator is the following:

1. Set the desired voltage
2. Turn on the FET
3. Wait the power on time
4. Check that the GPIO is a high:
5. If the GPIO is a high:
   - Tune the pads by setting the TUNE bit for > 0.5uS
   - Clear TUNE bit when tuning is complete
   - Turn on interface
6. If GPIO is low, turn off. – error.
7. Enable the GPIO to interrupt on high to low transition – This will cause an interrupt if the FET is shorted.

Table 8.23 Voltage Regulator control Register 1

VREG_CTL (0X0811 RESET=0X00)			**VOLTAGE REGULATOR CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	R	Always read ‘0’
6	VOLTAGE_SEL	R/W	0 = 3.0V 1 = 1.8V
5	TUNE	R/W	Set this bit to enable tuning of the variable voltage circuitry. Tuning the circuitry requires this bit to be on for at least 0.5uS
4	CRD_PWR_EN	R/W	Card Power Enable for FET 0 = Current Source is OFF/Disabled 1 = Current Source is ON/Enabled. If the regulator is enabled, the output of any GPIO sharing this pin must be disabled.
3:0	Reserved	R	Always read ‘0’

8.2.7 GPIO Pad Current Control

The GPIO current control is divided intro two Registers. The first registers control GPIO 0 to 26, which are GPIOs that are powered off VDD33. The second register controls GPIO 27~31 which are GPIOs powered of VAR_CRD_PWR.

Table 8.24 GPIO PAD Current Control 1 Register

PAD_CTL_GPIO_0_26 (0X0870 - RESET=0X00)			**PAD CURRENT CONTROL GPIO 0 TO 26**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.
1:0	SEL	R/W	00 = 6 mA Operation (default) 01 = 8 mA Operation 10 = 10 mA Operation 11 = 12 mA Operation **Note:** This register only has effect on active GPIOs.

Table 8.25 GPIO PAD Current Control 1 Register

PAD_CTL_GPIO_27_31 (0X0874 - RESET=0X00)			**PAD CURRENT CONTROL GPIO 27 TO 31**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.
1:0	SEL	R/W	00 = 6 mA Operation (default) 01 = 8 mA Operation 10 = 10 mA Operation 11 = 12 mA Operation **Note:** This register only has effect when GPIOS are enabled on SmartCard interface **Note:** These values subject to change

Table 8.26 Trace Data PAD Current Control 1 Register

PAD_CTL_TRACE (0X0878- RESET=0X00)			**TRACE DATA CURRENT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.

PAD_CTL_TRACE (0X0878- RESET=0X00)			**TRACE DATA CURRENT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
1:0	SEL	R/W	00 = 6 mA Operation (default) 01 = 8 mA Operation 10 = 10 mA Operation 11 = 12 mA Operation **Note:** This register affects TDO, TRACEDATA[3:0], and TRACECLK outputs **Note:** These registers also control the output of the UART TX line

Table 8.27 ETM Contol Register

ETM_CTL (0X0880- RESET=0X00)			**ETM CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	ARM_DBG_ENABLED	R	This bit reflects the state of the SEL_PROC_TAP 1 = Debugger is potentially connected to system 0 = Debugger is not connected.
6:3	Reserved	R	Always read “0”.
2	ETMEXTIN_1	R/W	General purpose triggers to the ETM. Can be used to generate start and stops in the trace capture.
1	ETMEXTIN_0	R/W	When set, a processor LOCKUP signal will generate a trigger to the ETM
0	ETMFIFOFULLEN	R/W	Setting this bit will cause the processor to be stalled whenever the ETM FIFO is full. This can cause unacceptable behavior of the part, but allows for 100% trace capability. Use for development only.

8.2.8 GPIO connection to timers

The GPIOs can be used as inputs to the timers. Any one of the 32 GPIOs can go to any one of the the timers.

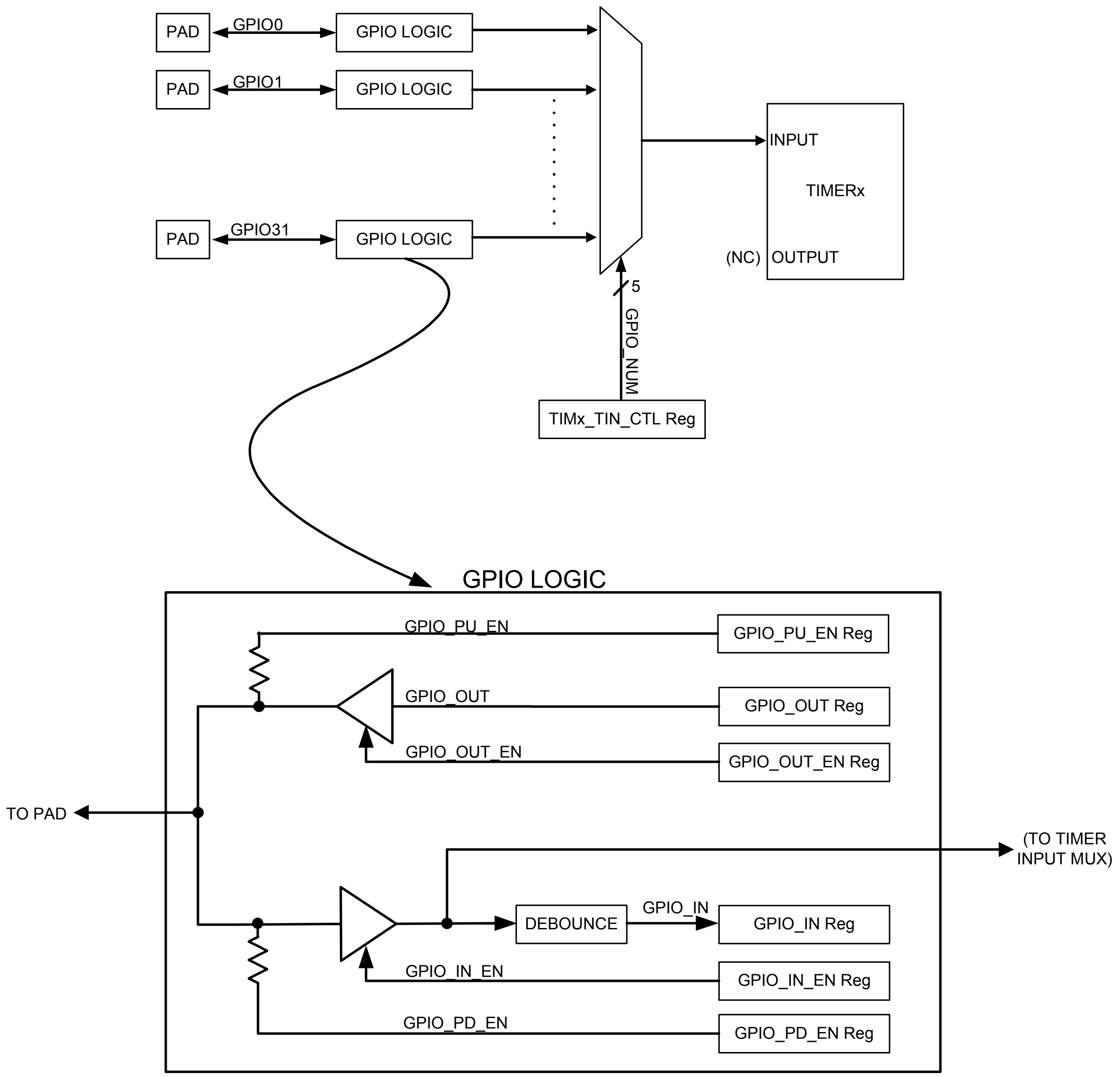


Figure 8.1 GPIO to Timer Interconnect

Table 8.28 Timer 0 GPIO Input Contol Register

TM0_TIN_CTL (0X0884- RESET=0X00)			**TIMER 0 INPUT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read “0”.
4:0	GPIO_NUM	R/W	Select specified GPIO (31-0) as timer's input.

Table 8.29 Timer 1 GPIO Input Contol Register

TM1_TIN_CTL (0X0885- RESET=0X00)			**TIMER 1 INPUT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read “0”.
4:0	GPIO_NUM	R/W	Select specified GPIO (31-0) as timer's input.

Table 8.30 Timer 2 GPIO Input Contol Register

TM2_TIN_CTL (0X0886- RESET=0X00)			**TIMER 2 INPUT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read “0”.
4:0	GPIO_NUM	R/W	Select specified GPIO (31-0) as timer's input.

Table 8.31 Timer 3 GPIO Input Contol Register

TM3_TIN_CTL (0X0887- RESET=0X00)			**TIMER 3 INPUT CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read “0”.
4:0	GPIO_NUM	R/W	Select specified GPIO (31-0) as timer's input.

Chapter 9 Two Pin Serial Port (UART)

9.1 General Description

The SEC2410/SEC4410 block incorporates an Inventra™ M16550S core, providing a fully programmable, universal asynchronous receiver/transmitter (UART) that is functionally compatible with the NS 16550AF device and with the established Inventra™ M16550A core.The UART is compatible with the 16450, the 16450 ACE registers and the 16C550A. The UART performs serial-to-parallel conversion on received characters and parallel-to-serial conversion on transmit characters. Two sets of baud rates are provided. The first is 24Mhz, and the second is 16MHz. When the 24Mhz MHz source clock is selected, standard baud rates from 50 to 115.2K are available. When the source clock is 16 MHz, baud rates from 125K to 1,000K are available. The character options are programmable for the transmission of data in word lengths of from five to eight, 1 start bit; 1, 1.5 or 2 stop bits; even, odd, sticky or no parity; and prioritized interrupts. The UART contains a programmable baud rate generator that is capable of dividing the input clock or crystal by a number from 1 to 65535. The UART is also capable of supporting the MIDI data rate. Refer to the Configuration Registers for information on disabling, powerdown and changing the base address of the UART. The interrupt from a UART is enabled by programming OUT2 of the UART to a logic “1”. OUT2 being a logic “0” disables that UART's interrupt.

9.1.1 Features

- Programmable word length, stop bits and parity
- Programmable baud rate generator
- Interrupt generator
- Loop-back mode
- Interface registers
- 16-byte Transmit FIFO
- 16-byte Receive FIFO
- Multiple clock sources
- Pin Polarity control
- Low power sleep mode

9.1.2 Block Diagram

General block diagram of the UART is given below:

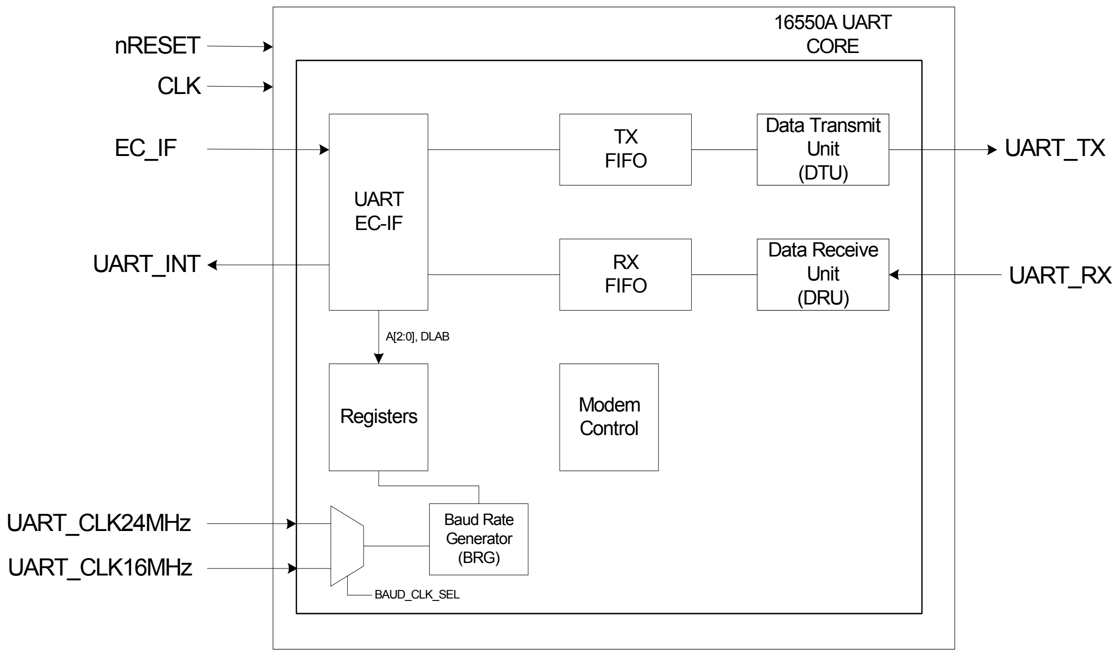


Figure 9.1 Serial Port (UART) Block Diagram

9.1.3 Block Diagram Signal List

Table 9.1 Serial Port (UART) Register Interface Port List

SIGNAL NAME	DIRECTION	DESCRIPTION
nRESET	input	HW reset
CLK	Input	Block operating clock
EC IF	I/O Bus	Bus used for register access
UART_INT	Output	Host Interrupt
UART_CLK_24MHz	Input	24MHz UART clock.
UART_CLK_16MHz	Input	16MHz UART clock.
UART_TX	Output	UART Transmit data pin
UART_RX	Input	UART Receive data pin

9.1.4 Transmit Operation

Transmission is initiated by writing the data to be sent to the TX Holding Register or to the TX FIFO (if enabled). The data is then transferred to the TX Shift Register together with a start bit and parity and stop bits as determined by settings in the Line Control Register. The bits to be transmitted are then shifted out of the TX Shift Register in the order Start bit, Data bits (LSB first), Parity bit, Stop bit, using the output from the Baud Rate Generator (divided by 16) as the clock.

If enabled, a TX Holding Register Empty interrupt will be generated when the TX Holding Register or the TX FIFO (if enabled) becomes empty.

When FIFOs are enabled (i.e. bit 0 of the FIFO Control Register is set), the M16550S can store up to 16 bytes of data for transmission at a time. Transmission will continue until the TX FIFO is empty. The FIFO's readiness to accept more data is indicated by interrupt.

9.1.5 Receive Operation

Data is sampled into the RX Shift Register using the Receive clock, divided by 16. The Receive clock is provided either by the Baud Rate Generator. A filter is used to remove spurious inputs that last for less than two periods of the Receive clock. When the complete word has been clocked into the receiver, the data bits are transferred to the RX Buffer Register or to the RX FIFO (if enabled) to be read by the CPU. (The first bit of the data to be received is placed in bit 0 of this register.) The receiver also checks that the parity bit and stop bits are as specified by the Line Control Register.

If enabled, an RX Data Received interrupt will be generated when the data has been transferred to the RX Buffer Register or, if FIFOs are enabled, when the RX Trigger Level has been reached. Interrupts can also be generated to signal RX FIFO Character Timeout, incorrect parity, a missing stop bit (frame error) or other Line Status errors.

When FIFOs are enabled (i.e. bit 0 of the FIFO Control Register is set), the M16550S can store up to 16 bytes of received data at a time. Depending on the selected RX Trigger Level, interrupt will go active to indicate that data is available when the RX FIFO contains 1, 4, 8 or 14 bytes of data.

9.2 Power, Clocks and Reset

9.2.1 Power

This block is only active if UART_POWER_EN is set to a '1' in DEV_CLK_EN register, otherwise this block is disabled and the clocks are shut off.

9.2.2 Clocks

The MCLK used for the logic in this block is 60Mhz. The UART_CLK is sourced from either a 24MHz or 16MHz clock. The two different clocks are required to ensure a sub 1% error for 50baud to 2Mbaud. In order to maintain communication with acceptable error, an accurate baud clock is required. The clock is selected using the BAUD_CLK_SEL bit in UART_CTL register.

9.2.3 Reset

Table 9.2 details the effect of nRESET on each of the runtime registers of the Serial Port.

Table 9.2 Reset Function Table

REGISTER/SIGNAL	RESET CONTROL	RESET STATE
Interrupt Enable Register	RESET	All bits low
Interrupt Identification Reg.	RESET	Bit 0 is high; Bits 1 - 7 low
FIFO Control	RESET	All bits low
Line Control Reg.	RESET	All bits low
MODEM Control Reg.	RESET	All bits low
Line Status Reg.	RESET	All bits low except 5, 6 high
MODEM Status Reg.	RESET	Bits 0 - 3 low; Bits 4 - 7 input
TXD1, TXD2	RESET	High
INTRPT (RCVR errs)	RESET/Read LSR	Low
INTRPT (RCVR Data Ready)	RESET/Read RBR	Low
INTRPT (THRE)	RESET/Read IIR/Write THR	Low
OUT2B	RESET	High
RTSB	RESET	High
DTRB	RESET	High
OUT1B	RESET	High
RCVR FIFO	RESET/ FCR1*FCR0/_FCR0	All Bits Low
XMIT FIFO	RESET/ FCR1*FCR0/_FCR0	All Bits Low

The Runtime register are reset on nRESET. Refer to Table 9.2 for effected registers and “Power, Clocks and Resets” chapter for definitions of nRESET.

See “Reset Interface” section for details on reset.

9.3 Interrupts

The Two Pin Serial Port (UART) can generate an interrupt event. The interrupt source is routed onto the UART_RX bit in the GIRQ4 Source Register, and is a level, active high signal.

9.4 Registers

The Two Pin Serial Port (UART) registers are located on the EC_SPB at offset address 0x4000.

Each instance of the Two Pin Serial Port (UART) has its own Logical Device Number, and Base Address as indicated in Table 9.3.

Table 9.3 Two Pin Serial Port (UART) **Base Address Table**

Two Pin Serial Port (UART) **INSTANCE**	**AHB BASE ADDRESS**
UART	EC_SPB + 4000h

Table 9.4 is a register summary for one instance of the Two Pin Serial Port (UART). Each EC address is indicated as an SPB Offset from its AHB base address. The following table summarizes the registers allocated for the Controller. The offset field in the following table is the offset from the Embedded Controller's (EC) Base Address.

Table 9.4 Two Pin Serial Port (UART) **Register Summary**

	DLAB (Note 9.2)	**EC_SPB INTERFACE**			**NOTES**
REGISTER NAME		**OFFSET ADDRESS**	**BYTE LANE**	**EC TYPE**	
Receive Buffer Register (RB)**,**	0	0x00	0	R	
Transmit Buffer Register (TB)	0	0x00	0	W	
Programmable Baud Rate Generator (and Divisor)	1	0x00	0	R/W	

Table 9.4 Two Pin Serial Port (UART) **Register Summary**

REGISTER NAME	DLAB (Note 9.2)	EC_SPB INTERFACE OFFSET ADDRESS	BYTE LANE	EC TYPE	NOTES
Programmable Baud Rate Generator (and Divisor)	1	0x01	1	R/W	
Interrupt Enable Register (IER)	0	0x01	1	R/W	
FIFO Control Register (FCR),	X	0x02	2	W	
Interrupt Identification Register (IIR)	X	0x02	2	R	
Line Control Register (LCR)	X	0x03	3	R/W	
Modem Control Register (MCR)	X	0x04	0	R/W	
Line Status Register (LSR)	X	0x05	1	R	
Modem Status Register (MSR)	X	0x06	2	R	
Scratchpad Register (SCR)	X	0x07	3	R/W	
		EC INTERFACE			
REGISTER NAME	N/A	OFFSET ADDRESS	BYTE LANE	EC TYPE	
UART Control Register		0x10	0	R/W	

Note 9.2 DLAB is Bit 7 of the Line Control Register

9.5 Register Summary

Table 9.5 Register Summary

ADDRESS (Note 9.3)	R/W	REGISTER NAME	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ADDR = 0 DLAB = 0	R	Receive Buffer r	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0 (Note 9.4)
ADDR = 0 DLAB = 0	W	Transmitter Holding r	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
ADDR = 1 DLAB = 0	R/W	Interrupt Enable r	Reserved				Enable Modem Status Interrupt (EMSI)	Enable Receiver Line Sta-tus Inter-rupt (ELSI)	Enable Trans-mitter Holding Register Empty Interrupt (ETH-REI)	Enable Receive d Data Avail-able Interrupt (ERDAI)

Table 9.5 Register Summary

ADDRESS (Note 9.3)	**R/W**	**REGISTER NAME**	**BIT 7**	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
ADDR = 2	R	Interrupt Ident. r	FIFOs Enabled (Note 9.8)	FIFOs Enabled (Note 9.8)	Reserved		Interrupt ID Bit (Note 9.8)	Interrupt ID Bit	Interrupt ID Bit	"0" if Interrupt Pending
ADDR = 2	W	FIFO Control r	RCVR Trigger MSB	RCVR Trigger LSB	Reserved		DMA Mode Select (Note 9.9)	XMIT FIFO Reset	RCVR FIFO Reset	FIFO Enable
ADDR = 3	R/W	Line Control r	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Number of Stop Bits (STB)	Word Length Select Bit 1 (WLS1)	Word Length Select Bit 0 (WLS0)
ADDR = 4	R/W	MODEM Control r	**Reserved**			**Loop**	**OUT2 (Note 9.6)**	**OUT1 (Note 9.6)**	**Request to Send (RTS)**	**Data Termi-nal Ready (DTR)**
ADDR = 5	R/W	Line Status r	**Error in RCVR FIFO (Note 9.8)**	**Trans-mitter Empty (Note 9.5)**	**Trans-mitter Holding Regis-ter**	**Break Interrupt**	**Fram-ing Error**	**Parity Error**	**Over-run Error**	**Data Ready**
ADDR = 6	R/W	MODEM Status r	Data Carrier Detect (DCD)	Ring Indica-tor (RI)	Data Set Ready (DSR)	Clear to Send (CTS)	Delta Data Carrier Detect (DDCD)	Trailing Edge Ring Indicator (TERI)	Delta Data Set Ready (DDSR)	Delta Clear to Send (DCTS)
ADDR = 7	R/W	Scratch r (Note 9.7)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR = 0 DLAB = 1	R/W	Divisor Latch (LS)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR = 1 DLAB = 1	R/W	Divisor Latch (MS)	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

UART Register Summary Notes:

Note 9.3 DLAB is Bit 7 of the Line Control Register (ADDR = 3).

Note 9.4 Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Note 9.5 When operating in the XT mode, this bit will be set any time that the transmitter shift register is empty.

Note 9.6 This bit no longer has a pin associated with it.

Note 9.7 When operating in the XT mode, this register is not available.

Note 9.8 These bits are always zero in the non-FIFO mode.

Note 9.9 Writing a one to this bit has no effect. DMA modes are not supported in this chip.

9.6 Detailed Description of Accessible Runtime Registers

9.6.1 Receive Buffer Register (RB)

Table 9.6 UART Receive Buffer)

UART_RX_DATA (DLAB=0) (OFFSET 0X00 RESET=0X00)			**UART RECEIVED DATA**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DATA	R	This register holds the received incoming data byte. Bit 0 is the least significant bit, which is transmitted and received first. Received data is double buffered; this uses an additional shift register to receive the serial data stream and convert it to a parallel 8 bit word which is transferred to the Receive Buffer register. The shift register is not accessible If enabled via IER[0], an RX Buffer Register Interrupt is generated when the buffer contains data to read. If the FIFOs are disabled, this register is undefined after reset. If the FIFOs are enabled, this register will return zero after a reset, if the RX FIFO is empty.

9.6.2 Transmit Buffer Register (TB)

Table 9.7 UART Transmit Buffer)

UART_TX_DATA (DLAB=0) (OFFSET 0X00 RESET=0X00)			**UART TRANSMIT DATA**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TX_DATA	W	This register contains the data byte to be transmitted. The transmit buffer is double buffered, utilizing an additional shift register (not accessible) to convert the 8 bit data word to a serial format. This shift register is loaded from the Transmit Buffer when the transmission of the previous byte is complete

9.6.3 Interrupt Enable Register (IER)

The lower four bits of this register control the enables of the five interrupt sources of the Serial Port interrupt. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of this register. Similarly, setting the appropriate bits of this register to a high, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and disables any Serial Port interrupt out of the SEC2410/SEC4410. All other system functions operate in their normal manner, including the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are described below.

Table 9.8 UART Interrupt Enable Register

UART_INTERRUPT_EN (DLAB=0) (OFFSET 0X01 RESET=0X00)			**UART INTERRUPT ENABLE**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read ‘0’
3	EMSI	R/W	This bit enables the MODEM Status Interrupt when set to logic "1". This is caused when one of the Modem Status Register bits changes state.
2	ELSI	R/W	This bit enables the Received Line Status Interrupt when set to logic "1". The error sources causing the interrupt are Overrun, Parity, Framing and Break. The Line Status Register must be read to determine the source
1	ETHREI	R/W	This bit enables the Transmitter Holding Register Empty Interrupt when set to logic "1".
0	ERDAI	R/W	This bit enables the Received Data Available Interrupt (and timeout interrupts in the FIFO mode) when set to logic "1"

9.6.4 FIFO Control Register (FCR)

Table 9.9 UART FIFO Control Register

UART_FIFO_CTL (DLAB=X) (OFFSET 0X02 RESET=0X00)			**UART FIFO CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	RECV_FIFO_TRIG	W	These bits are used to set the trigger level for the RCVR FIFO interrupt Value - Trigger level 00 - 1 Bytes 01 - 4 Bytes 10 - 8 Bytes 11 - 14 Bytes
5:3	Reserved	W	When read, these bit will contain the value in the UART_INT_ID register
2	CLR_XMIT_FIFO	W	Setting this bit to a logic "1" clears all bytes in the XMIT FIFO and resets its counter logic to “0”. The shift register is not cleared. This bit is self-clearing
1	CLR_RCV_FIFO	W	Setting this bit to a logic "1" clears all bytes in the RCVR FIFO and resets its counter logic to “0”. The shift register is not cleared. This bit is self-clearing.
0	EXRF	W	Enable XMIT and RECV FIFO. Setting this bit to a logic "1" enables both the XMIT and RCVR FIFOs. Clearing this bit to a logic "0" disables both the XMIT and RCVR FIFOs and clears all bytes from both FIFOs. When changing from FIFO Mode to non-FIFO (16450) mode, data is automatically cleared from the FIFOs. This bit must be a 1 when other bits in this register are written to or they will not be properly programmed.

Note: This is a write only register at the same location as the IIR.

9.6.5 Interrupt Identification Register (IIR)

Table 9.10 UART Interrupt Identification Register(IIR)

UART_IIR (DLAB=X) (OFFSET 0X02 RESET=0X01)			**UART INTERRUPT IDENTIFICATION REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	FIFO_EN	R	These two bits are set when the FIFO CONTROL Register bit 0 equals 1
5:4	Reserved	R	Always read ‘0’
3:1	INTLD	R	These three bits of the IIR are used to identify the highest priority interrupt pending as indicated by Table 9.11. In non-FIFO mode, Bit[3] is a logic “0”. In FIFO mode Bit[3] is set along with Bit[2] when a timeout interrupt is pending.
0	IPEND	R	This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic "0", an interrupt is pending and the contents of the IIR may be used as a pointer to the appropriate internal service routine. When bit 0 is a logic "1", no interrupt is pending.

By accessing this register, the CPU can determine the highest priority interrupt and its source. Four levels of priority interrupt exist. They are in descending order of priority:

1. Receiver Line Status (highest priority)
2. Received Data Ready
3. Transmitter Holding Register Empty
4. MODEM Status (lowest priority)

Information indicating that a prioritized interrupt is pending and the source of that interrupt is stored in the Interrupt Identification Register (refer to Table 9.11). When the CPU accesses the IIR, the Serial Port freezes all interrupts and indicates the highest priority pending interrupt to the CPU. During this CPU access, even if the Serial Port records new interrupts, the current indication does not change until access is completed. The contents of the IIR are described below.

Table 9.11 Interrupt Control Table

FIFO MODE ONLY	INTERRUPT IDENTIFICATION REGISTER			INTERRUPT SET AND RESET FUNCTIONS			
BIT 3	BIT 2	BIT 1	BIT 0	PRIORITY LEVEL	INTERRUPT TYPE	INTERRUPT SOURCE	INTERRUPT RESET CONTROL
0	0	0	1	-	None	None	-
	1	1	0	Highest	Receiver Line Status	Overrun Error, Parity Error, Framing Error or Break Interrupt	Reading the Line Status Register
		0		Second	Received Data Available	Receiver Data Available	Read Receiver Buffer or the FIFO drops below the trigger level.
1					Character Timeout Indication	No Characters Have Been Removed From or Input to the RCVR FIFO during the last 4 Char times and there is at least 1 char in it during this time	Reading the Receiver Buffer Register
0	0	1		Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if Source of Interrupt) or Writing the Transmitter Holding Register
	0	0		Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

9.6.6 Line Control Register (LCR)

This register contains the format information of the serial line. The bit definitions are:

Table 9.12 UART Line Control Register(LCR)

UART_LINE_CTL (DLAB=X) (OFFSET 0X03 RESET=0X01)			**UART LINE CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	DLAB	R/W	Divisor Latch Access Bit (DLAB). It must be set high (logic "1") to access the Divisor Latches of the Baud Rate Generator during read or write operations. It must be set low (logic "0") to access the Receiver Buffer Register, the Transmitter Holding Register, or the Interrupt Enable Register.
6	BREAK_CTL	R/W	Set Break Control bit. When bit 6 is a logic "1", the transmit data output (TXD) is forced to the Spacing or logic "0" state and remains there (until reset by a low level bit 6) regardless of other transmitter activity. This feature enables the Serial Port to alert a terminal in a communications system.
5	STICK_PARITY	R/W	Stick Parity bit. When parity is enabled it is used in conjunction with bit 4 to select Mark or Space Parity. When LCR bits 3, 4 and 5 are 1 the Parity bit is transmitted and checked as a 0 (Space Parity). If bits 3 and 5 are 1 and bit 4 is a 0, then the Parity bit is transmitted and checked as 1 (Mark Parity). If bit 5 is 0 Stick Parity is disabled. Bit 3 is a logic "1" and bit 5 is a logic "1", the parity bit is transmitted and then detected by the receiver in the opposite state indicated by bit 4.
4	PARITY_SEL	R/W	Even Parity Select bit. When bit 3 is a logic "1" and bit 4 is a logic "0", an odd number of logic "1"'s is transmitted or checked in the data word bits and the parity bit. When bit 3 is a logic "1" and bit 4 is a logic "1" an even number of bits is transmitted and checked.
3	PARITY_EN	R/W	Parity Enable bit. When bit 3 is a logic "1", a parity bit is generated (transmit data) or checked (receive data) between the last data word bit and the first stop bit of the serial data. (The parity bit is used to generate an even or odd number of 1s when the data word bits and the parity bit are summed).
2	STOP_BITS	R/W	This bit specifies the number of stop bits in each transmitted or received serial character. Table 9.13 summarizes the information
1:0	WORD_LEN	R/W	These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows: Value - Word Length 00 - 5 bits 01 - 6 bits 10 - 7 bits 11 - 8 bits The Start, Stop and Parity bits are not included in the word length

.

Table 9.13 Stop Bits

BIT 2	**WORD LENGTH**	**NUMBER OF STOP BITS**
0	--	1

Table 9.13 Stop Bits

BIT 2	WORD LENGTH	NUMBER OF STOP BITS
1	5 bits	1.5
	6 bits	2
	7 bits	
	8 bits	

Note 9.10 The receiver will ignore all stop bits beyond the first, regardless of the number used in transmitting.

9.6.7 Modem Control Register (MCR)

This 8-bit register controls the interface with the MODEM or data set (or device emulating a MODEM). The contents of the MODEM control register are described below.

Table 9.14 UART Modem Control Register(MCR)

UART_MODEM_CTL (DLAB=X) (OFFSET 0X04 RESET=0X00)			UART MODEM CONTROL REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read ‘0’
4	LOOPBACK	R/W	This bit provides the loopback feature for diagnostic testing of the Serial Port. When bit 4 is set to logic "1", the following occur: 1. The TXD is set to the Marking State (logic "1"). 2. The receiver Serial Input (RXD) is disconnected. 3. The output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input. 4. All MODEM Control inputs (nCTS, nDSR, nRI and nDCD) are disconnected. 5. The four MODEM Control outputs (nDTR, nRTS, OUT1 and OUT2) are internally connected to the four MODEM Control inputs (nDSR, nCTS, RI, DCD). 6. The Modem Control output pins are forced inactive high. 7. Data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the Serial Port. In the diagnostic mode, the receiver and the transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register

UART_MODEM_CTL (DLAB=X) (OFFSET 0X04 RESET=0X00)			**UART MODEM CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
3	OUT2	R/W	Output 2 (OUT2). This bit is used to enable an UART interrupt. When OUT2 is a logic "0", the serial port interrupt output is forced to a high impedance state - disabled. When OUT2 is a logic "1", the serial port interrupt outputs are enabled.
2	OUT1	R/W	This bit controls the Output 1 (OUT1) bit. This bit does not have an output pin and can only be read or written by the CPU.
1	RTS	R/W	This bit controls the Request To Send (nRTS) output. Bit 1 affects the nRTS output in a manner identical to that described above for bit 0.
0	DTR	R/W	This bit controls the Data Terminal Ready (nDTR) output. When bit 0 is set to a logic "1", the nDTR output is forced to a logic "0". When bit 0 is a logic "0", the nDTR output is forced to a logic "1".

9.6.8 Line Status Register (LSR)

Table 9.15 UART Line Status Register(MCR)

UART_LINE_STAT (DLAB=X) (OFFSET 0X05 RESET=0X60)			**UART LINE STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	FIFO_ERROR	R	This bit is permanently set to logic "0" in the 450 mode. In the FIFO mode, this bit is set to a logic "1" when there is at least one parity error, framing error or break indication in the FIFO. This bit is cleared when the LSR is read if there are no subsequent errors in the FIFO.
6	XMIT_EMPTY	R	Transmitter Empty. Bit 6 is set to a logic "1" whenever the Transmitter Holding Register (THR) and Transmitter Shift Register (TSR) are both empty. It is reset to logic "0" whenever either the THR or TSR contains a data character. Bit 6 is a read only bit. In the FIFO mode this bit is set whenever the THR and TSR are both empty
5	THRE	R	Transmitter Holding Register Empty (THRE). Bit 5 indicates that the Serial Port is ready to accept a new character for transmission. In addition, this bit causes the Serial Port to issue an interrupt when the Transmitter Holding Register interrupt enable is set high. The THRE bit is set to a logic "1" when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic "0" whenever the CPU loads the Transmitter Holding Register. In the FIFO mode this bit is set when the XMIT FIFO is empty, it is cleared when at least 1 byte is written to the XMIT FIFO. Bit 5 is a read only bit.

UART_LINE_STAT (DLAB=X) (OFFSET 0X05 RESET=0X60)			**UART LINE STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
4	BREAK_INT	R	Break Interrupt. Bit 4 is set to a logic “1” whenever the received data input is held in the Spacing state (logic “0”) for longer than a full word transmission time (that is, the total time of the start bit + data bits + parity bits + stop bits). The BI is reset after the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is indicated when the associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. Restarting after a break is received, requires the serial data (RXD) to be logic “1” for at least 1/2 bit time. Bits 1 through 4 are the error conditions that produce a Receiver Line Status Interrupt BIT 3 **Note:** whenever any of the corresponding conditions are detected and the interrupt is enabled
3	FRAME_ERROR	R	Framing Error. Bit 3 indicates that the received character did not have a valid stop bit. Bit 3 is set to a logic "1" whenever the stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level). The FE is reset to a logic "0" whenever the Line Status Register is read. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is indicated when the associated character is at the top of the FIFO. The Serial Port 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 takes in the 'data'
2	PARITY_ERROR	R	Parity Error. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. The PE is set to a logic "1" upon detection of a parity error and is reset to a logic "0" whenever the Line Status Register is read. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is indicated when the associated character is at the top of the FIFO.
1	OVERRUN_ERROR	R	Overrun Error. Bit 1 indicates that data in the Receiver Buffer Register was not read before the next character was transferred into the register, thereby destroying the previous character. In FIFO mode, an overrun error will occur only when 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 not transferred to the FIFO. The OE indicator is set to a logic "1" immediately upon detection of an overrun condition, and reset whenever the Line Status Register is read
0	DATA_READY	R	Data Ready. It is set to a logic "1" whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to a logic "0" by reading all of the data in the Receive Buffer Register or the FIFO

9.6.9 Modem Status Register (MSR)

This 8 bit register provides the current state of the control lines from the MODEM (or peripheral device). In addition to this current state information, four bits of the MODEM Status Register (MSR) provide change information.

These bits are set to logic "1" whenever a control input from the MODEM changes state. They are reset to logic "0" whenever the MODEM Status Register is read.

Table 9.16 UART Modem Status Register(MSR)

UART_MSR (DLAB=X) (OFFSET 0X06 RESET=0BXXXX0000)			**UART MODEM STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	DCD#	R	This bit is the complement of the Data Carrier Detect (nDCD) input. If bit 4 of the MCR is set to logic "1", this bit is equivalent to OUT2 in the MCR.
6	RI#	R	This bit is the complement of the Ring Indicator (nRI) input. If bit 4 of the MCR is set to logic "1", this bit is equivalent to OUT1 in the MCR.
5	DSR	R	This bit is the complement of the Data Set Ready (nDSR) input. If bit 4 of the MCR is set to logic "1", this bit is equivalent to DTR in the MCR.
4	CTS	R	This bit is the complement of the Clear To Send (nCTS) input. If bit 4 of the MCR is set to logic "1", this bit is equivalent to nRTS in the MCR.
3	DDCD	R	Delta Data Carrier Detect (DDCD). Bit 3 indicates that the nDCD input to the chip has changed state.
2	TERI	R	Trailing Edge of Ring Indicator (TERI). Bit 2 indicates that the nRI input has changed from logic "0" to logic "1".
1	DDSR	R	Delta Data Set Ready (DDSR). Bit 1 indicates that the nDSR input has changed state since the last time the MSR was read.
0	DCTS	R	Delta Clear To Send (DCTS). Bit 0 indicates that the nCTS input to the chip has changed state since the last time the MSR was read.

Note 9.11 Whenever bit 0, 1, 2, or 3 is set to a logic "1", a MODEM Status Interrupt is generated.

.

APPLICATION NOTE: The Modem Status Register (MSR) only provides the current state of the UART MODEM control lines in Loopback Mode. The SEC2410/SEC4410 does not support external connections for the MODEM Control inputs (nCTS, nDSR, nRI and nDCD) or for the four MODEM Control outputs (nDTR, nRTS, OUT1 and OUT2).

9.6.10 Scratchpad Register (SCR)

Table 9.17 UART Scratch Pad Register(SCR)

UART_SCRATCH (DLAB=X) (OFFSET 0X07 RESET=0X00)			**UART SCTRATCH PAD REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	SCRATCH	R/W	his 8 bit read/write register has no effect on the operation of the Serial Port. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

9.6.11 Programmable Baud Rate Generator (and Divisor)

The incoming clock is divided by the value held in DLL & DLM (1 - 65535) to produce the Baud Rate Generator Output signal (BAUD)

Table 9.18 UART Divisor Latch Low(DLL)

UART_DIV_LAT_LO (DLAB=1) (OFFSET 0X00 RESET=0X00)			UART DIVISOR LATCH LOW
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	BAUD_DIVISOR[7:0]	R/W	Least significant 8 bits of the baud rate divisor is stored here.

Table 9.19 UART Divisor Latch High(DHL)

UART_DIV_LAT_HI (DLAB=1) (OFFSET 0X01 RESET=0X00)			UART DIVISOR LATCH HIGH
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	BAUD_DIVISOR[14:8]	R/W	Most significant 8 bits of the baud rate divisor is stored here.

Note: DLL & DLM can only be updated if DLAB bit is set ("1"). Note too that, unlike the original device, division by 1 generates a BAUD signal that is constantly high.

The table below shows the divisor needed to generate a given baud rate from CLOCK inputs of 24 MHz. The effective clock enable generated is 16 x the required baud rate. For clock frequencies (fCLOCK) not covered by this table, the required divisor can be calculated as follows:

Divisor value = fCLOCK / (16 X desired baud rate)

Table 9.20 UART Baud Rates (24.00 MHz source)

DESIRED BAUD RATE	DIVISOR USED TO GENERATE 16X CLOCK	PERCENT ERROR
50	30000	0.00
75	20000	0.000
110	13636	0.00
134.5	11152	0.00
150	10000	0.00
300	5000	0.00
600	2500	0.00
1200	1250	0.00
1800	833	0.04
2000	750	0.00
2400	625	0.00
3600	417	0.08

Table 9.20 UART Baud Rates (24.00 MHz source)

DESIRED BAUD RATE	DIVISOR USED TO GENERATE 16X CLOCK	PERCENT ERROR
4800	313	0.16
7200	208	0.16
9600	156	0.16
19200	78	0.16
38400	39	0.16
57600	26	0.16
115200	13	0.16

Table 9.21 UART Baud Rates (16.00 MHz source)

DESIRED BAUD RATE	DIVISOR USED TO GENERATE 16X CLOCK	PERCENT ERROR
250K	4	0.00
500K	2	0.00
1000K	1	0.00

9.7 Detailed Description of Configuration Registers

9.7.1 UART Control Register

Table 9.22 UART Control

UART_CTL (EC_SPB + 0X43F0 RESET=0X00)			UART CONTROL REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read ‘0
3	BAUD_CLK_SRC_ALT		UART external reference clock selection. 0 : Baud clock is 24MHz 1 : Baud clock is 16Mhz
2	POLARITY	R/W	1 = UART_TX and UART_RX pins functions are inverted. 0 = UART_TX and UART_RX pins functions are not inverted.
1	POWER	R/W	For this device, this bit should always be ‘0’

UART_CTL (EC_SPB + 0X43F0 RESET=0X00)			**UART CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
0	BAUD_CLK_SRC	R/W	UART Clock Select 0 : Baud Clock is approximately 1.8432 Mhz derived as Internal mclk frequency / 1843200. 1 : Baud Clock is deterrminded by bit 3

Table 9.23 UART Configuration Activel

UART_ACTV (EC_SPB + 0X4330 RESET=0X00)			**UART CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0
0	ACTV	R/W	UART Active 0 (default) = The UART block is inactive 1 = The UART block is active. This bit needs to be set to enable the block

Chapter 10 Watchdog Timer

10.1 General Description

The function of the Watchdog Timer is to provide a mechanism to detect if the embedded controller has failed.

When enabled, the WATCHDOG Timer (WDT) circuit will generate a WDT Event if the user program fails to reload the WDT within a specified length of time known as the WDT Interval.

This timer can be held inactive via the WDT Stall feature if the Hibernation timer, Week Timer, or the JTAG interface are enabled and active. This featured if enabled can be used to avoid unintended system resets.

Some operations can be carried out without any delay, e.g., registers can be read at any time and disabling the WDT takes effect immediately. On the other hand, 'kicking' the WDT may have a latency of up to 1 32-kHz cycle (~ 30 us). Similarly, when the load register is altered, the WDT cannot be enabled for up to 1 32-kHz cycle.

10.2 Block Diagram

Block diagram of the WDT is given below:

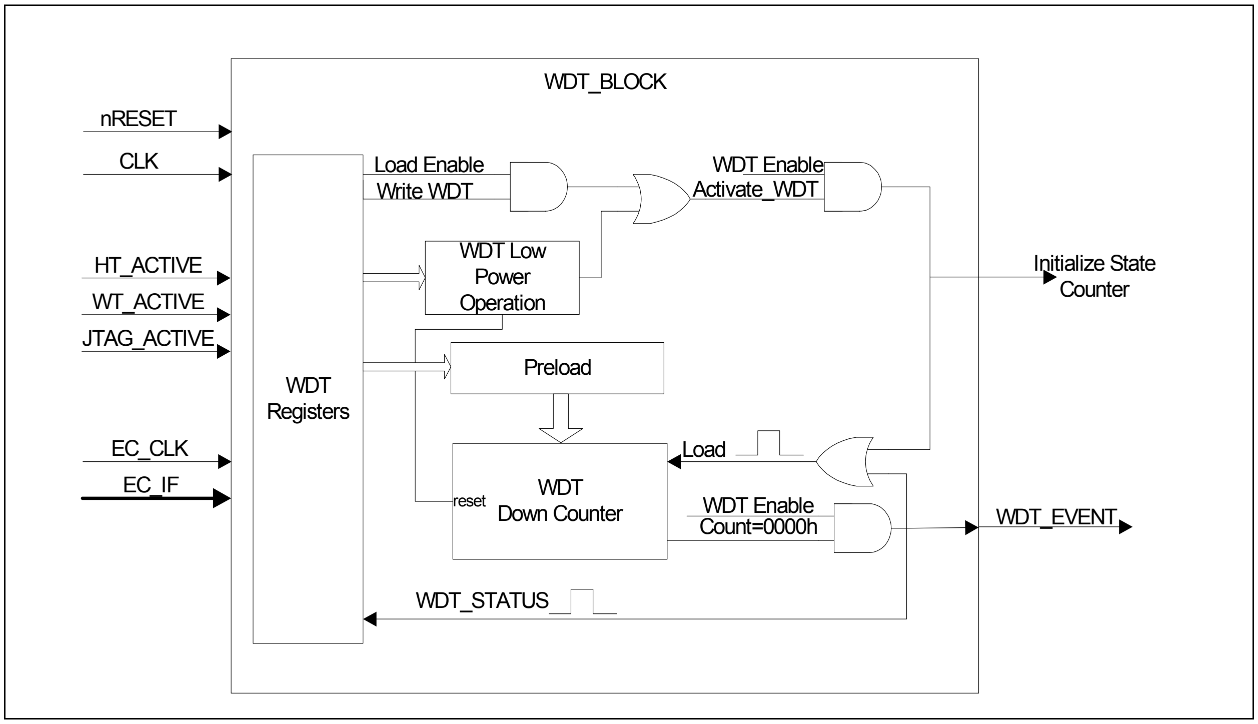


Figure 10.1 Watchdog Timer Interface Block Diagram

10.3 Watchdog Timer Interface Signal List

Table 10.1 Watchdog Timer Interface Signal List

SIGNAL NAME	DIRECTION	DESCRIPTION
nRESET	INPUT	Power on Reset to the block
CLK	INPUT	32.768kHz clock
HT_ACTIVE	INPUT	Signal indicating the Hibernation Timer is active and counting. See “Hibernation Timer” chapter.
WT_ACTIVE	INPUT	Signal indicating the Week Timer is active and counting. See “Week Alarm Interface” chapter.
JTAG_ACTIVE	INPUT	Signal indicating the JTAG interface is active.
EC_IF	I/O Bus	Bus used by microprocessor to access the registers in this block.
EC_CLK	INPUT	Clock used to access internal register.
WDT Event	OUTPUT	Pulse generated when WDT expires.

10.4 Clocks

This block has two clock inputs, the EC_CLK and CLK. The EC_CLKis used in the interface to the embedded controller accessible registers. The 32.768KHz CLK is the clock source for the Watchdog Timer functional logic, including the counter.

10.5 WDT Event output

In the SEC2410/SEC4410, the assertion of the WDT Event output causes a Watch-Dog Timer Forced Reset.

The WDT Event state is also retained through a Watch-Dog Timer Forced Reset in the WD_RESET bit of Reset control register (RESET_CTL).

The WDT Event output is not directly connected to an EC interrupt.

10.6 WDT Operation

10.6.1 WDT Activation Mechanism

The WDT is activated by the following sequence of operations during normal operation:

1. Load the WDT Load Register with the count value.
2. Set the WDT Enable bit in the WDT Control Register.

The WDT Activation Mechanism starts the WDT decrementing counter.

10.6.2 WDT Deactivation Mechanism

The WDT is deactivated by the clearing the WDT Enable bit in the WDT Control Register. The WDT Deactivation Mechanism places the WDT in a low power state in which clock are gated and the counter stops decrementing.

10.6.3 WDT Reload Mechanism

The WDT must be reloaded within periods that are shorter than the programmed watchdog interval; otherwise the WDT will underflow and a WDT Event will be generated and the WDT_STATUS bit will be set in the WDT Control Register. It is the responsibility of the user program to continually execute sections of code which reload the watchdog timer (WDT) causing the counter to be reloaded

There are two methods of reloading the WDT: a write to the WDT Kick Register or the WDT Activation Mechanism.

10.6.4 WDT Interval

The WDT Interval is the time it takes for the WDT to decrements from the WDT Load Register value to 0000h. The WDT Count Register value takes 1.007ms to decrement by 1 count.

10.6.5 WDT STALL Operation

The WDT has several events that can cause the WDT STALL. When a WDT STALL event is asserted, the WDT stops decrementing, and the WDT enters a low power state. When a WDT STALL event is de-asserted, the counter is reloaded with the programmed preload value and starts decrementing.

The WDT STALL feature has been implemented for convenience. If the system designer chooses not to utilize the WDT STALL feature, the WDT defaults with the WDT STALL feature disabled.

There are three Stall inputs to the WDT: HT_ACTIVE, WT_ACTIVE, JTAG_ACTIVE (corresponding to the Hibernation Timer, the Week Alarm Timer, & the J-TAG interface being active). The Stall inputs have individual enable bits: HT_STALL_EN, WT_STALL_EN, JTAG_STALL_EN bits in the WDT Control Register on page 85.

Table 10.2 WDT STALL event Behavior

WDT STALLInputs (activity indicator)	WDT Control Register		**WDT BEHAVIOR**	WDT Event **OUTPUT**
	STALL_EN BIT	WDT Enable **BIT**		
X	X	0	Counter is reset and not active. Clock source to counter is gated to save power.	0
X	0	1	Count is active. (see) If counter > 0000h	0
0	1	1		
X	X	1	Count is decremented to 0000h	1
1	1	1	Counter is not active. Clock source to counter is gated to save power.	0

Note 10.1 When the counter reaches 0000h it wraps to the preload value and starts counting down again. This creates a pulse on the WDT Event output.

Note 10.2 Anytime the Counter is deactivated, the clock source to the counters should be gated in the WDT block to conserve power. This is in addition to the Power and Clocking logic gating the source clocks.

10.7 Instance Description

There is one instance of the Watchdog Timer Interface block implemented in SEC2410/SEC4410located at address 0x40020400.

Table 10.3 Watchdog Timer Interface Register Summary

REGISTER NAME	ADDRESS
WDT Load Register	0x40020400
WDT Control Register	0x40020404
WDT Kick Register	0x40020408
WDT Count Register	0x4002040C

10.8 Detailed Register Descriptions

10.8.1 WDT Load Register

Table 10.4 WDT Load Register

WDT_LOAD (ADDR 0XF00400, RESET=0XFFFF)			WATCH DOG LOAD REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	WDT_LOAD	R/W	Writing this field reloads the Watch Dog Timer counter

10.8.2 WDT Control Register

Table 10.5 WDT Control Register

WDT_CTL (ADDR 0XF00404, RESET=0X00)			WATCH DOG TIMER CONTROL REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read ‘0’
4	JTAG_STALL_EN	R/W	This bit is used to enable the JTAG_ACTIVE (JTAG_RST# pin not asserted) WDT STALL Operation on page 84. 0= JTAG_ACTIVE WDT STALL Operation not enabled 1= JTAG_ACTIVE WDT STALL Operation enabled
3	WT_STALL_EN	R/W	Firmware must always set this bit to ‘0’.
2	HT_STALL_EN	R/W	Firmware must always set this bit to ‘0’.

WDT_CTL (ADDR 0XF00404, RESET=0X00)			**WATCH DOG TIMER CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
1	WDT_STATUS	R/W	WDT_STATUS is set by hardware if the last reset of SEC2410/SEC4410 was caused by an underflow of the WDT. See Section 10.6.3: *WDT Reload Mechanism* on page 84 for more information. This bit must be cleared by the EC firmware writing a '1' to this bit. Writing a '0' to this bit has no effect
0	WDT_ENABLE	R/W	The default of the WDT is inactive. In WDT Operation, the WDT is activated by the sequence of operations defined in Section 10.6.1: *WDT Activation Mechanism* and deactivated by the sequence of operations defined in Section 10.6.2: *WDT Deactivation Mechanism*. In WDT STALL Operation, hardware may be enabled to automatically activate and deactivate the WDT.

10.8.3 WDT Kick Register

Table 10.6 WDT Kick Register

WDT_KICK (ADDR 0XF00408, RESET=NA)			**WATCH DOG TIMER KICK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	KICK	W	This register is a strobe. Reads of this register return 0. Writes to this register cause the WDT to reload the WDT_LOAD value and start decrementing when the WDT_ENABLE bit in the WDT_CTL register is set to '1'. When the WDT_ENABLE but is cleared to '0', writes to this register have no effect.

10.8.4 WDT Count Register

Table 10.7 WDT Count Register

WDT_COUNT (ADDR 0XF0040C, RESET=0XFFFF)			**WATCH DOG COUNT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	WDT_COUNT	R	This register provide the current WDT count

Chapter 11 16-Bit Timer/Counter Block

11.1 General Description

The SEC2410/SEC4410 16-Bit Timer/Counter Block/Counter Block implements four 16-bit auto-reloading timer/counters. Each timer/counter is categorized as one of three types:

- General Purpose,
- Input-Only,
- Input/Output.

All timer/counters have four modes of operation:

- Timer,
- One-Shot,
- Event,
- Measurement.

In addition, each timer/counter can generate a unique wake-up interrupt to the EC. The clock for each timer/counter is derived from the system clock (60MHz) and can be divided down by a prescaler. Input-Only and Input/Output timers can also use an external input pin to clock or gate the counter. To aid operation in noisy environments the external input pin also has a selectable noise filter. If large counts are required, the output of each timer/counter can be internally connected to the next timer/counter.

The following section defines terms used in this chapter.

TERM	DEFINITION
Overflow	When the timer counter transitions from FFFFh to 0000h
Underflow	When the timer counter transitions from 0000h to FFFFh.
Timer Tick Rate	This is the rate at which the timer is incremented or decremented.

11.2 Block Diagram

Block diagram of one timer/counter is given below:

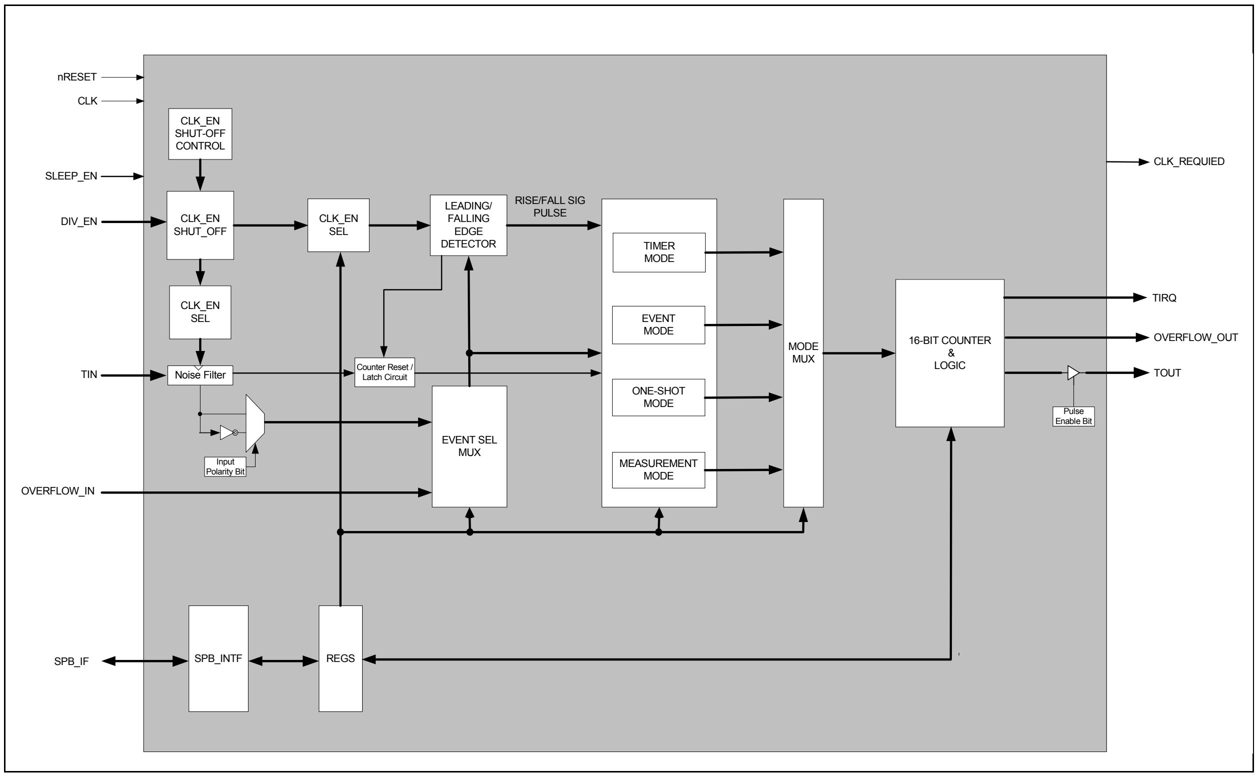


Figure 11.1 Block Diagram for Timer x

11.3 Signal List for Block Diagram

Table 11.1 Block Diagram Signal List Description

SIGNAL NAME	DIRECTION	DESCRIPTION
nRESET	INPUT	HW reset active low. All Timers are reseted to the default value.
CLK	INPUT	60MHz clock source to block.
SLEEP_EN	INPUT	Sleep Enable signals.
DIV_EN	INPUT	Clock Enables for supporting Filter and Timer frequencies.
TIN	INPUT	Input signal for timer
OVERFLOW_IN	INPUT	Overflow input signal.
SPB_IF	I/O Bus	Bus used by microprocessor to access the registers in this block.
CLK_REQUIRED	INPUT	Clock required.
TIRQ	OUTPUT	Timer Interrupt Request
OVERFLOW_OUT	OUTPUT	Overflow output signal.
TOUT	OUTPUT	Output signal.

11.4 Clocks, Reset and Power

11.4.1 Clocks

Input clock can be divided to support various frequencies in range from 60MHz to 469KHz. All supported frequencies are listed in Table 11.10: *Timer Clock Frequencies* on page 101. Independently, any of these clock frequencies may be selected for the filter clock via the FCLK[3:0] bits located in Section 11.8.2: *Timer x Clock and Event Control Register* on page 101.

The Event input is synchronized to FCLK and (if enabled) filtered by a three stage filter. The resulting recreated clock is used to clock the timer in Event mode. In Bypass Mode (Sync Only), the pulse width of the external signal must be at least 2x the pulse width of the FCLK source. If the Event input not in Bypass Mode (Sync and Filter), the pulse width of the external signal must be at least 4x the pulse width of the sync and filter clock

11.4.2 Reset

On nRESET all timers are reset to their default values. The timers are also reset by the RESET bit in each Timer x Control Register.

11.4.3 Low Power Modes

This block is designed to conserve power when it is either sleeping or a clock source is not required.

During normal operation, if the timer is disabled via the PD bit of the Timer x Control Register the TIMERx_CLK_REQ bit of the same register and the output signal CLK_REQUIRED are de-asserted. This indicates to the clock generator logic that this timer does not require the 60.00MHz clock source.

During Sleep modes the clock input is gated, the TIMERx_CLK_REQ bit of the Timer x Control Register and the output signal CLK_REQUESTED are asserted, and the interrupt output goes to the inactive state. When the block returns from sleep, if enabled, it will be restarted from the preload value.

The following table illustrates the low power mode options.

Table 11.2 Block Clock Gating in Low Power Modes

POWER DOWN (PD) BIT	SLEEP_ENABLE	BLOCK IDLE STATUS	TIMERX_CLK_REQ	STATE	DESCRIPTION
1	X	NOT IDLE	1	PREPARING to SLEEP	The core clock is still required for up to one Timer Clock period.
		IDLE	0	SLEEPING	The block is idle and the core clock can be stopped.
0	0	X	1	NORMAL OPERATION	The block in neither disabled by firmware nor commanded to SLEEP
	1	NOT IDLE	1	PREPARING to SLEEP	The core clock is still required for up to one Timer Clock period.
		IDLE	0	SLEEPING	The block is idle and the core clock can be stopped.

11.5 Noise Filter

The noise filter uses Filter Clock (FCLK) to filter the signal on the TINx pins. An external TINx pin must remain in the same state for three FCLK ticks before the internal state changes. The Filter Bypass bit in the Timer x Control Register is used to bypass the noise filter.

- The signal TIN may be optionally only synchronized, or synchronized and filtered depending on the filter bypass bit
- The minimum FCLK period must be at least 2X the duration of the TIN signal so that signal can be reliably captured in the bypass mode
- The minimum FCLK period must be at least 4X the duration of the TIN signal so that signal can be reliably captured in the non-bypass mode
- In One-Shot mode, the TIN duration could be smaller than a TCLK period. The filtered signal is latched until the signal is seen in the TCLK domain. This also applies in the filter bypass mode

11.5.1 Starting and Stopping

The SEC2410/SEC4410 timers can be started and stopped by setting and clearing the Timer Enable bit in the Timer Control Register in all modes, except one-shot.

11.6 Operating Modes

11.6.1 Timer Mode

The Timer mode of the SEC2410/SEC4410 is used to generate periodic interrupts to the EC. When operating in this mode the timer always counts down based on one of the internally generated clock sources. The Timer mode is selected by setting the Timer Mode Select bits in the Timer Control Register. See Section 11.8.1: *Timer x Control Register* on page 99.

The period between timer interrupts and the width of the output pulse is determined by the speed of the clock source, the clock divide ratio and the value programmed into the Timer Reload Register. The timer clock source and clock rate are selected using the Clock Source Select bits (TCLK) in the Timer x Clock and Event Control Register. See Section 11.8.2: *Timer x Clock and Event Control Register* on page 101

Table 11.3 Timer Mode Operational Summary

ITEM	DESCRIPTION
Timer Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Filter Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Count Operation	Down Counter
Reload Operation	When the timer underflows: RLOAD = 1, timer reloads from Timer Reload Reg RLOAD = 0, timer rolls over to FFFFh.

Count Start Condition	UPDN = 0 (timer only mode): ENABLE = 1
	UPDN = 1 (timer gate mode): ENABLE = 1 & TIN = 1;

Count Stop Condition	UPDN = 0: ENABLE = 0;
	UPDN = 1: (ENABLE= 0 | TIN = 0)

Interrupt Request Generation Timing	When timer underflows from 0000h to reload value (as determined by RLOAD) an interrupt is generated.
TINx Pin Function	Provides timer gate function
TOUTx Pin Function	TOUT toggles each time the timer underflows (if enabled).
Read From Timer	Current count value can be read by reading the Timer Count Register
Write to Preload Register	After the firmware writes to the Timer Reload Register asserting the RESET loads the timer with the new value programmed in the Timer Reload Register. Note: If the firmware does not assert RESET, the timer will automatically load the Timer Reload Register value when the timer underflows. When the timer is running, values written to the Timer Reload Register are written to the timer counter when the timer underflows. The assertion of Reset also copies the Timer Reload Register into the timer counter.
Selectable Functions	■ Reload timer on underflow with programmed Preload value (Basic Timer) ■ Reload timer with FFFFh in Free Running Mode (Free-running Timer) ■ Timer can be started and stopped by the TINx input pin (Gate Function) ■ The TOUTx pin changes polarity each time the timer underflows (Pulse Output Function)

11.6.1.1 Timer Mode Underflow

The SEC2410/SEC4410 timers operating in Timer mode can underflow in two different ways. One method, the Reload mode shown in Figure 11.2, is to reload the value programmed into the Reload register and continue counting from this value. The second method, Free Running mode Figure 11.3, is to set the timer to FFFFh and continue counting from this value. The underflow behavior is controlled by the RLOAD bit in the Timer Control Register.

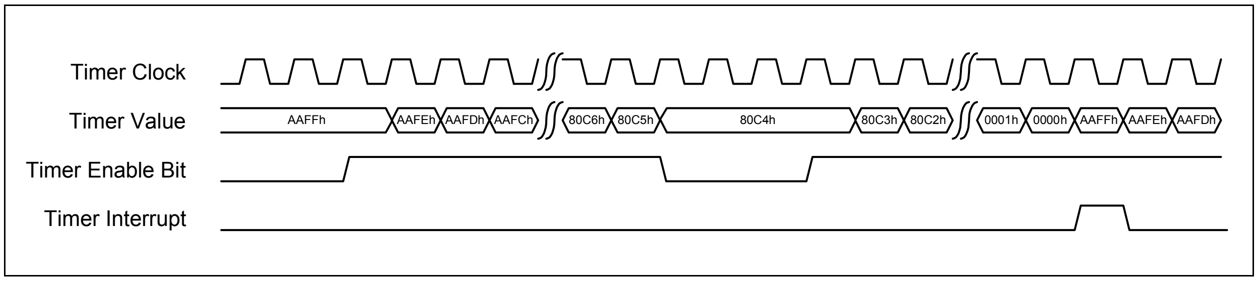


Figure 11.2 Reload Mode Behavior

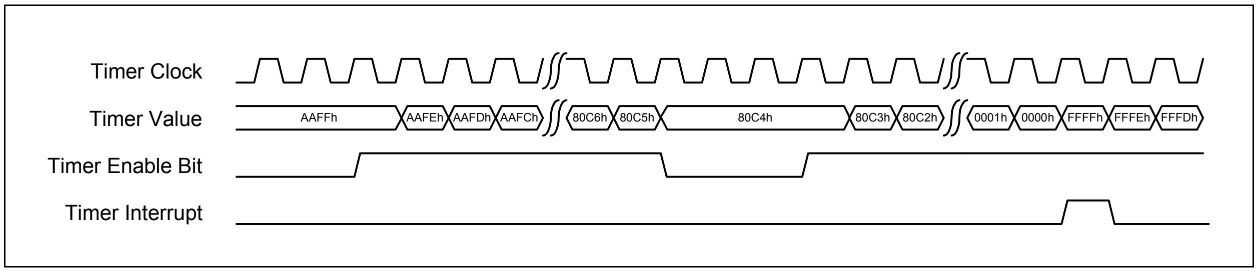


Figure 11.3 Free Running Mode Behavior

11.6.1.2 Timer Gate Function

The TINx pin on each timer can be used to pause the timer's operation when the timer is running. The timer will stop counting when the TINx pin is deasserted and count when the TINx pin is asserted. Figure 11.4 shows the timer behavior when the TINx pin is used to gate the timer function. The UPDN bit is used to enable and disable the Timer Gate function when in the Timer mode.

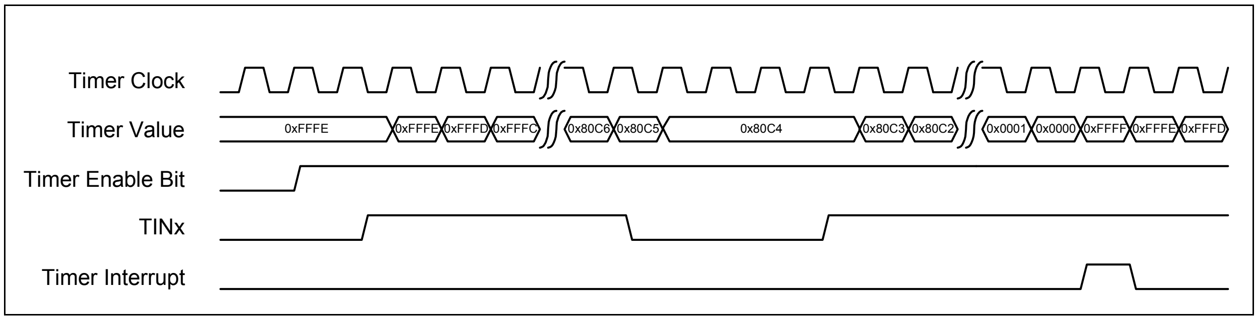


Figure 11.4 Timer Gate Operation

11.6.1.3 Timer Mode Pulse Output

The four Timers can be used to generate a periodic output pulse. The output pulse changes state each time the timer underflows. The output is also cleared when the EN bit is cleared. Figure 11.5 shows the behavior of the TOUTx pin when it is used as a pulse output pin.

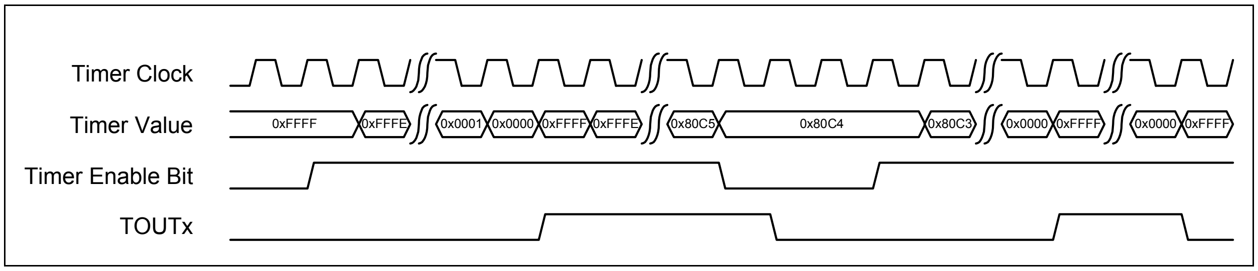


Figure 11.5 Timer Pulse Output

11.6.2 Event Mode

Event mode is used to count events that occur external to the timer. The timer can be programmed to count the overflow output from the previous timer or an edge on the TINx pin. The direction the timer counts in Event mode is controlled by the UPDN bit in the Timer Control Register. When the timer is in Event mode, the TOUTx signal can be used to generate a periodic output pulse when the timer overflows or underflows. Figure 11.5 illustrates the pulse output behavior of the TOUTx pin in event mode when the timer underflows.

The timer can be programmed using the Clock and Event Control register to respond to the following events using the EVENT bits and the bits: rising edge of TINx, falling edge of TINx, rising and falling edge of TINx, rising edge of overflow input, falling edge of the overflow input, and the rising and falling edges of the overflow input.

Table 11.4 Event Mode Operational Summary

ITEM	DESCRIPTION
Count Source	■ External signal input to TINx pin (effective edge can be selected by software) ■ Timer x-1 overflow
Timer Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Filter Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Count Operation	Up/Down Counter
Reload Operation	■ When the timer underflows: RLOAD = 1, timer reloads from Timer Reload Reg RLOAD = 0, timer rolls over to FFFFh. ■ When the timer overflows: RLOAD = 1, timer reloads from Timer Reload Reg RLOAD = 0, timer rolls over to 0000h.
Count Start Condition	Timer Enable is set (ENABLE = 1)
Count Stop Condition	Timer Enable is cleared (ENABLE = 0)
Interrupt Request Generation Timing	When timer overflows or underflows

Table 11.4 Event Mode Operational Summary (continued)

ITEM	DESCRIPTION
TINx Pin Function	Event Generation
TOUTx Pin Function	TOUT toggles each time the timer underflows/overflows (if enabled).
Read From Timer	Current count value can be read by reading the Timer Count Register
Write to Preload Register	After the firmware writes to the Timer Reload Register, asserting the RESET loads the timer with the new value programmed in the Timer Reload Register. Note: If the firmware does not assert RESET, the timer will automatically load the Timer Reload Register value when the timer underflows.
Selectable Functions	■ The direction of the counter is selectable via the UPDN bit. ■ Reload timer on underflow/overflow with programmed Preload value (Basic Timer) ■ Reload timer with FFFFh in Free Running Mode (Free-running Timer) ■ Pulse Output Function The TOUTx pin changes polarity each time the timer underflows or overflows.

11.6.2.1 Event Mode Operation

The timer starts counting events when the ENABLE bit in the Timer Control Register is set and continues to count until the ENABLE bit is cleared. When the ENABLE bit is set, the timer continues counting from the current value in the timer except after a reset event. After a reset event, the timer always starts counting from the value programmed in the Reload Register if counting down or from 0000h if counting up. Figure 11.6 shows an example of timer operation in Event mode. The RLOAD bit controls the behavior of the timer when it underflows or overflows.

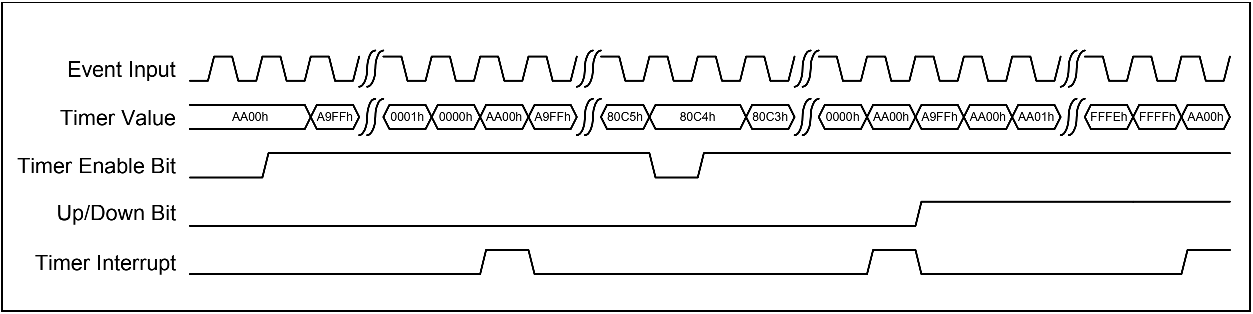


Figure 11.6 Event Mode Operation

11.6.3 One-Shot Mode

The One-Shot mode of the timer is used to generate a single interrupt to the EC after a specified amount of time. The timer can be configured to start using the ENABLE bit (Figure 11.7) or on a timer overflow event from the previous timer. See Section 11.8.2: *Timer x Clock and Event Control Register* on page 101 for configuration details. The ENABLE bit must be set for an event to start the timer. The ENABLE bit is cleared one clock after the timer starts. The timer always starts from the value in the Reload Register and counts down in One-Shot mode.

Table 11.5 One Shot Mode Operational Summary

ITEM	DESCRIPTION
Timer Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101

Table 11.5 One Shot Mode Operational Summary (continued)

ITEM	DESCRIPTION
Filter Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Count Operation	Down Counter
Reload Operation	When the timer underflows the timer will stop. When the timer is enabled timer starts counting from value programmed in Timer Reload Register. (RLOAD has no effect in this mode)
Count Start Condition	Setting the ENABLE bit to 1 starts One-Shot mode. The timer clock automatically clears the enable bit one timer tick later. **Note:** One-Shot mode may be enabled in Event Mode. In Event mode an overflow from the previous timer is used for timer tick rate.
Count Stop Condition	■ Timer is reset (RESET = 1) ■ Timer underflows
Interrupt Request Generation Timing	When an underflow occurs.
TINx Pin Function	One Shot External input
TOUTx Pin Function	The TOUTx pin is asserted when the timer starts and de-asserted when the timer stops
Read From Timer	Current count value can be read by reading the Timer Count Register
Write to Preload Register	After the firmware writes to the Timer Reload Register, asserting the RESET loads the timer with the new value programmed in the Timer Reload Register. Note: If the firmware does not assert RESET, the timer will automatically load the Timer Reload Register value when the timer underflows.
Selectable Functions	■ Pulse Output Function The TOUTx pin is asserted when the timer starts and de-asserted when the timer stops.

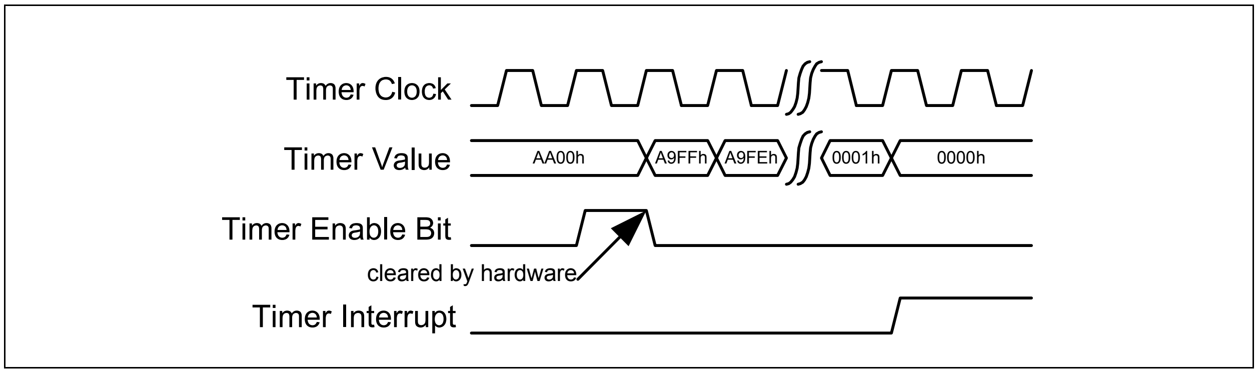


Figure 11.7 Timer Start Based on ENABLE Bit

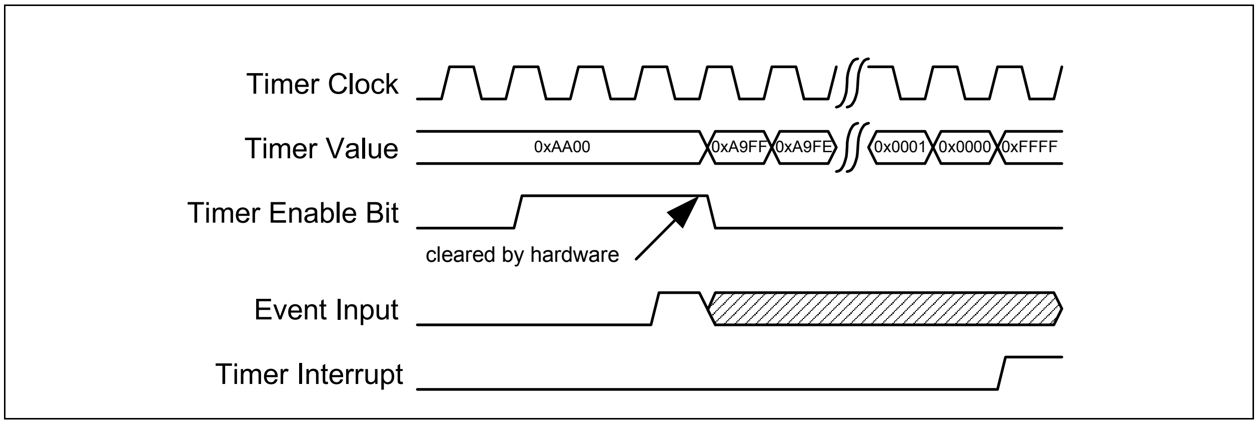


Figure 11.8 Timer Start Based on External Event

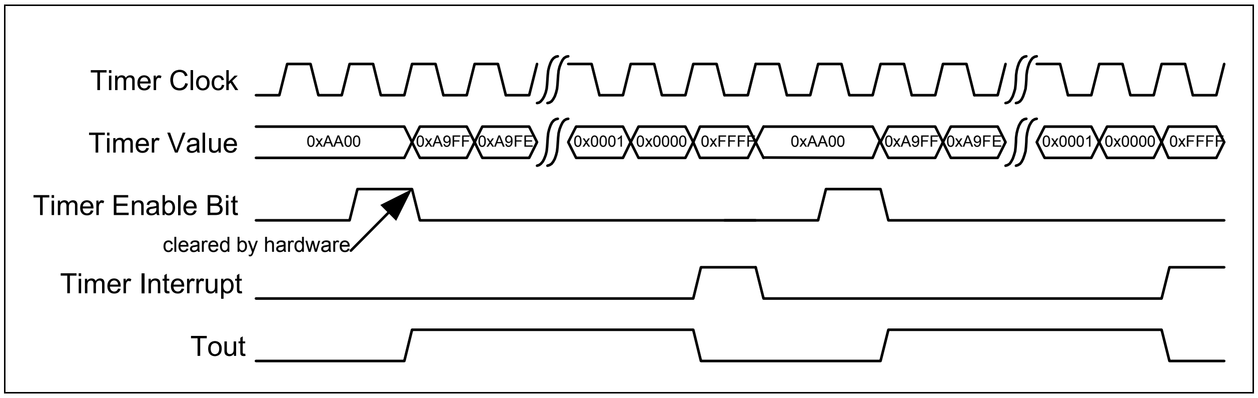


Figure 11.9 One Shot Timer with Pulse Output

11.6.4 Measurement Mode

The Measurement mode is used to measure the pulse width or period of an external signal. An interrupt to the EC is generated after each measurement or if the timer overflows and no measurement occurred. The timer measures the pulse width or period by counting the number of clock between edges on the TINx pin. The timer always stars counting at zero and counts up to 0xFFFF. The accuracy of the measurement depends on the speed of the clock being used. The speed of the clock also determines the maximum pulse width or period that can be detected.

Table 11.6 Measurement Mode Operational Summary

ITEM	DESCRIPTION
Timer Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Filter Clock Frequencies	This mode supports all the programmable frequencies listed in Table 11.10: *Timer Clock Frequencies* on page 101
Count Operation	■ Up Count ■ At measurement pulse's effective edge, the count value is transferred to the Timer Reload Register and the timer is loaded with 0000h and continues counting.

Table 11.6 Measurement Mode Operational Summary (continued)

ITEM	DESCRIPTION
Count Start Condition	■ Timer enable is set (ENABLE = 1)
Count Stop Condition	■ Timer is reset (RESET = 1) ■ Timer overflows ■ Timer enable is cleared (ENABLE = 0)
Interrupt Request Generation Timing	■ When timer overflows ■ When a measurement pulse's effective edge is input. (An interrupt is not generated on the first effective edge after the timer is started.)
TINx Pin Function	Programmable Input port or Measurement input
Read From Timer	When the Timer x Reload Register is read it indicates the measurement result from the last measurement made. The Timer x Reload Register reads 0000h if the timer overflows before a measurement is made.
Write to Timer	Timer x Reload Register is Read-Only in Measurement mode

11.6.4.1 Pulse Width Measurements

The timers measure pulse width by counting the number of timer clocks since the last rising or falling edge of the TINx input. To measure the pulse width of a signal on the TINx pin, the bits in the Clock and Event Control Register, must be set to start counting on rising and falling edges. The timer starts measuring on the next edge (rising or falling) on the TINx pin after the ENABLE bit is set. The Reload register stores the result of the last measurement taken. If the timer overflows, 0x0000 is written to the Reload register and the ENABLE bit is cleared stopping the timer. Figure 11.10 shows the timer behavior when measuring pulse widths.

The timer will not assert an interrupt in Pulse Measurement mode until the timer detects both a rising and a falling edge.

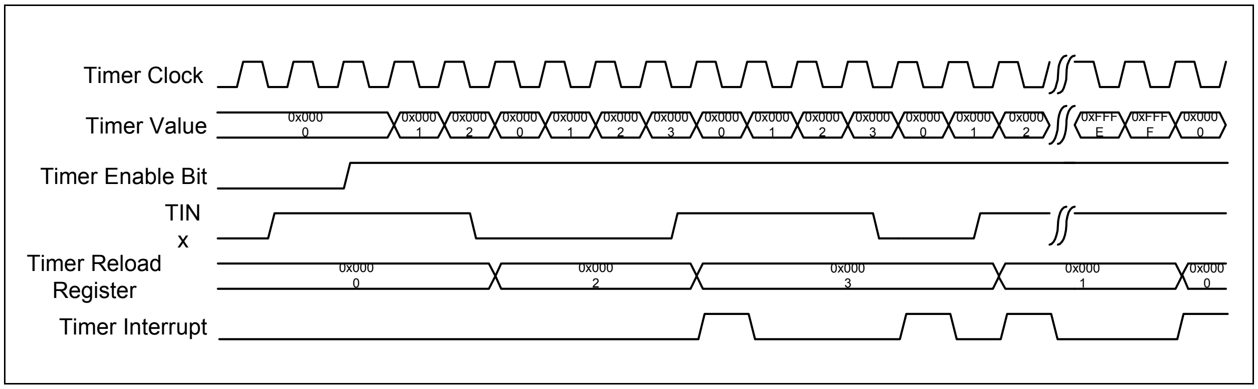


Figure 11.10 Pulse Width Measurement

11.6.4.2 Period Measurements

The timers in the SEC2410/SEC4410 measure the period of a signal by counting the number of timer clocks between either rising or falling edges of the TINx input. The measurement edge is determined by the bits in the Clock and Event Control Register. The timer starts measuring on the next edge (rising or falling) on the TINx pin after the ENABLE bit is set. The reload register stores the result of the last measurement taken. If the timer overflows, 0x0000 is written to the reload register. Figure 11.11 shows the timer behavior when measuring the period of a signal.

The timer will not signal an interrupt in period measurement mode until the timer detects either two rising edges or two falling edges.

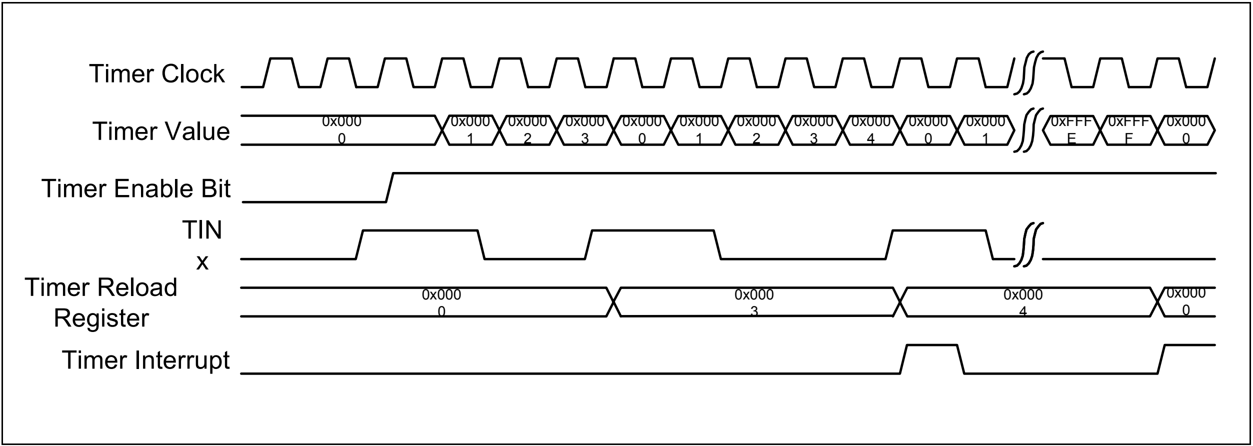


Figure 11.11 Pulse Period Measurement

11.7 16-Bit Counter/Timer Interface Register Summary

There are four instances of the 16-Bit Timer/Counter Block implemented in the SEC2410/SEC4410 enumerated as [0:3] with an overflow/underflow interface. Each instance of the 16-Bit Timer/Counter Block has its Base Address as indicated in Table 11.7.

Table 11.7 16-Bit Counter/Timer Interface Base Address Table

16-Bit Timer/Counter Block **INSTANCE**	**EC BUS ADDRESS OFFSET (BASE = 0X40020000)**
16-bit Timer.0	0x0C00
16-bit Timer.1	0x0C80 = 0x0C00 + 0x80
16-bit Timer.2	0x0D00 = 0x0C00 + 0x100
16-bit Timer.3	0x0D80 = 0x0C00h + 0x180

11.8 Detailed Register Descriptions

11.8.1 Timer x Control Register

Table 11.8 Timer x Control Register

TIMER_X_CTL (OFFSET 0X00 - RESET=0X0200)			**TIMER X CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:13	Reserved	R	Always read ‘0’
12	TIMERx_CLK_REQ	R	The TIMERX_CLK_ bit is a read-only bit that reflects the state of the TIMERx_CLK_REQ output signal. 0=Indicates the 60.00 MHz clock domain can be turned 'off' when appropriate 1=Indicates the 60.00 MHz clock domain is required to be 'on.' **Note:**
11	SLEEP_ENABLE	R	This bit is a read-only bit that reflects the state of the SLEEP_ENABLE signal. This signal stops the timer and resets the internal counter to the value in the Timer Reload Register. Once the timer is disabled, the TIMERX_CLK_ bits will be deasserted. This signal does not clear the Timer Enable bit if it is set. If the timer is enabled, the counter will resume operation when the SLEEP ENABLE signal is deasserted. The timer is held in reset as long as the input signal is asserted. 0=Normal timer operation. In Normal Mode, the timer operates as configured. When returning from a sleep mode, if enabled, the counter will be restarted from the preload value. 1=Sleep Mode Requested. In Sleep Mode, the timer is reset, the counter is disabled, and the TIMERx_CLK_REQ outputs are deasserted.
10	TOUT Polarity	R/W	This bit determines the polarity of the TOUT signal. In timer modes that toggle the TOUT signal, this polarity bit will not have a perceivable difference, except to determine the inactive state. in One-Shot mode this determines if the pulsed output is active high or active low. 0=Active high (default) 1=Active low
9	PD	R/W	Power Down. 0=The timer is in a running state (default). 1=The timer is powered down and all clocks are gated.
8	Filter Bypass	R/W	Filter Bypass permits TINx to bypass the noise filter and go directly into the timer 0=Filter enabled on TINx (default) 1=Filter bypassed on TINx
7	RLOAD	R/W	Reload Control. This bit controls how the timer is reloaded on overflow or underflow in Event and Timer modes, it has no effect in One Shot mode. 0=Roll timer over to FFFFh and continue counting when counting down and rolls over to 0000h and continues counting when counting up. 1=Reload timer from Timer Reload Register and continue counting.

TIMER_X_CTL (OFFSET 0X00 - RESET=0X0200)			**TIMER X CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
6	TOUT_EN	R/W	TOUT Enable 0=TOUT pin is pin in the inactive state (driven low) 1=TOUT function is enabled
5	UPDN	R/W	Up/Down. In Event mode this bit selects the timer count direction. Event Mode: 0=The timer counts down 1=The timer counts up Timer Mode: 0=TINx pin has no effect on the timer 1=TINx pin pauses the timer when deasserted
4	INPOL	R/W	Timer Input Polarity. This bit selects the polarity of the TINx input 0=TINx input is active low (inverted) 1=TINx input is active high (non-inverted)
3:2	MODE	R/W	Timer Mode Select - These bits control the timer mode. 00=Timer Mode 01=Event Mode 10=One Shot Mode 11=Measurement Mode
1	RESET	R/W	RESET: Timer Reset - This bit stops the timer and resets the internal counter to the value in the Timer Reload Register. This bit also clears the Timer Enable bit if it is set. This bit is self clearing after the timer is reset. Firmware must poll this RESET bit. 0=Normal timer operation 1=Timer reset **APPLICATION NOTE:** When the RESET takes effect interrupts are blocked. Interrupts are not blocked until RESET takes effect and the ENABLE bit is cleared. If interrupts are not desired, firmware must mask interrupt in the interrupt block.
0	ENABLE	R/W	ENABLE: Timer Enable - This bit is used to start and stop the timer. This bit does not reset the timer count but does reset the timer pulse output. This bit will be cleared when the timer starts counting in One-Shot mode. 0=Timer is disabled 1=Timer is enabled **Note:** This bit is cleared after the RESET cycle is done. Firmware must poll the RESET bit.

11.8.2 Timer x Clock and Event Control Register

Table 11.9 Timer X Clock and Event Control Register

TIMER_X_CLK_CTL (OFFSET 0X04, RESET=0X0000)			**TIMER CLOCK AND EVENT CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:12	Reserved	R	Always read ‘0’
11:8	FCLK	R/W	Filter Clock Select, is used to determine the clock source for the TINx noise filter. Available frequencies are the same as the Timer clock and are shown in Table 11.10
7	EVENT	R/W	This bit is used to select the count source when the timer is operating in event mode. 0=Timer x-1 overflow is count source 1=TINx is count source
6:5	EDGE	R/W	Edge Type Select. These bits are used to select the edge type that the timer counts. In One-Shot mode these bits select which edge starts the timer. See Table 11.11
4	Reserved	R	Always read ‘0’
3:0	TCLK	R/W	This field is the Timer Clock Select, used to determine the clock source to the 16-bit timer. Available frequencies are shown in Table 11.10

Table 11.10 Timer Clock Frequencies

TIMER CLOCK SELECT	**FREQUENCY SELECTED**
0000	60.00MHz
0001	30.00MHz
0010	15.00MHz
0011	7.5MHz
0100	3.75MHz
0101	1.88MHz
0110	0.94MHz
0111	469KHz
1xxx	Reserved

Table 11.11 Edge Operation

MODE	EDGE	OPERATION
EVENT	00	Counts falling edges
	01	Counts rising edges
	10	Counts rising and falling edges
	11	No event selected
ONE SHOT	00	Starts counting on a falling edge
	01	Starts counting on a rising edge
	10	Starts counting on a rising or falling edge
	11	Start counting when the Enable bit is set
MEASUREMENT	00	Measures the time between falling edges
	01	01=Measures the time between rising edges
	10	Measures the time between rising edges and falling edges and the time between falling edges and rising edges
	11	No event selected

11.8.3 Timer x Reload Register

Table 11.12 Timer X Count Register

TIMER_X_RELOAD (OFFSET 0X08, RESET=0XFFFF)			**TIMER RELOAD REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	TIMER_RELOAD	R/W	The Timer Reload register is used in Timer and One-Shot modes to set the lower limit of the timer. In Event mode the Timer Reload register sets either the upper or lower limit of the timer depending on if the timer is counting up or down. Valid values are 0001h - FFFFh. If the timer is running, the reload value will not be updated until the timer overflows or underflows **Note:** Programming a 0000h as a preload value is not a valid count value.

11.8.4 Timer x Count Register

Table 11.13 Timer X Count Register

TIMER_X_COUNT (OFFSET 0X0C, RESET=0XFFFF)			**TIMER COUNT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	TIMER_COUNT	R	The Timer Count register returns the current value of the timer in all modes.

Chapter 12 SPI Controller

12.1 Overview

The SPI controller has three basic modes of operation. When operating as a memory bus to a SPI ROM, it takes System AMBA bus accesses in the range 0x6000_0000 through 0x60FF_FFFF (16MBytes), and convert them to SPI ROM accesses, construct the data, and provide data back to the ARM along with the ready signal at the appropriate time.

In parallel with SPI ROM reading hardware is a 32 byte cache that keeps track of data that has been fetched.

The second mode of operation is for all SPI operations that are not fast reads or Trace FIFO accesses. In this mode, the firmware is responsible for setting up a command buffer, and control registers. The firmware then fires the command by setting a 'GO' bit. The firmware is also responsible for parsing the response from the SPI slave. These registers exist in the range of 0x6100_0000 to 0x6100_03FF.

The last mode of operation is for debugging. The firmware writes to System AHB addresses 0x6100BFFE and 0x6100BFFF to send out trace message to SMSC's trace FIFO board. The SPI controller sends out accesses to these locations as special messages that are ignored by the SPI ROM, but are intercepted by the debugging hardware.

The SPI interface is always enabled after reset. It can be disabled by setting the SPI_DISABLE bit in the UTIL_CONFIG1 register.

12.2 DEVICE OPERATION INSTRUCTIONS

There is only one operation supported automatically in hardware: FAST_READ. All instructions associated with Automatic Address Increment (AAI) are supported. Everything else is handled through firmware intervention.

Table 12.1 SPI opcodes

INSTRUCTION	DESCRIPTION	OP CODE CYCLE	ADDRESS CYCLE(S)	DUMMY CYCLE(S)	DATA CYCLE(S)	TOTAL	RESP
WRSR	Write Status Register	0x01	0	0	1	2	FW
Byte_program	To program one Data Byte	0x02	3	0	1	5	FW
READ	Read Slow Mode	0x03	3	0	1 to	5 to ¥	N/A
WRDI	Write Disable	0x04	0	0	0	1	FW
RDSR	Read Status Register	0x05	0	0	1 to	2 to	FW
WREN	Write Enable	0x06	0	0	0	1	FW
FAST_READ	Read Fast Mode	0x0B	3	1	1 to	6 to	HW
SCTR_ERASE	4 KByte Sector Erase	0x20 0xD7	3	0	0	4	FW
DUAL FAST_READ	Dual Read Fast Mode	0x3B	3	1	1 to	6 to	HW

Table 12.1 SPI opcodes (continued)

INSTRUCTION	DESCRIPTION	OP CODE CYCLE	ADDRESS CYCLE(S)	DUMMY CYCLE(S)	DATA CYCLE(S)	TOTAL	RESP
EWSR	Enable Write Status Register	0x50	0	0	0	1	FW
32BLK_ERASE	32 KByte Block Erase	0x52	3	0	0	4	FW
CHIP_ERASE	Erase full memory array	0x60 0xC7	0	0	0	1	FW
EBSY	Enable SO to output Busy during AAI programming	0x70	0	0	0	1	N/A
DBSY	Disable SO to output Busy during AAI programming	0x80	0	0	0	1	N/A
RDID	Read ID	0x90	3	0	1 to	5 to	FW
JEDEC_ID	JEDEC ID read	0x9F	0	0	3 to	4 to	FW
RDCR	Read Config Register	0xA1	0	0	1	2	FW
RDES	Read Electronic Signature	0xAB	3	0	1 to	5 to	FW
AAI_PROGRAM	Auto Address Increment programming	0xAD	3	0	2 to	6 to	N/A
64BLK_ERASE	64 KByte Block Erase	0xD8	3	0	0	4	FW
WRCR	Write Config Register	0xF1	0	0	1	2	FW

1. One bus cycle is eight clock periods.
2. Address bits above the most significant bit of each density should be set to 0x00.

12.3 Operation of the Hi-Speed Read Sequence

The SPI controller handles code reads going out to the SPI ROM Address automatically. When the controller detects a read, the controller drops the **SPI_CE**, and puts out a 0x0B, followed by the 24-bit address. The SPI controller then puts out a DUMMY byte. The next eight clocks clock in the first byte. When the first byte is clocked in a ready signal is sent back to the processor, and the processor gets one byte.

After the processor gets the first byte, its address will change. If the address is one more than the last address, the SPI controller will clock out one more byte. If the address in anything other than one more than the last address, the SPI controller will terminate the transaction by taking **SPI_CE** high. As long as the addresses are sequential, the SPI Controller will keep clocking in data.

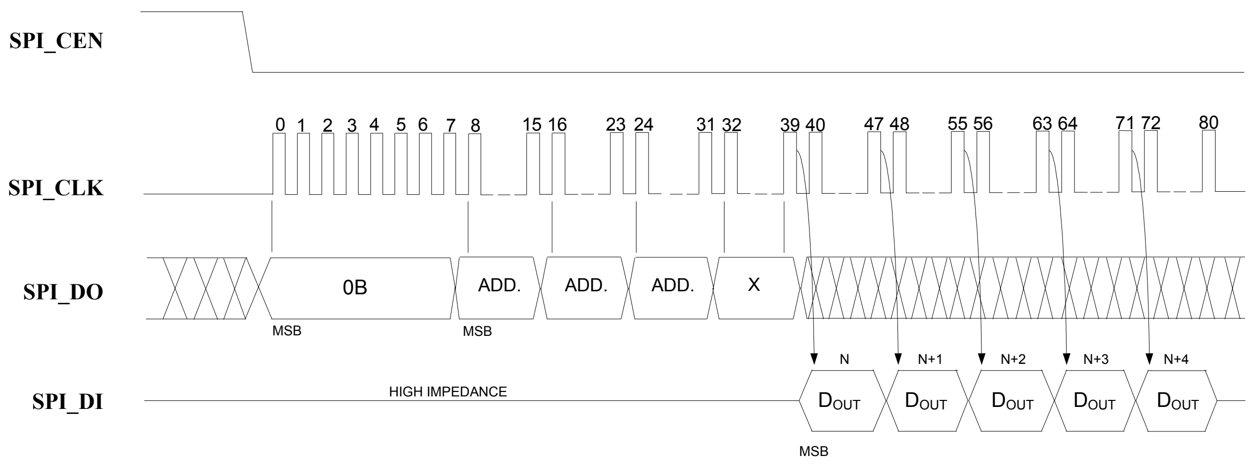


Figure 12.1 SPI Hi-Speed Read Sequence

12.4 Operation of the Dual Hi-Speed Read Sequence

The SPI controller also supports dual data mode. When configured in dual mode, the SPI controller will automatically handle reads going out to the SPI ROM. When the controller detects a read, the controller drops the **SPI_CEN**, and puts out a 0x3B (the value must be programmed into the SPI_FR_OPCODE Register), followed by the 24-bit address. The SPI controller then puts out a DUMMY byte. The next four clocks clock in the first byte. The data appears two bits at a time on data out and data in. When the first byte is clocked in a ready signal is sent back to the processor, and the processor gets one byte.

After the processor gets the first byte, the address will change. If the address is one more than the last address, the SPI controller will clock out one more byte. If the address in anything other than one more than the last address, the SPI controller will terminate the transaction by taking **SPI_CE** high. As long as the addresses are sequential, the SPI Controller will keep clocking in data.

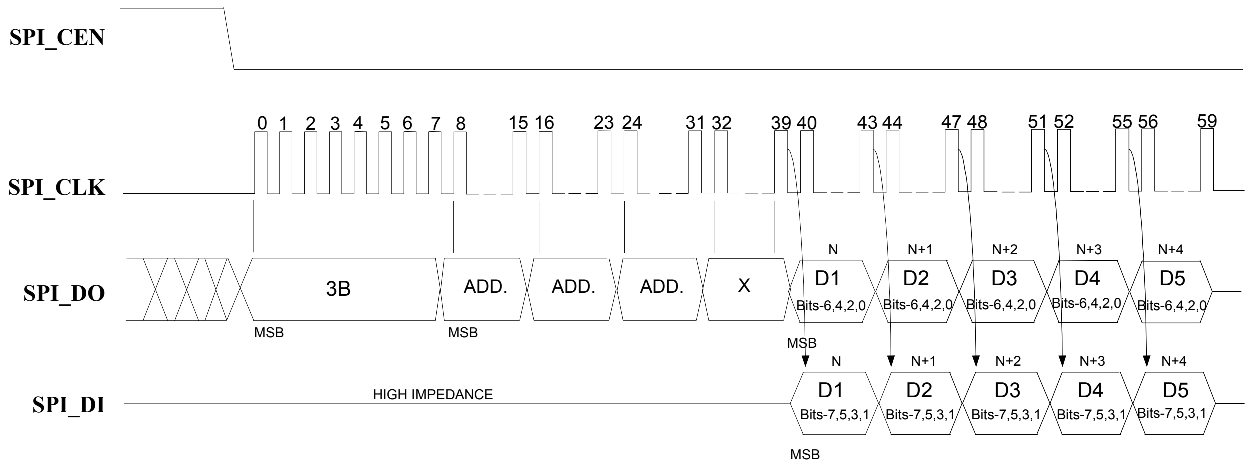


Figure 12.2 SPI Dual Hi-Speed Read Sequence

12.5 32 Byte Cache

There is a 32-byte pipeline cache, and associated with the cache is a base address pointer and a length pointer. Once the SPI controller detects a jump, the base address pointer is initialized to that address. As each new sequential data byte is fetched, the data is written into the cache, and the length is incremented. If the sequential run exceeds 32 bytes, the base address pointer is incremented to indicate the last 32 bytes fetched. If the SEC2410/SEC4410 does a jump, and the jump is in the cache address range, the fetch is done in 1 clock from the internal cache instead of an external access.

12.6 Interface Operation to SPI Port When Not Doing Fast Reads

There is an 8-byte command buffer: SPI_CMD_BUF[7:0]; an 8-byte response buffer: SPI_RESP_BUF[7:0]; and a length register that counts out the number of bytes: SPI_CMD_LEN. Additionally, there is a self-clearing **GO** bit in the SPI_CTL Register. Once the **GO** bit is set, the SEC2410/SEC4410 drops **SPI_CE**, and starts clocking. It will put out SPI_CMD_LEN X 8 number of clocks. After the first byte, the COMMAND, has been sent out, and the **SPI_DI** is stored in the SPI_RESP buffer. If the SPI_CMD_LEN is longer than the SPI_CMD_BUF, don't cares are sent out on the **SPI_DO** line.

12.6.1 This mode is used for program execution out of internal RAM or ROM.ERASE EXAMPLE

To perform a SCTR_ERASE, 32BLK_ERASE, or 64BLK_ERASE, the SEC2410/SEC4410 writes 0x20, 0x52, or 0xD8, respectively to the first byte of the command buffer, followed by a 3-byte address. The length of the transfer is set to 4 bytes. To do this, the SEC2410/SEC4410 first drops **SPI_CE**, then counts out 8 clocks. It then puts out the 8 bits of command, followed by 24 bits of address of the location to be erased on the **SPI_DO** pin. When the transfer is complete, the **SPI_CEN** goes high, while the **SPI_DI** line is ignored in this example.

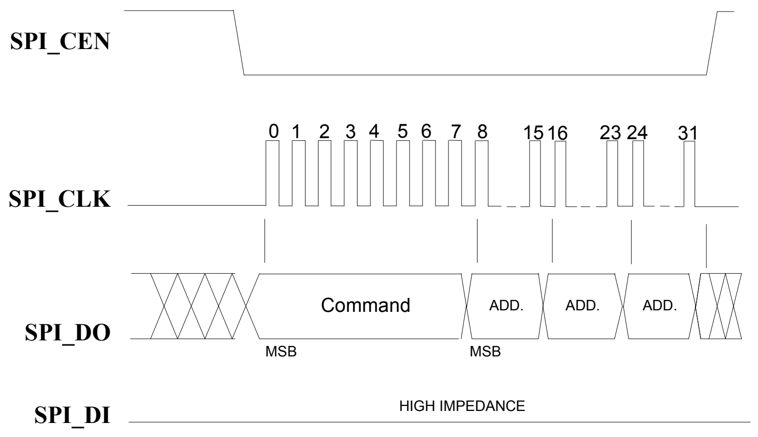


Figure 12.3 SPI Erase Sequence

12.6.2 BYTE PROGRAM EXAMPLE

To perform a Byte Program, the SEC2410/SEC4410 writes 0x02 to the first byte of the command buffer, followed by a 3-byte address of the location that will be written to, and one data byte. The length of the transfer is set to 5 bytes. The SEC2410/SEC4410 first drops **SPI_CE**, 8 bits of command are clocked out, followed by 24 bits of address, and one byte of data on the **SPI_DO** pin. The **SPI_DI** line is not used in this example.

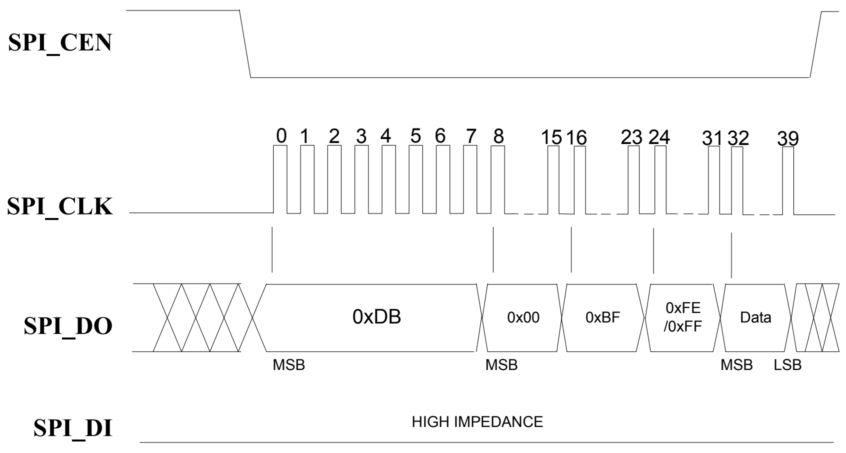


Figure 12.4 SPI Byte Program

12.6.3 COMMAND ONLY PROGRAM EXAMPLE

To perform a single byte command such as the following:

- WRDI
- WREN
- EWSR
- CHIP_ERASE
- EBSY
- DBSY

The SEC2410/SEC4410 writes the opcode into the first byte of the SPI_CMD_BUF and the SPI_CMD_LEN is set to one. The SEC2410/SEC4410 first drops **SPI_CE**, then 8 bits of the command are clocked out on the **SPI_DO** pin. The **SPI_DI** is not used in this example.

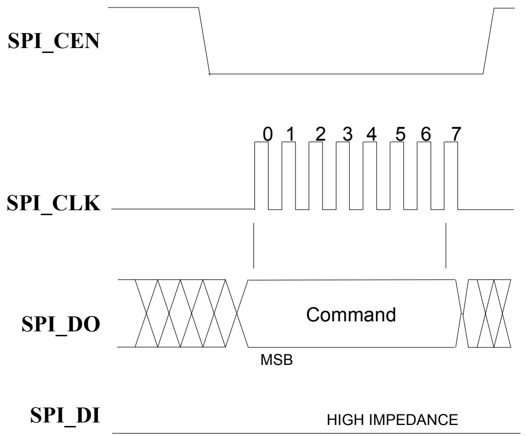


Figure 12.5 SPI Command Only Sequence

12.6.4 JEDEC-ID READ EXAMPLE

To perform a JEDEC-ID command, the SEC2410/SEC4410 writes 0x9F into the first byte of the SPI_CMD_BUF and the length of the transfer is 4 bytes. The SEC2410/SEC4410 first drops **SPI_CE**, then 8 bits of the command are clocked out, followed by the 24 bits of dummy bytes (due to the length being set to 4) on the **SPI_DO** pin. When the transfer is complete, the **SPI_CEN** goes high. After the first byte, the data on **SPI_DI** is clocked into the SPI_RSP_BUF. At the end of the command, there are three valid bytes in the SPI_RSP_BUF. In this example, 0xBF, 0x25, 0x8E.

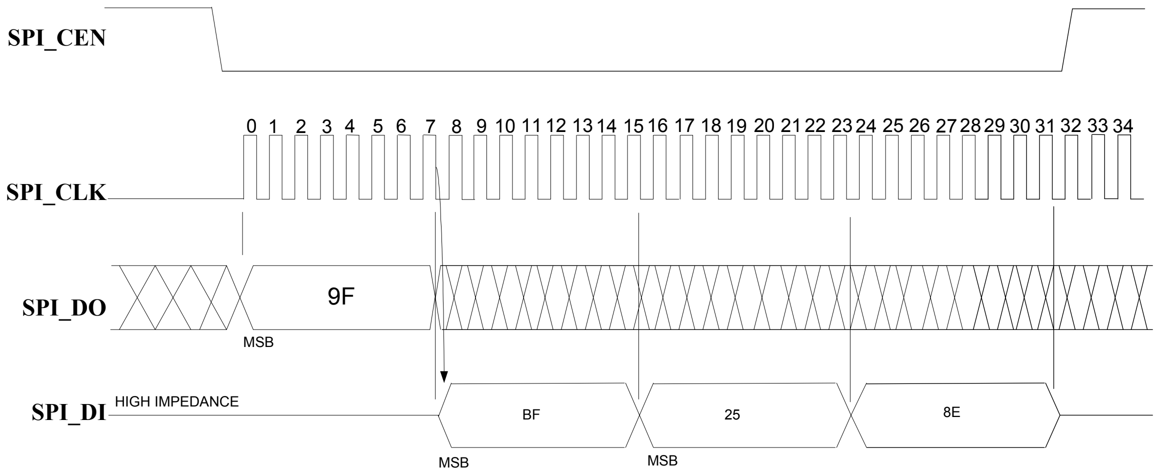


Figure 12.6 SPI JEDEC-ID Sequence

12.6.5 TRACE FIFO WRITE EXAMPLE

To perform a Trace FIFO write, the FW writes to either System AHB address 0x6100BFFE and 0x6100BFFF. The SPI controller treats these as special cases. For these two addresses, the SPI controller puts out the debug opcode, from the SP_TF_OPCODE register. It then puts out a 24 bit address, followed by the data from the System AHB register.

The writes go out as unrecognized commands to the ROM which will ignore them.

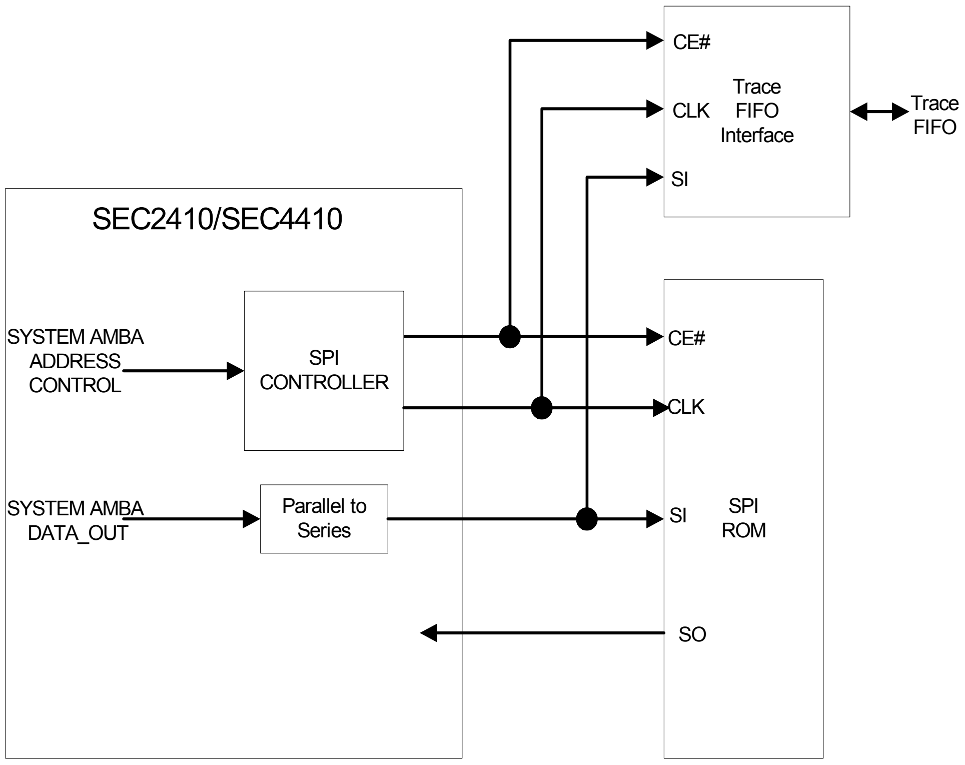


Figure 12.1 SPI Trace FIFO Write operation

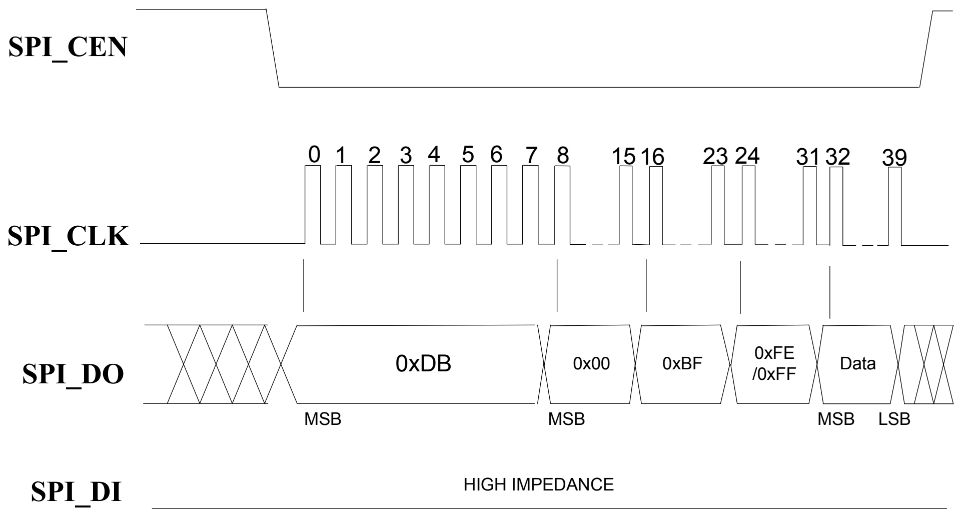


Figure 12.2 SPI Trace FIFO write example

12.6.6 Arbitrary length SPI access example

To do a SPI access of arbitrary length:

1. Set the MODE_SEL to the desired mode.
2. Set the SPI_CMD_LEN to 1.
3. Set the FORCE_CE bit in SPI_CTL. This forces the chip enable low.
4. Write the output byte to SPI_CMD_BUF
5. Hit the GO bit.
6. When the GO bit clears read the response from the SPI_RSP
7. If more data, Write the next byte to SPI_CMD_BUF and go to (4)
8. Once all data is done clear the FORCE_CE bit, this releases chip enable.

A length of one was shown for clarity. Mutliple bytes can be done at a time.

12.7 SPI Registers

All SPI control registers are at offset 0x6100_0000 on the system AHB bus. Values shown in registers are off

Table 12.2 SPI Mode Control Register

SPI_CTL (0X0000 - RESET=0X02)			SPI MODE CONTROL
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	SPI_SPEED	R	This bit reflects the strap option of the SPI_SPEED option during reset. This is to allow the firmware to know what speed it is operating at. 0: 30 Mhz 1: 60 Mhz
6:5	Reserved	R	Always read ‘0’
4	FORCE_CE	R/W	When this bit is set, it forces the SPI chip enable low. This bit should only be set when using the SPI command buffer. It should never be set when doing direct accesses from the AHB bus.
3	DUAL_OUT_EN	R/W	0:Dual output disabled for fast reads 1:Dual output enabled for fast reads.
2	MODE_SEL	R/W	This set the SPI clock mode 0: Mode 0 1: Mode 3
1	CACHE_EN	R/W	Enable the SPI cache
0	GO	R/W	This is a self clearing bit. Setting this bit will cause the SPI transaction to initiate.

Table 12.3 SPI Command Length Register

SPI_CMD_LEN (0X0001 - RESET=0X00)			SPI COMMAND LENGTH
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	CMD_LEN[7:0]	R/W	This is the length of the SPI transaction length for firmware initiated transactions.

Table 12.4 SPI Trace Fifo opcode

SPI_TF_OPCODE (0X0002 - RESET=0X00)			SPI TRACE FIFO OPCODE
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TF_OPCODE	R/W	This is the opcode used when the processor does a write to 0x6100BFFE or 0x6100BFFF. Use the value of 0xDB

Table 12.5 Fast Read Opcode

SPI_FR_OPCODE (0X0003 - RESET=0X0B)			**SPI FAST READ OPCODE**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	FR_OPCODE	R/W	This is the opcode used when the processor does a fast read 0x0B: Single output read 0x3B: Dual output read

Table 12.6 SPI Command Buffer

SPI_CMD_BUF (0X0008~0X000F - RESET=0X00)			**SPI COMMAND BUFFER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	SPI_CMD_BUF[0:7]	R/W	This buffer is used by processor to store outgoing SPI commands. See behavioral description.

Note: First byte to go out is SPI_CMD_BUF[0] at offset location 0x0008.

Table 12.7 SPI Response Buffer

SPI_RSP_BUF (0X0010~0X0017 - RESET=0X00)			**SPI RESPONSE BUFFER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	SPI_RSP_BUF[0:7]	R/W	This buffer is used by processor to store incoming SPI responses. See behavioral description.

Note: First byte to be written is SPI_RSP_BUF[0] at offset location 0x0010

Table 12.8 SPI PAD Current Control Register

PAD_CTL_SPI (0X0018 - RESET=0X00)			**PAD CURRENT CONTROL**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.
1:0	SEL	R/W	00 = 6 mA Operation (default) 01 = 8 mA Operation 10 = 10 mA Operation 11 = 12 mA Operation **Note:** This register only has effect when the SPI interface is enabled.

12.8 SPI Timing

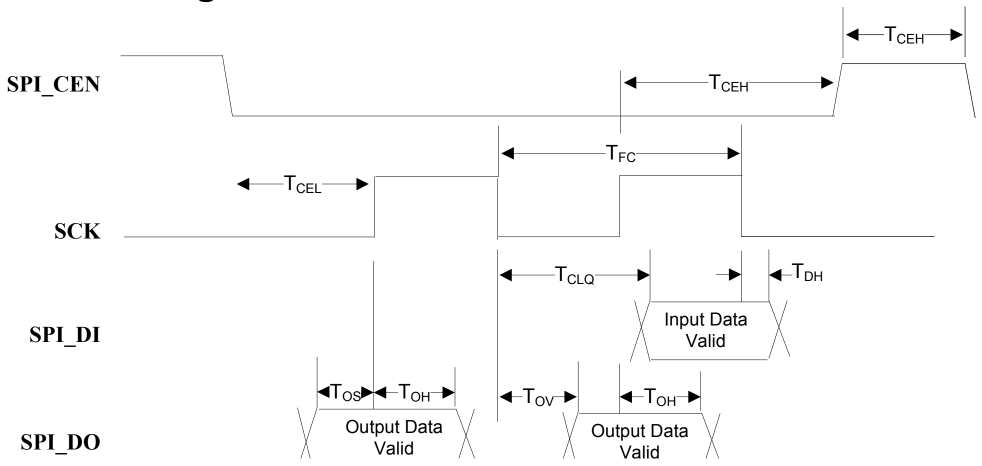


Figure 12.7 SPI Timing

Table 12.9 SPI Timing 60 MHz Operation

Name	Parameter	Min	Max	Unit
T_{FC}	Clock Frequency		60	MHz
T_{CEH}	Chip Enable High Time	50		ns
T_{CLO}	Clock to Input Data		9	ns
T_{DH}	Input Data Hold Time	0		ns
T_{OS}	Output Set up Time	5		ns
T_{OH}	Output Hold Time	5		ns
T_{OV}	Clock to Output Valid		4	ns
T_{CEL}	CE low to first clock	12		ns
T_{CEH}	Last clock to CE high	12		ns

Table 12.10 SPI Timing 30 MHz Operation

Name	Parameter	Min	Max	Unit
T_{FC}	Clock Frequency		30	MHz
T_{CEH}	Chip Enable High Time	100		ns
T_{CLO}	Clock to Input Data		13	ns
T_{DH}	Input Data Hold Time	0		ns
T_{OS}	Output Set up Time	5		ns
T_{OH}	Output Hold Time	5		ns
T_{OV}	Clock to Output Valid	4		ns
T_{CEL}	CE low to first clock	12		ns
T_{CEH}	Last clock to CE high	12		ns

Chapter 13 SmartCard Interface

13.1 Overview

SEC2410/SEC4410 has a SmartCard Interface based on the ISO/IEC 7816 Standard. The SmartCard resides on the System AHB bus at address offset 0x6200_0000

13.2 Interconnect to SmartCard terminal.

The interconnect to a SmartCard terminal come out of this block and is shown in the diagram below.

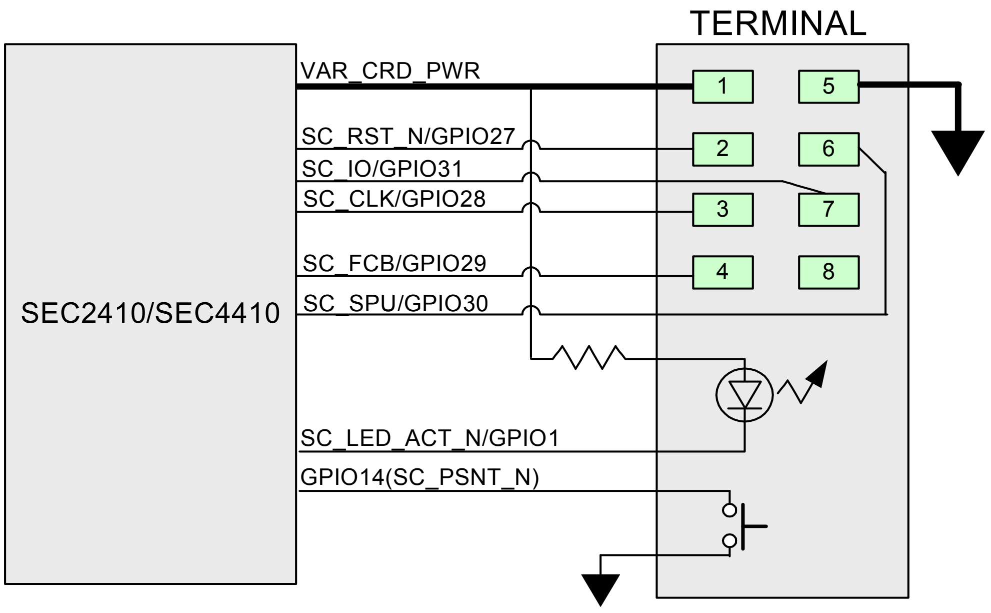


Figure 13.1 SmartCard Interconnect

13.3 General Description

The Smart Card Interface serves as the core of a Terminal, or Interface Device ("IFD"), which communicates with an insertable Smart Card, also called an "Integrated Circuit Card", or "ICC".

The Smart Card interface is a UART-like interface that supports the ISO 7816 asynchronous protocols named "T=0" and "T=1". It transmits and receives serial data via the SC_IO signal pin. Each byte transmitted or received is transferred as a character with a start bit, 8 data bits, a parity bit, and an amount of "guard time" (Stop bits) that depends on the protocol used and the declared characteristics of the card.

To initiate communication with the smart card, the smart card must be inserted into the terminal device. A mechanical or electrical sensor will detect this event, pulling the SC_PSNT_N(GPIO14) pin low to indicate that the electrical contacts are seated. The insertion of the card will cause a GPIO14 interrupt after the debounce period. If the system is in suspend state, the GPIO transition will cause the system to be woken up first, followed by the interrupt to the processor.

Once it is established that a Smart Card is present, firmware will use the VREG_CTL register to apply power to the card. Once the interface is powered, the terminal can initiate communication with the smart card by driving the SC_RST_N pin low. There are two types of resets: a cold reset and a warm reset. The cold reset sequence is used immediately after power is applied to the interface: it generates the SC_CLK output, sets the SC_IO pin as an input with a weak pull-up, and keeps the SC_RST_N pin low (its initial state) for a defined period of time after the clock starts running. The warm reset only affects the SC_RST_N pin, which is pulled low for a defined period of time: it requires that the interface already be powered and a steady clock be already applied to the card. Bits have been provided in the ICR register that may be controlled by software to initiate these sequences. When either of these resets terminates (SC_RST_N going high) the Smart Card will return a sequence of characters called the "Answer to Reset" or "ATR" message as defined by ISO 7816-3. The smart card is required to respond to a reset sequence as shown in the Cold Reset and Warm Reset timing diagrams (see Figure 13.8 and Figure 13.9 on page 131).

The first character of the ATR message, called "TS", is interpreted by hardware in SEC2410/SEC4410, determining the bit encoding convention used by the card (Direct or Inverse) as defined by ISO 7816-3, which defines the polarity and the order of the Data and Parity bits in the character. The TS byte, interpreted according to the convention it selects, is placed into the FIFO, and data received from that point onward is assembled according to the selected convention and loaded into the FIFO to be read by software.

The rest of the ATR response from the Smart Card returns the operational limits of the Smart Card. Software must interpret this response and set the SEC2410/SEC4410 runtime registers accordingly. During the ATR message, data will be received based on a default value of the bit time, called the Elementary Time Unit, abbreviated as “etu”. Two ATR parameters named F and D are used to define a new etu time. Once this is determined, software can program the BRG Divisor and the sampling rate for the Baud Rate generator accordingly. The hardware divides the 60MHz system clock, by the BRG Divisor and the sampling rate to determine the etu value (bit time). The SC_CLK frequency is generated by dividing the System clock by the SC_CLK_DIV DIVISOR field. Software will also set up the Extra Guard Time register (EGT), the Block Guard Time (BGT) register and the protocol mode ("T=0" or "T=1" mode) to set the required amount of guard time between character transmissions.

A negotiation phase called PPS may occur, or communication may begin immediately using the parameters provided by the card’s ATR message. In either case, all communication after the ATR message consists of individual “exchanges”, in which the IFD transmits a block of data and the ICC responds with a return message. For this reason, and because the response time from the ICC can be too short for software intervention, software will enable both the SEC2410/SEC4410 transmitter and receiver at the same time, and the receiver hardware will remain inactive until the transmission phase of the exchange has completed.

An additional “Stop Clock” feature has been provided to hold the SC_CLK output at a particular voltage level between exchanges, as may be allowed by the card for power savings. Clock switching is glitch free.

Hardware protocol timers, set according to default timings, will monitor the Smart Card interface during the Reset/ATR sequence for an unresponsive or defective card, based on the EMV, ISO and PC/SC timing requirements. If the ATR response is not received within the given time, or does not obey the required timings, a timer interrupt will result, and the software can take corrective action or initiate the deactivation sequence to stop and power-down the card.

After the ATR sequence, the same set of hardware timers are used, based on ATR parameters EGT, CWT, BWT, and/or WWT, to monitor timings for the subsequent data exchanges.

One of two protocols is selected, defined by a parameter T in the ATR message, and potentially negotiated in a PPS exchange. The protocol “T=0” is character-oriented, with parity error detection and re-transmission on a character-by-character basis. The protocol “T=1” is block-oriented, with an error-free link layer based on block re-transmission, resembling the X.25 communication standard. In the T=1 protocol, both individual character parity and a block check field are used to detect errors.

The SEC2410/SEC4410 FIFO is deep enough to hold an entire message of maximum length (259 bytes). It transmits data, pre-loaded into the FIFO, when the Transmit control bit is set by software. It immediately turns around, enabling the receiver to put data received back into the FIFO. The FIFO threshold interrupt is triggered by received data only, though a separate interrupt is available to signal when the Transmit phase has ended. The hardware has significant knowledge of the protocol being implemented, and can be set up to filter out bytes that would lead to a message longer than the FIFO depth.

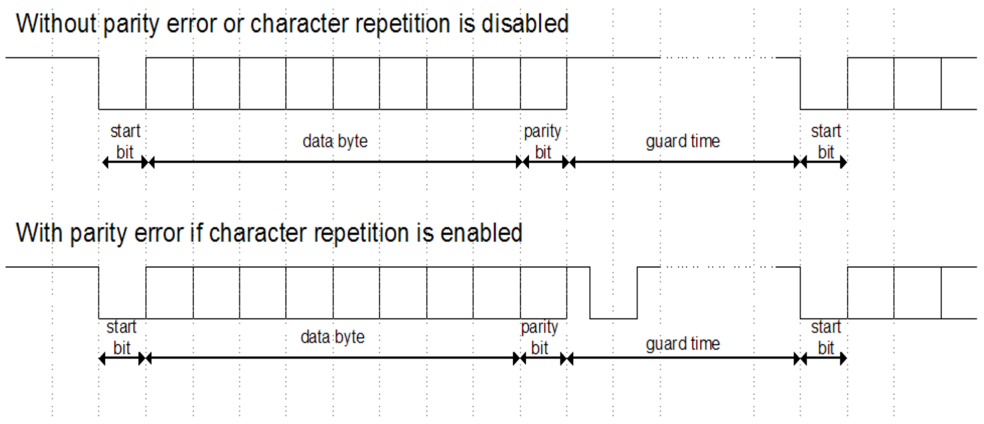
13.4 Character Framing

The SEC2410/SEC4410 meets the requirements for a character frame as defined by ISO 7816-3. The T=0 and T=1 protocol differ in the minimum amount of Guard Time: 2 etus for T=0, and 1 etu for T=1, which does not require a character-by-character Parity Error response.

Character parity is checked as each byte is received by hardware. If a parity error is detected when a byte is received, the Parity Error status bit will be set. This status bit can be polled by software, or it can be programmed to generate an interrupt and/or to deactivate the card in hardware. If character repetition is enabled (used in the T=0 protocol) the SEC2410/SEC4410 will pull the SC_IO line low following a received parity error, for the duration of 1 etu as defined by ISO 7816-3. While the SEC2410/SEC4410 is transmitting, if the card signals receipt with a parity error, the SEC2410/SEC4410 will repeat the character up to 4 additional times. Whether transmitting or receiving, failure after 5 transmissions of the same character will cause a Parity Error interrupt and/or hardware deactivation of the ICC.

Note: S/W should not try to initiate a RESYNCH until the transaction has completed, because the card may still be trying to send data to the IFD. Timeout timers and an "Activity Detection" bit are provided to assist software in this determination, in case of an error.

Figure 13.2 T=0 Mode Character Transmission and Repetition Diagram


Without parity error or character repetition is disabled
start bit
data byte
parity bit
guard time
start bit
With parity error if character repetition is enabled
start bit
data byte
parity bit
guard time
start bit

Note: Timing is measured in etu's. 1 etu = time to transmit 1 bit. The default etu is equal to 372/f, where f is the clock frequency.

Table 13.1 Character Frame Format

TRANSMISSION	DEFINITION
Start Bit	The I/O signal is held low for the duration of one etu after guard time before transmitting data

Table 13.1 Character Frame Format

TRANSMISSION	DEFINITION
Data Byte	The 8 bits immediately following the start bit that represents a single character byte. The logical value of the data byte transmitted is dependent on the convention selected by TS of the ATR. Direct Convention: logical ‘1’ equals VCC and bits are transmitted LSB first. Inverse Convention: logical ‘0’ equals VCC and bits are transmitted MSB first. Note: Data received is interpreted according to the Encoding Convention selected by the ICC.
Parity Bit	The Parity bit is used for error detection. It is used to provide Even Parity, operating on ‘1’ and ‘0’ as defined by the Convention. The Parity bit itself is also represented with the same polarity as the Data field, according to the selected Encoding Convention.
Guard Time	Guard Time is defined as the time between the transmission of the parity bit and the next start bit transmitted. During this time, both the transmitter and receiver release the bus. Only the receiver is permitted to pull the bus low during this time (in all except T=1) to indicate a parity error has occurred. Guard Time = Minimum Guard Time + Extra Guard Time (N); for $0 \leq N \leq 254$ Guard Time = Minimum Guard Time; for N=255. T=0 (including ATR and PPS) requires a minimum guard time of 2 etu’s. T=1 requires a minimum guard time of 1 etu. The minimum guard time is determined by whether T=0 or T=1 mode is chosen in the Protocol Mode register. Extra Guard Time (N) is programmable from 0 to 254 etu’s, as requested by the card in the ATR message. The default value is 0. The value of N received in the ATR should be directly programmed in the EGT register. If N=255 is programmed in the EGT register it will be treated the same as N=0.

13.5 Clocking and Baud Rate Generation

The frequency of the SC_CLK signal to the ICC, and the rate at which bits are transmitted and sampled, are determined from the 60Mhz system clock.

No other clock frequency is available in SEC2410/SEC4410.

13.5.1 Clock Rate Generation

The internal Clock Rate Generator determines the frequency of the clock to be provided to the ICC on the SC_CLK pin. This is expressed in the least-significant 6 bits of the SC_CLK_DIV register as a divisor on the system clock. To find the correct value, the Fi value is read from the card, and Fmax is determined. The divisor is chosen such that SC_CLK is the highest possible frequency without violating the Fmax parameter.

13.5.2 etu Rate Generation

The internal Baud Rate Generator (BRG) sets the duration of an etu (bit time). In the ATR message from the ICC, a divisor term (F) and a multiplier term (D) come from two 4-bit values Fi and Di. (If the ICC does not provide these values, the default is Fi=1 and Di=1, which specify a simple division by 372). The Fi and Di values are specified relative to the SC_CLK frequency. But within SEC2410/SEC4410, this must be translated to a simple divisor of the system clock.

There are two components to this divisor: a Sampling Mode and a Divisor Latch value (DL). The Divisor Latch value is held as a 16-bit value in the DLL/DLM register pair. The Sampling Mode is contained in the most-significant two bits of the SC_CLK register.

The value in the DLL/DLM registers is interpreted according to the separate Sampling Mode, held in the most-significant two bits of the SC_CLK register. The Sampling Mode is a pre-scaler and one of three valid settings:

- 0b00 means a prescaler of 31. This is not required for SEC2410/SEC4410
- 0b10 means a prescaler of 16
- 0b01 means no prescaler. The Divisor directly specifies the etu rate in units of the 60MHz clock, and each bit is sampled directly by that clock. This form gives better accuracy, but cannot be used for all etu rates because of the limit on the size of the Divisor value. Also, even in a non-standard application, it is not allowed to specify fewer than 16 sample times per etu.

For example assume during ATR,TA bits 8~5 = 0b0010 (Fi=558), and bits 4~1 = 0b0011 (Di=4) then Fmax = 6Mhz, and the desired divisor =

This means:

- Fmax = 6Mhz (based on Fi)
- Desired Divisor = 558/4 = 139.5.
- Desired Baud Rate =6.0Mhz/139.5 = 43010.7 bps

This means based on a 60Mhz clock the Divisor Latch value must be: 60Mhz/43011 = 1395. To set the SC_CLK frequency to Fmax, then SC_CLK divisor must be set to 60M/6M = 10.

13.5.3 Recommended etu Rates and Settings

Table 13.2 lists the valid etu rates supported, and the recommended settings of the DL Divisor (in the DLL/DLM registers) and the Sampling field of the CLK register that are used to select them.

If the sampling clock is set to 01, then the baud rate is simply calculated by:

BAUD_RATE = SYSTEM_FREQUENCY / (DL_DIVISOR)

Where: SYSTEM_FREQUENCY = 60,000,000 and DL = The concatenation of SC_DLM with SC_DLL

Table 13.2 Recommended Settings for Valid TA1 ETU Rates

FI (DEC)	DI (DEC)	SAMPLING FIELD (BINARY)	SCLK (ACTUAL) MHZ	DL DIVISOR VALUE (DECIMAL)	BAUD RATE (BITS/SEC)	ERROR (%)	NOTES
0	1	00	4	180	10753		
0	2	00	4	90	21505		
0	3	00	4	45	43011		
0	4	01	4	698	86022	0.07	
0	5	01	4	349	172043	0.07	
0	6	01	4	174	344086	0.22	
0	7	01	4	87	688172	0.22	
0	8	00	4	465	129032		
0	9	01	4	279	215054		
1	1	10	5	279	13441		
1	2	01	5	2232	26882		
1	3	01	5	1116	53763		

Table 13.2 Recommended Settings for Valid TA1 ETU Rates

FI (DEC)	DI (DEC)	SAMPLING FIELD (BINARY)	SCLK (ACTUAL) MHZ	DL DIVISOR VALUE (DECIMAL)	BAUD RATE (BITS/SEC)	ERROR (%)	NOTES
1	4	01	5	558	107527		
1	5	01	5	279	215054		
1	6	01	5	140	430108	0.36	
1	7	01	5	70	860215	0.36	
1	8	01	5	372	161290		
1	9	01	5	223	268817	0.09	
2	1	00	6	180	10753		
2	2	00	6	90	21505		
2	3	01	6	1395	43011		
2	4	01	6	689	86022	0.07	
2	5	01	6	349	172043	0.07	
2	6	01	6	174	344086	0.22	
2	7	01	6	87	688172	0.22	
2	8	01	6	465	129032		
2	9	01	6	279	215054		
3	1	00	7.5	192	10753		
3	2	00	7.5	96	21505		
3	3	00	7.5	48	43011		
3	4	00	7.5	24	86022		
3	5	01	7.5	372	172043		
3	6	01	7.5	186	344086		
3	7	01	7.5	93	688172		
3	8	00	7.5	16	129032		
3	9	01	7.5	298	215054	0.13	
4	1	00	12	180	10753		
4	2	00	12	90	21505		
4	3	00	12	45	43011		
4	4	01	12	698	86022	0.07	
4	5	01	12	349	172043	0.07	
4	6	01	12	174	344086	0.22	
4	7	01	12	87	688172	0.22	

Table 13.2 Recommended Settings for Valid TA1 ETU Rates

FI (DEC)	DI (DEC)	SAMPLING FIELD (BINARY)	SCLK (ACTUAL) MHZ	DL DIVISOR VALUE (DECIMAL)	BAUD RATE (BITS/SEC)	ERROR (%)	NOTES
4	8	00	12	15	129032		
4	9	01	12	279	215054		
5	1	00	15	192	10753		
5	2	00	15	96	21505		
5	3	00	15	48	43011		
5	4	00	15	24	86022		
5	5	00	15	12	172043		
5	6	01	15	186	344086		
5	7	01	15	93	688172		
5	8	00	15	16	129032		
5	9	01	15	298	215054	0.13	
6	1	00	20	180	10753		
6	2	00	20	90	21505		
6	3	00	20	45	43011		
6	4	01	20	698	86022	0.07	
6	5	01	20	349	172043	0.07	
6	6	01	20	174	344086	0.22	
6	7	01	20	87	688172	0.22	
6	8	00	20	15	129032		
6	9	00	20	9	215054		
9	1	10	5	384	9766		
9	2	10	5	192	19531		
9	3	10	5	96	39063		
9	4	01	5	768	78125		
9	5	01	5	384	156250		
9	6	01	5	192	312500		
9	7	01	5	96	625000		
9	8	01	5	512	117118		
9	9	01	5	307	195313	0.07	
10	1	10	7.5	384	9766		
10	2	10	7.5	192	19531		
10	3	10	7.5	96	39063		

Table 13.2 Recommended Settings for Valid TA1 ETU Rates

FI (DEC)	DI (DEC)	SAMPLING FIELD (BINARY)	SCLK (ACTUAL) MHZ	DL DIVISOR VALUE (DECIMAL)	BAUD RATE (BITS/SEC)	ERROR (%)	NOTES
10	4	01	7.5	768	78125		
10	5	01	7.5	384	156250		
10	6	01	7.5	192	312500		
10	7	01	7.5	96	625000		
10	8	10	7.5	32	117118		
10	9	01	7.5	307	195313	0.07	
11	1	10	10	384	9766		
11	2	10	10	192	19531		
11	3	10	10	96	39063		
11	4	10	10	48	78125		
11	5	10	10	24	156250		
11	6	01	10	192	312500		
11	7	01	10	96	625000		
11	8	01	10	512	117118		
11	9	01	10	307	195313	0.07	
12	1	10	15	384	9766		
12	2	10	15	192	19531		
12	3	10	15	96	39063		
12	4	01	15	768	78125		
12	5	01	15	384	156250		
12	6	01	15	192	312500		
12	7	01	15	96	625000		
12	8	10	15	32	117118		
12	9	01	15	307	195313	0.07	
13	1	10	20	384	9766		
13	2	10	20	192	19531		
13	3	10	20	96	39063		
13	4	01	20	768	78125		
13	5	01	20	384	156250		
13	6	01	20	192	312500		
13	7	01	20	96	625000		

Table 13.2 Recommended Settings for Valid TA1 ETU Rates

FI (DEC)	DI (DEC)	SAMPLING FIELD (BINARY)	SCLK (ACTUAL) MHZ	DL DIVISOR VALUE (DECIMAL)	BAUD RATE (BITS/SEC)	ERROR (%)	NOTES
13	8	01	20	32	117118		
13	9	01	20	307	195313	0.07	

13.6 16-bit General Purpose Counter

A 16-bit general-purpose down counter is located in the DCL, DCH register pair. Writing to these registers stores the preload value for the counter. Reading these registers will yield the current count value. Once the counter is enabled and begins counting, it will continue counting down either until it reaches 0000h or until a new preload value is written to the counter. At 0000h the counter wraps around to FFFFh and will generate the General Purpose Down Counter interrupt.

The counter is clocked by a 10kHz clock input (i.e., 100usec/lsb) derived from the system clock.

The counter loads the stored preload value and begins counting when the Counter Enable bit is set to ‘1’. On a POR or when the Counter Interrupt Enable bit is cleared to ‘0’, the preload value used by the counter is initialized to FFFFh. Setting the Counter Enable bit to ‘1’ loads the current preload value. This allows software to write the preload value before enabling the counter. Therefore, when this enable bit is set to ‘1’ the counter begins counting down from the preload value, which will be either the default preload value (FFFFh) or a programmed preload value. The Counter Enable bit is located in the LCR register.

To write the Pre-load value:

If the counter is disabled, the DCL and DCH registers may be written in any order. If the counter is enabled, write the LSB first into the DCL register. Writing the MSB into the DCH register loads the pre-load value into the counter and resets the divider used to scale the clock. The counter, if enabled, begins counting down as soon as the preload value is loaded into the register and the clock is re-initialized.

To read the Count value:

Read the LSB first from the DCL register. Reading the DCL register latches the MSB of the count value into the DCH register.

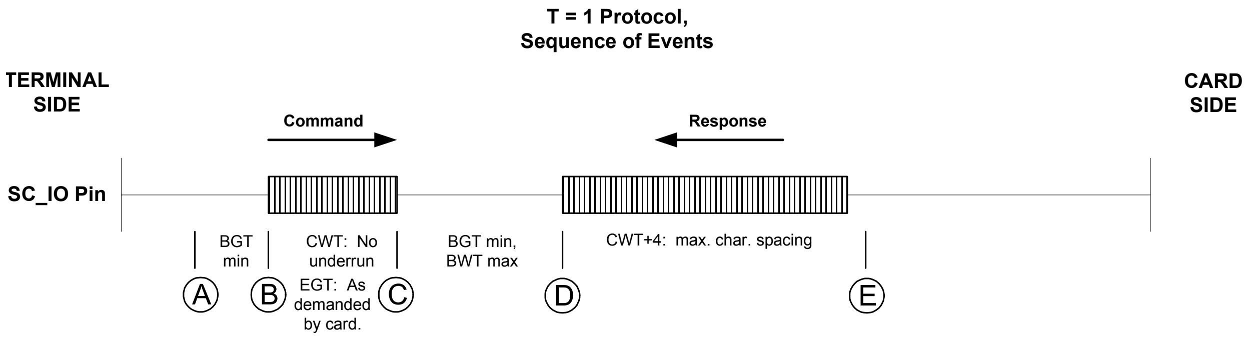
13.7 T=1 Operation

In T=1 mode, a transmission is immediately followed by received data. Therefore, when the Receiver is newly enabled (see the FCR register), this is interpreted as meaning that the receiver will begin accepting data only when transmission is finished. According to the various standards, the card is supposed to have a minimum turnaround delay before it starts transmitting data, but in practice the controller does not rely on that, and will accept data as soon as the last character has been transmitted.

13.7.1 Operation of Timers in T=1 Mode

Transactions between the Controller and a Smart Card are performed in an exchange of data: the Controller transmits a command, and the Smart Card must respond. Because the Smart Card is allowed to respond very quickly after receiving the last byte of the command, the timers must be set up before the command is sent, and software cannot interact with the exchange until the response has been received, or a timeout has occurred. Both of these events trigger an interrupt.

Figure 13.3 T=1 Events



Character min. Guard Times are guaranteed on transmit and monitored on receipt.

In Figure 3.1, "T=1 Exchange", the sequence of events is shown in the exchange of data with the Smart Card. The operation of the controller at points A, B, C, D and E is described below.

Setup before First T=1 Transmission

Software directly pre-loads the Guard Timer BGT reload register with a value based on the BGT parameter from the ATR message. The Guard Timer resolution is one etu.

Software loads the Guard Timer EGT reload register with a value based on the current EGT.

Software enables the Guard Timer, which is used to inhibit transmission until it underflows.

The initial state of the Guard Timer is waiting for a transmitted character for EGT timing, so the first time it is enabled the first BGT value must be guaranteed by software using different means, before progressing to Point A.

13.7.1.1 Point A: Software initiates exchange.

Software writes the entire message to be transmitted into the FIFO.

Software writes the value 0x02 to the FIFO Threshold register, to get an interrupt when three bytes have been received in response.

Software loads the Timeout timer with the current BWT value, in units of 1.25 milliseconds.

Software loads the CWT Timer with a value based on the current CWT value, and enables the CWT Timer.

Software enables both the Transmitter and the Receiver. Transmission begins after any delay imposed by the Guard Time, proceeding to Point B.

Software waits for interrupts occurring at Point E.

13.7.1.2 Point B: Transmission begins.

First character is fetched from FIFO.

Transmission of first character begins.

At each transmitted character, the Guard Timer reloads from its EGT Reload Register (EGT value).

At the end of each character, after the 1 etu of mandatory Guard Time, the Guard Timer counts down, and it inhibits transmission until it underflows. On underflow, the Guard Timer permits transmission and stops.

Characters will be fetched from the FIFO and held until the EGT value from the Guard Timer expires.

When the FIFO becomes empty of characters to be transmitted, the SmartCard will immediately disable the transmitter (clearing the FTE bit in the FCR register), and will transition to the Receive phase of the exchange.

13.7.1.3 Point C: Preparation for Reception

When the entire Transmit message has been sent, the Timeout timer begins monitoring for the first received character. When it is received, the Timeout timer stops and does nothing else until software re-enables it. If instead the Timeout Timer underflows (at the BWT time), it stops, disables the Receiver (by clearing the FRE bit in the FCR register) and presents the TMO interrupt.

In a second mode of operation (WTX), the Timeout Timer will continue running and posting interrupts, for counting down (in software) the number of underflows of this timer before detecting an error. In this mode, the underflow simnply reloads and continues, posting the interrupt, but it does not automatically disable the receiver. When the appropriate number of underflows has occurred, the software will place the timer back into BWT mode, and it will then interrupt, stop, and disable the receiver if it underflows again.

13.7.1.4 Point D: Message being received

At the first received Start bit, the CWT timer begins operation. This timer counts in units of etu. It has been loaded by software, before transmission, with the maximum distance between received characters. The value also includes the tolerance value (4 or 5 etu) which is required by the EMV standard. This timer is reloaded, and retriggered, on receipt of each character. If it elapses, it stops, clears the FRE bit to disable the Receiver to the FIFO, and posts the CWT interrupt request.

After the first three bytes have been received, the FIFO Threshold interrupt is posted. Software reads three bytes from the FIFO, and interprets them to determine the remaining length of the response from the card. Software re-sets the FIFO Threshold to the expected number of bytes, minus 1.

13.7.1.5 Point E: End of Message

The end of a message will be detected either by software, seeing the FIFO Threshold interrupt, or by the CWT Timer interrupt if not enough characters come in. (The CWT Timer event will also set the Threshold interrupt automatically.) If too many characters are received, software will detect this from extra bytes in the FIFO. If enough characters are received that the FIFO overflows, the OE interrupt is set. Both the OE and CWT Timer event disable the Receiver from placing any more characters into the FIFO, by clearing the FRE bit in the FIFO Control Register.

13.8 T=0 Operation

The T=0 protocol is highly interactive, and there is no timeout constraint placed on the Controller side. For this mode, to support newer high bit rates, there are new timer interactions defined for this mode, and a new pair of state machines is needed in order to filter incoming data.

In T=0 mode, unless ATR mode is also specified, a transmission is immediately followed by received data. Therefore, when in T=0 mode and not ATR mode, and the Receiver is newly enabled (see the FCR register), this is interpreted as meaning that the receiver will begin accepting data only when transmission is finished. According to the various standards, the card is supposed to have a minimum turnaround delay before it starts transmitting data, but in practice the controller does not rely on that, and will accept data as soon as the last character has been transmitted.

T=0 protocol commands specify the length of the expected response from the card. Therefore, software can be interrupted once by the FIFO Threshold interrupt, when the entire expected message has been received, or when it has been ended prematurely by the card (Timeout Timer [WWT] error, EOM interrupt for early SW1/SW2 presentation, or Parity Error).

13.8.1 T=0 Timer Operation

In T=0 mode, the Guard Timer will be used to guarantee the DGT requirement (turnaround guard time) when beginning transmission, and to insert the extra guard time (EGT) delay between characters. DGT and EGT are not monitored when receiving from the card.

As when beginning T=1 mode, the Guard Timer is not effective until at least one character has been transmitted or received. Therefore, when software enables the Guard Timer for the first time, it must guarantee by other means that the DGT guard time has elapsed before enabling the Transmitter.

In T=0 mode, the Timeout Timer will be used to monitor the card's performance relative to WWT, which defines both the maximum allowed turn-around time in a card's response, and the maximum allowed spacing between characters while the card is transmitting. In this mode, the Timeout timer will start on the last transmitted character, will reload and continue on each received character, but will post an interrupt, disable the receiver and stop if it underflows.

The minimum character guard time (2 etu) on transmission will be guaranteed by the fact that T=0 mode is selected in the Protocol Mode register. On transmission, the guard period will be monitored only for a Parity Error response from the Smart Card, and not for any other form of interference.

13.9 T=0 Byte Filtering

There is a new consideration regarding FIFO space. The Smart Card may insert NULL characters at various points in the communication, whose purpose is to reset the Timeout Timer (being used for WWT). Also, there is an unpredictable number of INS bytes, which signal when a card is prepared to transfer only one byte instead of the whole remaining block. A pair of state machines are provided to filter out these extra bytes in a T=0 exchange, thus ensuring that no valid exchange will ever overflow the FIFO.

Both state machines filter only bytes that are being received from the card, but they are called "Incoming" and "Outgoing" based on the nature of the command being executed. The direction is defined relative to the card, so that "Outgoing" means reading data out of the card, and "Incoming" means writing data into the card.

The special "Procedure Bytes" are those bytes sent by the card that are not data. These are:

- **NULL**, encoded as 0x60, which is used as padding to reset the WWT timing monitor
- **SW1**, encoded as 0x61-0x6F and 0x90-0x9F. This is the first byte of status, which flags the end of a transfer. It is always followed by one byte, **SW2**, which completes the status indication and is the last byte of the transaction.
- **INS** and $\overline{\textbf{INS}}$ are used as flags, and represent a true (**INS**) and complemented ($\overline{\textbf{INS}}$) echo of the Instruction byte (sent by the terminal) that is being executed by the card. The encodings of **INS** and $\overline{\textbf{INS}}$ are such that they can never be confused with **NULL** or **SW1**.

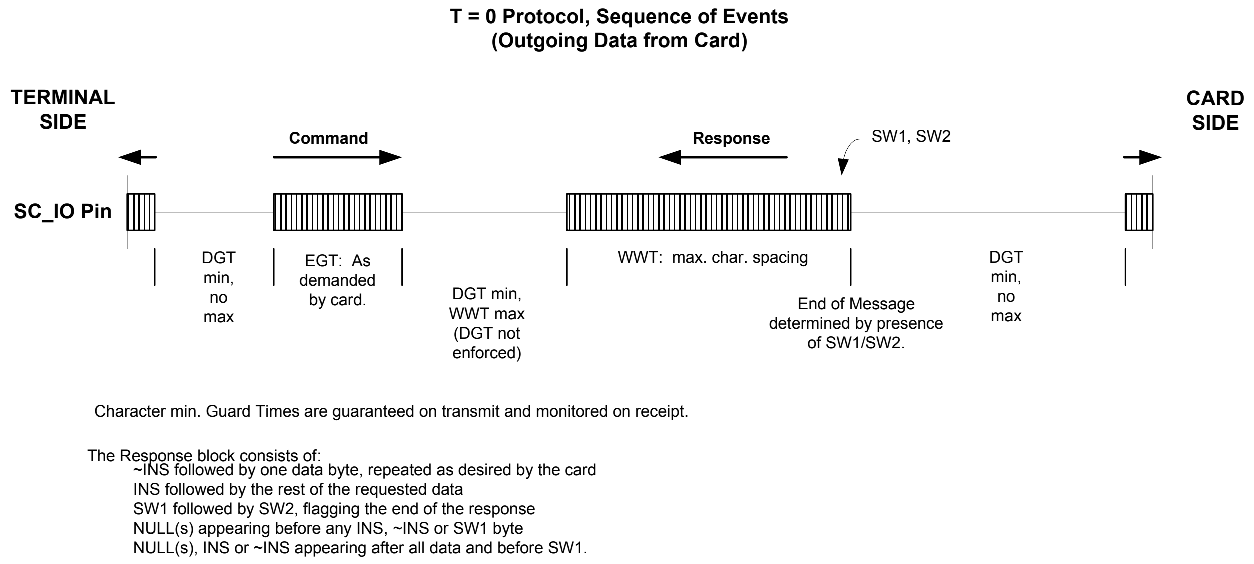
13.9.0.1 T=0 Outgoing Byte Filter

The first ("Outgoing") state machine is used when a command is being issued that reads data from the card. In this scenario, the card responds on receipt of the command, and it does not stop transmitting until the entire requested block of data has been transferred. The format of this response is variable depending on the card's performance. The Outgoing state machine, then, filters out the variable portions of this response, leaving only the outgoing data and status, which will be of a predictable maximum size of 258 bytes (256 bytes of information data plus the status bytes SW1 and SW2).

To operate this filter, software specifies in the register set the number of data bytes it intends to read from the Smart Card, and the INS byte value that it intends to send. It then enables the state machine with the dedicated Enable bit (OSME, in the Protocol Mode Register), and transmits its command. When the transmission is completed (as determined by the Message Length register used for transmission), the state machine becomes active. As the card responds, any NULL characters at appropriate places are detected and discarded, and all INS and INS procedure bytes are discarded, leaving only the data bytes and the two status bytes (SW1 and SW2) to be placed into the FIFO.

A typical sequence of events for a T=0 Outgoing exchange is shown in Figure 3.2.

Figure 13.4 Outgoing T=0 Command Sequence


T = 0 Protocol, Sequence of Events
(Outgoing Data from Card)
TERMINAL SIDE
CARD SIDE
Command
Response
SW1, SW2
SC_IO Pin
DGT min, no max
EGT: As demanded by card.
DGT min, WWT max (DGT not enforced)
WWT: max. char. spacing
End of Message determined by presence of SW1/SW2.
DGT min, no max
Character min. Guard Times are guaranteed on transmit and monitored on receipt.
The Response block consists of:
~INS followed by one data byte, repeated as desired by the card
INS followed by the rest of the requested data
SW1 followed by SW2, flagging the end of the response
NULL(s) appearing before any INS, ~INS or SW1 byte
NULL(s), INS or ~INS appearing after all data and before SW1.

A state diagram for the Outgoing Byte Filter is shown in Figure 3.3. It accepts from software:

A 9-bit count of the number of data bytes expected from the card, initialized by software to be in the range of 1 to 256 (00h written by software to the 8-bit FLL register sets the count to 256, not zero). This number of data bytes are collected and placed into the FIFO, followed by the SW1 and SW2 bytes, for a total of 258 bytes maximum.

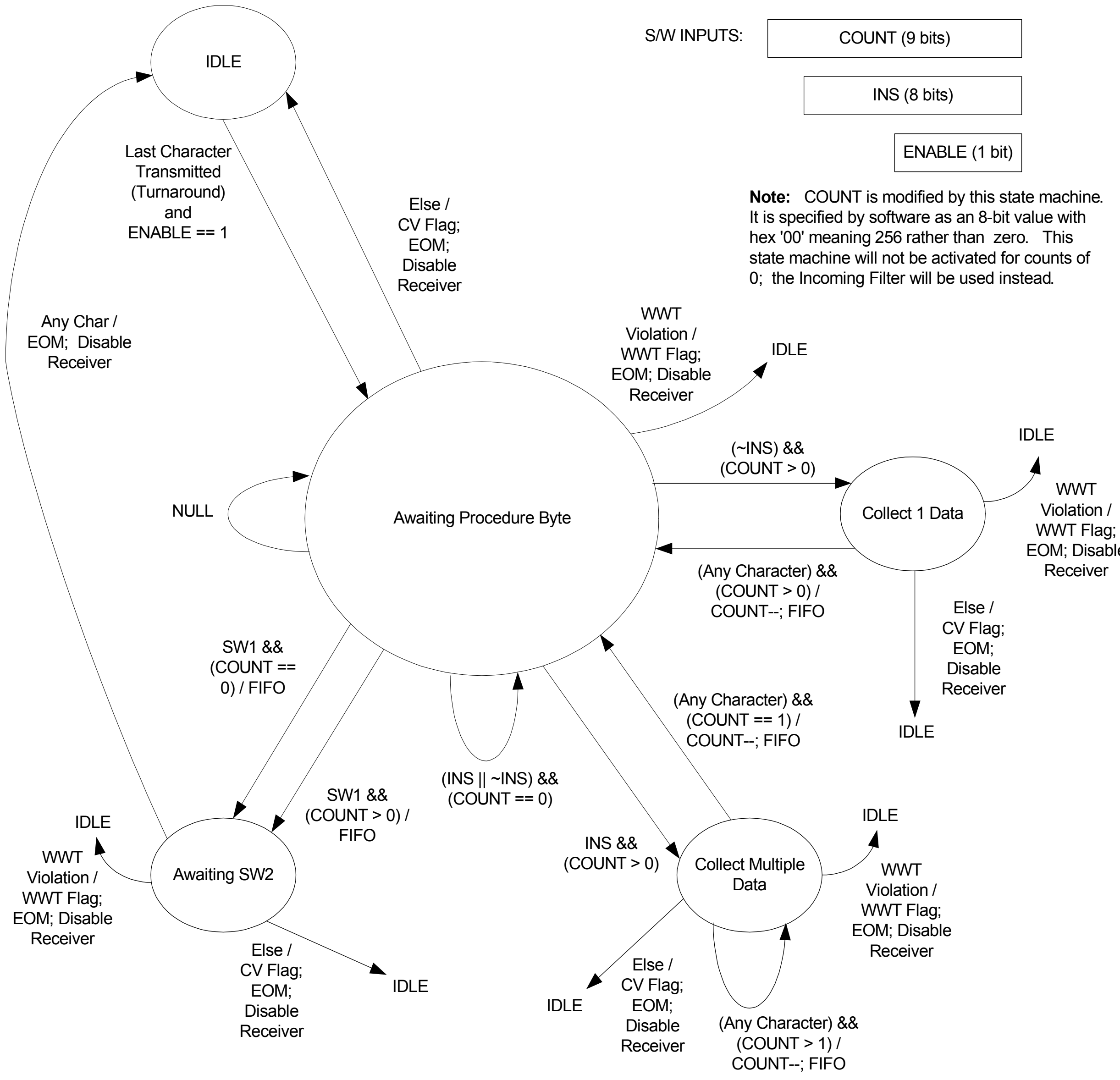
The INS byte being sent to the card. This defines the encodings of the INS and INS procedure bytes.

An Enable bit (OSME, in the Protocol Mode Register) for this specific state machine. When the Enable bit is turned on, the state machine will wait for the transmitter to finish transmitting the command to the card, then it will start filtering the response.

When the state machine detects the end of a message, or a fatal error in communication, it activates the EOM interrupt (End of Message), and disables the receiver. If it is terminating communication because of an error in encoding, it will also set the CV (Code Violation) error status bit. If the Timeout Timer (measuring WWT) underflows during a received message, it will also disable the receiver and stop the state machine. The EOM interrupt will be posted in this case, and also the TMO interrupt from the Timeout Timer itself.

As characters are received, the least-significant 8 bits of count may be examined by reading the FLL register. The value 00h, which might mean 0 or 256, can be interpreted by looking at the FIFO count to determine whether any characters have been received.

Figure 13.5 T=0 Outgoing Byte Filter State Diagram


S/W INPUTS:
COUNT (9 bits)
INS (8 bits)
ENABLE (1 bit)
Note: COUNT is modified by this state machine. It is specified by software as an 8-bit value with hex '00' meaning 256 rather than zero. This state machine will not be activated for counts of 0; the Incoming Filter will be used instead.
IDLE
Last Character Transmitted (Turnaround) and ENABLE == 1
Else / CV Flag; EOM; Disable Receiver
Any Char / EOM; Disable Receiver
WWT Violation / WWT Flag; EOM; Disable Receiver
IDLE
Awaiting Procedure Byte
NULL
(~INS) && (COUNT > 0)
Collect 1 Data
IDLE
WWT Violation / WWT Flag; EOM; Disable Receiver
(Any Character) && (COUNT > 0) / COUNT--; FIFO
Else / CV Flag; EOM; Disable Receiver
IDLE
SW1 && (COUNT == 0) / FIFO
(Any Character) && (COUNT == 1) / COUNT--; FIFO
(INS || ~INS) && (COUNT == 0)
SW1 && (COUNT > 0) / FIFO
IDLE
WWT Violation / WWT Flag; EOM; Disable Receiver
Awaiting SW2
Else / CV Flag; EOM; Disable Receiver
IDLE
INS && (COUNT > 0)
Collect Multiple Data
IDLE
WWT Violation / WWT Flag; EOM; Disable Receiver
IDLE
Else / CV Flag; EOM; Disable Receiver
(Any Character) && (COUNT > 1) / COUNT--; FIFO

13.9.1 T=0 Incoming Byte Filter

This state machine is active when a command is being executed that writes data into the card. In spite of this, the bytes being filtered are only the responses that are coming from the card. When the Controller is intending to transmit data, the state machine is simpler, because there are fewer ways that the Smart Card can respond. The command is executed in multiple exchanges between the Controller and the card, and as far as the Controller hardware is concerned, each of these (starting with transmission of a 5-byte command header from the Controller) is an independent exchange. See Figure 3.4 for an example of an T=0 Incoming command sequence.

A state diagram for the Incoming Byte Filter is shown in Figure 3.5.

When expecting an **INS** or $\overline{\textbf{INS}}$ response, this filter will remove only initial **NULL** bytes from the Smart Card's responses, leaving the **INS** or $\overline{\textbf{INS}}$ response byte in the FIFO for software to interpret. When expecting an **SW1** byte (when the count of data to be transferred is zero), any initial **NULL**, **INS** or **INS** byte is discarded. Software must provide a valid Count value, along with INS and the Enable bit

(ISME, in the Protocol Mode Register), for each Transmit/Receive exchange of information in the command sequence.

The Incoming byte filter does not interpret the Count in the same way as the Outgoing byte filter. For the Incoming byte filter, a value of 00h provided by software in the FLL register actually means zero, and the maximum valid count value is 254 for T=0 Incoming traffic. The FLL register is not changed except by software, so there is no ambiguity in values as there is when software reads the FLL register under the Outgoing filter.

Figure 13.6 IINcoming T=0 Command Sequence Example

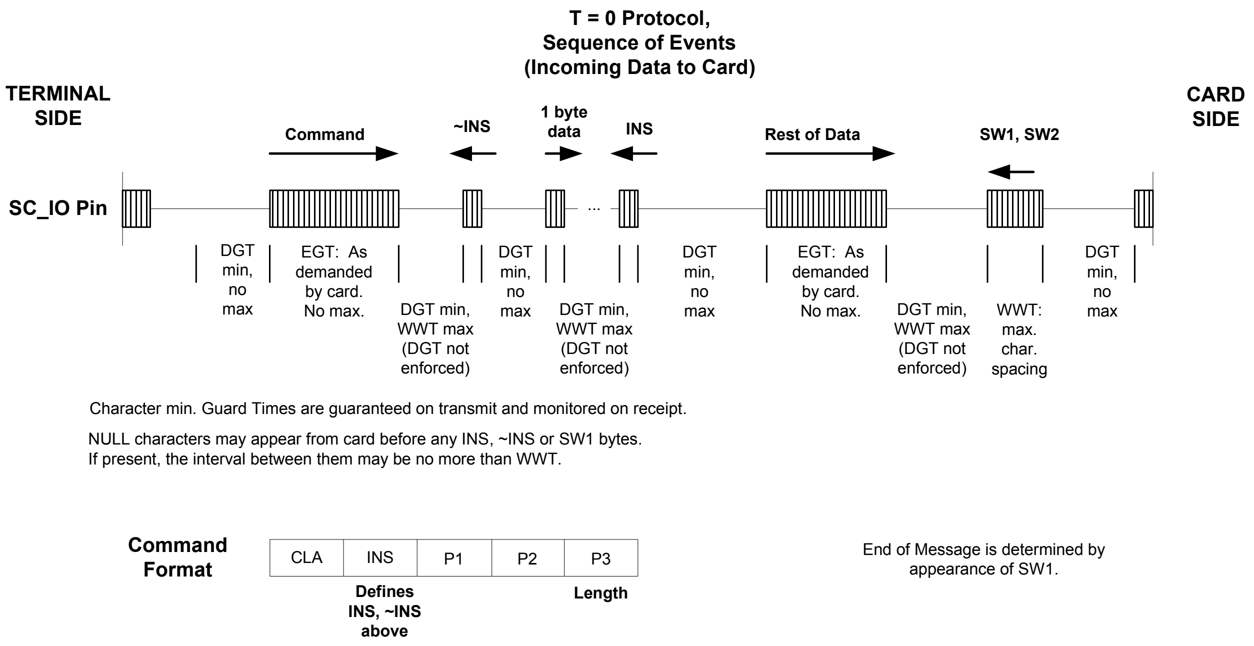
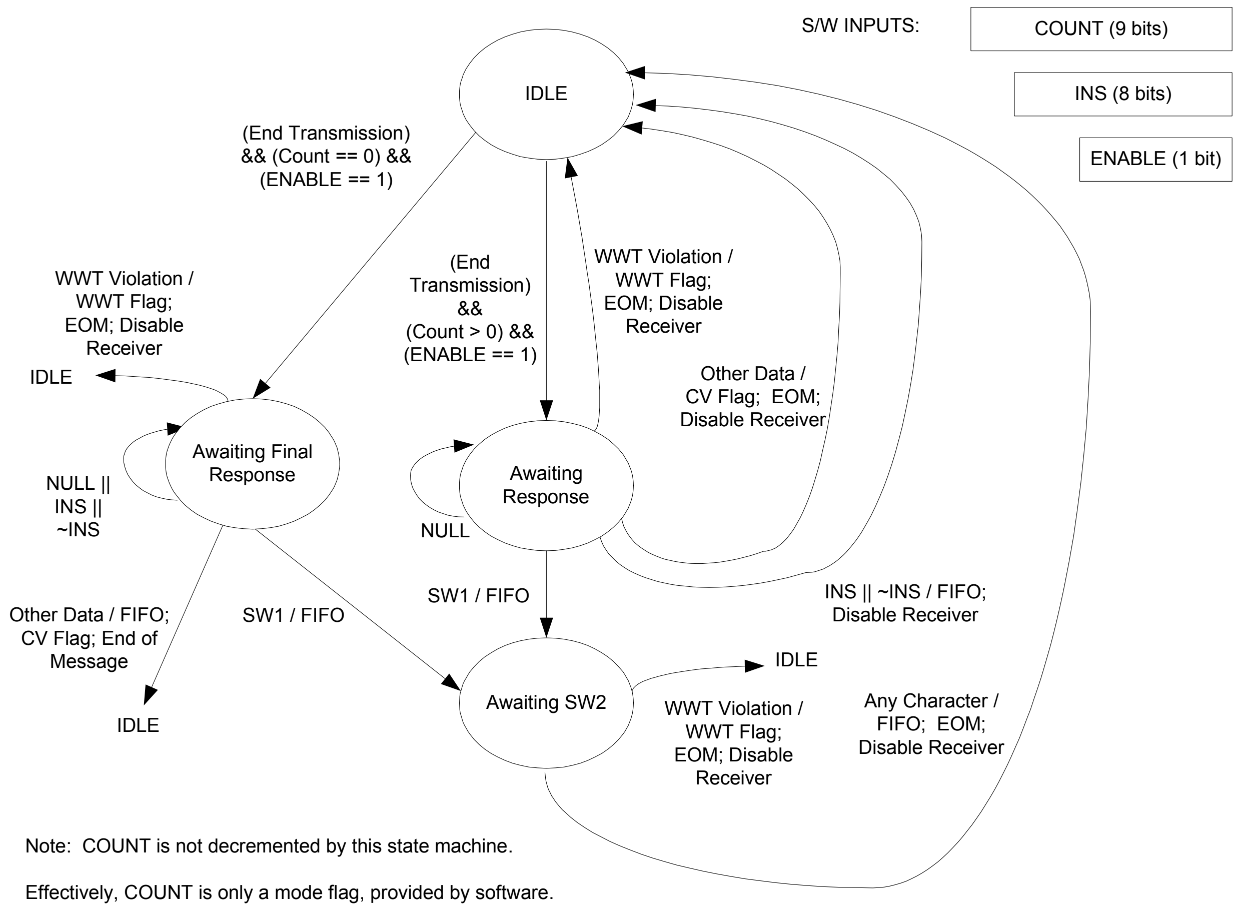

T = 0 Protocol,
Sequence of Events
(Incoming Data to Card)
TERMINAL SIDE
CARD SIDE
Command
~INS
1 byte data
INS
Rest of Data
SW1, SW2
SC_IO Pin
...
DGT min, no max
EGT: As demanded by card. No max.
DGT min, WWT max (DGT not enforced)
DGT min, no max
DGT min, WWT max (DGT not enforced)
DGT min, no max
EGT: As demanded by card. No max.
DGT min, WWT max (DGT not enforced)
WWT: max. char. spacing
DGT min, no max
Character min. Guard Times are guaranteed on transmit and monitored on receipt.
NULL characters may appear from card before any INS, ~INS or SW1 bytes.
If present, the interval between them may be no more than WWT.
Command Format
CLA
INS
P1
P2
P3
Defines INS, ~INS above
Length
End of Message is determined by appearance of SW1.

Figure 13.7 T=0 Incoming Byte Filter State Diagram


S/W INPUTS:
COUNT (9 bits)
INS (8 bits)
ENABLE (1 bit)
IDLE
(End Transmission) && (Count == 0) && (ENABLE == 1)
(End Transmission) && (Count > 0) && (ENABLE == 1)
WWT Violation / WWT Flag; EOM; Disable Receiver
Other Data / CV Flag; EOM; Disable Receiver
Awaiting Final Response
Awaiting Response
NULL || INS || ~INS
NULL
Other Data / FIFO; CV Flag; End of Message
SW1 / FIFO
INS || ~INS / FIFO; Disable Receiver
Awaiting SW2
Any Character / FIFO; EOM; Disable Receiver

Note: COUNT is not decremented by this state machine.

Effectively, COUNT is only a mode flag, provided by software. Software provides non-zero here unless SW1 is expected. If it is '0', INS and ~INS are also discarded, as well as NULL.

If SW1 occurs when it is not expected (COUNT>0), then it and SW2 are both received. Software must parse the SW1 byte to determine that it expects an SW2 byte from the FIFO.

13.9.2 ATR Reception

The ATR (“Answer to Reset”) sequence is a series of bytes sent by the Smart Card in response to the Reset signal from the Controller. Certain timers and specialized circuitry are used in receiving the ATR information.

Figure 13.8 ATR Sequence, Cold Reset

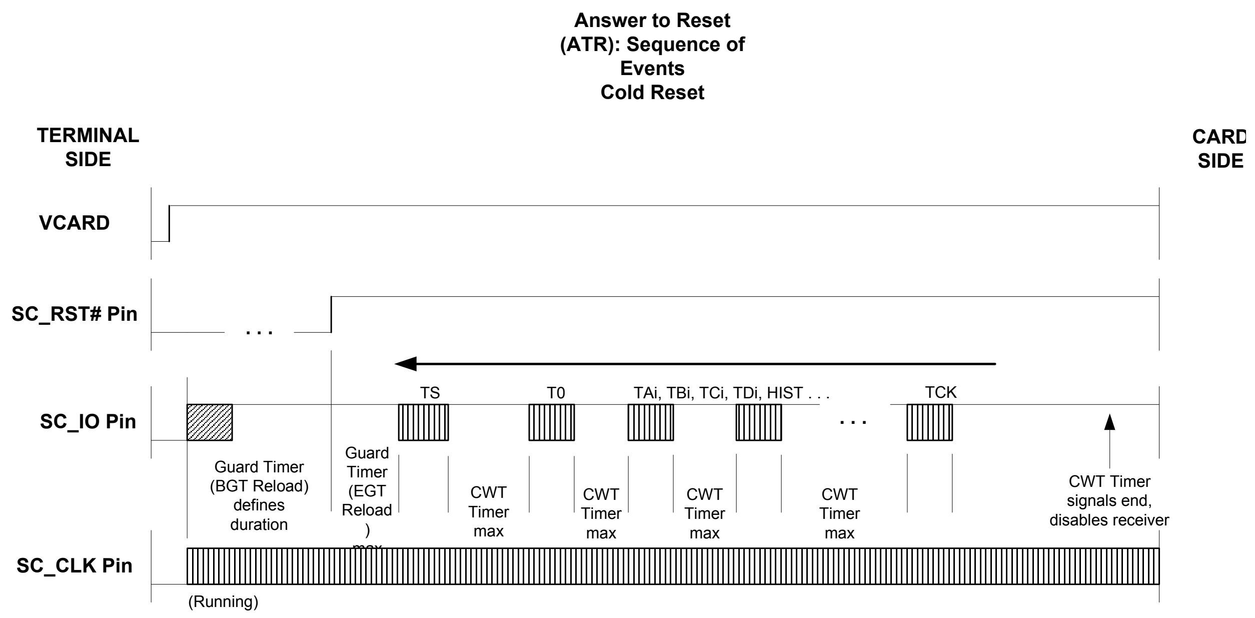
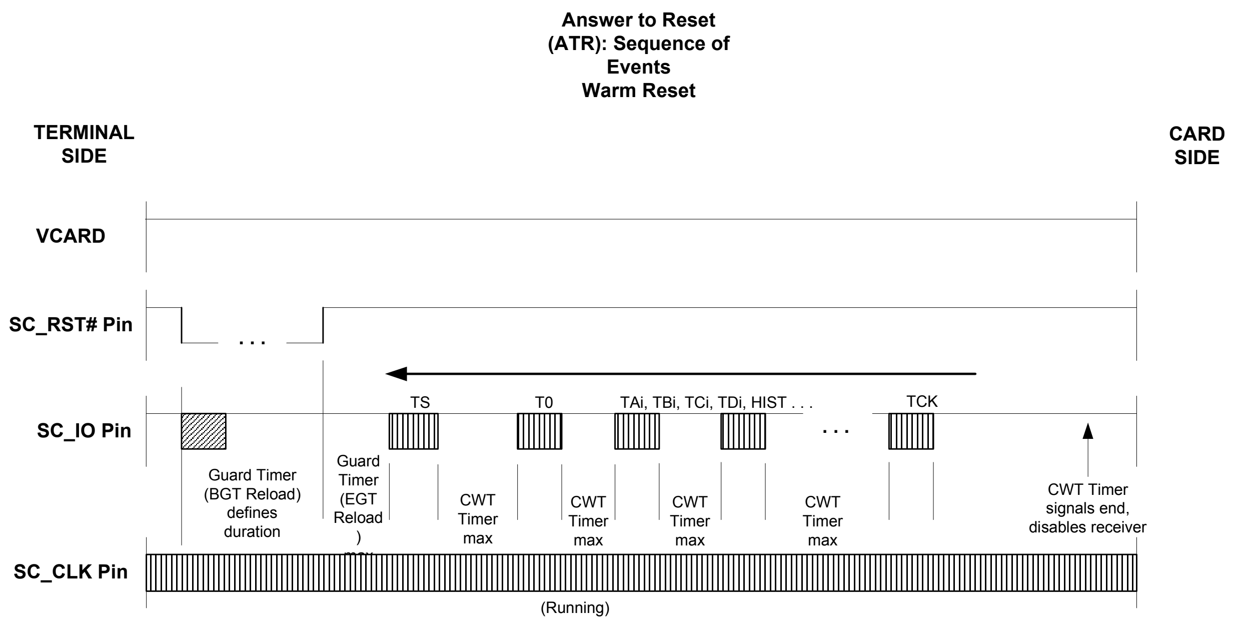

Answer to Reset (ATR): Sequence of Events Cold Reset
TERMINAL SIDE
CARD SIDE
VCARD
SC_RST# Pin
SC_IO Pin
SC_CLK Pin
TS
T0
TAi, TBi, TCi, TDi, HIST . . .
TCK
Guard Timer (BGT Reload) defines duration
Guard Timer (EGT Reload) max
CWT Timer max
CWT Timer signals end, disables receiver
(Running)

Figure 13.9 ATR Sequence, Warm Reset


Answer to Reset (ATR): Sequence of Events Warm Reset
TERMINAL SIDE
CARD SIDE
VCARD
SC_RST# Pin
SC_IO Pin
SC_CLK Pin
TS
T0
TAi, TBi, TCi, TDi, HIST . . .
TCK
Guard Timer (BGT Reload) defines duration
Guard Timer (EGT Reload) max
CWT Timer max
CWT Timer signals end, disables receiver
(Running)

To anticipate the ATR sequence, the Controller is placed by software into a special mode called “ATR”. In this mode, two of the timers are in a special mode to validate the timing of the sequence. Figure 3.6 shows the sequence of events in a Cold Reset, where power has been removed from the card. Figure 3.7 shows the sequence of events in a Warm Reset, where power is maintained, but a new SC_RST_N pulse is applied to reset the card.

In preparing for the ATR sequence, the software must establish the default etu time: the equivalent of TA1=0x11, or 372 periods of the selected SC_CLK frequency.

At the beginning of the sequence, the two reload registers of the Guard Timer determine the duration of the Reset pulse and measure the response time from the the Smart Card to enforce a valid delay. After the first character, the CWT Timer starts, and counts the maximum amount of time the card is allowed to spend between characters. When the CWT timer expires, an interrupt (CWT) is sent to the software, which can then read the message from the FIFO. This event will also set the FIFO Threshold interrupt active. Software will be able to parse the message and determine whether it is complete.

Software may, rather than using the CWT timer for this purpose, set thresholds for the FIFO such that it is periodically interrupted either by the individual characters or by larger expected fields. The CWT timer will still be useful as an error indication.

The first byte (TS) is interpreted by hardware. One of two values is allowed, which from that point onward determines the “convention” used by the card. The possible conventions used are listed below. “L” means a bit time with the SC_IO pin held low, and “H” means a bit time with the SC_IO pin held high.

Direct Convention, which is signalled by the TS bit sequence LHHLHHHLLHHH. In this convention, bits of a character are sent least-significant bit first, ‘0’ bits in the data field are represented by the Low state, and a true Even parity is used. This byte will appear as 0x3B in the FIFO.

Inverse Convention, which is signalled by the TS bit sequence LHHLLLLLLHHH. In this convention, bits of a character are sent most-significant bit first, ‘0’ bits in the data field are represented by the High state, and an inverted Even parity bit is used (appearing as a parity error to any circuit reading it according to the Direct convention). This byte will appear as 0x3F in the FIFO.

The Direct or Inverse convention will be selected automatically by hardware after receiving the TS byte after a rising edge on the SC_RST_N signal. This setting will be reported in the TSM bit of the Protocol Status register, and will be used to interpret all characters until the next SC_RST_N pulse. If any TS value other than the two above is seen, the receiver will be disabled, and the CV bit (Code Violation) will be set in the PRIP register to indicate the error. If a FIFO threshold larger than one byte was selected, the eventual CWT timer interrupt will both set the FIFO Threshold interrupt and alert the software to look at the error flag.

While power is not applied to the card, the terminal is required to hold the SC_RST_N, SC_CLK and SC_IO pins low (not floating). When power is first applied to the card (a Cold Reset, shown in Figure 3.6), the SC_RST_N pin must be held low until SC_CLK begins running. SC_IO must rise to its idle state (high) after power has been applied, and no later than 200 cycles of SC_CLK. The SC_RST_N pin must then be set high between 108 and 120 default etu times after the clock starts.

When the card has already been initialized from a Cold Reset, it may be reset without removing power. This is a “Warm Reset”, shown in Figure 3.7. In this case, the clock keeps running, SC_IO should remain high, and the time range of 108 to 120 default etu times applies to the width of the SC_RST_N pulse.

13.9.3 Guard Time Algorithm

A special case occurs under some circumstances, in which software thinks that an exchange is finished, but the card does not, and keeps transmitting characters. One such case is when a parity error occurs in a T=1 message. The FIFO stops receiving characters after the faulty one (for diagnostic purposes, to indicate the character with the error), and signals to software an End of Message with an error.

In this circumstance, it is necessary that any transmission commanded by the software (e.g., the packet complaining about the parity error) must wait until the card is finished transmitting. However, if the card is insane and does not stop transmitting, then software must be informed of this error so that the card can be deactivated. The Guard Time Algorithm hardware serves both of these purposes.

A specific error flag is provided (TF) , and a timing register (GSR), to support this feature. The feature is not optional, and so it cannot be disabled.

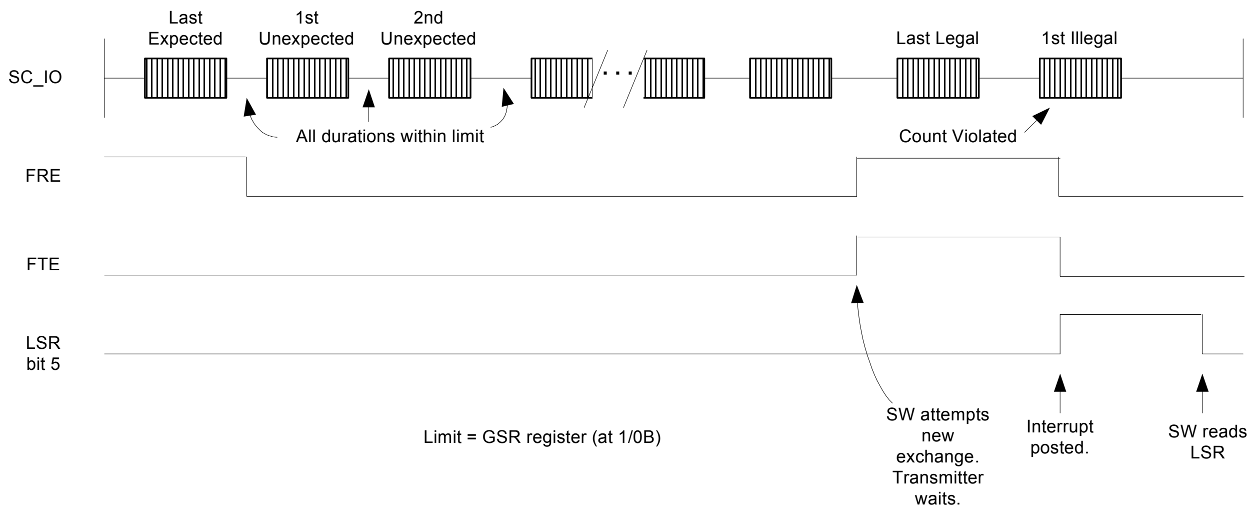
The GSR register (Guard Spacing Register) is programmed by software with the expected maximum spacing between received characters in units of etus, including extra guard time EGT. (This information might be considered redundant, but is required in a separate register by the implementation.) The value in the GSR register is interpreted as a maximum amount of time allowed from start bit to start bit, and so it must be at least

As each new character is received within this window, an internal counter ("CPT") is decremented once. This counter restarts, starting from the maximum legal number of characters in a packet (258 for T=0, 259 for T=1) as soon as characters start being received in an exchange, regardless of whether the receiver remains enabled or not, and regardless of errors. The CPT counter reloads and stops when no character is received within the GSR window.

If software attempts to transmit while this counter is still active, the transmission is inhibited and held pending. If, however, while a transmisson is pending, the CPT count underflows, then the transmission is abandoned, and the TF error (Transmit Failure) is posted, which is an interrupt. See Figure 3.8 for this case. Note that, in T=0 mode, the Incoming or Outgoing filter remains applied as selected, so that any Procedure Bytes (NUL, INS, ~INS) are not counted.

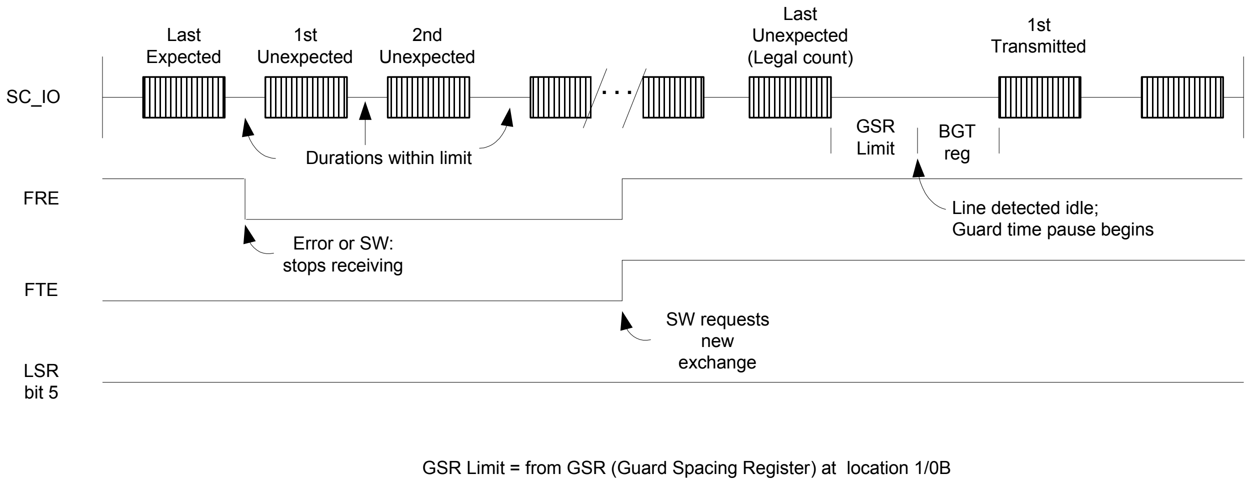
If there is no such error, then, after the vacant window time has passed, the transmitter waits for the designated Guard Time amount (DGT or BGT) and begins transmitting. See Figure 3.9 for this case.

Figure 13.10 Guard Time Algorithm with Error, Transmit Abandoned



Error: Transmit attempted and Card has been transmitting too long.

Figure 13.11 Guard Time Algorithm, No Error, Transmit Held


Last Expected
1st Unexpected
2nd Unexpected
Last Unexpected (Legal count)
1st Transmitted
SC_IO
Durations within limit
GSR Limit
BGT reg
FRE
Line detected idle; Guard time pause begins
Error or SW: stops receiving
FTE
SW requests new exchange
LSR bit 5
GSR Limit = from GSR (Guard Spacing Register) at location 1/0B

Most Normal Case: Early Cut-off (e.g. T=1 Parity Error).
Transmitted response is delayed until Card is Idle.

13.9.4 Card power for SmartCard Interface.

The pins on this interface are powered by VAR_CRD_PWR. If the Smartcard interface is not used, the VAR_CRD_PWR can be used to implement variable voltage GPIOs. The control for the regulator is in the the GPIO block.

The power to the SmartCard should not be turned on till a card is detected. When there is no card present, enable the synchronous SmartCard interface, turn all the bits to inputs, and enable the pull-down resistors. This will ensure that the output signals are held at ground. Once a card is detected, Enable the power first, wait at least 100mS then enable the asynchronous or sysnchronous interface as necessary.

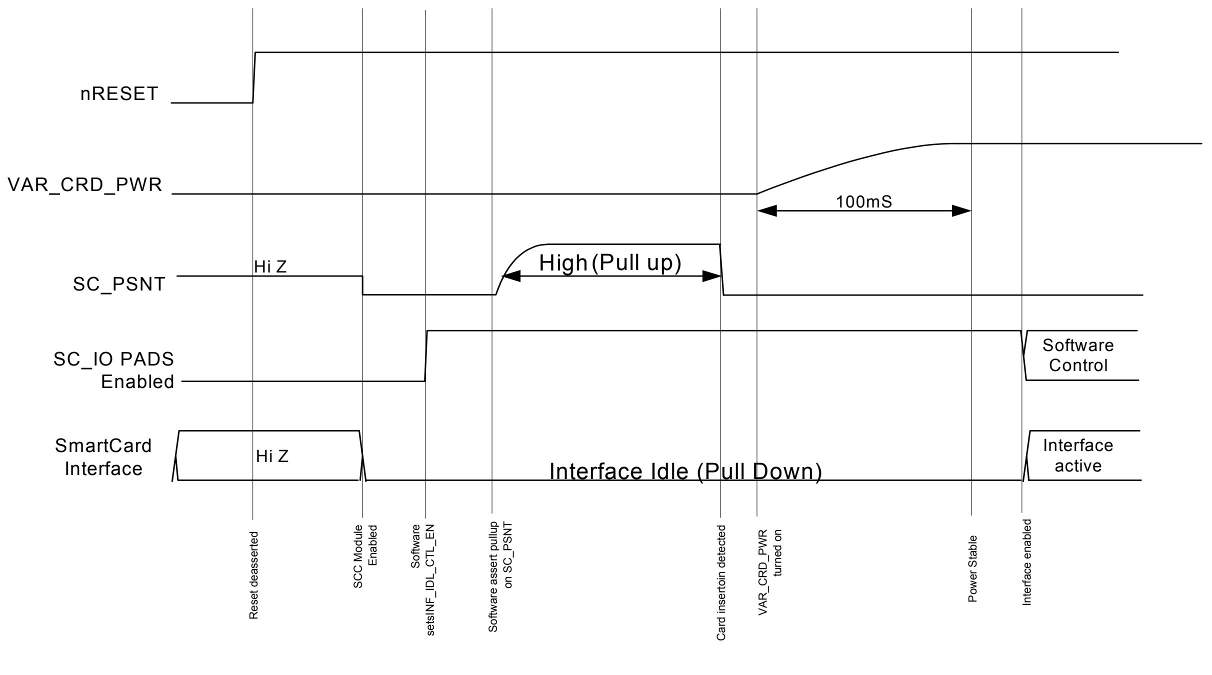


Figure 13.12 SmartCard Powerup

13.9.5 LED control for SmartCard Interface.

The Smartcard LED can be driven in one of three ways. It can be driven directly by the Smartcard IP in asynchronous mode. This mode is selected by setting the SC_LED_SEL bit in the GPIO block. When running in synchronous mode or GPIO mode, firmware must control the LED directly. The LED can either be set to blink, automatically, or run under full manual control. Blinking is controlled by the LED1_GPIO1_CTL. Full manual is done by controlling the register directly.

13.9.6 Enabling the Synchronous SmartCard Interface.

The Synschronous interface is enabled through the control register in the wrapper block.

13.10 Top level of the SmartCard Interface.

Note that the Smartcard interface can also be used as GPIOs. The synchronous block can be used as bit addressable GPIOs, or it can be configured to output the signals from the GPIO block itself.

The muxing of the signals of the three different interfaces is shown in the figure below. The selection of whether the GPIOs or the SmartCard logic controls the pins is controlled by SC_GPIO_EN in PIN_MUX_SEL register

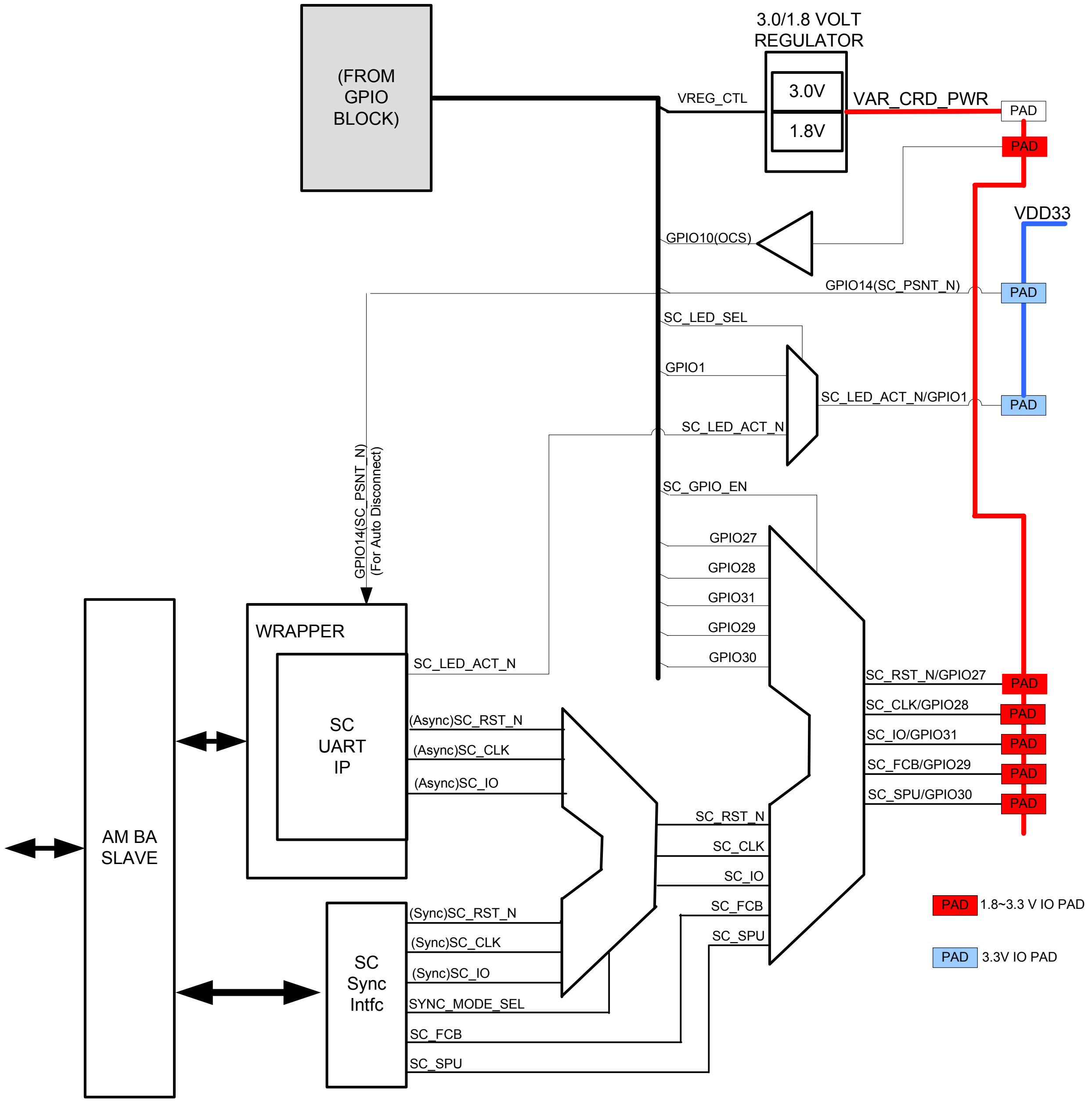


Figure 13.13 SmartCard Muxing

13.11 Register Map

Table 13.3 SmartCard Controller Registers

OFFSET ADDRESS	NAME	R/W	DESCRIPTION	PAGE
0x0000	SC_TBR/RBR	R/W	8 bit FIFO Data	
0x0001	SC_IEN	R/W	Interrupt enable	
0x0002	SC_IID	R	Interrupt ID	
0x0003	SC_LCR	R/W	Line control	
0x0004	SC_IMR	R/W	Interface Monitor	
0x0005	SC_LSR_	R	Line status	
0x0006	SC_BMC	R/W	Block Master Control	
0x0007	SC_ICR	R/W	Interface Control	
0x0008~ 0x000B	SC_FIFO_DATA	R/W	32 bit FIFO Data	
0x000C	SC_PRS	R/W	Protocol Status	
0x000D	SC_PRIP	R/W	Protocol/Timer Interrupts Pending	
0x000E	SC_PRIE	R/W	Protocol/Timer Interrupts Enables	
0x000F	SC_TMS	R	Timer Status	
0x0010~ 0x0011	SC_DLL/DLM	R/W	Baud Rate Divisor	
0x0012	SC_FCR	R/W	FIFO Control	
0x0013	SC_TOH	R/W	Timeout Timer	
0x0014~ 0x0015	SC_TOL/TOM	R/W	Timeout Timer	
0x0016 ~ 0x0017	SC_DCL/DCM	R/W	Down Counter	
0x0018 ~ 0x0019	SC_CWTL/CWTM	R/W	CWT Timer reload value	
0x001A	SC_GSRH	R/W	Guard Time High Register	
0x001B	SC_GSR	R/W	Guard algorithm Spacing Register	
0x001C	SC_EGT	R/W	Guard Timer Reload A	
0x001D	SC_BGT	R/W	Guard Timer Reload B	
0x001E	SC_PRM	R/W	Protocol Mode	
0x001F	SC_TCTL	R/W	Timer Control	
0x0025	SC_CLK_DIV	R/W	Frequency control	
0x0026	SC_CFG	R/W	SC Configuration	
0x0027	SC_LEDC	R/W	LED Control	
0x0028~ 0x0029	SC_FTHL/FTHM	R/W	FIFO Threshold	

Table 13.3 SmartCard Controller Registers

OFFSET ADDRESS	NAME	R/W	DESCRIPTION	PAGE
0x002A~ 0x002B	SC_FC	R	Number of bytes in FIFO	
0x002C	SC_FLL	R/W	Filter Length	
0x002D	SC_FINS	R/W	Filter INS Byte	
0x0030 ~ 0x0035	SC_TR0~4	R/W	Test Registers	
0x0080	SC_CTL	R/W	SC Control register	
0x0081	PAD_CTL_SC	R/W	Pad current control	
0x0090	SC_SYNC_RST	R/W	Syncronous mode Reset	
0x0094	SC_SYNC_CLK	R/W	Syncronous mode Clock	
0x0098	SC_SYNC_FCB	R/W	Syncronous mode FCB	
0x009C	SC_SYNC_SPU	R/W	Syncronous mode SPU	
0x00A0	SC_SYNC_IO	R/W	Syncronous mode Datat	
0x00A4	SC_SYNC_ALL	R/W	Syncronous mode All	

13.12 SmartCard Wrapper Control Registers

Table 13.4 Smart Card Control Register

SC_CTL (0X0080- RESET=0X00)			SMART CARD CONTROL REGISTER
BYTE	NAME	R/W	DESCRIPTION
7	INTERFACE_ENABLE	R/W	If the interface is not enabled, the interface pins are tri-stated. This bit must be cleared when using the pins in GPIO mode.
6	INF_IDLE_CTL_EN	R/W	Enable automatic control of interface idle condition. Setting this bit will force the interface into idle mode. It will automatically assert pull-down on the SC IOs. This is to force the interface into an all low state. Clearing this bit, all IOs are controlled by the SCC. This bit has no effect when INTERFACE_ENABLE = 0.
5	Reserved	R	Always read ‘0’
4	SC_EN_10K_PU	R/W	This bit changes the pull-up strength on the SmartCard interface 0 = 20Kohm pull-up 1 = 10Kohm pull-up
3	Reserved	R	Reserved for other implementations

SC_CTL (0X0080- RESET=0X00)			**SMART CARD CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
2	SC_SLOW_CLK	R/W	Must be set when SC_CLK is running under 10Mhz.
1	SC_MODE	R/W	Forces the pads into a low current SmartCard mode with increased hysteresis. This applies to all SmartCard pins except SC_CLK
0	SYNC_MODE_SEL	R/W	Setting this bit put the Smart card interface into the synchronous mode.

13.12.1 Automatic Control of Idle Condition on Smart Card Interface

Smart Card specification requires that the interface signals be held at zero until a card is inserted, power is applied to the card, and the reset sequence is started. The INF_IDLE_CTL_EN bit works in conjunction with the INTERFACE_ENABLE bit to do this. When the interface is in the idle state, (INTERFACE_ENABLE =0), pull-downs are enabled, and the control signals are driven zero. As soon as the interface is enabled, (INTERFACE_ENABLE =1) control of IO pad signals reverts to the Smart Card Controller (SCC). See figure Figure 13.12: *SmartCard Powerup*.

Table 13.5 Smart Card Current Control Register

PAD_CTL_SC (0X0081 - RESET=0X00)			**PAD CURRENT CONTROL**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.
1:0	SEL	R/W	00 = TBD (default) 01 = TBD Operation 10 = TBD Operation 11 = TBD Operation **Note:** This register only has effect when the SC interface is enabled.

13.13 Synchronous Interface Registers

All registers in the Synchronous interface must be byte addressable. This allows the firmware to toggle the output using byte writes without affecting any other register bits. There are five control lines associated with the interface that are controlled by five identical registers.

Table 13.6 Smart Card Sync RST Control Register

SC_SYNC_RST (0X0090- RESET=0X00000000)			**SMART CARD CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:14	Reserved	R	Always read ‘0’
13	INPUT_EN	R/W	‘1’ Input is enabled ‘0’ Input is disabled
12	OUTPUT_EN	R/W	‘1’ Output is enabled ‘0’ Output is disabled
11	FAST_OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the signal is driven low when the data is ‘0’ and when the data transitions to ‘1’, it is actively driven high for one clock cycle before being tri-stated.
10	OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the SC_RST output is driven open drain. ‘0’ are driven, ‘1’ are tri-stated.
9	PULL_UP_EN	R/W	When set, it enables the pull-up to this pin.
8	PULL_DN_EN	R/W	When set, it enables the pull-down to this pin.
7:2	Reserved	R	Always read ‘0’
1	RST_IN	R	This bit reflects the state of the SC_RST pin when select muxes are set to SmartCard mode and synchronous mode.
0	RST_OUT	R/W	This bit reflects the state of the SC_RST pin when select muxes are set to SmartCard mode and synchronous mode.

Table 13.7 Smart Card Sync CLK Control Register

SC_SYNC_CLK (0X0094- RESET=0X00000000)			**SMART CARD SYNC CLOCK CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:14	Reserved	R	Always read ‘0’
13	INPUT_EN	R/W	‘1’ Input is enabled ‘0’ Input is disabled
12	OUTPUT_EN	R/W	‘1’ Output is enabled ‘0’ Output is disabled
11	FAST_OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the signal is driven low when the data is ‘0’ and when the data transitions to ‘1’, it is actively driven high for one system clock cycle before being tri-stated.
10	OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the SC_CLK output is driven open drain. ‘0’ are driven, ‘1’ are tri-stated.
9	PULL_UP_EN	R/W	When set, it enables the pull-up to this pin.
8	PULL_DN_EN	R/W	When set, it enables the pull-down to this pin.
7:2	Reserved	R	Always read ‘0’

SC_SYNC_CLK (0X0094- RESET=0X00000000)			SMART CARD SYNC CLOCK CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
1	CLK_IN	R	This bit reflects the state of the SC_CLK pin when select muxes are set to SmartCard mode and synchronous mode.
0	CLK_OUT	R/W	This bit reflects the state of the SC_CLK pin when select muxes are set to SmartCard mode and synchronous mode.

Table 13.8 Smart Card Sync FCB Control Register

SC_SYNC_FCB (0X0098)- RESET=0X00000000)			SMART CARD FCB CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:14	Reserved	R	Always read ‘0’
13	INPUT_EN	R/W	‘1’ Input is enabled ‘0’ Input is disabled
12	OUTPUT_EN	R/W	‘1’ Output is enabled ‘0’ Output is disabled
11	FAST_OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the signal is driven low when the data is ‘0’ and when the data transitions to ‘1’, it is actively driven high for one system clock cycle before being tri-stated.
10	OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the SC_FCB output is driven open drain. ‘0’ are driven, ‘1’ are tri-stated.
9	PULL_UP_EN	R/W	When set, it enables the pull-up to this pin.
8	PULL_DN_EN	R/W	When set, it enables the pull-down to this pin.
7:2	Reserved	R	Always read ‘0’
1	FCB_IN	R	This bit reflects the state of the SC_FCB pin when select muxes are set to SmartCard mode. Synchronous or asynchronous mode does not matter.
0	FCB_OUT	R/W	This bit reflects the state of the SC_FCB pin when select muxes are set to Smart synchronous mode.Synchronous or asynchronous mode does not matter.

Table 13.9 Smart Card Sync SPU Control Register

SC_SYNC_SPU (0X009C- RESET=0X00000000)			SMART CARD SPU CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:14	Reserved	R	Always read ‘0’

SC_SYNC_SPU (0X009C- RESET=0X00000000)			**SMART CARD SPU CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
13	INPUT_EN	R/W	‘1’ Input is enabled ‘0’ Input is disabled
12	OUTPUT_EN	R/W	‘1’ Output is enabled ‘0’ Output is disabled
11	FAST_OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the signal is driven low when the data is ‘0’ and when the data transitions to ‘1’, it is actively driven high for one system clock cycle before being tri-stated.
10	OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the SC_SPU output is driven open drain. ‘0’ are driven, ‘1’ are tri-stated.
9	PULL_UP_EN	R/W	When set, it enables the pull-up to this pin.
8	PULL_DN_EN	R/W	When set, it enables the pull-down to this pin.
7:2	Reserved	R	Always read ‘0’
1	SPU_IN	R	This bit reflects the state of the SC_SPU pin when select muxes are set SmartCard mode. Synchronous or asynchronous mode does not matter.
0	SPU_OUT	R/W	This bit reflects the state of the SC_SPU pin when select muxes are set to SmartCard mode. Synchronous or asynchronous mode does not matter.

Table 13.10 Smart Card Sync IO Control Register

SC_SYNC_IO (0X00A0- RESET=0X00000000)			**SMART CARD IO CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:14	Reserved	R	Always read ‘0’
13	INPUT_EN	R/W	‘1’ Input is enabled ‘0’ Input is disabled
12	OUTPUT_EN	R/W	‘1’ Output is enabled ‘0’ Output is disabled
11	FAST_OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the signal is driven low when the data is ‘0’ and when the data transitions to ‘1’, it is actively driven high for one system clock cycle before being tri-stated.
10	OPEN_DRAIN	R/W	If this bit is set, and the mode is output, the SC_IO output is driven open drain. ‘0’ are driven, ‘1’ are tri-stated.
9	PULL_UP_EN	R/W	When set, it enables the pull-up to this pin.
8	PULL_DN_EN	R/W	When set, it enables the pull-down to this pin.
7:2	Reserved	R	Always read ‘0’

SC_SYNC_IO (0X00A0- RESET=0X00000000)			SMART CARD IO CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
1	IO_IN	R	This bit reflects the state of the SC_IO pin when select muxes are set to SmartCard mode as well as synchronous mode.
0	IO_OUT	R/W	This bit reflects the state of the SC_IO pin when select muxes are set to Smart synchronous mode.

13.13.0.1 Sync All register

The SC_SYNC_ALL register provides parallel control to read and write all of the Smart Card pads at the same time. The bits CARD_RST_CNTL, CARD_CLK_CNTL, CARD_IO_CNTL, CARD_FCB_CNTL, CARD_SPU_CNTL provide read (and write) access to the respective Synchronous register's IN (and OUT) bits respectively.

The Synchronous register controls for each pad, such as INPUT_EN, OUTPUT_EN, FAST_OPEN_DRAIN, OPEN_DRAIN, PULL_UP, PULL_DOWN in the respective registers need to be programmed before write access to this register.

Note: The Smartcard 2 interface does not have C4, C8 pins defined.

Table 13.11 Smart Card Sync All Control Register

SC_SYNC_ALL (0X00A4- RESET=0X00000000)			SMART CARD ALL CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read '0'
5	CARD_SPU_CNTL (CARD_C8_CNTL)	R/W	A read indicates the status of SC_SYNC_SPU.SPU_IN bit. A write to this bit writes the SC_SYNC_SPU.SPU_OUT bit.
4	CARD_FCB_CNTL (CARD_C4_CNTL)	R/W	A read indicates the status of SC_SYNC_FCB.FCB_IN bit. A write to this bit writes the SC_SYNC_FCB.FCB_OUT bit.
3	CARD_IO_CNTL	R/W	A read indicates the status of SC_SYNC_IO.IO_IN bit. A write to this bit writes the SC_SYNC_IO.IO_OUT bit.
2	CARD_CLK_CNTL	R/W	A read indicates the status of SC_SYNC_CLK.CLK_IN bit. A write to this bit writes the SC_SYNC_CLK.CLK_OUT bit.
1	CARD_RST_CNTL	R/W	A read indicates the status of SC_SYNC_RST.RST_IN bit. A write to this bit writes the SC_SYNC_RST.RST_OUT bit.
0	CARD_VCC_CNTL	R/W	This bit when reset disables power to the Smart Card 1 (or 2) pads. Resetting this bit causes masking of PWR_SC1_EN (or PWR_SC2_EN) bit in POWER_CTL1 register, controlling the voltage regulators to the smartcard pads. This bit when set enables the PWR_SC1_EN (or PWR_SC2_EN) bits to control the voltage regulators to the smart card pads. The Voltage applied is indicated by non-zero values of PWR_SC1_EN (or PWR_SC2_EN) bits.

13.13.1 Synchronous Interface Output

The timing diagram shows how the output behaves under different register setting for the synchronous interface when configured as an output.

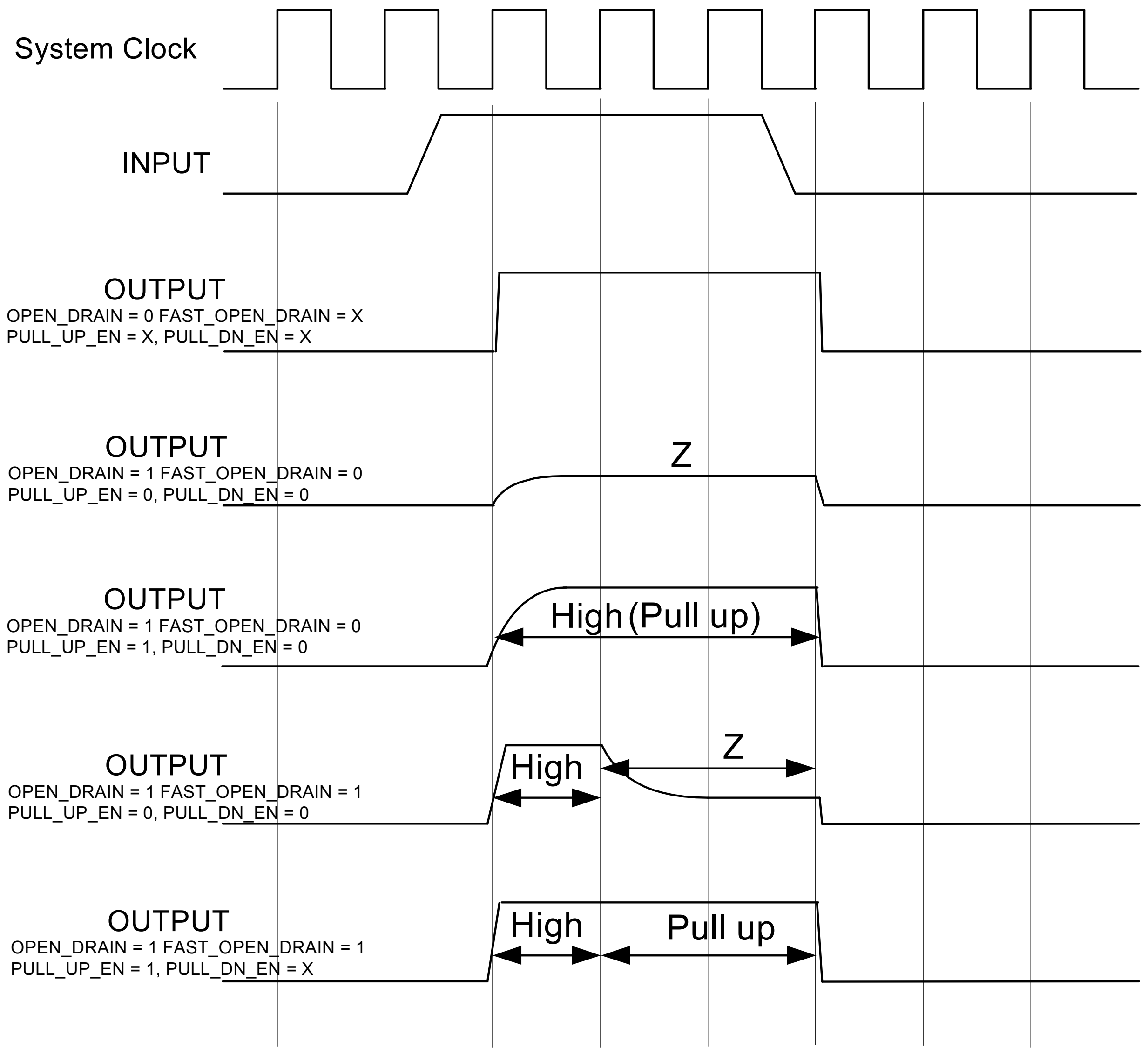

System Clock
INPUT
OUTPUT
OPEN_DRAIN = 0 FAST_OPEN_DRAIN = X
PULL_UP_EN = X, PULL_DN_EN = X
OUTPUT
OPEN_DRAIN = 1 FAST_OPEN_DRAIN = 0
PULL_UP_EN = 0, PULL_DN_EN = 0
Z
OUTPUT
OPEN_DRAIN = 1 FAST_OPEN_DRAIN = 0
PULL_UP_EN = 1, PULL_DN_EN = 0
High (Pull up)
OUTPUT
OPEN_DRAIN = 1 FAST_OPEN_DRAIN = 1
PULL_UP_EN = 0, PULL_DN_EN = 0
High
Z
OUTPUT
OPEN_DRAIN = 1 FAST_OPEN_DRAIN = 1
PULL_UP_EN = 1, PULL_DN_EN = X
High
Pull up

Figure 13.1 SmartCard Synchronous Output Configurations

13.14 Power

The SmartCard block is enabled when the SC_POWER_EN in turned on in the DEV_CLKEN register.

13.14.1 Card Deactivation

The Smart Card is deactivated by the following sequence:

First, SC_RST# is pulled low.

Next, SC_IO and SC_CLK are pulled low.

Finally, card power is removed from the SmartCard..

These are also the states of these pins while the SmartCard interface is reset or unpowered (VCC2 removed).

All receives are blocked when the SC_RST# pin goes low. If the smart card interface is not in loop back mode and the SC_RST# pin goes low while receiving a data byte, the following will occur: Any data received after the SC_RST# pin goes low will be ignored, and any partially-received character will be deleted.

13.14.2 Normal Deactivation

Under normal operation, the ICC-aware application will request that the Smart Card driver deactivate the smart card interface when the user has finished communicating with the smart card. This application will call the driver, which will deactivate the card by writing to the ICR register in the proper sequence, and then the user will be notified that the card can be removed from the device.

13.14.3 Unexpected Card Removal

If a card is removed (i.e., the SC_PSNT# signal goes high) before software has performed the deactivation sequence, then the hardware will initiate auto shutdown and deactivate the smart card interface. The GPIO interrupt will then be asserted to inform software that this has happened.

Any time the SC_PSNT# pin is high the hardware will hold the ICR register in its reset state. This will force the Smart Card interface pins to remain in their low states, and the 5V_EN# and 3V_EN# pins to float high, ensuring that the ICC's power remains off.

13.15 Asynchronous Interface Registers

13.15.1 Asynchronous Mode Registers

Table 13.12 Smart Card Transmit/Recive Buffer Register

SC_TBR_RBR (0X0000- RESET=0XXX)			**SMART CARD TRANSMIT/RECEIVE BUFFER REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	DATA	R?W	Writing to this register causes the byte to be written to the FIFO, and an internal count is incremented for determining the length of the message to be transmitted. Writing too much information will cause the message to be silently truncated to the length of the FIFO. Reading from this register causes a byte to be read from the FIFO. This decrements the FIFO Count register. If the FIFO Count register is already zero, this causes the UE bit in the Line Status register to be set to '1', and the receiver is disabled from writing to the FIFO.

Table 13.13 Smart Card Interrupt Enable Register

SC_IEN (0X0001- RESET=0X00)			**SMART CARD INTERRUPT ENABLE REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	PRTI	R/W	’1’ enables the Protocol and Timer interrupt. The sources of this interrupt are itemized in register PRIP.
6	OCSI	R/W	Set to ‘0’. Do not use for SEC2410/SEC4410.
5	GPI	R/W	Set to ‘0’. Do not use for SEC2410/SEC4410.
4	PTI	R/W	Set to ‘0’. Do not use for SEC2410/SEC4410
3	Reserved	R/W	Always write this bit as ‘0’.
2	RLSI	R/W	‘1’ enables an interrupt on Line Status errors: Parity, Framing, Overflow or Underflow
1	THRRI	R/W	‘1’ enables an interrupt when the transmitter has finished transmission of a message, including the minimum Guard Time (Stop bits).
0	RDAI	R/W	‘1’ enables an interrupt when FIFO data is available to read, either by the Threshold value or by any data at all in the FIFO after a Timeout condition (e.g. the CWT Timer).

13.15.1.1 Interrupt Identification

By accessing this register, the host CPU can determine the highest priority interrupt and its source. Four levels of priority interrupt exist. They are, in descending order of priority:

Receiver Line Status (highest priority)

Received Data Ready

Transmitter Holding Register Empty or Threshold has been reached

Protocol / Timer Interrupt

Information indicating that a prioritized interrupt is pending and the source of that interrupt is stored in the SC Interrupt Identification Register (refer to Interrupt Control Table). When the CPU accesses the IIR, the Smart Card Interface freezes all interrupts and indicates the highest priority pending interrupt to the CPU. During this CPU access, even if the Smart Card Interface records new interrupts, the current indication does not change until either the interrupt is re-enabled or the event causing the interrupt is cleared and re-asserted. The contents of the SC_IIR are described below.

Note: Interrupts are re-enabled by writing a ‘1’ to the interrupt enable bit. This bit does not need to be cleared to re-enable interrupts.

Table 13.14 Smart Card Interrupt Identification Register

SC_INT_ID (0X0002- RESET=0B00XX00XX1)			**SMART CARD INTERRUPT IDENTIFICATION REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	PRTI	R/W	’1’ indicates the presence of a Protocol or Timer interrupt. The sources of this interrupt are itemized in register PRIP, and are cleared by reading that register.
6	OCSI	R/W	Do not use, SC_IEN to keep diabled
5	GPI	R/W	Do not use, SC_IEN to keep diabled
4	PTI	R/W	Do not use, SC_IEN to keep diabled
3	FTO	R/W	FIFO Timeout. ‘1’ indicates a FIFO Data Timeout caused by the CWT timer, or by the Timeout Timer in T=0 mode, rather than the amount of received data reaching the Threshold value. It also indicates that the Receiver will be delivering no more data bytes to the FIFO. This bit is not an interrupt source, but is instead a status bit, which should be examined when processing the RDAI interrupt. This bit is cleared by emptying or resetting the FIFO.
2:1	PRI	R/W	If the IP bit in this register is ‘0’ (active), then this field holds the source of the interrupt
0	IP	R/W	A ‘0’ in this bit position indicates that an interrupt is pending, and that the PRI field of this register indicates the highest priority level pending. A ‘1’ in this position means that no interrupt is pending.

Note: The traditional UART FIFO Control Register functions are no longer in a write-only register at this address. Instead, the FCR register is a read/write register at location 1/02, and the Threshold is in a separate pair of registers.

Table 13.15 Interrupt Control Table

INTERRUPT ID REGISTER FIELDS											
PRTI	**OCSI**	**GPI**	**PTI**	**FTO**	**PRI**		**IP**				
BITS											
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**PRIORITY LEVEL & ENABLE**	**INTR. TYPE**	**INTR. SOURCE**	**INTR. RESET CONTROL**
X	NA	NA	NA	X	X	X	1	-	None	None	-
X	NA	NA	NA	X	1	1	0	First IEN bit 2	Line Status	Overrun Error, Parity Error, Frame Error, Underflow Error, or TF (Guard Algorithm Timeout)	Reading the Line Status Register

Table 13.15 Interrupt Control Table

INTERRUPT ID REGISTER FIELDS											
PRTI	OCSI	GPI	PTI	FTO	PRI		IP				
BITS											
7	6	5	4	3	2	1	0	PRIORITY LEVEL & ENABLE	INTR. TYPE	INTR. SOURCE	INTR. RESET CONTROL
X	NA	NA	NA	0	1	0	0	Second IEN bit 0	Received Data Available	Receiver Data Available	Reading from the FIFO until its level drops below the threshold level
X	NA	NA	NA	1	1	0	0	Second IEN bit 0	Character Timeout Indication	CWT or Timeout Timer underflow with data in FIFO.	Reading from the FIFO
X	NA	NA	NA	X	0	1	0	Third IEN bit 1	Transmit Finished	Transmit Phase of Exchange is complete	Reading the IID Register
1	NA	NA	NA	X	0	0	0	Fourth IEN bit 7	Protocol Timer Timeout	GP Counter underflow (normal) or Timeout, CWT or Guard timer underflow (errors)	Reading the PRIP Register

Table 13.16 Smart Card Line Control Register

SC_LCR (0X0003- RESET=0X00)			SMART CARD LINE CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:6	DLAB	R	These bits are forced to zero.
5	DCEN	R/W	General Purpose Down Counter Enable. ‘1’ starts the counter. See Section, "16-bit General Purpose Counter," for details.
4:2	Reserved	R	Always read ‘0’

SC_LCR (0X0003- RESET=0X00)			SMART CARD LINE CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
1	APDE	R/W	Automatic Parity-error Deactivate Enable. '1' causes the ICC to be deactivated by hardware upon a non-recoverable parity error. The device must also be in T=0 mode for this to occur. If the CRE bit is also '0', this will occur without performing character repetition or signalling to the ICC.
0	CRE	R/W	Character Repeat Enable. '1' enables character repeat in T=0 mode if a Parity Error is signalled by the ICC.

Table 13.17 Smart Card Interface Monitor Register

SC_INTF_MON (0X0004- RESET=0B00XX0XX0)			SMART CARD INTERFACE MONITOR REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	FFULL	R/W	FIFO Full. This bit indicates that the FIFO is completely full with data to be transmitted.
6	Reserved	R	Always read '0'
5	PSNT	R/W	This pin reflects the state of the SC_PSNT_N pin.
4	CRMV	R/W	Card removed. This bit is set to '1' when a card is being removed. It is a read-only '1', and cannot be cleared by software, as long as the debounced version of the SC_PSNT_N signal is high. When SC_PSNT_N goes low, this bit can be cleared by writing a '1' to it. While this bit is '1', the ICR register is held to its default state, which holds the signals SC_IO, SC_CLK and SC_RST_N all low
3	FTH	R/W	'1' indicates the presence of a FIFO Threshold interrupt request.
2	RST_N	R/W	Indicates the current state of the SC_RST_N pin.
1	IO	R/W	Indicates the current state of the SC_IO pin.
0	CRPT	R/W	Indicates, in T=0 mode or ATR mode, whether any characters needed to be repeated to the ICC. This bit may be cleared by writing a '1' to it. This is an indicator only.

Table 13.18 Smart Card Line Status Register

SC_LSR (0X0005- RESET=0XXX)			SMART CARD LINE STATUS REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	ETR	R/W	Indicates whether a Parity Error (Bit 2) occurred in the Transmit phase ('0') or the Receive phase ('1') of an exchange.
6	TRANSMIT_EMPTY	R/W	This bit is cleared to '0' at the beginning of transmission, and is set to '1' when the transmission completes, including guard time (Stop bit[s]) of the last character

SC_LSR (0X0005- RESET=0XXX)			**SMART CARD LINE STATUS REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
5	TRANSMIT_FAILURE	R/W	This indicates that a Guard Time Algorithm failure occurred
4	UNDERFLOW_ERROR	R/W	'1' indicates that a software error has caused an attempt to read from the FIFO while it is empty. Since this can add indeterminate bytes to a message, the receiver is disabled to the FIFO, by clearing the FRE bit.
3	FRAMING_ERROR	R/W	'1' indicating that a Framing Error has been seen on received data. It disables the receiver from the FIFO, by clearing the FRE bit in the FCR register upon its occurrence, after placing the character with the error into the FIFO. Reading this register clears this bit
2	PARITY_ERROR	R/W	'1' indicating a Parity Error. It disables the receiver or the transmitter from the FIFO upon its occurrence, by clearing the FRE or FTE bit in the FCR register. If the error is seen while receiving, the FRE bit will be cleared after receiving the character with the error into the FIFO. Reading this register clears this bit. If the APDE bit in the LCR register is '1', the error will also deactivate the ICC immediately by hardware action.
1	OVERRUN_ERROR	R/W	'1' indicates that too much data has been received from the ICC, so that the FIFO became completely full and lost a character. This error disables the receiver or the transmitter from the FIFO upon its occurrence, by clearing the FRE bit. Note that attempting to transmit a message longer than the FIFO length will silently truncate the message, but will not set this bit.
0	DATA_READY	R/W	'1' indicates that the FIFO is not empty of received data. This bit is not affected by reading this register.

Note: All bits except Bit 0 are automatically cleared after reading this register.

Table 13.19 Smart Card Block Master Control Register

SC_BMC (0X0006- RESET=0X00)			**SMART CARD BLOCK MASTER CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read '0'
1	GIE	R/W	Global Interrupt Enable bit. A '0' in this bit position disables all interrupts from the Smart Card interface
0	MRST	R/W	Software-controlled Master Reset control bit. Set this bit to '1' to reset the Smart Card block. The Configuration section is not affected, and the GPIO section is not affected except that interrupts are disabled in the IEN register. When the bit returns to '0', hardware is indicating that the reset is complete.

Table 13.20 Smart Card Interface Control Register

SC_ICR (0X0007- RESET=0B00XX01000)			**SMART CARD INTERFACE CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	RST_N	R/W	SC_RST_N Pin Control. The default value (0) holds the SC_RST_N pin low. A '1' in this bit causes the SC_RST_N pin to drive high. This bit may be written to '1' or '0' by software, and the first underflow of the Guard Timer, while the Protocol Mode register is indicating ATR Mode, sets this bit to '1', and causes the SC_RST_N pin to rise as part of the Reset/ATR sequence.
6	ENG	R/W	Enable Guard Timer. Writing '1' enables the Guard Timer to begin counting at the next triggering event. Writing '0' has no effect: to clear this bit, write '1' to the RSG bit in the Timer Control register. This bit is cleared by hardware in ATR mode when the first Start bit is seen, or on an underflow from the BGT reload. In the second case, an interrupt request is also presented
5:4	VPIN	R/W	Not used for SEC2410/SEC4410.
3	CSTP	R/W	Clock Stop. '1' stops the SC_CLK signal either high or low, depending on the CSTL bit. '0' causes the SC_CLK signal to run. This signal is initially '1' on reset, causing SC_CLK to be stopped in the low state.
2	CSTL	R/W	Clock Stop Level. When the CLKSTP bit is set, this bit indicates the state in which the SC_CLK pin should stop: '1' means stop the clock high, '0' means stop the clock low. This bit is initially '0' on reset, causing SC_CLK to be stopped in the low state
1	IO	R/W	SC_IO Pin Control. The default value (0) forces the SC_IO pin low. Writing a '1' to this bit enables the SC_IO pin to float and to drive high
0	IOPU	R/W	'1' enables a weak pull-up device on the SC_IO pin. This device is internally disabled while the transmitter is actively driving the SC_IO pin.

Table 13.21 Smart Card Data Register

SC_DATA (0X0008~0X000B-RESET=0XXXXXXXXX)			**SMART CARD DATA REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:0	DATA	R/W	Perform all transfers at the location DATA, regardless of size. Transferring a value at the DATA location has the same effect as transferring the individual bytes (LS byte first) at the TBR/RBR location (0000), but is more efficient for the larger data types.

Table 13.22 Smart Card Protocol Status Register

SC_PRS (0X000C- RESET=0X00)			**SMART CARD PROTOCOL STATUS REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5	SMB	R/W	State Machine Busy. A ‘1’ in this bit indicates that a transfer is in progress. A ‘0’ indicates that no transfer is in progress (idle/finished)
4	PWR	R/W	This bit is forced to ‘0’
3	ACTV	R/W	Activity Bit. ‘1’ indicates that a character has been received since the last time this bit was cleared by software. This bit is cleared by software, by writing a ‘0’ to this bit location. (This is the only writable bit in this register.) Only the RSE bit in the FCR register has to be ‘1’ in order for this bit to detect activity, and the FRE bit does not have to be ‘1’
2	GPH	R/W	Guard Timer Phase. This bit indicates the current phase of operation for the Guard Timer: ‘0’ = next reload will be from EGT register, ‘1’ = next reload will be from BGT register
1	TSM	R/W	TS Mode. This bit indicates the current Convention: ‘0’ = Direct, ‘1’ = Inverse. Writing a ‘1’ to the ATR bit in the Protocol Mode register initializes this bit to ‘0’, and it can be manipulated using some Test Register features. Otherwise, it is a read-only bit.
0	TSC	R/W	TS Captured. ‘1’ indicates that a Convention has been automatically captured from an ATR TS byte. Writing a ‘1’ to the ATR bit in the Protocol Mode register initializes this bit to ‘0’, and it can be manipulated using some Test Register features. Otherwise, it is a read-only bit.

Table 13.23 Smart Card Protocol Interrupt Pending Register

SC_PRIP (0X000D- RESET=0X00)			**SMART CARD PROTOCOL INTERRUPT PENDING REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	GPT	R/W	‘1’ = General Purpose Down Counter Interrupt
6	TSW	R/W	‘1’ = Timeout waiting for the TS byte in ATR mode. (Guard Timer, EGT reload phase.)
5	TMO	R/W	‘1’ = Timeout on the Timeout Timer (WWT, BWT or WTX)
4	CWT	R/W	‘1’ = Timeout on the CWT Timer (CWT, or timeout waiting for the ATR TS byte)
3	NULL	R/W	This bit if set indicates to the processor that a NULL byte was received. This bit may be used in T=0 mode, to detect NULL byte reception, and indicate to host software
2	EOM	R/W	‘1’ = End of Message indication from one of the T=0 Filter State Machines. If communication terminates prematurely or with an error, the CV bit will also be ‘1’.

SC_PRIP (0X000D- RESET=0X00)			**SMART CARD PROTOCOL INTERRUPT PENDING REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
1	COLL	R/W	'1' = That bus contention was detected during a transfer.
0	CV	R/W	This is a status bit, not an interrupt source. A ‘1’ indicates that a Code Violation has occurred; either a bad TS value during ATR, or, in T=0 mode with a Filter State Machine enabled, either an unrecognized Procedure Byte or an ST1 byte earlier than expected

Warning: Reading the SC_PRIP register clears all the bits, and removes the associated interrupt requests.

Table 13.24 Smart Card Protocol Interrupt Enable Register

SC_PRIE (0X000E- RESET=0X00)			**SMART CARD PROTOCOL INTERRUPT PENDING REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	GPT	R/W	‘1’ enables General Purpose Down Counter Timeout.
6	TSW	R/W	‘1’ enables TSW Timeout waiting for the TS byte in ATR mode. (Guard Timer, EGT reload phase.
5	TMO	R/W	‘1’ enables TMO Timeout on the Timeout Timer
4	CWT	R/W	‘1’ enables CWT Timeout on the CWT Timer
3	NULL	R/W	1’ enables NULL interrupt
2	EOM	R/W	‘1’ enables EOM End of Message
1	COLL	R/W	'1' enables Collision Interrupt
0	CDVI	R/W	'1' enables Code Violation

Note: This register enables the interrupts coming from the PRIP register

Table 13.25 Smart Card Timer Status Register

SC_TMS (0X000F- RESET=0X00)			**SMART CARD TIMER STATUS REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read ‘0’
4	GS_MAX_TIMEOUT	R/W	'1' indicates a GSR timeout occured.
3	TORUN	R	‘1’ indicates that the Timeout Timer has been triggered and is running
2	Reserved	R	Always read ‘0’

SC_TMS (0X000F- RESET=0X00)			SMART CARD TIMER STATUS REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
1	CRUN	R	‘1’ indicates that the CWT Timer has been triggered and is running
0	GRUN	R	‘1’ indicates that the Guard Timer has been triggered and is running

Table 13.26 Smart Card Baud Divisor LSB Register

SC_DLL (0X0010- RESET=0X00)			SMART CARD BAUD DIVISOR LSB REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	BAUD_DIV_7_0	R/W	These are the lower 8 bits of the 16 bit baud rate divisor. The most signicant 2 bits are held in the SC_DLM register. The baud rate divisor, with the Sampling field of the CLK register, divides the etu rate from the SMC input clock.

Table 13.27 Smart Card Baud Divisor MSB Register

SC_DLM (0X0011- RESET=0X00)			SMART CARD BAUD DIVISOR MSB REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	BAUD_DIV_15_8	R/W	These are the most significant 8 bits of the 16 bit baud rate divisor. The least signicant 8 bits are held in the SC_DLL register. The baud rate divisor, with the Sampling field of the CLK register, divides the etu rate from the SMC input clock.

Table 13.28 Smart Card FIFO Control Register

SC_FCR (0X0012- RESET=0X00)			SMART CARD FIFO CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’

SC_FCR (0X0012- RESET=0X00)			**SMART CARD FIFO CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
5	RFS	R/W	Receiver FIFO Status, Read-only (RO). This bit indicates whether the Receiver is actively prepared to place characters into the FIFO. It may not match the FRE bit, if the Receiver is still waiting for a trigger to begin (e.g., waiting for transmission to complete).
4	RSS	R/W	Receiver Sampling Status, Read-only (RO). This bit indicates whether the Receiver is actively sampling for characters. It may not match the RSE bit, if the Receiver is still waiting for a trigger to begin. For example, in ATR mode, it may not yet be active, pending a rising edge on the SC_RST_N pin.
3	RSE	R/W	Receiver Sampling Enable, R/W. '1' written to this bit enables the Receiver to sample the SC_IO pin for characters. In ATR mode, the sampling does not occur immediately, but waits for a rising edge on the SC_RST_N pin first. This bit is read-write, and is cleared by an incoming error (e.g. repeated parity error in T=0 mode, or CWT violation in T=1 mode, or Overrun Error). While the receiver is sampling, the BGT or DGT value in the Guard Timer Register continues to be used to inhibit the Transmitter, regardless of the state of the FRE bit.
2	FRST	R/W	FIFO Reset, write-only, reads as '0' always. '1' written to this bit resets the FIFO to an Empty state. If an error has occurred while transmitting to the card, this function must be used to re-initialize the FIFO.
1	FRE	R/W	FIFO Receive Enable. Allows reception into the FIFO. Except in ATR mode, a transmission has to occur before the receiver is actually activated. In ATR mode, a rising edge must occur on the SC_RST_N pin before the receiver is activated. This bit is turned off by errors occurring during reception or transmission (e.g., CWT timeout error); otherwise software must turn it off after receipt of a message, to prepare for the next exchange
0	FTE	R/W	FIFO Transmit Enable. Writing '1' to this bit triggers transmission from the FIFO. This bit is turned off by the normal end of transmission, when all bytes in the FIFO have been transmitted. It is also turned off by errors occurring during transmission (e.g., parity error after retransmissions in T=0 mode).

Note: This register provides control for FIFO access, and enables the Receiver and the Transmitter

Table 13.29 Smart Card Timeout Timer MSB Reload Register

SC_TOH (0X0013- RESET=0X00)			**SMART CARD TIMEOUT TIMER M<SB RELOADREGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMER_RELOAD_HI	R/W	This register holds bits 23:16 the reload value for the Timeout Timer in units of 1.25 milliseconds.

Table 13.30 Smart Card Timeout Timer LSB Reload Register

SC_TOL (0X0014- RESET=0X00)			**SMART CARD TIMEOUT TIMER LSB RELOAD REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMER_RELOAD_LO	R/W	This register holds bit 7:0 of the reload value for the Timeout Timer in units of 1.25 milliseconds.

Table 13.31 Smart Card Timeout Timer mid Reload Register

SC_TOM (0X0015- RESET=0X00)			**SMART CARD TIMEOUT TIMER MID RELOAD REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMER_RELOAD_MIDI	R/W	This register holds 15:8 of the reload value for the Timeout Timer in units of 1.25 milliseconds.

Table 13.32 Smart Card Down Counter LSB Register

SC_DCL (0X0016- RESET=0XFF)			**SMART CARD DOWN COUNTER LSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	DOWN_CNT_LO	R/W	This register holds the LSB of the General Purpose Down Counter.

Table 13.33 Smart Card Down Counter MSB Reload Register

SC_DCM (0X0017- RESET=0XFF)			**SMART CARD DOWN COUNTER MSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	DOWN_CNT_HI	R/W	This register holds the MSB of the General Purpose Down Counter

Table 13.34 Smart Card CWT Timer LSB Reload Register

SC_CWTL (0X0018- RESET=0X00)			**SMART CARD CWT TIMER LSB RELOAD REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMER_RELOAD_LO	R/W	This register holds the LSB of the reload value for the CWT Timer.

Table 13.35 Smart Card CWT MSB Reload Register

SC_CWTM (0X0019- RESET=0X00)			**SMART CARD CWT TIMER MSB RELOAD REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TIMER_RELOAD_HI	R/W	This register holds the MSB of the reload value for the CWT Timer.

Table 13.36 Smart Card Guard Algorithm Spacing Register

SC_GSRH (0X001A- RESET=0X00)			**SMART CARD GUARD ALGORITHM SPACING REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	GUARD_ETUS[15:8]	R/W	This register holds the MS of the maximum spacing between characters, specified as the number of etus from the leading edges of consecutive Start bits.

Table 13.37 Smart Card Guard Algorithm Spacing Register

SC_GSR (0X001B- RESET=0X00)			**SMART CARD GUARD ALGORITHM SPACING REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	GUARD_ETUS[7:0]	R/W	This register holds LS byte of the maximum spacing between characters, specified as the number of etus from the leading edges of consecutive Start bits.

Table 13.38 Smart Card Guard Timer Reload A Register

SC_EGT (0X001C- RESET=0X00)			**SMART CARD GUARD TIME RELOAD A REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	RELOAD_A	R/W	This register holds the Extra Guard Time value in T=0 or T=1 mode. In ATR mode, this register holds the maximum number of etu's allowed from the rising edge of SC_RST_N to the Start bit of the TS byte. If the timer elapses, the TSW interrupt is asserted, and the Receiver is disabled to the FIFO. Values are expressed in units of etu. The SC_PRM Register must be written after writing to this register, in order to latch the change.

Table 13.39 Smart Card Guard Timer Reload B Register

SC_BGT (0X001D- RESET=0X00)			**SMART CARD GUARD TIME RELOAD B REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	RELOAD_B	R/W	This register holds the BGT value in T=1 mode, or the DGT value in T=0 mode, preventing transmission until the specified number of etu's has elapsed since the last received character. Monitoring of characters for this purpose does not depend on whether the receiver is enabled to the FIFO. This timer must be enabled, or it will not delay transmission. In ATR mode, this register holds the desired width of the SC_RST_N pulse (Warm Reset) or the duration of the clock before the removal of SC_RST_N. Values are expressed in units of etu. The SC_PRM Register must be written after writing to this register, in order to latch the change.

13.15.1.2 Protocol Mode Register

The Guard Time Reload registers EGT and BGT must be initialized to their desired values before writing to this register. Changing them afterward may fail to register the change.

All non-reserved bits are read/write. The ATR bit may be set to '1' only if the TE1 bit is also set to '0'. Valid settings for these two bits are:

ATR Mode: ATR=1 and TE1=0. In this mode, the Protocol Timers and the Receiver are conditioned to expect an ATR message from the ICC. Character framing is as per the T=0 protocol. This is the one case where the Receiver does not wait for the Smart Card Interface to transmit first: instead, it waits for a rising edge on the SC_RST_N pin, which is being controlled by the Guard Timer.

T=0 Mode: ATR = 0 and TE1 = 0. In this mode, character framing and parity handling are as per the T=0 protocol. The Receiver waits until a message has been transmitted before it becomes active.

T=1 Mode: ATR = 0 and TE1 = 1. In this mode, character framing and parity handling are as per the T=1 protocol. The Receiver waits until a message has been transmitted before it becomes active.

The OSME and ISME bits are mutually exclusive: only one of them may be set to ‘1’, and neither may be set to ‘1’ without the TE1 bit also being set to ‘0’ and the ATR bit set to ‘0’.

Table 13.40 Smart Card Protocol Mode Register

SC_PRM (0X001E- RESET=0X00)			**SMART CARD REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read ‘0’
4	ISME	R/W	‘1’ means that the Incoming Filter State Machine is enabled. The TE1 bit and ATR bit must also be set to ‘0’.
3	OSME	R/W	‘1’ means that the Outgoing Filter State Machine is enabled. The TE1 bit and ATR bit must also be set to ‘0’
2	Reserved	R	Always read ‘0’
1	TE1	R/W	‘0’ means that T=0 character framing is being used, either in T=0 protocol commmunication or receiving the ATR message. ‘1’ means the T=1 protocol is being used. This bit may not be set to ‘1’ with any of bits ATR, OSME or ISME also set to ‘1’
0	ATR	R/W	Answer To Reset mode. ‘1’ means that a Reset sequence is to be presented, expecting a response from the card. The TE1 bit must also be ‘0’ in this mode. Writing a ‘1’ to this bit also clears the TSC and TSM bits in the Protocol Status register, which causes the first byte received to be interpreted by hardware as the TS byte, setting the bit encoding convention based on what is received.

Table 13.41 Smart Card Timer Control Register

SC_TCTL (0X001F- RESET=0X00)			**SMART CARD TIMER CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7	RSG	R/W	Resets the Guard Timer. This bit reads as ‘0’ always. Writing a ‘1’ to this bit clears the ENG bit in the Interface Control register to ‘0’, and removes any pending interrupt request from the Guard Timer. (The ENG bit, which enables the Guard Timer, is in the Interface Control register so that the Guard Timer may be started atomically with the presentation of SC_RST_N and SC_CLK to the Smart Card.)
6:5	Reserved	R	Always read ‘0’
4	RSC	R/W	Resets the CWT Timer. This bit reads as ‘0’ always. Writing a ‘1’ to this bit clears the ENC bit to ‘0’, and removes any pending interrupt request from the CWT Timer.

SC_TCTL (0X001F- RESET=0X00)			SMART CARD TIMER CONTROL REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
3	ENC	R/W	Writing ‘1’ enables the CWT Timer to begin counting at the next triggering event. Writing ‘0’ has no effect: to clear this bit, write ‘1’ to the RSC bit in the Timer Control register. This bit is cleared by hardware action in order to stop the timer.
2	WTX	R/W	‘1’ Places the Timeout Timer in WTX Mode; ‘0’ places it in BWT Mode. In WTX mode, the Timeout Timer underflow reloads the Timeout Timer instead of stopping it, and the Receiver is not disabled on underflow.
1	RSTO	R/W	Resets the Timeout Timer. This bit reads as ‘0’ always. Writing a ‘1’ to this bit clears the ENTO bit to ‘0’, and removes any pending interrupt request from the Timeout Timer.
0	ENTO	R/W	Writing ‘1’ enables the Timeout Timer to begin counting at the next triggering event. Writing ‘0’ has no effect: to clear this bit, write ‘1’ to the RSTO bit in the Timer Control register. This bit is cleared by hardware action in order to stop the timer.

Table 13.42 Smart Card Clock Divisor Register

SC_CLK_DIV (0X0025- RESET=0X00)			SMART CARD CLOCK DIVISOR REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:6	SAMPLING		This field indicates a divisor to apply from the DLL/DLM value in order to get the final etu rate: 00 divide by 31 10 divide by 16 01 divide by 1 11 is reserved for future use. The SC_CLK Divisor field is reduced in size to 6 bits
5:0	DIVISOR	R/W	This field gives the divisor to apply to the SEC2410/SEC4410 system clock in order to generate the SC_CLK signal to the ICC.

Table 13.43 Smart Card Configuration Block Register

SC_CFG (0X0026- RESET=0XXX)			SMART CARD REGISTER
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	Reserved	R	Do not modify

In SEC2410/SEC4410, the SC_CFG is hardwired to zero.

Table 13.44 Smart Card LED Control Register

SC_LEDC (0X0027- RESET=0X00)			**SMART CARD LED CONTROL REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always read ‘0’
2	LMD	R/W	LED Mode. 0 = LED is controlled by the LED Control field in this register. 1 = LED is controlled by activity on the SC_IO pin. When there is activity on the SC_IO pin the LED will blink at an approximate 5.5Hz rate with a 50% duty cycle
1:0	LCTL	R/W	LED Control. 00 = Off 01 = Blink at 1Hz rate with a 50% duty cycle (0.5 sec on, 0.5 sec off) 10 = Blink at ½ HZ rate with a 25% duty cycle (0.5 sec on, 1.5 sec off) 11 = On

13.15.1.3 Fifo Threshold Registers

These registers hold the FIFO threshold for received bytes. The FIFO Threshold interrupt is asserted when the number of received bytes in the FIFO exceeds the number provided here. For example, set these registers to 0000 hex to be interrupted on every byte received. The interrupt is also asserted on a timeout of the CWT Timer, or of the Timeout Timer in T=0 mode, regardless of the contents of these registers.

These registers have no effect on transmission: the number of bytes present in the FIFO at the time that the FTE bit is set to ‘1’ determines the length of the messsage transmitted.

Table 13.45 Smart Card FIFO Threshold LSB Register

SC_FTHL (0X0028- RESET=0X00)			**SMART CARD FIFO THRESHOLD LSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	FIFO_THRESHOLD_LO	R/W	This register hold the LSB FIFO threshold for received bytes.

Table 13.46 Smart Card FIFO Threshold MSB Register

SC_FTHM (0X0029- RESET=0X00)			**SMART CARD FIFO THRESHOLD MSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	FIFO_THRESHOLD_HI	R/W	This register hold the MSB FIFO threshold for received bytes.

13.15.1.4 Fifo Count Registers

This register pair holds the number of bytes currently in the FIFO.

While setting up for transmission, and during transmission, this register tracks bytes being transmitted. If there is an error in transmission, the transmitter stops and this register holds the number of bytes remaining in the FIFO. In case of a transmission error, the FIFO must be reset using the FRST bit in the FCR register. This action will also clear these registers to zero. During transmission (i.e., while the Receiver is not active), the value in these registers is not compared against the Threshold value in the FTHL/FTHM register pair.

While the Receiver is active, this register pair also tracks the number of bytes in the FIFO, and this value is compared against the FIFO Threshold in the FTHL/FTHM register pair in order to provide the FIFO Threshold interrupt.

To determine whether an error happened during the Transmit or Receive phase of an exchange (and hence which count is being displayed in this register), software may inspect the ETR bit in the Line Status register.

Table 13.47 Smart Card FIFO Count LSB Register

SC_FCL (0X002A- RESET=0X00)			**SMART CARD FIFO COUNT LSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	FIFO_COUNT_LO	R	This register hold the LSB of the FIFO count in bytes.

Table 13.48 Smart Card FIFO Count MSB Register

SC_FCM (0X002B- RESET=0X00)			**SMART CARD FIFO COUNT MSB REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	FIFO_COUNT_HI	R	This register hold the MSB of the FIFO count in bytes.

Table 13.49 Smart Card Filter Length Register

SC_FLL (0X002C- RESET=0X00)			**SMART CARD FILTER LENGTH REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	FILTER_LEN	R/W	This register holds the number of expected data bytes in a T=0 exchange, for the sake of the T=0 Filter state machines. This register is decremented as needed by the Outgoing Filter state machine. An initial value of 00h, when the Outgoing filter is activated, is interpreted as 256. An initial value of 00h, when the Incoming filter is activated, is interpreted as 0. Any T=0 command that does not involve a data transfer will use the Incoming filter with an initial count of 00h. This register returns the least-significant 8 bits of the current Count value when read.

Table 13.50 Smart Card INS Code Register

SC_FINS (0X002D- RESET=0X00)			**SMART CARD FILTER STATE MACHINE INS CODE REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	INS	R/W	This register holds the INS byte for the current T=0 exchange, so that the T=0 Filter state machines can recognize the INS and INS Procedure Bytes

Table 13.51 Smart Card Test Registers

SC_TEST (0X0035, - RESET=0X00)			**SMART CARD TEST REGISTERS**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
7:0	TEST	R/W	This register is reserved for SMSC test purposes

13.16 AC Timing specification

There is no AC specification included for the following reasons:

1. The interface is asynchronous, so by definition, there is no timing relationship between the clock edge and other signals. There is a relationship between the number of clocks and data signal.
2. The signals are open drain. The rise time is determined by the pull-up resistor and load impedance.
3. The fall time has to be less than 1usec. This is guaranteed by the minimum current output.

Chapter 14 AES Interface with SRAM

14.1 Overview

SEC2410/SEC4410 has an AES cryptographic engine and 10KB SRAM connected to the AMBA bus.

There is also an control register set that programmed from the CR_X32 bus.

SRAM appears as an AMBA as a slave device. The SRAM is zero wait state at all times.

14.2 Block diagram.

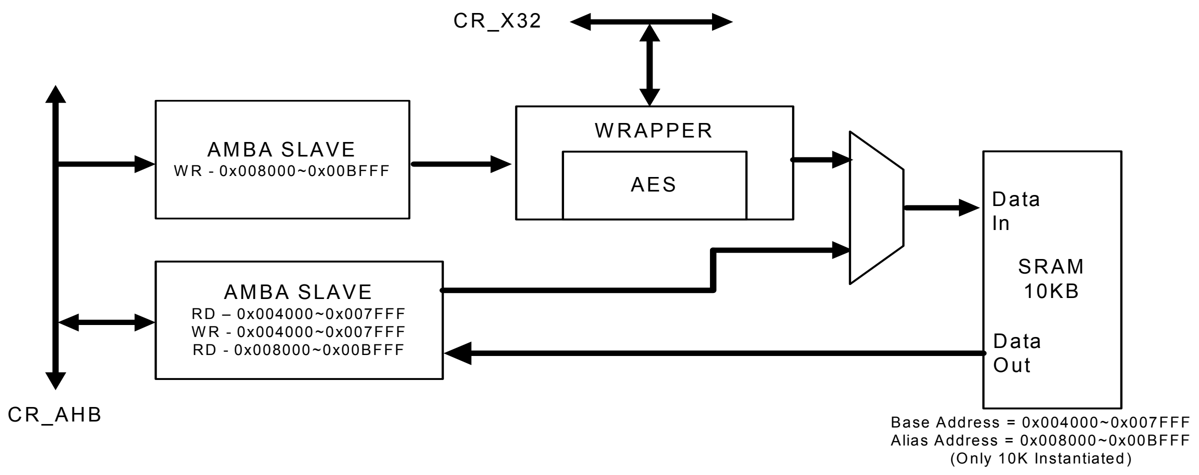


Figure 14.1 AES - SRAM Block Diagram

There is 16 Kbyte of SRAM space that resides in the Card Reader AMBA (CR_AHB) memory space. The 16 KByte SRAM space appears as an AMBA slave device. The address of the SRAM is aliased twice. The SRAM appears at address 0x004000 through 0x007FFF and address 0x008000 through 0x004BFFF. When the SRAM is read or written in the first address range, the AES block is bypassed. When the SRAM is read at the second range, the data is read bypassed. When the SRAM is written in the second range, the data passes through the AES block and is modified. Whether it is encrypted or decrypted, depends on how the AES block is programmed at that time.

Table 14.1 SRAM Decode Summary

CR_AHB ADDRESS	PROCESSOR ADDRESS	READ/WRITE	MODE	RAM OFFSET
0x004000 ~ 0x007FFF	CR_AHB+4000 ~ CR_AHB+7FFF	Both	Pass through	0x0000 ~ 0x3FFF
0x008000 ~ 0x00BFFF	CR_AHB+8000 ~ CR_AHB+BFFF	Read	Pass through	0x0000 ~ 0x3FFF
0x008000 ~ 0x00BFFF	CR_AHB+8000 ~ CR_AHB+BFFF	Write	Encrypt/Decrypt	0x0000 ~ 0x3FFF

When accessing the AES block, only aligned 32 bit accesses are allowed. Any other kind of access will lead to UNALIGNED_ACCESS interrupt. The AES interface hardware is not required to do any more than that.

Direct accesses to the SRAM can be 8, 16 or 32 bit with the normal AMBA restrictions.

14.3 SRAM Access when AES disabled.

When the AES is disabled, the AES block is in bypassed mode. All accesses are passed directly passed to the SRAM.

14.3.1 Key Descriptor

The wrapper in the AES encryption block has space for 7 keys. Only 2 keys are instantiated for SEC2410/SEC4410. KEY_NUM = 1, and KEY_NUM = 2. Each entry in the Key table has the encryption and decryption keys.

All the information in the Key Descriptor is exclusively used in Automatic mode. In Manual mode, firmware must load all the AES IP registers manually.

If KEY_NUM=0, then AES is set to bypass mode.

Table 14.2 Encrypt/Decrypt Descriptor Entry

OFFSET	NAME	R/W	DESCRIPTION
0x00	ENC_AES_KEY_7_0	R/W	Bits 63:0 of the AES key. Used for AES128, AES192 and AES256 Encyption
0x08	ENC_AES_KEY_15_8	R/W	Bits 127:64 of the AES key. Used for AES128, AES192 and AES256 Encryption
0x10	ENC_AES_KEY_23_16	R/W	Bits 191:128 of the AES key. Used for AES192 and AES256 Encryption
0x18	ENC_AES_KEY_31_24	R/W	Bits 255:192 of the AES key. Used for AES256 Encryption
0x20	DEC_AES_KEY_7_0	R/W	Bits 63:0 of the AES key. Used for AES128, AES192 and AES256 Decryption
0x28	DEC_AES_KEY_15_8	R/W	Bits 127:64 of the AES key. Used for AES128, AES192 and AES256 Decryption
0x30	DEC_AES_KEY_23_16	R/W	Bits 191:128 of the AES key. Used for AES192 and AES256 Decryption
0x38	DEC_AES_KEY_31_24	R/W	Bits 255:192 of the AES key. Used for AES256 Decryption.
0x40	AES_IV_7_0	R/W	Bits 63:0 of Initial Vector
0x48	AES_IV_15_8	R/W	Bits 127:64 of Initial Vector
0x50	AES_CNTR	R/W	Initial counter for the counter mode operation.
0x54	AES_NONCE	R/W	Initial Nonce value
0x58	AES_KEY_CTL	R/W	32 bit Key control (See AES_KEY_CTL Entry)

Note: The entries in this table are little endian. For example ENC_AES_KEY_7_0 bits 7~0 are at address 0, bits 15~8 are at address 1, 23~16 are at location 2 etc. Byte accesses should be done accordingly.

The Key tuple that will be used for a particular transfer will be determined by the KEY_NUM parameter supplied by the FMDU.

The Decrypt key must be generated in Software based on the Encrypt key. Alternatively, the firmware can use the hardware to generate the key.

Table 14.3 Key Control Descriptor Entry

AES_KEY_CTL RESET=0X00000000)			**AES KEY CONTROL ENTRY**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:16	Reserved	R	Always read ‘0’
15:13	AES_MODE	R/W	This register indicates the mode of operation. 3’b000: Indicates ECB mode of operation. 3’b001: Indicates CBC mode of operation. 3’b010: Indicates Counter mode algorithm. 3’b011: Indicates CCM (not supported). 3’b100: Indicates GCM/GHASH (Not supported).
12:10	BLOCK_SIZE	R/W	These bits are set by the Processor to indicate the block size of the of the transfer. 000 – 256 Bytes 001 – 512 Bytes 010 – 1024 Bytes 011 – 2048 Bytes 100 – 4096 Bytes 111 - Unlimited size All others reserved for future use.
9:6	Reserved	R	Always read ‘0’
5:4	KEYSIZE	R/W	Indicates keysize to be used 01b indicates AES128 10b indicates AES192 11b indicates AES256.
3:1	Reserved	R	Always read ‘0’
0	NO_IV_CTR_LOAD	R/W	‘1’ – Disable the auto loading of IV, CTR initial value, and Nonce at beginning of an AES operation. Firmware must manually load them directly into AES IP’s register in advance of the operation. ‘0’ – Enable the auto loading of IV, CTR initial value, and Nonce at beginning of an AES operation Note: This only applies to the initial loading of the values. Subsequent loading is controlled by the BLOCK_SIZE. For large file transfers, the size should be set to 111b.

14.4 Key Table

The key Table is the table that holds the Encrypt/Decrypt keys.

Table 14.4 Encrypt/Decrypt Key Descriptor Table

KEY_NUM	ADDRESS	DESCRIPTOR
0	NA	Plain Text Only
1	CR_X32+ 0x3400	KEY Descriptor 1
2	+0x80	KEY Descriptor 2
3	+0x80	KEY Descriptor 3
4	+0x80	KEY Descriptor 4
5	+0x80	KEY Descriptor 5
6	+0x80	KEY Descriptor 6
7	+0x80	KEY Descriptor 6

The Encrypt/Decrypt table must not be modified when the system is running. The conflict with FMDU access with processor access will result in errors.

14.5 Operation

All encryption is done when plain text is read from a source and is written through the AES block into the SRAM as cipher text.

All decryption is done when cipher text is read from a source and is written through the AES block into the SRAM as plain text.

No modification is done to data when it is read from the SRAM, whether it be plain text or cipher text.

14.5.1 AES data and address alignment rules:

All accesses that pass through the AES block must be DWORD aligned. Starting address for the block of data must be 4 byte aligned. Writes that are not DWORD aligned will result in improper operation and cause UNALIGNED_ACCESS exceptions. All data that need to be encrypted or decryped must be in blocks of 16 bytes. For data that is not a multiple of 16 bytes the firmware must pad the data to be a multiple of 16 bytes. For example 10 bytes must be padded to 16 bytes. 40 bytes must be padded to 48 bytes. Violating this rule causes an UNDERRUN interrupt when AES operation terminates.

If 10 bytes are to be encyrpted, the block must be padded to 16 bytes. It is important to remember that the entire 16 bytes of the cipher text must be used to recover the plain text. The recovered text will contain the padded data.

Data is encrypted and decrypted in the order that it is received. For every data word receveived, the address must increment correspondingly. If CBC or CTR mode is used, the full transfer must be done in sequence. With ECB, it is possible to change the order in which 16byte blocks are encrypted, but must be done with great care.

Violating size rules will cause an UNDERRUN interrupt at termination of an AES operation. The padding data will become a part of a 16-byte cipher text on encryption. This means that the entire 16-byte of cipher text will be needed to retrieve plain text on decryption. This is applied to ECB and CBC mode only and not applied to CTR mode. Since plain/cipher text is just exclusive-ORed with output of cipher/inverse-cipher function in CTR mode, there is no relationship between each byte in CTR mode.

14.5.2 Automatic Mode operation:

When operating in Automatic mode, the firmware must load all the wrapper registers. This include:

1. Encrypt/Decrypt Descripter table: (For each entry).
   - Encrypt Key: Load the 128/192/256 key into the AES_ENCRYPT_KEY_3_0 through AES_ENCRYPT_KEY_31_28 registers. These registers must be loaded for both Encrypt and Decrypt operation. These registers must be loaded for all modes.
   - Decrypt key: Load the last round expanded key into AES_DECRYPT_KEY_3_0 through AES_DECRYPT_KEY_31_28 registers. These registers must be loaded for both Encrypt and Decrypt operation. Regardless of the mode.
   - Initial Vector: Load the Initial Vector into AES_IV_3_0 and AES_IV_7_4
   - Counter and NONCE if required for the mode.
   - AES mode, BLOCK_SIZE, and KEYSIZE, the AES_KEY_CTL entry.
2. Program the FMDU LUN Descriptor as required.
3. When a CBW arrives, the FMDU supplies the AES block with the following tuple:
   - KEY_NUM: The number of the key to be used for encryption or decryption. The KEY_NUM is used to index into the descripter table.
   - REVERSE_MODE: Normally this bit is zero. When zero, it is assumed that writes to the media are encrypted and reads are decrypted. If the bit is set, then reads from the media are encrypted, and writes are encrypted.
   - READ/WRITE_N: If this bit is one, then the direction of data flow is from the media to the host. If it is a zero, then the data is from the host to the media. This is equivalent to bit 7 of the bmCBWFlags register in the CBW.
   - AES_CTL_WR: Every time a new CBW comes in the FMDU provides a pulse to the AES block with KEY_NUM value from selected LUN's descriptor table to mark the new transfer.
   - Every time FMDU sends a CSW out, it provides a pulse with KEY_NUM = "0" to mark the end of transfer.
4. The hardware does an exclusive OR of the READ/WRITE_N and REVERSE_MODE bits to generatethe ENCRYPT/DECRYPT_N signal required by the AES hardware to operate.
5. The AES wrapper state machine take the registers that have been programmed, and writes them into the AES IP.

14.5.3 Manual Mode operation:

When operating in manual mode, the firmware must load all the AES IP registers manually. This done using the CSR registers. The registers that must be programmed include:

1. The KEYSIZE field in the AES_CTL register must be programmed before loading the AES_KEY registers.
2. Encrypt Key: Load the 128/192/256 key into the AES_KEY_7_0 through AES_KEY_31_24 registers.
3. Initial Value: Load the initial value into AES_IV_7_0 and AES_IV_15_8
4. Counter and NONCE: Load Counter and NOUNCE if required for the mode.
5. Program the AES_CTL register.
6. Write data through the AES block using manual reads and writes, or DMA reads and writes.

14.6 Register Map

All registers in the AES control block reside on the CR_X32 bus.

14.6.1 AES Wrapper registers

Wrapper registers.

Table 14.5 AES Control Register

AES_CTL (CR_X32 + 0X3000 – RESET=0X00000000)			**AES CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:16	Reserved	R	Always read ‘0’
15:13	AES_MODE	R/W	In manual mode operation, this register indicates the mode of operation. 3’b000: Indicates ECB mode of operation. 3’b001: Indicates CBC mode of operation. 3’b010: Indicates Counter (CTR)mode algorithm. 3’b011: Indicates CCM (not supported). 3’b100: Indicates GCM/GHASH (Not supported). In Automatic mode operation, these bits have no effect. Instead, the AES_MODE of AES_KEY_CTL register in the key Descriptor table entry is used.
12:10	BLOCK_SIZE	R/W	In manual mode operation, these bits are set by the Processor to indicate the block size of the of the transfer. 000 – 256 Bytes 001 – 512 Bytes 010 – 1024 Bytes 011 – 2048 Bytes 100 – 4096 Bytes 111 - Unlimited size All others reserved for future use. In Automatic mode operation, these bits have no effect. In Automatic mode, the BLOCK_SIZE of AES_KEY_CTL register in the key Descriptor table entry is used.
9:7	Reserved	R	Always read ‘0’
6	FRAME_VALID	R/W	When operating under Firmware control AUTO_MODE = 0 (manual mode), this bit must be set and cleared by the firmware. The rising edge of this signal initializes the transfer. The falling edge denotes the end of the transfer. The signal must be high the whole time that data is being transferred. This signal must frame every transaction.
5:4	KEYSIZE	R/W	In manual mode operation, these bits are set by the processor to indicate the keysize to be used 01b indicates AES128 10b indicates AES192 11b indicates AES256. In Automatic mode operation, these bits have no effect. In Automatic mode, the KEYSIZE of AES_KEY_CTL register in the key Descriptor table entry is used.

AES_CTL (CR_X32 + 0X3000 – RESET=0X00000000)			**AES CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
3	AUTO_MODE	R/W	‘1’ - indicates Automatic mode. FMDU supplies the KEY_NUM, READ/WRITE_N, REVERSE_MODE signals. The value of AESCMD_CIPHEREN is automatically controlled by the AES Wrapper hardware. ‘0’ - indicates Manual mode. The firmware supplies the BLOCK_SIZE, ENCRYPT_DECRYPT and AESCMD_CIPHEREN (from this register)
2	ENCRYPT_DECRYPT	R/W	‘1’ - indicates Encrypt ‘0’ - indicates Decrypt.
1	AESCMD_CIPHEREN	R/W	‘1’ indicates that data needs to be encrypted or decrypted. ‘0’ indicates data is not modified. To load Key data into the AES core, this bit must be ‘0’
0	AES_ENABLE	R/W	If this bit is not set, the AES block is completely bypassed. SRAM access is direct. This has effect in both Automatic and Manual mode operation.

Table 14.6 AES Interrupt Status Register

AES_STAT (CR_X32 + 0X3004 – RESET=0X00000000)			**AES CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:6	Reserved	R	Always read ‘0’
5	FRAME_VALID_AUTO	R/W	This bit indicates the state or FRAME_VALID during automatic FMDU operation. The bit is set when the FMDU starts an AES operation, and cleared when the operation is complete. This bit can also be used to abort the operation. Writing a "1" will abort the current AES operation and clear this bit. Writing ‘0’ has no effect.
4	AES_IN_USE	R	This bit indicates that there is data in flight in the AES engine, or the FIFOs in the wrapper logic around the AES block. When using encryption or decrypion in manual mode, this bit must be checked before reading back processed data.
3	AES_RESIDUAL	R/W	This bit is set if AES operation is terminated with data left to be processed. Write ‘1’ to clear this bit. Write ‘0’ has no effect.
2	FIFO_OVERFLOW	R/W	This bit is set if there is a FIFO overflow in the AES IN FIFO. Write ‘1’ to clear this bit. Write ‘0’ has no effect.
1	UNALIGNED_ACCESS	R/W	This bit is set if there is an attempt to write anything other than an aligned 32 bit access to the AES address. Write ‘1’ to clear this bit. Write ‘0’ has no effect.

AES_STAT (CR_X32 + 0X3004 – RESET=0X00000000)			**AES CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
0	UNDER_RUN	R/W	This bit indicates that previous transfer did not end on an 128 bit boundary. It is detected when a CBW_NEW comes in and the AES block is not at an 128 bit boundary. Write ‘1’ to clear this bit. Write ‘0’ has no effect.

These bits go to the interrupt controller. If the interrupts need to be masked, they are done in the controller.

Table 14.7 AES CSR Control register

AES_CSR_CTL (0X3010 - RESET=0X00)			**AES CSR CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[7:2]	Reserved	R	Always read ‘0’
1	CSR_RD_STRB	R/W	Trigger an Endpoint Read to the CSR registers. Bit self clears when operation is complete. ''0' = No Action '1' = A Read operation is triggered
0	CSR_WR_STRB	R/W	Trigger an Endpoint Write to the CSR registers. Bit self clears when operation is complete. ''0' = No Action '1' = A write operation is triggered

Table 14.8 AES CSR Data Register

AES_CSR_DATA_LO (0X3014~0X3017 - RESET=0X00000000)			**AES CSR DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[31:0]	CSR_DATA[31:0]	R/W	AES CSR Data Register D31:0. This value is written into the CSR register pointed to by the AES_CSR_ADDR register.

Table 14.9 AES CSR Data High Register

AES_CSR_DATA_HI (0X3018~0X301B - RESET=0X00000000)			**AES CSR DATA HIGH REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[31:0]	CSR_DATA[63:32]	R/W	AES CSR Data Register D63:32. This value is written into the CSR register pointed to by the AES_CSR_ADDR register.

Table 14.10 AES CSR Address Register

AES_CSR-ADDR (0X301C - RESET=0X00)			**AES CSR ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	R	Always read ‘0’
[6:3]	CSR_ADDR[6:3]	R/W	The address of the AES CSR register to be programmed is stored in this register.
2:0	Reserved	R	Always read ‘0’

14.6.2 AES IP registers

These registers can only be accessed using the CSR registers.

14.6.2.1 Accessing AES IP register

The registers in the AES IP block can only be accessed 64 bits at time. To write to a particular register, the following steps are required:

1. Program the address of the register into AES_CSR_ADDR.
2. Program the 64 bit data value into AES_CSR_DATA_HIGH and AES_CSR_DATA_LOW
3. Write a ‘1’ into the AES_CSR_CTL register CSR_WR_STRB bit.

To read a particular registers, the following steps are required:

1. Program the address of the register into AES_CSR_ADDR.
2. Write a ‘1’ into the AES_CSR_CTL register CSR_RD_STRB bit.
3. Read the value of the register using the AES_CSR_DATA_HIGH and AES_CSR_DATA_LOW registers.

Table 14.11 AES Contoller IP Register

CSR Offset	NAME	R/W	DESCRIPTION	Source	PAGE
AES IP Registers					
0x000	AES_KEY_7_0	R/W	Bits 63:0 of the AES key. Used for AES128, AES192 and AES256.	AES IP	
0x008	AES_KEY_15_8	R/W	Bits 127:64 of the AES key. Used for AES128, AES192 and AES256.	AES IP	
0x010	AES_KEY_23_16	R/W	Bits 191:128 of the AES key. Used for AES192 and AES256.	AES IP	
0x018	AES_KEY_31_24	R/W	Bits 255:192 of the AES key. Used for AES256.	AES IP	
0x020	AES_IV_7_0	R/W	Bits 63:0 of Initial Vector	AES IP	
0x028	AES_IV_15_8	R/W	Bits 127:64 of Initial Vector	AES IP	
0x030	AES_CTR_ADDR	R/W	Initial counter for the counter mode operation.	AES IP	
0x038	AES_CMD	R/W	AES Command	AES IP	
0x040	AES_AAD_15_8	R/W	Do not use (Additional Authentication Data)	AES IP	
0x048	AES_AAD_7_0	R/W	Do not use (Additional Authentication Data)	AES IP	
0x050	COUNTA_COUNTC	R/W	Do not use (Length of AAD)	AES IP	

Table 14.12 AES_KE_7_0 Register

AES_KEY_7_0 (*TALLICA IP*) OFFSET = 0X000			**AES_KEY_7_0**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_KEY_7_0	R/W	Bits 63:0 of the AES key. Used for AES128, AES192 and AES256.

Table 14.13 AES_KE_15_8 Register

AES_KEY_15_8 (*TALLICA IP*) OFFSET = 0X008			**AES_KEY_15_8**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_KEY_15_8	R/W	Bits 127:64 of the AES key. Used for AES128, AES192 and AES256.

Table 14.14 AES_KEY_23_16 Register

AES_KEY_23_16 (*TALLICA IP*) OFFSET = 0X010			**AES_KEY_23_16**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_KEY_23_16	R/W	Bits 191:128 of the AES key. Used for AES192 and AES256.

Table 14.15 AES_KEY_31_24 Register

AES_KEY_31_24 (*TALLICA IP*) OFFSET = 0X018			**AES_KEY_31_24**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_KEY_31_24	R/W	Bits 255:192 of the AES key. Used for AES256.

Table 14.16 AES_IV_7_0 Register

AES_1V_7_0 (*TALLICA IP*) OFFSET = 0X020			**AES_IV_7_0**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_IV_7_0	R/W	Bits 63:0 of Initial Vector

Table 14.17 AES_IV_15_8 Register

AES_1V_15_8 (*TALLICA IP*) OFFSET = 0X028			**AES_IV_15_8**
BIT	NAME	R/W	DESCRIPTION
63:0	AES_IV_15_8	R/W	Bits 127:64 of Initial Vector

Table 14.18 AES_CTR_ADDR Register

AES_CTR_ADDR (*TALLICA IP*) OFFSET = 0X030			**AES_CTR_ADDR**
BIT	**NAME**	**R/W**	**DESCRIPTION**
63:0	AES_CTR_ADDR	R/W	Initial counter for the counter mode operation.

Table 14.19 AES Command

AES COMMAND (*TALLICA IP*) OFFSET = 0X038			**AES COMMAND**
BIT	NAME	R/W	DESCRIPTION
31:16	AES_BE_R	R/W	These bits indicate the byte enables for the last 16-byte block of AAD data (aes_aad_eof_r active) in register mode. These bits also indicate the valid byte for input data for the last 16-byte block.
15:13	AES_MODE_R	R/W	This register indicates the mode of operation. 3'b000: = ECB mode of operation. 3'b001: = CBC mode of operation. 3'b010: = Counter mode algorithm. 3'b011: = CCM Not-supported 3'b100: = GCM/GHASH Not supported
12	AES_AAD_EOF_R	R/W	When high indicates that this is the last block of the AAD data.
11	AES_EOF_R	R/W	Indicates that the next 8 byte data written to the input interface is the last block of the frame/packet.
10	Reserved	R	Reserved.
9	ORDY	R	This read only bit is high when the output pipeline stage of the AES core is ready to accept data (16 B).
8	IRDY	R	This read only bit is high when the input pipeline stage of the AES core is ready to accept data (16 B).
7	Reserved	R	Reserved
6	KEY_EXPAND_MODE	R/W	This bit if high, it indicates that the block must perform the initial key expansion to compute the decrypt key, and store it in the key registers, before decryption starts. This bit resets itself after the Key expansion is done. This bit must be set for decryption, and an encrypt key is written for decryption. The core will convert will perform key expansion before first block decryption.
5:4	KEYSIZE_R	R/W	‘01’ = AES128, ‘10’ = AES192 ‘11’ = AES256.
3	AES_INSERT_IV_R	R/W	This register is equivalent in functionality to the aes_insert_iv_r interface signal. This bit when set will indicate to the core to insert 16 bytes of IV data (from the output stage cipher registers) after the last 16 byte data that was written to the input.
2	ENCRYPT_DECRYPT	R/W	‘0’ = Decrypt ‘1’ = Encrypt
1	AESCMD_CIPHEREN	R/W	This bit if zero, allows data to pass through.
0	REG_CNTL	R/W	This bit powers up to zero. If zero, the external signals aes_cipheren, aes_en_decrypt_l, aes_mode_r[2:0], aes_keysize_r[[1:0] control the operation of the block. This bit if one, enables bits in AES_CMD to control the operation of the block.

Chapter 15 OTP Interface

15.1 General Description

SEC2410/SEC4410 has an 8 Kbit OTP (IP from Sidense) which is organized as 1K X 8 bit.

The configuration registers are accessable through the System AHB bus.

15.2 Basic Read Write Access to OTP Memory

15.2.1 Reading from Main OTP memory

The READ command reads data from the OTP's internal DATA register. The data will then be available in the OTP_RD_DATA register. Below is the sequence of a typical READ. The hardware will automatically drive MACRO_SEL signal during the access.

a. Write to the OTP_ADDR register, bits 0-9 and leaving bit 10 set to '0'.

b. Write to the OTP_CMD register, setting bit 1 (READ) to '1'. This will initiate the READ command. This bit will be cleared by hardware.

c. Either poll the OTP_STATUS register until the BUSY bit is cleared, or wait till the interrupt triggers.

d. Read the OTP_RD_DATA register.

Figure 15.1 OTP Read Timing

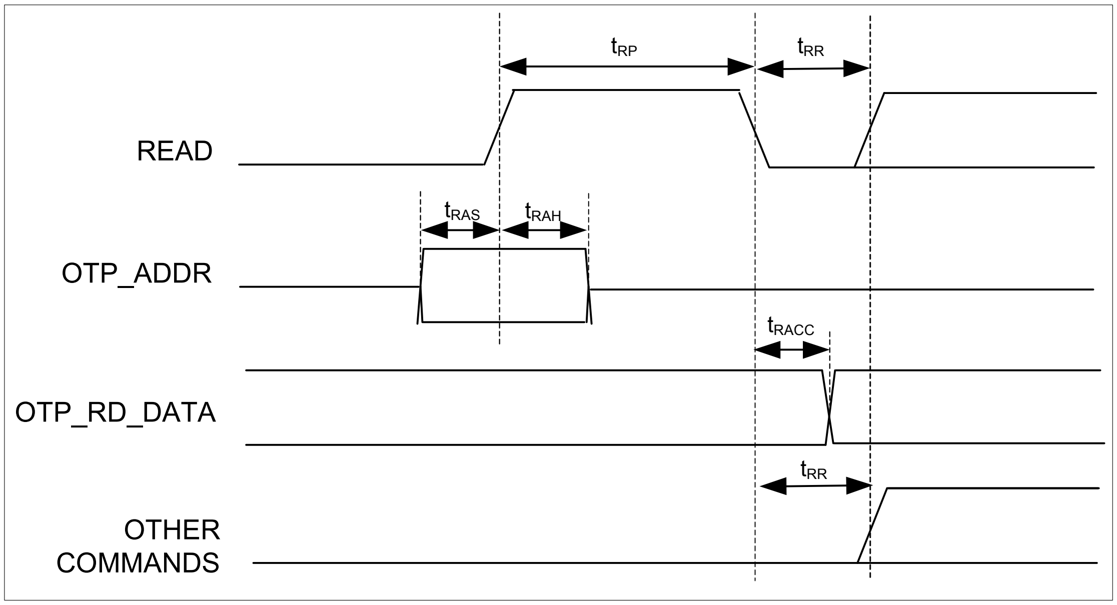


Table 15.1 OTP Read Timing

NAME	IP SPEC MIN	DESIGN MIN	DESRIPTION
t_{RP}	40ns	50ns	Command Pulse
t_{RR}	10ns	16.7ns	Command Recovery time
t_{RAS}	5ns	16.7ns	Address Setup Time
t_{RAH}	1ns	16.7ns	Address Hold Time
t_{RACC}	5ns	16.7ns	Read Access time

15.2.2 Programming Main OTP memory

All even address in the memory have unprogrammed default state of ‘0’. All odd addresses in the memory have unprogrammed default state of ‘1’. Programming is done one bit at a time. The PGM command programs the content of the OTP's Data register to the memory location specified by the OTP's address pins A[10:0]. In order to program the OTP, the charge pump has to be enabled MRA[0] -> VPP CHARGE PUMP Enable and disabled after programming. The user should verify if all the bits have been successfully programmed and repeat this process. Below is the sequence of a typical PGM.

a. Enable the charge pump, and wait 20 uSec for the charge pump to warm up. To do this set MRA[0]=1, MRB[6:4]=111
b. Since there is only one bank in the system, the hardware automatically sets ALL_BANKS to '1'. Do not modify this bit.
c. Set the default programming voltage. MRA[15,13,12]=000, MRB[3:0]=0010.
d. Write the user data to be programmed to the OTP_WR_DATA register. Hardware will automatically inverse the bits as required, and run as many cycles as required. For even addresses, the number of cycles is equal to the number of ones. For odd addresses the number of cycles is equal to the number of zeros.
e. Write to the OTP_ADDR register with the location to be written. For differential mode, hardware will automatically use that address and that address plus one.
f. Program the OTP_PPW register with the programming pulse width. For the first attempt use the regular pulse width (t_{PP1}).
g. Firmware writes to the OTP_CMD register, setting bit 2 (PROGRAM) to '1'. This will initiate the PGM command. This bit will be cleared by hardware.
h. Poll the OTP_STATUS register. Wait till BUSY is clear before executing next command. Alternatively, wait for the interrupt to trigger. If using interrupts, the interrupt must be cleared by writing a ‘1’ to the bit.
i. Go back to step (e) and keep programming till the array is done.
j. To verify the data was programmed correctly, with enough margin, read back the data with read voltage VREF1. To do this, MRA[15,13:12]=000, MRB[3:0]=1010
k. Read back OTP_MRA_RD_DATA to verify everything was done correctly.
l. Write to the OTP_CMD register, setting bit 1 (READ) to '1'. This will initiate the READ command. This will transfer data at the OTP_ADDR location to the OTP_RD_DATA register. This bit will be cleared by hardware.
m. Read of the OTP_RD_DATA register. If the data matches the expected value the program was successful, move onto the next byte. Record all the bits that fail.
n. To reprogram the bits that failed, program the OTP_PPW register with the programming pulse width. For the failed bits use the longer pulse width (t_{PP2}).
o. Repeat the programming steps.

p. Repeat the verification steps as before, except with the read voltage set to VREF2. To do this, set MRA[15,13:12]=000, MRB[3:0]=0111.

q. Keep repeating the program and verify cycles till the part is programmed. If the number of cycles exceeds TBD, and the part has not fully programmed, then fail the part.

r. Turn off charge pump when done.

15.2.2.1 Example: Hardware programming Even address one bit at a time

For this example, the Address 0x1A4, so the unprogrammed byte is all zeros. The data is 0xC9 (0b11001001). It has four bits set (bits 0, 3, 6, and 7), so it will take four bit writes to program the ones.

Step 1: Program bit 0 (0000 0001) : Data to write to data register 1111 1110

Step 2: Program bit 3 (0000 1000) : Data to write to data register 1111 0111

Step 3: Program bit 6 (0100 0000) : Data to write to data register 1011 1111

Step 4: Program bit 7 (1000 0000) : Data to write to data register 0111 1111

15.2.2.2 Example: Hardware programming Odd address one bit at a time

For this example, the Address 0x1A5, so the unprogrammed byte is all ones. The data is 0xC9 (0b11001001). It has four bits cleared (bits 1, 2, 4, and 5), so it will take four bit writes to program the ones.

Step 1: Program bit 1 (1111 1101) : Data to write to data register 0000 0010

Step 2: Program bit 2 (1111 1011) : Data to write to data register 0000 0100

Step 3: Program bit 4 (1110 1111) : Data to write to data register 0001 0000

Step 4: Program bit 5 (1101 1111) : Data to write to data register 0010 0000

Figure 15.2 OTP Program Timing

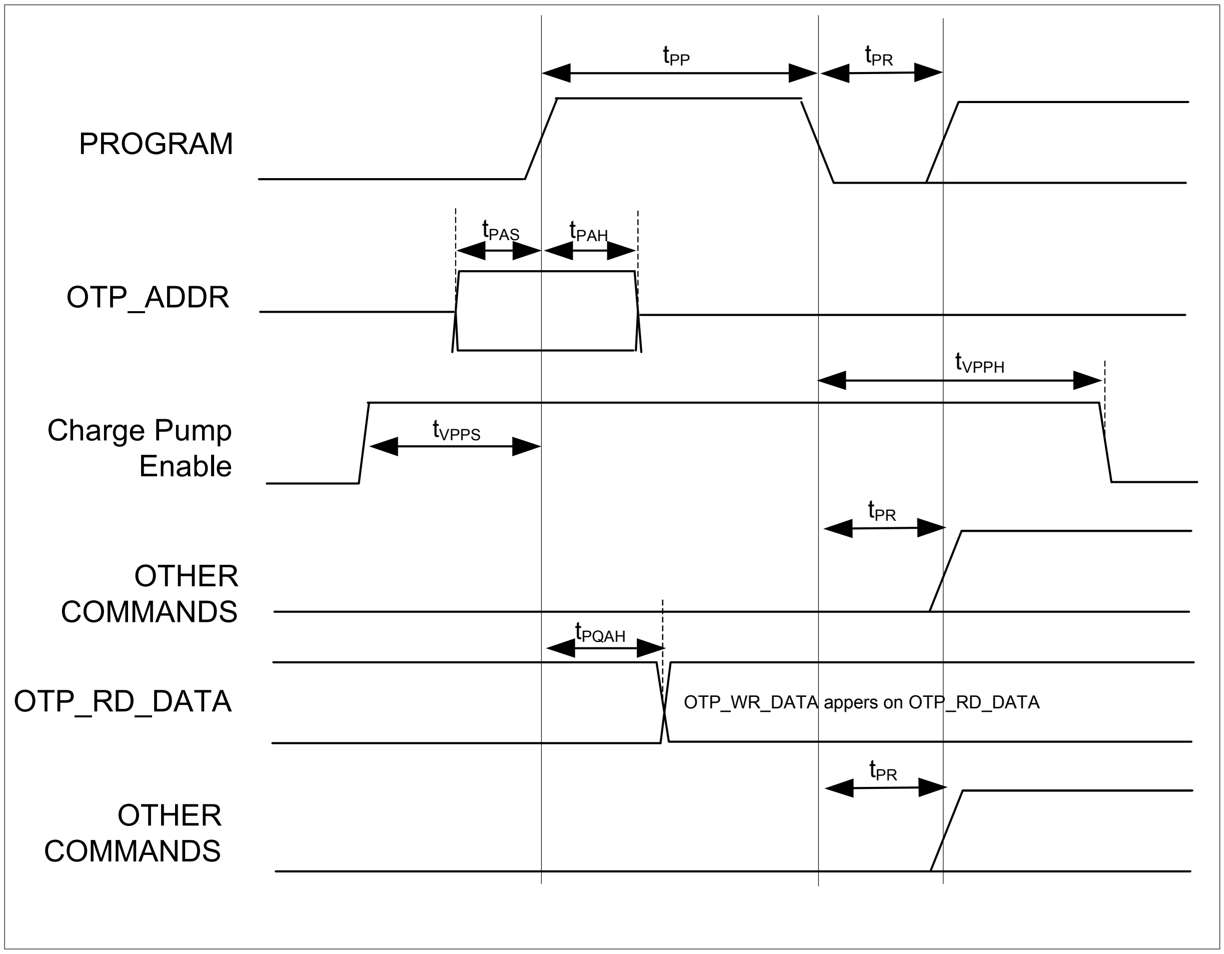


Table 15.2 OTP Program Timing

NAME	IP SPEC MIN	DESIGN MIN	DESRIPTION
t_{PP}	100us /1000us	200us/ 1000us	Program Pulse Width. Use lower value (t_{PP1}) for the first attempt, the second value (t_{PP2}) for subsequent attempts.
t_{PR}	1000ns	1200ns	Program Recovery time
t_{PAS}	2ns	16.7ns	Address Setup Time
t_{PAH}	5ns	16.7ns	Address Hold Time
t_{PQH}	10ns	16.7ns	Program to Data Register Hold Time
t_{VPPH}	1000mS	900mS	Charge Pump maximum idle time
t_{VPPS}	10uS	12uS	Charge Pump minimum warm up time

15.2.3 WRITE to Boot Row

In addition to the regular memory array, every sector includes a specific number of additional rows, called boot rows, for testing and memory bookkeeping purposes. The Boot Row block is addressed using OTP_ADDR register, bit 0-3 with bit 10 set to '1'. Only the even rows are programmable. The boot rows at address 0 and 2 in sector 0 is loaded into the Special Register and boot rows at address 0 and 2 in sector 1 is loaded into the Redundancy Register at power-up reset sequence or applying the RESET command.

a. Firmware writes to the OTP_ADDR register. Only bits 0-3 are valid and bit 10 has to be set to '1'.
b. Firmware writes the inverse of the data to be programmed to the OTP_WR_DATA register.
c. Firmware writes to the OTP_CMD register, setting bit 0 (WRITE) to '1'. This will initiate the WRITE command. This bit will be cleared by HW.
d. Firmware needs to poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

Boot rows must be programmed and verified following the same programming algorithm as the main memory. The even boot rows are also OTP.

15.2.4 Read from Boot Row

a. Firmware writes to the OTP_ADDR register. Only bits 0-3 are valid and bit 10 has to be set to '1'.
b. Firmware writes to the OTP_CMD register, setting bit 1 (READ) to '1'. This will initiate the READ command. This bit will be cleared by HW.
c. Firmware needs to poll the OTP_STATUS register until bit 0 (BUSY) becomes '0'
d. Firmware reads the OTP_RD_DATA register

The boot reads are the same as reads from main memory, with the exception that the MSB of the address is set. Boot rows are always read in single-ended mode, except during RESET load of OTP_RR_RD_DATA and OTP_SR_RD_DATA registers, which are done in redundant mode.

15.3 Other available OTP operations

15.3.1 Write to DATA Register

The WRITE command writes data on the D[7:0] pins to the OTP's Data register. The DATA register is then used to do the write into the nNV store. Below is the sequence of a typical WRITE.

a. Write to the OTP_CMD register, setting bit 19 (ALL_BANKS) to '1'. Do this one time and leave it set.
b. Write the inverse of the data to be programmed to the OTP_WR_DATA register.
c. Write to the OTP_CMD register, setting bit 0 (WRITE) to '1'. This will initiate the WRITE command. This will fire the state machine that will do the write. This bit will be cleared by Hardware.
d. Poll the OTP_STATUS register. Wait till the BUSY bit is clear before next command.

The data in the DATA register will not be written in the the NV store till a program cycle has been run.

15.3.2 Precharge Command

The PCH command compares the last read data with the contents of the input latch in the OTP_WR_DATA register and removes from the OTP_WR_DATA register all the bits that have been already programmed. The PCH sets the STATUS pin low if all the bits have been programmed. Below is the sequence of a typical PCH.

a. Completed a PGM and the BUSY bit is set to '0'.
b. Firmware writes to the OTP_CMD register, setting bit 1 (READ) to '1'. This will initiate the READ command. This bit will be cleared by HW.
c. Firmware writes to the OTP_CMD register, setting bit 3 (PRECHARGE) to '1'. This will initiate the PCH command. This bit will be cleared by HW.
d. Firmware needs to poll the OTP Status register until bit 0 (BUSY) becomes '0'.
e. Look at the STATUS bit (OTP Status register bit 2). If it's a '0', then PGM was successful.

15.3.3 Compare Command

The COMP command compares the last read data with the inverted contents of the DATA register and sets the STATUS pin high in case of a full match. Below is the sequence of a typical COMP.

a. Completed a PGM and the BUSY bit is set to '0'.
b. Firmware writes to the OTP_CMD register, setting bit 4 (COMPARE) to '1'. This will initiate the COMP command. This bit will be cleared by HW.
c. Firmware needs to poll the OTP Status register until bit 0 (BUSY) becomes '0'.
d. Look at the STATUS bit (OTP Status register bit 2). If it's a '1', then PGM was successful.

15.3.4 Write to Mode Register

The WRITE_MR command writes data on the D_MR[7:0] pins to the Mode Data register. Below is the sequence of a typical WRITE_MR.

a. Firmware writes data to the OTP_M_WR_DATA register.
b. Firmware writes to the OTP_CMD register, setting bit 8 (WRITE_MR) to '1'. This will initiate the WRITE_MR command. This bit will be cleared by HW.
c. Firmware needs to poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.
d. The contents of the Mode register are available from the OTP_M_RD_DATA register.

15.3.5 Accessing Auxiliary Mode Registers

In addition to the main Mode Register, the Sidense IP is equipped with Auxiliary Mode Registers (MRA and MRB) controlling internal voltage regulators and charge pumps.

These registers are accessed by programming the main Mode register, and then writing that value into the auxiliary registers using the AUX_UPDATE command and the MRA and MRB bits. The AUX_UPDATE command copies the content of the OTP_M_WR_DATA register to either the Auxiliary Mode A register (MRA bit = 1, MRB bit = 0) or Auxiliary Mode B register (MRA bit = 0, MRB bit =1).

a. Write the data to the OTP_M_WR_DATA register.
b. Write to the OTP_CMD register, setting bit 10 (AUX_UPDATE) to '1', and either bit 11 (MRA) or bit 12 (MRB) to a '1'. This will initiate the AUX_UPDATE command. These bits will be cleared by HW.
c. Poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.
d. The contents of the Mode Register A are available from the OTP_MRA_RD_DATA register, and the contents of the Mode Register B are available from the OTP_MRB_RD_DATA register.

15.3.6 Reseting Mode Registers

The RESET_MR command resets only the Mode register. Below is the sequence of a typical RESET_MR.

a. Write to the OTP_CMD register, setting bit 13 (RESET_MR) to '1'. This will initiate the RESET_MR command. This bit will be cleared by HW.

b. Poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

The RESET_M command resets the Mode register and Auxiliary Mode registers. Below is the sequence of a typical RESET_M.

a. Write to the OTP_CMD register, setting bit 14 (RESET_M) to '1'. This will initiate the RESET_M command. This bit will be cleared by HW.

b. Poll the OTP_STATUS register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

15.3.7 LOAD_QR Command

The LOAD_QR command copies the content of the DATA Register to the Redundancy register while MR[9] is set to a '1'. This is a way to test the redundancy scheme without burning OTP boot row. The timing of the LOAD_QR pulse is identical to the AUX_UPDATE and WRITE command pulses. Below is the sequence of a typical LOAD_QR.

a. Write a 0x200 to OTP_M_WR_DATA, to set bit 9 of Mode register.

b. Write to the OTP_CMD register, setting bit 8 (WRITE_MR) to '1'. This will initiate the WRITE_MR command. This bit will be cleared by HW.

c. Poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

d. Write to the OTP_CMD register, setting bit 20 (LOAD_QR) to '1'. This will initiate the LOAD_QR command. This bit will be cleared by Hardware.

e. Polll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

f. The contents of the Redundancy register are available from the OTP_RR_RD_DATA register.

15.3.8 QS_TEST Command

The QS_TEST command copies the content of the DATA Register to the Special register while MR[9] is set to a '0'. This is a way to test the redundancy scheme without burning OTP boot row. The timing of the LOAD_QR pulse is identical to the AUX_UPDATE and WRITE command pulses. Below is the sequence of a typical LOAD_QR.

a. Write a 0x0000 to OTP_M_WR_DATA, to clear bit 9 of Mode register.

b. Write to the OTP_CMD register, setting bit 8 (WRITE_MR) to '1'. This will initiate the WRITE_MR command. This bit will be cleared by HW.

c. Poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

d. Write to the OTP_CMD register, setting bit 25 (QS_TEST) to a '1' and bit 20 (LOAD_QR) to '1'. This will drive the QS_TEST pin to a '1' and initiate the LOAD_QR command. These bits will be cleared by HW

e. Polll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

f. The contents of the Special register are available from the OTP_SR_RD_DATA register.

15.4 Enabling and Disabling Charge Pump

The charge pump must be enabled before programming the OTP. The charge pump is enabled and disabled by bit 12 of Auxiliary Mode Register A. After enabling the charge pump the firmware must wait 20 uSec to allow the charge pump to warm up before using. When not programming, the charge pump should be turned off. There is a requirement that the charge pump must not be left idle for more than 1000 mSec.

To enable the charge pump:

a. Write the data to the 0x0001 OTP_M_WR_DATA register.

b. Write to the OTP_CMD register, setting bit 10 (AUX_UPDATE) to '1', and set bit 11 (MRA) to ‘1’ and bit 12 (MRB) set to a ‘0’. This will initiate the AUX_UPDATE command to Auxiliary Mode Register A. These bits will be cleared by HW.

c. Firmware needs to poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

To disable the charge pump:

a. Write the data to the 0x0000 OTP_M_WR_DATA register.

b. Write to the OTP_CMD register, setting bit 10 (AUX_UPDATE) to '1', and bit 11 (MRA) set to ‘1’ and bit 12 (MRB) set to a ‘0’. This will initiate the AUX_UPDATE command to Auxiliary Mode Register A. These bits will be cleared by HW.

c. Firmware needs to poll the OTP Status register. When bit 0 (BUSY) is a '0', then it's Ok to execute another command.

15.5 Locking the OTP Memory

The Special register is used to implement the OTP Lockdown. The Special Register is 8 bits wide and bit 7 is used as the lockout bit. The Special Register is loaded at power-on reset or nRESET. It is loaded with the bitwise OR of boot rows 0 and 2 of sector 0.

Once bit 7 is set, the the OTP memory is locked. The OTP memory is no longer accessible to read or write through the JTAG port. The OTP is still read accessible to the processor.

See section on writing to boot row to program the lockout bit.

15.6 OTP Interrupt

An OPT_READY interrupt is generated whenever BUSY is low.

Once bit 7 is set, the the OT

15.7 OTP Registers

All these registers reside on the system AHB bus at address offset TBD.

Table 15.3 OTP Address Register

OTP_ADDR (0X0000~0X0001- RESET=0X0000)			**OTP ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:11	Reserved	R	Always read ‘0’
10	BOOT_ADDR	R/W	'1' = Boot Block address; '0' = Main Block address This bit is directly connected to the Sidense IP input A[10]
9:0	OTP_ADDR	R/W	Address input. Program the address to be written or read from the NV store. These bits are directly connected to the Sidense IP input A[9:0]

Table 15.4 OTP Write Data Register

OTP_WR_DATA (0X0004- RESET=0X00)			**OTP WRITE DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	OTP_WR_DATA	R/W	Parallel data to be written to the DATA register. The DATA register is then transferred to the NVSTORE at the OTP_ADDR location with a program command These bits are directly connected to the Sidense IP inputs D[7:0]

Table 15.5 OTP Read Data Register

OTP_RD_DATA (0X0004- RESET=0X00)			**OTP WRITE DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	OTP_RD_DATA	R	Parallel data from DATA register. This is the data read from the NV store at the location pointed to by OTP_ADDR. To get fresh data, a READ operation must be done before reading this register. These bits are directly connected to the Sidense IP outputs Q[7:0]

Table 15.6 OTP Mode Write Register

OTP_M_WR_DATA (0X000C~0X000D- RESET=0X0000)			**OTP MODE WRITE DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	OTP_MODE_WR_DATA	R/W	Parallel data that will be written to the Mode register using a WRITE_MR command in the OTP_CMD register These bits are directly connected to the Sidense IP inputs D_MR[15:0]

Table 15.7 OTP Mode Read Register

OTP_M_RD_DATA (0X0010~0X0011- RESET=0X0000)			**OTP MODE READ DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	OTP_MODE_RD_DATA	R	Parallel data to be read back from Mode register These bits are directly connected to the Sidense IP outputs Q_MR[15:0]

Table 15.8 OTP Auxiliary Mode A Read Register

OTP_MRA_RD_DATA (0X0014~0X0015- RESET=0X0000)			**OTP MODE A READ DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	OTP_MRA_RD_DATA	R	Parallel data from Auxially Mode A register. Please refer to the Sidense documentation for bit definition. These bits are directly connected to the Sidense IP outputs Q_MRA[15:0]

Table 15.9 OTP MODE B Read Data Register

OTP_MRB_RD_DATA (0X0018~0X0019- RESET=0X0000)			**OTP MODE B READ DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	OTP_MRB_RD_DATA	R/W	Parallel data from Auxiliary Mode B register. Please refer to the Sidense documentation for bit definition. These bits are directly connected to the Sidense IP outputs Q_MRB[15:0]

Table 15.10 OTP Special Read Data Register

OTP_SR_RD_DATA (0X001C- RESET=0X00)			**OTP SPECIAL READ DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	OTP_SR_RD_DATA	R	Parallel data from special DATA register These bits are directly connected to the Sidense IP outputs Q_SR[7:0]

Table 15.11 OTP Redundancy Read Data Register

OTP_RR_RD_DATA (0X0020- RESET=0X00)			**OTP REDUNDANCY READ DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	OTP_RR_RD_DATA	R	Parallel data from redundancy DATA register. These bits are directly connected to the Sidense IP outputs Q_RR[7:0]

Table 15.12 OTP Command Register

OTP_CMD (0X0024~0X0027- RESET=0X00080000)			**OTP COMMAND REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:30	Reserved	R	Always read ‘0’
29	OTP_PGM_BIT	R/W	0 - Byte writes 1 - Bit writes
28:27	OTP_MODES	R/W	00 - single-ended 01 - differential 10 - redundant 11 - differential-redundant
26	RED_EN	R/W	Redundancy enable
25	QS_TEST	R/W	Load Special Register from DATA register
24	PWR_DWN	R/W	Disable all internal voltage generators and references
23:21	Reserved	R	Always read ‘0’
20	LOAD_QR	R/W	Load Redundancy Register from DATA register
19	ALL_BANKS	R/W	Select all banks. Should always be set to ‘1’ for SEC2410/SEC4410 as there is only one bank.
18:16	AB[2:0]	R/W	Bank Address (SEC2410/SEC4410 only has one bank)
15	RESET	R/W	RESET command
14	RESET_M	R/W	Reset Mode and Auxiliary Mode registers

13	RESET_MR	R/W	Reset Mode register only
12	MRB	R/W	Select Mode Register B
11	MRA	R/W	Select Mode Register A
10	AUX_UPDATE	R/W	Write to Auxliary Mode Register A or B based on MRA and MRB
9	WRITE_AUX	R/W	WRITE_MR followed by AUX_UPDATE
8	WRITE_MR	R/W	WRITE command to Mode Register
7:5	Reserved	R	Always read ‘0’
4	COMPARE	R/W	COMP Command
3	PRECHARGE	R/W	PCH Command
2	PROGRAM	R/W	PGM Command
1	READ	R/W	READ command
0	WRITE	R/W	WRITE command

Table 15.13 OTP Status Register

OTP_STATUS (0X0028- RESET=0X02)			**OTP STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	OTP_READY_INTR	R/W	This bit is set whenever BUSY transisitions from ‘1’ to ‘0’. A one in this bit causes a OTP_READY interrupt in the interrupt controller. Write a ‘1’ to clear this bit. Writes of ‘0’ have no effect.
6	Reserved	R	Always read ‘0’
5	ILLEGAL_ADDRESS	R/W	This bit is set whenever an illegal access occurs in the OTP address space. With this bit, an AHB bus error will be generated. Write a ‘1’ to clear this bit. Writes of ‘0’ have no effect.
4	OTP_LOCKDOWN	R	PGM to OTP has been disabled
3	VPP_MON	R	Charge pump in on
2	STATUS	R	Status of different processed operation.
1	PWR_UP	R	The OTP is correctly powered up
0	BUSY	R	OTP Read, Write or Programming is in progress

15.7.1 OTP timer registers

There are five timers in this module to control the timing for READ, PGM (high and low pulses), PCH and COMP commands. The countdown values are programmable via its respective register.

Table 15.14 OTP Programminng Pulse Width Register

OTP_PPW (0X002C~0X002F- RESET=0X00000000)			**OTP PROGRAMMING PULSE WIDTH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TPP	R/W	Programming Pulse Width in terms of system clocks. The minimum value to be used is 10,000 (~167uS) for first attempt. For subsequent attempts use 100,000.

Table 15.15 OTP Program Recovery Width Register

OTP_PRW (0X0030~0X0033- RESET=0X00000000)			**OTP PROGRAM RECOVERY WIDTH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TPR	R/W	Program Recovery Width in terms of system clock. The minimum value to be used is 100 (~1.6uS).

Table 15.16 OTP Read Pulse Width Register

OTP_RPW (0X0034~0X0037- RESET=0X00000000)			**OTP READ PULSE WIDTH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TRP	R/W	Read pulse width in terms of system clock. The minimum value to be used is 5 (~80nS).

Table 15.17 OTP Precharge Pulse Width Register

OTP_PCPW (0X0038~0X003B- RESET=0X00000000)			**OTP PROGRAMMING RECOVERY WIDTH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TPCP	R/W	Precharge pulse Width in terms of system clock. The minimum value to be used is 5 (~80nS).

Table 15.18 OTP Compare Pulse Width Register

OTP_CPW (0X003C~0X003F- RESET=0X00000000)			**OTP COMPARE PULSE WIDTH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TCP	R/W	Compare Pulse Width in terms of system clock. The minimum value to be used is 5 (~80nS).

Chapter 16 USB Controller Description

16.1 Overview

The USB block in the SEC2410/SEC4410 consists of three major parts. The first is the USB PHY, the second is the UDC20 USB interface. The final portion is the transport interface which is located between the UDC20 and the AMBA bus.

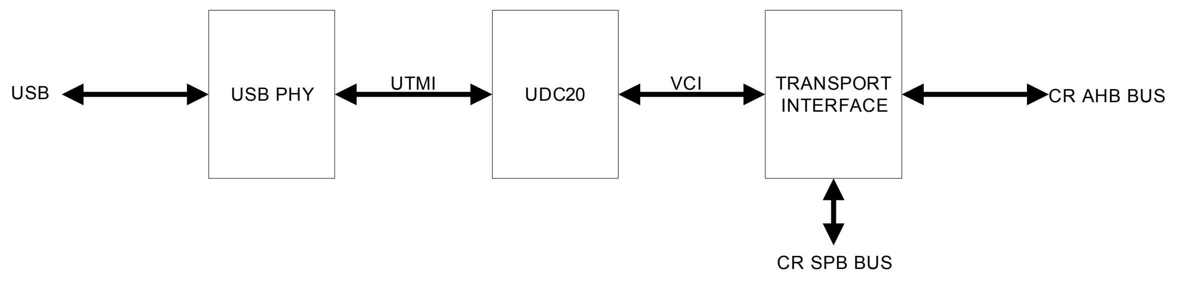


Figure 16.1 USB Interface

The USB PHY has the USB interface on one end, and a UTMI interface on the other. The parallel-to-serial / serial-to-parallel conversion, bit stuffing, and NRZI coding / decoding are handled in the PHY block. The PHY is capable of operating in the USB 1.1 and 2.0 modes.

The UDC20 is a USB low-level protocol interpreter. The UDC20 controls the USB bus protocol, packet generation / extraction, PID / Device ID parsing, and CRC coding / decoding with autonomous error handling. It is capable of operating either in USB 1.1 or 2.0 compliant modes. It has autonomous protocol handling functions like stall condition clearing on setup packets, suspend / resume / reset conditions, and remote wake-up. It also autonomously handles the error conditions such as retry for CRC errors, Data toggle errors, and generation of NYET, STALL, ACK and NACK depending on the endpoint buffer status.

During the power down state, the SIE clock is stopped. The SIE can asynchronously detect a USB Reset and/or USB Resume condition and wake-up the Processor.

The SIE block contained in the SEC2410/SEC4410 is fully USB2.0 Specification Compliant.

The transport interface sits between the VCI bus interface of the UDC20 and the AMBA bus. It does the buffer management and endpoint control to allow transfer between the VCI bus and the AMBA bus.

16.1.1 SIE Transport Interface

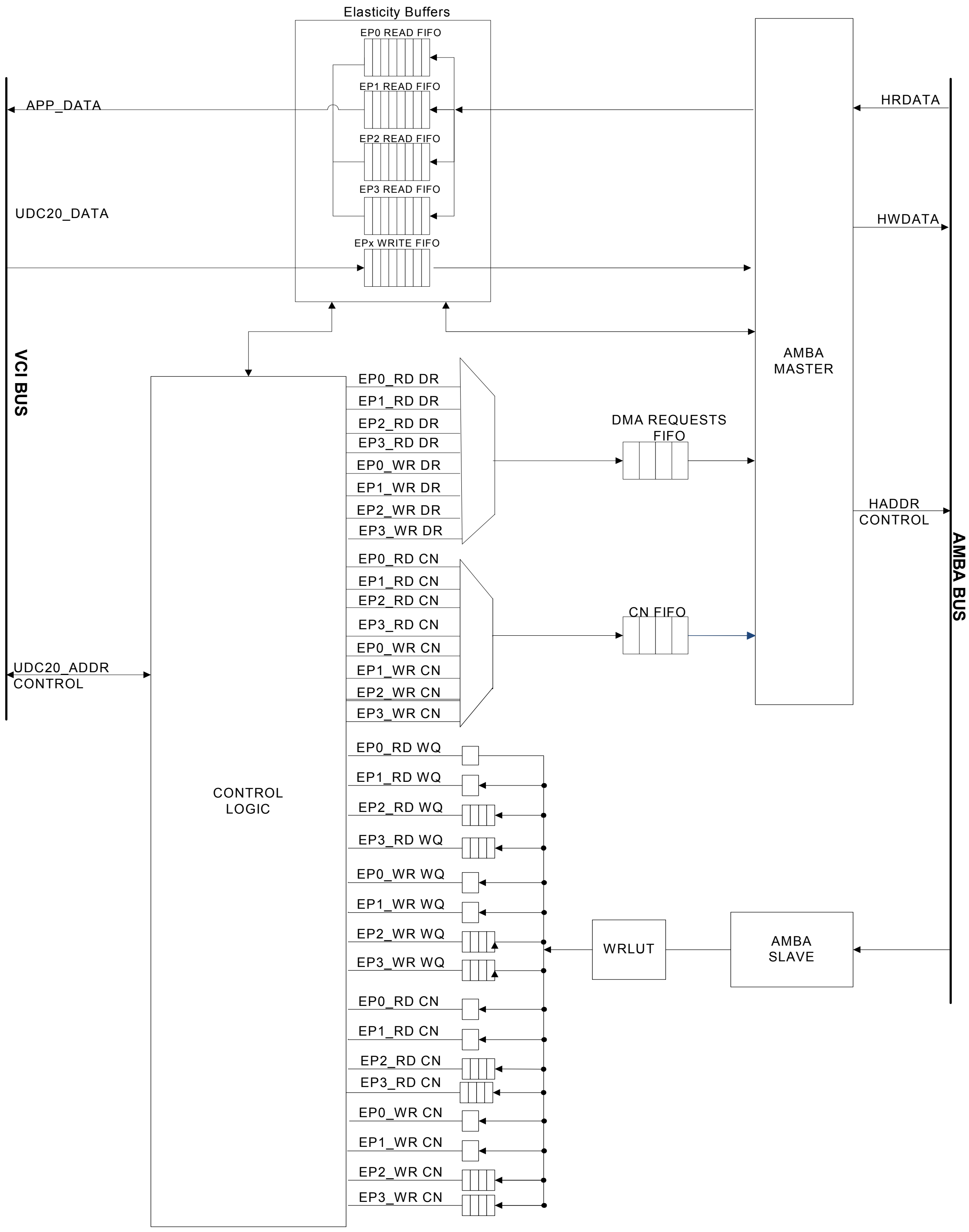


Figure 16.2 SIE Transport Interface

The SIE transport interface is responsible for bridging the VCI accesses to the AMBA bus.

16.1.2 SIE Transport AMBA Slave

The AMBA slave block used in the SIE transport interface is identical to the AMBA slave block used by the FMDU and the CR_AHB bridge. The AMBA slave decodes accesses to its address range. If the address is in the range of the work queues, the data is treated as a WQE. If in that range, the WQE context is used as an index into the Work Request Look Up Table (WRLUT) to fetch the Work Request and Completion Notification. The WR and CN is forwarded to the queueing logic, along with the number of the endpoint being addressed.

If the access is in the 1K address space of the AMBA slave, but not in the address range of the work queues, the data is forwarded without doing a table lookup.

16.1.3 SIE Transport AMBA Master

The AMBA master block used in the SIE transport interface is identical to the AMBA master block used by the FMDU. There are three inputs to the AMBA master. They are a CN, and a DMA interface. The AMBA master fulfills the CN and DMA in ping pong fashion.

16.1.4 CN write:

The CN is a single 32 bit write on the AMBA bus, where the AMBA address is the CN address, and the AMBA data is a concatenation of the context, last bit and buffer length. See section Completion Notification (CN).

16.1.4.1 DMA interface:

The DMA the same thing as a WR, with the added field of the endpoint number. If the control logic wanted to read 10 bytes into EP0_READ from buffer location 0x008000, followed by 200 bytes write to buffer location 0x008800 (the endpoint does not matter), it would issue two tuples:

5) 0x008000, x010, EP0_READ

6) 0x008800,0x200, EPx_WRITE

The AMBA master would get control of the bus, read out memory, starting at location 0x8000 and write into EP0_READ memory. When that transfer is over, it start on the second one. It would read out of the EPx_Write FIFO and write into memory starting at location 0x008800. The endpoint number tells it what Fifo to read or write.

16.1.5 SIE Transport Work/CN Queues

The AMBA slave block puts out the WR, CN, Endpoint number and a strobe. The Endpoint number and the strobe are used to latch in the WR and CN into the appropriate endpoint queue. For Sparrow, the queue for EP0 and EP1 is only one deep. For EP2 it is four deep. Future designs may be different. Writing more elements than the queue is deep will lead to unpredictable results.

The queue for the WR are implemented as Fifos when written from the AMBA slave side. From the control logic side they can be read in parallel. There is also the additional ability for the control logic to write to the top element from the control logic side.

The ability to write back into the top of the work queue fifo is used in the following way. If the posted buffer is bigger than an MTU, then at the end of a packet, the remainder is written back into the top of the queue.

For example: Assume the top element is address 0x008000, length 0x800, and the associated CN is address 0x00F000 and the context is 0x05.

The first packet arrives. It is length 0x200. Since the buffer is not complete, the CN cannot be sent. So the hardware writes 0x200 bytes into the buffer. To remember where it is, the hardware writes back into the fifo, this time the top element address is 0x008200, length 0x200, CN address 0x00F000, context is 0x05.

The second packet arrives. For this packet, the buffer address is 0x008200, and the length is 0x200. The packet length is 0x200. This time the buffer is fully used up. A CN is posted to show that the buffer has been fulfilled.

The ability to read the WRs in parallel is important for Endpoint 2 since multiple buffers may be required to make up an MTUs worth of data. Multiple WRs need to be read for small packets.

16.1.6 SIE Transport Elasticity buffers

There are four buffer used in Sparrow for Elasticity. On the read side, there is one buffer per endpoint. The buffers are necessary because the UDC20 requires that on reads, the data be provided one clock after the request. Since the AMBA bus has latency associated with it, each endpoint has the start of the buffer pre-fetched and stored locally. When the read request comes in, the data is sent from the Fifo, and at the same time, a request goes out to the AMBA master to fetch the rest of the data. The depth of the Fifo is greater than the latency of the AMBA bus so that it ensures that the VCI bus is never starved for data.

As a buffer is posted on a read queue, that read endpoint is not enabled (from the VCI's perspective) until the pre-fetch buffer is filled.

All the write endpoints share a common fifo. As soon as a buffer is posted for a write endpoint, that endpoint is ready to accept data. When a write occurs from the VCI bus, the data does to the EPx_WRITE_FIFO, if the associated write endpoint is enabled. If the endpoint is not enabled, the accesses is NAKed. In parallel with the data being written, the control logic looks up the WR information associated with that endpoint. The control logic posts a DMA request to AMBA master for a data write, with the address taken out of the WR for that endpoint. The data accumulates in the FIFO while the AMBA master waits for access to the bus. Once bus access is granted, the write Fifo will be drained much faster than it will be written to from the VCI side.

16.1.7 Endpoint Control

The endpoint control registers control how the transport interface interacts with endpoint accesses from the VCI bus of the UDC20.

16.1.7.1 Controlling EP0

EP0 in SEC2410/SEC4410 is a control endpoint. It can only operate in Continuous mode. EP0_READ and EP0_WRITE must be programmed the same for correct operation. If CONT_MODE is set to '0', the endpoint is disabled except for inbound Setup packets. BLK_XFER_EN has no effect on this endpoint.

If CONT_MODE is set to '1' and there is a buffer available, the transport interface will "ACK" an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed.

Since EP0 is not a DMA endpoint, only one buffer can be posted for EP0_READ and one for EP0_WRITE. A buffer is available for transfer for EP0_WRITE if one of the following conditions is met:

1. The buffer is bigger than or equal to an MTU size.
2. The buffer has the last bit set. If the incoming packet is bigger than the buffer, the OVERFLOW error is set for the transfer.

A buffer is available (enabled) for transfer for EP0_READ if the following condition is met:

1. The buffer is smaller or equal to the pre-fetch size, the whole buffer must be pre-fetched before the endpoint can be enabled.
2. If the buffer is bigger than the pre-fetch buffer, but less than or equal to an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data.
3. If the buffer is bigger then an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data in the

MTU. Once an MTU's worth of data has gone, the available buffer is decremented by an MTU size. The remaining buffer is available for the next transfer.

EP0_WRITE will always accept Setup packets, as long as the USB interface is enabled.

EP0_WRITE will set the LAST_BIT in a completion notification if a short packet comes in, it is not a SETUP packet, and the NO_LAST bit is cleared. In all other conditions, the LAST_BIT in the completion notification is cleared.

If more than one buffer is posted to EP0_READ or EP0_WRITE, it is an error with unpredictable results.

The MTU size for EP0_READ and EP0_WRITE must be programmed to 64 Bytes.

WR_COUNT has no meaning for EP0.

EP_CNT has no meaning for this endpoint.

16.1.7.2 Controlling EP1

EP1 in SEC2410/SEC4410 is a interrupt endpoint. It can only operate in Continuous mode. EP1_READ and EP1_WRITE should be programmed the same for normal operation, but there is no restriction about programming them differently. If CONT_MODE is set to '0', the endpoint is disabled. BLK_XFER_EN has no effect on this endpoint.

If CONT_MODE is set to '1' and there is a buffer available, the transport interface will "ACK" an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed.

Since EP1 is not a DMA endpoint, only one buffer can be posted for EP1_READ and one for EP1_WRITE. A buffer is available for transfer for EP0_WRITE if one of the following conditions is met:

1. For EP1_WRITE, the buffer is bigger than or equal to an MTU size.'
2. For EP1_WRITE the buffer has the last bit set. If the incoming packet is bigger than the buffer, the OVERFLOW error is set for the transfer.

A buffer is available for transfer for EP1_READ if the following condition is met:

1. The buffer is smaller or equal to the pre-fetch size, the whole buffer must be pre-fetched before the endpoint can be enabled.
2. If the buffer is bigger than the pre-fetch buffer, but less than or equal to an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, the data from the pre-fetch buffer is sent out, while an AMBA read request is made for the remainder of the data.
3. If the buffer is bigger then an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data in the MTU. Once an MTU's worth of data has gone, the available buffer is decremented by an MTU size. The remaining buffer is available for the next transfer.

If more than one buffer is posted to EP1_READ or EP1_WRITE, it is an error with unpredictable results.

EP1_WRITE will set the LAST_BIT in a completion notification if a short packet comes in, and the NO_LAST bit is cleared. In all other conditions, the LAST_BIT in the completion notification is cleared.

The MTU size for EP1_READ and EP1_WRITE can be programmed to any valid value. The value in the Endpoint MTU register must exactly match the value in the UDC20 configuration for that endpoint.

WR_COUNT has no meaning for EP1.

EP_CNT has no meaning for this endpoint.

16.1.7.3 Controlling EP2

EP2 in SEC2410/SEC4410 is a Bulk endpoint, with DMA capability. It can operate in Continuous mode or DMA (Block Transfer) mode. EP2_READ and EP2_WRITE should be programmed the same for normal operation, but there is no restriction about programming them differently. If both CONT_MODE and BLK_XFER_EN are set to ‘0’, the endpoint is disabled.

In Continuous mode, CONT_MODE is set to ‘1’ and BLK_XFER_EN is set to ‘0’. If there is a buffer available, the transport interface will “ACK” an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed.

In DMA mode, CONT_MODE is set to ‘0’ and BLK_XFER_EN is set to ‘1’. BLK_XFER_EN must be enabled after setting the EP_CNT for the endpoint to the length of the transfer. If there is a buffer available, the transport interface will “ACK” an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed. During a DMA transfer EP_CNT is reduced by the packet size after each transfer. The reduction can only be done after getting an “good” status from the VCI bus for that transfer. When the count goes down to zero, BLK_XFER_EN is cleared, and BLK_XFER_COMPLETE is set. If the last reduction results in EP_CNT going less than zero, it is an error, and the OVERFLOW bit is set.

A DMA transfer can be terminated prematurely by the following events:

1. If the DMA is to the USB host, (EP2_READ active) and a buffer arrives with the LAST bit set, it will terminate the transfer. If EP_CNT does not go to zero, then the RESIDUAL interrupt is set, indicating an error.
2. If the DMA is from the USB host (EP2_WRITE active), and the host sends a short packet, it will terminate the transfer. If EP_CNT does not go to zero, then the RESIDUAL interrupt is set, indicating an error.
3. If the DMA is from the USB host (EP2_WRITE active), and the host sends a ZLP, it will terminate the transfer. The ZLP interrupt is set, and the RESIDUAL interrupt is set, indicating an error.

At the end of each successful packet, the hardware must readjust the WR addresses and length, and issue CNs as appropriate.

There can be up to four buffers available queued up and ready for EP2_READ and EP2_WRITE. A buffer is available for transfer for EP0_WRITE if one of the following conditions is met:

1. The buffer is bigger than or equal to an MTU size.
2. The buffer has the last bit set. If the incoming packet is bigger than the buffer, the OVERFLOW error is set for the transfer.
3. There are at least two buffers in the queue.

If only a single buffer has been posted, a buffer is available for transfer for EP2_READ if the following condition is met:

1. The buffer at the head of the queue is smaller or equal to the pre-fetch size, and has the last bit set, the whole buffer must be pre-fetched before the endpoint can be enabled.
2. If the buffer at the head of the queue is bigger than the pre-fetch buffer, but less than or equal to an MTU, and has the last bit set, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data.
3. If the buffer at the head of the queue is bigger then an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data in the MTU. Once an MTU’s worth of data has gone, the available buffer is decremented by an MTU size. The remaining buffer is available for the next transfer.

If the buffer at the head of the queue has the last bit set, the previous set of conditions apply regardless of the number of buffers posted. If the last bit is not set, and two or more buffers have been posted, a buffer is available for transfer for EP2_READ if the following condition is met:

1. If the sum of the lengths of the top two buffers is less than or equal to the pre-fetch buffer size, both buffers must be pre-fetched before the endpoint can be enabled.
2. If the buffer at the top is smaller than the pre-fetch buffer, and the sum of the top two buffers is greater than the pre-fetch size, the first buffer must be pre-fetched, and enough of the second buffer to fill the pre-fetch buffer. When an IN occurs, the remainder of the second buffer up to an MTU size is fetched.
3. If the buffer at the top is bigger than the pre-fetch buffer, then enough of the buffer is fetched to fill the pre-fetch buffer. When an IN occurs, the remainder of the first buffer is fetched immediately followed by enough of the second buffer to fulfill an MTU.

If more than four buffers are posted to EP2_READ or EP2_WRITE, it is an error with unpredictable results.

The MTU size for EP2_READ and EP2_WRITE can be programmed to any valid value. The value in the Endpoint MTU register must exactly match the value in the UDC20 configuration for that endpoint.

WR_COUNT for EP2_READ is the number of elements in EP2_READ work queue, and correspondingly WR_COUNT for EP2_WRITE is the number of elements in EP2_WRITE work queue. This number is not affected by the number of CNs pending.

16.1.7.4 Controlling EP3

EP3 in SEC2410/SEC4410 is a Bulk endpoint, with DMA capability. It can operate in Continuous mode or DMA (Block Transfer) mode. EP3_READ and EP3_WRITE should be programmed the same for normal operation, but there is no restriction about programming them differently. If both CONT_MODE and BLK_XFER_EN are set to ‘0’, the endpoint is disabled.

In Continuous mode, CONT_MODE is set to ‘1’ and BLK_XFER_EN is set to ‘0’. If there is a buffer available, the transport interface will “ACK” an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed.

In DMA mode, CONT_MODE is set to ‘0’ and BLK_XFER_EN is set to ‘1’. BLK_XFER_EN must be enabled after setting the EP_CNT for the endpoint to the length of the transfer. If there is a buffer available, the transport interface will “ACK” an endpoint data request. If there is no buffer available, endpoint data requests will be NAKed. During a DMA transfer EP_CNT is reduced by the packet size after each transfer. The reduction can only be done after getting an “good” status from the VCI bus for that transfer. When the count goes down to zero, BLK_XFER_EN is cleared, and BLK_XFER_COMPLETE is set. If the last reduction results in EP_CNT going less than zero, it is an error, and the OVERFLOW bit is set.

A DMA transfer can be terminated prematurely by the following events:

1. If the DMA is to the USB host, (EP3_READ active) and a buffer arrives with the LAST bit set, it will terminate the transfer. If EP_CNT does not go to zero, then the RESIDUAL interrupt is set, indicating an error.
2. If the DMA is from the USB host (EP3_WRITE active), and the host sends a short packet, it will terminate the transfer. If EP_CNT does not go to zero, then the RESIDUAL interrupt is set, indicating an error.
3. If the DMA is from the USB host (EP3_WRITE active), and the host sends a ZLP, it will terminate the transfer. The ZLP interrupt is set, and the RESIDUAL interrupt is set, indicating an error.

At the end of each successful packet, the hardware must readjust the WR addresses and length, and issue CNs as appropriate.

There can be up to four buffers available queued up and ready for EP3_READ and EP3_WRITE. A buffer is available for transfer for EP0_WRITE if one of the following conditions is met:

1. The buffer is bigger than or equal to an MTU size.
2. The buffer has the last bit set. If the incoming packet is bigger than the buffer, the OVERFLOW error is set for the transfer.

3. There are at least two buffers in the queue.

If only a single buffer has been posted, a buffer is available for transfer for EP3_READ if the following condition is met:

1. The buffer at the head of the queue is smaller or equal to the pre-fetch size, and has the last bit set, the whole buffer must be pre-fetched before the endpoint can be enabled.
2. If the buffer at the head of the queue is bigger than the pre-fetch buffer, but less than or equal to an MTU,and has the last bit set, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data.
3. If the buffer at the head of the queue is bigger then an MTU, the pre-fetch buffer must be filled before the endpoint can be enabled. If an IN comes in, an AMBA read request is made for the remainder of the data in the MTU. Once an MTU's worth of data has gone, the available buffer is decremented by an MTU size. The remaining buffer is available for the next transfer.

If the buffer at the head of the queue has the last bit set, the previous set of conditions apply regardless of the number of buffers posted. If the last bit is not set, and two or more buffers have been posted, a buffer is available for transfer for EP3_READ if the following condition is met:

1. If the sum of the lengths of the top two buffers is less than or equal to the pre-fetch buffer size, both buffers must be pre-fetched before the endpoint can be enabled.
2. If the buffer at the top is smaller than the pre-fetch buffer, and the sum of the top two buffers is greater than the pre-fetch size, the first buffer must be pre-fetched, and enough of the second buffer to fill the pre-fetch buffer. When an IN occurs, the remainder of the second buffer up to an MTU size is fetched.
3. If the buffer at the top is bigger than the pre-fetch buffer, then enough of the buffer is fetched to fill the pre-fetch buffer. When an IN occurs, the remainder of the first buffer is fetched immediately followed by enough of the second buffer to fulfill an MTU.

If more than four buffers are posted to EP3_READ or EP3_WRITE, it is an error with unpredictable results.

The MTU size for EP3_READ and EP3_WRITE can be programmed to any valid value. The value in the Endpoint MTU register must exactly match the value in the UDC20 configuration for that endpoint.

WR_COUNT for EP3_READ is the number of elements in EP3_READ work queue, and correspondingly WR_COUNT for EP3_WRITE is the number of elements in EP3_WRITE work queue. This number is not affected by the number of CNs pending.

16.1.7.5 Stalling Endpoints

Every endpoint in the USB interface can be stalled. There are two stall bits, STALL and PERSISTANT_STALL. Only one of the bits should be set by the firmware at one time. If the STALL bit is set for an endpoint, the hardware responds with a STALL handshake to an access to that endpoint. The response clears the STALL bit. The UDC20 keeps an internal copy of the STALL bit, and responds autonomously with STALLs on the USB bus until it receives a ClearFeature Endpoint Stall. At that point it clears its internal STALL bit, and normal operation resumes.

The PERISTANT_STALL is used in conjunction with EP2 Mass Storage Class protocol. The PERSISTANT_STALL is set and cleared by the firmware. If the bit is set for a particular endpoint, the hardware always responds with a STALL to any access to that endpoint.

16.1.7.6 Handshake Interrupts

Every endpoint in the USB interface can be has three handshake interrupts associated with it. These are ACK, NAK and ZLP. For a particular endpoint, these interrupts are set when there is a corresponding event happens. The interrupts are cleared by the firmware or when a reset occurs.

16.1.7.7 Resetting Endpoints

The firmware resets an endpoint by writing a ‘1’ to the EP_RESET bit in the endpoint control register. Resetting discards all information in the endpoint. When an endpoint is finished with the reset sequence it must clear the EP_RESET bit.

16.1.7.8 Flushing Endpoints

If a FLUSH comes into EPx_READ or EPx_WRITE, it is handled in the following way.

1. If the endpoint is in the middle of sending or receiving a packet, wait until that packet handling is completed, and the endpoint has gone back to the idle state.
2. If the endpoint is idle, and there is no work queue element posted, do nothing. Any pending completion notification in the outgoing CN queue are not affected and will go out normally.
3. If there are work queue elements pending, acts as if a zero length packet arrived for each one. That is, create a normal completion notification for that element, with a data length of zero. As each element is “transferred” to the completion queue, the WQE count goes down. Eventually all will be transferred to the completion queue.
4. Once all the pending CNs have gone out, and the endpoint is empty of both WQ elements and CNs, set the flush done bit.

When the firmware wants to FLUSH and endpoint, it will first disable the endpoint. This means CONT_MODE = ‘0’ for EP0 and EP1, and CONT_MODE = ‘0’ and BLK_XFER_EN = ‘0’ for EP2. After disabling the Endpoint, the firmware issues the flush to the Endpoint. The firmware is responsible for clearing the EP_COUNT if the FLUSH happened during a DMA transfer.

When the AutoCBW processor issues a FLUSH, it is to recover buffers it wants to reuse. For AutoCBW operations, the EP2 is always set to CONT_MODE operation.

16.1.8 Autonomous USB Protocol

16.1.8.1 Automatic Retries - Out Transactions

If a packet is received with an incorrect data toggle, the SIE will ACK, but ignores the data packet.

If more than 64 bytes received on EP0 or EP1, or if more than 512 bytes are received on EP2, the USB SIE will ignore the packet and set the appropriate “STALL” bit until the host acknowledges the condition by sending a “CLEAR FEATURE ENDPOINT STALL” command for that endpoint, or, in the case of Endpoint 0, a SETUP is received.

If an error occurs during an OUT transaction, the hardware rewinds its read pointer back to buffer point where the MTU started. The host then sends another OUT token and retransmits the packet.

Once the packet has been successfully received, the appropriate interrupt bit is set. The SIE can handle any number of back-to-back retries, but the host determines how many times a packet is retried.

If an endpoint has no available buffer, then the SIE sends a NACK. The exception is setup packets on EP0, which are always ACKed.

16.1.8.2 Automatic Retries - In Transactions

If an timeout (No response from the host / lost ACK) occurs during an IN transaction, the hardware reloads its USB SIE side buffer read pointer back to the beginning of the failed MTU. The host then sends another IN token and the SIE re-transmits the packet with the same data toggle PID.

Once the Processor has successfully received the packet (only upon ACK received by SIE), the appropriate endpoint interrupt bit is set. The SIE can handle any number of back-to-back retries, but the host determines how many times a packet is retried.

Upon reception of a SETUP token followed by the 8 byte DATA-0 packet on EP0, the internal DTOG bit for both EP0_WRITE and EP0_READ are set to one.

16.1.8.3 Configuring the UDC20

The SETUP_CMD_ADDR register must be programmed in the UDC20. The default value in the part will conflict with Status register. Additionally, the UDC20 powers up with the internal endpoints unconfigured. The firmware must configure the UDC20 for each individual endpoint using the UDC_CSR_ADDR, UDC_CSR_DATA and UDC_CSR_CFG registers. The parameters that must be programmed are mtu size, alternate interface number, interface number configuration number, type of endpoint, and endpoint number.

The following is an example of a configuration of EP0 and EP2 using Processor “C” code:

```
CR_AHB unsignedchar volatile udc_csr_cntrl_at_ 0x40012C00;// UDC20 CSR bus config register
CR_AHB unsignedlong volatile udc_csr_data_at_ 0x40012C04;// UDC20 CSR bus data
CR_AHB unsignedshort volatile udc_csr_addr_at_ 0x40012C0E;// UDC20 CSR bus address
#define SIE_EP0_RD_CSR_ADDR 0x0
#define SIE_EP0_WR_CSR_ADDR 0x4
#define SIE_EP1_RD_CSR_ADDR 0x8
#define SIE_EP1_WR_CSR_ADDR 0xC
#define SIE_EP2_RD_CSR_ADDR 0x10
#define SIE_EP2_WR_CSR_ADDR 0x14

#define SIE_EP3_RD_CSR_ADDR 0x18
#define SIE_EP3_WR_CSR_ADDR 0x1C
```

```
// Program the setup command address register
        udc_csr_data = 0x00000100;// setup command address register = 0x100
        udc_csr_addr = 0x0;                   // select register 0
        udc_csr_ cntrl = 0x1;                 // do a write


// Program EP0, read and write are the same register for EP0
        udc_csr_data = 0x02000000; // size = 64, alt =0, intf=0, conf=0, type= cntrl, ep= 0
        udc_csr_addr = SIE_EP0_RD_CSR_ADDR;  // select register address 4
        udc_csr_ cntrl = 0x1;                 // do a write


// Program EP2
        udc_csr_data = 0x100000D2; // size = 512, alt =0, intf=0, conf=1, type= bulk, dir=in, ep= 2
        udc_csr_addr = SIE_EP2_RD_CSR_ADDR; // select register address 0x10
        udc_csr_ cntrl = 0x1;                 // do a write

        udc_csr_data = 0x100000C2; // size = 512, alt =0, intf=0, conf=1, type= bulk, dir=out, ep= 2
        udc_csr_addr = SIE_EP2_WR_CSR_ADDR; // select register address 0x14
        udc_csr_ cntrl = 0x1; // do a write
```

16.1.9 USB Events

There are several events, which cause different parts of the SIE to be initialized. The following is the list of events and the respective actions.

16.1.9.1 USB Bus Reset

USB Bus Reset is recognized only when the clocks are running. If the device is in SUSPEND mode with the clocks stopped, a USB RESET will be first recognized as a RESUME event and if the WU_FM_SRC_1 bit for RESUME is unmasked, will restart the clocks. The USB RESET can only then be detected. Upon recognition it causes the following:

1. All SIE endpoint registers are set to their POR values, all stall conditions, the SETUP bit, are cleared. The PID sequencers, internal DTOG are reset for all endpoints
2. The following registers will be set to their POR values: SIE_SRC, USB_CONF, and endpoint control registers.
3. If the USB_RESET in USB_STAT is unmasked, then a ISR_0 interrupt (USB_STAT) is generated to the Processor.
4. The following registers will be set to their POR values: ISR_0, Endpoint Control registers.

Note 16.3 EP0 is always available once the device is powered and the USB Bus Reset has been received.

16.1.9.2 Suspend

This is detected by the SIE when the idle condition on the USB bus occurs for a duration of more than 3ms. Upon detection of this condition the SIE sets the SUSPEND interrupt bit of SIE_SRC. It shuts down the USB PHY, and shuts down the clocks coming to it. No firmware intervention is required to shut down the PHY. After getting the interrupt, the Processor can either do nothing, or shut the clocks off completely. This is done using the options in the CLOCK_CTL register. If the hardware detects a wake-up event while the Processor tries to shutdown the PLL, the PLL will be automatically restarted. If an attempt is made to disable the PLL while the UDC20 is connected to the USB bus, and it is not in suspend, the PLL will remain enabled.

If the UDC20 is disconnect from the USB bus, the UDC20 block automatically goes to the suspend state.

16.1.9.3 Setup token Arrival

When a SETUP token is recognized, the following sequence happens.

1. Independent of the state of SETUP bit, the setup data packet is received on EP0_WRITE and ACK is sent for the received setup packet.
2. The stall condition, if any, for EP0_READ and EP0_WRITE are cleared.
3. The internal DTOG bit for both EP0_READ and EP0_WRITE are set to one.
4. The data is stored at the location pointed to by WRLUT entry 15.
5. The SETUP bit in SEI_SRC register is set.
6. EP_RESET is set for EP0_READ and EP0_WRITE.
7. A completion notification is sent out for the Setup packet, using WRLUT entry 0 for the parameters.
8. EP0_WRITE is flushed. Any accumulated data is sent out and Completion Notifications are posted.
9. EP0_READ will discard any accumulated data not yet transferred to the host.
10. When the flush is completed, EP_RESET is set, and the endpoint is Reset.
11. The Processor will re-enable the endpoint once it has processed the Setup packet.

12. Until the endpoint is re-enabled, all OUT packets to EP0_WRITE are NACKed.
13. Until the endpoint is re-enabled, all IN packets to EP0_READ are NACKed.

16.1.9.4 Resume

This global resume condition is recognized asynchronously and does not require the SIE clock running. Upon recognition it causes the following. A USB RESET will be interpreted as a RESUME if it occurs while clocks are stopped.

On detecting the resume conditions from the powerdown state, the following happens:

1. The ring oscillator is started.
2. The system clock source is set to the ring oscillator.
3. The PLL is started.
4. The RESUME bit in the WU_FM_SRC_1 register is set.
5. A resume interrupt is generated if the interrupt is unmasked.
6. The SIE and USB PHY resumes from power down state.
7. The hardware wait for the PLL to lock.
8. Once PLL locks, the system clock is switched to the PLL

16.1.9.5 Remote Wake-up

When the Processor is required to go into power down state, the Processor simply clears the ROSC_EN bit and the PLL_EN bit. The hardware detects the powerdown condition of the processor and shuts down the PLL and ring oscillator. When a remote wake-up event happens, the ring oscillator is started.

16.1.9.6 Standard Device Requests

The SIE also handles autonomously several standard device requests received on Endpoint 0. These requests are: SET_CONFIGURATION, GET_CONFIGURATION, SET_FEATURE_ENDPOINT_HALT, CLEAR_FEATURE_ENDPOINT_HALT, SET_FEATURE_REMOTE_WAKE_UP, CLEAR_FEATURE_REMOTE_WAKE_UP, GET_INTERFACE, SET_INTERFACE and SET_ADDRESS. These events (except SET_ADDRESS, GET_INTERFACE (always returns 0), and GET_CONFIGURATION) are indicated in the SIE_SRC register, which can generate an interrupt to the Processor core’s INT3 line. The configuration number, resulting from the SET_CONFIGURATION command is stored in the USB_CONF register. This value is used when reporting to the host on a GET_CONFIGURATION Command, also. All other device requests are handled normally and will generate the SETUP status bit when received.

Note 16.4 The SIE hardware will automatically return a STALL to all non-zero-length packets during a status out phase.

16.1.9.7 SIE Configurations

Upon POR or the detection of USB RESET, the Configuration of the device is cleared to “0”. The host may change its Configuration state to “1” with a “SET CONFIGURATION” command on Endpoint 0. All other Configuration number requests by the host will result in a STALL condition on Endpoint 0. For Configuration 0, only Endpoint 0 RX and TX are enabled, while all endpoints are enabled for Configuration 1.

16.1.9.8 SIE Disconnect

The SIE must be disconnected from the USB bus when a cable disconnected is detected on VBUS. The SIE connects when VBUS is applied unless held in reset externally.

16.2 SIE Registers

16.2.1 SIE Registers in AHB space

16.2.1.1 Work Queue Entries

In the 24 bit model the SIE is at base address 0x00F000. The SIE has 4 read endpoints and 4 write endpoints. The work queue addresses for the SIE is shown in the following table. Work Queue depth varies by endpoint number. EP0 and EP1 are only one deep, while EP2 and EP3 is 4 deep.

Table 16.1 SIE Work Queue Addresses

EP NUMBER	READ	WRITE	WQ DEPTH
0	0x00F100	0x00F180	1
1	0x00F108	0x00F188	1
2	0x00F110	0x00F190	4
3	0x00F118	0x00F198	4

16.2.2 SIE Registers in SPB space

16.2.2.1 Work Request Look Up Table (WRLUT)

The SIE has 16 Work Request entries in it Work Request Look up table.

Table 16.2 SIE WRLUT

ADDRESS	CONTEXT	WORK REQUEST
CR_X32 + 0x2D00	0	Work Request 0
CR_X32 + 0x2D10	1	Work Request 1
CR_X32 + 0x2D20	2	Work Request 2
CR_X32 + 0x2D30	3	Work Request 3
CR_X32 + 0x2D40	4	Work Request 4
CR_X32 + 0x2D50	5	Work Request 5
CR_X32 + 0x2D60	6	Work Request 6
CR_X32 + 0x2D70	7	Work Request 7
CR_X32 + 0x2D80	8	Work Request 8
CR_X32 + 0x2D90	9	Work Request 9
CR_X32 + 0x2DA0	10	Work Request 10
CR_X32 + 0x2DB0	11	Work Request 11
CR_X32 + 0x2DC0	12	Work Request 12
CR_X32 + 0x2DD0	13	Work Request 13
CR_X32 + 0x2DE0	14	Work Request 14

ADDRESS	CONTEXT	WORK REQUEST
CR_X32 + 0x2DF0	15	Setup Work Request 15

Note 16.5 For the SIE, Work Request entry 15 is dedicated to the EP0 setup packets.

16.2.2.2 Endpoint Control Registers

EP0 Read, EP0 Write, EP1 Read, EP1 Write are all simple Endpoints.

EP2 Read, EP2 Write, EP3 Read and EP3 Write are DMA Endpoints.

EP1, EP2, EP3 can be configured as Control, Interrupt, Bulk or Isochronous without restriction.

The base address of all the endpoints are according to the table below:

Table 16.3 SIE Endpoints Table

ADDRESS	NAME	TYPE
CR_X32 + 0x2F00	EP0_READ	Basic
CR_X32 + 0x2F10	EP1_READ	Basic
CR_X32 + 0x2F20	EP2_READ	DMA
CR_X32 + 0x2F30	EP3_READ	DMA
CR_X32 + 0x2F80	EP0_WRITE	Basic
CR_X32 + 0x2F90	EP1_WRITE	Basic
CR_X32 + 0x2FA0	EP2_WRITE	DMA
CR_X32 + 0x2FB0	EP3_WRITE	DMA

In the SIE, the ENABLE bit and the STALL bit have meaning.

EP0_WRITE and EP0_READ are always enabled and all other endpoints are enabled when configuration is set to 1.

For TX endpoints, when the STALL bit is set to a “1”, EP0 TX will respond with the STALL handshake to IN tokens. This bit set by the Processor. Receipt of a SETUP packet or USB RESET clears this bit. Writing a “0” to this bit has no effect.

For RX endpoints, when STALL bit is set to a “1”, EP0 will respond with the STALL handshake to OUT tokens EXCEPT a SETUP, which it will ACK unconditionally. This bit set by the Processor. Receipt of a SETUP packet or USB RESET clears this bit. Writing a “0” to this bit has no effect.

16.2.3 Configuration Registers

Refer to the UDC20 Revision 1.8 for the correct programming values of the CSR registers.

Table 16.4 UDC CSR Control Register

UDCCSR_CNTR (CR_X32 + 0X2C00 RESET=0X00)			**UDC CSR DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[7:2]	Reserved	R	Reserved. Always reads ‘0’
1	CSR_RD_STRB	R/W	Trigger an Endpoint Read to CSR registers. This bit self clears when the operation is complete. ‘0’=No Action ‘1’=A read operation is triggered
0	CSR_WR_STRB	R/W	Trigger an Endpoint Write to CSR registers. This bit self clears when the operation is complete. ‘0’=No Action ‘1’=A write operation is triggered

Table 16.5 UDC CSR Data Register

UDCCSR_DATA (CR_X32 + 0X2C04 RESET=0X00000000)			**UDC CSR DATA REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[31:0]	CSR_DATA[31:0]	R/W	UDC CSR Data Register D31:0. This value is written into the CSR register pointed to by the UDCCSR_ADDR register.

Table 16.6 UDC CSR Address Register

UDCCSR_ADDR (CR_X32 + 0X2C0E~0X2C0F - RESET=0X0000)			**UDC CSR ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[15:2]	CSR_ADDR[15:2]	R/W	The address of the UDC CSR register to be programmed is stored in this register.
1:0	Reserved	R	Always read ‘0’

Table 16.7 USB Configuration Number Register

USB_CONF (CR_X32 + 0X2C01 - RESET=0X00)			USB CONFIGURATION NUMBER REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always returns a “0”.
3:0	CONFIG	R	Reflects the current configuration number of the SEC2410/SEC4410 system as set by the USB Host.

Table 16.8 SIE Configuration Register

SIE_CONF (CR_X32 + 0X2C02 - RESET=0XC8)			SIE CONFIGURATION REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	BUS_PWR_EN	R/W	“1” = Enables SEC2410/SEC4410 as bus power device. “0” = Disables SEC2410/SEC4410 as bus power device and enables it as self-power device. Default value is “1”. The setting of this bit does not do any bus power/self-power switching or power control. The hardware checks the setting of this bit when it replies to the Get_Status or Get_Configuration Request from Host.
6	DISCONNECT	R/W	1 = Forces the PHY to the DISCONNECT state When the device is in Full Speed (FS) mode, the RTERM resistor is removed from the USB+ pin. When the device is High Speed (HS) mode, the termination resistors on USB+/USB- are removed. 0 = Normal operation. Default value is “1”.
[5:4]	Reserved	R	These bits always read “0”.
3	DATA_CONCAT	R/W	0: No data concatenation/separation is performed. 1: Data Concatenation/Separation mode: Treats data between two buffers as consecutive if these have the same direction and Endpoint number. Default value is “1”.
2	SPEED	R	“1” = High speed operation, if Host is capable. “0”= Full Speed operation.
1	RESUME	R/W	“1” = Forces the SIE to transmit Resume Signaling (“K” State) on the line, if this capability has been enabled by the SET_FEATURE_REMOTE_WAKEUP command from the USB host. This bit is set by the Processor after it wakes up from a power down state, for remote wake-up operation. Internal software times the duration of this signaling in accordance with the USB specifications. “0”= Normal operation
0	Reserved	R	Always read ‘0’

16.2.3.1 FIFO threshold register

A register has been added to contain the SIE read fifo threshold.

Table 16.9 SIE FIFO Threshold Register

SIE_FIFO_THRESHOLD (CR_X32 + 0X2C03 - RESET=0X03)			**SIE FIFO THRESHOLD REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read ‘0’
3:0	FIFO_THRESHOLD	R/W	The four bits threshold sets the pre-fetch threshold for the SIE Read FIFO. The setting is in DWORDS and the trip point is one more than the value programmed into this register. The default value is 3. That sets the threshold at 4 double-word level, i.e. 16-byte. The threshold will be ignored for buffers with a length less than the threshold that have the last bit is set.

Table 16.10 USB Bus Status Register

USB_STAT (CR_X32 + 0X2C08 - RESET=0X00)			**USB BUS STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	This bit always reads “0”.
0	USB_RESET	R/W	“1” = Indicates that a USB Reset has been detected. This bit is set after the USB reset signal from the UDC20 is removed.

Note 16.6 The bits in this register are cleared by writing a ‘1’ to the corresponding bit. These bits are OR'd, if unMASKED in the USB_MSK register, and drive the USB_STAT bit in the ISR_0 register.

Note 16.7 These bits are routed to the interrupt controller to individual interrupt lines. If the ineterrupts need to be masked, It must be done inside the interrupt controller. Individual interrupts are cleared by writing a “1” to the bit location.

Table 16.11 SIE Source Register

SIE_SRC (CR_X32 + 0X2C0A - RESET=0X00)			**SIE SOURCE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	W	Always read ‘0’
6	CLR_STALL	R/W	Set to “1” if a CLEAR_FEATURE_ENDPOINT_HALT command is received on any endpoint by the SIE. Which endpoint’s STALL condition is cleared can be determined by examining their CTL registers.
5	SET_CONF	R/W	Set to “1” if a SET_CONFIGURATION command is received on endpoint 0 by the SIE and the resulting configuration is set and reported in the USB_CONFIG register.

SIE_SRC (CR_X32 + 0X2C0A - RESET=0X00)			SIE SOURCE REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
4	SUSPEND	R/W	Suspend – If 3ms of IDLE state are detected by the hardware, the PHY goes to Full Speed mode, 2ms later, this bit will be set.
3	SET_INTF	R/W	Set to “1” if a SET_INTERFACE command is received on endpoint 0 by the SIE.
2	SETUP	R/W	“1”= A SETUP packet was received on Endpoint 0. If another SETUP packet is received on Endpoint 0 while this bit is high, the bit will go low and then immediately high again, to signal the duplicate SETUP. Receipt of a setup packet disables the endpoint Receipt of a setup packet flushes EP0_WRITE and EP0_READ and sets corresponding EP_RESET bits
1:0	Reserved	R	Always read ‘0’

Note 16.8 These bits are routed to the interrupt controller to individula interrupt lines. If the ineterrupts need to be masked, It must be done inside the interrupt controller. Individual interrupts are cleared by writing a “1” to the bit location.

Table 16.12 Endpoint Interrupt Register

SIE_EP_INTR (CR_X32 + 0X2C0C - RESET=0X00)			ENDPOINT REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	SIE_EP3_WR	R	EP3 Write Endpoint interrupt. Sets EP3_WR in GIRQ19. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
6	SIE_EP3_RD	R	EP3 Read Endpoint interrupt. Sets EP3_RD in GIRQ19. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
5	SIE_EP2_WR	R	EP2 Write Endpoint interrupt. Sets EP2_WR in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
4	SIE_EP2_RD	R	EP2 Read Endpoint interrupt. Sets EP2_RD in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
3	SIE_EP1_WR	R	EP1 Write Endpoint interrupt. Sets EP1_WR in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
2	SIE_EP1_RD	R	EP1 Read Endpoint interrupt. Sets EP1_RD in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
1	SIE_EP0_WR	R	EP0 Write Endpoint interrupt. Sets EP0_WR in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
0	SIE_EP0_RD	R	EP0 Read Endpoint interrupt. Sets EP0_RD in the interrupt controller. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.

16.3 USB 2.0 PHY

This block adheres to the UTMI specification for the PHY.

16.3.1 Phy Control register

Register to control the Phy.

Table 16.13 Phy control Register

SIE_PHY_CTL (CR_X32 + 0X2C20 - RESET=0X03)			**SIE PHY CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
4:5	BOOST_IOUT_A	R/W	USB electrical signaling drive strength Boost Bit for Phy. ‘00’ = Normal electrical drive strength. ‘01’ = Elevated electrical drive strength (+7.4% boost). ‘10’ = Elevated electrical drive strength (+14.7% boost). ‘11’ = Elevated electrical drive strength. (+25% boost)
3	PORT_SWAP	R/W	0 = Upstream DP DM not swapped 1 = Upstream DP DM swapped
2:0	SQUELCH	R/W	These three bits control the Squelch setting of the Phy Values to be provided by design

Table 16.14 HSIC Tuning Register

SIE_HSIC_TUNE (CR_X32 + 0X2C24 - RESET=0X03)			**SIEHSIC TUNING REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
6			
5			
4:0	RTUNE	R/W	These three bits control the Squelch setting of the Phy Values to be provided by design

Chapter 17 USB2 Phy and HSIC Interface

17.1 USB 2.0 Phy and HSIC

This section describes the registers used to modidy the behavior of the USB 2.0 Phy and HSIC interface. The registers in this section reside on the CRX32 bus in the address range 0x40007C00-0x40007FFF. OFFSET in the registers below is 0x40007C00.

There is one set of registers for the USB2 Phy control, starting at OFFSET + 0x000, and a second set of registers for the HSIC port starting at OFFSET + 0x200. The incoming address is 32 bits wide, but the address space reserved for the configuration registers is 1KByte. Thus only bits [9:0] of the address are used. The data path is assumed to be 32 bits.

The registers are aligned to 32 bit boundaries.

17.2 USB2 Phy Control Registers

Following sections describe the functionality of each register used in the control of the USB2 Phy.

Table 17.1 AFE HS Control Register

AFE_HS_CTRL (OFFSET+0X000 - RESET= 0X00)			**AFE HS CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
8	AFE_PORT_SWAP	R/W	0 = Upstream DP DM not swapped 1 = Upstream DP DM swapped
7:6	AFE_HS_TXVALID	R/W	HS Transmit Valid Mask. Indicates which bits of the AFE_HS_TXVALID[1:0] bus is valid. 2'b00: No bits valid 2'b01: LSB valid 2'b10: Invalid combination 2'b11: Both bits valid
5:4	AFE_HS_TXDATA	R/W	HS Transmit Data. Driver data is transmitted LSB first.
3	AFE_HS_TERM_EN	R/W	HS Termination Control. Enable the 45kohm termination on DP and DM when active.
2	AFE_HS_CS_EN	R/W	HS Current Source Enable. 1'b0: Driver powered-down 1'b1: Driver powered-up This signal will be asserted active whenever the port is in Hi-Speed mode.
1:0	AFE_HS_DRIVE	R/W	HS Output Current. 2'b00: Nominal 17.78mA 2'b01: Increase by 4% 2'b10: Increase by 8% 2'b11: Increase by 12%

Table 17.2 AFE Regulator Test Register

AFE_TEST_REG_IN (OFFSET+0X004 - RESET= 0X00)			**AFE REGULTOR TEST REGISTER**
BIT	**NAME**	R/W	**DESCRIPTION**
7	SQUELCH_HS_DISC_PWR_DOWN_MODE	R/W	Squelch/HS disconnect power down mode. When enabled, enabling of squelch/HS disconnect is a function of tx_active.
6:5	REF_VOLTAGE_TEST	R/W	Reference Voltage Test. Varies reference voltage levels for squelch and HS disconnect. Also muxes reference voltage to ATEST. 00: Default 01: -25mV change 10: +25mV change 11: 1.2V ref voltage muxed to ATEST. atest_sel must be enabled.
4	RX_CURRENT_SEL	R/W	RX Current Select. When enabled, bias current into FS/HS differential RX, HS squelch, and HS disconnect is increased by an additional 10uA.
3:2	HS_TX_RISEFALL_ADJUST	R/W	HS TX rise/fall adjust. 00: Default 01: +18% 10: -18% 11: -12%
1:0	LS_TX_RISEFALL_ADJUST	R/W	LS TX rise/fall adjust. 00: Default 01: +100% 10: -50% 11: -30%

Table 17.3 AFE Test Enable Register

AFE_TEST_EN (OFFSET+0X008 - RESET= 0X00)			**AFE TEST ENABLE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	AFE_RPU_DP_EN	R/W	Rpu DP Termination Control. Enable the 1.5Kohm termination on DP when active.
6	AFE_RPU_DM_EN	R/W	Rpu DP Termination Control. Enable the 1.5Kohm termination on DP when active.
5	AFE_RPD_DP_EN	R/W	Rpd DP Termination Control. Enable the 15Kohm termination on DP when active.
4	AFE_RPD_DM_EN	R/W	Rpd DM Termination Control. Enable the 15Kohm termination on DM when active.
3:2	XCVRSELECT	R/W	Transceiver Select. This signal selects between the LS, FS and HS transceivers. Switch synchronous with clk60. 2'b00: HS mode 2'b01: FS mode 2'b10: LS mode 2'b11: LS data-rate with FS rise/fall times (and EOP/IDLE) Note: xcvrselect must change state only when the device is not actively transmitting or receiving

1	AFE_HS_TXACTIVE	R/W	HS Driver Active. Places the HS driver in low power mode when disabled (during IDLE). afe_hs_cs_en must be enabled. 1'b0: Driver in low-power mode 1'b1: Driver in active transmit mode (17.78mA source active)
0	TEST_1P2_EN	R/W	HS Current Source Enable. 1'b0: Driver powered-down 1'b1: Driver powered-up This signal should be active whenever the port is in HS mode.

Table 17.4 AFE Test Register Select Register

AFE_TEST_REG_SEL (OFFSET+0X00C - RESET= 0X00)			**AFE TEST REGISTER SELECT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	AFE_REG_SEL	R/W	AFE test register select. 1'b0: passes functional signals to afe 1'b1: passes test signals to afe This signal muxes. To the afe, the test signals from chp_tst or the phy test reg with corresponding functional signals.

Table 17.5 Frequency Tune Bypass Register

FTUNE_BYPASS_CTL (OFFSET+0X010 - RESET= 0X00)			**FREQUENCY TUNE BYPASS CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	FTUNE_BYPASS	R/W	When enabled, the PLL frequency tune is controlled by the ftune_bypass_setting[27:23] bits

Table 17.6 Frequency Tune Setting Register

FTUNE_SETTING (OFFSET+0X014 - RESET= 0X00)			**FREQUENCY TUNE SETTING REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	Reserved	R	Always read ‘0’
4:0	FTUNE_BYPASS_ SETTING	R/W	Bypass value for PLL frequency tune. Effective when FTUNE_BYPASS is enabled.

Table 17.7 PLL Control Register

PLL_CTL (OFFSET+0X018 - RESET= 0X00)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5	VCO_UP	R/W	Increase VCO gain by 20% and center frequency by 10%
4	VCO_DN	R/W	Decrease VCO gain by 20% and center frequency by 10%
3	FORCE_ZTC_EN	R/W	Selects ZTC current for PLL
2	CP_OFF	R/W	Disable PLL charge pump when enabled.
1:0	CP_TRIM	R/W	Adjust charge pump output current. 00: Default (25uA) 01: 30uA 10: 35uA 11: 50uA

Table 17.8 PLL Regulator Control Register

PLL_REG_CTL (OFFSET+0X01C - RESET= 0X00)			**PLL REGULATOR CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always read ‘0’
2	INCREASE_REGLOAD	R/W	Adds additional static current to PLL regulator output for larger Cload stability 1'b0: Default. 250pF load. 1'b1: 1nF load.
1	REG_LOAD_COMOP_ENB	R/W	Disables no-loading compensation for PLL regulator. 1'b0: Disable 1'b1: Enable
0	REGULATOR_COMP_EN	R/W	Disables fast-transient compensation for PLL regulator. 1'b0: Disable 1'b1: Enable

Table 17.9 BandGap Control Register

BANDGAP_CTL (OFFSET+0X020 - RESET= 0X00)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’
1	INCREASE_VBG	R/W	Increase bandgap voltage by 2%.
0	DECREASE_VBG	R/W	Decrease bandgap voltage by 2%.

Table 17.10 Regulator Tune Bypass Control Register

RTUNE_BYPASS_CTL (OFFSET+0X024 - RESET= 0X00)			REGULATOR TUNE BYPASS CONTROL REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	RTUNE_BYPASS	R/W	When enabled, the output impedance tuning circuit is controlled by the RTUNE12_BYPASS_SETTING[5:0] and RTUNE33_BYPASS_SETTING[5:0] bits.

Table 17.11 HSIC Regulator Tune Register

HSIC_RTUNE_SETTING (OFFSET+0X028 - RESET= 0X00)			HSIC REGULATOR TUNE SETTINGREGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5:0	RTUNE12_BYPASS _SETTING	R/W	Bypass setting for 40/50ohm HSIC impedance tune.

Table 17.12 USB Regulator Tune Register

USB_RTUNE_SETTING (OFFSET+0X02C - RESET= 0X00)			USB REGULATOR TUNE SETTING REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5:0	RTUNE33_BYPASS _SETTING	R/W	Bypass setting for 45 ohm HS/FS impedance tune.

Table 17.13 ATEST Mux Control Register

AFE_MX_CTL (OFFSET+0X030 - RESET= 0X00)			AFE MUX CONTROL REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read ‘0’
3	MONLF_EN	R/W	Mux the PLL loop filter to ATEST. ATTEST_SEL must be enabled.
2	IZTC_TEST	R/W	Mux the port-7 100uA ZTC current to ATEST. ATTEST_SEL must be enabled
1	IRTC_TEST	R/W	Mux the port-7 100uA RTC current to ATEST. ATTEST_SEL must be enabled.
0	V12_REGULATO R_TEST	R/W	Mux 1.2V PLL sub-regulator supply to ATEST. ATTEST_SEL must be enabled.

Table 17.14 ATEST Select Tune Register

ATEST_SEL (OFFSET+0X034 - RESET= 0X00)			**ATEST SELECT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	ATTEST_SEL	R/W	Enable Analog Test Point: 1'b0: ATEST disabled. 1'b1: ATEST enabled. Only drive one analog signal to ATEST at a time.

Table 17.15 AFE Squelch Filter Test Register

AFE_SQFLT_TST (OFFSET+0X038 - RESET= 0X00)			**AFE SQUELCH FILTER TEST REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	AFE_SQFLT_TST	R/W	AFE squelch filter back off 1'b0: New filter - 2b quantization 1'b1: Old filter - 1b quantization

Table 17.16 AFE Test JK Register

AFE_TESTJK_EN (OFFSET+0X03C - RESET= 0X00)			**AFE TEST JK ENABLE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	AFE_TESTJK_EN	R/W	TEST J/K Enable. Controller drives this pin directly when in TEST_J or TEST_K mode. When enabled, the DC VOH level of the HS driver is lowered by 4%

Table 17.17 HSIC Upstream 50ohm Enable Register

HSIC_UP_EN50 (OFFSET+0X040 - RESET= 0X00)			**HSIC UPSTREAM 50 OHM ENABLE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	HSIC_UP_EN50	R/W	HSIC 50ohm Driver Enable (Strobe/Data). Selects the driver output impedance. 1'b0: 40ohm driver 1'b1: 50ohm driver.

Table 17.18 AFE Register

AFE_HS_CTRL (OFFSET+0X044 - RESET= 0X00)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:16	Reserved	R	Always read '0'
20:15	PHY_AFE_RTUNE12_OUTPUT	R	When rtune_bypass is disabled, these bits report the result of the 40/50? HSIC impedance tune.
14	PHY_AFE_TUNE_COMP	R	State of HSIC comparator output.
13	PHY_AFE_TUNE_1P2_EN	R	State of HSIC tune enable control
12	PHY_AFE_TUNE_INT	R	State of internal tune_en control.
11:7	PHY_AFE_FTUNE_OUTPUT	R	When ftune_bypass is disabled, these bits report the result of the automatic PLL frequency tune
6	PHY_AFE_RTUNE_DONE	R	Signals when the output impedance compensation is complete.
5:0	PHY_AFE_RTUNE_OUTPUT	R	When rtune_bypass is disabled, these bits report the result of the 45? HS/FS impedance tune.

Table 17.19 Phy Upstream Regulator Control Register

PHY_UP_REG (OFFSET+0X048 - RESET= 0X00)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read '0'
1	PHY_UP_AFE_HS_SQUELCH_B	R	AFE High Speed Squelch Indicates when the HS oversampled data is valid. Active low. 1'b0: Data valid 1'b1: Data is invalid
0	PHY_UP_AFE_DISC	R	AFE High Speed Disconnect Indicates when the line is disconnected in HS mode. This signal should only be strobed during HS EOP on the 32nd bit time. 1'b0: Normal condition 1'b1: Disconnect condition

17.3 HSIC Configuration Registers

Table 17.20 Drive Host Resume Low Register

DRV_HOST_RESUME_0 (OFFSET+0X200 - RESET= 0XE8)			DRIVE HOST RESUME 0 REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DRV_HOST_RESUM[7:0]	R/W	Low byte of a 16-bit register. This register controls the minimum time that the HSIC PHY will drive resume signaling back to the SIE. If the HSIC host drives something shorter than this, then the HSIC PHY will drive the minimum time stored in this register. Units are in 1 us increments

Table 17.21 Drive Host Resume High Register

DRV_HOST_RESUME_1 (OFFSET+0X201 - RESET= 0X03)			DRIVE HOST RESUME 1 REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DRV_HOST_RESUM[15:8]	R/W	High byte of a 16-bit register. This register controls the minimum time that the HSIC PHY will drive resume signaling back to the SIE. If the HSIC host drives something shorter than this, then the HSIC PHY will drive the minimum time stored in this register. Units are in 1 us increments

Table 17.22 Drive Remote Wakeup Low Register

DRV_RMWKUP_0 (OFFSET+0X204- RESET= 0XB8)			DRIVE REMOTE WAKEUP 0 REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DRV_RMWKUP[7:0]	R/W	Low byte of a 16-bit register. This register controls the minimum time the HSIC PHY will drive a remote wake up signal to the HSIC host. Units are in 1 us increments

Table 17.23 Drive Remote Wakeup High Register

DRV_RMWKUP_1 (OFFSET+0X205 - RESET= 0X0B)			**DRIVE REMOTE WAKEUP 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DRV_RMWKUP[15:8]	R/W	High byte of a 16-bit register. This register controls the minimum time the HSIC PHY will drive a remote wake up signaling to the HSIC host. Units are in 1 us increments

Table 17.24 Remote Wakeup Host Drive Low Register

RMWKUP_HOST_DRV_0 (OFFSET+0X208 - RESET= 0XE8)			**REMOTE WAKEUP HOST DRIVE 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	RMWKUP_HOST_DRV[7:0]	R/W	Low byte of a 16-bit register. RMWKUP_HOST_DRV is the minimum time the HSIC PHY will drive a remote wake up signal to the HSIC host. The HSIC PHY will always drive the minimum time allotted by this register. Units are in 1 us increments.

Table 17.25 Remote Wakeup Host Drive High Register

RMWKUP_HOST_DRV_1 (OFFSET+0X209 - RESET= 0X03)			**REMOTE WAKEUP HOST DRIVE 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	RMWKUP_HOST_DRV[15:8]	R/W	High byte of a 16-bit register. RMWKUP_HOST_DRV is the minimum time the HSIC PHY will drive a remote wake up signal to the HSIC host. The HSIC PHY will always drive the minimum time allotted by this register. Units are in 1 us increments.

Table 17.26 TWTDCH Low Register

TWTDCH_0 (OFFSET+0X20C - RESET= 0X58)			**TWTDCH 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TWTDCH[7:0]	R/W	Low byte of a 16-bit register. TWTDCH is the turnaround time from SIE driving a chirp K to the HSIC PHY initiating the chirp KJKJKJ protocol back to the SIE. Refer to the USB 2.0 Specification section 7.1.7.5 for additional information. Units are a period of 60 Mhz.

Table 17.27 TWTDCH High Register

TWTDCH_1 (OFFSET+0X20D - RESET= 0X02)			**TWTDCH 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TWTDCH[15:8]	R/W	High byte of a 16-bit register. TWTDCH is the turnaround time from SIE driving a chirp K to the HSIC PHY initiating the chirp KJKJKJ protocol back to the SIE. Refer to the USB 2.0 Specification section 7.1.7.5 for additional information. Units are a period of 60 Mhz.

Table 17.28 TDCHBIT Low Register

TDCHBIT_0 (OFFSET+0X210 - RESET= 0XB9)			**TDCHBIT 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TDCHBIT[7:0]	R/W	Low byte of a 16-bit register. TDCHBIT is the period of each chirp K or J driven to the SIE by the HSIC PHY. See USB 2.0 Specification section 7.1.7.5 Units are a period of 60 Mhz.

Table 17.29 TDCHBIT High Register

TDCHBIT_1 (OFFSET+0X211 - RESET= 0X0B)			**TDCHBIT 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	TDCHBIT[15:8]	R/W	High byte of a 16-bit register. TDCHBIT is the period of each chirp K or J driven to the SIE by the HSIC PHY. See USB 2.0 Specification section 7.1.7.5 Units are a period of 60 Mhz.

Table 17.30 - USB Reset Filter Timer Low Register

USB_RST_FILT_0 (OFFSET+0X214 - RESET= 0XB4)			**USB RESET FILTER TIMER 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	USB_RST_FILT[7:0]	R/W	Low byte of a 16-bit register. USB_RST_FILT is the filter time for the HSIC RESET bus state. This is used only when entering the HSIC reset state. Units are a period of 60 Mhz.

Table 17.31 USB Reset Filter Timer High Register

USB_RST_FILT_1 (OFFSET+0X215 - RESET= 0X00)			**USB RESET FILTER TIMER 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	USB_RST_FILT[15:8]	R/W	High byte of a 16-bit register. USB_RST_FILT is the filter time for the HSIC RESET bus state. This is used only when entering the HSIC reset state. Units are a period of 60 Mhz.

Table 17.32 USB Resume Filter Timer Low Register

USB_RESUME_FILT_0 (OFFSET+0X218 - RESET= 0XB4)			**USB RESUME FILTER 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	USB_RESUME_ FILT[7:0]	R/W	Low byte of a 16-bit register. USB_RESUME_FILT is the filter time for HSIC RESUME bus state. This is used only when the HSIC host is initiating the resume signaling. Units are a period of 60 Mhz.

Table 17.33 USB Resume Filter Timer High Register

USB_RESUME_FILT_1 (OFFSET+0X219 - RESET= 0X00)			**USB RESUME FILTER 1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	USB_RESUME_ FIL[15:8]	R/W	High byte of a 16-bit register. USB_RESUME_FILT is the filter time for HSIC RESUME bus state. This is used only when the HSIC host is initiating the resume signaling. Units are a period of 60 Mhz.

Table 17.34 IDLE Filter Low Register

IDLE_FILT_0 (OFFSET+0X21C - RESET= 0X60)			**IDLE FILTER 0 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	IDLE_FILT[7:0]	R/W	Bits [7:0] of a 20-bit register. IDLE_FILT is the filter time for the HSIC IDLE bus state. This is used only in a very specific circumstance. After the HSIC PHY is woken up from a suspended state and the clock has stabilized, if the HSIC lines show an IDLE bus state instead of a RESET or RESUME, then the PHY will filter the stored time in the register and go back to suspend. Units are a period of 60 Mhz

Table 17.35 IDLE Filter Mid Register

IDLE_FILT_1 (OFFSET+0X21D - RESET= 0XEA)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	IDLE_FILT[15:8]	R/W	Bits [15:8] of a 20-bit register. IDLE_FILT is the filter time for the HSIC IDLE bus state. This is used only in a very specific circumstance. After the HSIC PHY is woken up from a suspended state and the clock has stabilized, if the HSIC lines show an IDLE bus state instead of a RESET or RESUME, then the PHY will filter the stored time in the register and go back to suspend. Units are a period of 60 Mhz.

Table 17.36 IDLE Filter Mid Register

IDLE_FILT_2 (OFFSET+0X21E - RESET= 0X00)			**REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	IDLE_FILT[19:16]	R/W	Bits [19:16] of a 20-bit register. IDLE_FILT is the filter time for the HSIC IDLE bus state. This is used only in a very specific circumstance. After the HSIC PHY is woken up from a suspended state and the clock has stabilized, if the HSIC lines show an IDLE bus state instead of a RESET or RESUME, then the PHY will filter the stored time in the register and go back to suspend. Units are a period of 60 Mhz.
3:0	Reserved	R	Always read 0

Chapter 18 Flash Media DMA Controller (FMDU)

18.1 Overview

The FMDU control registers resides on the CR_X32 bus. All data transfers happen on the CR_AHB bus. The Flash Media DMA Controller (FMDU) device provides a data transfer interface between the AMBA bus and two Secure Digital (SD). Each controller has a set of software interface registers that are used to control the media devices by Processor.

18.2 FMDU Block Diagram

\

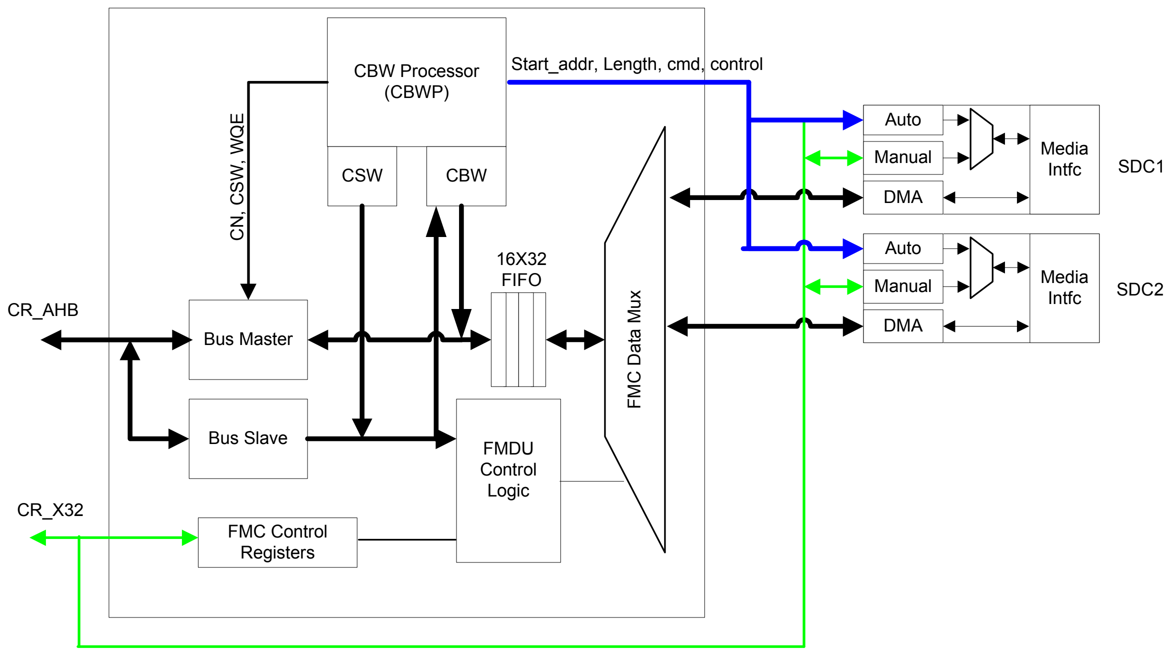


Figure 18.1 Flash Media DMA Controller Block Diagram

18.3 FMDU General description

The main function of the FMDU is to support the transfer of data between RAM and the different type of flash media controllers (FMCs). The direction of the transfer is determined by the command issued on the FMI interface. The FMDU has two modes of operation. In Manual mode operation, the Processor sets up the transfer. It must program the appropriate device with the correct length, starting address etc. The Processor then programs the FMDU mux to point to the FMC for the current access, programs the length in the EP_CNT register, then initiates the transfer by setting the BLK_XFER_EN bit in the EP_CTL register. As work requests come in, the data is transferred between memory and the selected FMC.

The second mode of operation is Auto mode. In this mode the CBW come directly to the FMDU. The FMDU parses the CBW. If the CBW is not valid in any way, the CBW is forwarded to the Processor. If the CBW is valid, but the LUN is not marked as Auto Enabled, the CBW is forwarded to the Processor. If the CBW is valid, and one of the five commands supported by the Auto CBW processor, the CBW is processed. If the CBW is a “Test Unit Ready”, the CBWP checks the ready bit for that LUN, and sends out a response as appropriate. If the command is a Read(10), Read(12), Write (10) Write (12), and the selected LUN is marked as an auto mode device, the FMDU sends the SCSI starting address, length, unit identification, and the command to the FMC selected in the LUN of the

CBW. At the same time the FMDU programs the EP_CNT register with the length of the transfer and sets the BLK_XFER_EN. The selected FMC programs the device(s) attached to it with the starting address, and the length, then signals the FMDU that it is ready for the data transfer. As work requests come in, the data is transferred between memory and the selected FMC.

In auto mode the FMDU simply sends out the SCSI commands as a uniform command on the FMI interface. Read Multiple and Write Multiple are handled by the individual FMCs.

All start addresses sent from the FMDU to the FMCs are in bytes. All LBA lengths sent from the FMDU to the individual FMCs use a 512Byte LBA size. It is up to the individual FMCs to convert the lengths to the native block size of the media in use. The FMC gets the native block size out of the LUN descriptor table. This field is encoded as BLOCK_SIZE in LUN_CTL2.

18.4 Manual CBW processing

In Manual mode operation, the Processor sets up the transfer. The Processor must receive and process the CBW. Once the CBW comes in, the Processor must program the FMI interface with the appropriate length and starting address. It must program the EP_CNT register with the length of the transfer, and then enable the transfer. Once the transfer is complete, the Processor must post the CSW to finish the transaction.

To select an FMC manually, the Processor must select a device using the CBW_COMMAND register.

18.5 Automatic CSW processing

In auto mode, if the AUTO_CSW bit is set then automatic CSW generation is enabled. A pre-programmed CSW will go out if the DMA finishes with no error, and the media write is successful. The pre-programmed CSW is written into the CSW_BUF, and the completion notification goes out by the hardware using the Work Request entry dedicated to the CSW in the WRLUT. If the AUTO_CSW bit is NOT set then automatic CSW generation is disabled. When a transfer finishes with or without an error, the CBW_ONCE_INTR will be asserted to inform the processor of the end of the transaction.

If the AUTO_CSW bit is disabled, the CBW_ONCE_INTR does not indicate a CBW_ONCE transfer completion. The processor must handle the CSW

In manual mode, using CBW once, it is permissible to turn on the AUTO_CSW bit, however it has no effect. The processor must handle CSW.

18.6 Automatic CBW processing

The Automatic CBW processor is endpoint 1 of the FMDU. This distinguishes it from endpoint 0 which is the data endpoint. The CBWP is turned on by setting the AUTO_CBW bit. The automatic CBW processor has the following States:

a. OFF – The automatic CBW processor is turned off.
b. WAIT_CBW – The CBW processor is waiting for CBW
c. PROCESS_CBW – Processing a CBW
d. WAIT_Processor – Wait for the Processor to resolve a CBW
e. WAIT_FMC – Wait for the selected FMC to get ready.
f. DATA_PHASE – In the process of moving data
g. SEND_CSW – Data is done, send the CSW
h. ERROR – Error occurred inform the Processor

18.6.1 CBW Processing -OFF

The CBW processor stays in this state as long as the ENABLE bit in the AUTO_CBW is disabled. In this state the Processor does all CBW processing. The processing of Work Requests for data movement is not affected.

18.6.2 CBW Processing – WAIT_CBW

On entering this state, the CBW processor posts two buffers to SIE EP2 WRITE. The first is to receive the a CBW. The CBWP stays in this state until a CBW arrives or AUTO_CBW is disabled. This state is entered after the CSW state. See example operation. The CBWP stays in this state until the CN arrives for the CBW, in which case it goes to the PROCESS_CBW state.

18.6.3 CBW Processing – PROCESS_CBW

In this state the CBW processor parses the CBW. If any error is detected in the CBW, or if the LUN is not valid, the CBW will be forwarded to the Processor for processing. If the current state of the CBW processor is PROCESS_DATA, or SEND_CSW, and a CBW comes in, the HW must set the PHASE_ERROR bit, and forward the CBW to the Processor. The CBW may be valid, but the protocol is out of synchronization, and Processor intervention is required. If the CBW comes in at the appropriate time, the HW must validate the CBW. If a CBW comes in that will not be handled by the CBWP, that CBW will be forwarded to the Processor, and the CBW Processor will go to the WAIT_Processor state.

Please refer to the USB Mass Storage Class, Bulk Only Transport Revision 1.0 for a description of the fields in the CBW. To validate a CBW, the CBW processor must do the following:

1. Check that the Signature in the CBW matches the CBW_SIGNATURE in the CBW_SIG register. If the Signature does not match, the packet is unknown, set the SIGNATURE_FAIL bit, and forward the CBW to the Processor. The Processor will stall that Endpoint.
2. Check that the CBW length matches the value in the CBW_LENGTH in the CBW_LEN register. If length does not match, the packet is unknown, set the CBW_PKT_LEN_FAIL bit, and forward the CBW to the Processor. The Processor will stall that Endpoint.
3. Check that the bCBWCBLength is equal to the length of the SCSI command. This check is only valid for Read10, Read12, Write10, Write12, or Test Unit Ready. If the length does not match, set the CBWCB_LEN_FAIL bit, and forward the CBW to the Processor. The Processor will stall that Endpoint.
4. If the CHK_RSRV bit is set in the AUTO_CBW registers, check that all the reserved bits in the CBW are ‘0’. The reserved bits are specified in the USB Mass Storage Class specification. If all the bits are not ‘0’ set the RSRV0_FAIL bit, and forward the CBW to the Processor. The Processor will stall that Endpoint.
5. Check that the LUN number matches the MAX_LUN in the MAX_LUN register. If the LUN number is too large, set the MAX_LU_FAIL bit and forward CBW to Processor. If the LUN is not enabled for AutoCBW processing, set UNSUPPORTED_LUN interrupt and forward to the Processor.
6. Now that the Mass Storage portion of the CBW has been validated, Check the SCSI command. If the command is NOT Read10, Read12, Write10, Write12 or Test Unit Ready, pass the CBW to the Processor. If the SCSI command is one of those five commands, then process the SCSI command.

Process the SCSI Command:

1. If dCBWDataTransferLength not equal to SCSI length, then forward the CBW to the processor with the CB_LEN_FAIL bit set. For the actual calculation, the bCBWDataTransferLength is in bytes, while the SCSI length is in blocks. The SCSI length must be multiplied by the LBA.
2. Check that the SCSI command is one of the five supported by the ACBWP. If not, fire the SCSI_ERROR interrupt.

3. Use the LUN in the CBW to index into the LUN_DESCRIPTOR table.
4. If the start LBA address is larger than the Media Capacity in the LUN_DESCRIPTOR table, then set the SCSI_ERROR forward the CBW to the Processor. The compare can be done using block counts or byte counts. If the obsolete Relative address bit is set, set SCSI_ERROR and forward the CBW to the Processor.
5. If the start LBA address plus the length is larger than the Media Capacity, then set the SCSI_ERROR and LEN_FAIL and forward the CBW to the Processor. The compare can be done using block counts or byte count.
6. If the Direction in the SCSI command is opposite to the Direction in the Mass Storage header set the PHASE_ERROR bit. Forward the CBW to the Processor.
7. If the starting address plus length of the transfer exceeds the capacity of the media being addressed, then set the DEV_CAP_FAIL bit and forward to the Processor.
8. Calculate the Physical starting address by adding the LBA starting address to the Offset in the LUN_DESCRIPTOR table.

If the SCSI commands passes, the FMDU processes the command, If it is a read there are two possibilities.

1. The CONT_RD bit is set for this LUN, and if the calculated physical start address matches the internal start address. The next access to the media is started at starting address in the SCSI command. The FMDU command is CONTINUE_READ.
2. The CONT_RD bit is not set for this LUN, or the calculated physical start address does not match the internal start address. The next access to the media is started at new calculated starting address. The FMDU command is NEW_READ.

If the SCSI commands passes, and the command is a write, and the LUN is not write protected, there are two possibilities.

1. The CONT_WR bit is set for this LUN, and the calculated physical starting address matches the internal start address. The next access to the media is started at starting address in the SCSI command. The FMDU command is CONTINUE_WRITE.
2. The CONT_WR bit is not set for this LUN or the calculated physical start address does not matche the internal start address. The next access to the media is started at starting address in the SCSI command. The FMDU command is NEW_WRITE.

If the LUN is write protected, the WRITE_PROT interrupt is generated to the Processor.

To execute the read or write command, the CBWP sets the FMC_DEV_SEL to the LUN addressed in the CBW. It sets the EP_CNT register with the length, and send out an encoded version of the SCSI command on the FMI bus. The encoded version of the command will have at least the following information:

1. Start Address in the media. This is the calculated physical start address. This is always 32 bit logical block address, which is equivalent to a 41 byte address. It is the same regardless of the size in the SCSI command.
2. Length of transfer.
3. Block Size (encoded)
4. Command (NEW_READ, CONT_READ, NEW_WRITE, CONT_WRITE)
5. Identifier for the FMC selected.
6. ID of the unit within the controller. This is only valid for FMCs that have multiple devices.
7. Device Type (only has meaning to certain controllers)

8. Timing and control signals. (CONT_WR, CONT_RD etc)
9. Error Signals

If the SCSI commands passes, and the command is a Test Unit Ready, the hardware copies the TAG and the LUN READY bit into the CSW and goes directly to SEND_CSW stage.

Once the CBW is processed, the CBW processor is free to use that buffer again. If the CBW is forwarded to the Processor, the CBWP relies on the Processor to generate the appropriate completion notification.

If the CBW command is a READ, then it means there is a buffer trapped at SIE EP2 WRITE work queue. In that case the CBWP must issue a FLUSH to the SIE to release that buffer.

18.6.4 CBW Processing – WAIT_Processor

This state is entered when the CBW has been forwarded to the Processor. The Processor has two ways of dealing with this. The first is that the Processor will disable auto CBW processing. When this happens, the CBWP is disabled, and when it is re-enabled, the state machine starts over. The second way is the RESUME bit is hit, and the CBWP goes to the WAIT_FMC state assuming that the Processor has programmed the LUN and FMDU registers appropriately.

The AutoCBW processor normally issues two buffer to SIE EP2 WRITE. The first buffer will receive the CBW, which it will forward to the host in this state. The second buffer may or may not be used. If the USB host does not post another OUT, the second buffer will be pending on the queue of SIE EP2 WRITE. The Processor can clear this buffer with a flush if it desires.

If the USB host posts an OUT after the CBW, the second buffer will be used, and will be posted on the FMDU work queue, waiting to be processed. If the Processor is very quick to respond to the forwarded CBW, it may be possible that the second buffer is still on the SIE work queue, when the Processor gets the CBW. The Processor is then responsible for waiting until the buffer is correctly issued.

18.6.5 CBW Processing – WAIT_FMC

After posting the encoded SCSI command, the CBW processor stays in this state waiting for the FMC to signal it is ready. This is a widely variable time. The FMC may start immediately, or it may take hundreds of microseconds.

18.6.6 CBW Processing – DATA_PHASE

This state is entered when the FMC signals it ready for the data transfer. In this state the CBW processor reads or writes blocks of data. If the transfer is a read (USB IN) the CBWP assigns buffers from it WRLUT pool. As completions come in, more data is read and enqueued. If the transfer is a write, the data is written to the media as it comes in with the Work Request. This continues until the EP_CNT count has been reached. If the EP_CNT count ends in the middle of a buffer, the HW is responsible for send out the last Completion Notification with the LAST_BUFFER bit set. If in the middle of the data phase, a CBW comes in, the HW must set the PHASE_ERROR bit, and send the packet to the Processor. At the start of the DATA_PHASE, the HW knows the length of the transfer. If a short packet arrives before that length is met, then an error has occurred. The HW sets the DATA_ERROR bit and interrupts the Processor.

The HW must be able to distinguish between a data packet that coincidentally has the CBW signature in the first DWORD and real CBW.

18.6.7 CBW Processing – SEND_CSW

In this state the CBW processor constructs the CSW. The CSW is stored in a buffer that is accessible from the AHB bus. For writes to the media the CBW processor must wait until the write is acknowledged by the media.

The CSW buffer contain the following 13 bytes:

1. 0~3 SIGNATURE of CSW (4Bytes)
2. 4~7TAG of the CSW copied from CBW (4Bytes)
3. 8~11RESIDUE, always 0 unless there is an error (4bytes)
4. 12Status (1 byte).

The CBWP then sends out a Work Queue Element that references that Work Request that points to that CSW in its internal buffer. This is done by referencing the Work request that is dedicated to the CSW.

After sending the completion notification for the CSW, to the SIE, wait for notification that the CSW has been sent to the host, that is wait for the CN from the SIE. This allows the hardware to detect a phase error if the CSW is lost in transmission.

18.6.8 CBW Processing – ERROR

This state is entered if there was an error during CBW processing. In this state the CBW processor wait for the Processor to reset it.

18.6.9 CBW Forwarding

To forward a CBW to the Processor, the CBWP writes the received context of the CBW it failed to process to the address in the CBW_FRWRD_ADDR register. The Processor can use this context to lookup the Work request in the FMDU WRLUT.

18.7 ACBW WRLUT buffer allocation

When using the ACBW processor, the WRLUTs must be programmed in exactly the way the ACBW is expectiing. The table below shows how the entiries must be programmed. This scheme only applies when using the ACBW processor, and need not be followed for manual DMA.

The WRLUT entry number 5 is an aliased entry, to make sure that two buffers are posted at the SIE to avoid generating NYETs.

Table 18.1 ACBW WRLUT Usage

ENTRY	USAGE	DATA REF	CN ADDR	CONTEXT
0	CBW Buffer	Don't Care	SIE_EP2	Point to SIE's CBW Buffer
1	Data Buffer A	BUF_A	SIE_EP2	Point to SIE's Data Buffer A
2	Data Buffer B	BUF_B	SIE_EP2	Point to SIE's Data Buffer B
3	Data Buffer C	BUF_C	SIE_EP2	Point to SIE's Data Buffer C
4	CSW Buffer	Don't Care	SIE_EP2	Point to SIE's CSW Buffer
5	Alias to Data Buffer A	Don't Care	SIE_EP2	Alias pointer to SIEs Data Buffer A (required for correct operation)

18.8 FMDU Registers

18.8.1 FMDU CR_AHB Registers

The FMDU is at base address CR_AHB+F400. The FMDU has 1 read endpoints and 1 write endpoints (in fact each endpoint is implemented a single bidirectional endpoint). The work queue addresses for the FMDU is shown in the following table.

Table 18.2 FMDU Work Queue Addresses

EP NUMBER	READ	WRITE	WQ DEPTH
0	CR_AHB+F500	CR_AHB+F580	4
1	CR_AHB+F508	CR_AHB+F588	1

EP1 is the control endpoint used by the ACBW.

The FMDU has two buffers that are accessible from the AHB bus. They are a 32 byte buffer for the the CBW, and a 13 byte buffer for the CSW.

Table 18.3 FMDU Register Summary

ADDRESS	NAME	DESCRIPTION	R/W	DEFAULT
CR_AHB+F400 ~ CR_AHB+F41F	CBW_BUF	CBW Buffer	R/W	0x00
CR_AHB+F420 ~ CR_AHB+F42C	CSW_BUF	CSW Buffer	R	0x00

18.8.2 Work Request Look Up Table (WRLUT)

The FMDU has 6 Work Request entries in it Work Request Look up table. The entries must be used as specified by table below.

Table 18.4 FMDU WRLUT

OFFSET ADDRESS CRX_32+	CONTEXT	WORK REQUEST #	USAGE
0x1100	0	Work Request 0	CBW Buffer
0x1110	1	Work Request 1	Data Buffer
0x1120	2	Work Request 2	Data Buffer
0x1130	3	Work Request 3	Data Buffer
0x1140	4	Work Request 4	CSW Work
0x1150	5	Work Request 5	Alias Data buffer used by ACBW

It is important to note that when in Auto CBW processing mode, the CBWP must override the completion address in the Work request. The CBWP must modify the completion address by setting and clearing bit 7 as appropriate. The reason for doing this is to avoid the Processor having to come in and change the notification address when the data transfer direction is switched for writes to reads.

18.8.3 Endpoint Control Registers

The FDMU has one channel, that is two endpoints: EP0 Read, EP0 Write. Both are DMA Endpoints.

The base address of all the endpoints are according to the table below:

Table 18.5 FMDU Endpoints Table

ADDRESS CRX_32+	NAME	TYPE
0x1300	EP0_READ	DMA
0x1380	EP0_WRITE	DMA

The interrupts generated EP0_READ and EP0_WRITE are routed to FMDU_EP0_RD_INTR and FMDU_EP0_WR_INTR in the interrupt controller. The EP1 is the control endpoint used by the CBWP, and has no Endpoint Control Registers.

18.8.4 Auto CBW Processing Registers

Table 18.6 Automatic CBW Processing Control Register

AUTO_CBW_CTL (0X1000 - RESET=0X00)			AUTOMATIC CBW PROCESSING REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	AUTO_CBW	R/W	Enable Automatic CBW processing. If none of the LUNs are enabled for automatic CBW processing, this bit should be turned off. If this bit is turned off in the middle of a transaction, it has no effect until the AutoCBW processor gets to the IDLE state, after it sends out a CSW.
6	AUTO_CSW	R/W	Enable Automatic CSW processing. If none of the LUNs are enabled for automatic CSW processing, this bit should be turned off.
5	CBW_ONCE	R/W	This is a self clearing bit. Set by the Processor when it wants to process a single CBW. That CBW must be loaded into the CBW_BUF before this bit is set.
4	RESUME	R/W	This is a self clearing bit. If set to ‘1’ by the Processor, it forces the CBWP to resume at the WAIT_FMC state.
3	XFER_ACTIVE	R	This bit is set by the hardware to indicate that a transfer is currently in progress. Firmware should not make changes in registers while this bit is active. The only exception is to clear the AUTO_CBW bit.
2	TAIL_MODE	R/W	Enable the Tail request interrupt generation
1	CHK_RSRV	R/W	If this bit is ‘1’, then check that all reserved bits in the CBW are ‘0’
0	RESET	R/W	Writing a ‘1’ to this bit clears the Auto CBW processing logic. The bit will self clear when the reset is complete. It will also clear the FRAME_VALID_AUTO signal to the AES block if set.

18.8.5 CBW_ONCE usage

This bit is used by the Processor to allow it to take advantage of the Auto CBW processor. In this mode the AUTO_CBW is not set. The processor constructs a CBW, and loads it into the CBW_BUF. It sets the CBW_ONCE bit, to kick off the processing. The auto cbw processor treats this like any other CBW with the exception that at the end, it stops. When it done with the transaction it clears the CBW_ONCE bit. If the Processor attempts to write another CBW before the CBW_ONCE bit is cleared, unpredictable results can happen.

At the end of the transfer the CBW_ONCE_INTR is asserted. The interrupt indicates the hardware is ready to accept the next CBW.

18.8.6 XFER_ACTIVE usage

This bit is used by as an indication to the processor that a transfer is in progress. This bit is also used to drive the Card Reader ACTIVITY LED. See the LED0_GPIO0 control and LED1_GPIO1 control registers for details.

18.8.7 TAIL_MODE usage

SEC2410/SEC4410 does not support the tail mode.

This bit is set by Processor when working with unmapped media. To optimize throughput, the copy head / copy tail sequence is only used when an out of sequence transfer has to be done. When an out of sequence transfer is required, the CBWP signals the Processor to do the copy tail before proceeding with a new transfer. The interrupt is set when one of the following conditions are met:

1. *Timeout: 1 ms has passed with no traffic to the last media accessed.*
2. *The latest transfer is not a continuation of the previous transfer.*
3. *An error occurred during the transfer.*
4. *An error occurred in the new CBW. There will be a tail mode interrupt. AFTER the tail mode interrupt is cleared, the particular CBW error interrupt will file.*

The tail once bit acts as a stall command to the Auto CBWP. Once the Processor does the copy tail using the FMI interface, it clears the TAIL_MODE bit, and the Auto CBWP continues processing.

Table 18.7 Automatic CBW Processing Control Register 2

AUTO_CBW_CTL_2 (0X1001 - RESET=0X04)			**AUTOMATIC CBW PROCESSING REGISTER 2**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always read ‘0’
2	DATA_CONCAT	R/W	0: No data concatenation/separation is performed. 1: Data Concatenation/Separation mode: Treats data between two buffers as consecutive if these have the same direction and Endpoint number. Default value is ‘1’
1	ABORT_AES	R/W	This is a self clearing bit. When set it will terminate the current AES frame.
0	DISABLE_D0	R/W	Writing a ‘1’ to this bit forces the Auto CBW processor not to issue a DATA0 buffer. This reduces throughput by causing NYETs on the USB bus. It should only when AES encryption/Decryption is used.

18.8.8 Disable_D0 Usage

The ACBW processor tries to keep two buffers posted at the SIE Write (OUT) endpoint at all times. This is done to avoid the SIE issuing NYETs to the host when only one buffer is available. When the host sees NYETs it will throttle back the bandwidth allocated to the device. This has a long term penalty associated with it.

Issuing two buffers has the following potential race condition. If the host issues an OUT CBW, followed immediate by the data, then the decision to encrypt or not encrypt the data has to be made after the complete arrival of the CBW, and before the arrival of the data. This is not a problem for the hardware. However if fiirmware intervention is required in the making of the decision, then the DISABLE_D0 bit must be set. This bit forces only one buffer to be issued. When only one buffer is issued, the firmware can hold off issuing the data buffer until it has completed processing of the CBW.

18.8.9 Data0 Timing constraints with AES

When AES is used, it is a requirement that the parameters required by the AES block be provided within 300ns of receipt of the CBW. The timing constraint comes from the fact that on USB OUTs, the first Data packet will follow the CBW almost immediately. The gaps in the packet timing come from the USB 2.0 specification 7.1.18.2. The first constraint is that if a host is transmitting two packets in a row, the inter-packet gap must be at least 88 bit time. The second constraint is that when transmitting after receiving a packet, host and devices must provide an inter-packet gap of at least 8 bit times.

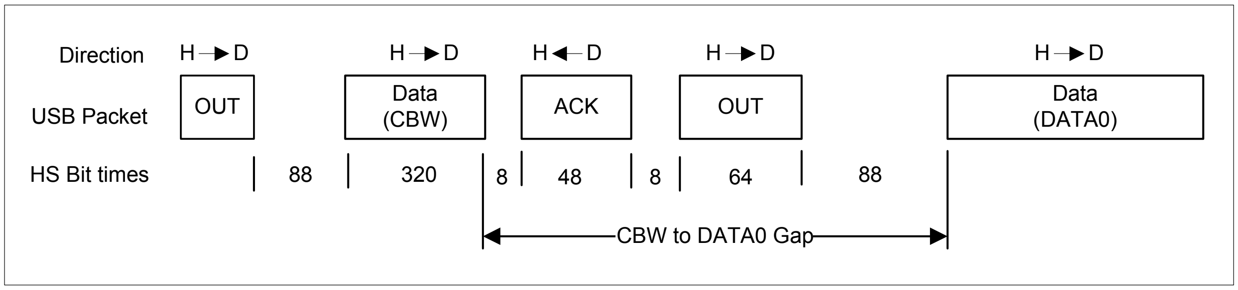


Figure 18.2 Data0 Timing Constraint

The total CBW to Data0 gap is 216 HS bit times which is 450 ns.

Table 18.8 CBW Failure Mode Register

AUTO_CBW_STAT (0X1008 - RESET=0X00)			CARD STATUS REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	PHASE_ERROR	R/W	Protocol Phase error.
6	DATA_ERROR	R/W	Protocol Error in DATA_PHASE
5	SCSI_ERROR	R/W	SCSI command error
4	MAX_LUN_FAIL	R/W	The LUN ID is greater than the max LUN ID.
3	RSRV0_FAIL	R/W	All the Reserved bits are not zero.
2	CB_LEN_FAIL	R/W	CBW dCBWDataTransferLength did not match the SCSI length

AUTO_CBW_STAT (0X1008 - RESET=0X00)			**CARD STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
1	WRITE_PROT	R/W	If set, it indicates that an attempt was made to write to a device that was write protected.
0	SIGNATURE_FAIL	R/W	The Signature does not match

Note 18.3 These interrupts are only enabled when Auto CBW processing is enabled.

Note 18.4 Write ‘1’ to clear any interrupt bit. A write of ‘0’ has no effect.

Note 18.5 A one in any bit will generate a AUTO_CBW_INTR interrupt in ther interrupt controller.

Table 18.9 CBW Stat2 Mode Register

CBW_STAT2 (0X100A - RESET=0X00)			**CARD STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	R	Always read ‘0’
6	TAIL_REQUEST	R/W	This interrupt occurs when the Auto CBW needs the Processor to complete the COPY TAIL portion of an unmapped media.
5	UNSUPPORTED_LUN	R/W	This interrupt occurs when a CBW comes in for a LUN for which the AutoCBW processor has not been enabled.
4	CBWCB_LEN_FAIL	R/W	The CBWCB length is different from the length of the SCSI command. This field is only valid for the commands handled by the AutoCBW processor.(Read10, Read12, Write10, Write12, TestUnitReady.)
3	DEV_CAP_FAIL	R/W	The starting address plus the length of the transfer exceeds the capacity of the logical unit being addressed.
2	CBW_IDLE	R/W	This interrupt is set if the AutoCBW processor is IDLE and the AUTO_CBW bit is off. This interrupt cannot be used as a signal back to the Processor that the AutoCBW processor is done with the current transfer and halted.
1	CBW_PKTLEN_FAIL	R/W	The CBW length does not match programmed in the CBW_LEN register.
0	CBW_ONCE_INTR	R/W	This bit is set at the end of a CBW_ONCE transaction. , or end of an AUTO_CBW transaction that has the AUTO_CSW=0.

Note 18.6 A one in any bit will generate a interrupt in the interrupt controller.

Note 18.7 Write ‘1’ to clear any interrupt bit. A write of ‘0’ has no effect.

Table 18.10 CBW Stat3 Mode Register

AUTO_CBW_STAT3 (0X100C - RESET=0X00)			**CARD STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’
1	B1P_RECEIVE	R/W	This bit is set when the Auto CBWP is active and has received the first data buffer.
0	XFER_DONE	R/W	This bit is set after the Auto CBWP has completed a CSW successfully.

Note 18.8 Write ‘1’ to clear any interrupt bit. A write of ‘0’ has no effect.

Table 18.11 Work Request Length

WR_LENGTH (0X1016~0X1017 - RESET=0X0000)			**CARD STATUS MASK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15;12	Reserved	R	Always read ‘0’
11:0	LENGTH[11:0]	R/W	Length of the first data buffer sent by the USB host received after a new CBW.

18.8.10 Work request length

The Auto CBWP always issues 2 buffers to the SIE. One buffer is for the CBW and the other is for the data. For the cases which the Processor is forwarded the CBW for processing, the Processor sometimes needs to process the data that comes after the CBW. The WR_LENGTH registers holds the length of the first data buffer after the CBW.

Table 18.12 CBW Buffer

CBW_BUF (CR_AHB+F400~F42F - RESET=0X00)			**CBW BUFFER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
0:31	CBW_BUF[0:31]	R/W	32 Byte buffer. This buffer is used to store the incoming CBW.

The CBW buffer is accessible from the AHB bus. This allows other devices to directly send a CBW to the CBW processor without necessarily going through the AMBA AHB RAM. It also allows the Processor to access the CBW in the case of an exception.

Table 18.13 CSW Buffer

CSW_BUF (CR_AHB+F420 ~ F42C - RESET=0X00)			**CSW BUFFER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
0:12	CSW_BUF[0:12]	R	13 Byte buffer. This buffer is used to store the 13 byte outgoing CSW. This buffer is written to by the HW and and can only be read by the firmware.

The CSW buffer is accessible from the AHB bus. This allows other devices to directly read the CSW without necessarily going through the AMBA AHB RAM.

Table 18.14 CBW Signature Register

CBW_SIG (0X1010 RESET=0X00000000)			**CBW SIGNATURE REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	CBW_SIGNATURE	R/W	32 bit value loaded by the Processor to identify a CBW.

Table 18.15 CBW Length Register

CBW_LEN (0X1003 - RESET=0X00)			**CBW LENGTH REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Reserved, always read ‘0’
5:0	CBW_LENGTH	R/W	6 bit value loaded by the Processor to identify a CBW.

Table 18.16 CBW MAX LUN Register

MAX_LUN (0X1002 - RESET=0X00)			**MAXIMUM LUN REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Reserved, always read ‘0’
3:0	MAX_LUN	R/W	The Processor programs this register with the maximum number of LUNs configured. If the LUN ID requested by the host, is greater than this number, then an error has occurred.

Table 18.17 CBW Forwarding Address Register

CBW_FRWRD_ADDR (0X1018~0X101B - RESET=0X00000000)			**FORWARD CBW ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:24	Reserved	R	Always read ‘0’

CBW_FRWRD_ADDR (0X1018~0X101B - RESET=0X00000000)			FORWARD CBW ADDRESS REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
23:0	CBW_FRWD_ADDR	R/W	This is the address to which the CBWP writes the context of the CBW it failed to process.

Table 18.18 Error Context Register

ERROR_CXT (0X1005 - RESET=0XFF)			ERROR CONTEXT REGISTER
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	R	Always read '1'
6:0	CONTEXT	R/W	This register is loaded with the context of the current buffer being used in the DMA when there is any kind of error.

Note 18.9 This context is used to identify the buffer with an error.

18.8.11 Work request Status bits

There are two bits associate with each work request used by the Auto CBWP. The table below shows their usage.

Table 18.19 WR_STAT descripiton

WR_STAT	DESCRIPTION
00	Free: This means the AutoCBW processor has control of the associated buffer. The buffer has not been assigned to a function yet. The firmware writes 00 to WR_STAT to enable a buffer to be used by the Auto CBW processor..
01	In Use External: This means tha the Auto CBW processor is in control of the buffer, and it has issued the buffer to the SIE or processor.
10	In Use Internal: This means tha the Auto CBW processor is in control of the buffer, and it buffer is in the FMDU being processed.
11	Disabled. This means that the Auto CBW processor may not use the associated buffer. The Firmware is free to use the buffer. The firmware writes 11 to WR_STAT to disable a buffer from being used by the Auto CBW processor..

Table 18.20 WR Status Register0

WRK_REQ_STATUS0 (0X101C - RESET=0X00)			WORK REQUEST STATUS REGISTER (CBW)
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’.
1:0	WR_STAT	R/W	00: Free. ACBWP has control. 01: In use. Currently inside an EP outside the FMDU. 10: In use. Currently inside the FMDU EP0. 11: Disabled. F/W has control.

Table 18.21 WR Status Register1

WRK_REQ_STATUS1 (0X101D - RESET=0X00)			WORK REQUEST STATUS REGISTER (DATA0)
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’.
1:0	WR_STAT	R/W	00: Free. ACBWP has control. 01: In use. Currently inside an EP outside the FMDU. 10: In use. Currently inside the FMDU EP0. 11: Disabled. F/W has control.

Table 18.22 WR Status Register2

WRK_REQ_STATUS2 (0X101E - RESET=0X00)			WORK REQUEST STATUS REGISTER (DATA2)
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’.
1:0	WR_STAT	R/W	00: Free. ACBWP has control. 01: In use. Currently inside an EP outside the FMDU. 10: In use. Currently inside the FMDU EP0. 11: Disabled. F/W has control.

Table 18.23 WR Status Register3

WRK_REQ_STATUS3 (0X101F - RESET=0X00)			WORK REQUEST STATUS REGISTER (DATA3)
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’.
1:0	WR_STAT	R/W	00: Free. ACBWP has control. 01: In use. Currently inside an EP outside the FMDU. 10: In use. Currently inside the FMDU EP0. 11: Disabled. F/W has control.

18.9 LUN Descriptor Table

The LUN Descriptor Table is a table to describe eight LUNs. The Processor interrogates the devices that are attached, and fills the table. The table gets modified whenever a device is either attached or detached. When a CBW comes in, the LUN becomes an index into the LUN descriptor table. The Descriptor that is retrieved contains the information required to process that CBW.

The contents in the LUN Descriptor Table is only utilized by the CBW. The CBWP puts information on the FMI interface based on the contents in the LUN Descriptor Table and a received CBW. This is done by overwriting the FMI Registers with the information above.

For manual mode operation, the Processor should set up FMI Registers and other necessary registers manually. The LUN descriptors have no effect.

Table 18.24 LUN Descriptor Table

LUN	ADDRESS	DESCRIPTOR
0	0x0x40001200	LUN Descriptor 0
1	+0x10	LUN Descriptor 1
2	+0x20	LUN Descriptor 2
3	+0x30	LUN Descriptor 3
4	+0x40	LUN Descriptor 4
5	+0x50	LUN Descriptor 5
6	+0x60	LUN Descriptor 6
7	+0x70	LUN Descriptor 7

18.10 LUN Descriptor

A Lun Descriptor describes a LUN. Each descriptor is eight bytes long. There are eighth descriptors in the FMDU. Any descriptor can be assigned to any controller, or any partition in any controller.

Table 18.25 LUN Descriptor

ADDRESS	SIZE	DESCRIPTOR
0	8 bit	LUN_CTL1
1	8 bit	LUN_CTL2
2~3	16-Bit	LUN_TIMEOUT
4~7	32 bit	LUN_CAPACITY
8~B	32 bit	LUN_OFFSET
C	8 bits	LUN_CTL3
D~F	24 bit	Reserved

Table 18.26 LUN Control Descriptor

LUN_CTL1 (RESET=0X00)			**LUN CONTROL1 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	AUTO_CBW_ENABLE	R/W	Enable Auto CBW processing for this LUN.
6	AUTO_CSW_ENABLE	R/W	Enable auto CSW processing for this LUN
5	LUN_READY	R/W	Set to ‘1’ by Processor when this LUN has been initialized and is ready for operation. Set to ‘0’ by Processor when this LUN is no longer in ready state for any reason Hardware uses this bit to respond to Test Unit Ready commands
4	CONT_RD	R/W	Set to one by Processor when this LUN is capable of continuous Read operation.
3	CONT_WR	R/W	Set to one by Processor when this LUN is capable of continuous Write operation.
2:0	DEVICE	R/W	Physical Device Selector 000 - Reserved 001 - Reserved 010 - Reserved 011 - Secure Digital Controller1 100 - Secure Digital Controller2 All others reserved.

Table 18.27 LUN Control2 Descriptor

LUN_CTL2 (RESET=0X00)			**CARD CONTROL2 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	WRITE_PROT	R/W	When set to one, the LUN is write protected. Any attempt to write to the device will result in a an interrupt being generated to the Processor.
6:4	DEV_TYPE	R/W	This is an enumeration of the different device type supported by the controller. If there are multiple control paths for the state machines inside the card reader, this allows the SW to select the control path appropriate to the state machine.
3:1	BLOCK_SIZE	R/W	These bits are set by the Processor to indicate the block size of the LUN. 000 – 256 Bytes 001 – 512 Bytes 010 – 1024 Bytes 011 – 2048 Bytes 100 – 4096 Bytes All others reserved.
0	ID	R/W	This bit is used to identify the unit within a controller.

Note: The BLOCK_SIZE is the block size of the LUN addressed. It should not be confused with the SCSI 512 byte sector size in the CBW. This BLOCK_SIZE is used to break the transfer down to the size that is best suited to the media.

Table 18.28 LUN Timeout Register

LUN_TIMEOUT (RESET=0X0000)			**TIMEOUT REGISTER FOR FMDU**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	TIMEOUT	R/W	This register holds the timeout to be used for FMDU operations for that particular LUN. The timer starts when the DMA starts, either in Manual or Automatic mode. Every time a packet arrives, the timer is restarted. If the DMA does not complete before the timer expires a LUN_TIMEOUT is generated. Note this does not deal with the case where packets arrive very slowly. This timer is to detect hardware failures. Whenever there is a timeout, the TIMEOUT_ERR bit is set. Each count represents 100uS.

The combination of the LUN capacity plus LUN offset make a virtual LUN. That means a single physical device may be partitioned into as many as eight logical devices.

Table 18.29 LUN Capacity Register

LUN_CAPACITY (RESET=0X00000000)			**LUN CAPACITY REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	CAPACITY	R/W	This register holds the Capacity of the LUN in terms of SCSI blocks. This register is used to check that the starting and ending addresses in the CBW are within the device.

Table 18.30 LUN Offset Register

LUN_OFFSET (RESET=0X00000000)			**LUN OFFSET REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	OFFSET	R/W	This register holds the Offset value for the LUN. The unit is LBAs. This allows the firmware to assign multiple LUNs to the same physical device. The value in this register is applied to the starting address value in the CBW.

Table 18.31 LUN Control Descriptor

LUN_CTL3 (RESET=0X00)			**LUN CONTROL3 REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read ‘0’
3	REVERSE_MODE	R/W	If this bit is ‘0’ then Encrypt when writing to media, and Decrypt when reading from media. If this bit is ‘1’ then Decrypt when writing to media, and Encrypt when reading from media.
2:0	AES_KEY_SEL	R/W	These bits determine which AES Key will be used with this LUN. This value is always forwarded to the AES block. 000b – No Encryption / Decryption to be used with this LUN 001b – Use Key number 1 010b – Use Key number 2 011b – Use Key number 3 100b – Use Key number 4 101b – Use Key number 5 110b – Use Key number 6 111b – Use Key number 7

18.11 FMDU Control Registers

Table 18.32 FMC Mode Control Register

FMC_DEV_SEL (0x1004 **- RESET=0X00)**			**FMC MODE CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always read ‘0’
2:0	SELECT	R/W	000b: LUN0 is selected. 001b: LUN1 is selected. 010b: LUN2 is selected. 011b: LUN3 is selected. 100b: LUN4 is selected. 101b: LUN5 is selected. 110b: LUN6 is selected. 111b: LUN7 is selected. The default setting is 000b

When the FMC_DEV_SEL is programmed with the LUN, the media controller in the LUN_CTL1 register is connected to the FMDU. The information in the LUN descriptor for that LUN is available to the media controller being addressed. i.e. When this register is set, the block SIE, device type, write protect etc, is available to the media controller.

In manual operation the Processor writes to this register. This must occur with every CBW, even if the register is already pointing to the correct LUN. The hardware uses the write to this register to initiate the transaction.

In auto mode, the hardware sets this register based on the LUN ID in the CBW. The auto CBW processor must provide a strobe to indicate that the data on the command bus is valid. It is the responsibility of the individual media controllers to recognize when they are being accessed.

Table 18.33 Endpoint Interrupt Register

FMC_EP_INTR (0x100E - RESET=0X0000)			**ENDPOINT INTERRUPT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read ‘0’
1	EP0_WR	R	EP0 Write Endpoint interrupt. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.
0	EP0_RD	R	EP0 Read Endpoint interrupt. Set if any of the unmasked interrupts are set within the endpoint. This bit must be cleared by clearing the condition in the endpoint.

Table 18.34 LUN Timeout Interrupt Register

LUN_TIMOUT_INTR (0x1006 - RESET=0X00)			**ENDPOINT INTERRUPT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:1	Reserved	R	Always read ‘0’
0	LUN_TIMEOUT	R/W	A timeout occurred on active LUN Write ‘1’ to clear the bit. Write ‘0’ has no effect.

Note: A ‘1’ in any of these bits causes an interrupt on LUN_TIMEOUT line in the interrupt controller

18.12 Flash Media Interface (FMI) Registers

The Flash Media Interface (FMI) is the common interface to the FMCs. This interface is accessible to both the Processor and the Auto CBW processor. When the Auto CBW processor is turned off, the Processor can directly access the Auto CBW interface to the FMCs. This allows the Processor to mimic the behavior of the Auto CBW processor.

The ACBWP will over write these registers when it is operating. Writing to these registers while the Auto CBW processor is operating, will causes errors.

Table 18.35 FMI Command Register

CBW_COMMAND **(**0x1020 **- RESET=0X00)**			**CBW COMMAND REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	CMD_VALID	R/W	Set by the Processor to indicate that it has loaded a valid command into the CBW interface registers. This bit is cleared by the HW after the current command has been accepted. In auto mode this bit is controlled by the CBWP.
6	Reserved	R	Alway read ‘0’
5:3	DEV_SELECT	R/W	Physical Device Selector 000 - Reserved 001 - Reserved 010 - Reserved 011 - Secure Digital Controller1 100 - Secure Digital Controller2 All others reserved. In auto mode, this is overwritten with DEVICE field of LUN_CTL register of active LUN by the CBWP.
2:0	COMMAND	R/W	The commands are: 000 – New Read 001 – Continue Read 010 – New Write 011 – Continue Write 100 – Manual Read 101 – Manual Write All others reserved. In auto mode, this is overwritten by the CBWP.

Note: All the Direct interface registers must be programmed before writing to the command register to initiate a transfer.

Table 18.36 FMI Options Register

CBW_OPTIONS (0x1021 **- RESET=0X00)**			**CBW OPTIONS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	Reserved	R	Always read ‘0’
6:4	BLOCK_SIZE	R/W	This is the block size for the transfer 000 – 256 Bytes 001 – 512 Bytes 010 – 1024 Bytes 011 – 2048 Bytes 100 – 4096 Bytes All others reserved. In auto mode, this is overwritten with BLOCK_SIZE field of LUN_CTL2 register of active LUN by the CBWP.
3	DEV_ID	R/W	This bit is used to identify the unit within a controller. In auto mode, this is overwritten with ID field of LUN_CTL2 register of active LUN by the CBWP
2:0	DEV_OPTIONS	R/W	This is an enumeration of the different device type supported by the controller. If there are multiple control paths for the state machines inside the card reader, this allows the SW to select the control path appropriate to the state machine. In auto mode, this is overwritten with DEV_TYPE field of LUN_CTL2 register of active LUN by the CBWP

Table 18.37 FMI Transfer Length Register

CBW_TRANS_LEN (0X102C~0X102F - RESET=0X00000000)			**CBW TRANSFER LENGTH REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:0	TRANS_LENGTH	R/W	32 bit value loaded by the Processor to control the length of a transfer. The length is in bytes. The lower bits will be zero for sector aligned transfers. **Note:** In auto mode, this is overwritten with the CBW transfer length by the CBWP.

Table 18.38 FMI CBW Start Address Register

CBW_START_ADDR (0X1024~0X1029 - RESET=0X00000000)			**CBW START ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
40:0	START_ADDR	R/W	40 bit value loaded by the Processor to identify to control the length of a transfer. The address is in bytes. The lower bits will be zero for sector aligned transfers. ADDR Bits 0x1027 - bits 40~33 0x1026 - bits 32~25 0x1025 - bits 24~17 0x1024 - bits 16~9 0x1029 - bits 8~1 0x1028 - bit 0 **Note:** Bit 0 of the starting address appears as bit 7 of location 0x1028 **Note:** In auto mode, this is overwritten with the SCSI LBA by the CBWP.

Chapter 19 Secured Digital (SD/MMC) Controller

19.1 Overview

SEC2410/SEC4410 has two SD controllers (SDC). The SDC of SEC2410/SEC4410 has three interfaces. The first is the Manual Command Interface (MCI). This is over the CR_AHB bus. The Processor gains access to the internal control registers through this interface. The SDC decodes any access cycles to SDC interface I/O address in the CR_AHB space. The SDC acts as a functional bridge agent capturing the requests from the Processor. Then the SDC drives the SD bus interface to deliver the commands, address or data to the SD device.

The second interface is the FMDU Command Interface (FMI). This interface comes from the FMDU. This interface is controlled by the Auto CBW generator of the FMDU. The main information passed over this interface is the Start Address, Length of access, Command, and control signals. In auto mode, the SDC is responsible for programming the media during a transaction, and for informing the FMDU when it is ready or busy.

The third interface is the Data interface. The data movement is over this interface. The usage of this interface is the same whether the Manual Command Interface or the FMDU Command Interface is used.

The control registers are located on the CR_X32 bus.

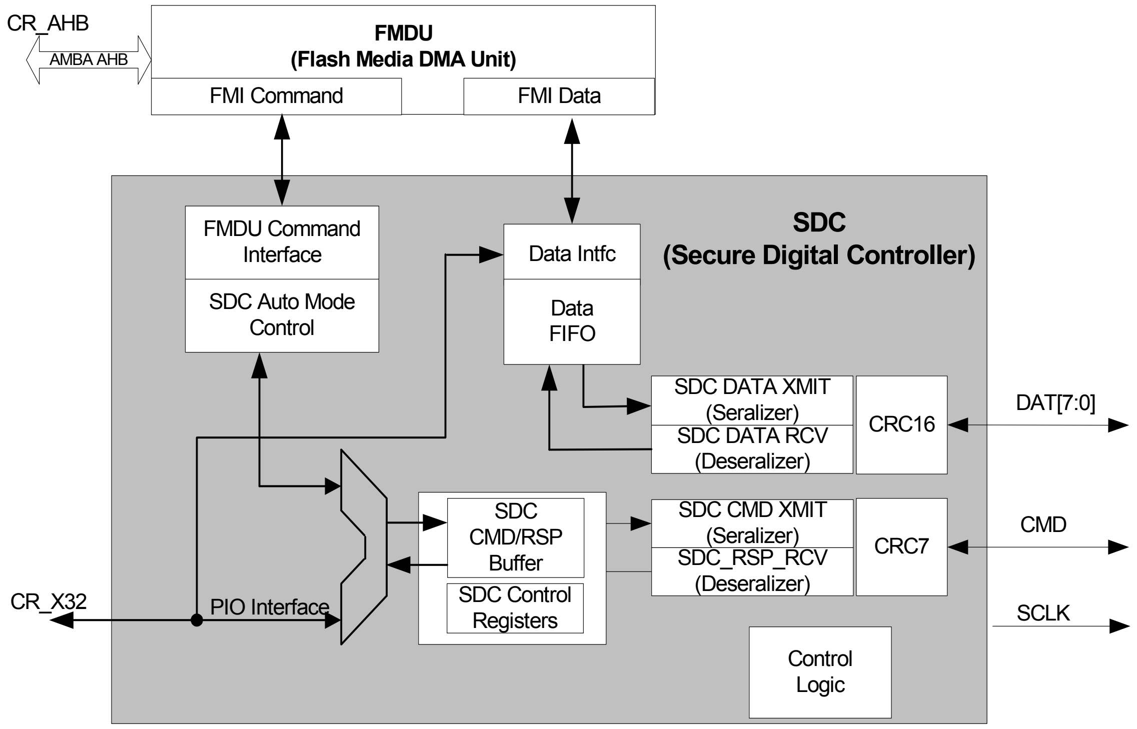


Figure 19.1 Secure Digital Controller Block Diagram

19.2 PIO Mode Access

19.2.1 Command Packet

In PIO mode the control logic is used to control the SDC bus interface manually. The main command interface to the media for the Processor is the CMD_RSP_BUF buffer. If the SDC_RDY is set, the SDC is ready to receive commands. The Processor writes a SD command into the buffer and sets the SEND_CMD bit in the SDC_MODE_CTL registers. The SDC controller sends out the command with the appropriate header and trailer, and puts the response back in the CMD_RSP_BUFF.

To select the SD controller manually, the Processor must select a LUN using the FMC_DEV_SEL register. The DEVICE field in the LUN_CTL1 register for that LUN must point to the SD controller. The SD controller gets the media information for device from the LUN descriptor table for that LUN.

For outgoing commands, the SDC first sends out a ‘0’ then a ‘1’, followed by the command contents. The SDC then sends out a 7 bit CRC followed by the stop bit. The Processor only puts in the command contents, the header and trailer and generated by the SDC. Refer to the *SD Memory Card Specifications, Part 1, Physical Layer Specification Version 1.1,June17 2004, SD Group* for the timing requirement.

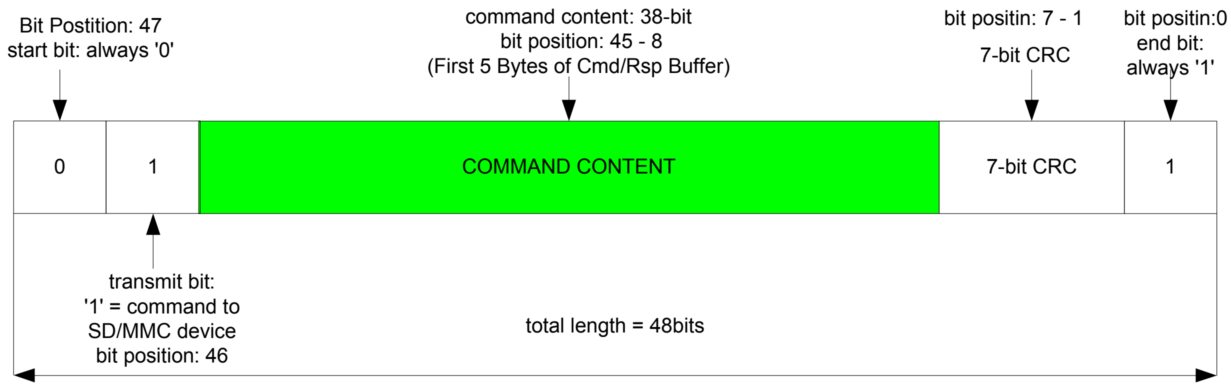


Figure 19.2 Command Packet Format

The CMD_RSP_BUF is loaded according to the table below.

Table 19.1 SDC Command Buffer Programming

Location	Value
CMD_RSP_BUF[0].	XX
CMD_RSP_BUF[1].	Argument[31:24]
CMD_RSP_BUF[2].	Argument[23:16]
CMD_RSP_BUF[3].	Argument[15:8]
CMD_RSP_BUF[4].	Argument[7:0]

After loading the CMD_RSP_BUF, the Processor initiates the transfer by writing non-zero bits to the SEND_CMD bit. The value written to these bits must correspond to the size of the expected response. This response size programmed into the RSP_SIZE bits is given in the table below.

Table 19.2 SDC SEND_CMD Programming

SD Response	RSP_SIZE	Value
None	CMD_NO_RSP	00
R1	CMD_48b_RSP	01
R1b	CMD_48b_BSY_RSP	11
R2	CMD_136b_RSP	10
R3	CMD_48b_RSP	01
R6	CMD_48b_RSP	101

Note: The firmware must check the SD/MMC busy bit, and not send any command other than GET_STATUS when the media is busy.

As soon as a command is issued, the SDC_RDY is removed to indicate that the device is busy. The bit is set again when the full response from the media has been received. This indicates that the device is ready for the next command. The SDC_RDY can also be used to set an interrupt to the Processor. When the command is issued, a timer is started. If the timer expires before the command and response is complete, the TIMEOUT_ERR interrupt is set. The timer is restarted with every command. If a CRC error occurs the CRC interrupt is generated.

19.2.2 Response Packet

There are two different sized response packets 48 bits and 136 bits. The header and trailer is stripped by the SDC and the response content is stored in the CMD_RSP_BUF. The data is stored such that the command index bit 5 through 0 are stored in CMD_RSP_BUF[0] bits 5 through 0. The most significant two bits of CMD_RSP_BUF is always zero.

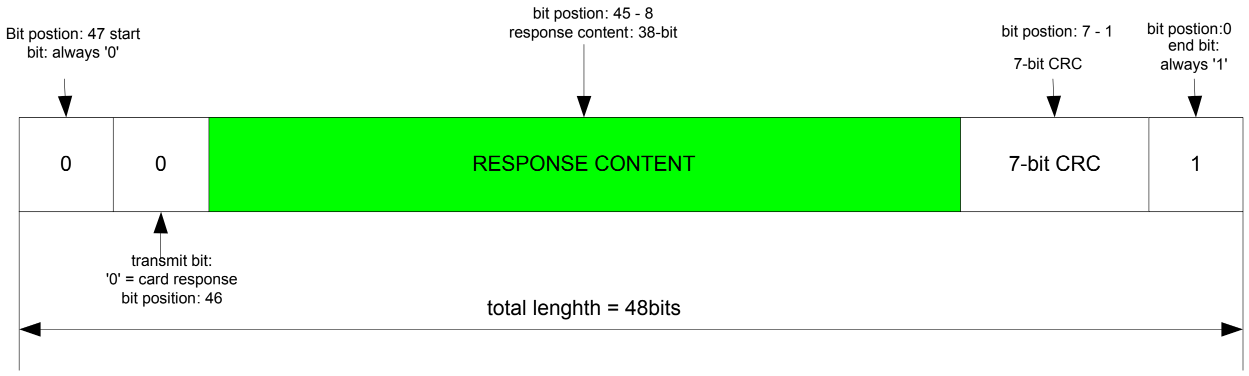

Bit postion: 47 start bit: always '0'
bit postion: 45 - 8 response content: 38-bit
bit postion: 7 - 1
7-bit CRC
bit postion:0 end bit: always '1'
0
0
RESPONSE CONTENT
7-bit CRC
1
transmit bit: '0' = card response bit position: 46
total lenghth = 48bits

Figure 19.1 48-bit Response Packet Format

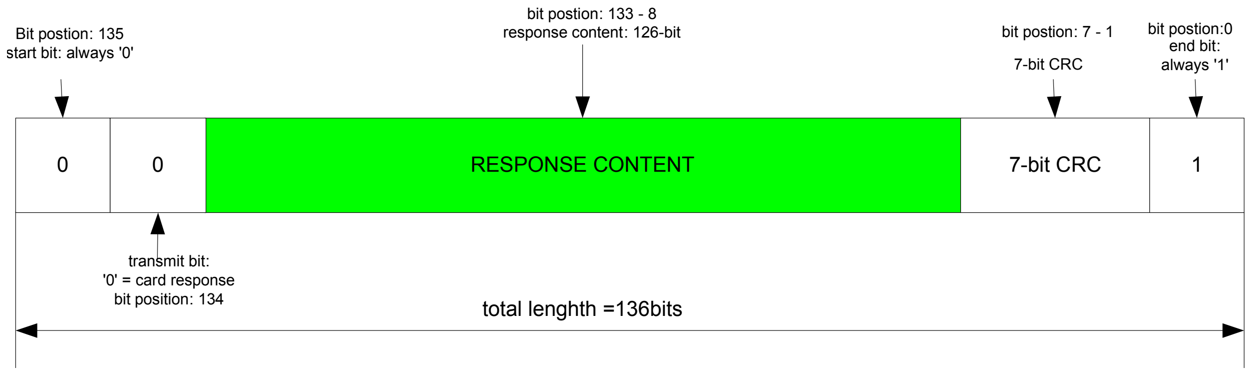


Figure 19.2 136-bit Response Packet

When the RESPONSE PACKET is received, the SDC checks the CRC and stores the RESPONSE CONTENT in the CMD_RSP_BUF Buffer. Then, SDC sets the SDC_RDY bit in the SDC_STAT register to indicate that the Command Response cycle is complete.

In PIO mode the data is moved using the data fifo and the byte count register. The firmware must load SDC byte count registers with the count of the data to be moved. After that the firmware reads or writes the individual bytes from the SDC_DATA register. When reading or writing this register the firmware must pay attention to the FULL and EMPTY flags of the FIFO.

PIO mode only supports short transfers of 2Kbytes or less. For transfers greater than the 2 Kbytes, the FMI interface MANUAL_READ or MANUAL_WRITE must be used.

Example: Application specific command

This is an example of an SCR register read. The Processor goes through the following steps:

1. The CMD_RSP_BUF is loaded with ACMD55, 00, 00, 00, 00.
2. The SDC byte count is loaded with 8, the expected byte count of the data phase.
3. Block transfer is enabled.
4. The SEND_CMD is set and RES_SIZE is loaded with 01 because the expected command response is 48 bits. This starts the transaction.
5. The command response is loaded into the CMD_RSP_BUF.
6. The data response is loaded into the Fifo memory.
7. The Processor drains the FIFO.
8. If a CRC error occurs, this sets the CRC error bit.

Note: The hardware must correctly deal with simultaneous command and data. The CRC must be calculated correctly and checked correctly. If the transfer size is required for CRC calculations, it is fetched from the FMC_EP_CNT register transparently from the Processor.

19.3 FMDU Command Interface

19.3.1 FMI Interface

The command interface from the FMDU to the SDC if the FMI interface. Please refer to the FMI microarchitecture specification for details. The interface has the following functional signals:

1. Start Address in the media. (fmdu_start_addr)
2. Length of transfer.(fmdu_xfer_length)
3. Block Size (encoded)(fmdu_block_size)

4. Command (read or write)(fmdu_cmd)
5. Identifier for the FMC selected.(fmdu_dev_sel)
6. ID of the unit within the controller. (fmdu_dev_id)
7. Timing and control signals.(xxa_valid, xxx_ack)
8. Error Signals(fmc_err)

19.3.2 Behavior

When the SDC receives an FMDU command it behaves in one of the following ways. If the previous state of the SDC was idle, it initates a new transfer. If the previous state was not idle, then the SDC sees if the new transfer is a new command or a continuation of the previous command. If it is not a continuation, the SDC terminates the old transfer, and initiates the new one. If it is a continuation, the SDC continues the old transfer. The behavior is somewhat different between reads and writes. There are four cases described:

1. NEW_READ
2. CONTINUE_READ
3. NEW_WRITE
4. CONTINUE_WRITE
5. MANUAL_READ
6. MANUAL_WRITE

All commands used to control the media must use the CMD_RSP_BUF buffer. This is required so that in the event of a error, the Processor can parse the response buffer to identify the error.

If the device options in the LUN descriptor table is set to 4, then the SDC will use LBA addressing, otherwise byte addressing will be used.

If the SDC is operating in MMC Streaming mode, the transfer is started and stopped manually by the Processor. The length is ignored and the device is programmed to stream data. In all probability the FMDU endpoint is programmed for continuous operation. Once streaming starts, it continues until a new command is issued. If the FMDU endpoint is not programmed for continuous mode, the streaming ends when the block transfer is complete.

19.3.2.1 SD/MMC FMDU Command – NEW_READ

The SDC goes through the following steps:

1. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512. The length is loaded into an internal length counter.
2. The starting address is converted to a byte address by multiplying the block address by 512. If in MMC streaming mode (MMC_STREAM = 1) then length has no meaning.
3. If there is a transaction in progress, close the current transaction with a stop command.The appropriate SD read multiple command is generated to the media using the starting address calculated above.
4. The response from the media is checked for errors, if there is no error, the SDC kicks off the DMA.
5. Once the DMA starts, the SDC controller tracks the data transfer. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.
6. When the DMA is over, the SDC will freeze in the current busy state if the CONT_RD bit is set. If the CONT_RD is not set, the SDC closes the transaction by sending a stop command to the media and goes to the Idle state. . If a new new command does not come within 1ms, the SDC will terminate the transaction by sending a stop command to the media.

19.3.2.2 SD/MMC FMDU Command – CONTINUE_READ

The SDC goes through the following steps:

1. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512. The length is loaded into an internal length counter.
2. If there is a transaction in progress, the SDC signals the FMDU that it is ready. The FMDU kicks off the DMA. Go to step 6.
3. If there is not a transaction in progress, the starting address is converted to a byte address by multiplying the block address by the 512.
4. The appropriate SD read multiple command is generated to the media using the starting address calculated above.
5. The response from the media is checked for errors, if there is no error, the SDC kicks off the DMA.
6. Once the DMA starts, the SDC controller decrements its length counter every byte. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.
7. When the DMA is over, the SDC will freeze in the current busy state if the CONT_RD bit is set. If the CONT_RD is not set, the SDC closes the transaction by sending a stop command to the media and goes to the Idle state. . If a new new command does not come within 1ms, the SDC will terminate the transaction by sending a stop command to the media.

19.3.2.3 SD/MMC FMDU Command – NEW_WRITE

The SDC is Idle and a NEW_WRITE comes in. The SDC goes through the following steps:

1. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512. The length is loaded into an internal length counter. If in MMC streaming mode (MMC_STREAM = 1) then transfer length has no meaning.
2. The starting address is converted to a byte address by multiplying the block address by the 512. The starting address is loaded into an internal start address.
3. If there is a transaction in progress, close the current transaction with a stop command.
4. The SDC generates an APP Command to the media using the RCA programmed in the RCA register.
5. If the CONT_WR bit is NOT set, the SDC will generate the appropriate multi-block pre-erase commands to the media. If it is set, the pre-erase will not be used.
6. The SDC will generate the write multiple command to the media using the starting address calculated before.
7. With each command the response from the media is checked for errors, if there is no error, the SDC kicks off the DMA.
8. Once the DMA starts, the SDC controller advances its internal start address every byte, and decrements its length counter. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.
9. When the DMA is over, the SDC will freeze in the current busy state, unless the CONT_WR is not set. If the CONT_WR is not set, the SDC closes the transaction by sending a stop command to the media, and goes to the idle state. . If a new new command does not come within 1ms, the SDC will terminate the transaction by sending a stop command to the media.

19.3.2.4 SD/MMC FMDU Command – CONTINUE_WRITE

The SDC is Idle and a CONTINUE_WRITE comes in, and the CONT_WR bit is set. The SDC goes through the following steps:

1. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512. The length is loaded into an internal length counter.
2. If there is a transaction in progress, the SDC signals the FMDU that it is ready. The FMDU kicks off the DMA. Go to step 9.
3. If there is not a transaction in progress, the starting address is converted to a byte address by multiplying the block address by 512.
4. The SDC generates an APP Command to the media using the RCA programmed in the RCA register.
5. If the CONT_WR bit is NOT set, the SDC will generate the appropriate multi-block pre-erase commands to the media. If it is set, the pre-erase will not be used.
6. The SDC will generate the write multiple command to the media using the starting address calculated before.
7. With each command the response from the media is checked for errors, if there is no error, the SDC kicks off the DMA.
8. Once the DMA starts, the SDC controller decrements its length counter every byte. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.
9. When the DMA is over, the SDC will freeze in the current busy state, unless the CONT_WR is not set. If the CONT_WR is not set, the SDC closes the transaction by sending a stop command to the media, and goes to the Idle state. If a new new command does not come within 1ms, the SDC will terminate the transaction by sending a stop command to the media.

19.3.2.5 FMDU Command – MANUAL_READ

For MANUAL_READ transfers the SDC assumes that the Processor has programmed all the FMDU control registers and the FMDU's EP0_WRITE registers to setup a transfer. The SDC also assumes the Processor has programmed the SDC with PIO accesses to setup the control portion of the transfer. Once the FMDU and SDC is setup, the Processor issues the FMI command MANUAL_READ command.

The firmware does the following setup before issuing an FMI command

1. Wait for the SDC to return to an IDLE state
2. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512.
3. The starting address is converted to a byte address by multiplying the block address by 512. If in MMC streaming mode (MMC_STREAM = 1) then length has no meaning.
4. The appropriate SD read multiple command is generated to the media using the starting address calculated above.
5. The response from the media is checked for errors, if there is no error, the Processor issues a FMI MANUAL_READ command.

The SDC goes through the following steps:

1. Once the SDC gets a MANUAL_READ command, the FMI length is loaded into an internal length counter,

2. The SDC controller kicks off the DMA. Once the DMA starts, the SDC controller tracks the data transfer. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.

On completion of the transfer, the firmware gets a BLK_XFER_COMPLETE interrupt from the FMDU. At that point the Firmware does the following:

1. The firmware closes the transaction by sending a stop command to the media.

19.3.2.6 FMDU Command – RAW_WRITE

For RAW_WRITE transfers the SDC assumes that the Processor has programmed all the FMDU control registers and the FMDU's EP0_READ registers to setup a transfer. The SDC also assumes the Processor has programmed the SDC with PIO accesses to setup the control portion of the transfer. Once the FMDU and SDC is setup, the Processor issues the FMI command RAW_READ command.

The firmware does the following setup before issuing an FMI command

1. Wait for the SDC to return to an IDLE state
2. If the media is greater than 4Gbyte, then LBA is used. If is less than 4GByte, the length is converted into a byte address by multiplying the length by 512. The length is loaded into an internal length counter. If in MMC streaming mode (MMC_STREAM = 1) then transfer length has no meaning.
3. The starting address is converted to a byte address by multiplying the block address by the 512. The starting address is loaded into an internal start address.
4. The firmware generates an APP Command to the media.
5. The firmware will generate the appropriate multi-block pre-erase commands to the media.
6. The firmware will generate the write multiple command to the media using the starting address calculated before.
7. With each command the response from the media is checked for errors, if there is no error, the firmware issues the MANUAL_WRITE command on the FMI interface.

The SDC goes through the following steps:

1. Once the SDC gets a MANUAL_WRITE command, the FMI length is loaded into an internal length counter,
2. Once the DMA starts, the SDC controller decrements its length counter with every byte transferred. When the length counter goes to zero, the clocks stop to the media at the correct bit number, including control bits.

On completion of the transfer, the firmware gets a BLK_XFER_COMPLETE interrupt from the FMDU. At that point the Firmware does the following:

1. The firmware closes the transaction by sending a stop command to the media.

19.4 SD Registers

Table 19.3 SDC Mode Control Register

SDC_MODE_CTL (CR_X32 + 0X1800 - RESET=0X00)			**SDC_MODE_CTL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	SD_SPEED	R/W	These bits are used to control the clock generator in the Secure Digital Controller (SDC). The clock signal is used to drive the clock input to the SD/MMC device. 000: 200 KHz (default) 001: 20 MHz 010: 24 MHz 011: 48 MHz, HS_SD enabled 100: 15 MHz – Special frequency for certain customers 101: 60 MHz - used ot overclock the interface 110: Streaming Mode(See STREAM_FREQ register) 111: 0 MHz (shut down) SD/MMC device should run at 200 KHz, when the SDC/MMC is in card identification mode only. The SDC controller may generate wait states to the Processor if accessed immediately after the clock speed is switched.
4	SDIOINT_EN	R/W	SDIO INTERRUPT ENABLE: Enables SDC monitoring of SDIO interrupts on DAT1 ‘0’ = No SDIO Interrupt Monitoring Support. ‘1’ = SDIO Interrupt Support is enabled. SDC will set the SDIO_INTR bit of the SDC_STAT register when an SDIO interrupt is detected.
3	SD_MMC_INTF_EN	R/W	Clearing this bit to “0”, disables the interface of SD/MMC. All of output signals to SD/MMC device are placed into high impedance state including any pull-up resistors. All of input associated with the interface are disabled and may be left open without causing additional leakage currents to flow internally. Setting this bit to “1”, allows these signals operate normally. Setting SDC_RST will not clear this bit.
2	SEND_CMD	R/W	This bit is used to manually send SD commands to the media. The size of the response is programmed into these bits. Writing a one initiates the transfer. Hardware clears this bit after receiving the response.
1:0	RSP_SIZE	R/W	The size of the response is programmed into these bits. NO_RSP (00b): command with no response 48b_RSP (01b): 48-bit response 136b_RSP (10b): 136-bit response. 48b_RSP (11b): 48-bit response with busy

Table 19.4 SDC Control Register

SDC_CTL (CR_X32 + 0X1801 - RESET=0X00)			**SDC_CTL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	SDIO_WAIT	R/W	SDIO READ WAIT: Forces the SDC to assert DAT2 to a READ/WAIT capable SDIO card in order to pause an active read transfer. '0' = Normal READ operation. '1' = WAIT is asserted by the SDC during the next interrupt period, and will remain asserted until this bit is cleared.
6	SDC_RST	R/W	Setting this bit to "1", resets SDC including registers and state machines of the SDC block. This bit is self-cleared, when the SDC reset process has completed.
5	CRC_DIS	R/W	Setting this bit to "1", disables the CRC logic for checking the data over DAT lines from or to the SD device. Clearing this bit to "0", enables the CRC logic for checking the data from or to the SD device.
4	RSP_CRC_DIS	R/W	Setting this bit to 1 , disables the CRC checker for all response packets. . Firmware is responsible for setting this bit, when the CRC-7 checking is not necessary for the response packet
3:2	BUS_SIZE	R/W	Enables data transfers on a 1, 4 or 8 bit data bus. '00' = Transfers occur on a 1-bit data bus (DAT 0), and the DAT[7:1] lines are tri-stated. Default '01' = Transfers occur on a 4-bit data bus (DAT[3:0]) DAT[7:4] lines are tri-stated. '10' = Transfers occur on a 8-bit data bus (DAT[7:0]) Firmware is responsible for setting and clearing these bits.
1	FORCE_CLOCK	R/W	It forces SD Controller to activate clock to the media.
0	PRE_ERASE_WR	R/W	Enable pre-erase sequence for write transfers.

19.4.0.7 FORCE_CLOCK

There is a requirement that SD and MMC devices have a running clock for at least 74 clock cycles before they are accessed. The SDC normally disable the clock to the device when not being accessed. To meet the 74 clock cycle requirement, the firmware must set the FORCE_CLOCK bit for a minimum of 370uS before accessing the device. The bit must be set after the FET for the device has been enabled, the power-on delay time has elapsed, and the interface has been enabled.

Table 19.5 SDC Control2 Register

SDC_CTL2 (CR_X32 + 0X180D - RESET=0X00)			**SDC_CTL2**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	ALWAYS_FORCE12	R/W	If this bit is set, it forces the SD controller to issue cmd12 at the end of a data transfer without waiting for 1ms counter to expire. This bit is set and cleared by the firmware

SDC_CTL2 (CR_X32 + 0X180D - RESET=0X00)			**SDC_CTL2**
BIT	**NAME**	**R/W**	**DESCRIPTION**
6	FORCE_CMD12	R/W	If this bit is set it forces the SD controller to issue cmd12 without waiting for 1ms counter to expire. If the bit is set during an active CBW transaction, cmd12 will be issued immediately after transaction. If it is set during 1ms waiting period the cmd12 is issued immediately. Bit is reset by hardware SDC_RDY is set. Bit can only be set if if SDC_RDY is not set.
5	CMD_TIMEOUT_STOP_E NABLE	R/W	‘0’ = The SDC will NOT stop on a timeout error. ’1’ = The SDC will stop on a timeout error.
4:2	DEV_TYPE	R/W	These bits control the device type attached to the SD controller Device Type Description 000 MMC 001 MMC_STREAM 010 SD Classic 011 Reserved 100 SD PRO 101 Reserved 110 Reserved
1	STOP_STREAM	R/W	This is a self clearing bit. When set it will stop MMC streaming at the next block boundary. This bit will clear once streaming stops. It is up the firmware to issue a stop command to the MMC device.
0	Reserved	R	Always read ‘0’

The firmware sets the FORCE_CMD12 when it wants to take control of the SDC controller. Setting this bit forces the hardware to expire the timer and send a CMD12 command to the SD device. Once the CMD12 is complete, the hardware clears the FORCE_CMD12 bit. The firmware must wait for this bit to clear before issuing any commands to the SDC. Failure to do so will result in unpredictable behavior.

If the 1 ms timer is already expired when the FORCE_CMD12 bit is set, it has no effect on the hardware, and the bit is cleared immediately.

Table 19.6 Command/Response Buffer

CMD_RSP_BUF (CR_X32 + 0X1810 RESET=0X40) (CR_X32 + 0X1811~ 0X1820 - RESET=0X00)			**COMMAND/RESPONSE BUFFER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
0:16	CMD_RSP_BUF[0:16]	R/W	This buffer is used by Processor to store outgoing SD commands. This buffer is also where the SDC writes the responses from the media. See behavioral description.

Note: CMD_RSP_BUFF[0] register default should be 0x40. When F/W tries to write this register, bits[7:6] will not be writeable. When response is loaded by the H/W, this register will reflect what ever is sent by the Media.

Table 19.7 MMC Streaming Frequency Register

STREAM_FREQ (CR_X32 + 0X180C - RESET=0X14)			**MMC STREAMING FREQUENCY DIVIDER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	DIVIDER	R/W	This register holds the controls the clock in MMC streaming mode. Frequency (KHz) = 12000/(N+1) Examples : 0x00 = 12 MHz 0x05 = 2 MHz 0x3B = 200 KHz 0xFF = 46.95 KHz

Table 19.8 SD Controller (SDC) Status Register

SDC_STAT (CR_X32 + 0X1802 RESET=0X00)			**SDC STATUS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	CMD_ERR	R/W	This is set, when a command error was detected. This bit is cleared by the Processor by reseting the SDC controller..
6	DATA_ERR	R/W	This is set, when a data error was detected. This bit is cleared by the Processor by reseting the SDC controller..
5	CMD_TIMEOUT_ERR	R/W	This is set, when a SDC detected timeout error during execution of command. This bit is cleared by the Processor by writing a ‘1’ to it.
4	Reserved	R	Always read ‘0’
3	SDC_CMD_RDY	R	This bit is set by the SDC to indicate that command state machine it is ready for the next command. This bit is cleared by the SDC at the start of a command, and is set when the command completes. If the command fails to complete due to an error, this bit does not get set.
2	SDIO_WAIT_INT	R/W	SDC indicates that Read Wait interrupt on line DAT[2] of SD interface was asserted. This bit is cleared by the Processor by writing a ‘1’ to it.
1	SDIO_INTR	R/W	This bit is set by the SDC when an SDIO Interrupt is detected on the DAT1 line. The firmware is responsible for clearing the SDIO card interrupt prior to clearing this bit. Since an SDIO Interrupt is level sensitive, the SDC will continue to assert this bit until the interrupt source is cleared. This bit is cleared by the Processor by writing a ‘1’ to it.
0	SDC_RDY	R	This bit is set by the SDC to indicate it is ready for the next command. This bit is cleared by the SDC at the start of a command, and is set when the command completes. If the command fails to complete due to an error, this bit does not get set.

Note 19.3 The bits in this registers are cleared by writing a “1” to the corresponding bit.

Note 19.4 Firmware needs to acquire the card status from SD/MMC card for the specific flash programming error.

Note 19.5 All the bits in the regisgter go to the interrupt controller

Note 19.6 The mask bit do not prevent the status in the SDC_STAT register from being set, only from setting the SDC_INTR bit in INT5 register.

Note 19.7 When operating in AUTO mode the Processor is responsible for masking off all non-ERROR interrupts.

Table 19.9 Relative Card Address0

RCA0 (CR_X32 + 0X1804~0X1805 - RESET=0X0000)			**RELATIVE CARD ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	RCA[31:16]	R/W	Relative Card Address for device 0

Table 19.10 Relative Card Address1

RCA1 (CR_X32 + 0X1806~0X1807 - RESET=0X0001)			**RELATIVE CARD ADDRESS REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	RCA[31:16]	R/W	Relative Card Address for device 1

Table 19.11 Card Status Read Error Mask

CRD_STAT_RD_ERR_MSK (CR_X32 + 0X1808~0X1809 - RESET=0XFFFF)			**CARD STATUS READ ERROR MASK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	RD_MASK[15:0]	R/W	A ‘1’ in any bit position disables detection of an error in the card status shifted left by 16. For example bit 15 is OUT_OF_RANGE, bit 10 is WP_VIOLATION.

Table 19.12 Card Status Write Error Mask

CRD_STAT_WR_ERR_MSK (CR_X32 + 0X180A - RESET=0XFFFF)			**CARD STATUS READ ERROR MASK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	WR_MASK[15:0]	R/W	A ‘1’ in any bit position disables detection of an error in the card status shifted left by 16. For example bit 15 is OUT_OF_RANGE, bit 10 is WP_VIOLATION.

Table 19.13 Timeout Register

SD_TIMEOUT (CR_X32 + 0X180E ~ 0X180F - RESET=0X00FF)			**TIMEOUT REGISTER FOR SD**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:0	SD_TIMEOUT	R/W	This register holds the timeout to be used for SD operations. For Command operations, this timeout is started when the command is issued. The timeout is stopped when the last bit of the response is received. For Data operation, if the media is busy for more than the timeout period the timeout interrupt is set. Whenever there is a timeout, the TIMEOUT_ERR bit is set. Each count represents 100 µS.

Table 19.14 SDC Transmit / Receive Data Register

SDC_DATA (CR_X32 + 0X1844 - RESET=0X00)			**SDC_TX_DB**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:0	D[7:0]	R/W	This is an eight bit wide FIFO. When the FIFO_FULL bit of SDC_FIFO_CTL register is “1”, writing data to this register will have unpredictable results. Reads from this register when FIFO_EMP is set, will return meaningless data. The length of a command should be programmed into the SDC_BC register, and the related command issued before this FIFO is written or read.

Note 19.8 The SDC_BC should be programmed first before writing to the SDC_DATA fifo.

Table 19.15 SDC Byte Count Register

SDC_BC (CR_X32 + 0X1848~0X1849 - RESET=0X00)			**SDC BYTE COUNT HIGH**
BIT	**NAME**	**R/W**	**DESCRIPTION**
15:11	Reserved	R	Always read ‘0’
10:0	SDC_BC [10:0]	R/W	This register contains the byte count of SDC counter register.

Note 19.9 The SDC_BC should be programmed with the number of bytes during a PIO Data Transfer. These registers should be programmed before issuing the command.

Table 19.16 SDC Fifo Control Register

SDC_FIFO_CTL (CR_X32 + 0X184A - RESET=0X20)			**SDC FIFO CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	Reserved	R	Always read ‘0’
5	FIFO_EMPTY	R	1 : FIFO empty 0 : FIFO contains data. This bit is set to “1” by writing the SDC_RST bit of SDC_MODE_CTL Register.
4	FIFO_F ULL	R	1 : FIFO full 0 : FIFO has empty space. This bit is cleared to “0” by writing the SDC_RST bit of SDC_MODE_CTL Register.
3	Reserved	R/W	Always read ‘0’
2	VAR_DAT_LEN	R/W	If this bit is set during the manual mode data transfer the SDC will use block size as defined by the SDC_BC register instead of the value delivered from FMDU on the FMI interface. Block size defined by the SDC_BC register must be double-word aligned. Start address and the length. Which means that allowed sizes are 4B, 8B, 12B, ... 1024B.
1	SEND_PIO_RD	R/W	When set to a ‘1’ it initiates PIO RD. The bit will be cleared by the hardware once the end of the transfer was detected (amount of transferred data match byte count as indiated by the SDC_BC register.
0	SEND_PIO_WR	R/W	When set to a ‘1’ it initiates PIO WR . The bit will be cleared by the hardware once the end of the transfer was detected (amount of transferred data match byte count as indiated by the SDC_BC register.

Table 19.17 (SDC SD Classic Card Delay Control Register

SDC_SD_CLASSIC_CLK_DEL (CR_X32 + 0X184B - RESET=0X59)			**SDC CLOCK DELAY CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	RDDATA_RSP_DEL	R/W	It defines amount of registers in synchronizer for read data and command respond. “2’b00” -> 2-stages synchronizer “2’b01” -> 3-stages synchronizer “2’b10” -> 4-stages synchronizer
5:4	SYNC_DEL	R/W	It defines amount of delay registers for internal control logic. “2’b00” -> 2-stages logic “2’b01” -> 3-stages logic “2’b10” -> 4-stages logic
3:0	CLK_DEL	R/W	It defines delay of the clock capturing read data from the card.

Table 19.18 SDC SD PRO Card Delay Control Register

SDC_SD_PRO_CLK_DEL (CR_X32 + 0X184C - RESET=0X54)			**SDC CLOCK DELAY CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	RDDATA_RSP_DEL	R/W	It defines amount of registers in synchronizer for read data and command respond. “2’b00” -> 2-stages synchronizer “2’b01” -> 3-stages synchronizer “2’b10” -> 4-stages synchronizer
5:4	SYNC_DEL	R/W	It defines amount of delay registers for internal control logic. “2’b00” -> 2-stages logic “2’b01” -> 3-stages logic “2’b10” -> 4-stages logic
3:0	CLK_DEL	R/W	It defines delay of the clock capturing read data from the card.

Table 19.19 SDC MMC Card Delay Control Register

SDC_MMC_CLK_DEL (CR_X32 + 0X184D - RESET=0X95)			**SDC CLOCK DELAY CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	RDDATA_RSP_DEL	R/W	It defines amount of registers in synchronizer for read data and command respond. “2’b00” -> 2-stages synchronizer “2’b01” -> 3-stages synchronizer “2’b10” -> 4-stages synchronizer
5:4	SYNC_DEL	R/W	It defines amount of delay registers for internal control logic. “2’b00” -> 2-stages logic “2’b01” -> 3-stages logic “2’b10” -> 4-stages logic
3:0	CLK_DEL	R/W	It defines delay of the clock capturing read data from the card.

Table 19.20 PAD Current Control Register

PAD_CTL_SD (CR_X32 + 0X1880 RESET=0X00)			**PAD CURRENT CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:2	Reserved	R	Always read “0”.
1:0	SEL	R/W	00 = 6 mA Operation (default) 01 = 8 mA Operation 10 = 10 mA Operation 11 = 12 mA Operation This register controls the current limit of the SD controller pins.

19.5 Secured Digital Controller (SDC) Description

19.5.1 SD Data Format

Three types of SD devices are supported by SDC: SD memory card, High-Speed SD (HS-SD) Memory card, and MMC card.

The MMC device uses three signal lines to transfer the data and command. The three signals are SD_CLK, SD_CMD and SD_D0. The transfer on CMD and DAT is serial. Both CMD and DAT transfer are synchronized with SD_CLK. The data packet is shown as in the figure below entitled “Data Format with DAT0 Only”.

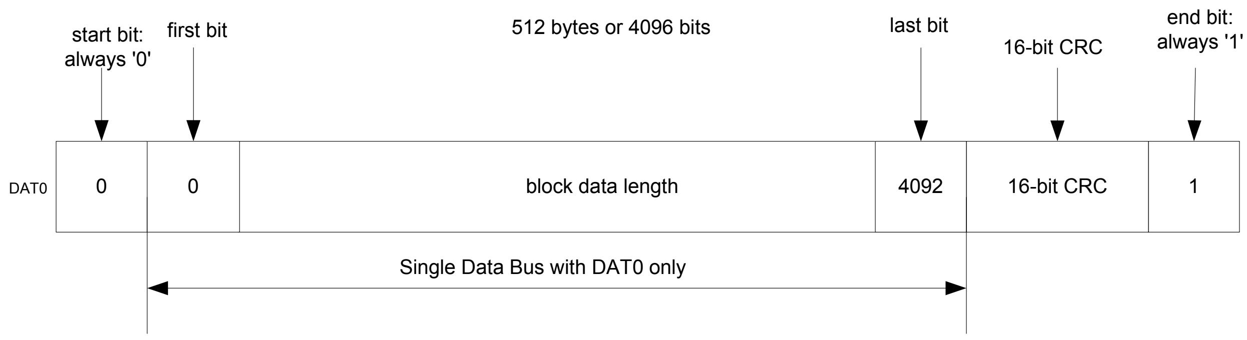


Figure 19.10 Data Format with DAT0 only

The SD memory card works the same way as the MMC card, except that SD memory card has a 4-bit data line. The SDC can drive the 4-bit data line and the logic associated with it when the BUS_TYPE bit in the SDC_CTL register is set to “1”. The data packet format is shown in Figure 15-6: Data Format with DAT3-015-6: Data Format with DAT3-0. The first nibble (0,1,2,3) of 512-byte or 4096-bit incoming data from SD/MMC device is stored in bits 7-4 of the least significant byte of the EP2 buffer. The second nibble (4,5,6,7) of 512-byte or 4096-bit is stored in bits 3-0 of the least significant byte of the EP2 buffer. The last nibble (4092, 4093, 4094, 4095) of 512-byte or 4096-bit incoming data from SD/MMC device is stored in bits 3-0 of the most significant byte of the EP2 buffer.

Refer to Byte/Bit Ordering for the byte and bit ordering of data transmission and reception.

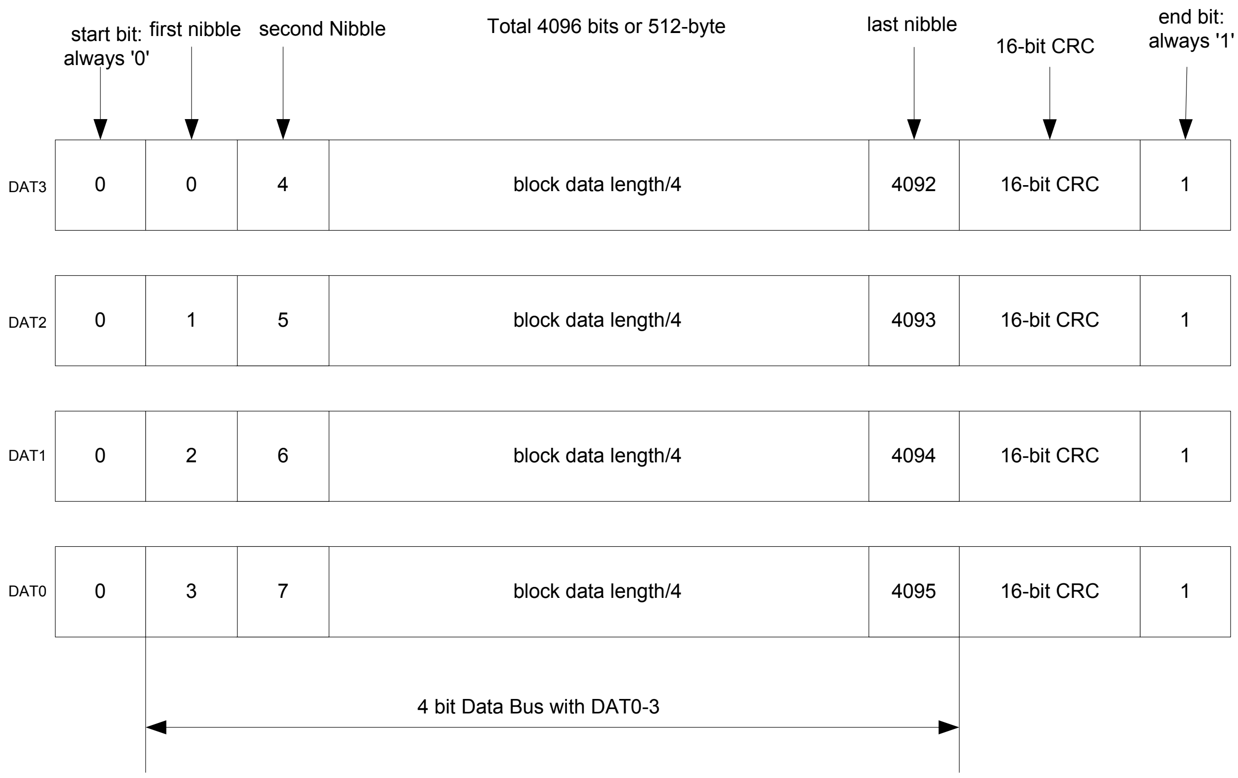


Figure 19.11 Data Format with DAT3-0

SD Clock Modulation during Block Data Transfer

SDC modulates the SD_CLK signal during the block data read/write transfer.

In the block data write operation, SDC drives the SD_CLK and data, when it has data to transmit to the SD/MMC device. Otherwise, SDC stops the SD_CLK and not drive the data line when it is not ready to transmit the data to the SD/MMC device.

In the block data read operation, SDC drives the SD_CLK and sample the data lines, when it is ready to receive the data from SD/MMC device. Otherwise, SDC stops the SD_CLK and does not sample the data line, when it is not ready to receive the data from SD/MMC device.

SD_CLK stay high when SDC stop driving SD_CLK signal.

19.5.2 SDC CRC Generator and Checker

CRC16 Generator/Checker is implemented to support the CRC generation and checking on the DAT lines. The 16-bit CRC is appended to the last bit of data to be transmitted from the SDC_DATA_XSR register.

The CRC16 Generator/Checker checks the incoming 16-bit CRC after the last byte of data is received from the SDC device. The CRC_ERR bit in SDC_STAT register is set, if there is a CRC error detected.

CRC7 Generator/Checker is implemented to support the CRC generation and checking for the COMMAND and RESPONSE data transfer. A 7-bit CRC is generated for any COMMAND packet to the SD device.

As to the CRC7 Checker, it is implemented to check the incoming RESPONSE packet, if the CRC check is needed.

Refer to the mode setting table in the section of SDC_MODE_CTL register for the CRC of each type of command and response.

The generator polynomial for CRC16 and CRC7 are listed here:

CRC16: G(X) = X16 + X12 + X5 + 1

CRC7:G(X) = X7 + X3 _+ 1

19.5.3 SD/MMC Power Control

SDC power control is done using a GPIO pin, and is not part of this functional block. Each SD device has its own power-on detection circuit. The card goes into a known state, after the power-on. No explicit hardware reset signal is necessary. The power to the device can be enabled or disabled by the power control signal.

After the power is enabled, the firmware should not access the SD/MMC device within the 1 msec.

It is up to the firmware to determine if a SD or a MMC card is inserted or removed by inquiring the device.

19.5.4 SDC Signal Timing

For SD Classic timing please use the timing diagrams in Section 6.7 of the SD Specifications part 1, Physical Layer Specification Version 2.00 dated May 9, 2006. The following table is a substitute for Table 6-5 in the specification.

Table 19.21 SD Classic Read/Write Timing

PARAMETER	SYMBOL	MIN	MAX	UNIT	REMARK
Clock CLK (All values are referred to min (VIH) and max (VIL))					
Clock frequency Data Transfer Mode @ 50% Duty cycle	f_{PP}	0	25	MHz	CCARD 10 pF (1 card)
Clock frequency Identification Mode	f_{OD}	0	400	kHz	CCARD 10 pF (1 card)
Clock low time	t_{WL}	20	21	ns	CCARD 10 pF (1 card)
Clock high time	t_{WH}	20	21	ns	CCARD 10 pF (1 card)
Clock rise time	t_{TLH}		5	ns	CCARD 10 pF (1 card)
Clock fall time	t_{THL}		5	ns	CCARD 10 pF (1 card)
Inputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Input set-up time (2410 write to card)	t_{ISU}	7		ns	CCARD 10 pF (1 card)
Input hold time (2410 write to card)	t_{IH}	7		ns	CCARD 10 pF (1 card)

Table 19.21 SD Classic Read/Write Timing

PARAMETER	SYMBOL	MIN	MAX	UNIT	REMARK
Outputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Output Delay time during Data Transfer Mode (2410 read from card)	t_{ODLY}	0	14	ns	CCARD 40 pF (1 card)
Output Delay time during Identification Mode (2410 read from card)	t_{ODLY}	0	50	ns	CCARD 40 pF (1 card)

For SD High Speed Mode timing please use the timing diagrams in Section 6.8 of the SD Specifications part 1, Physical Layer Specification Version 2.00 dated May 9, 2006. The following table is a substitute for Table 6-6 in the specification.

Table 19.22 SD High Speed Read/Write Timing

PARAMETER	SYMBOL	MIN	MAX	UNIT	REMARK
Clock CLK (All values are referred to min (VIH) and max (VIL))					
Clock frequency Data Transfer Mode @ 50% Duty cycle	f_{PP}	0	48	MHz	CCARD 10 pF (1 card)
Clock frequency Identification Mode	f_{OD}	0	400	kHz	CCARD 10 pF (1 card)
Clock low time	t_{WL}	10	11	ns	CCARD 10 pF (1 card)
Clock high time	t_{WH}	10	11	ns	CCARD 10 pF (1 card)
Clock rise time	t_{TLH}		2.5	ns	CCARD 10 pF (1 card)
Clock fall time	t_{THL}		2.5	ns	CCARD 10 pF (1 card)
Inputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Input set-up time (2410 write to card)	t_{ISU}	8		ns	CCARD 10 pF (1 card)
Input hold time (2410 write to card)	t_{IH}	4		ns	CCARD 10 pF (1 card)
Outputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Output Delay time during Data Transfer Mode (2410 read from card)	t_{ODLY}	0	14	ns	CCARD 40 pF (1 card)

Table 19.22 SD High Speed Read/Write Timing

PARAMETER	SYMBOL	MIN	MAX	UNIT	REMARK
Output hold time during Identification Mode (2410 read from card)	t_{ODLY}	2.5		ns	CCARD> 15 pF (1 card)

For SD 60Mhz Timing, Section 6.8 of the SD Specifications part 1, Physical Layer Specification Version 2.00 dated May 9, 2006. The following table is a substitute for Table 6-6 in the specification. The 60Mhz timing is not part of the SD specification. It is done to match card readers that over clock the SD interface.

Table 19.23 SD 60Mhz Read/Write Timing

PARAMETER	SYMBOL	MIN	MAX	UNIT	REMARK
Clock CLK (All values are referred to min (VIH) and max (VIL))					
Clock frequency Data Transfer Mode @ 50% Duty cycle	f_{PP}	0	60	MHz	CCARD 10 pF (1 card)
Clock frequency Identification Mode	f_{OD}	0	400	kHz	CCARD 10 pF (1 card)
Clock low time	t_{WL}	5	9	ns	CCARD 10 pF (1 card)
Clock high time	t_{WH}	5	9	ns	CCARD 10 pF (1 card)
Clock rise time	t_{TLH}		2.5	ns	CCARD 10 pF (1 card)
Clock fall time	t_{THL}		2.5	ns	CCARD 10 pF (1 card)
Inputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Input set-up time (2410 write to card)	t_{ISU}	5		ns	CCARD 10 pF (1 card)
Input hold time (2410 write to card)	t_{IH}	3		ns	CCARD 10 pF (1 card)
Outputs CMD, DAT (referenced to CLK at 50% of V_{DD})					
Output Delay time during Data Transfer Mode (2410 read from card)	t_{ODLY}	0	11	ns	CCARD 15 pF (1 card)
Output hold time during Identification Mode (2410 read from card)	t_{ODLY}	2.5		ns	CCARD> 15 pF (1 card)

19.5.5 Byte/Bit Ordering

The byte order of data block transmitted or received to or from SD/MMC card is in little-endian order. The byte order of data block transmitted or received to or from FMDU is in little-endian order as well. Therefore, the least significant byte (LSB) of data block is the first byte to be transmitted or received by SDC, and the most significant byte (MSB) of data block is the last byte to be transmitted or received by SDC.

The most significant bit (MSb) of each byte is the first bit to be transmitted or received by SDC. The least significant bit (LSb) of each byte is the last bit to be transmitted or received by SDC.

SDC SDIO Support:

SEC2410/SEC4410 Supports the following enhanced SDIO Features:

- SDIO Interrupt (1-bit and 4-bit modes)
- SDIO Read Wait
- SDIO Suspend Resume

19.5.5.1 SDIO Interrupt Support:

The capability to support SDIO Interrupts is card dependent, firmware must verify that an SDIO card has the capability to support Interrupts prior to enabling interrupt support in the SDC.

For Interrupt support:

1. Firmware Responsibility

i. Verify that the SDIO card supports Interrupts

j. Enable Interrupt support by asserting the SDIOINT_EN bit

k. When an interrupt occurs (signified by SDIO_INTR = ‘1’)
   i. Access SDIO card and determine the source of the interrupt
   ii. Clear the SDIO card interrupt (if necessary)
   iii. Clear the SDIO_INTR

2. Hardware Responsibility

l. When SDIOINT_EN = ‘1’, monitor DAT[3] & DAT[1] for an interrupt condition on DAT1 (when a data transfer is not in progress)

m. If an interrupt condition is detected, assert the SDIO_INTR bit

Note 19.12 For Interrupt support when the card is placed in a low-power state (i.e. SD Card clock is stopped), the SDC and the card must both be configured for 1-bit operation.

Note 19.13 The SDC hardware does not support wakeup from a low power mode. To support wake-up from SDIO, the DAT1 line must be connected to a GPIO configured as an input. That GPIO must have the interrupts enabled. That interrupt must be routed to the WAKE_SRC register.

Note 19.14 During a Multi-Block read operation, An SD Card will continue to transfer data until a Stop Command is issued. The SDC will have been programmed for a particular data transfer size, and after the proper amount of data is transferred, the SDC will begin to monitor for SDIO Interrupts. Since the SD Card is still transferring data, the SDC will see the data transfer as an SDIO Interrupt. Firmware must do one of the following:

1. Mask interrupts during the interval that the SDC will mistakenly misinterpret a transfer as an SDIO Interrupt.

2. Temporarily disable SDIO Interrupt support until a stop command is received by the SD Card.

19.5.5.2 SDIO Read Wait Support

The capability to support SDIO Read Wait is card dependent; firmware must verify that an SDIO card has the capability to support Read Wait prior to enabling interrupt support in the SDC.

For Read Wait support:

1. Firmware Responsibility:

n. Set the SDIO_WAIT bit when a READ WAIT operation is required

o. Send command(s) to the SDIO card

p. When operation is complete, clear the SDIO_WAIT bit

2. Hardware Responsibility:

q. When SDIO_WAIT = ‘1’
 i. Retain all current read state and data information for the read operation that is being stalled.
 ii. Assert the READ WAIT condition on DAT2 (according to SDIO spec timing restrictions, and delay sending any SD commands until DAT2 is asserted).
 iii. When DAT2 is asserted, send commands as directed by firmware.

r. When SDIO_WAIT = ‘0’
 i. Negate DAT2
 ii. Resume the stalled read operation (if applicable)
 iii. Return to normal operation.

19.5.5.3 SDIO Suspend/Resume Support:

The SEC2410/SEC4410 hardware has had no specific enhancements to support SDIO Suspend/Resume. Suspend/Resume support is dependent upon firmware.

Chapter 20 Second Secured Digital (SD/MMC) Controller

SEC2410/SEC4410 has a second SD controller (SD2). The control registers are located on the CR_X32 bus. Table below shows the register addresses of the SD2.

Table 20.1 SD2/MMC2 Controller (SDC2) Interface Registers

ADDRESS In CR_X32 Space	NAME	R/W	DESCRIPTION	Access In Configuration	PAGE
0x2800	SDC_MODE_CTL	R/W	SDC2 Mode Control		
0x2801	SDC_CTL	R/W	SDC2 Control		
0x280D	SDC_CTL2	R/W	SDC2 Control		
0x2802		R/W	SDC2 Status Register		
0x2803	_MSK	R/W	SDC2 Mask Register		
0x2804	RCA0	R/W	Relative Card Address		
0x2806	RCA1	R/W	Relative Card Address		
0x2808	RD_ERROR_MSK	R/W	Read Error Mask		
0x280A	WR_ERROR_MSK	R/W	Write Error Mask		
0x280C	STREAM_FREQ	R/W	MMC Streaming divider		
0x280E~ 0x280F	SD_TIMEOUT	R/W	SD TIMEOUT register		
0x2810~ 0x281F	CMD_RSP_BUF	R/W	Command Response Buffer		
0x2844	SDC_DATA	R/W	PIO mode data		
0x2848~ 0x2849	SDC_BC	R/W	Byte Count		
0x284A	SDC_FIF0_CTL	R/W	PIO Data Fifo Control		
0x284B	SDC_SD_CLASSIC_CLK_DEL	R/W	SD classic mode clock delay		
0x284C	SDC_SD_PRO_CLK_DEL	R/W	SD Pro mode clock delay		
0x284D	SDC_MMC_CLK_DEL	R/W	MMC mode clock delay		
0x2880	PAD_CTL_SD	R/W	Pad current control		

Chapter 21 Card Reader AHB Bus

21.1 CR_AHB Bus

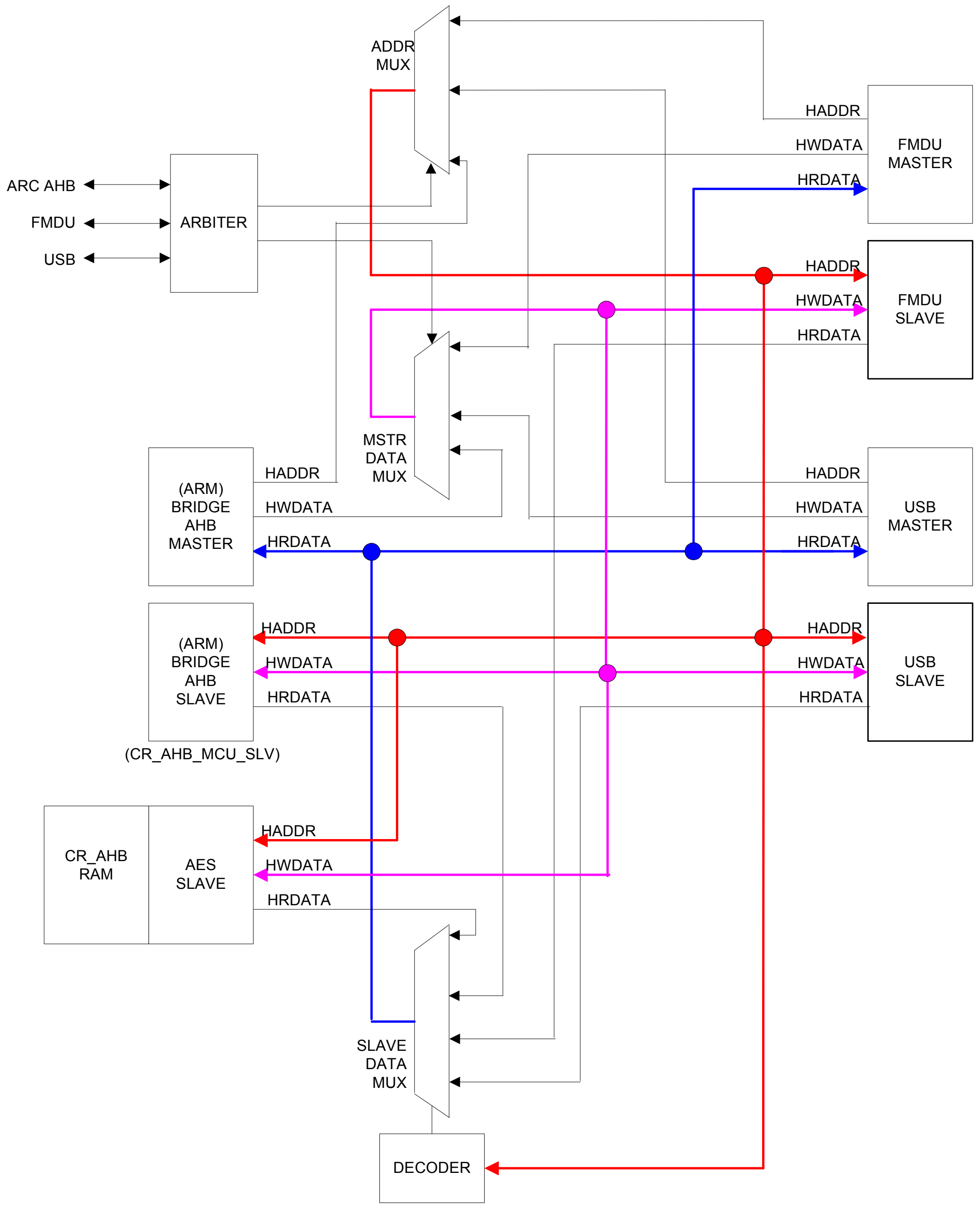


Figure 21.1 Card Reader AHB Bus Block Diagram

21.2 Card Reader AHB Bus Width

In SEC2410/SEC4410, the card reader AHB bus is 20 bits wide. The Processor puts out a 32 bit address. Any processor access between 0x20000000 and 0x200FFFFF is sent to the Card Reader AHB bus. On the CR_AHB bus itself, the upper eight bits are not propagated.

From the CR_AHB perspective, all device addresses are 20 bits. To access the same device, the processor must pre-pend 0x200 to the address. For example, the FMDU slave address on the CR_AHB bus is 0x00F400. To access the same slave, the processor puts out 0x2000F400.

21.3 Card Reader AHB Arbiter

In SEC2410/SEC4410, a round robin arbiter is used for the AHB. All devices have equal priority. If the arbiter sees a bus master put out two back to back bus idle cycles with another bus master waiting, it will remove the grant from the first and give it the second bus master.

This implementation relies on the good behavior of the bus master. The Bus Arbiter has a table for the high water mark for how long an individual bus master may have control of the bus. If the Bus Master stays on longer than the throttle count, and there is another pending request, its grant is removed and given to the next requester. The throttle count is only relevant when there are other requests pending. Each bus master must release the bus when it has no further data. Bus parking is not permitted.

The throttle count is set to single default value. The Processor must come and set the priorities as dictated by system conditions. Throttle count is clock cycle based. It is counted the same whether the cycle is idle or busy. Improper settings will break the system. All the registers in the CR_AHB space. The processor is free to change the values at any time. It may take one or more arbitration cycles before the new value takes effect.

The Arbiter will grant the bus for at least 4 clock cycles once there is a new owner of the bus. After that, the ownership will change based on the throttle register settings and system activity.

Table 21.1 Arbiter Throttle Count Register

ARB_THROTTLE (RESET 0X10)			**ARBITRATION THROTTLE REGISTER**
BIT	NAME	R/W	DESCRIPTION
7:0	COUNT[7:0]	R/W	This is Maximum number of clock cycles that a device can have control of the bus.

Table 21.2 Arbiter Throttle Table

ENTRY NUMBER	REGISTER ADDRESS CR_X32 OFFSET	DEVICE
0	0x0C40	FMDU
1	0x0C41	SIE
2	0x0C42	Processor

Table 21.3 Suggested Values

DEVICE	HIGH MARK
FMDU	18
SIE	18
Processor	4

21.4 AHB Bus Masters

The arbiter relies on the good behavior of the Bus Masters. Good behavior constitutes the following things:

1. All Bus Master are 32 bits wide. With easy expansion to 64 bits in the future.
2. All Bus Master operate at 60Mhz,
3. Bus Masters send out completion notifications from the previous completed transactions first, before transferring new data.
4. Bus Masters should not occupy the bus with Idle cycles. If they have no data to move, they must relinquish the bus and re-request the bus when they have data to move. If the Arbiter detects more than two back to back idle cycles, and there are is another Bus Master waiting, it will remove the grant from the first bus master and give the bus to the waiting device.
5. Once a Bus Master has ownership of the bus, it must relinquish control of the bus when its grant is removed.
6. Once a Bus Masters has ownership of the bus it must try to do as much data movement as possible, including concatenating data from different endpoints.
7. All devices are Big Endian on the bus. From the processors perspective they are little endian as the endianess is swapped through the bridge.

21.5 AHB Bus Slaves Addresses

Table 21.4 AHB Slave Addresses

DEVICE	PROCESSOR ADDRESS	CR_AHB ADDRESS
FMDU	CR_AHB+F400	0x00F400
SIE	CR_AHB+F000	0x00F000
Processor	CR_AHB+EC00	0x00EC00

21.6 Transport Mechanism

All devices have an AMBA interface. All communication between devices happens over the AMBA bus, with minimal usage of side band signals. All data transfers use the AMBA bus with no other dedicated data paths. These interfaces have both a master interface and a slave interface. All data transfers are done from device to memory or memory to device. Only control information is transferred from device to device.

Data movement through the AMBA bus hardware is controlled by the Processor using the Transport Interface. This Interface is intended to be transport agnostic. Implementations over USB, or Flash Media are interchangeable.

The key components of the Transport interface are:

1. Work Queues (WQ). Each device has a WQ for every endpoint in the device. A channel is made of a endpoint pair. The WQs are the resource used to submit Work Requests to the Transport Interface.
2. Work Request (WR) describes a block of memory intended for use in a data transfer. The Work Request contains a pointer to the memory block, the length of the block, and attributes to modify the transfer, control completion notification, and indicate status.
3. Work Queue Elements (WQEs) is a reference to a work request. It is posted to a Work Queue in order to initiate the transfer of data. Passing the reference instead of the whole work request minimizes the amount of data that must be transferred during a DMA operation. In actual operation the WQE is passed through a look up table to retrieve the Work Request. It is the Work Request that is enqueued.
4. Completion Notifications (CNs). A CN is a mechanism for a device to inform a recipient that it is done with a WR. The CN is identical and indistinguishable from the WQE in content.
5. Completion Queues (CQ). This is the queue that a device posts Completion Notification as it completes a work request. There is one Completion queue per device.
6. Work Request Look-up Table (WRLUT) This is a table in each device for WR to be cached so they can be referenced with a WQE. Without the WRLUT, the whole Work Request would have to be posted.

The basic operation of the system is described in the following way: The Processor assigns one or more work requests to a device in a transport queue. A work request is simply a buffer with an address, length, “context”, completion address, and status. Every device looks at it Work Queue, pops the WR from the top of the queue, and executes. Depending on the WR type, a device either reads or writes to that buffer. When the WR is fulfilled, the device generates a Completion Notification (CN) to notify the recipient that the device is done with the buffer. The CN notification is sent to the recipient (completion) address in the WR, and the data written is the context to identify the buffer, length, and status. The bus interface has no other knowledge of what is going on in the system.

To minimize the number of bus cycles that are required to move data around, the Work Requests are “cached” in each device during initialization in a look-up table called the Work Request Look Up Table (WRLUT). From then on, all that is passed around is a reference to the WR in the table. This reference is called a Work Queue Element WQE. To assign a particular WR to an endpoint, the Processor simple pushes the WQE onto the transport queue of that endpoint. The endpoint pops the WQE, and uses it to lookup the WR in its LUT, and then executes it. When the device is done, it generates a Completion Notification that goes into its completion queue. The completion notification consists of the completion address, context and status. When the bus is available to the device the completion notification goes out. The outgoing completion notification appears to the recipient as a write to its transport queue.

There is no relationship between the size of a work request buffer, and the MTU of the device. If a work request is bigger than the MTU size of the device, the device must put multiple block contiguously into the work request buffers. If the work request is smaller than the MTU, the device must aggregate the work requests until it has enough space to do a MTU’s worth of data transfer. If the last bit is set, the buffer goes out even if it is less than an MTU.

21.7 Basic Data Flow

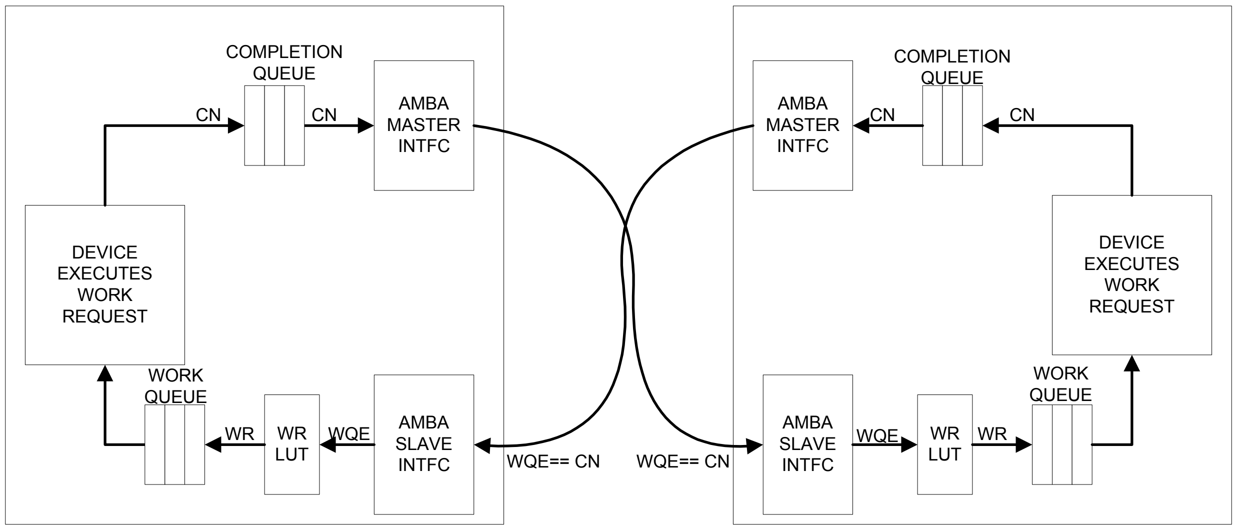


Figure 21.2 Basic Data Flow

The basic data flow for all transfers are the same. Before the transfer starts, the Processor initializes the Work Request Look Up Table (WRLUT) for the devices involved in the transfer. The Processor then primes the transfer by writing in the WQEs for the source device. At the source device the WQEs are fed through the WRLUT and the Work Request data is retrieved. The source device executes the WR. Once the WR has been executed, the source device creates a completion notification. The completion notifications goes out when the source device has control of the bus. The completion notification becomes the WQE at the Sink device. The Sink device uses the LUT to fetch the WR. The sink device executes the work request, generates its CN, which goes out on the bus and becomes the WQE for the Source device. The cycle continues until the link is broken.

21.8 Work Queue Element (WQE)

21.8.1 Short Format

The WQE is a references to a work request. There are two formats for the WQE. The first is a single byte format. This format is identified by the single byte write. In this format the data transfer is accomplished by writing a single byte to a work queue. This is the format used for most transfers. The Processor almost exclusively uses this format. The hardware identifies this format by a single byte write. The format given in the table below:

Table 21.5 WQE Short Format

BITS	NAME	DESCRIPTION
D7	LAST_BUFFER	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	CONTEXT[6:0]	This field contains the reference to the Work Request being used.

Note: For little Endian, the address of a single byte write to the work queue is always base + 0b00.

21.8.2 Override Format

The second format is the override format. This format is identified by a 32 bit write on the AMBA bus. In this format, the length in the WR is overridden by the length passed in the WQE. The reason for this mode is to allow the devices to signal each other that a transaction is completed. For instance, suppose the WR specifies that a 512Byte buffer is posted to the SIE, and a 31 byte CBW arrives. When informing the recipient of that buffer, it is necessary to inform the recipient that the length is 31 bytes and not 512 bytes as specified in the Work Request. The Processor never has to use override mode as it can always over write the length in the WR, whereas other devices cannot. If the Processor opts to use this format, it must use a 32 bit access mode in the CR_AHB to AHB bridge. For simplicity, this is the only format that is used by the hardware.

The override length only applies to actual data length, which means buffers that contain data. If an empty buffer is posted, like to a Write Endpoint, the length of the buffer is always the length posted in the WRLUT.

The format is given below:

Table 21.6 WQE Override Format

BITS	NAME	DESCRIPTION
D31~D28	Reserved	Not Used
D27~D24	Status	Not Used Currently
D23~D8	LENGTH	This length overrides the length in the work request. Only applies when data is written to a buffer. The length of an empty buffer is the length in the wrok request. For SEC2410/SEC4410 the length is truncated to the LS 12 bits
D7	LAST_BUFFER	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	CONTEXT[6:0]	This field contains the reference to the Work Request being used.

Note: For little Endian, the address of a word write to the work queue is always base + 0b0000.

21.8.3 Completion Notification (CN)

The WQE and the CN are identical in structure. The WQE and CN are references to a work request. There are two formats for the CN. The first is a single byte format, given in the table below:

Table 21.7 WQE/CN Short Format

BITS	NAME	DESCRIPTION
D7	LAST_BUFFER	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	CONTEXT[6:0]	This field contains the reference to the Work Request being used.

In this format the data transfer is accomplished by writing a single byte to a work queue. This is the format used for most transfers. The Processor only uses this format. The second format is the override format which is given below:

Table 21.8 WQE/CN Override Format

BITS	NAME	DESCRIPTION
D31~D28	Reserved	Not Used
D27~D24	Status	Not Used Currently
D23~D8	LENGTH	This length overrides the length in the work request. For SEC2410/SEC4410 the length is truncated to the LS 12 bits
D7	LAST_BUFFER	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	CONTEXT[6:0]	This field contains the reference to the Work Request being used.

In the override mode, the length in the WR is overridden by the length passed in the CN. This only applies to buffers that contain data. If a buffer is empty, like a buffer posted to a Write Endpoint, the length is always the length in the WRLUT.

The device generating the CN sets the Last_buffer bit for one of two reasons. If a short packet came in, or second the Last bit is set in the Work Request for that buffer. Setting this bit informs the recipient that this is the last buffer in the transaction. For simplicity, this is the only format that is used by the hardware.

21.8.4 Work Request Look Up Table (WRLUT)

Each AMBA device has a Work Request Look-up Table.(WRLUT). The WRLUT is a “cache” of the Work Request that will be assigned to that device. This table is loaded at device initialization. Once the WRLUT has been initialized, the WR can be accessed by the WQE. The context in the WQE is the index into the WRLUT.

WRLUT can have in theory up to 127 entries. Most devices has only 4 entries. Devices that have multiple endpoints such as the SIE has 16 entries. The WR request number is specific to a device. Refer to the individual devices to find the size of the individual tables.

Table 21.9 WRLUT Format

OFFSET	CONTEXT	WORK REQUEST
0x0	0	Work Request 0
0x10	1	Work Request 1
0x20	2	Work Request 2
0x30	3	Work Request 3
0x40	4	Work Request 4
0x50	5	Work Request 5

There is no need to have separate WRLUTs per endpoint because only a single WQE element can be written in a single clock cycle. A single table can be used to service all the Endpoints.

The Work Queue depth of individual endpoints will vary within a device.

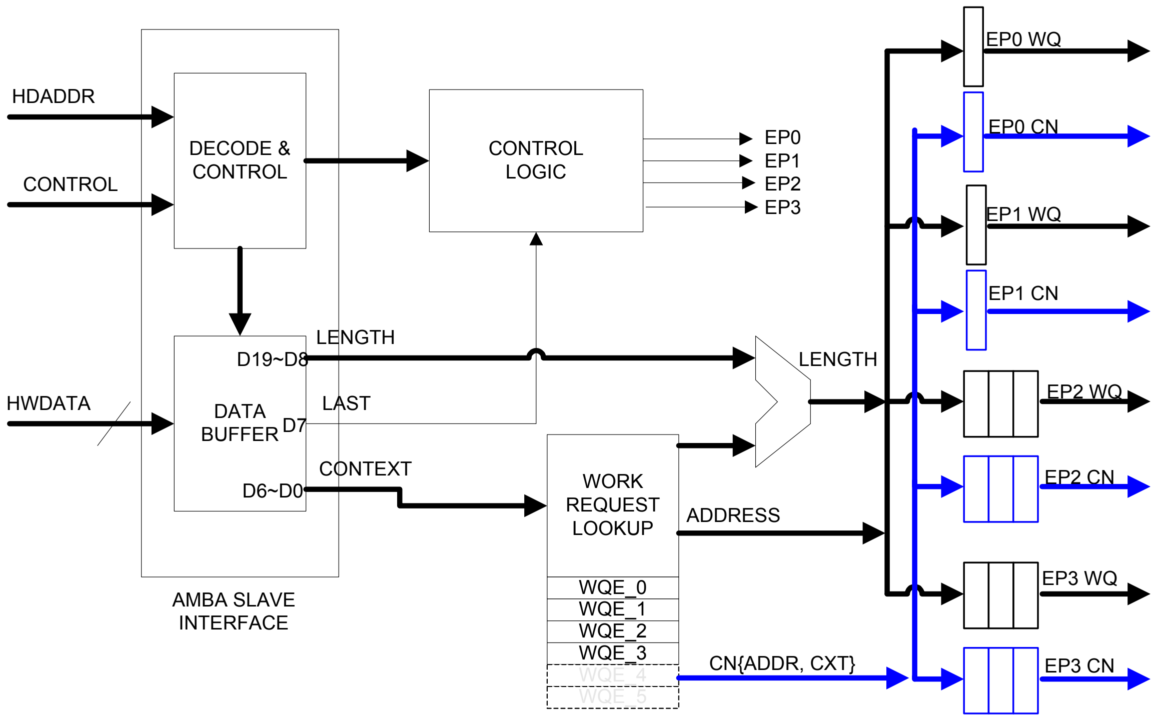


Figure 21.3 Queue Lookup

21.8.5 Work Queue (WQ)

The Work Queue (WQ) is where all the Work Requests are queued up. There is one WQ per endpoint. The WQs are accessed via the AMBA AHB bus slave interface. The AMBA decode block detects the address and allows writes to the Queue controller. The primary interface into the queue controller is a 20 bit wide Buffer into which the Work Queue Element (WQE) is written. The WQE is used to reference the WRLUT. The Work Request that is accessed is written into the Work Queue that is being accessed. The data written into the data buffer has the following definition:

Table 21.10 WQE Slave Input Format

BITS	NAME	DESCRIPTION
D31~D28	Reserved	Not Used
D27~D24	Status	Not Used Currently
D23~D8	LENGTH	This length overrides the length in the work request. For SEC2410/SEC4410 the length is truncated to the LS 12 bits
D7	LAST_BUFFER	If last_buffer = 1, this indicates that this is the last buffer of the transaction.
D6~D0	CONTEXT[6:0]	This field contains the reference to the Work Request being used.

There is one Work Queue per endpoint. The address of the queue is Base address + 0x100. Each queue is offset every eight addresses. There is a limit that any single device can only have 16 read endpoints and 16 write endpoints. The address for each endpoint is defined in the following table:

Table 21.11 Work Queue Address Small

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Address						0	1	R/W	Endpoint Number				0	0	0

Table 21.12 Work Queue Address Large

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

Address Bits 15~0, Bits 31~9

Upper bits of the address. For the small model only 16 bits are decoded by the AMBA AHB slave interface, for the large model, 32 bits are decoded.

“1” Bit 8

Bit 8 must be a one to get 0x100 base.

RW Bit 7

The RW bit determines whether this is a read = 0 or write = 1 endpoint.

Endpoint number Bit 6~3

The number of the endpoint being written to.

Rounding Bit 2~0

The last three bits of the address must be zero. This forces the Work queue to be 8 byte aligned. This is to allow for 64-bit busses in the future.

Work Queue Addresses

The Processor is at base address 0x00EC00. The Processor has 4 read endpoints and 4 write endpoints. The work queue addresses for the Processor is shown in the following table.

Table 21.13 Processor Work Queue Addresses

EP NUMBER	READ	WRITE
0	0x00ED00	0x00ED80
1	0x00ED08	0x00ED88
2	0x00ED10	0x00ED90
3	0x00ED18	0x00ED98

SIE Queue Addresses

In the small model the SIE is at base address 0x00F000. The SIE has 4 read endpoints and 4 write endpoints. The work queue addresses for the SIE is shown in the following table.

Table 21.14 SIE Work Queue Addresses

EP NUMBER	READ	WRITE
0	0x00F100	0x00F180
1	0x00F108	0x00F188
2	0x00F110	0x00F190
3	0x00F118	0x00F198

FMDU Queue Addresses

In the small model the FMDU is at base address 0x00F400. The FMDU has 1 read endpoints and 1 write endpoints. In the actual implementation, each endpoint is implemented as a single bidirectional endpoint. The work queue addresses for the FMDU is shown in the following table.

Table 21.15 FMDU Work Queue Address

EP NUMBER	READ	WRITE
0	0x00F500	0x00F580

21.9 Flushing a Work Queue

The Processor may occasionally be required to flush a Work Queue. To do this, the processor clears the FLUSH_DONE. The processor then writes a 0xFF to the work queue context byte. The 0xFF flushes all WQE in that endpoint only. Any other endpoints in that device is not affected. The value of 0xFF is special in that it forces the hardware to ignore the length, even though the last bit is set. The FLUSH_DONE bit is set when the flush is complete. The EP DMA Count will not be preserved after the flush occurs. FW must check the count, status before issuing the Flush.

For the DMA Reads from Memory (IN) direction, termination of the transfer will occur as soon as the flush is detected. Any pending CNs will go out but will not continue transferring data until an End of Packet of End of Buffer is reached.

For the DMA Writes to Memory (OUT) direction the hardware will keep running until an End Of Packet for the current packet it reached, then retire the memory buffer with a CN. If there is no data in the Elasticity Buffer then the transfer is just terminated.

21.10 Work Request (WR)

The Work Request (WR) describes a block of memory intended for use in a data transfer. The Work Request contains a pointer to the memory block, the length of the block, and attributes to modify the transfer, control completion notification, and indicate status.

Table 21.16 Work Request Descriptor

OFFSET	BYTE 3	BYTE 2	BYTE 1		BYTE 0
0x0	Not Used in 24 bit Model	Data Reference			
0x4	Not Used in 24 bit Model	Completion Address			
0x8	Not Used in 24 bit Model		Length		
0xC	Not Used in 24 bit Model		Reserved	L	Context

Table 21.17 Work Request Descriptor Fields

FIELD	LENGTH (IN BITS)	DESCRIPTION
Data Reference[1]	16/24/32	Pointer to data buffer in memory. Small model assumes 64 K address space. Large model assumes 32 bit address
Completion Address1	16/32	Pointer into memory where the completion notification needs to be sent.
Length[1]	12/16/32	Length in bytes of data buffer. For SEC2410/SEC4410 the length is truncated to the LS 12 bits
Context	7	This field is supplied by the FW and used to determine the appropriate buffer during completion processing.
(L) Last	1	Used to denote the last buffer (EOM) in Works that support the concept of a transaction.

[1] The size of these variables depends on the system configuration. For SEC2410/SEC4410 a 12 bit length and 24 bit Data reference is used.

21.11 Last Bit

21.11.1 Using the Last bit in the Work Request

If the device is reading from RAM and sending the data out, and the last bit is set in a Work Request, it forces the device to send out the buffer without waiting for the MTU to complete. For instance, if the MTU size is 512 bytes, and there is a 200 Byte buffer already queued to go out, and the new buffer is 100 bytes with the Last bit set, the device must send out both 200 and the 100 byte buffers without waiting for the 512 byte MTU to complete.

If the device is inputting data and writing to RAM and the last bit is set in a Work Request, it enables the receive without waiting for MTU's worth of buffer space, and it is also used to detect an overflow error.

21.11.2 Override mode Last Bit

The hardware generates the Last bit in override mode. The last bit is generated when a device is writing to RAM, and a short packet comes in. If the MTU size is 512 bytes, and there is a 1000 Byte buffer queued for input, any packet less than the MTU size will cause the Completion Notification to go out with the Last Bit set. Suppose a 31 byte CBW comes in, the buffer will be retired and a completion notification will be generated with the last bit set, and length = 31.

The last bit is also generated with the DMA count terminates. Suppose the DMA count is 2.2 K, and 1 K buffers are in use. The last buffer will be posted with 0.2 K length and the last bit set.

21.11.3 Last Bit Usage rules.

When the device is the source (writing to RAM) it uses the following rules:

Table 21.18 Last Bit Rule for Source

WQE LAST	WR LAST	RULE
0	0	Use length in Work Request. Only receive when available buffer size >= MTU.
0	1	Use length in Work Request. Enable receive without waiting for MTU's worth of buffer space. Received packet should be less than or equal to WR length. If there is data overflow, then error in system. If an MTU's worth of data comes in, and the buffer is greater than an MTU, store the data, and decrease the available buffer by an MTU. Do not retire the buffer. If a short packet comes in, or the buffer is filled, retire the buffer. Send out CN with last bit set.
1	X	Use length in WQE. Enable receive without waiting for MTU's worth of buffer space. Received packet should be less than or equal to WQE length. If there is data overflow, then error in system. If an MTU's worth of data comes in, and the buffer is greater than an MTU, store the data, and decrease the available buffer by an MTU. Do not retire the buffer. If a short packet comes in retire, or the buffer is filled, the buffer. Send out CN with last bit set.

When the device is the Sink (reading from RAM) it uses the following rules:

Table 21.19 Last Bit Rule for Sink

WQE LAST	WR LAST	RULE
0	0	Use length in Work Request. Only Transmit when available buffer size >= MTU.
0	1	Use length in Work Request. Enable transmit and transmit until end of buffer. If buffer is bigger than MTU, send out an MTU. Decrease available buffer by an MTU. Do not retire buffer. If buffer is less than MTU, a short packet goes out. If buffer is off length zero, a ZLP goes out. Send out CN with last bit not set.
1	X	Use length in WQE. Enable transmit and transmit until end of buffer. If buffer is bigger than MTU, send out an MTU. Decrease available buffer by an MTU. Do not retire buffer. If buffer is less than MTU, a short packet goes out. If buffer is of length zero, a ZLP goes out. Send out CN with last bit not set.

For DMA endpoints that are in DMA mode (block transfer enabled), the hardware will effectively “auto generate” the Last bit if the remaining portion of the current Work Request is greater than the Endpoint transfer count and the remainder is less than an MTU. If the packet that arrives overruns the buffer, then it is an error condition, and the overrun interrupt is generated.

A source must send out a Completion Notification with the LAST bit set if the LAST bit was set when the buffer was enqueued via the Completion Notification from the sink.

A sink must not send out a Completion Notification with the LAST bit set if the LAST bit was set when the buffer was enqueued via the Completion Notification from the souce.

If a source endpoint On Source if EP count goes to 0 before the current WR is filled, terminate transfer and retire the buffer (send out the CN with the last bit set).

On Source if EP count goes to 0 before the packet ends, set overflow, terminate transfer and retire the buffer (send out the CN with the last bit set).

On SINK if EP count goes to 0 before the current WR is emptied, set overflow, terminate transfer and retire the buffer (send out the CN without the last bit set), write no more data (into SIE/FMDU) then is in EP CNT.

21.12 Endpoint Control Registers

There are two kinds of endpoints. The first is a simple endpoint which has no DMA capability. Its Work Queue is one deep. It does not use the EP_CNT register.

The second kind of endpoint does have DMA Capability. Its Work Queue is four deep. It uses the EP_CNT registers.

If any error occurs in the middle of a DMA transfer, the transfer stops. The firmware may determine how much of the transfer was completed by reading the EP_CNT registers.

DMA is enabled by loading the EP_CNT registers and setting the BLK_XFR_EN bit. Firmware should not change the configuration setting that affect the block transfer, after BLK_XFR_EN bit is set to “1” or the block transfer has started.

21.12.1 Endpoint Descriptor Registers

Table 21.20 DMA Endpoint Descriptor

OFFSET	BYTE3	BYTE 2	BYTE 1	BYTE 0
0x0	INTERRUPT_MASK	INTERRUPT	EP_CTL_EXT	EP_CTL
0x4	Not Used	Not Used	MTU Size	
0x8	EP_CNT (DMA Transfer Count)			

Note: Only 12 bit MTU sizes need be supported in SEC2410/SEC4410

Table 21.21 EP Control Register

EP_CTL (RESET=0X00)			**EP DMA CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:5	WR_COUNT	R	Count of the number of Work Request in the Work Queue. For Non-DMA endpoints, the maximum WR_COUNT = 1.
4	FLUSH_DONE	R/W	This bit must be cleared by the Processor before issuing a FLUSH command to the work queue. This bit will be set when the flush is completed.
3	Reserved	R/W	Always read ‘0’
2	NO_LAST	R/W	If this bit is set, the last bit is not set on the last outgoing buffer. This does not affect how the endpoint deals with incoming buffers.
1	CONT_MODE	R/W	Continuous mode. When this bit is set, the DMA runs as long as there is data available. If this bit is not set, then BLK_XFER_EN controls transfer of data. For Non-DMA endpoints, this bit is always ‘1’.

0	BLK_XFR_EN	R/W	Setting this bit to “1”, initiates the block data transfer between the EP and RAM memory. BLK_XFR_EN bit should be set, only after the EP_CNT is loaded. For Non-DMA endpoints, or in continuous mode, this bit is always ‘0’. This bit is cleared, when any one of the following event occurs: The block data transfer is completed successfully. Any CRC error Flash programming error for SDC during data transfer An ECC error from the NDC

21.12.2 Continuos Mode

If in continuous mode CONT_MODE = ‘1’, the endpoint will move data as soon as it has a buffer. If not in continuous mode, the endpoint will not move data, no matter how many buffers are posted, unless the BLK_XFR_EN bit is set. CONT_MODE bit can only be cleared by the Processor.

In Continuous mode, if the last bit is set, it has two implications. The first is that the buffer should be used without waiting for an MTU worth of data. The second is that if the buffer ends on an MTU boundary, and SEND_ZLP bit is set, the endpoint will send out a ZLP after the buffer.

Table 21.22 EP Control Extended

EP_CTL_EXT (RESET=0X00)			**EP CONTROL EXTENDED REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:6	EP_TYPE	R/W	These bits define the endpoint type. These bits must be programmed when the UDC20 is configured. 00 - Control 01 - Isochronous 10 - Bulk 11 - Interrupt These bits have no meaning for devices that do not have multiple endpoint types.
5	Reserved	R	Always Read ‘0’
4	PERSISTANT_STALL	R/W	For endpoints that support Stalls. When set to a “1”, EP will respond with the STALL handshake appropriate to the protocol. This bit always reads ‘0’ on devices that do not support Stalls. For the SIE this bit can only cleared by the Processor. If USB host sends ClearFeatureEndpointHalt command, this bit is not affected. For EP0, Setup clears Stall bit as well
3	STALL	R/W	For endpoints that support Stalls. When set to a “1”, EP will respond with the STALL handshake appropriate to the protocol. This bit always reads ‘0’ on devices that do not support Stalls. For the SIE this bit is cleared by the hardware when any request is made to that EP or USB Reset occurs. For the SIE, writing a ‘0’ to this bit has no effect if the bit is high. If USB host sends SetFeautreEndpointHalt command, this bit is not affected. For EP0, Setup clears Stall bit as well
2	EP_RESET	R/W	This self clearing bit resets the endpoint. The reset flushes the work queue, and returns the endpoint to the idle condition. Note: For the SIE EP0_READ endpoint, the arrival of an USB setup packet sets this bit.

1	SEND_ZLP	R/W	Force Endpoint to send a ZLP when the end of transmission is indicated on the current endpoint, and the transmission ends on an MTU boundary.
0	Reserved	R	Always Read ‘0’

Table 21.23 EP Interrupt Register

INTERRUPT (RESET=0X00)			**EP INTERRUPT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	DEVICE_ERROR	R/W	When 1, this bit indicates that an error has occurred during the transfer, and the transfer did not complete. These are internal hardware errors that should never occur. These include reading an empty FIFO, writing a full FIFO, the state machines getting into illegal states etc. On receiving this error, the Processor has to determine the nature of the error by interrogating the device the endpoint is servicing. This bit is cleared by writing a ‘1’ to it.
6	RESIDUAL	R/W	When 1, this bit indicates that an there is a residual left in the terminal count. This means that a short packet, or a buffer with the last bit set occurred before the transfer count reached zero. This bit is only used in Block transfer mode. In Continuous mode, this bit is a Don’t Care in the IN direction. This bit is cleared by writing a ‘1’ to it.
5	OVERRUN	R/W	When 1, this bit indicates that an overrun has occurred on the Endpoint. The overrun can occur if the host reads or writes more than the available buffer size. In Continuous mode, this bit is a Don’t Care in the IN direction. This bit is cleared by writing a ‘1’ to it.
4	ZLP	R/W	Indicates that the device received a ZLP on this endpoint. This bit is cleared by writing a ‘1’ to it.
3	ACK	R/W	Indicates that the device responded with a ACK in response to a request to this endpoint. This bit is cleared by writing a ‘1’ to it.
2	NAK	R/W	Indicates that the device responded with a NAK in response to a request to this endpoint. This bit is cleared by writing a ‘1’ to it.
1	Reserved	R	Always read 0
0	BLK_XFR_COMPLET E	R/W	When ‘1’ indicates the current block transfer is complete. This bit is only set after the Completion Notification for the last byte transferred has been posted. Failure to do will result in race conditions. If an error occurs during the transfer, this bit is not set. Completions occur if EP_CNT goes to zero, and none of the error bits (RESIDUAL, OVERRUN) are set, or no device error occurred to abort the transfer. This bit is cleared by writing a ‘1’. This bit is also reset when the BLK_XFER_EN is set. Always ‘0’ for simple endpoints. This bit is always ‘0’ in continuous mode. This bit is cleared by writing a ‘1’ to it.

Table 21.24 EP Interrupt Mask Register

INT_MSK (RESET=0XFF)			**EP INTERRUPT MASK REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7	ERROR	R/W	When ‘1’, prevents the generation of this interrupt.
6	RESIDUAL	R/W	When ‘1’, prevents the generation of this interrupt.
5	ZLP	R/W	When ‘1’, prevents the generation of this interrupt.
4	OVERRUN	R/W	When ‘1’, prevents the generation of this interrupt.
3	ACK	R/W	When ‘1’, prevents the generation of this interrupt.
2	NAK	R/W	When ‘1’, prevents the generation of this interrupt.
1	Reserved	R	Always read ‘1’
0	BLK_XFR_COMPLETE	R/W	When ‘1’, prevents the generation of this interrupt.

Table 21.25 MTU Size Register

MTU_SIZE (RESET=0X0000)			**MTU SIZE**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[15:12]	Reserved	R	Always read ‘0’
11:0	D[11:0]	R/W	MTU size.

Note: The MTU register must be programmed with appropriate size for the endpoint in use. For example, SIE EP2 must be set to 64 bytes in USB 1.1 mode, and 512 bytes in USB 2.0 mode. Incorrect programming will result in unpredictable behavior.

Table 21.26 EP Transfer Count Register

EP_CNT (RESET=0X00000000)			**EP TRANSFER COUNT REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
[31:0]	D[31:0]	R/W	DMA Transfer Count in Bytes

On Source if EP count goes to 0 before the current WR is filled, terminate transfer and retire the buffer and send out the Completion Notification with the last bit set. If the data keeps coming after the transfer count has gone to zero, that is, the transfer count goes to 0 before the packet ends, set overflow, terminate transfer and retire the buffer and send out the CN with the last bit set.

On SINK if EP count goes to 0 before the current WR is emptied, set overflow, terminate transfer and retire the buffer (send out the CN without the last bit set), write no more data (into SIE/FMDU) then is in EP CNT.

21.13 Usage Examples

21.13.1 SIE EP2 to FMDU under Processor control Example

For this example assume an auto transfer of USB OUTs between the SIE EP2 and the FMDU. The host first sends down a CBW, with a request for a 64 K transfer. The Processor gets the CBW, programs the 64 K transfer, waits for the completion, and sends a CSW, back to the USB host. For this example the MTU size is 512 bytes for both the SIE and FMDU. Here is the sequence of events:

1. In order to accept the CBW, the Processor must first create a buffer in memory to accept the CBW. Assume that buffer is at location 4321, and the length is 31 bytes, and the processors reference i.e. context is 77.
2. The Processor flushes all queues to be used. This puts all queues in a known state.
3. Once the Work Request entry has been created, and the work queues initialized, the Processor can push a WQE on the WQ. It does this by writing a 6 to the WQ for SIE EP2 Write (OUT) endpoint. The address is SIE Base address + 0x190.
4. When the CBW arrives, the Processor receives a completion notification. In the completion notification, the context of 77 tell the Processor the ID of the WR that was fulfilled. On receiving the notification, the processor parses the CBW and determines that the transfer is an OUT of 64 Kbytes. If the incoming packet is bigger than 31 bytes, it means the protocol is broken. The SIE hardware stops accepting data after 31 bytes, sets the OVERRUN bit and stops. It signals Processor that an error has occurred through an interrupt and status.
5. The processor programs the appropriate LUN in the FMDU with the data from the CBW.
6. For this example, the Processor sets up three buffer in memory for the auto transfer. The buffers are BUFF_A, BUFF_B and BUFF_C. Assume that BUFF_A is at location 3000, with length 700, BUFF_B is at location 4000 with length 400, and BUFF_C is at location 5000 with length 600.
7. These three buffers are converted to WR and written into the WRLUTs of the SIE and FMDU. **In reality, these buffers are assigned at initialization time**.
8. Assume that in the FMDU that the buffers are assigned in the following order: BUFF_A = WR0, BUFF_B = WR1, and BUFF_C = WR2 in the FMDU WRLUT.
9. Assume in the SIE that the buffers are assigned in the following order: BUFF_A = WR10, BUFF_B = WR12, and BUFF_C = WR14, in the SIE WRLUT.

10. BUFF_A in the FMDU WRLUT WR0 looks like the following:
    a. Data Reference = 3000- The address of BUFF_A
    b. Length = 700- The length of BUFF_A
    c. Context = 10- The location of BUFF_A in the SIE WRLUT
    d. Completion Address = SIE_EP2- Communication is with SIE EP2 (Write)
    e. Last = 0- Keep going.

11. BUFF_A in the SIE WRLUT WR10 looks like the following:
    a. Data Reference = 3000- The address of BUFF_A
    b. Length = 700- The length of BUFF_A
    c. Context = 0- The location of BUFF_A in the FMDU WRLUT
    d. Completion Address = FMDU- Communication is with SIE EP2 (Write)
    e. Last = 0- Keep going.

12. BUFF_B in the FMDU WRLUT WR1 looks like the following:
    a. Data Reference = 4000- The address of BUFF_B
    b. Length = 400- The length of BUFF_B
    c. Context = 12- The location of BUFF_B in the SIE WRLUT

d. Completion Address = SIE_EP2- Communication is with SIE EP2 (Write)
e. Last = 0- Keep going.

13. BUFF_B in the SIE WRLUT WR12 looks like the following:
    a. Data Reference = 4000- The address of BUFF_B
    b. Length = 400- The length of BUFF_B
    c. Context = 1- The location of BUFF_B in the FMDU WRLUT
    d. Completion Address = FMDU- Communication is with SIE EP2 (Write)
    e. Direction = 1- Data from Device to RAM
    f. Last = 0- Keep going.

14. BUFF_C in the FMDU WRLUT WR2 looks like the following:
    a. Data Reference = 5000- The address of BUFF_C
    b. Length = 600- The length of BUFF_C
    c. Context = 10- The location of BUFF_C in the SIE WRLUT
    d. Completion Address = SIE_EP2- Communication is with SIE EP2 (Write)
    e. Last = 0- Keep going.

15. BUFF_C in the SIE WRLUT WR10 looks like the following:
    a. Data Reference = 5000- The address of BUFF_C
    b. Length = 600- The length of BUFF_C
    c. Context = 2- The location of BUFF_C in the FMDU WRLUT
    d. Completion Address = FMDU- Communication is with SIE EP2 (Write)
    e. Last = 0- Keep going.

16. Once the WRLUTs are loaded, the Processor loads the SIE EP2 CNT and FMDU EP0 CNT registers with 64K, the length of the transfer.
17. The Processor then pushes 3 WQE onto the WQ of SIE EP2. In this example it writes 10, 12, and 14 to the SIE EP2 Write WQ.
18. The SIE EP2 sees that it has buffers available. It starts accepting data. The first 512 byte packet gets put into BUFF_A. BUFF_A is still not complete. The next 512 bytes come in. The hardware puts 188 bytes into BUFF_A which completes all 700 bytes of BUFF_A. The next 324 bytes go into BUFF_B which becomes partially complete.
19. As soon as BUFF_A is complete, a completion notification goes out. BUFF_A is WR10 in the SIE WRLUT. The completion address is FMDU, and the context is 0.
20. The FMDU gets the Completion Notification as a Work Queue Element. The WQE references WR0. The FMDU references its WRLUT and gets the correct WR (BUFF_A). In the case of the FMDU, the direction is from RAM. The FMDU fetches 512 bytes and writes it out to FLASH memory. The FMDU cannot release BUFF_A because there is 188 bytes of data left in it.
21. As soon as the SIE is done with BUFF_B, it posts the completion notification. When the FMDU gets the notification it has enough data to send out a second MTUs worth of data to FLASH memory. Once the second MTU goes out, the FMDU has two completion notifications to send out. The first one is for BUFF_A which is WR0 in its WRLUT. By the programming values, the completion address is the SIE EP2 Write, and the context is 10. The second completion notification is BUFF_B which is WR1 in the FMDU WRLUT. The completion address is SIE EP2 Write, and the context is 12.
22. The completion notification from the FMDU is seen as WQE writes to it Work queue by the SIE. While the SIE is in the process of filling BUFF_C, it has BUFF_A and BUFF_B posted to it work queue.

23. This figure eighth traffic as shown in the data flow diagram continues until the full 64 Kbytes are transferred. If the 64 KByte transfer finishes in the middle of a buffer, that buffers completion notification will go out with the last bit set, and the length equal to the actual length.
24. The first completion notification will come from the SIE EP2 because it is the source. Once the SIE EP2 counter is complete, it will start NAKing all packets until it is re-enabled. The processor will probably mask the source completion interrupt.
25. At the end of the transfer, the Processor gets a block transfer complete interrupt from the sink, in this case the FMDU. It is a requirement that the interrupt be generated after the completion notification goes out for the last byte of the transfer.
26. Once the block transfer is complete, the Processor creates a CSW in memory. The Processor creates a WR in the SIE WRLUT, then it posts a WQE to SIE EP2 IN endpoint. In reality the setup will be done while the block transfer is in progress, and the Processor will just post the WQE.
27. If during the transmission a short packet arrived on SIE EP2, the completion notification would have the Last bit set. This would indicate that there was a problem during the transmission. The short packet would cause the FMDU terminate the DMA and signal the processor that an error occurred.
28. At the end of the transfer, there are three buffers sitting in the SIE EP2 Work Queue with the endpoint disabled. The Processor must flush these buffers before the next transaction.

21.13.2 SIE EP2 to FMDU under AutoCBW OUT Example

For this example assume an auto transfer of USB OUTs between the SIE EP2 and the FMDU. The host first sends down a CBW, with a request for a 64K transfer. The CBWP gets the CBW, programs the 64K transfer, waits for the completion, and sends a CSW, back to the USB host. For this example the MTU size is 512 bytes for both the SIE and FMDU. In the FMDU, the data transfer occurs through endpoint 0, and the control to the CBWP is done through endpoint 1. To avoid unnecessary NYETing of CBWs from the USB host, it is a requirement that the Auto CBW processor always posts two buffers in order to ACK the CBW when it is sent. To accomplish this, an extra WRLUT entry is duplicated for this purpose. Here is the sequence of events:

1. The Processor sets up 3 data buffers in memory. Each data buffer is 512 bytes long. For this example, the three buffers are BUF_A at location 0x008048, BUF_B at location 0x008248, and BUF_C at location 0x008448. Each buffer length is 0x200. There is additionally a CBW buffer (CBW_BUF) at 0x00F400, length 0x20, and a CSW buffer (CSW_BUF) at location 0x00F420. In Auto mode, the CBW buffer and CSW buffers are in the FMDU address space and not in main memory.
2. Having allocated the buffers, the Processor sets up the WRLUT in both the FMDU and the SIE. In the FMDU, WRLUT entries are assigned in the following way:

s. Entry 0:CBW buffer - ‘CBW_BUF’, CN Address = SIE EP2
t. Entry 1:Data buffer 0 - ‘BUF_A’, CN Address = SIE EP2
u. Entry 2:Data buffer 1 - ‘BUF_B’, CN Address = SIE EP2
v. Entry 3:Data buffer 2 - ‘BUF_C’, CN Address = SIE EP2
w. Entry 4:CSW buffer - ‘CSW_BUF’, CN Address = SIE EP2

3. In the SIE WRLUT entries are assigned in the following way:

x. Entry 0:CBW buffer - ‘CBW_BUF’, CN Address = FMDU EP1
y. Entry 1:Data buffer 0 - “BUF_A’, CN Address = FMDU EP0

z. Entry 2:Data buffer 1 - ‘BUF_B’, CN Address = FMDU EP0

aa. Entry 3:Data buffer 2 - ‘BUF_C’, CN Address = FMDU EP0

ab. Entry 4:CSW buffer - ‘CSW_BUF’, CN Address = FMDU EP1

ac. Entry 5:Data buffer 0 - “BUF_A`’, CN Address = FMDU ***EP1***

Note: BUF_A and BUF_A` are the same buffer with different CN addresses.

4. Once the tables have been initialized, the Processor flushes all queues to be used. This puts all queues in a known state, the AutoCBW processor is enabled.
5. CBWP post a WQE to receive the CBW. It does this by writing a 0x80 to the WQ for SIE EP2 Write (OUT) endpoint. Bit 7 is set to force the Last bit, and the 6 because of the WRLUT entry number. The address is SIE Base address + 0x190. The CN address for entry 0, is the FMDU Endpoint 1 read. This CN ensures that the CN comes to the CBWP, and not to the data endpoint. Additionally, the CBWP posts a data buffer (0x5) to the same SIE queue. Two buffers are posted to allow SIE to ACK USB OUT requests. The CN address for the data buffer is FMDU EP1_READ, the CBWP control endpoint.
6. When the CBW arrives, the CBWP receives a completion notification. In the completion notification, the context of 0 tell the CBWP the ID of the WR that was fulfilled. The CBW is sitting in the internal memory of the FMDU and not main memory because of the address chosen. On receiving the notification, the processor parses the CBW and determines that the transfer is an OUT of 64Kbytes. If the incoming packet is bigger than 31 bytes, it means the protocol is broken. The SIE hardware stops accepting data after 31 bytes, sets the OVERRUN bit and stops. It signals Processor that an error has occurred through an interrupt and status.
7. The CBWP programs the appropriate internal registers, and the FMI interface with the data from the CBW, and the information from the LUN Descriptor table for the LUN being addressed. The EP_CNT for FMDU EP0_READ is loaded with the length of the transfer.
8. At this point, BUFF_A is sitting at the SIE EP2_WRITE queue. In all proabability, the buffer is still empty. At most, it is starting to get filled. Since the operation is an OUT, the CBWP waits for BUF_A` to to be posted to it. When that buffer arrives, the CBWP reposts that buffer to the FMDU (itself at EP0), but this time the buffer used is enrty #1. The CBWP then posts BUFF_B and BUFF_C to SIE_EP2_WRITE queue. If the transfer size is less than 3 sectors, less buffers are posted. The CN address for all the data buffers is the FMDU EP0_READ work queue.
9. The order in which the buffers are assigned is irrelevant.
10. As each buffer is filled, the SIE sends CNs to the FMDU. The CN comes in as a work request to the FMDU endpoint 0. The FMDU reads the data out of memory and sends to the the target device.
11. As each buffer is consumed by the device, EP_CNT is decremented by the size of the buffer. The consumed buffer then goes out as a CN back to the SIE. This continues until the EP_CNT goes to zero, at which point the endpoint is disabled. A disabled endpoint does not post more completion.
12. Once the EP_CNT has gone to zero, and the CBWP has received an acknowledge from the card controller, the CBWP writes the CSW buffer and posts a CN for the CSW. In this examble, it does this by writing a 0x84 to the SIE EP2_READ work queue.
13. After the USB host reads the CSW, the SIE sends a CN to the FMDU. In this example, it does this by writing a 0x4 to FMDU EP1 endpoint. The CN for the CSW is the last act of the transfer.
14. The cycle is repeated by the CBWP posting a buffer for the CBW at the SIE EP2_WRITE queue.

21.13.3 SIE EP2 to FMDU under AutoCBW IN Example

For this example assume an auto transfer of USB INs between the SIE EP2 and the FMDU. The host first sends down a CBW, with a request for a 64K transfer. The CBWP gets the CBW, programs the 64K transfer, waits for the completion, and sends a CSW, back to the USB host. For this example the MTU size is 512 bytes for both the SIE and FMDU. In the FMDU, the data transfer occurs through endpoint 0, and the control to the CBWP is done through endpoint 1. Here is the sequence of events:

1. The Processor sets up 3 data buffers in memory, and programs the SIE and FMDU WRLUTs just like the previous example.
2. Once the tables have been initialized, the Processor flushes all queues to be used. This puts all queues in a known state, the AutoCBW processor is enabled.
3. CBWP post a WQE to receive the CBW. It does this by writing a 0x80 to the WQ for SIE EP2 Write (OUT) endpoint. Bit 7 is set to force the Last bit, and the 6 because of the WRLUT entry number. The address is SIE Base address + 0x190. The CN address for entry 6, is the FMDU Endpoint 1read. This CN ensures that the CN comes to the CBWP, and not to the data endpoint. Additionally, the CBWP posts a data buffer (0x5) to the same SIE queue. Two buffers are posted to allow SIE to ACK USB OUT requests. The CN address for the data buffer is FMDU EP1_READ.
4. When the CBW arrives, the CBWP receives a completion notification. In the completion notification, the context of 0 tell the CBWP the ID of the WR that was fulfilled. The CBW is sitting in the internal memory of the FMDU and not main memory because of the address chosen. On receiving the notification, the processor parses the CBW and determines that the transfer is an IN of 64Kbytes. If the incoming packet is bigger than 31 bytes, it means the protocol is broken. The SIE hardware stops accepting data after 31 bytes, sets the OVERRUN bit and stops. It signals Processor that an error has occurred through an interrupt and status.
5. At this point, BUFF_A` is sitting at the SIE EP2_WRITE queue. To recover this buffer, the CBWP issues a FLUSH to SIE EP2_WRITE work queue. This releases the buffer.
6. The CBWP programs the appropriate internal registers, and the FMI interface with the data from the CBW, and the information from the LUN Descriptor table for the LUN being addressed. The EP_CNT for FMDU EP0_WRITE is loaded with the length of the transfer.
7. Having issued the FLUSH the CBWP has control of all three data buffers. Since the operation is an IN, the CBWP posts BUFF_A, BUFF_B and BUFF_C to FMDU EP0_WRITE queue. If the transfer size is less than 3 sectors, less buffers are posted. The CN address for all the data buffers is the SIE EP2 READ work queue.
8. As each buffer is filled, the FMDU sends CNs to the SIE. The CN comes in as a work request to the SIE EP2_READ . The SIE reads the data out of memory and sends it to the USB host on INs. As each buffer is consumed, the SIE send CNs for the buffer back to the FMDU for reuse.
9. As each buffer is sent to the SIE, EP_CNT is decremented by the size of the buffer. This continues until the EP_CNT goes to zero, at which point the endpoint is disabled. A disabled endpoint drops any work requests coming to it. A disabled endpoint also does not post completion.
10. Once the EP_CNT has gone to zero, and the CBWP has received an acknowledge from the card controller, the CBWP writes the CSW buffer and posts a CN for the CSW. In this examble, it does this by writing a 0x87 to the SIE EP2_READ work queue. The CBWP does not have to wait for the data to be sent to the host before sending the CSW because the CSW is enqueued behind the data because of the enforced ordering.
11. After the USB host reads the CSW, the SIE sends a CN to the FMDU. In this example, it does this by writing a 0x4 to FMDU EP1 endpoint. The CN for the CSW is the last act of the transfer.
12. The cycle is repeated by the CBWP posting a buffer for the CBW at the SIE EP2_WRITE queue.

Chapter 22 Bootloader

The secure bootloader in the SEC2410/SEC4410 provides the ability to boot and execute a firmware image from multiple media sources in both secure and non-secure modes. The secure and non-secure boot modes search for a firmware boot image on the available boot sources: SPI Flash or Flash Media (SD1/MMC1 and SD2/MMC2). If a valid image is found, the bootloader will execute the image. The secure bootloader also provides the functionality to download a firmware image over USB into internal memory for execution. The firmware boot image details are outlined in the *Secure Bootloader Reference Guide* [7].

The secure boot mode requires that a firmware boot image is stored on a boot source in an encrypted state that is authenticated and decrypted for execution in the device. The encryption and authentication keys for the secure boot will be stored in a lockable store in OTP. The keys in the OTP are provisioned by the manufacturer prior to the initial boot image being stored on the boot source. The secure boot mode does not allow code to execute in place on the external SPI Flash, rather the image is loaded into the instruction SRAM from the external source.

The secure bootloader also provides the functionality to update the firmware boot image on the media source by a USB host. Through various procedures, the bootloader can enter firmware upgrade mode, which provides the capability for an application on the USB host to store a valid boot image on the source media. The bootloader detects that it should not attempt to load the image in the media source, but instead enter firmware upgrade mode.

The non-secure boot mode does not require authentication of the boot image and can be executed from the external SPI Flash. The firmware image can either be solely copied to instruction SRAM and only execute from instruction SRAM or it can be partially located and executed in the external SPI Flash and instruction SRAM.

22.1 Boot Process

22.1.1 Booting from Multiple Devices

The bootloader supports booting from multiple sources. The bootloader will attempt to boot from the acceptable devices in the following order:

1. SPI Flash
2. SD1/MMC1
3. SD2/MMC2

22.1.2 Locating the Firmware Image

After power on reset, the SEC2410/SEC4410 initially boots to internal ROM. Code execution begins by searching for a valid firmware image on one of the acceptable boot devices. The presence of a valid boot image is searched in the following order:

1. The bootloader checks to see whether firmware update mode has been enabled. When enabled, firmware update is forced and the bootloader will not attempt to boot a firmware image. In this case, the bootloader ends the boot process and switches to the firmware update mode. If this mode has not been enabled, the bootloader will continue with the boot process.
2. The OTP security configuration is the next item checked by the bootloader. If a valid signature is detected, the bootloader continues to step 3. Otherwise, non-secure mode is selected and the bootloader continues to step 4.
3. In secure mode, the bootloader firmware checks an additional bit to determine whether secure mode was selected. If set, the bootloader with continue in secure mode; if not set, the bootloader with proceed in non-secure mode.

4. Starting with the SPI Flash device the bootloader checks whether the device is present; if present it initializes the SPI Flash interface and continues to step 7.
5. If the SPI Flash device does not initialize successfully or is not present, the bootloader repeats step 4 with the next device (the order is provided in Section 22.1.1).
6. If the bootloader is unable to detect and initialize a device, the bootloader ends the boot process and switches to the firmware update mode.
7. The bootloader firmware checks whether the media device contains a valid firmware image using the method outlined in Section 22.1.2.2: *Validate Firmware Image Existence*. If a valid firmware image exists, the bootloader continues to the next step. If a valid image is not found, step 3 is repeated with the next device. If there are no remaining devices to check, the bootloader ends the boot process and switches to the firmware update mode.

22.1.2.1 Detect and Initialize Devices

The bootloader will attempt to detect and initialize the available devices. The boot devices will be initialized based on the method specific to the boot device type. These methods are outlined in the *Secure Bootloader Reference Guide* [7].

SPI Flash:

The SPI Flash device will be detected and initialized using the String ID in the Media Layout Descriptor (MLD). The existence of the String ID will verify that an external SPI Flash device exists and that the SPI has been initialized for read/write access to the device. Since the SPI controller supports SPI interface modes 0 and 3 and supports dual read mode, four attempts to read the String ID are made.

1. Mode 0, dual output mode (opcode 0x3b)
2. Mode 3, dual output mode (opcode 0x3b)
3. Mode 0, single output mode (opcode 0x0b)
4. Mode 3, single output mode (opcode 0x0b)

If the String ID is not read correctly after the 4th configuration then it is assumed a valid SPI Flash device is not available. This can occur if either a SPI Flash device is not connected or the MLD has not been programmed in the SPI Flash memory.

SD/MMC Media:

The SD/MMC media devices will be initialized per the SD/MMC specification. The media devices will be detected as present based on the state of the card detect GPIO.

22.1.2.2 Validate Firmware Image Existence

The existence of a valid firmware image on the boot device is validated with the following sequence.

1. The bootloader reads the media layout descriptor from the boot device.
2. The bootloader then checks that the String ID is valid by comparing the expected string with contents in the media layout descriptor. If the String ID matches, the process continues to step 3. If the String ID does not match then a valid firmware image does not exist on this device and the process ends. The bootloader will then move to the next device; otherwise if it is the last device, the bootloader will end the boot process and switch to the firmware update mode.
3. The bootloader checks that the media layout descriptor (MLD) is valid by calculating the digital signature of all fields of the MLD except the 16-byte signature. The MLD signature is calculated by using the special algorithm, where the signature will be produced by moving the MLD fields through the AES 16 bytes at a time. The result of the last 16-byte encryption will produce the signature that is compared to in the signature field of the MLD. If the signature is valid continue to setup 4. Otherwise the bootloader will move to the next device. If it is the last device, the bootloader will end the boot process and switch to the firmware update mode.

4. The bootloader finds the firmware region in the MLD by searching for a used region entry with the firmware image type region identifier. If the firmware region is not discovered the bootloader will move to the next device. If it is the last device, the bootloader will end the boot process and switch to the firmware update mode. If the firmware region is discovered, continue to step 5.
5. The bootloader gets the logical block address (LBA) on the boot device of the firmware on by reading the start LBA from the firmware region in the MLD. The start address is recorded since the existence of a firmware image has been verified.

22.1.3 Secure Boot

In secure mode, the bootloader performs the following sequence:

1. The bootloader reads the MAC and AES keys from the OTP secure store.
2. The bootloader calculates the CMAC subkeys K1 and K2 based on the method defined in RFC 4493 [15].
3. The bootloader reads the firmware image header and the clear text header and validates and decrypts the contents of the encrypted firmware image header.
4. The bootloader loads the firmware image into RAM. During this process, it calculates the MAC and decrypts the image as it is loaded into memory.
5. Once entire image is loaded into memory the bootloader validates the computed MAC against the MAC supplied in the firmware package. If the calculated MAC matches, the loaded image continues to boot. If the MAC does not match, then secure boot for this device ends.

22.1.3.1 Firmware Header Validation Sequence

The bootloader performs the firmware header validation in the following order:

1. The bootloader reads the firmware image header and clear text header from the boot device. The bootloader then validates and decrypts the contents of the firmware image header.
2. The AES block is programmed with the stored MAC key, IV, and counter value.
3. The last 16-byte block of the header is determined and is XORed with the K2 subkey. All contents of the firmware image header except the header MAC field to the MAC cipher are then applied.
4. The calculated CMAC is checked to see whether it matches the provided CMAC in the firmware image header. If the CMAC matches, then the header is valid and proceeds to step 5 to decrypt the header. If it is not valid, then the secure boot process ends and the device enters the firmware update mode.
5. The AES block is then programmed with the stored AES key, IV, and counter value.
6. All contents of the firmware image header except the header MAC field to the AES cipher to produce the decrypted contents of the header.

22.1.3.2 Firmware Load with Authentication and Decryption Sequence

For a secure load, the bootloader will traverse the Load Section Entry Table in the decrypted firmware image header and load each section entry into RAM. There, it processes through the MAC cipher, then is decrypted in the AES Cipher and copied into correct memory location.

1. Starting from the first load section entry, the bootloader checks whether the section is marked as in use and whether the section is marked to be copied into the designated destination address. If these both of these conditions are true, the bootloader proceeds to step 2; otherwise it proceeds to step 5.
2. The bootloader obtains the size, image source location (LBA and offset), and memory address location for the load section. It then checks that the state load section size does not exceed the size of the identified memory.

3. Block data is then read from the load region's boot image source location.
4. The AES block is programmed with the MAC Key setup that includes IV, counter, and nonce values.
5. The block of data is checked whether it is the last 16-byte block of data in the firmware image. If it is not the last block of data in the firmware image, continue to step 6. Otherwise, if it the last block of data the block is XORed with the subkey K2, and the bootloader continues to step 6.
6. The block of data is applied to the MAC cipher where the result is stored in the MAC cipher buffer.
7. The current value of the IV and Counter is read in the AES block and cached in the MAC key setup.
8. The AES block is then programmed with the AES Key setup that includes IV, counter, and nonce values.
9. The block of data is applied to the AES cipher to decrypt the block where the result is stored in the AES cipher block.
10. The current value of the IV and Counter in the AES block is read and cached in the AES key setup.
11. The encrypted contents are copied into the memory destination location based on the specified destination size.
12. If the load section exceeds the size of the 512-byte block, the bootloader will increment to the next block for the source and destination and repeat step 3 until the complete contents of the load section have been loaded. If the complete contents have been loaded then the bootloader proceeds to the next step.
13. The next load section is incremented and if it not the last load section then it repeats from step 1 to begin loading the section. Otherwise the load is complete, and the decrypted contents are stored in the corresponding memory location and the MAC cipher buffer contains the calculated MAC.

22.1.4 Non-Secure Boot

For a non-secure boot, the boot image that is loaded can execute in the external SPI Flash and/or execute from internal instruction SRAM. The non-secure bootloader will read the firmware image header and load the contents of each valid load section.

1. The bootloader starts at the first load section entry and checks whether the section is marked as in use and marked to be copied into the designated destination address. If these conditions are true, the bootloader proceeds to step 2; otherwise it proceeds to step 5.
2. The bootloader obtains the size, image source location (LBA and offset), and memory address location for the load section. It then checks that the state load section size does not exceed the size of the identified memory. If it does exceed the memory size, then it is considered an invalid boot image and the bootloader exits to the firmware update mode. Otherwise the bootloader continues to step 3.
3. Block data is then read from the load region's boot image source location and copied into the memory destination location based on the specified destination size.
4. If the load section exceeds the 512-byte block size, step 2 is repeated until the complete contents of the load section have been loaded. If the complete contents have been loaded, then the bootloader proceeds to the next step.
5. The bootloader increments to the next load section and if it is not the last load section then it repeats from step 1 to begin loading the section. Otherwise the load is complete.

Chapter 23 Clock system

23.1 Overview

The clock control block resides as an AMBA slave device on the System AHB bus at Address 0x62000400

23.2 Ring Oscillator

The ring oscillator is 15Mhz maximum. If the ring oscillator is turned off, any wake-up event will start it. Once it has started, the Processor can turn it off manually through the CLOCK_CTL register. To shutdown the ring oscillator, the Processor can do it through the CLOCK_CTL register.

The Processor always wakes up on the ring oscillator. If the processor switches to the PLL, either automatically or manually, it has no way to switch back to the ring oscillator.

Activity on any of the pins that can cause wake-up events will cause the ring oscillator to start running, If spurious wake up events occur, it will start the ring oscillator and wake up the processor. Once the ring oscillator starts, only the processor can shut it down.

23.3 PLL

The PLL frequency is 60Mhz. If the PLL is turned off, a USB wake-up event will start it. Once it has started, the Processor can turn it off manually through the CLOCK_CTL register. To shutdown the ring oscillator, the Processor clears the PLL_EN bit.

When switching from the ring oscillator to the PLL, the firmware just sets the PLL_ENABLE bit and waits for clock valid.

23.4 System Clock Shutdown:

To shutdown the ring oscillator, the processor clears the ROSC_EN bit. To shutdown the PLL the processor clears the PLL_EN bit. The ROSC_EN bit is set by a wake-up event.

23.4.1 System Clock Wake-up

If the clock is stopped, and a USB wake-up event is detected, and the SIE_POWER_EN is set, the following happens:

1. The ring oscillator is started automatically by the hardware.
2. The system clock source is set to the ring oscillator.
3. The firmware is free to access non-synchronous devices.
4. The PLL is started.
5. The hardware wait for the PLL to lock.
6. Once PLL locks, the system clock is switched to the PLL
7. The firmware must wait until the PLL is locked before using the SIE or other synchronous devices.
8. Non-synchronous devices can be accessed at any time.

If the clock is stopped, and a NON-USB wake-up event is detected, the following happens:

1. The ring oscillator is started automatically by the hardware.

2. The system clock source is set to the ring oscillator.
3. The firmware is free to access non-synchronous devices.
4. The PLL is NOT started.
5. The firmware can opt to stay on the ring oscillator.
6. The firmware can opt to enable the PLL.
7. If the PLL is enabled, the system clock is switched to the PLL, once the PLL locks
8. The firmware must wait until the PLL is locked before using the SIE or other synchronous devices.

If the PLL is running, it is always the clock source once lock is achieved.

23.5 Clock Control Registers

These registers control the clock generation as well as gating to various modules in the silicon.

Table 23.1 System Clock Control

CLOCK_CTL (0X0400 - RESET=0X06)			**MCU CLOCK CONTROL**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:3	Reserved	R	Always reads “0”.
2	ROSC_EN	R/W	“0” = Ring Oscillator Disable. “1” = Ring Oscillator Enable. ROSC_EN must be set to “1” before the MCU can be switched to the internal Ring Oscillator Clock source. Default value is “1”. Any wake-up event enables the ring oscillator automatically. This bit will remain on it an attempt is made to turn it off while wake-up events are occurring.
1	PLL_EN	R/W	Automatically enabled by the hardware on USB wake-up events, if the SIE_POWER_EN bit is set by firmware to enable the PLL., this bit will remain on if an attempt is made to turn it off while wake-up events are occurring.
0	PLL_STABLE	R	PLL 60 MHz stable ‘1’ = 60 MHz oscillator is stable. ‘0’ = 60 MHz oscillator is not stable.

23.5.1 Clock control block diagram.

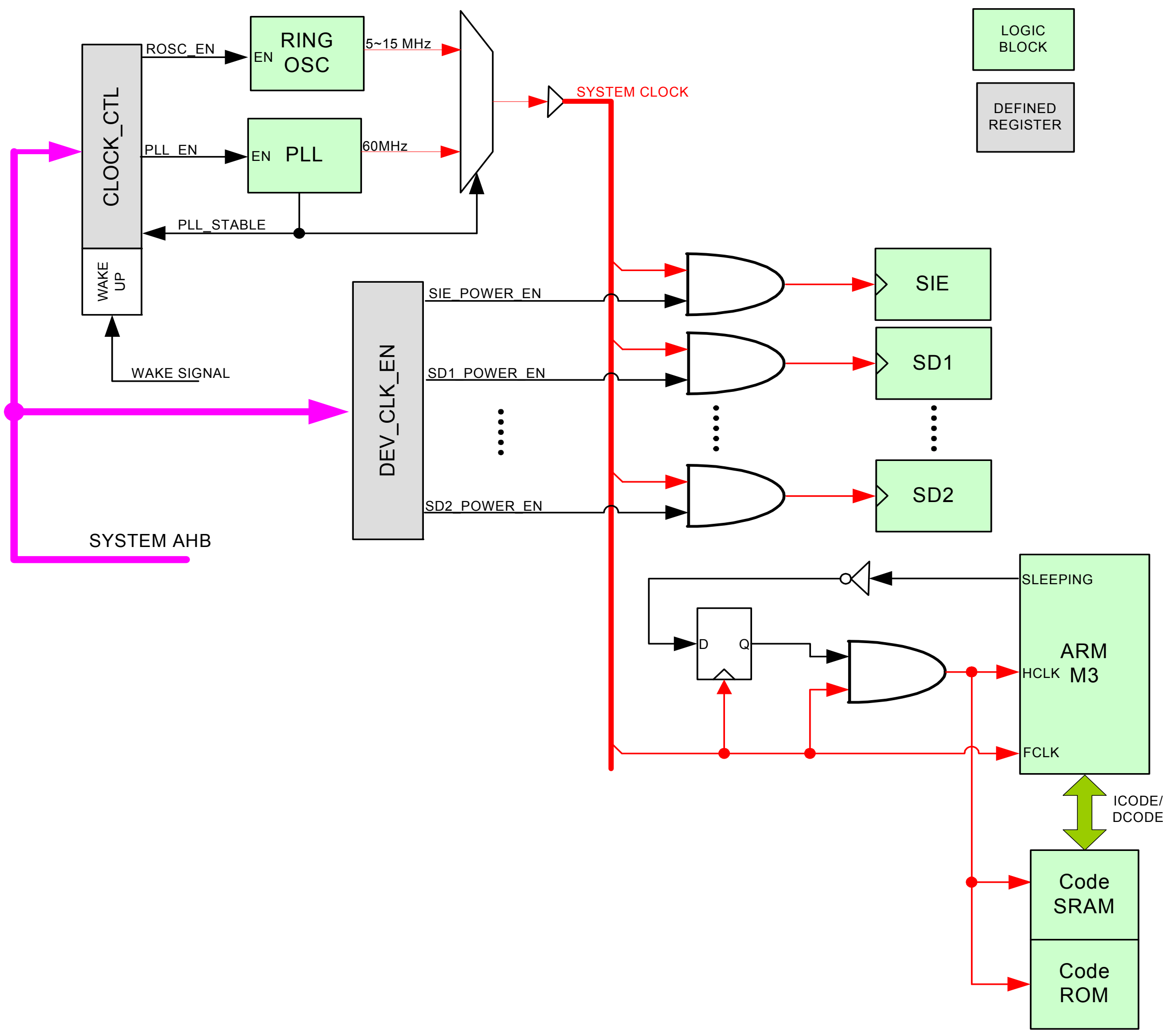


Figure 23.1 Clock Control Block Diagram

Note: The block diagram is meant as a logical representation of the clock control and gating circuitry. Actual implementation may be different.

Table 23.2 Device Clock Control Register

DEV_CLK_EN (0X0404 – RESET=0X00000000)			**DEVICE CLOCK ENABLE REGISTER**
BYTE	**NAME**	**R/W**	**DESCRIPTION**
31:15	Reserved	R	Always read ‘0’
14	TIMER_CLK_EN	R/W	Clock control for the Timer block
13	AMBA_SRAM_EN	R/W	Clock control for the AMBA SRAM
12	FMDU_CLK_EN	R/W	Clock control for the FMDU. When is bit is cleared, the control logic is shutdown.
11	AES_CLK_EN	R/W	Clock control for the AES. When is bit is cleared, the control logic is shutdown. This bit does not disable the SRAM.
10	SPI_CLDEK_EN	R/W	Clock control for the SPI master interface. When is bit is cleared, the control logic is shutdown, the interface is disabled.
9	SC_CLK_EN	R/W	Clock control for the SmartCard. When is bit is cleared, the control logic is shutdown, the interface is disabled.
8	UART_CLK_EN	R/W	Clock control for the UART. When is bit is cleared, the control logic is shutdown, the interface is disabled.
7	SIE_CLK_EN	R/W	Clock control for SIE. When this bit is cleared, the clocks to the SIE are shut down, and the SIE PHY is turned off. Even a USB resume condition will not wake the SIE or start the PLL. When this bit is turned on for the first time, there is a reset generated to the UDC20 to put it into a known state.
6:5	Reserved	R	Always Read ‘0’
4	SDC2_CLK_EN	R/W	Clock control for the SD2 Controller. When is bit is cleared, the control logic is shutdown, the interface is disabled.
3	SDC1_CLK_EN	R/W	Clock control for the SD1 Controller. When is bit is cleared, the control logic is shutdown, the interface is disabled.
2:1	Reserved	R	Always Read ‘0’

Setting these bit to a zero will causes device to enter the low-power state. The exact nature of what gets powered down will be done together with design.

23.6 Power States

SEC2410/SEC4410 works in one of four power states.

Table 23.3 SEC2410/SEC4410 Power States

POWER STATE	RING OSC	PLL	DESCRIPTION
P0	X	ON	PLL in running. Ring Oscillator may or may not be running. Clock tree is runnig at 60Mhz. All clocks are on, the device is fully active. Entry into this state is controllerd by firmware, or by reset. A wake up event will not change this state.
P1	X	ON	PLL in running. Ring Oscillator may or may not be running. Clock is gates to devices that are not in use. HCLK to the processor is stopped when Processor goes into sleep mode. All devices being clocked are at 60Mhz. Entry and exit into this state is controllerd by firmware. A wake up event will not change this state. Reset will cause transition to P0 State.
P2	ON	OFF	PLL in not running. Ring Oscillator is running. Clock is gates to devices that are not in use. HCLK to the processor is stopped when Processor goes into sleep mode. All devices being clocked are at the ring oscillator frequency of below 15Mhz. USB activity cannot be done in this state. Entry into this state is controllerd by firmware. A wake up event will restart the PLL, causing a transition to P1 State. Reset will cause transition to P0 State.
P3	OFF	OFF	PLL in not running. Ring Oscillator is not running. The chip is in the SUSPEND state. Entry into this state is controllerd by firmware. The only way to exit this state is a wake up event or reset. Reset will cause a transition to P0 state. A wake up event will cause transition to P1 state. Reset will cause transition to the P0 State.

23.7 Waking up from Suspend

Wake up for SEC2410/SEC4410 is done using ARM's Wake-up Interrupt Controller (WIC). Since the clocks are stopped in SUSPEND, there are only two ways to wake up from Suspend. A transition on an unmasked GPIO line, or a USB Resume Event. The GPIO transititions can be rising or falling. The transitions have masks in each direction. These masks have no effect on the normal GPIO interrupts that are generated for the NVIC during normal operation. These masks are exclusively used for determining the wake up signal.

If the processor is in deep sleep, the TIMER can be bring it out of deep sleep.

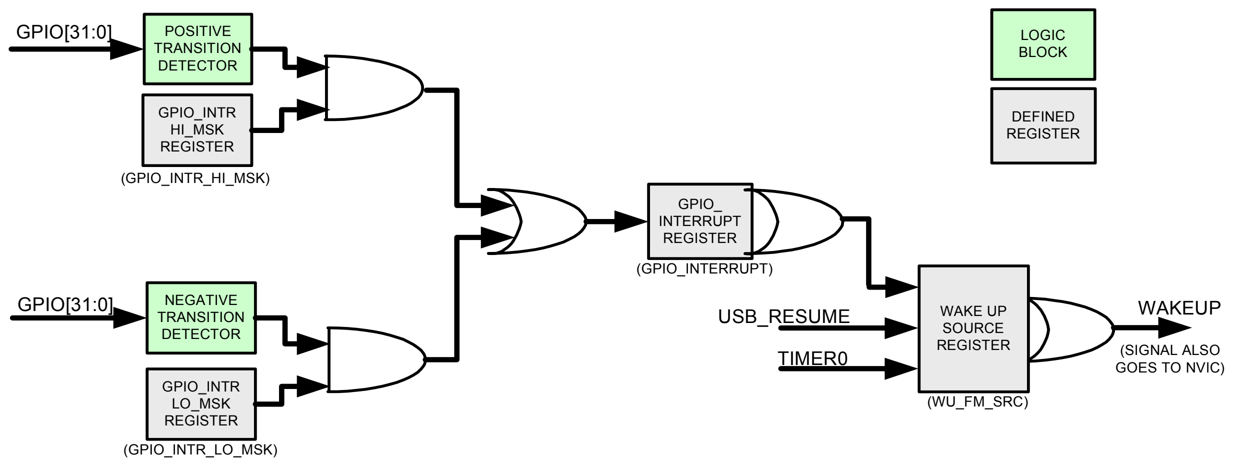


Figure 23.2 Wake Up Block Diagram

Note: See table 12.9 and 12.10 for definition of GPIO_INTR_LO_MSK and GPIO_INTR_HI_MSK registers.

Table 23.4 Wakeup Souce Register

WU_FM_SRC (0X0408 – RESET=0X00)			**WAKEUP & FROM SOURCE**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read “0”.
3	RESUME	R/W	This bit is set on detection of Global Resume state (when there is a transition from the "J" state while in Global Suspend). Write ‘1’ to clear this bit.
2	GPIO_INTR	R	This bit will be set, if there is any bit set in the GPIO_INTERRUPT register. This bit can only be cleared when all the unmasked bits in GPIO_INTERRUPT register (address 0x0410-0x0413) has been cleared.
1	SYS_BUSY	R/W	When the processor disables the PLL and Ring Oscillator, there is time while the system in busy shutting down. The processor is required to monitor this bit while the system is shutting down. On wakeup the processor must monitor this bit till it goes to ‘0’ before resuming normal processing. This bit is under hardware control and cannot be used to generate an interrupt because it is always masked. Write ‘1’ to clear this bit.
0	TIMER0	R/W	When this bit is set, it means a timer interrupt has occured.

Note: Bits are cleared by writing a ‘1’ to the corresponding bit.

Note: Unmasked Wakeup Source bits restart the processor when its clock is stopped. This restarts the Ring Oscillator and crystal oscillator for the processor to resume from <500µA operation.

Note: TIMER0 is there to allow for situations where the clock to the processor is stopped, but not to the timer. This allows the timer to wake the processor from deep sleep.

Table 23.5 Wakeup Souce Mask Register

WU_FM_MSK (0X040C – RESET=0XFF)			**WAKEUP & FROM SOURCE**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R	Always read '1
3	RESUME	R/W	When ‘1’, prevents the generation of this interrupt
2	GPIO_INTR	R/W	When ‘1’, prevents the generation of this interrupt
1	SYS_BUSY	R	Always read '1'
0	TIMER0	R/W	When ‘1’, prevents the generation of this interrupt

Table 23.6 GPIO Interrupt Register

GPIO_INTERRUPT (0X0410~0X0413- RESET=0X00000000)			**GPIO INTERRUPT REGISTER**
BIT	NAME	R/W	DESCRIPTION
31:0	GPIO[31:0]	R/W	1 = A level change has occurred on GPIO[31:0].

Note: This register is set and masked independantly of the Nested Vectored Interrupt Controller.

Note: Writing a “1” (one) to a bit clears the bit and enables the detection of the next level transition. If not masked by the corresponding bit in the GPIO_MSK register, “1” in any bit in this register will force a “1” on the GPIO_INTR on the WU_FM_SRC register

Note: Interrupts from the GPIOs can be programmed to be on the rising edge, falling edge or both. If both mask bits are set, the GPIO interrupts will not occur. The interrupt can only occur in the direction that is not masked. See Chapter 12 for description of mask registers.

23.8 Simulation Register

This register is provided for use with simulation. The firmware can write this register. The output of this register do not have any other function.

Table 23.7 Simulation Control Register 1

SIMULATION_CTL (0X0440 RESET=0X00000000)			**SIMULATION CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31	EOT	R/W	End of Test. Simulation firmware sets this bit to indicate the simulation is over.
30	FW_ERROR	R/W	Firmware Error. Simulation firmware sets this bit to indicate that it encounters an error
29:2	USER_DEF	R/W	These bits are user defined.
1:0	ENVIRONMENT	R	These bits allow the firmware to know the environment it is running under. These bits are forced. 00 - ASIC 01 - FPGA 10 - ASIC Simulation 11 - FGPA Simulation

23.9 Test control registers

This register additional test and configuration control.

Table 23.8 Clock Test Control Register 1

CLK_TEST_CTL (0X0700 RESET=0X00000005)			**CLOCK AND TEST CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:3	Reserved	R	Always read ‘0’
2	VREG_CLIM	R/W	Enables current limit on VARIO voltage regulator
1	UTMI_CLK_ALWAYS_ON	R/W	When this bit is set, the UTMI clocks will be clock gated by the SIE_CLK_EN register bit. When this bit is cleared, the UTMI clocks is always enabled.
0	RESUME_FILTER_EN	R/W	When this bit is set, the Card Reader USB wakeup/resume event (RESUME bit in WU_FM_SRC registers) is filtered for 8 clocks. When this bit is cleared, the Card Reader USB wakeup/resume event (RESUME bit in WU_FM_SRC registers) is not filtered.

Table 23.9 Device Reset Length Control Register 1

DEV_RESET_LEN (0X0704 RESET=0X00000097)			**DEVICE RESET LENGTH CONTROL REGISTER**
BIT	**NAME**	**R/W**	**DESCRIPTION**
31:16	Reserved	R	Always read ‘0’
15:0	DEV_RESET_CNT	R/W	Determines length of the soft reset pulse width. The pulse length is (DEV_RESET_COUNT – 1’b1) X ring_oscillator_period. Note that the ring oscillator period can vary from 67 ns to 200 ns. Firmware must program a value of at least 2.

Chapter 24 Reset state

There are two different resets that the device experiences. One is a hardware reset (either from the internal POR reset circuit or via the **RESET_N** pin) and the second is a USB Bus Reset.

A reset via the external nRESET pin as well as internal RESET generated by the POR module causes the following:

1. All registers are set to their default values.
2. All endpoints are disabled.
3. If SEC2410/SEC4410 was in power down state, then it is cleared.

The external crystal oscillator is allowed to run.

24.1 USB Bus Reset

When the SEC2410/SEC4410 is in SUSPEND mode with the clocks stopped, a USB reset will be recognized as a RESUME event (not a reset) and if the **WU_FM_SRC_1** bit for RESUME is unmasked, the SEC2410/SEC4410 will restart the clocks. Only while the clocks are running will a USB reset be detected. When recognized, the SEC2410/SEC4410 does the following:

1. All SIE endpoint registers are set to their power-on-reset (POR) values, all stall conditions, and the **SETUP** bit are cleared. The PID sequencers, internal DTOG are reset for all endpoints
2. The following registers will be set to their POR values: SIE_SRC, USB_CONF, and endpoint control registers.
3. If the **USB_RESET** in USB_STAT is unmasked, then a ISR_0 interrupt (**USB_STAT**) is generated to the processor.
4. The following registers will be set to their POR values: ISR_0, endpoint control registers.

Note: EP0 is always available once the device is powered and the USB Bus Reset has been received.

24.2 USB Reset in HSIC Mode

To initiate a reset, the HSIC host must drive the HSIC lines to the reset state. The reset state is achieved by pulling both **HSIC_STROBE** and **HSIC_DATA** low for a minimum of 10 ms. After exiting the reset state, the HSIC host goes through a reset recovery period of 10 ms where only SOFs are passed down to the HSIC device. After the reset recovery period, normal traffic is allowed to begin.

24.3 Internal POR Hardware Reset

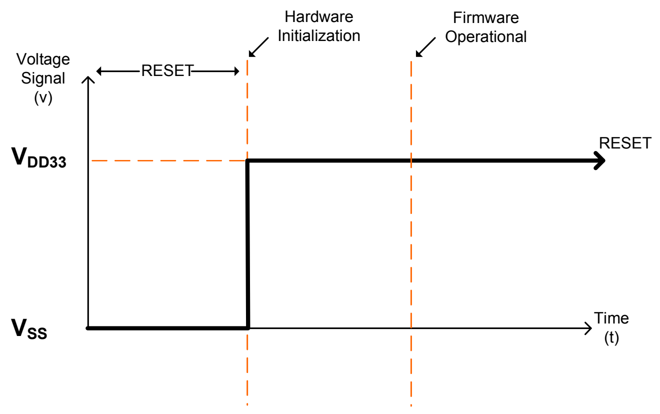
All reset timing parameters are guaranteed by design.

24.4 External Hardware Reset

A valid hardware reset is defined as assertion of **RESET_N** for a minimum of 1 µs after all power supplies are within operating range. While reset is asserted, the device (and its associated external circuitry) consumes less than 500 µA of current from the upstream USB power source.

Assertion of **RESET_N** (external pin) causes the following:

1. The PHY is disabled, and the differential pairs will be in a high-impedance state.
2. All transactions immediately terminate; no states are saved.
3. All internal registers return to the default state.
4. The internal crystal oscillator is halted.
5. The PLL is halted.



24.5 Soft Reset

There are three soft reset mechanisms in SEC2410/SEC4410. The first is DEV_RESET which is set by the firmware to reset the device. The second is the watchdog reset when enabled. The third is a LOCKUP signal from the processor.

Table 24.1 Reset control Register 1

RESET_CTL (0X0420 RESET=0X00)			**RESET CONTROL REGISTER 1**
BIT	**NAME**	**R/W**	**DESCRIPTION**
7:4	Reserved	R/W	Always read ‘0’
3	SYSRESETREQ	R/W	This bit is set when the Processor issues a reset request signal. It is equivalent to a DEV_RESET, with the exception that this bit itself is not cleared. It allow the firmware to identify that a system reset occured. A hard reset clears this bit. This bit cleared by writing a ‘1’ to it.
2	LOCKUP	R/W	This bit is set when the Processor issues a LOCKUP signal due to an unrecoverable error. It is equivalent to a DEV_RESET, with the exception that this bit itself is not cleared. It allow the firmware to identify that a lockup reset occured. A hard reset clears this bit. This bit cleared by writing a ‘1’ to it.
1	DEV_RESET	R/W	Setting this bit to “1”, sets all of the registers into a known state as specified in each register. This bit is cleared, when the RESET process has completed. This bit is self clearing
0	WD_RESET	R/W	This bit is set when the watchdog fires. It is equivalent to a DEV_RESET, with the exception that this bit itself is not cleared. It allow the firmware to identify that a watch dog reset occured. A hard reset clears this bit. This bit cleared by writing a ‘1’ to it.

Setting the DEV_RESET bit to 1, is equivalent to asserting the RESET pin.

24.6 State of the pins after reset

Figure 24.1 Pin Reset States

Table 24.2 Legend for Pin Reset States Table

SYMBOL	DESCRIPTION
Y	Hardware enables function
0	Output driven low
1	Output driven high
--	Hardware disables function
Z	Hardware disables output driver (high impedance)
PU	Hardware enables pull-up
PD	Hardware enables pull-down
HW	Hardware controls function, but state is protocol dependent
(FW)	Firmware controls function through registers
VDD	Hardware supplies power through pin, applicable only to **CARD_PWR** pins
none	Hardware disables pad

Table 24.3 SEC2410/SEC4410 64-Pin Reset States

		RESET STATE		
PIN	PIN NAME	OUT-PUT	PU/PD	IN-PUT
39	SD1_D0	Z		
38	SD1_D1	Z		
50	SD1_D2	Z		
49	SD1_D3	Z		
47	SD1_D4	Z		
45	SD1_D5	Z		
41	SD1_D6	Z		
40	SD1_D7	Z		
46	SD1_CMD	Z		

Table 24.3 SEC2410/SEC4410 64-Pin Reset States (continued)

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
42	SD1_CLK	Z		
36	GPIO6 (SD1_WP)	Z		
37	GPIO15 (SD1_nCD)	Z		
53	GPIO12	Z		
24	SD2_D0/GPIO18	Z		
23	SD2_D1/GPIO19	Z		
34	SD2_D2/GPIO20	Z		
33	SD2_D3/GPIO21	Z		
32	SD2_D4/GPIO22	Z		
30	SD2_D5/GPIO23	Z		
26	SD2_D6/GPIO24	Z		
25	SD2_D7/GPIO25	Z		
27	SD2_CLK/GPIO26	Z		
31	SD2_CMD/GPIO17	Z		
21	GPIO7 (SD2_WP)	Z		
22	GPIO16 (SD2_nCD)	Z		
6	SPI_CEN	(H)	PU	
7	SPI_CLK/GPIO4	Z		
9	SPI_DI/GPIO2	Z		
8	SPI_DO(SPD_SEL)/GPIO5	Z		
58	SC_RST_N/GPIO27	Z		
56	SC_CLK/GPIO28	Z		
55	SC_IO/GPIO31	Z		
60	VAR_CRD_PWR/GPIO10	Z		
57	SC_SPU/GPIO30	Z		
53	GPIO14 (SC_PSNT_N)	Z		
52	SC_ACT/LED_B0/GPIO1	Z		
54	SC_FCB/GPIO29	Z		

Table 24.3 SEC2410/SEC4410 64-Pin Reset States (continued)

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
4	USBDP	Z		
5	USBDM	Z		
4	HSIC_STROBE	Z		
5	HSIC_DATA	Z		
63	RBIAS			ANALOG
64	VDDA33			ANALOG
62	XTAL1/CLKIN			ANALOG
61	XTAL2			ANALOG
3	GPIO3 (VBUS_DET)	Z		Yes
12	JTAG_TMS	Z		Yes
13	JTAG_TDO	Z		
14	JTAG_TDI	Z		Yes
15	JTAG_TCK	Z		Yes
16	JTAG_TRST	Z		Yes
2	RESET_N	Z		Yes
18	CR_ACT/LED_A0/GPIO0	Z		
1	TEST	Z	PD	Yes
29	CRD_PWR/GPIO9	Z		
20	UART_TX_GPIO11	Z		
19	UART_RX/GPIO12	Z		
48	GPIO13	Z		
-	VDD33			ANALOG

Table 24.4 SEC2410/SEC4410 72-Pin Reset States

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
44	SD1_D0	Z		
43	SD1_D1	Z		

Table 24.4 SEC2410/SEC4410 72-Pin Reset States (continued)

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
55	SD1_D2	Z		
54	SD1_D3	Z		
52	SD1_D4	Z		
50	SD1_D5	Z		
46	SD1_D6	Z		
45	SD1_D7	Z		
51	SD1_CMD	Z		
47	SD1_CLK	Z		
41	GPIO6(SD1_WP)	Z		
42	GPIO15(SD1_nCD)	Z		
48	GPIO12	Z		
27	SD2_D0/GPIO18	Z		
26	SD2_D1/GPIO19	Z		
39	SD2_D2/GPIO20	Z		
38	SD2_D3/GPIO21	Z		
35	SD2_D4/GPIO22	Z		
33	SD2_D5/GPIO23	Z		
29	SD2_D6/GPIO24	Z		
28	SD2_D7/GPIO25	Z		
30	SD2_CLK/GPIO26	Z		
34	SD2_CMD/GPIO17	Z		
24	GPIO7 (SD2_WP)	Z		
25	GPIO16 (SD2_nCD)	Z		
6	SPI_CEN	(H)	PU	
7	SPI_CLK/GPIO4	Z		
9	SPI_DI/GPIO2	Z		
8	SPI_DO(SPD_SEL)/GPIO5	Z		
63	SC_RST_N/GPIO27	Z		

Table 24.4 SEC2410/SEC4410 72-Pin Reset States (continued)

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
61	SC_CLK/GPIO28	Z		
60	SC_IO/GPIO31	Z		
65	VAR_CRD_PWR/GPIO10	Z		
62	SC_SPU/GPIO30	Z		
58	GPIO14 (SC_PSNT_N)	Z		
57	SC_ACT/LED_B0/GPIO1	Z		
59	SC_FCB/GPIO29	Z		
4	USBDP	Z		
5	USBDM	Z		
4	HSIC_STROBE	Z		
5	HSIC_DATA	Z		
70	RBIAS			ANALOG
71	VDDA33			ANALOG
69	VDD12PLL_MON			ANALOG
68	XTAL1/CLKIN			ANALOG
67	XTAL2			ANALOG
2	GPIO3 (VBUS_DET)	Z		Yes
12	JTAG_TMS	Z		Yes
13	JTAG_TDO	Z		
14	JTAG_TDI	Z		Yes
15	JTAG_TCK	Z		Yes
16	JTAG_TRST	Z		Yes
1	RESET_N	Z		Yes
21	CR_ACT/LED_A0/GPIO0	Z		
72	TEST	Z	PD	Yes
32	CRD_PWR/GPIO9	Z		
23	UART_TX/GPIO11	Z		

Table 24.4 SEC2410/SEC4410 72-Pin Reset States (continued)

		RESET STATE		
PIN	**PIN NAME**	**OUT-PUT**	**PU/PD**	**IN-PUT**
22	UART_RX/GPIO12	Z		
53	GPIO13	Z		
-	VDD33			ANALOG
69	VDD12_BYP			ANALOG
20	TRACEDATA0	Z		
19	TRACEDATA1	Z		
18	TRACEDATA2	Z		
17	TRACEDATA3	Z		
37	TRACECLK	Z		
20	SWV	Z		

Chapter 25 DC Parameters

25.1 Maximum Guaranteed Ratings

PARAMETER	SYMBOL	MIN	MAX	UNITS	COMMENTS
Storage Temperature	T_{STOR}	-55	150	°C	
Lead Temperature			325	°C	Soldering < 10 seconds
3.3V supply voltage	V_{DD33}, V_{DDA33}	-0.5	4.0	V	
Voltage on USB+ and USB- pins		-0.5	5.5	V	
Voltage on **GPIO10**		0	3.63	V	When internal power FET operation of these pins are enabled, these pins may be simultaneously shorted to ground or any voltage up to 3.63 V indefinitely, without damage to the device as long as V_{DD33} and V_{DDA33} are less than 3.63 V and T_A is less than 70°C.
Voltage on any signal pin		-0.5	3.6	V	
Voltage on **XTAL1**		0	2.0	V	

Note 25.1 Stresses above the specified parameters may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at any condition above those indicated in the operation sections of this specification is not implied.

Note 25.2 When powering this device from laboratory or system power supplies the Absolute Maximum Ratings must not be exceeded or device failure can result. Some power supplies exhibit voltage spikes on their outputs when the AC power is switched on or off. In addition, voltage transients on the AC power line may appear on the DC output. When this possibility exists, a clamp circuit should be used.

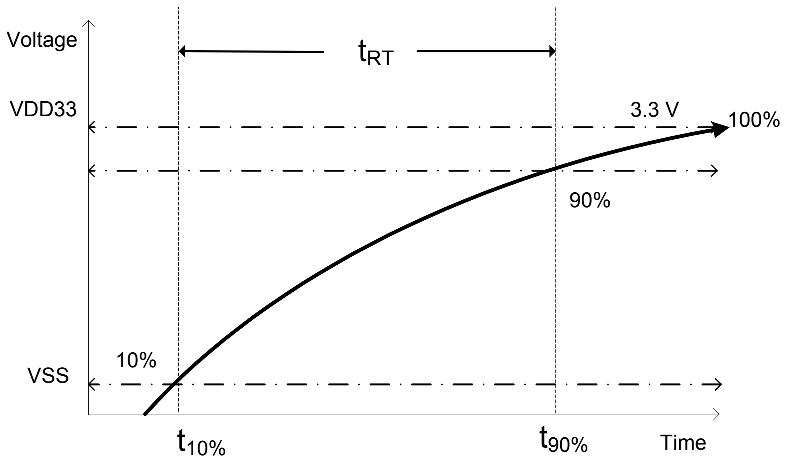


Figure 25.1 Supply Rise Time Models

25.2 Operating Conditions

PARAMETER	SYMBOL	MIN	MAX	UNITS	COMMENTS
3.3V supply voltage	V_{DD33}, V_{DDA33}	3.0	3.6	V	A 3.3 V regulator with an output tolerance of 1% must be used if the output of the internal power FET's must support a 5% tolerance.
3.3V supply rise time	t_{RT33}	0	400	µs	(Figure 25.1)
Voltage on USB+ and USB- pins		-0.5	5.5	V	If any 3.3V supply voltage drops below 3.0V, then the MAX becomes: (3.3 V supply voltage) + 0.5 ≤ 5.5
Voltage on any signal pin		-0.5	4.0	V	
Voltage on **XTAL1**		0	1.2	V	

25.3 DC Electrical Characteristics

TA = 0C - 70C, VDD33, VDDA = +3.3 V ± 0.3 V

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	COMMENTS
I, IPU, IPD Type Input Buffer						TTL Levels
Low Input Level	V_{ILI}			0.8	V	
High Input Level	V_{IHI}	2.0			V	
Pull Down	PD		52		µA	
Pull Up	PU		53		µA	
IS Type Input Buffer						TTL Levels
Low Input Level	V_{ILI}			0.8	V	
High Input Level	V_{IHI}	2.0			V	
Hysteresis	V_{HYSI}		500		mV	
ICLK Input Buffer						
Low Input Level	V_{ILCK}			0.4	V	
High Input Level	V_{IHCK}	2.2			V	
Input Leakage	I_{IL}				µA	V_{IN} = 0 to V_{DD33}
Input Leakage (All I and IS buffers)						
Low Input Leakage	I_{IL}	-10		+10	µA	V_{IN} = 0
High Input Leakage	I_{IH}	-10		+10	µA	V_{IN} = V_{DD33}

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	COMMENTS
O12 Type Buffer						
Low Output Level	V_{OL}			0.4	V	I_{OL} = 12 mA @ V_{DD33}= 3.3V
High Output Level	V_{OH}	V_{DD33} - 0.4			V	I_{OH} = -12 mA @ V_{DD33}= 3.3V
Output Leakage	I_{OL}	-10		+10	µA	V_{IN} = 0 to V_{DD33} (Note 25.3)
I/O12, I/O12PU, I/O12PD Type Buffer						
Low Output Level	V_{OL}			0.4	V	I_{OL} = 12 mA @ V_{DD33}= 3.3V
High Output Level	V_{OH}	V_{DD33} - 0.4			V	I_{OH} = -12 mA @ V_{DD33}= 3.3V
Output Leakage	I_{OL}	-10		+10	µA	V_{IN} = 0 to V_{DD33} (Note 25.3)
Pull Down	PD				µA	
Pull Up	PU				µA	
IO-U (Note 25.5)						
I-R (Note 25.6)						
Integrated Power FET Set to 200 mA, I/O200 Type Buffer						
Output Current (Note 25.7)	I_{OUT}	200			mA	$Vdrop_{FET}$ = 0.46V
Short Circuit Current Limit	I_{SC}	100			mA	$Vout_{FET}$ = 0V
On Resistance (Note 25.7)	R_{DSON}			2.1	Ω	I_{FET} = 70 mA
Output Voltage Rise Time	t_{DSON}			800	µs	C_{LOAD} = 10 µF
Supply Current Unconfigured	I_{CCINT}					@ V_{DD33}, V_{DD33A} = 3.3 V
Hi-Speed Host	$I_{CCINTHS}$		45	52	mA	
Full-Speed Host	$I_{CCINITFS}$		42	48	mA	
Supply Current Active	I_{CC}					@ V_{DD33}, V_{DD33A} = 3.3 V
Hi-Speed 1 SD Card			80	90	mA	(Note 25.9)
Hi-Speed 2 SD Cards			85	95	mA	
Full-Speed 1 SD Card			60	70	mA	
Full-Speed 2 SD Cards			65	75	mA	
Supply Current Standby	I_{CSBY}		700	1900	µA	

Note 25.3 Output leakage is measured with the current pins in high impedance.

Note 25.4 See Appendix A for USB DC electrical characteristics.

Note 25.5 See the *USB 2.0 Specification* [1], Chapter 7, for USB DC electrical characteristics

Note 25.6 RBIAS is a 3.3 V tolerant analog pin.

Note 25.7 Output current range is controlled by program software. The software disables the FET during a short circuit condition.

Note 25.8 The 3.3 V supply should be at least at 75% of its operating condition before the 1.2 V supply is allowed to ramp up.

Note 25.9 The current values listed do not include the contribution from the I/O200 type buffer for external SD card power.

25.4 Capacitance

T_A = 25°C; fc = 1 MHz; VDD33 = 3.3 V

Table 25.1 Pin Capacitance

PARAMETER	SYMBOL	LIMITS			UNIT	TEST CONDITION
		MIN	TYP	MAX		
Clock Input Capacitance	C_{XTAL}			2	pF	All pins (except USB pins and pins under test) are tied to AC ground.
Input Capacitance	C_{IN}			10	pF	
Output Capacitance	C_{OUT}			20	pF	

25.5 Package Thermal Specifications

Thermal parameters are measured or estimated for devices with the exposed pad soldered to thermal vias in a multi-layer 2S2P PCB per JESD51. Thermal resistance is measured from the die to the ambient air.

Table 25.2 64-Pin QFN Package Thermal Parameters

PARAMETER	SYMBOL	VALUE	UNIT
Thermal Resistance	Θ_{JA}	TBD	°C/W
Junction-to-Top-of-Package	Ψ_{JT}	TBD	°C/W

25.6 ESD and Latch-Up Performance

Table 25.3 ESD and Latch-up Performance

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	COMMENTS
ESD PERFORMANCE						
ESD JESD22-A114-B	Human Body Model (HBM)	-8		8	kV	Device
ESD JESD22-A115-A	Sensitivity Testing Machine Model (MM)	-300		300	V	
ESD JESD22-C101-A	Charged Device Model (CDM)	-500		500	V	
LATCH-UP PERFORMANCE						
JESD78B @ 125ºC	Latch-up	200			mA	

Notes: ESD Performance: The *MIN* column specifies the largest negative voltage the device can absorb without damage and the max number is the largest positive voltage the device can absorb without damage. Latch-up Performance: The latch-up parameter in Table 25.3 shows the highest current pulse that is tested without causing latch-up. The current is usually much larger than the current that normally would flow in or out of the pin.

25.7 ESD Performance

The SEC2410/SEC4410 is protected from ESD strikes. By eliminating the requirement for external ESD protection devices, board space is conserved, and the board manufacturer is enabled to reduce cost. The advanced ESD structures integrated into the SEC2410/SEC4410 protect the device whether or not it is powered up.

25.7.1 Component-level ESD Testing

Electrostatic Discharge (ESD) tolerance is measured for devices by subjecting the device to positive and negative pulses from a charged model. All signal pins are tested with all permutations of power and ground pins. Power pins are also tested to all permutations of ground pins and ground pins to all other ground pins. The device undergoing JEDEC ESD tests is unpowered.

25.8 Latch-up Testing

Latch-up testing is done on a powered device which is configured to be in a low power state. A current pulse is applied to each signal pin while the power supply current is monitored. If there is no noticeable increase in power supply current, the device passes latch-up failure.

Chapter 26 AC Specifications

Refer to the appropriate specification document for each flash media device or USB interface as outlined in Appendix B: *(References)* on page 322.

26.1 Oscillator/Clock

Crystal: Parallel Resonant, Fundamental Mode, 24 MHz ± 100 ppm.

External Clock: 50% Duty cycle ± 10%, 24 MHz ± 100 ppm, Jitter < 100 ps rms.

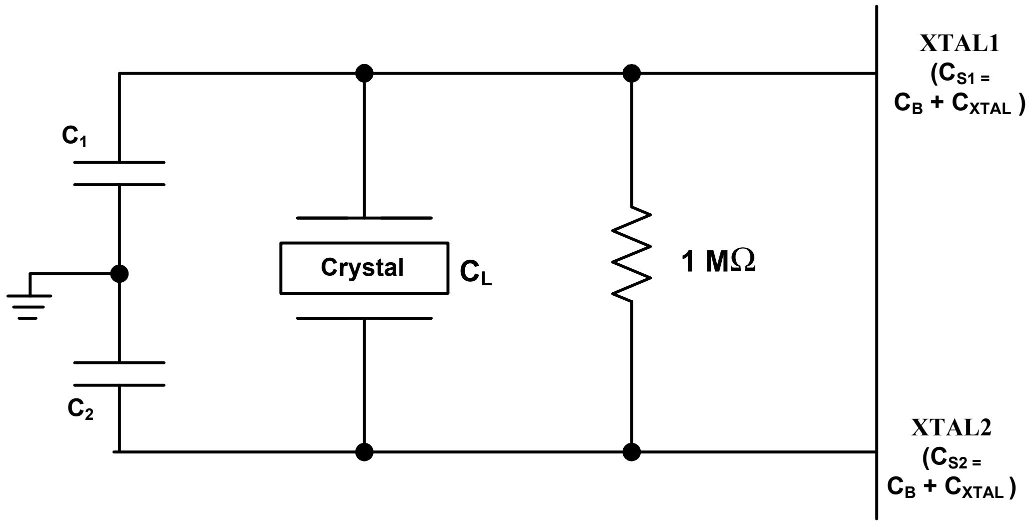


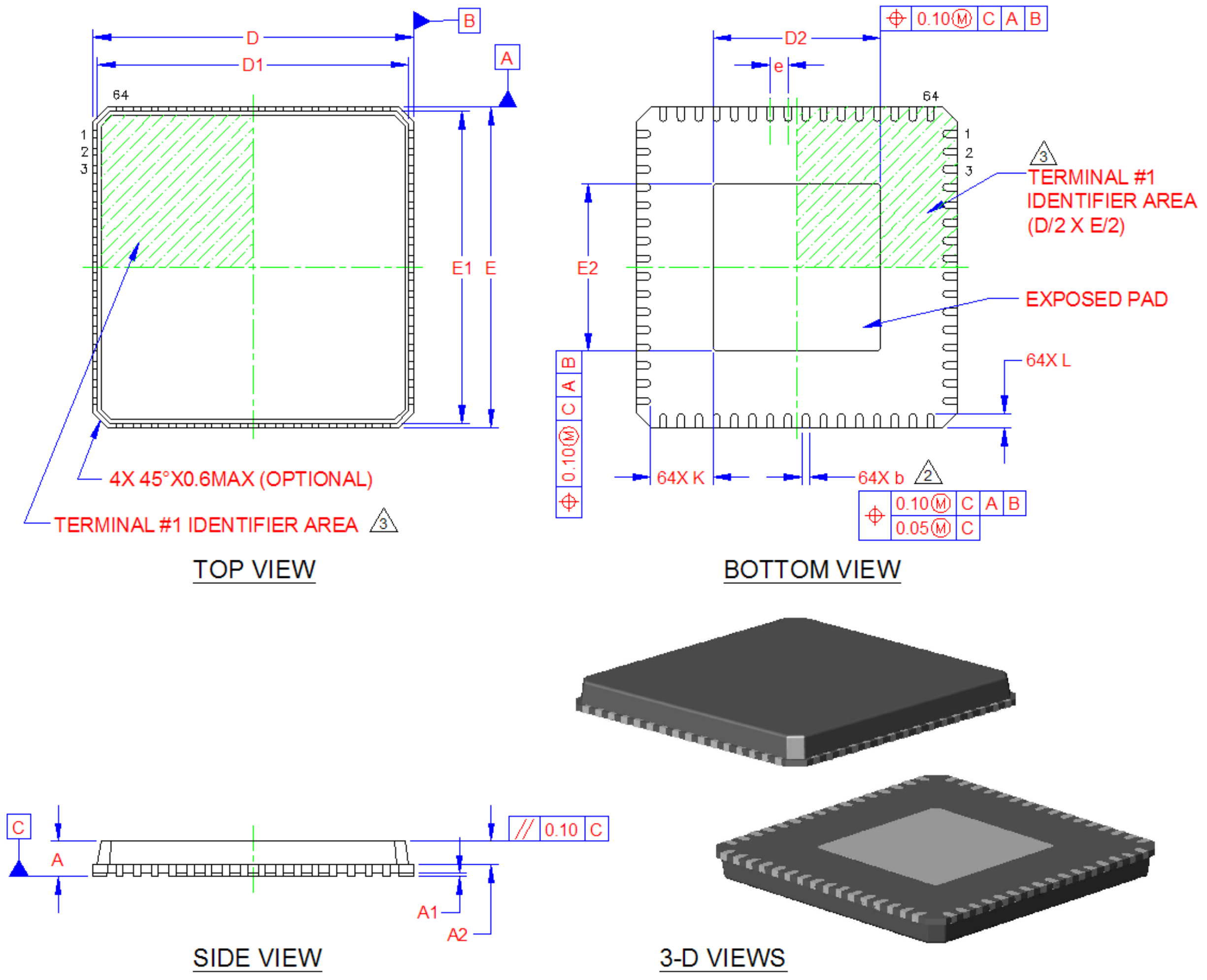
Figure 26.1 Typical Crystal Circuit

Note: C_B equals total board/trace capacitance.

$$C_L = \frac{(C_1 + C_{S1}) \times (C_2 + C_{S2})}{(C_1 + C_{S1} + C_2 + C_{S2})}$$

Figure 26.2 Formula to find value of C_1 and C_2

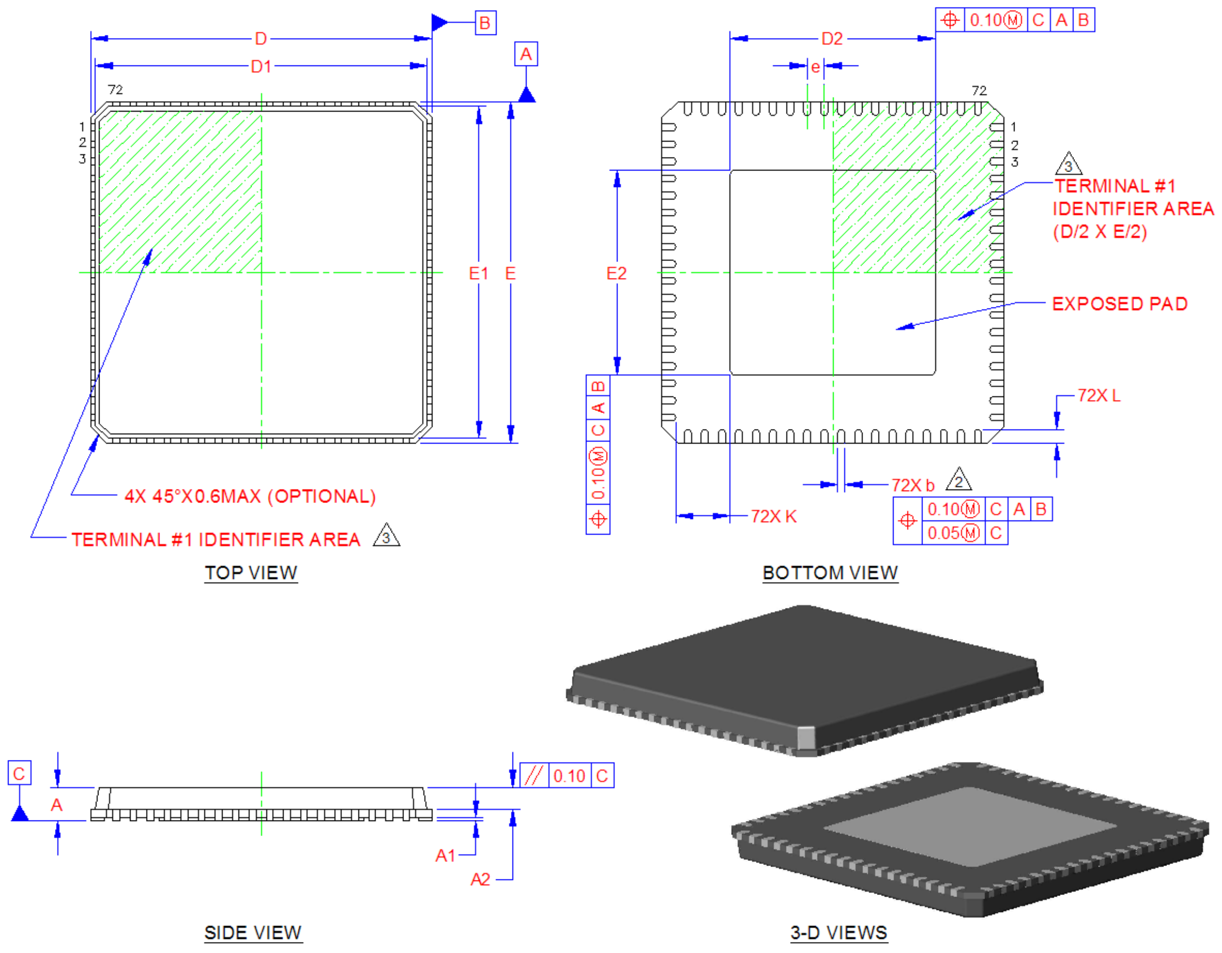
Chapter 27 Package Outline



COMMON DIMENSIONS					
SYMBOL	MIN	NOM	MAX	NOTE	REMARK
A	0.80	0.85	1.00	-	OVERALL PACKAGE HEIGHT
A1	0	0.02	0.05	-	STANDOFF
A2	-	0.65	0.80	-	MOLD CAP THICKNESS
D/E	8.90	9.00	9.10	-	X/Y BODY SIZE
D1/E1	8.65	8.75	8.85	-	X/Y MOLD CAP SIZE
D2/E2	4.60	4.70	4.80	-	X/Y EXPOSED PAD SIZE
L	0.30	0.40	0.50	-	TERMINAL LENGTH
b	0.18	0.25	0.30	2	TERMINAL WIDTH
K	1.55	-	-	-	CENTER PAD TO PIN CLEARANCE
e	0.50 BSC			-	TERMINAL PITCH

NOTES:
1. ALL DIMENSIONS ARE IN MILLIMETER.
2. DIMENSIONS "b" APPLIES TO PLATED TERMINALS AND IT IS MEASURED BETWEEN 0.15 AND 0.30 mm FROM THE TERMINAL TIP.
3. DETAILS OF TERMINAL #1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE AREA INDICATED. THE TERMINAL #1 IDENTIFIER MAY BE EITHER A MOLD OR MARKED FEATURE.

Figure 27.1 SEC2410/SEC4410 64-Pin QFN Package Drawing



COMMON DIMENSIONS					
SYMBOL	MIN	NOM	MAX	NOTE	REMARK
A	0.80	0.85	1.00	-	OVERALL PACKAGE HEIGHT
A1	0	0.02	0.05	-	STANDOFF
A2	-	0.65	0.80	-	MOLD CAP THICKNESS
D/E	9.90	10.00	10.10	-	X/Y BODY SIZE
D1/E1	9.65	9.75	9.85	-	X/Y MOLD CAP SIZE
D2/E2	5.90	6.00	6.10	-	X/Y EXPOSED PAD SIZE
L	0.30	0.40	0.50	-	TERMINAL LENGTH
b	0.18	0.25	0.30	2	TERMINAL WIDTH
K	1.50	-	-	-	CENTER PAD TO PIN CLEARANCE
e	0.50 BSC			-	TERMINAL PITCH

NOTES:
1. ALL DIMENSIONS ARE IN MILLIMETER.
2. DIMENSIONS "b" APPLIES TO PLATED TERMINALS AND IT IS MEASURED BETWEEN 0.15 AND 0.30 mm FROM THE TERMINAL TIP.
3. DETAILS OF TERMINAL #1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE AREA INDICATED. THE TERMINAL #1 IDENTIFIER MAY BE EITHER A MOLD OR MARKED FEATURE.

Figure 27.2 SEC2410/SEC4410 72-Pin QFN Package Drawing

Chapter 28 Revision History

Table 28.1 Customer Revision History

REVISION LEVEL & DATE	SECTION/FIGURE/ENTRY	CORRECTION
Rev. 1.0 (03-07-13)	All	Initial release.

Appendix A (Acronyms and Definitions)

AHB: AMBA Host Bus

AMBA: Advanced Microcontroller Bus Architecture

CBWP: Auto CBW Processor

CN: Completion Notification

CRC: Cyclic Redundancy Checking

EC: Embedded Controller ARM-M3

ECC: Error Checking and Correcting

FM: Flash Media

FMC: Flash Media Controller

LDU: Logical Device Number

MMC: MultiMediaCard

MTU: Maximum Transmission Unit

NVIC: Nested Vectored Interrupt Controller

SC: Smart Card

SCC: Smart Card Controller

SD: Secure Digital

SDC: Secure Digital Controller

TPC: Transport Protocol Code.

WIC: Wake-up Interrupt Controller

WQ: Work Queue

WQE: Work Queue Element

WR: Work Request

WRLUT: Work Request Look Up Table

Appendix B (References)

[1] Universal Serial Bus Specification, Version 2.0, April 27, 2000 (12/7/2000 and 5/28/2002 Errata)
USB Implementers Forum, Inc. http://www.usb.org

[2] I^{2}C-Bus Specification Version 1.1
NXP (formerly a division of Philips). http://www.nxp.com

[3] System Management Bus Specification, version 1.0
SMBus. http://smbus.org/specs/

[4] MicroChip 24AA02/24LC02B (Revision C)
Microchip Technology Inc. http://www.microchip.com/

[5] JEDEC Specifications: JESD76-2 (June 2001) and J-STD-020D.1 (March 2008)
JEDEC Global Standards for the Microelectronics Industry.http://www.jedec.org/standards-documents

[6] Software Development Reference Guide
SMSC. http://www.smsc.com

[7] Secure Bootloader Reference Guide
SMSC. http://www.smsc.com

[8] SmartCard ISO IEC 7816
http://www.smartcardalliance.org/pages/smart-cards-intro-standards#isoiec-standards

[9] MultiMediaCard System Specification Version 4.3 (JEDEC JESD84-A43)
http://www.jedec.org/standards-documents/docs/jesd-84-a43

[10] SD Specifications, Part 1, Physical Layer Specification Version 3.0, March, 2009 (SDXC Legacy mode - 2TB Storage) http://www.sdcard.org

[11] SD Specifications, Part 2, File System Specification Version 3.0, April 2009
http://www.sdcard.org

[12] SD Specifications, Part 3, Security Specification Version 2.0, June 7th, 2006
http://www.sdcard.org

[13] SD Specifications, Part E1, Secure Digital Input/Output (SDIO) card Specification, Version 2.0, Feb. 8, 2007, SD Group.http://www.sdcard.org

[14] USB Mass Storage Class, Bulk Only Transport, Revision 1.0, September 31, 1999
http://www.usb.org

[15] RFC 4493 The AES-CMAC Algorithm June 2006
http://www.ietf.org/rfc/rfc4493.txt