VA108X0
ARM® Cortex®-M0 based Processor
Programmers Guide
July 2020 Version 1.20

VA108X0

VA108X0 Programmers Guide

1 Overview

This document describes the programmer's view of the VORAGO VA108X0 ARM® Cortex®-M0 based microcontrollers.

1.1 Related Documents

- ARM® Documents (Available from http://infocenter.arm.com)
  - Cortex®-M0 Generic User Guide
  - Cortex®-M0 Technical Reference Manual
  - AMBA® 3 AHB-Lite Protocol Specification
  - AMBA® 3 APB Protocol Specification
- NXP Documents (Available from http://www.nxp.com)
  - I²C-bus specification and user manual
- IEEE
  - 1149.1-2013 - IEEE Standard for Test Access Port and Boundary-Scan Architecture
- VORAGO Documents
  - VA10800/VA10820 Data Sheet

2 Functional Description

The VA10800 and VA10820 MCUs contains an ARM® Cortex®-M0 processor, and a related set of peripherals.

2.1 Features

- Processor Core
  - ARM® Cortex®-M0 processor
    - Up to 50 MHz
    - SysTick Counter
    - Single Cycle Multiply
  - ARM® Cortex®-M0 built-in Nested Vectored Interrupt Controller (NVIC)
    - 32 Interrupt sources with unique priority level
    - Tail chaining supported
  - CoreSight™ compliant debug access via JTAG based Debug interface
    - 4 Breakpoint Comparators
    - 2 Data Watch Point Comparators
    - JTAG Debug Port
- Memory
  - 32kbyte Data Memory
  - 128kbyte Code Memory
    - Loaded from external Serial Peripheral Interface (SPI) based memory at startup
    - Configurable boot delay, boot speed, and boot checking
  - Byte level Error Correct and Detect (EDAC) logic (VA10820 Only)
  - Scrub Engine (VA10820 Only)
  - 128 Kbyte serial NVM (VA10830 only)
- Peripherals
  - 2 UARTs
    - 16 word Transmit and Receive FIFOs
    - Fractional baud rate generation
      - Supports baud rates up to 115200 with system clocks above 2MHz
    - Supports 5, 6, 7, 8 and 9 bits
    - Supports Even, Odd, and No parity
    - Stop Bits 1 or 2
    - Supports Break generation and detection

- Error detection
  - FIFO overflow
  - Framing error
  - Parity error
  - Break detection
- Configurable Interrupt generation
  - FIFO level (fully configurable)
  - Receive Timeout
  - Error

- 3 SPI Ports (One is designated Master only)
  - Supports all 4 modes of Motorola's SPI Specification
  - Word/Frame size of 4 to 16 bits
  - 16 word Transmit and Receive FIFOs
  - Block mode support for larger Frame sizes
  - Master mode rates up to 1/4 the system clock
  - Slave mode rates up to 1/12 the system clock
  - Configurable Interrupt generation
    - FIFO level (fully configurable)
    - FIFO Overflow
    - Receive Timeout
  - 2 Ports Configurable as Master or Slave
  - 1 Port is Master Only
    - Uses the SPI Boot ROM pins after startup
- I^2C
  - Standard I^2C-compliant bus inference
  - Dedicated open-drain pins supporting I^2C Fast-mode
  - Configurable as Master or Slave
  - 16 word Transmit and Receive FIFOs
  - Configurable Interrupt generation
    - FIFO empty/full level programmable
  - Note: Fast-Mode non-obstruct feature is not supported
- GPIO
  - 2 GPIO Ports, Up to 56 pins total
    - 32-bit port A
    - 24-bit port B
  - Configurable direction control of individual bits

- Bit level mask register allows single instruction setting or clearing of any bits in one port.
- Configurable interrupt detect on individual bits
  - Level or Edge sensitive
- Configurable Pulse mode on individual bits
- Configurable (0-3) cycle delay on individual bits

- IO Configuration
  - Manages GPIO/SPI/UART IO configurations:
    - Glitch filters
    - Pull-up/Pull-down
    - Signal inversion
    - Pseudo open-drain
    - Maps Timer, SPI and UART blocks to specific pins
- Counters/Timers
  - 24 Counter/Timers
  - Advanced trigger modes
    - Start/Stop based on other Counter/Timers or GPIO signals
    - Multiple trigger sources
  - Configurable output event
    - One cycle zero detect
    - Active mode
    - Divide by 2
    - PWM compare
- Interrupt Select
  - Maps >100 possible interrupt sources to the 32 NVIC inputs
  - Configurable source for alternate Interrupts
    - NMI
    - Watchdog Reset
    - Memory Error Reset
    - Processor Receive Event
- System Configuration
  - Memory Control
    - Data memory clear on reset
    - Code memory reload on reset
    - Code memory write protect
    - Code/Data memory Scrub rate (VA10820/VA10830 Only)

- Code/Data memory error injection for testing (VA10820/VA10830 Only)
- Code/Data memory SBE/MBE counters (VA10820/VA10830 Only)
- Code/Data memory SBE/MBE Interrupt control (VA10820/VA10830 Only)
- EDAC Syndrome calculation support (VA10820/VA10830 Only)

  - Register for Scrub rate control. (Performs EDAC read sequentially through both data and code memory.) (VA10820/VA10830 Only)
  - GPIO Glitch Filter rate control
  - Peripheral Configuration
    - Enable/Disable/Reset individual peripherals

- JTAG
  - 2 Serial Controllers on same pins
  - M0 Debug Controller
    - Provides access to M0 Debug port
  - VORAGO Controller
    - Provides standard boundary scan
    - Provides BIST access to memories
    - Provides eFuse access
    - Provides Test mode access
      - Scan
      - IDDQ
      - I/O Parametric
      - Configuration of Boot sequence
      - Configuration of Memory Margin

- eFuse
  - 32-bit Unique ID Number Support
  - Custom part configuration information
    - SPI ROM interface – Delay, Speed, Size, Checking
  - Multiple reconfiguration support (limited to 30 times)

2.2 Block Diagram

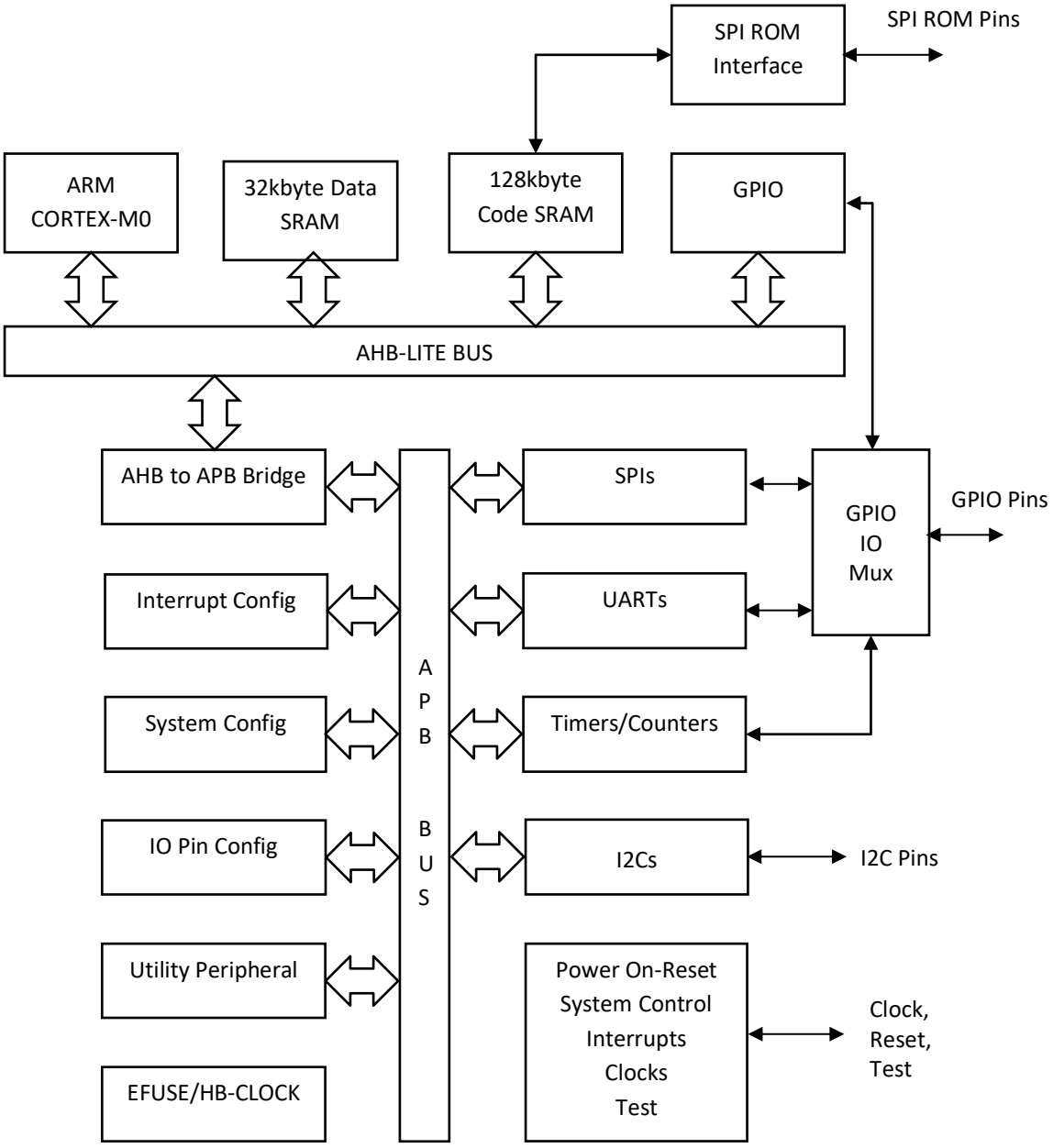
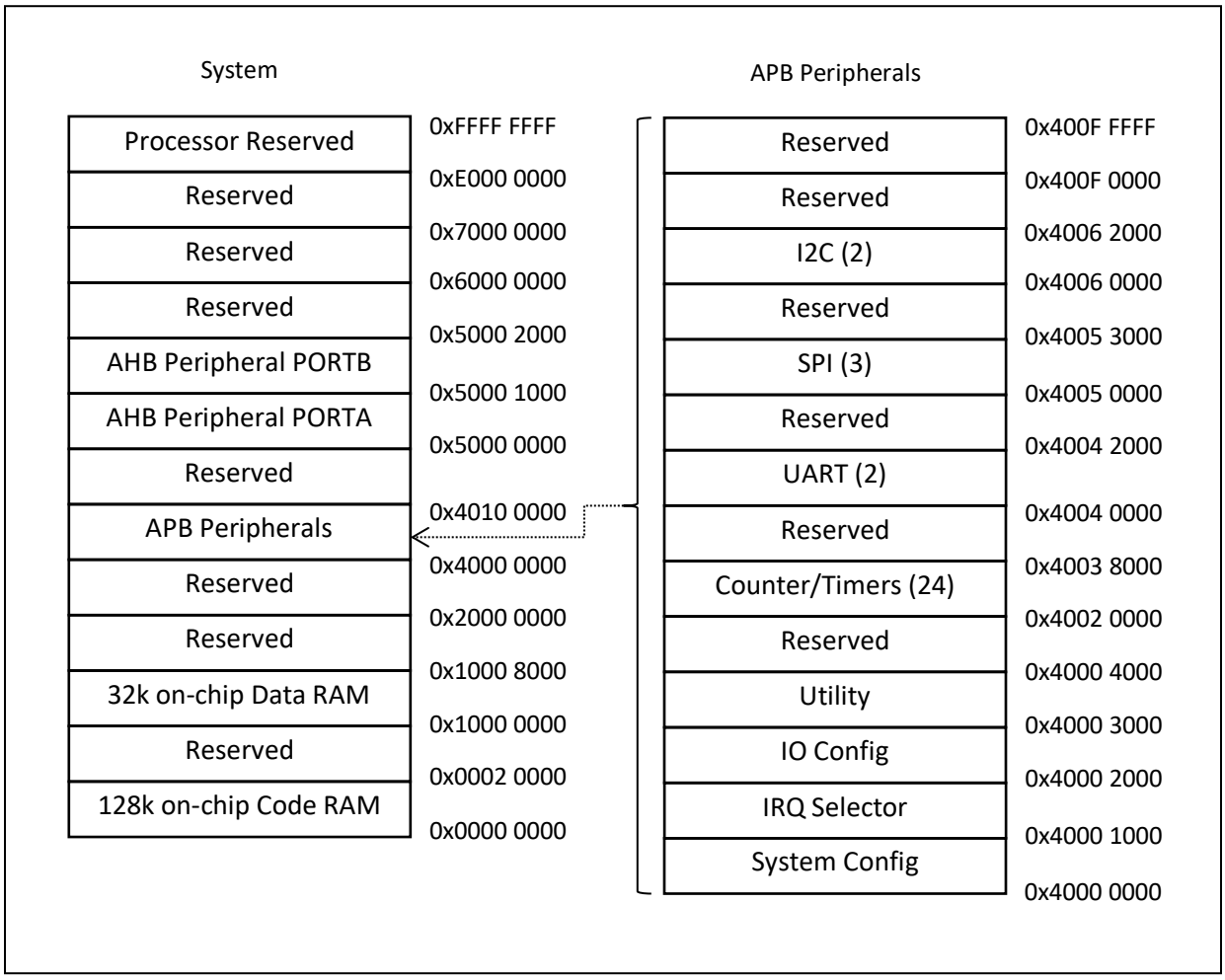


Figure 1 - Chip Block Diagram

2.3 Memory Map

The processor chip contains several memory areas shown below:



2.4 Power-Up Sequence

The VA10800/VA10820 auto-detects the Power-Up condition and begins operations by loading the internal code memory from an external Serial Peripheral Interface (SPI) based memory. The VA10830 auto-detects the Power-Up condition and begins operations by loading the internal code memory from an in-package Serial Peripheral Interface (SPI) based memory. After loading the code memory, the processor follows a normal ARM® Cortex®-M0 start sequence.

The Power-Up sequence is triggered by the internal Power-On-Reset detection logic and controlled by the internal nominal 1MHz oscillator. When the Power-Up condition is

triggered, the primary internal logic (and the ARM® Cortex®-M0 processor) is held in reset until the Power-Up sequence completes. This sequence consists of the following steps:

1. 1ms delay from 1000 cycles of the nominal 1MHz oscillator (this allows VDD/VDDIO to reach higher levels)
2. Read internal eFuse to get the configuration data.
3. Timed delay based on eFuse configuration data and the nominal 1MHz oscillator.
4. Release of reset to the SPI boot controller. This begins the process of loading the internal code memory with data from the external SPI boot memory. The memory is read starting at address 0, in blocks of 128 bytes. Some aspects of this SPI based boot process are customizable (such as SPI speed, SPI latency, boot memory size, etc.); these are documented in the JTAG and eFuse sections.
5. Release of reset to the ARM® Cortex®-M0 processor. This begins the execution of code by the processor.

The EXTRESETn pin can also be used to delay the Power-Up Sequence. If EXTRESETn is active (low) at any time during the initial 1ms delay, the Power-Up sequence is stalled until EXTRESETn is inactive (high).

In addition to loading the code memory, the Power-Up reset sequence (step 4) will initialize the data memory to all zero values. This allows correct error-detect and correct (EDAC) syndromes to be generated for all memory data.

The CLK pin must be valid starting at step 4. Valid clock means stable values (below V_{IL} or above V_{IH}) and meeting the minimum Clock high and low times. This allows the clock to be off as long as the level is valid. Steps 4 and 5 clock source is the CLK pin. If the CLK pin, does not have a valid input, steps 4 and 5 will not occur and no code will be executed.

2.4.1 Power-Up and Reset Behavior of pins

This section describes the Power-Up and Reset behavior of the GPIO pins on the device.

- At Power-up, an internal SYNC_PORESET signal is asserted asynchronously (without a clock required). For all other reset events the internal SYNC_PORESET signal is asserted synchronously to the clock.
- The SYNC_PORESET signal directly asserts the internal HRESET signal. This signal synchronously resets the processor and peripherals.
- The IO interface unit places all the GPIO pins in the high-Z state while HRESET is active.
- The HRESET is de-asserted after loading the program code from external memory when the processor starts its boot sequence.

- Processor boot code can then configure the pins as needed by the peripheral control registers.

2.5 Other Resets

In addition to the Power-Up reset, the device can be reset from other events:

- EXTRESETn pin
- SYSRESETREQ from software
- Hardware events configured by IRQ Selector Peripheral or the System Controller Peripheral:
    - Processor Lockup
    - Watchdog Timer
    - Memory Errors (Single or Multibit errors from the EDAC memory controller)

Based on previous software configuration (in the RST_CNTL_ROM and RST_CNTRL_RAM registers), these reset events may or may not re-initialization the memories similar to the Power-Up reset sequence.

2.6 I^2C pins

The VA108X0 contains 2 sets of dedicated I^2C pins and related I^2C controllers. Each controller can act as both an I^2C master and an I^2C slave simultaneously. These interfaces are capable of operating at up to 100 kbits/s in Standard-mode, and up to 400 kbits/s in Fast-mode.

Note, that due to the ESD protection used in the ***HARDSIL™*** process, the device does not meet the non-obstruct feature of I^2C Fast-Mode when this device is powered off. If this non-obstruct feature is required in a system using this part, it will need to be implemented external to the device.

3 ARM® Cortex®-M0 processor

Primary documentation on the Cortex®-M0 processor can be found in the ARM® Documents:

- Cortex®-M0 Generic User Guide
- Cortex®-M0 Technical Reference Manual

The included ARM® Cortex®-M0 processor is configures as follows:

- All memory access is in little-endian format
  - Code Memory is at addresses: 0x00000000-0x0001FFFF
  - Data Memory is at addresses: 0x10000000-0x10007FFF
- Nested Vectored Interrupt Controller (NVIC) with 32 interrupts
  - Assignment of interrupts is based on the IRQ Selector Peripheral
  - Low latency IRQ mode
- System timer (SysTick) is present
  - The SysTick Calibration (SYST_CALIB) registers reports: 0xC000000, which indicates:
    - NOREF - no alternative reference clock source
    - SKEW – CLK does not guarantee an exact multiple of 10ms
- Single cycle multiple is present
- Processor sleep mode supported
  - Includes clock gating for reduced power
- Full debug port is present
  - Using JTAG interface
  - 4 breakpoint comparators
  - 2 data watchpoint comparators
- Processor External Event Inputs signal available from IRQ Selector Peripheral
- System reset request (SYSRESETREQ bit in the AIRCR register) will issue a system reset request.
- Wake-up interrupt controller (WIC) is not present
  - Deep sleep with power disabled is not supported.

4 Peripherals

4.1 System Configuration Peripheral (Software label = SYSCONFIG)

The system configuration peripheral contains registers that monitor and configure system wide operation.

Table 1 - System Configuration Peripheral Registers

Name	Access	Address offset	Description	Reset value
RST_STAT	RW	0x000	System Reset Status Register	See Details
RST_CNTL_ROM	RW	0x004	System Reset Control ROM Register	0x0000 003F
RST_CNTL_RAM	RW	0x008	System Reset Control RAM Register	0x0000 003F
ROM_PROT	RW	0x00C	ROM Memory Protection	0x0000 0000
ROM_SCRUB	RW	0x010	ROM Scrub Period	0x0000 0000
RAM_SCRUB	RW	0x014	RAM Scrub Period	0x0000 0000
ROM_TRAP_ADDR	RW	0x018	ROM EDAC Trap Address	0x0000 0000
ROM_TRAP_SYND	RW	0x01c	RAM EDAC Trap Syndrome	0x0000 0000
RAM_TRAP_ADDR	RW	0x020	RAM EDAC Trap Address	0x0000 0000
RAM_TRAP_SYND	RW	0x024	RAM EDAC Trap Syndrome	0x0000 0000
IRQ_ENB	RW	0x028	Enable IRQs	0x0000 0000
IRQ_RAW	R	0x02c	Raw IRQ Status	-
IRQ_END	R	0x030	Enabled IRQ Status	-
IRQ_CLR	W	0x034	Clear IRQ Status	-
RAM_SBE	RW	0x038	Count of RAM SBE Errors	0x0000 0000
RAM_MBE	RW	0x03c	Count of RAM MBE Errors	0x0000 0000
ROM_SBE	RW	0x040	Count of ROM SBE Errors	0x0000 0000

ROM_MBE	RW	0x044	Count of ROM MBE Errors	0x0000 0000
IOCONFIG_CLKDIV0	R	0x048	IO Filter Clock 0 Divide value	0x0000 0000
IOCONFIG_CLKDIV1	RW	0x04c	IO Filter Clock 1 Divide value	0x0000 0000
IOCONFIG_CLKDIV2	RW	0x050	IO Filter Clock 2 Divide value	0x0000 0000
IOCONFIG_CLKDIV3	RW	0x054	IO Filter Clock 3 Divide value	0x0000 0000
IOCONFIG_CLKDIV4	RW	0x058	IO Filter Clock 4 Divide value	0x0000 0000
IOCONFIG_CLKDIV5	RW	0x05c	IO Filter Clock 5 Divide value	0x0000 0000
IOCONFIG_CLKDIV6	RW	0x060	IO Filter Clock 6 Divide value	0x0000 0000
IOCONFIG_CLKDIV7	RW	0x064	IO Filter Clock 7 Divide value	0x0000 0000
ROM_RETRIES	RO	0x068	ROM Boot Retry Count	0x0000 0000
REFRESH_CONFIG	RW	0x06c	Register Refresh Rate	0x0000 0000
TIM_RESETS	RW	0x070	TIM Reset Control	0xFFFF FFFF
TIM_CLK_ENABLES	RW	0x074	TIM Clock Enable Control	0x0000 0000
PERIPHERAL_RESET	RW	0x078	Peripheral Reset Control	0xFFFF FFFF
PERIPHERAL_CLK_ENABLES	RW	0x07c	Peripheral Clock Enable Control	0x0000 0000
LOCKUP_RESET	RW	0x080	Lockup Reset Enable	0x0000 0001
-	-	0x084 – 0xFEC	Reserved	-
EF_CONFIG	RO	0xFF0	eFuse Config Register	-
EF_ID	RO	0xFF4	eFuse ID Register	-
PROCID	RO	0xFF8	Processor ID Register	See Details
PERID	RO	0xFFC	Peripheral ID Register	0x0180 07e1

4.1.1 RST_STAT Register

The system reset status register reports the source of the last system reset (including Power On Reset events). Multiple bits may be set if multiple resets were asserted at the same time. Bits are set in this registers by reset events. Bits are cleared by writing to this register.

Bit	Symbol	Description
0	POR	Power On Reset
1	EXTRST	External Reset
2	SYSRSTREQ	SYSRESETREQ from processor via software
3	LOCKUP	System LOCKUP asserted
4	WATCHDOG	Watchdog Reset asserted
5	MEMERR	Memory Error Reset asserted
31:6	Reserved	Reserved, Reads as 0

4.1.2 RST_CNTL_ROM Register

The system reset ROM control register controls which reset sources cause the Code RAM (ROM) to be reloaded from the external SPI ROM at reset. A one in any bit enables the given reset source. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
0	POR	Power On Reset (This bit is read-only as 1)	1
1	EXTRST	External Reset	1
2	SYSRSTREQ	SYSRESETREQ from processor via software	1
3	LOCKUP	System LOCKUP asserted	1
4	WATCHDOG	Watchdog Reset asserted	1
5	MEMERR	Memory Error Reset asserted	1
31:6	Reserved	Reserved, Reads as 0	

4.1.3 RST_CNTL_RAM Register

The system reset RAM control register controls which reset sources cause the Data RAM to be cleared on reset. A one in any bit enables the given reset source. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
0	POR	Power On Reset (This bit is always 1)	1
1	EXTRST	External Reset	1
2	SYSRSTREQ	SYSRESETREQ from processor via software	1

3	LOCKUP	System LOCKUP asserted	1
4	WATCHDOG	Watchdog Reset asserted	1
5	MEMERR	Memory Error Reset asserted	1
31:6	Reserved	Reserved, Reads as 0	

4.1.4 ROM_PROT Register

The ROM memory protection register controls write access to the CODE RAM. If a write access is attempted to the CODE RAM while it is write protected an access fault will be generated.

Bit	Symbol	Value	Description	Reset value
0	WREN		ROM Write Enable	0
		0	Code RAM is write protected	
		1	Code RAM is writable	
31:1	Reserved		Reserved, Reads as 0	

4.1.5 ROM_SCRUB Register

The ROM scrub register configures the background scrub rate for the Code RAM (ROM). A zero value disables the background scrub. A non-zero value sets the reset value for the scrub counter. When the counter reaches zero, it is restarted with the configured value, and a scrub request is issued. Each scrub request checks a single 32-bit memory word, and increments a scrub address register.

Note that the scrub controller does not exist in the VA10800 version.

Bit	Symbol	Description	Reset value
23:0	VALUE	Scrub Counter Reset value	0
30:24	Reserved	Reserved, reads as 0	
31	RESET	Write-Only bit, writing a 1 resets the counter	0

4.1.6 RAM_SCRUB Register

The RAM scrub register configures the background scrub rate for the Data RAM. A zero value disables the background scrub. A non-zero value sets the reset value for the scrub counter. When the counter reaches zero, it is restarted with the configured value, and a scrub

request is issued. Each scrub request checks a single 32-bit memory word, and increments a scrub address register.

Note that the scrub controller does not exist in the VA10800 version.

Bit	Symbol	Description	Reset value
23:0	VALUE	Scrub Counter Reset value	0
30:24	Reserved	Reserved, reads as 0	
31	RESET	Write-Only bit, writing a 1 resets the counter	0

4.1.7 ROM_TRAP_ADDR Register

The ROM EDAC TRAP ADDRESS register along with the ROM EDAC TRAP SYND register provides a way to write bad EDAC data to the Code RAM (ROM) for testing purposes. When TRAP is ENABLEd and a write operation to the configured memory address is requested, then the ROM EDAC TRAP SYND register is used instead of the normal computed Syndrome bits. Note that since memory is configured as a 32-bit word, the lower 2 address bits are not used in the compare. Also, only those address bits that access the Memory are compared.

Note that EDAC is not supported in the VA10800 version.

Bit	Symbol	Description	Reset value
1:0	Reserved	Reserved	0
30:2	ADDR	Address bits for TRAP match	0
31	ENABLE	Enable TRAP mode	0

4.1.8 ROM_TRAP_SYND Register

The ROM EDAC TRAP ADDRESS register along with the ROM EDAC TRAP SYND register provides a way to write bad EDAC data to the Code RAM (ROM) for testing purposes (See ROM_TRAP_ADDR Register description).

Bit	Symbol	Description	Reset value
4:0	SYND0	4 bit syndrome for data bits 7:0	0
9:5	SYND1	4 bit syndrome for data bits 15:8	0
14:10	SYND2	4 bit syndrome for data bits 23:16	0
19:15	SYND3	4 bit syndrome for data bits 31:24	0
31:20	Reserved	Reserved, Reads as 0	0

4.1.9 RAM_TRAP_ADDR Register

The RAM EDAC TRAP ADDRESS register along with the RAM EDAC TRAP SYND register provides a way to write bad EDAC data to the Data RAM (RAM) for testing purposes. When TRAP is ENABLEd and a write operation to the configured memory address is requested, then the RAM EDAC TRAP SYND register is used instead of the normal computed Syndrome bits. Note that since memory is configured as a 32-bit word, the lower 2 address bits are not used in the compare. Also, only those address bits that access the Memory are compared.

Note that EDAC is not supported in the VA10800 version.

Bit	Symbol	Description	Reset value
1:0	Reserved	Reserved	0
30:2	ADDR	Address bits for TRAP match	0
31	ENABLE	Enable TRAP mode	0

4.1.10 RAM_TRAP_SYND Register

The RAM EDAC TRAP ADDRESS register along with the RAM EDAC TRAP SYND register provides a way to write bad EDAC data to the Data RAM (RAM) for testing purposes (See RAM_TRAP_ADDR Register description).

Bit	Symbol	Description	Reset value
4:0	SYND0	4 bit syndrome for data bits 7:0	0
9:5	SYND1	4 bit syndrome for data bits 15:8	0
14:10	SYND2	4 bit syndrome for data bits 23:16	0
19:15	SYND3	4 bit syndrome for data bits 31:24	0
31:20	ENABLE	Reserved, Reads as 0	0

4.1.11 IRQ_ENB Register

The IRQ enable register configures which EDAC (Error Detect And Correct) sources will trigger Interrupts. Either the Data RAM or the Code RAM (ROM) can generate a single-bit or multi-bit error signal. This register is only reset by a Power-On-Reset.

Note that EDAC is not supported in the VA10800 version, so these interrupt sources do not exist in that version.

Bit	Symbol	Description	Power-on-Reset value
0	RAMSBE	Enable from Data RAM single-bit error	0

1	RAMMBE	Enable from Data RAM multi-bit error	0
2	ROMSBE	Enable from Code RAM (ROM) single-bit error	0
3	ROMMBE	Enable from Code RAM (ROM) multi-bit error	0
31:4	Reserved	Reserved, Reads as 0	

4.1.12 IRQ_RAW Register

The IRQ raw status register is a read-only register that reports raw status of the IRQ sources. This is the value before the enable is applied.

Bit	Symbol	Description	Reset value
0	RAMSBE	IRQ from Data RAM single-bit error	0
1	RAMMBE	IRQ from Data RAM multi-bit error	0
2	ROMSBE	IRQ from Code RAM (ROM) single-bit error	0
3	ROMMBE	IRQ from Code RAM (ROM) multi-bit error	0
31:4	Reserved	Reserved, Reads as 0	

4.1.13 IRQ_END Register

The IRQ ENableD status register is a read-only register that reports enabled status of the IRQ sources. This is the value after the enable is applied.

Bit	Symbol	Description	Reset value
0	RAMSBE	IRQ from Data RAM single-bit error	0
1	RAMMBE	IRQ from Data RAM multi-bit error	0
2	ROMSBE	IRQ from Code RAM (ROM) single-bit error	0
3	ROMMBE	IRQ from Code RAM (ROM) multi-bit error	0
31:4	Reserved	Reserved, Reads as 0	

4.1.14 IRQ_CLR Register

The IRQ clear status register is a write-only register that clears IRQ sources. Writing a 1 in the given bits clears the given IRQ source (these do not need to be written back to 0).

Bit	Symbol	Description
0	RAMSBE	IRQ from Data RAM single-bit error
1	RAMMBE	IRQ from Data RAM multi-bit error
2	ROMSBE	IRQ from Code RAM (ROM) single-bit error

3	ROMMBE	IRQ from Code RAM (ROM) multi-bit error
31:4	Reserved	Reserved

4.1.15 RAM_SBE Register

The RAM SBE register provides a count of the number of Data RAM single-bit errors that have been generated. A zero value can be written to this register to clear it. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
15:0	COUNT	Error Count	0
31:16	Reserved	Reserved, Reads as 0	

4.1.16 RAM_MBE Register

The RAM MBE register provides a count of the number of Data RAM multi-bit errors that have been generated. A zero value can be written to this register to clear it. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
15:0	COUNT	Error Count	0
31:16	Reserved	Reserved, Reads as 0	

4.1.17 ROM_SBE Register

The ROM SBE register provides a count of the number of Code RAM(ROM) single-bit errors that have been generated. A zero value can be written to this register to clear it. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
15:0	COUNT	Error Count	0
31:16	Reserved	Reserved, Reads as 0	

4.1.18 ROM_MBE Register

The ROM MBE register provides a count of the number of Code RAM (ROM) multi-bit errors that have been generated. A zero value can be written to this register to clear it. This register is only reset by a Power-On-Reset.

Bit	Symbol	Description	Power-on-Reset value
15:0	COUNT	Error Count	0
31:16	Reserved	Reserved, Reads as 0	

4.1.19 IOCONFIG_CLKDIV0-7 Registers

The IOCONFIG0 to IOCONFIG7 registers provide access to the divide value used to generate the IO configuration filter clocks 0 through 7. These clocks can be used to filter glitches on the configurable IO pins or as cascade sources on the Timer/Counters. A zero value in a given register will disable the associated clock. A 1 value provides the system clock. A 2 value provides the system clock divided by 2. Note that IOCONFIGCLKDIV0 is read-only and has a fixed value of 0x0000 0001 (which is the system clock).

Bit	Symbol	Description	Reset value
31:0	COUNT	Clock divide value. 0 will disable the clock.	0x0000 0000

4.1.20 ROM_RETRIES Register

The ROM RETRIES register provides a count of the number of Code RAM (ROM) block retries that happened during the ROM code fetch on the ROM SPI port during the reset startup. This register is read only. If more than 255 retries happened, the count value will be 255.

Bit	Symbol	Description	Reset value
7:0	COUNT	Error Count	0
31:8	Reserved	Reserved, Reads as 0	

4.1.21 REFRESH_CONFIG Register

The REFRESH CONFIG register provides control of the Register Refresh Rate. Registers are clock gated to reduce power; however, clock gating prevents TMR registers from self-correcting. To allow this self-correct, the registers are refreshed periodically. The REFRESH CONFIG register controls the refresh rate.

Note that the refresh controller does not exist in the VA10800 version.

Bit	Symbol	Value	Description	Reset value
15:0	DIVCOUNT		This specifies the clock divided value (plus 1) for the refresh rate. Registers are	0x000f

			clocked every (DIVCOUNT+1)*16 cycles.	
29:16	Reserved		Reserved, Reads as 0	
31:30	TESTMODE		Special Test Mode configuration	0x0
		00/01	Normal	
		10	Force refresh off	
		11	Force refresh on constantly	

4.1.22 TIM_RESET Register

The TIM RESET register provides control of the resets to the Timer/Counters. Each bit controls the reset of a given TIM. The resets are active low, so setting a given bit to 0 will hold the related TIM in reset.

Bit	Symbol	Description	Reset value
23:0	RESET[i]	Resetn of a given TIM	0xffffff
31:24	Reserved	Reserved, Reads as 1	0xff

4.1.23 TIM_CLK_ENABLE Register

The TIM CLK ENABLE register provides control of the clock enables to the Timer/Counters. Each bit controls the clock enable signal of a given TIM. The clock enables are active high, so setting a given bit to 1 will enable the clock to a given TIM. When the given enable is 0, the related TIM will not normally be clocked (except by the refresh logic).

Bit	Symbol	Description	Reset value
23:0	RESET[i]	Clock Enable of a given TIM	0x000000
31:24	Reserved	Reserved, Reads as 0	0x00

4.1.24 PERIPHERAL_RESET Register

The PERIPHERAL RESET register provides control of the resets of selected peripherals. Each bit controls the reset of a given peripheral. The resets are active low, so setting a given bit to 0 will hold the related peripheral in reset.

Bit	Symbol	Description	Reset value
0	PORTA	Resetn of a PORTA	1
1	PORTB	Resetn of a PORTB	1
3:2	Reserved	Reserved	3
6:4	SPI[i]	Resetn of SPI ports	7
7	Reserved	Reserved	3

9:8	UART[i]	Resetn of UART ports	3
15:10	Reserved	Reserved	0x3f
17:16	I2C[i]	Resetn of I^2C ports	3
19:18	Reserved	Reserved	3
20	SYSTEM	Resetn of System Config	1
21	IRQSEL	Resetn of IRQ Selector	1
22	IOCONFIG	Resetn of IO Config	1
23	UTILITY	Resetn of Utility Peripheral	1
24	PORTIO	Resetn of PORT IO interface	1
31:25	Reserved	Reserved	0xffff

4.1.25 PERIPHERAL_CLK_ENABLE Register

The PERIPHERAL CLK ENABLE register provides control of the clock enables of selected peripherals. Each bit controls the clock enable signal of a given peripheral. The clock enables are active high, so setting a given bit to 1 will enable the clock to a given peripheral. When the given enable is 0, the related peripheral will not normally be clocked. In VA10820/VA10830 the peripherals are still clocked occasionally based on the REFRESH_CONFIG register setting.

Note: For a peripheral to access any GPIO, both the peripheral and the IOCONFIG module clocks must be enabled.

Bit	Symbol	Description	Reset value
0	PORTA	Clock Enable of PORTA	0
1	PORTB	Clock Enable of PORTB	0
3:2	Reserved	Reserved	0
6:4	SPI[i]	Clock Enable of SPI ports	0
7	Reserved	Reserved	0
9:8	UART[i]	Clock Enable of UART ports	0
15:10	Reserved	Reserved	0
17:16	I2C[i]	Clock Enable of I^2C ports	0
19:18	Reserved	Reserved	0
20	Reserved	Reserved	0
21	IRQSEL	Clock Enable of IRQ Selector	0
22	IOCONFIG	Clock Enable of IO Configuration block	0
23	UTILITY	Clock Enable of Utility	0
24	GPIO	Clock Enable of GPIO interface	0
31:25	Reserved	Reserved	0

4.1.26 Lockup Reset Register

The Lockup reset register controls if the system is reset on a processor lockup event or not.

Bit	Symbol	Value	Description	Reset value
0	LREN		Lockup Reset Enable	1
		0	Processor is not reset on lockup	
		1	Processor is reset on lockup	
31:1	Reserved		Reserved, Reads as 0	

4.1.27 EF_CONFIG Register

This is a read-only register that returns the EF_CONFIG value read from the eFuse block at boot time. Programming the eFuse configuration controls the BOOT parameters as defined in the below table. The eFuse contents can be modified by the user as detailed in section 5.3.

Bit	Reset State	Name	Description
1:0	0x1	ROM_SPEED	This value specifies the speed of ROM_SCK: 0 – CLK divide by 2 (@50MHz => 25MHz) 1 – CLK divide by 6 (@50MHz => 8.33MHz) 2 – CLK divide by 12 (@50MHz => 4.2MHz) 3 – CLK divide by 52 (@50MHz => 962kHz)
4:2	0x0	ROM_SIZE	This value specifies how much of the full 128K byte address space is loaded from the external SPI memory after a reset. The values are: 0 – Full 128k Bytes of address space 1 – First 64K bytes of address space 2 – First 32K bytes of address space 3 – First 16K bytes of address space 4 – First 8K bytes of address space 5 – First 4K bytes of address space The part of the address space not loaded is initialized to zero.
5	0	ROM_NOCHECK	When set to 1, the ROM check is skipped. When set to 0, the ROM check is enabled. In ROM check mode, each 128-byte block of data from the SPI ROM is read twice. If the two reads get different results, the process is repeated until it read the same twice.
8:6	0x4	BOOT_DELAY	This value specifies the boot delay from the end of the Power-On-Sequence to the release of Reset. The delay

			is based on cycles of the internal nominal 1MHz oscillator. 0 – 1 cycles (~ 0ms) 1 – 1000 cycles (~1ms) 2 – 3000 cycles (~3ms) 3 – 10000 cycles (~10ms) 4 – 30000 cycles (~30ms) 5 – 100000 cycles (~100ms) 6 – 300000 cycles (~300ms) 7 – 500000 cycles (~500ms)
16:9	0x03	ROM_READ	SPI ROM read instruction code.
21:17	0x00	ROM_LATENCY	Number of bits of latency from Address until data from the SPI ROM.
23:22	0x1	ROM_ADDRESS	Rom Address Mode. Specifies the number of bits/bytes to use for addressing the SPI ROM. 0 – 16 bit / 2 bytes 1 – 24 bit / 3 bytes 2 – 32 bit / 4 bytes
24	1	ROM_DLYCAP	ROM SPI Delayed capture. When 1, the ROM_SI data is captured on clock falling edge instead of the normal SPI mode of the rising edge. This allows almost a full SCK clock cycle for the SPI ROM to respond from address to data. This mode allows a faster SPI cycle time.
25	0	ROM_STATUS	In this mode, the first data byte from the SPI ROM following an address is taken as a status byte. A Zero status data byte indicates valid data follows. A non-zero status data byte indicates the device is busy, and that the SPI transaction should be restarted.
26-30	0x0		These bits control internal read timing and must be maintained at this value (0x0).
31	0	ENABLE	Enables the config settings. When this bit is 0, the other bits in this register are ignored. When this bit is 1, the other bits in this register are used.

4.1.28 EF_ID Register

This is a read-only register that returns the EF_ID value read from the eFuse block at boot time. Production devices will have a unique value stored in this location when shipped from Vorago. It is possible via JTAG commands for end users to program IDs in this 32-bit register. See section 5.3 for details on how to program the eFuse.

4.1.29 PROCID Register

This is a read-only register that identifies the processor ID.

Part Number	PROCID
VA10820/VA10830	0x0400 37E3
VA10800	0x0400 47E3

4.1.30 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0180 07e1).

4.2 IRQ Selector Peripheral (Software label = IRQSEL)

The IRQ selector peripheral contains registers that configure which internal interrupt sources are connected to the interrupt signals feeding the processor NVIC (Nested Vectored Interrupt Controller). The NVIC controller provides 32 maskable interrupts, and one Non-Maskable Interrupt (NMI). In addition, an interrupt source can be configured to generate a RXEV event signal to the processor, trigger the WatchDog reset condition, or trigger the MemoryError reset condition.

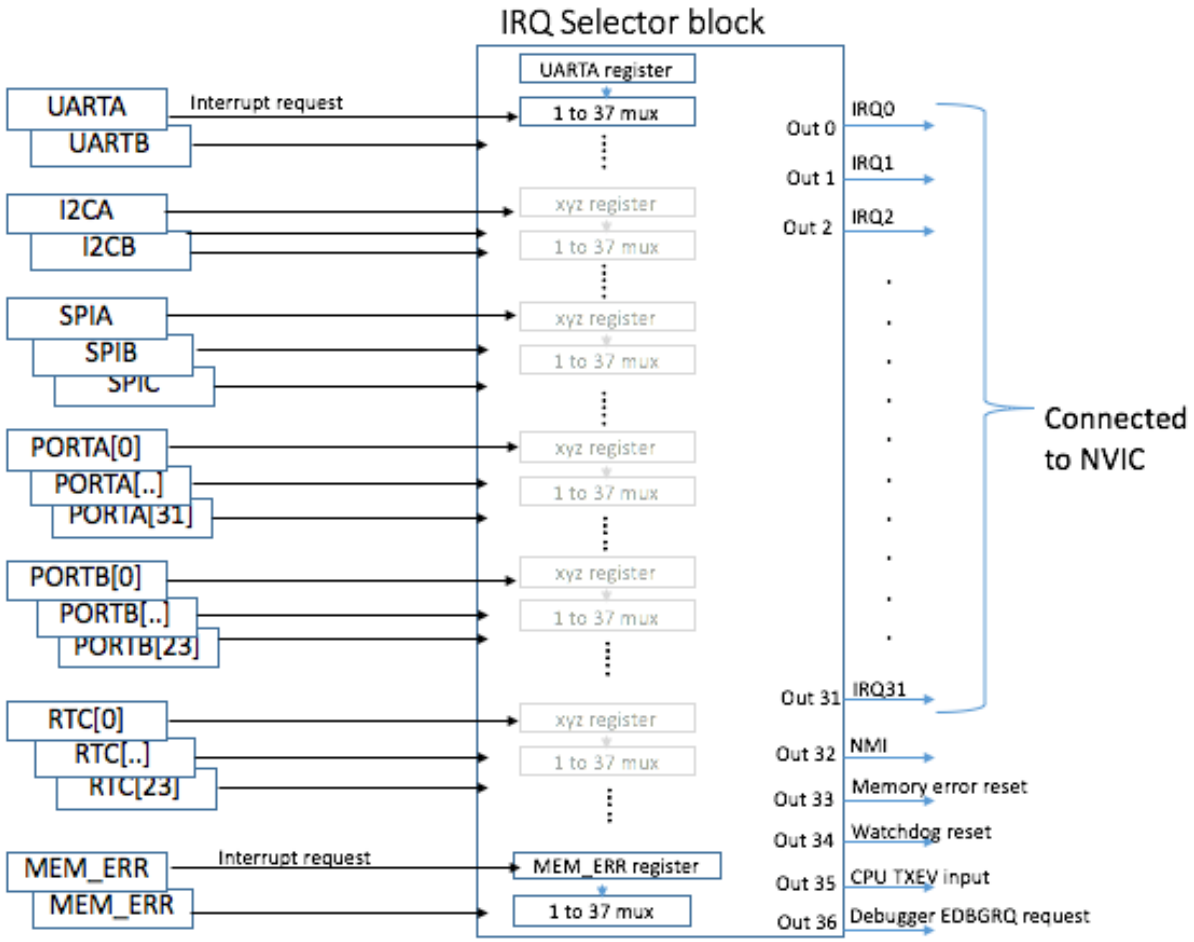


Figure 2 - Block diagram of IRQ selector peripheral

The IRQ Selector provides one register for each possible internal interrupt source. This register contains the index number of which destination it is connected to. Multiple internal interrupt sources can be connected to the same destination; in which case all the sources are ORed together.

The peripheral must have its clock enabled to update any of the registers. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. Once the peripheral has been configured, the clock need not be enabled to allow interrupts to pass through the block.

Table 2 - IRQ Selector Peripheral registers

Name	Access	Address offset	Description	Reset value
PORTA[0]	RW	0x000	PORTA Bit 0	0XFFFF FFFF
PORTA [1-30]	RW	0x004-0x078	PORTA Bits 1- 30	0XFFFF FFFF
PORTA [31]	RW	0x07C	PORTA Bit 31	0XFFFF FFFF
PORTB[0]	RW	0x080	PORTB Bit 0	0XFFFF FFFF
PORTB [1-22]	RW	0x084-0x0D8	PORTB Bits 1- 22	0XFFFF FFFF
PORTB [23]	RW	0x0DC	PORTB Bit 31	0XFFFF FFFF
-	-	0xE0-0xFC	Reserved	0XFFFF FFFF
TIM[0]	RW	0x100	TIM0	0XFFFF FFFF
TIM[1-23]	RW	0x104-0x058	TIM1-23	0XFFFF FFFF
TIM[24]	RW	0x15C	TIM24	0XFFFF FFFF
-	-	0x160-0x17C	Reserved	0XFFFF FFFF
UART[0]	RW	0x180	UART 0 IRQ	0XFFFF FFFF
UART[1]	RW	0x184	UART 1 IRQ	0XFFFF FFFF
-	-	0x188-0x18C	Reserved	0XFFFF FFFF
SPI[0]	RW	0x190	SPI 0 IRQ	0XFFFF FFFF
SPI[1]	RW	0x194	SPI 1 IRQ	0XFFFF FFFF

SPI[2]	RW	0x198	SPI 2 IRQ	0XFFFF FFFF
-	RW	0x19C	Reserved	0XFFFF FFFF
I2C_MS[0]	RW	0x1A0	I^2C 0 Master IRQ	0XFFFF FFFF
I2C_MS[1]	RW	0x1A4	I^2C 1 Master IRQ	0XFFFF FFFF
	-	0x1A8-0x1AC	Reserved	0XFFFF FFFF
I2C_SL[0]	RW	0x1B0	I^2C 0 Slave IRQ	0XFFFF FFFF
I2C_SL[1]	RW	0x1B4	I^2C 1 Slave IRQ	0XFFFF FFFF
	-	0x1B8-0x1BC	Reserved	0XFFFF FFFF
INT_RAM_SBE	RW	0x1C0	Internal memory RAM SBE IRQ	0XFFFF FFFF
INT_RAM_MBE	RW	0x1C4	Internal memory RAM MBE IRQ	0XFFFF FFFF
INT_ROM_SBE	RW	0x1C8	Internal memory ROM SBE IRQ	0XFFFF FFFF
INT_ROM_MBE	RW	0x1CC	Internal memory ROM MBE IRQ	0XFFFF FFFF
TXEV	RW	0x1D0	Processor TXEV	0XFFFF FFFF
-	-	0x1D4-0x7FC	Reserved	-
IRQS[0]	RO	0x800	Current IRQ0 Status	-
IRQS[1-30]	RO	0x804-0x878	Current IRQ1-30 Status	-
IRQS[31]	RO	0x87C	Current IRQ31 Status	-
-	-	0x880-0x8E4	Reserved	-
EDBGRQ	RO	0x8E8	Current External Debug Request	-
MERESET	RO	0x8EC	Current Memory Error Reset	-
WATCHDOG	RO	0x8F0	Current Watchdog Reset	-
RXEV	RO	0x8F4	Current RXEV Status	-
NMI	RO	0x8F8	Current NMI Status	-

-	RW	0x900-0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0181 07e1

4.2.1 Interrupt Select Register

Each of the interrupt select registers (PORTA[0] through TXEV) contains a value that determines which destination the interrupt source is connected to. When multiple interrupt sources are connected to the same destination, the final destination receives the logical OR of the configured sources.

Bit	Symbol	Value	Description	Reset value
31:0			Interrupt Selection Value	0XFFFF FFFF
	IRQ_DST_IRQ_0	0x0000 0000	Processor IRQ0	
	IRQ_DST_IRQ_X	0x0000 00001-0x0000 0001E	Processor IRQ1-30	
	IRQ_DST_IRQ_31	0x0000 001F	Processor IRQ31	
		0x0000 0020-0xFFFF FFF9	Reserved	
	IRQ_DST_EDBGRQ	0XFFFF FFFA	Debugger EDBGRQ request	
	IRQ_DST_MERESET	0XFFFF FFFB	Reset Controller - Memory Error	
	IRQ_DST_WATCHDOG	0XFFFF FFFC	Reset Controller - Watchdog	
	IRQ_DST_RXEV	0XFFFF FFFD	Processor Event (RXEV)	
	IRQ_DST_NMI	0XFFFF FFFE	Processor NMI	
	IRQ_DST_NONE	0XFFFF FFFF	Disabled, no destination selected	

4.2.2 Interrupt Status Register

Each of the interrupt status registers (IRQS[0] through NMI) contains a value that indicates if the given Interrupt (or pseudo interrupt) is active.

Bit	Symbol	Description	Reset value
0	ACTIVE	Interrupt is active	0
31:1	-	Reserved, Reads as 0	0

4.2.3 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0181 07e1).

4.3 IO Configuration Peripheral (Software label = IOCONFIG)

The following diagram shows a general block diagram of the Interface associated with each IO pin.

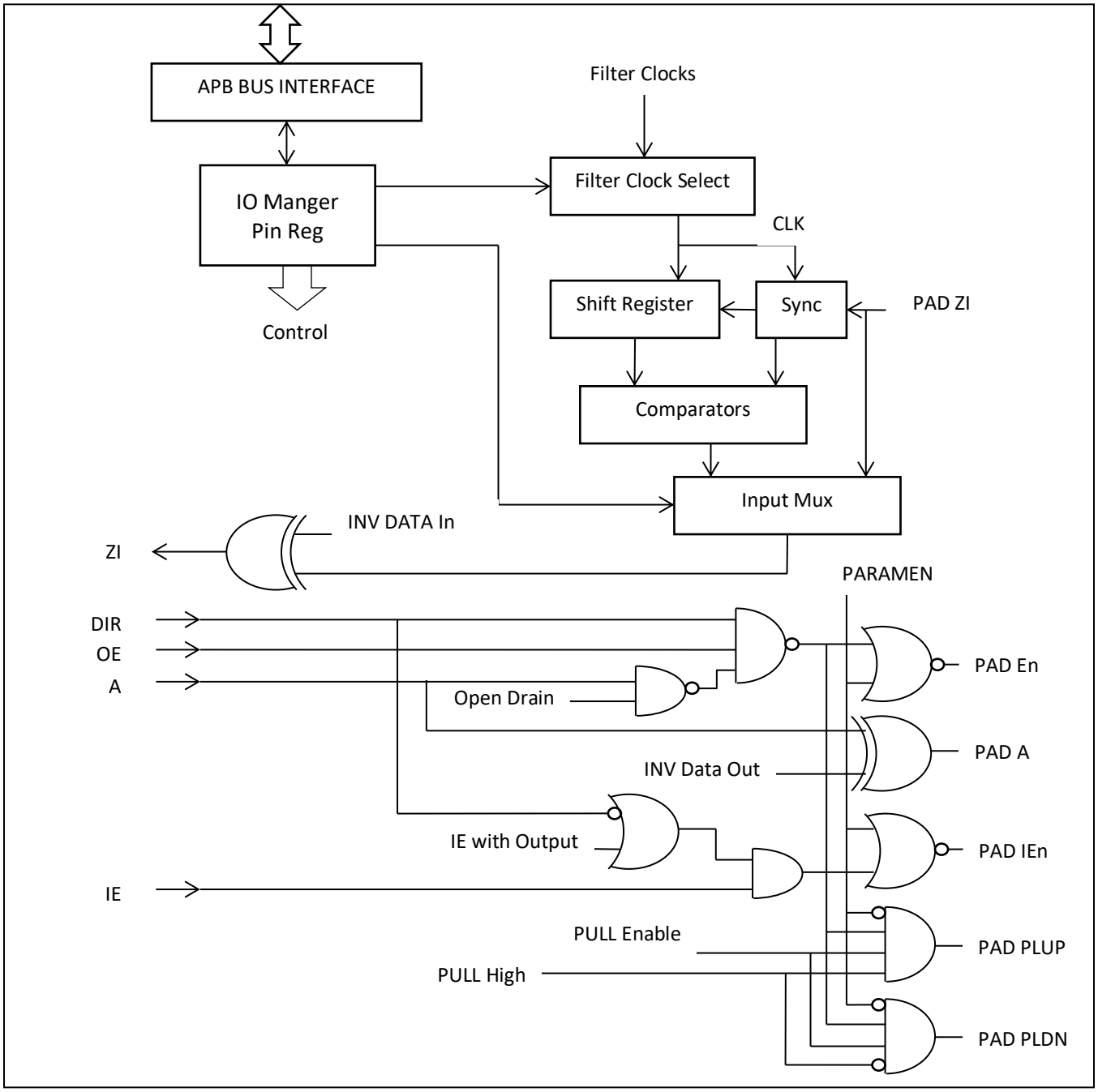


Figure 3 – IO Interface Block Diagram

The IO configuration peripheral contains registers that configure behavior and function associated with the GPIO pins. Input pins can be configured with: glitch filtering or clock synchronization logic, pull-up, pull-down, and input inversion logic. Outputs can be configured as open-drain and output inversion logic.

Input glitch filtering is done by sampling the signal from 2 to 5 times and requiring that all the samples be the same value. The sample clock can be picked from 8 different clock sources that are defined in the System Configure Peripheral as IO Filter clocks 0-7 (0 is always the system clock).

The default reset value results in clock synchronization being selection on all inputs. When a pin is configured as MISO of a master SPI port it is recommended that the clock synchronization be turned off to reduce the input latency. This is valid as this signal should already be synchronized to the output SPI clock.

The peripheral must have its clock enabled to update any of it configuration registers. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. The clocks to input filtering logic are not disabled when the peripheral clock is disabled. As such, once the peripheral has been configured, the clock need not be enabled to allow IO function to operate.

The table below shows the list of IO Configuration Registers.

Table 3 - IO Configuration Peripheral Registers

Name	Access	Address offset	Description	Reset value
PORTA[0]	RW	0x000	PORTA Bit 0	0X0000 0000
PORTA [1-30]	RW	0x004-0x078	PORTA Bits 1- 30	0X0000 0000
PORTA [31]	RW	0x07C	PORTA Bit 31	0X0000 0000
PORTB[0]	RW	0x080	PORTB Bit 0	0X0000 0000
PORTB [1-22]	RW	0x084-0x0D8	PORTB Bits 1- 22	0X0000 0000
PORTB [23]	RW	0x0DC	PORTB Bit 23	0X0000 0000
-	RW	0x0E0-0x1DC	Reserved	-

-	RW	0x1E0-0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0182 07e1

4.3.1 IO Configuration Register

Each of the IO configuration registers contains a value that configures a given IO pin. Open drain mode is not a true open drain, but is emulated by only enabling the output driver when the output signal is being driven low.

Bit	Symbol	Value	Description	Reset value
2:0	FLTTYPE		Configures the Filter type	0
		0	Synchronize to system clock only	-
		1	Direct input without synchronization	-
		2-5	Filter 1-4 clock cycles (Sample 2-5 times)	-
		6-7	Reserved	
5:3	FLTCLK	0-7	IO Filter Clocks Select 0-7 (0 is always system clock)	0
6	INVINP		Enable Input Inversion (1 inverts)	0
7	IEWO		Enable Input even when in output mode. In this mode the input receiver is enabled even if the direction is configured as an output. This allows monitoring of output values.	0
8	OPENDRN		Enable Open-Drain Mode (1 enables)	0
9	INVOUT		Enable Output Inversion (1 inverts)	0
10	PLEVEL		Controls the direction of the PEN pull value. 1 is pull-up, 0 is pull-down.	0
11	PEN		Enable Internal Pull up/down on the pins.	0
12	PWOA		Enable Pull up/down even when output is active. The Default is to disable pull up/down when output is actively driven. This bit enables the pull up/down all the time.	0
14:13	FUNSEL		Selects in function connected to the given pin. See Function Selection below.	0
15	-		Reserved for FUNSEL	0
16	IODIS		Disable the IO buffer. When set this turns off both the input and output buffer of the IO cell. Thus totally disabling the IO cell. When in this mode the input value will appear to be 0.	0
31:17			Reserved, Read as 0	

4.3.2 Loopback Mode

A special Loop-Back mode is enabled when IODIS is enabled (is 1) and IEWO is enabled (is 1). In this mode, the internal GPIO is connected to itself, if no function is selected (FUNSEL), or the GPIO is connected to the selected function (FUNSEL).

4.3.3 Function Selections

The default configuration is for all the pins to be assigned to the GPIO pins PORTA and PORTB. Using the FUNSEL bits of each IO Configuration register, the pins can be connected to alternate peripherals, such as the UART or SPI signals. The following table shows the available function selections. Each GPIO pin can be configures to be 1 of the 4 columns independent of the other GPIO pins.

Note the GPIO PORTA and PORTB input registers can always read the pin input value no matter what function is selected on the pins. The UART and SPI interface will only receive the value on the pins when the proper function mode is selected for the pin. When nothing is configured to be connected to the UART or SPI peripheral signals, these peripherals will receive a 1 on the SPI_SSELx and UARTx_RX signals, and 0 on all other signals.

Note: If multiple pins are configured as the same UART or SPI function that is an input, the UART/SPI input will receive the logical AND or OR of the pin values. SPI_SSELx and UARTx_RX pins are logical ANDed while all other pins are logical ORed. In general the part should not be configured to do this.

When a pin configured as one of the selection choices (other than GPIO), the pin direction is configured based selected peripheral. So, when connected to an SPI signal the direction is based on the SPI peripherals master/slave mode selection. When connected to a UART the signal direction is output for Tx/RTSn and input for Rx/CTSn. When connect to TIM or other status signals the direction is output.

Table 4 - IO Function Selection

Pin/Function-0	Function-1	Function-2	Function-3
PORTA[31]	SPI_SCKA	TIM[23]	UARTA_TX
PORTA[30]	SPI_MOSIA	TIM[22]	UARTA_RX
PORTA[29]	SPI_MISOA	TIM[21]	UARTA_RTSn
PORTA[28]	SPI_SSELAn[0]	TIM[20]	UARTA_CTSn
PORTA[27]	SPI_SSELAn[1]	TIM[19]	UARTB_TX
PORTA[26]	SPI_SSELAn[2]	TIM[18]	UARTB_RX
PORTA[25]	SPI_SSELAn[3]	TIM[17]	UARTB_RTSn
PORTA[24]	SPI_SSELAn[4]	TIM[16]	UARTB_CTSn

PORTA[23]	SPI_SSELAn[5]	SPI_SSELBn[5]	SPI_SSELCn[1]
PORTA[22]	SPI_SSELAn[6]	SPI_SSELBn[6]	SPI_SSELCn[2]
PORTA[21]	SPI_SSELAn[7]	SPI_SSELBn[7]	SPI_SSELCn[3]
PORTA[20]	SPI_SSELCn[1]	SPI_SCKB	SPI_SSELCn[4]
PORTA[19]	SPI_SSELCn[2]	SPI_MOSIB	UARTB_TX
PORTA[18]	SPI_SSELCn[3]	SPI_MISOB	UARTB_RX
PORTA[17]	SPI_TXEMPTYA	SPI_SSELBn[0]	UARTA_TX
PORTA[16]	SPI_TXEMPTYB	SPI_SSELBn[1]	UARTA_RX
PORTA[15]	TIM[15]	SPI_SSELBn[2]	UARTA_RTSn
PORTA[14]	TIM[14]	SPI_SSELBn[3]	UARTA_CTSn
PORTA[13]	TIM[13]	SPI_SSELBn[4]	UARTB_RTSn
PORTA[12]	TIM[12]	SPI_SSELBn[5]	UARTB_CTSn
PORTA[11]	TIM[11]	SPI_SSELBn[6]	LOCKUP (Out)[2]
PORTA[10]	TIM[10]	SPI_SSELBn[7]	SYSRESETREQ (Out)[2]
PORTA[9]	TIM[9]	UARTA_TX	SLEEPING (Out)[2]
PORTA[8]	TIM[8]	UARTA_RX	HALTED (Out)[2]
PORTA[7]	TIM[7]	UARTA_RTSn	TXEV (Out)[2]
PORTA[6]	TIM[6]	UARTA_CTSn	RXEV (In)[2]
PORTA[5]	TIM[5]	UARTB_RTSn	EDBGRQ (In)[2]
PORTA[4]	TIM[4]	UARTB_CTSn	
PORTA[3]	TIM[3]	UARTB_TX	I2CB_SCL[1]
PORTA[2]	TIM[2]	UARTB_RX	I2CB_SDA[1]
PORTA[1]	TIM[1]	I2CA_SCL[1]	DBGRESTARTED (Out)[2]
PORTA[0]	TIM[0]	I2CA_SDA[1]	DBGRESTART (In)[2]
PORTB[23]	UARTA_TX	SPI_SSELCn[2]	TIM[23]
PORTB[22]	UARTA_RX	SPI_SSELCn[1]	TIM[22]
PORTB[21]	UARTB_TX	UARTA_RTSn	TIM[21]
PORTB[20]	UARTB_RX	UARTA_CTSn	TIM[20]
PORTB[19]	SPI_SCKB	UARTB_TX	TIM[19]
PORTB[18]	SPI_MOSIB	UARTB_RX	TIM[18]
PORTB[17]	SPI_MISOB	UARTB_RTSn	TIM[17]
PORTB[16]	SPI_SSELBn[0]	UARTB_CTSn	TIM[16]
PORTB[15]	SPI_SSELBn[1]	SPI_SCKB	TIM[15]
PORTB[14]	SPI_SSELBn[2]	SPI_MOSIB	TIM[14]
PORTB[13]	SPI_SSELBn[3]	SPI_MISOB	TIM[13]
PORTB[12]	SPI_SSELBn[4]	SPI_SSELBn[0]	TIM[12]
PORTB[11]	SPI_SSELBn[5]	SPI_SSELBn[1]	TIM[11]

PORTB[10]	SPI_SSELBn[6]	SPI_SSELBn[2]	TIM[10]
PORTB[9]	UARTA_TX	SPI_SCKA	SPI_SSELCn[1]
PORTB[8]	UARTA_RX	SPI_MOSIA	SPI_SSELCn[2]
PORTB[7]	UARTB_TX	SPI_MISOA	SPI_SSELCn[3]
PORTB[6]	UARTB_RX	SPI_SSELAn[0]	TIM[6]
PORTB[5]	SPI_SCKB	SPI_SSELAn[6]	TIM[5]
PORTB[4]	SPI_MOSIB	SPI_SSELAn[5]	TIM[4]
PORTB[3]	SPI_MISOB	SPI_SSELAn[4]	TIM[3]
PORTB[2]	SPI_SSELBn[0]	SPI_SSELAn[3]	TIM[2]
PORTB[1]	SPI_SSELBn[1]	SPI_SSELAn[2]	TIM[1]
PORTB[0]	SPI_SSELBn[2]	SPI_SSELAn[1]	TIM[0]

Notes:

1 – This I^2C mapping to GPIO pins is primarily intended for device testing.

When I^2C signals are mapped to GPIO:

- The dedicated I^2C signal is disabled.
- The I^2C buffer is only emulated by disabling the output driver when a 1 is to be output.
- There is no slew rate control on the output.
- There is not analog glitch filter on the input.
- The IO Configuration FLTTYPE should be set to 1 = No Synchronization.
- The IO Configuration IEWO must be set to 1 = Input enabled when Output is enabled.
- The IO Configuration OPENDRN must bet set to 1 = OpenDrain emulation.

2 – These processor status/control signals are available for special purpose use. They are not intended for general use.

4.3.4 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0182 07e1).

4.4 In-package NVM Memory (VA10830 Only)

The VA10830 auto-detects the power-up condition using low voltage detect circuits. The processor begins operation by automatically clearing the contents of all internal SRAMs to 0x00. The processor will load the internal SRAM code memory from an SPI based NVM memory located in the same package via externally connected ROM SPI pins. An approximately 20 MHz internal oscillator will be used for this stage of boot activity. The loading of the internal SRAM will take place after the BOOT_DELAY time seen in the SYSCONFIG's EF_CONFIG register. After loading the code memory, the processor follows a normal Arm Cortex M0 start-up sequence.

The VA10830 device incorporates a FRAM device consistent with the electrical interface of the Cypress FM25V20A. See the datasheet for this device at cypress.com for more details.

Although both the MCU die and the NVM die are included in the package, connections between them must be done externally. Figure 4 shows the connections from the MCU to the FRAM externally to the package. ROM_MISO, ROM_MOSI, ROM_SCK, and ROM_SS are external package pins. Likewise, NVM_MISO, NVM_MOSI, NVM_SCK, and NVM_CSn are external package pins.

A flash loader interface for development tools will be provided by VORAGO to program the user software into the internal NVM.

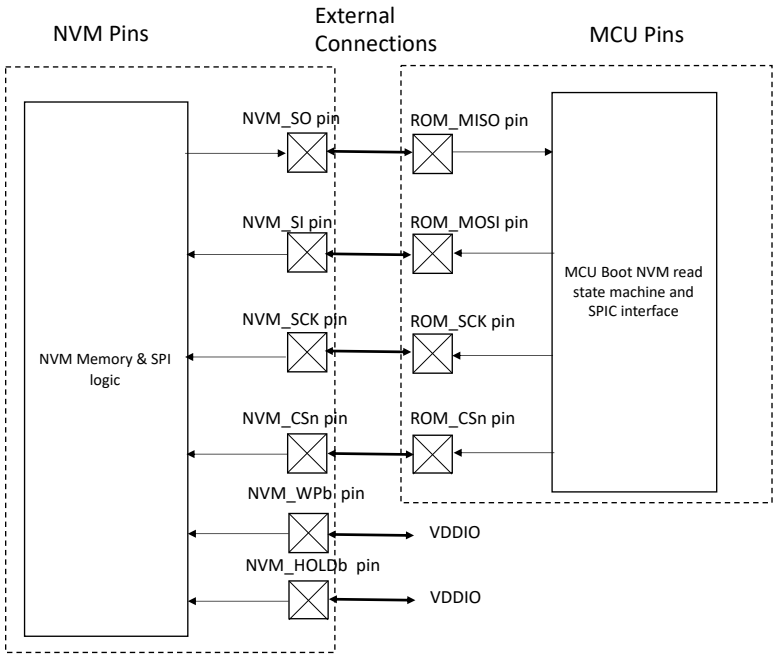


Figure 4 – NVM Connections To MCU

4.4.1 SPI NVM Interface

This section contains information on the SPI memory interface to the internal NVM that is needed for programming or verifying the internal NVM via SPIC. The only way to program the internal NVM is via the MCU SPIC interface.

Note: To enable writing to the internal NVM, external pin NVM_WPb must be set to VDDIO.

Note: For proper operation, external pin NVM_HOLDb must always be set to VDDIO.

4.4.2 SPI NVM Interface

ROM_MISO, ROM_MOSI, ROM_SCK, and ROM_CSn are external package pins. Likewise, NVM_MISO, NVM_MOSI, NVM_SCK, and NVM_CSn are external package pins. The MCU SPI pins must be connected externally to the NVM SPI pins on the VA10830.

External NVM Pin Name	I/O Type (NVM Chip)	Pin Description	I/O Type (MCU pin)	External SPIC MCU Connection
NVM_CSn	Input	Chip select	Output	ROM_CSn
NVM_SCK	Input	Serial clock	Output	ROM_SCK
NVM_MOSI	Input	Serial input	Output	ROM_MOSI
NVM_MISO	Output	Serial output	Input	ROM_MISO
NVM_WPb	Input	Write protect	N/A	Must be tied to VDDIO to write to the NVM
NVM_HOLDb	Input	Unused	N/A	Must always be tied to VDDIO

4.4.3 Internal NVM Registers (VA10830 only)

The internal NVM registers are accessible via SPIC. The NVM supports both SPI mode 0 (clock polarity CPOL=0, clock phase CPHA=0) and SPI mode 3 (clock polarity CPOL=1, clock phase CPHA=1), both with the MSB sent first.

The communication protocol from the VA10830 to the NVM device is controlled by opcodes. These opcodes specify the commands from the MCU (SPI bus master) to the NVM (SPI bus

slave). The first byte transmitted by the bus master is the opcode. Following the opcode, any necessary address and data bytes are transmitted.

4.4.4 NVM Memory Command Information

There are nine opcode commands that can be issued from the MCU to the NVM. These opcodes are listed in the table below.

Name	Description	Opcode
WREN	Set write enable latch	0x06
WRDI	Reset write enable latch	0x04
RDSR	Read Status register	0x05
WRSR	Write Status register	0x01
READ	Read memory data	0x03
FSTRD	Fast read memory data	0x0B
WRITE	Write memory data	0x02
SLEEP	Enter Sleep mode	0xB9
RDID	Read device ID	0x9F

4.4.5 Write Enable (WREN)

The internal NVM will power up with writes disabled. The single-byte WREN (opcode only) command must be issued before any write operation. Sending the WREN command allows the MCU to issue opcodes for write operations, including writing the Status Register (WRSR) and writing memory locations (WRITE). To enable writing to the internal NVM, including the Status Register, external pin NVM_PROTn must be set to VDDIO.

4.4.6 Status Register (RDSR and WRSR opcodes)

The NVM Status Register is accessed by issuing the RDSR (for read) or the WRSR (for write) opcodes. A WREN command must be issued prior to a WRSR.

The RDSR command allows the MCU to verify the contents of the Status Register to determine the state of the write-protection features. Following the RDSR opcode, the NVM will return one byte with the contents of the Status Register.

The WRSR commands allow the MCU to change the write-protection features. Following the WRSR command, the SPI must send the new data byte to be written to the Status Register.

<table>
<tr><th>Bit Position</th><th>Description</th><th>Opcode</th></tr>
<tr><td>0</td><td>Don’t care</td><td>Reads as 0</td></tr>
<tr><td>1</td><td>Write Enable (WEL)</td><td>Indicates whether the device is write-enabled.
WEL = 1 -> Write enabled
WEL = 0 -> Write disabled</td></tr>
<tr><td>2</td><td>Block Protect bit 0</td><td rowspan="2">Used for block protection
<table><tr><th>BP1</th><th>BP0</th><th>Protected address range</th></tr><tr><td>0</td><td>0</td><td>None</td></tr><tr><td>0</td><td>1</td><td>0x30000 to 0x 3FFFF</td></tr><tr><td>1</td><td>0</td><td>0x20000 to 0x 3FFFF</td></tr><tr><td>1</td><td>1</td><td>0x00000 to 0x 3FFFF</td></tr></table></td></tr>
<tr><td>3</td><td>Block Protect bit 1</td></tr>
<tr><td>4-5</td><td>Don’t care</td><td>Reads as 0</td></tr>
<tr><td>6</td><td>Don’t care</td><td>Reads as 1</td></tr>
<tr><td>7</td><td>Write Protect Enable bit (WPEN)</td><td>Used to enable the function of Write Protect Pin (NVM_PROTn)
<table><tr><th>WEL</th><th>WPEN</th><th>NVM _PROTn</th><th>Protected Blocks</th><th>Unprotected Blocks</th><th>Status Register</th></tr><tr><td>0</td><td>X</td><td>X</td><td>Protected</td><td>Protected</td><td>Protected</td></tr><tr><td>1</td><td>0</td><td>X</td><td>Protected</td><td>Unprotected</td><td>Unprotected</td></tr><tr><td>1</td><td>1</td><td>0</td><td>Protected</td><td>Unprotected</td><td>Protected</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Protected</td><td>Unprotected</td><td>Unprotected</td></tr></table></td></tr>
</table>

4.4.7 Writing the NVM Memory (WRITE opcode)

All write operations must begin with the WREN command. Only unprotected memory locations are writable. To write a byte of data to the NVM, the WRITE opcode must be issued, followed by a three-byte address containing the 18-bit address of the first data byte to be written. Subsequent bytes are data bytes (MSB first), which are written sequentially. The

NVM internal address is incremented automatically as long as the NVM chip-enable (ROM_SS) pin is held low, and the rising edge of ROM_SS will terminate the write operation.

4.4.8 Resetting the Write-Enable (WRDI opcode)

To reset the write-enable of the NVM, the WRDI opcode must be issued. The WRDI command is a single byte (opcode only). The WRDI opcode will clear the WEL bit of the Status Register.

4.4.9 Reading the NVM Memory (READ opcode)

To read a byte of data to the NVM, the READ opcode must be issued, followed by a three-byte address containing the 18-bit address of the first data byte to be written. The NVM will drive out the read data bytes (MSB first) on each set of eight ROM_SCK cycles. The NVM internal address is incremented automatically as long as the NVM chip-enable (ROM_SS) pin is held low, and the rising edge of ROM_SS will terminate the read operation.

4.4.10 Fast-Reading the NVM Memory (FSTRD opcode)

The FSTRD opcode is similar to the READ opcode except that a 'dummy' byte must be issued between the three bytes of address and the first data byte being driven out from the NVM. Subsequent data bytes will follow the first byte as in a READ operation. The NVM internal address is incremented automatically as long as the NVM chip-enable (ROM_SS) pin is held low, and the rising edge of ROM_SS will terminate the read operation.

4.4.11 Device ID Register (RDID opcode)

The NVM Device ID Register is accessed by issuing the RDID opcode. Following the RDID opcode, the NVM will return nine bytes. The first six bytes will read 0x7F. The seventh byte will return the manufacturer ID, 0xC2. The eighth and ninth bytes will return the two-byte product ID, 0x2508.

4.4.12 Entering Sleep Mode (SLEEP opcode)

The NVM can be put into a low-power mode by issuing the SLEEP opcode. The SLEEP command is a single byte (opcode only). While NVM is in SLEEP mode, the ROM_SCK and ROM_MOSI pins are ignored, but the NVM continues to monitor the state of the ROM_SS pin. On the next falling edge of ROM_SS, the NVM device will wake up from SLEEP and return to normal operation.

4.5 Utility Peripheral (Software label = UTILITY)

The utility peripheral contains registers that access dedicated hardware support engines. This includes:

- EDAC (Error Detect and Correct) syndrome encoders and decoders.
  - 7-bit code for 32-bit data words
  - 8-bit code for 64-bit data words
  - 4x5 bit codes for 4x8 bit data words
  - These can be used to generate and check EDAC values used on memories within various VORAGO parts.

The EDAC syndrome encoders and checkers all used the same input registers (SYND_DATA0, SYND_DATA1, and SYND_SYND). The outputs of each of the different encoders and checkers are available in unique registers.

The peripheral must have its clock enabled to update any of it registers. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral.

Table 5 - Utility Peripheral Registers

Name	Access	Address offset	Description	Reset value
SYND_DATA0	RW	0x000	Data Register 0	0x0000 0000
SYND_DATA1	RW	0x004	Data Register 1	0x0000 0000
SYND_SYND	RW	0x008	Syndrome Data Register	0x0000 0000
SYND_ENC_32	R	0x00C	32/37 Edac Encode Register	-
SYND_CHECK_32_DATA	R	0x010	32/37 Edac Decode Data Register	-
SYND_CHECK_32_SYND	R	0x014	32/37 Edac Decode Syndrome Register	-
SYND_ENC_64	R	0x018	64/72 Edac Encode Register	-
SYND_CHECK_64_DATA0	R	0x01c	64/72 Edac Decode Data0 Register	-
SYND_CHECK_64_DATA1	R	0x020	32/37 Edac Decode Data1 Register	-

SYND_CHECK_64_ SYND	R	0x024	32/37 Edac Decode Syndrome Register	-
SYND_ENC_32_52	R	0x028	32/52 Edac Encode Register	-
SYND_CHECK_32_52_DATA	R	0x02c	32/52 Edac Decode Data Register	-
SYND_CHECK_32_52_ SYND	R	0x030	32/52 Edac Decode Syndrome Register	-
-	-	0x034 – 0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0084 07e1

4.5.1 SYND_DATA0/SYND_DATA1/SYND_SYND Registers

The 3 registers SYND_DATA0, SYND_DATA1, and SYND_SYND provide input values to the various EDAC encoder and decoder logic blocks. SYND_DATA0 and SYND_DATA1 provide up to a 64 bit data input register, and SYND_SYND provides up to a 20 bit Syndrome input register.

Table 6 – SYND Data 0 register (SYND_ DATA0)

Bit	Symbol	Description	Reset value
31:0	VALUE	Provides bits 31:0 of the full 64 bit DATA register	0x0000 0000

Table 7 – SYND Data 1 register (SYND_DATA1)

Bit	Symbol	Description	Reset value
31:0	VALUE	Provides bits 63:32 of the full 64 bit DATA register	0x0000 0000

Table 8 – SYND Syndrome register (SYND_SYND)

Bit	Symbol	Description	Reset value
19:0	VALUE	Provides bits 19:0 of the Syndrome input register	0x0000 0000
31:20		Reserved	

4.5.2 SYND_ENC_32 Registers

The SYND ENC 32 register provides read access to the 32/39 EDAC Syndrome encoder. The output is a 7 bit EDAC code word generated from bits 31:0 of the SYND_DATA0 register value. This is the EDAC code word used on VORAGO parts PA11S/PA12S for their internal memory.

Bit	Symbol	Description	Reset value
6:0	VALUE	Computed Syndrome Value	-
31:7		Reads as 0	

4.5.3 SYND_CHECK_32_DATA/SYND_CHECK_32_SYND Registers

The SYND_CHECK32_DATA and SYND_CHECK_32_SYND registers provide read access to the 32/39 EDAC Syndrome decoder. This is a check of the data bits 31:0 of the SYND_DATA0 register and bits 6:0 of the SYND_SYND register. SYND_CHECK_32_DATA provides the original data if no errors are detected and the corrected data if a single bit error is detected. SYND_CHECK_32_SYND provides the original syndrome if no errors are

detected and the corrected syndrome if a single bit error is detected; in addition, this register contains the check status bits.

Table 9 – SYND Check 32/39 data register (SYND_CHECK_32_DATA)

Bit	Symbol	Description	Reset value
31:0	VALUE	Corrected Data Value	-

Table 10 – SYND Check 32/39 syndrome register (SYND_CHECK_32_SYND)

Bit	Symbol	Description	Reset value
6:0	VALUE	Corrected Syndrome Value	-
7		Reads as 0	
8	SBE	Single Bit Error Detect Status	
9	MBE	Multiple Bit Error Detect Status	
31:10		Reads as 0	

4.5.4 SYND_ENC_64 Registers

The SYND ENC 64 register provides read access to the 64/72 EDAC Syndrome encoder. The output is an 8 bit EDAC code word generated from bits 63:0 of the SYND_DATA01/SYND_DATA0 register value. This is the EDAC code word used on VORAGO part families: SB018E and SB036S.

Bit	Symbol	Description	Reset value
7:0	VALUE	Computed Syndrome Value	-
31:8		Reads as 0	

4.5.5 SYND_CHECK_64_DATAx/SYND_CHECK_64_SYND Registers

The SYND_CHECK64_DATA0, SYND_CHECK64_DATA1 and SYND_CHECK_64_SYND registers provide read access to the 64/72 EDAC Syndrome decoder. This is a check of the data bits 63:0 of the SYND_DATA0/ SYND_DATA1 register and bits 7:0 of the SYND_SYND register. SYND_CHECK_64_DATA0 and SYND_CHECK_64_DATA1 provide the original data if no errors are detected and the corrected data if a single bit error is detected. SYND_CHECK_64_SYND provides the original syndrome if no errors are detected and the corrected syndrome if a single bit error is detected; in addition, this register contains the check status bits.

Table 11 – SYND Check 64/72 data register (SYND_CHECK_64_DATA0)

Bit	Symbol	Description	Reset value
31:0	VALUE	Bits 31:0 of the Corrected Data Value	-

Table 12 – SYND Check 64/72 data register (SYND_CHECK_64_DATA1)

Bit	Symbol	Description	Reset value
31:0	VALUE	Bits 63:32 of the Corrected Data Value	-

Table 13 – SYND Check 64/72 syndrome register (SYND_CHECK_64_SYND)

Bit	Symbol	Description	Reset value
7:0	VALUE	Corrected Syndrome Value	-
8	SBE	Single Bit Error Detect Status	-
9	MBE	Multiple Bit Error Detect Status	-
31:10		Reads as 0	

4.5.6 SYND_ENC_32_52 Registers

The SYND END 32_52 register provides read access to the 32/52 EDAC Syndrome encoder. The output is a 20 bit EDAC code word generated from bits 31:0 of the SYND_DATA0 register value. This is the EDAC code word used on VORAGO parts VA108X0 and GLM003A/GLM004A for internal memory. The 20-bit code is composed of four 5-bit syndromes generated for each 8 bits of the input data.

Bit	Symbol	Description	Reset value
4:0	VALUE	Computed Syndrome Value for Bits 7:0	-
9:5	VALUE	Computed Syndrome Value for Bits 15:8	-
14:10	VALUE	Computed Syndrome Value for Bits 23:16	-
19:15	VALUE	Computed Syndrome Value for Bits 31:24	-
31:20		Reads as 0	

4.5.7 SYND_CHECK_32_52_DATA/SYND_CHECK_32_52_SYND Registers

The SYND_CHECK32_52_DATA and SYND_CHECK_32_52_SYND register s provide read access to the 32/52 EDAC Syndrome decoder. This is check of the data bits 31:0 of the SYND_DATA0 register and bits 19:0 of the SYND_SYND register.
SYND_CHECK_32_52_DATA provides the original data if no errors are detected and the corrected data if a single bit error is detected. SYND_CHECK_32_52_SYND provides the original syndrome if no errors are detected and the corrected syndrome if a single bit error is detected; in addition this register contains the check status bits.

Table 14 – SYND Check 32/52 data register (SYND_CHECK_32_52_DATA)

Bit	Symbol	Description	Reset value
31:0	VALUE	Corrected Data Value	-

Table 15 – SYND Check 32/52 syndrome register (SYND_CHECK_32_52_SYND)

Bit	Symbol	Description	Reset value
19:0	VALUE	Corrected Syndrome Value	-
23:20		Reads as 0	
27:24	SBE	Single Bit Error Detect Status of each 8 bits: Bit 24 – Is status of Data Bits 7:0 Bit 25 – Is status of Data Bits 15:8 Bit 26 – Is status of Data Bits 23:16 Bit 27 – Is status of Data Bits 31:24	-
31:28	MBE	Multiple Bit Error Detect Status of each 8 bits: Bit 28 – Is status of Data Bits 7:0 Bit 29 – Is status of Data Bits 15:8 Bit 30 – Is status of Data Bits 23:16 Bit 31 – Is status of Data Bits 31:24	-

4.5.8 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0084 07e1).

4.6 General Purpose IO Peripheral (Software label = GPIO)

The following diagram shows a general block diagram of the GPIO Interface.

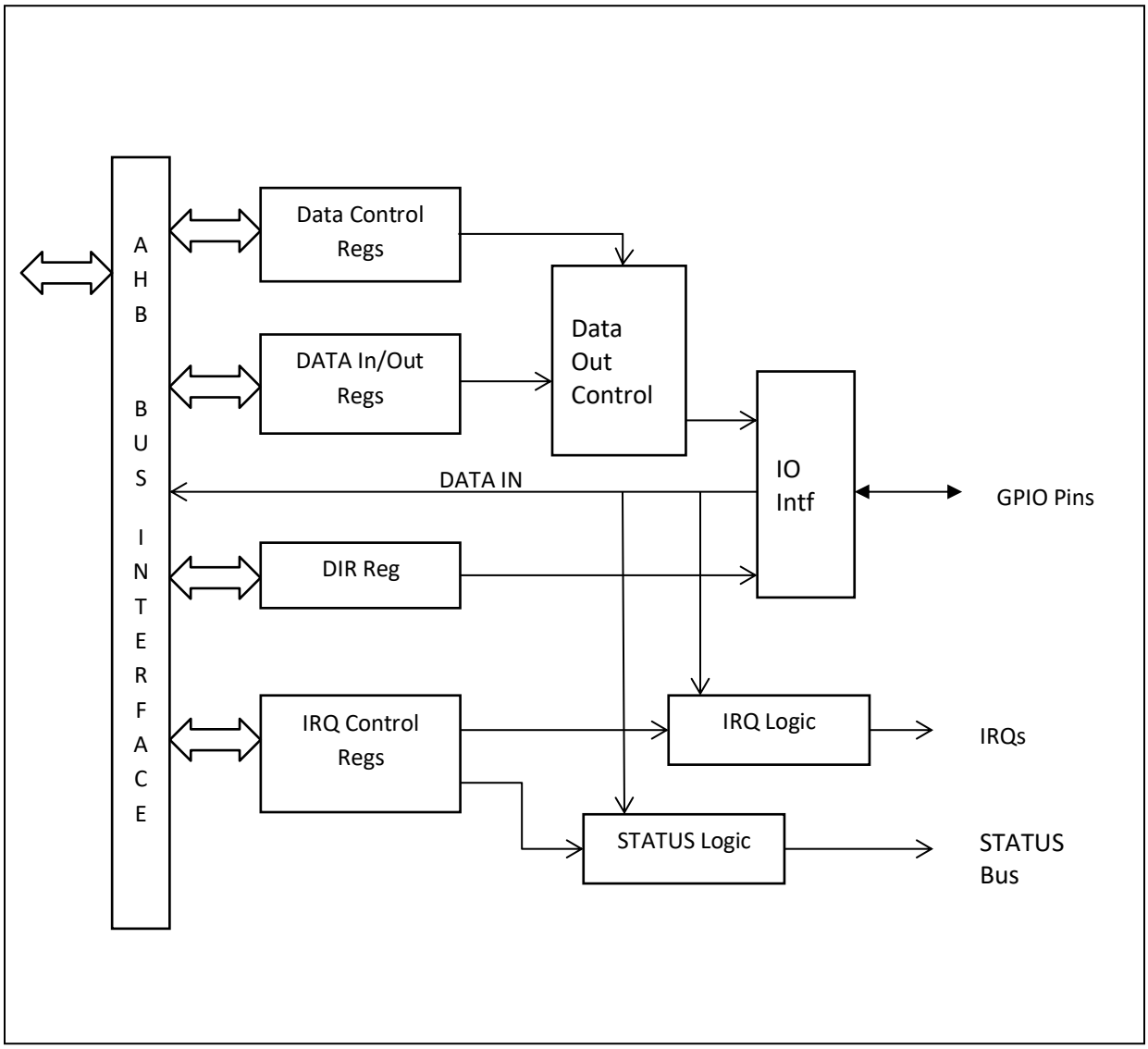


Figure 5 - GPIO Block Diagram

The general purpose IO (GPIO) peripheral consists of 2 GPIO banks. The first bank is PORTA and is 32 bits wide. The second bank is PORTB and is 24 bits wide. This gives a total of 56 GPIO pins.

The IO Configuration peripheral controls which bits of the GPIO peripheral are actually connected to the GPIO pins.

The peripheral must have its clock enabled to update any of it registers. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral.

Table 16 – GPIO Bank Location

Name	Address offset	Description
GPIO0	0x0000-0x0FFC	GPIO 0 - PORTA
GPIO1	0x1000-0x1FFC	GPIO 1 - PORTB
	0x2000-0xFFFC	Reserved

Table 17 – GPIO Peripheral Registers

Name	Access	Address offset	Description	Reset value
DATAIN	R	0x000	Data Input	-
DATAINRAW	R	0x004	Data Input Raw	-
DATAOUT	RW	0x008	Data Output	0x0000 0000
DATAOUTRAW	RW	0x00C	Data Output Raw	
SETOUT	W	0x010	Set Data Outputs to 1	-
CLROUT	W	0x014	Set Data Outputs to 0	-
TOGOUT	W	0x018	Toggle Data Outputs	-
DATAMASK	RW	0x01c	Data In/Out Mask	0x0000 0000
DIR	RW	0x020	Direction (1 is output, 0 is input)	0x0000 0000
PULSE	RW	0x024	Pulse Mode	0x0000 0000
PULSEBASE	RW	0x028	Pulse Base Value	0x0000 0000
DELAY1	RW	0x02c	Delay 1 Cycle	0x0000 0000
DELAY2	RW	0x030	Delay 2 Cycles	0x0000 0000
IRQ_SEN	RW	0x034	Interrupt Sense	0x0000 0000
IRQ_EDGE	RW	0x038	Interrupt Both Edge	0x0000 0000
IRQ_EVT	RW	0x03c	Interrupt Event	0x0000 0000
IRQ_ENB	RW	0x040	Interrupt Enable	0x0000 0000

IRQ_RAW	R	0x044	Raw Interrupt Status	0x0000 0000
IRQ_END	R	0x048	Enabled Interrupt Status	0x0000 0000
EDGE_STATUS	R	0x04c	Edge Detect Status	0x0000 0000
-	-	0x050-0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0040 07e1

4.6.1 DATAIN Register

The DATAIN register provides read access to the input data value for the GPIO port. Even when a GPIO pin is assigned to a different function (such as SPI, or UART), the data on that input pin can be read from this register. The data read is masked with the DATAMASK register, so only bits enabled by the DATAMASK will be read, non-enabled bits read as 0.

When a pin is configured as an output (either by the DIR register, or the function selection), the IENWO bit of the IO configuration register can be used to enable the input buffer for reading the output value.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Input value	-

4.6.2 DATAINRAW Register

The DATAINRAW register provides read access to the input data value for the GPIO port. Even when a GPIO pin is assigned to a different function (such as SPI, or UART), the data on that input pin can be read from this register. The data read is NOT masked with the DATAMASK register, so all bits can be read.

When a pin is configured as an output (either by the DIR register, or the function selection), the IENWO bit of the IO configuration register can be used to enable the input buffer for reading the output value.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Input value	-

4.6.3 DATAOUT Register

The DATAOUT register provides write access to the output data value for the GPIO port. The data bits written are masked with the DATAMASK register, so only the bits enabled by the DATAMASK are updated.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Output value	0x0000 0000

4.6.4 DATAOUTRAW Register

The DATAOUTRAW register provides write access to the output data value for the GPIO port. The data bits written are written to the full data output register without using the DATAMASK.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Output value	-

4.6.5 SETOUT Register

The SETOUT register provides alternate write access to the GPIO output value. For those bits that contain a 1 in the write value to this register the matching bit in the GPIO output value register will be set to 1. For those bits that contain a 0 in the write value to this register the matching bit in the GPIO output value will remain unchanged. Updating of the data out register with this register is NOT dependent on the DATAMASK register.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Set value	-

4.6.6 CLROUT Register

The CLROUT register provides alternate write access to the GPIO output value. For those bits that contain a 1 in the write value to this register the matching bit in the GPIO output value register will be set to 0. For those bits that contain a 0 in the write value to this register the matching bit in the GPIO output value will remain unchanged. Updating of the data out register with this register is NOT dependent on the DATAMASK register.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Clear value	-

4.6.7 TOGOUT Register

The TOGOUT register provides alternate write access to the GPIO output value. For those bits that contain a 1 in the write value to this register the matching bit in the GPIO output value register will be inverted from its current value. For those bits that contain a 0 in the write value to this register the matching bit in the GPIO output value will remain unchanged. Updating of the data out register with this register is NOT dependent on the DATAMASK register.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Toggle value	-

4.6.8 DATAMASK Register

The DATAMASK register provides a mask register that is used when the DATAIN and DATAOUT registers are accessed. The DATAMASK register defines which bits are allowed to be read or updated by those accesses. When a bit is 1 in the DATAMASK resister the corresponding bit in the DATAIN or DATAOUT register can be accessed. When a bit is 0 in the DATAMASK resister the corresponding bit in the DATAIN or DATAOUT register cannot be accessed.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO Mask value	0xffff ffff

4.6.9 DIR Register

The DIR register configures individual bits as input or output signals.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO direction value	0x0000 0000
		0	Bit is an input	
		1	Bit is an output	

4.6.10 PULSE Register

The PULSE register provides control of the output data register. For those bits that contain a 1 in this register, PULSE mode will be enabled. When in pulse mode, a given bit will automatically be reset back to its PUSLEBASE value following an update to its value.

As an example, if PULSE mode is enabled for a bit and the corresponding PULSEBASE value is 0, then setting a 1 to the corresponding data bit will cause the GPIO bit to go high for

exactly one system clock cycle, and then reset back to 0 on the next system clock cycle. Setting the corresponding data bit to 0 results in no change to the GPIO bit.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO pulse mode value	0x0000 0000
		0	Pulse mode disabled	
		1	Pulse mode enabled	

4.6.11 PULSEBASE Register

The PULSEBASE register is used with the PULSE data register to define the default state of a bit that is in pulse mode.

Bit	Symbol	Description	Reset value
31:0	VALUE	GPIO PulseBase value	0x0000 0000

4.6.12 DELAY1 and DELAY2 Registers

The DELAY1 and DELAY2 registers provides control of the timing of the output signals. Setting a bit in the DELAY1 registers delays the GPIO output signal by 1 system clock cycle from its normal timing. Setting a bit in the DELAY2 registers delays the output signal by 2 system clock cycles from its normal timing. Setting a bit in the DELAY1 and DELAY2 registers delays the output signal by 3 system clock cycles from its normal timing. This allows signals to be skewed a few cycles relative to each other.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO delay enable	0x0000 0000
		0	Delay disabled	
		1	Delay enabled	

4.6.13 IRQ_SEN, IRQ_EDGE, and IRQ_EVT Registers

The IRQ_SEN, IRQ_EDGE, and IRQ_EVT registers configure which types of transitions on the GPIO bits will generate status values and potentially interrupts. Status detection can be configured to give the current value of the GPIO bits, or to detect edge transitions of the GPIO bits.

When configured to detect edge transition, the output status is active for a single cycle when the edge is detected. In addition, an edge detect status register is set to record the edge

detection. This edge detect status register can be read (as EDGE_STATUS) to get it values, which also clears it. The persistent edge detect status register is not used for interrupt generation or GPIO output status.

The IRQ_SEN register configures level verses edge status detection.

The IRQ_EDGE register configures dual edge detection when edge detection is enabled.

The IRQ_EVT register configures the level value or edge type that triggers the status.

Note that even if the IRQ_ENB bits are disabled, the status detection logic is still active and can detect edges. The status logic can be used independent of the interrupt enable to control cascade sources of the TIM peripheral.

Table 18 IRQ_SEN Register

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO sense value	0x0000 0000
		0	Status is generated from signal transition	
		1	Status is generated from signal level	

Table 19 IRQ_EDGE Register

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO both edge value	0x0000 0000
		0	Status is generated from signal transition as configured by IRQ_EVT bits	
		1	Status is generated for either L->H or H->L transition of signal	

Table 20 IRQ_EVT Register

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO level/edge value	0x0000 0000
		0	Status is generated when signal is LOW for IRQ_SEN mode level, or a H->L transition in IRQ_SEN transition mode edge.	
		1	Status is generated when signal is HIGH for IRQ_SEN mode level, or a L->H transition in IRQ_SEN transition mode edge.	

4.6.14 IRQ_ENB Register

The IRQ_ENB register configures IRQ enable for individual GPIO input bits based on the status detection logic (IRQ_SEN, IRQ_EDGE, and IRQ_EVT registers).

Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO enable value	0x0000 0000
		0	Interrupt is disabled	
		1	Interrupt is enabled	

4.6.15 IRQ_RAW Register

The IRQ_RAW register provides a real time read access to the RAW event status of each GPIO bit. This register is not latched and may not indicate edge sensitive events.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO raw IRQ status value	0x0000 0000
		0	Interrupt is not active	
		1	Interrupt is active	

4.6.16 IRQ_END Register

The IRQ_END status register provides read access to the Enabled interrupt status of each GPIO bit. This is the logical AND of the IRQ_RAW status bits and the IRQ_ENB bits.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO enabled IRQ status value	0x0000 0000
		0	Interrupt is not active	
		1	Interrupt is active	

4.6.17 EDGE_STATUS Register

The EDGE_STATUS register provides read / write access to the edge detect status register. This register is set when a GPIO edge change is detected as configured by the IRQ_SEN, IRQ_EDGE, and IRQ_EVT registers. Individual bits of this register are cleared by writing a 0.

Bit	Symbol	Bit Value	Description	Reset value
31:0	VALUE		GPIO edge status	0x0000 0000
		0	No edge has been detected since the last register update	
		1	At least one edge has been detected since the last register update	

4.6.18 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0040 07e1).

4.7 Timer/Counter Peripheral (Software label = TIMER)

The following diagram shows a general block diagram of the Timer/Counter Interface.

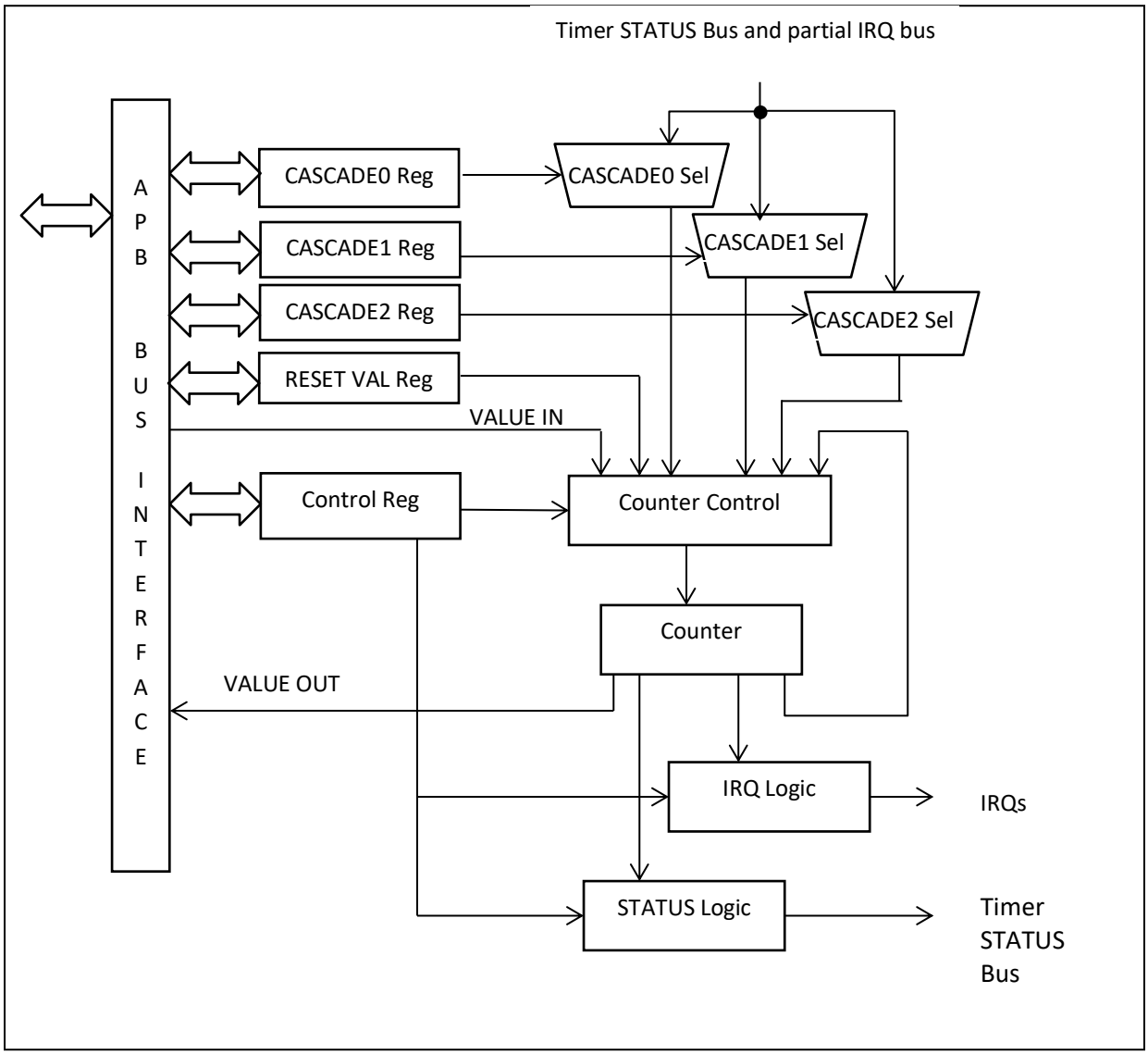


Figure 6 – Timer/Counter Block Diagram

The Timer/Counter peripheral contains 24 counter/timers. These can be configured as timers or event counters. They can be free running or triggered by system events such as other timer interrupts, GPIO interrupts, memory correction unit interrupts (VA10820/VA10830 only) and the Cortex-M0 TXE interrupt signal. Each timer can be connected to a port pin to create pulse width modulated (PWM) outputs, complex pulse trains or single edge events. See the IO

Function Selection table in the IOCONFIG section for which port pins are associated with the 24 timer channels.

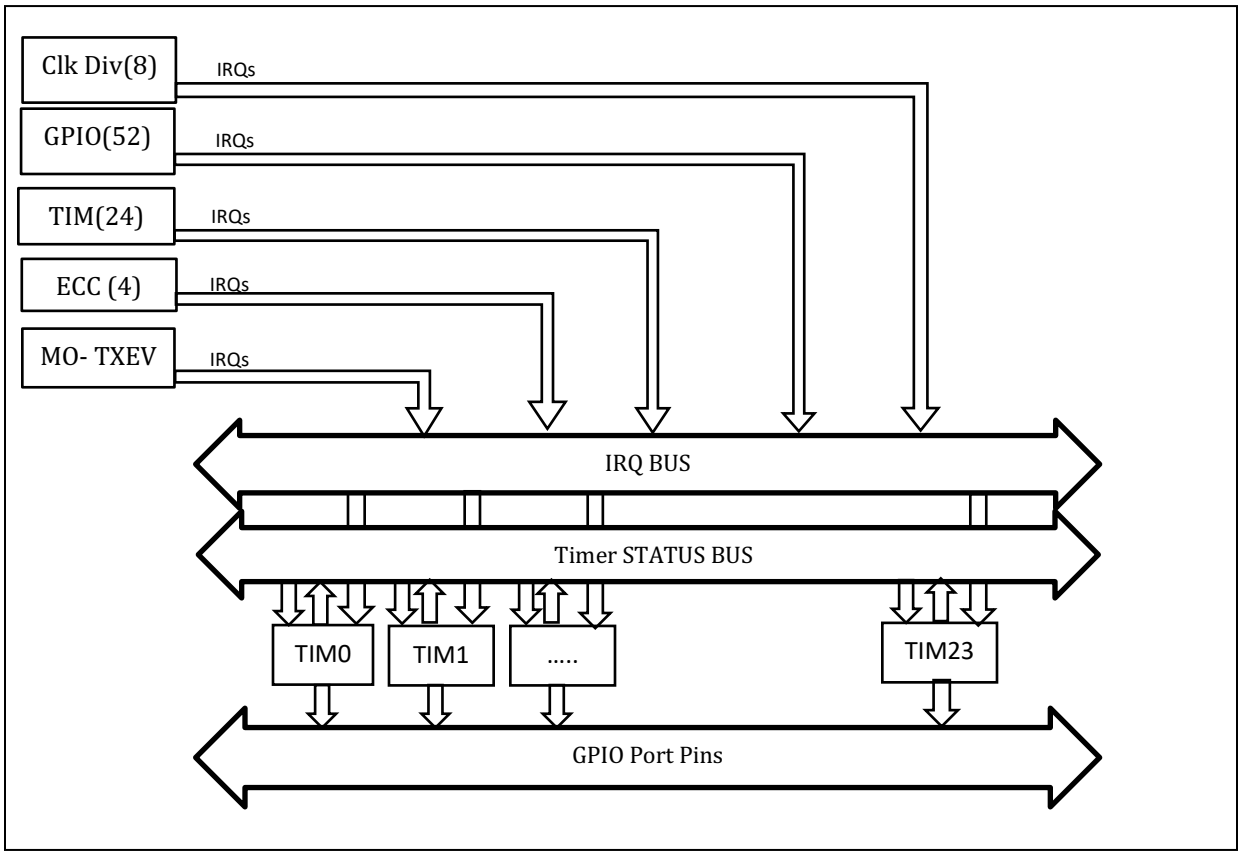


Figure 7 – Timer peripheral chip interconnect

The IO Configuration peripheral can be configured to allow the status of a given counter/timer to be output on the GPIO pins.

Table 21 – TIM Group Location

Name	Address offset	Description
TIM0	0x00000-0x00FFC	TIM 0
TIM1	0x01000-0x01FFC	TIM 1
TIM2	0x02000-0x02FFC	TIM 2
TIM3	0x03000-0x03FFC	TIM 3
TIM4	0x04000-0x04FFC	TIM 4
TIM5	0x05000-0x05FFC	TIM 5
TIM6	0x06000-0x06FFC	TIM 6
TIM7	0x07000-0x07FFC	TIM 7
TIM8	0x08000-0x08FFC	TIM 8

TIM9	0x09000-0x09FFC	TIM 9
TIM10	0x0A000-0x0AFFC	TIM 10
TIM11	0x0B000-0x0BFFC	TIM 11
TIM12	0x0C000-0x0CFFC	TIM 12
TIM13	0x0D000-0x0DFFC	TIM 13
TIM14	0x0E000-0x0EFFC	TIM 14
TIM15	0x0F000-0x0FFFC	TIM 15
TIM16	0x10000-0x10FFC	TIM 16
TIM17	0x11000-0x11FFC	TIM 17
TIM18	0x12000-0x12FFC	TIM 18
TIM19	0x13000-0x13FFC	TIM 19
TIM20	0x14000-0x14FFC	TIM 20
TIM21	0x15000-0x15FFC	TIM 21
TIM22	0x16000-0x16FFC	TIM 22
TIM23	0x17000-0x17FFC	TIM 23

Table 22 - TIM Registers

Name	Access	Address offset	Description	Reset value
CTRL	RW	0x000	TIM Control Register	0x0000 0000
RST_VALUE	RW	0x004	TIM Restart Value	0x0000 0000
CNT_VALUE	RW	0x008	TIM Counter Value	0x0000 0000
ENABLE	RW	0x00C	TIM Enable Bit	0x0000 0000
CSD_CTRL	RW	0x010	TIM Cascade Control	0x0000 0000
CASCADE0	RW	0x014	TIM Cascade0	0x0000 0000
CASCADE1	RW	0x018	TIM Cascade1	0x0000 0000
CASCADE2	RW	0x01c	TIM Cascade2	0x0000 0000
PWMA_VALUE	RW	0x020	TIM PWMA Value	0x0000 0000
PWMB_VALUE	RW	0x024	TIM PWMB Value	0x0000 0000

-	-	0x024-0xFC8	Reserved	-
-	-	0xFD0-0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0111 07e1

4.7.1 Timer setup example

The timer block can be used to measure pulse widths on a port pin, to generate pulses on a port pin or to internally generate intervals. The following steps are provided to show a typical sequence for configuring a PWM output.

- Enable the clock source for the GPIO and IOCONFIG in the PERIPHERAL_CLK_ENABLE register the SYSCONFIG peripheral
- Enable the clock source of the Timer in the TIM_CLK_ENABLE of the SYSCONFIG peripheral
- Set pin assignment for port pin in the IO Function select register in the IOCONFIG peripheral.
- Set the period of the PWM in RST_VALUE (TIM restart value)
- Set the duty cycle of the PWM in PWMA_VALUE
- If using interrupts:
  - Set the IRQ_ENB bit in the CTRL register
  - Configure the IRQ Selector Peripheral
  - Set the NVIC priority of the interrupt
  - Enable the NVIC interrupt
- If using timer as other timer cascade input:
  - Set the IRQ_ENB bit in the CTRL register
- Configure the CTRL register to ENABLE the block and to set the STATUS_SEL field to PWMA.

4.7.2 CTRL Register

The CTRL register configures a Counter/Timer. Each counter/timer is 32 bits and counts down from its restart value to zero when enabled. On making the count transition from 1 to 0, the counter is considered DONE; at which time it can stop or automatically restart.

The peripheral must have its clock enabled and the counter enabled too for it to count. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. When the ENABLE bit is set to zero, the counter will no longer count. Disabling the clock from the

System Configuration Peripheral will not disable the clock until the counter is no longer ACTIVE.

The counter has a concept of Enabled and Active. Enabled means it can count as determined by the cascade settings. Active means the counter has *started counting*, but not finished. *Started counting*, means the count value has changed its value at least once since it was enabled to count. The counter ACTIVE bit goes to 0 from several different cases:

- The counter was Disabled (ENABLE set to 0).
- The counter reached 0, and AUTO_DISABLE was set.
- The counter reached 0, and AUTO_DEACTIVATE was set.
- The counter was disabled (ENABLE set to 0) from a Cascasde2 condition.

Bit	Symbol	Value	Description	Reset value
0	ENABLE		Enables the counter (1 is enabled)	0
1	ACTIVE		Read-Only bit that indicates if the counter is active. A counter becomes active on the first clock after it is enabled and its count value changes. (See previous paragraph for more details on conditions that set this bit to zero.)	0
2	AUTO_DISABLE		When set to 1, the counter is automatically disabled (changing the ENABLE and ACTIVE bits to 0) when the count reaches 0.	0
3	AUTO_DEACTIVATE		When set to 1, the counter is automatically deactivated (changing the ACTIVE bit to 0) when the count reaches 0. This only applies when AUTO_DISABLE is 0 (continuous counting mode). When AUTO_DISABLE is 1, AUTO_DEACTIVATE is implied.	0
4	IRQ_ENB[1]		Enable interrupt on count reaching 0. The generated interrupt is active for 1 cycle.	0
7:5	STATUS_SEL		Timer Status Select. This field determines when the timer output is active. Timer status (the output signal of the timer) is a signal that can be used to cascade to other timers and can output on a GPIO pin. This output can be configured in the following modes:	0

		0	A one cycle pulse when the counter transitions to 0.	
		1	Output the ACTIVE status bit.	
		2	Toggle between 1/0 every time the counter reaches 0. Basically, a divide by 2 output clock of the timer.	
		3	PWMA output value. 1 when Counter Value >= PWMA Value 0 when Counter Value < PWMA Value	
		4	PWMB output value. 1 when Counter Value < PWMA Value and >= PWMB 0 when Counter Value >= PWMA Value or < PWMB	
		5	Output the ENABLED status bit.	
		6	PWMA active mode. 1 when CounterValue <= PWMA Value and >0 0 otherwise	
		7	Reserved	
8	STATUS_INV		Invert the value selected by STATUS_SEL	0
9	REQ_STOP		When 1, the counter is requested to stop (changing the ENABLE bit to 0, and changing the ACTIVE bit to 0) on the next normal count cycle (even if the counter is not yet at zero).	
31:10	-		Reserved	0

Note:

1. Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

4.7.3 RST_VALUE Register

The RST_VALUE register configures the restart value for the counter. This value is used to reload an enabled counter when its value has reached 0.

Note if this value is set to 0, the counter becomes a divide by 1 counter; which is basically an always done counter. This may be useful for special cascading modes.

Bit	Symbol	Description	Reset value
31:0	VALUE	Restart Value for the Counter	0x0000 0000

4.7.4 CNT_VALUE Register

The CNT_VALUE register provides access to the current count value in the timer. This value can be read or written. If the counter is enabled, then writing 0 value to the counter will trigger any events that would normally happen if the counter had reached 0 by counting.

Bit	Symbol	Description	Reset value
31:0	VALUE	Current Value for the Counter	0x0000 0000

4.7.5 ENABLE Register

The ENABLE register provides individual bit access to the ENABLE bit in the Control Register. This is an alternate access to the same control bit.

Bit	Symbol	Description	Reset value
0	ENABLE	Enables the counter (1 is enabled)	0
31:1		Reserved	

4.7.6 CSD_CTRL Register

The CSD_CTRL register configures count enables on the Counter/Timer. The counter can count on every clock cycle, or only count when a specified Cascade condition is true. Up to 2 Cascade conditions can be selected from other TIM counters, or GPIO signals. When 2 Cascade conditions are selected, they can be configured as logical AND or OR of the conditions.

Bit	Symbol	Value	Description	Reset value
0	CSDEN0		Cascade control 0. When this bit is 1, the counter only counts when the selected Cascade signal is active.	0
1	CSDINV0		Invert Cascade 0. When this bit is 0 the Cascade signal 0 is active high, when this bit is 1 it is active low.	0
2	CSDEN1		Cascade control 1. When this bit is 1, the counter only counts when the selected Cascade signal is active.	0

3	CSDINV1		Invert Cascade 1. When this bit is 0 the Cascade signal 1 is active high, when this bit is 1 it is active low.	0
4	DCASOP		Dual Cascade Operation. Specifies the required operation when both Cascade 0 and Cascade 1 are active.	0
		0	Logical AND of both cascade signals	
		1	Logical OR of both cascade signals	
5	Reserved			0
6	CSDTRG0		Cascade control 0 (if enabled) is used in Trigger mode. In trigger mode, the counter will only start counting (become ACTIVE) with the selected Cascade signal is active. Once the counter is counting (is ACTIVE) the Cascade control is ignored.	0
7	CSDTRG1		Cascade control 1 (if enabled) is used in Trigger mode. In trigger mode the counter will only start counting (become ACTIVE) with the selected Cascade signal is active. Once the counter is counting (is ACTIVE) the Cascade control is ignored.	0
8	CSDEN2		Cascade control 2. When this bit is 1, the counter will stop (ENABLE will go to 0) instead of decrementing at the next update time, if the selected Cascade signal is active. This mode is similar to the REQ_STOP control bit, but it is signaled by a Cascade source.	0
9	CSDINV2		Invert Cascade 2. When this bit is 0 the Cascade signal 2 is active high, when this bit is 1 it is active low.	0
10	CSDTRG2		Cascade control 2 (if enabled) is used in Trigger mode. In trigger mode, the counter is automatically disabled (ENABLE and ACTIVE bits will go to 0) if the corresponding Cascade2 level-sensitive input source is active when the count reaches 0. If the counter is not zero, the Cascade control is ignored. Note: Do not use edge sensitive cascade sources in this mode.	0

4.7.7 CASCADE0, CASCADE1, and CASCADE2 Register

The CASCADE0, CASCADE1, and CASCADE2 registers configure the cascade enable source for the Counter/Timer. The value in the register selects the cascade source. CASCADE0 and CASCADE1 are used to control the counting and activation of the counter. CASCADE2 is used to request stopping of the counter (ENABLE goes to 0).

The source of the cascade input is the interrupt output from GPIO, other Timer channels, the memory check unit, the transmit event (TXEV) signal from the M0 core or the I/O configuration clock dividers. The interrupt must be enabled on the sourcing peripheral block.

Table 23 – CASCADE0, CASCADE1, and CASCADE2 Register

Bit	Symbol	Description	Reset value
6:0	CASSEL	Cascade source selection value	0x00
31:7	-	Reserved	0x000 0000

Table 24 - Cascade Selection Codes

Value	Symbol	Source	Description
0	TIM_CAS_SRC_PORTA_0	PORTA[0] RAW IRQ	RAW Status from PORTA
1-30	TIM_CAS_SRC_PORTA_1 TIM_CAS_SRC_PORTA_30	PORTA[1-30] RAW IRQ	
31	TIM_CAS_SRC_PORTA_31	PORTA[31] RAW IRQ	
32	TIM_CAS_SRC_PORTB_0	PORTB[0] RAW IRQ	RAW Status from PORTB
33-54	TIM_CAS_SRC_PORTB_1 TIM_CAS_SRC_PORTB_22	PORTB[1-22] RAW IRQ	
55	TIM_CAS_SRC_PORTB_23	PORTB[23] RAW IRQ	
56-64		-	Reserved
64	TIM_CAS_SRC_TIM_0	TIM 0 TIMERDONE	Timer Done signals from TIM[1]
65-86	TIM_CAS_SRC_TIM_1 TIM_CAS_SRC_TIM_22	TIM 1-22 TIMERDONE	
87	TIM_CAS_SRC_TIM_23	TIM 23 TIMERDONE	
88-95		-	Reserved
96	TIM_CAS_SRC_RAM_SBE	RAM_SBE	Single-Bit Error from Data RAM

97	TIM_CAS_SRC_RAM_MBE	RAM_MBE	Multi-Bit Error from Data RAM
98	TIM_CAS_SRC_ROM_SBE	ROM_SBE	Single-Bit Error from Code RAM (ROM)
99	TIM_CAS_SRC_ROM_MBE	ROM_MBE	Multi-Bit Error from Code RAM (ROM)
100	TIM_CAS_SRC_TXEV	TXEV	Processor TXEV signal
101-119		-	Reserved
120	TIM_CAS_SRC_IOCONFIG_CLKDIV_0	IOCONFIG_CLKDIV0	IO Configuration Clock Divider 0
121-126	TIM_CAS_SRC_IOCONFIG_CLKDIV_1 TIM_CAS_SRC_IOCONFIG_CLKDIV_6	IOCONFIG_CLKDIV1-6	IO Configuration Clock Divider 1-6
127	TIM_CAS_SRC_IOCONFIG_CLKDIV_7	IOCONFIG_CLKDIV7	IO Configuration Clock Divider 7

Notes:

1. TIMERDONE is a one cycle pulse generated by each timer when it transitions to a count value of zero.

4.7.8 PWMA_VALUE Register

The PWMA_VALUE register provides control bits for the PWMA and PWMB output modes. These bits provide a value that can be compared to the current count value to generate the PWM output. This can be used to create a variable duty cycle divider signal.

Bit	Symbol	Description	Reset value
31:0	VALUE	PWM compare value	0x0000 0000

The below figure provides an example PWM waveform with PWMA set to 0x0D00.

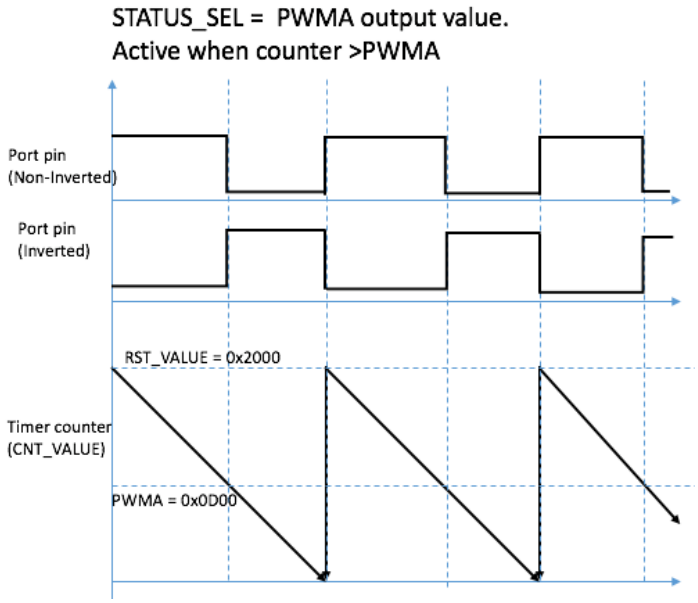


Figure 8 - Example waveform with PWMA being selected

4.7.9 PWMB_VALUE Register

The PWMB_VALUE register provides control bits for the PWMB output mode. These bits provide a value that can be compared to the current count value to generate the PWM output. This can be used to create a variable duty cycle divider signal.

Bit	Symbol	Description	Reset value
31:0	VALUE	PWM compare value	0x0000 0000

Using the Timer in PWMB mode allows for other channels to have edges occur slightly offset which can be useful in motor control applications. PWMB also allows PWM waveforms to be center-aligned which can greatly help reduce harmonics of high frequency power switching devices. The below figure shows an example waveform on a port pin with PWMB selected.

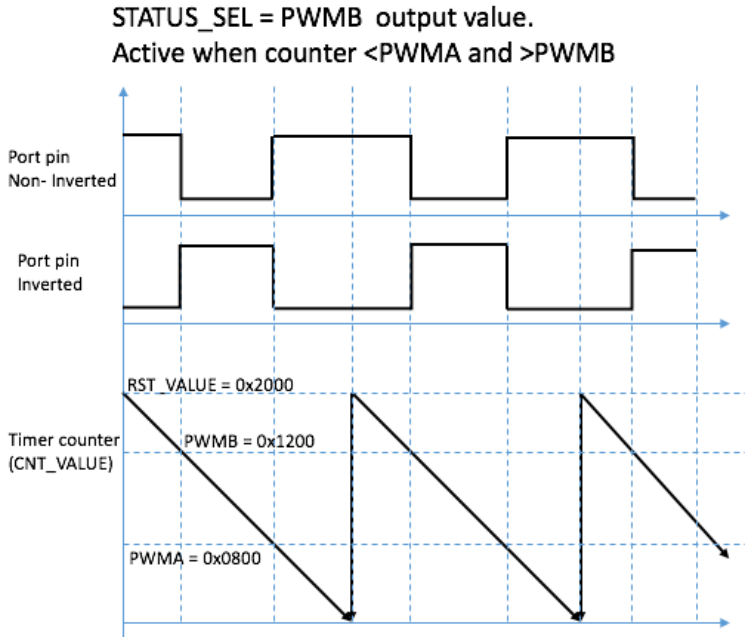


Figure 9 - Example of PWMB mode of operation.

4.7.10 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0111 07e1).

4.8 UART Peripheral (Software label = UARTA & UARTB)

The following diagram shows a general block diagram of the UART Interface.

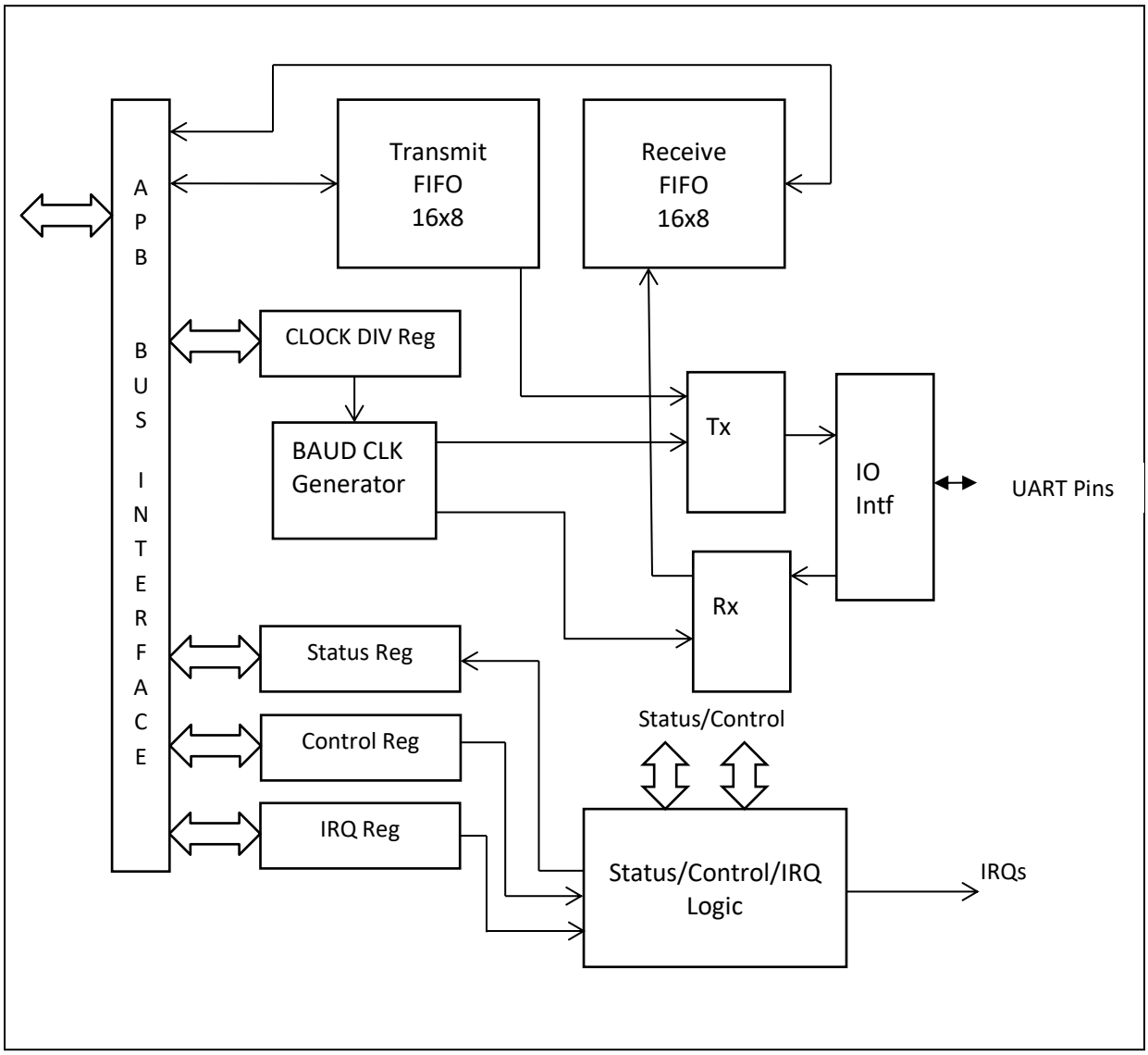


Figure 10 - UART Block Diagram

The UART peripheral contains 2 UART banks. Each UART bank provides a general UART with independent Transmit and Receive sections, each with a 16 byte FIFO.

To have the UART connected to input/output pins, the IO Configuration peripheral must be properly configured to connect the UART to some pins. It is recommended that the pins be configured in the IO Configuration peripheral before enabling a given UART.

Table 25 – UART Bank Location

Name	Address offset	Description
UART0	0x0000-0x0FFC	UART 0 - UARTA
UART1	0x1000-0x1FFC	UART 1 - UARTB
	0x2000-0xFFFC	Reserved

Table 26 - UART Registers

Name	Access	Address offset	Description	Reset value
DATA	RW	0x000	Data register	0x0000 0000
ENABLE	RW	0x004	Tx/Rx enable register	0x0000 0000
CTRL	RW	0x008	Control Register	0x0000 0000
CLKSCALE	RW	0x00C	Clock Divider register	0x0000 06c8
RXSTATUS	R	0x010	Rx Status Register	0x0000 0000
TXSTATUS	R	0x014	Tx Status Register	0x0000 0000
FIFO_CLR	W	0x018	Clear Status Register	-
TXBREAK	W	0x01C	Break Transmit Register	-
ADDR9	RW	0x020	9-Bit Self Address Register	0x0000 0000
ADDR9MASK	RW	0x024	9-Bit Address Mask Register	0x0000 0000
IRQ_ENB	RW	0x028	IRQ Enable Register	0x0000 0000
IRQ_RAW	R	0x02C	IRQ Raw Status Register	0x0000 0000
IRQ_END	R	0x030	IRQ Enabled Status Register	0x0000 0000
IRQ_CLR	W	0x034	IRQ Clear Register	0
RXFIFOIRQTRG	RW	0x038	Rx Fifo Trigger level register	0x0000 0008
TXFIFOIRQTRG	RW	0x03C	Tx Fifo Trigger level register	0x0000 0008

RXFIFORTSTRG	RW	0x040	Rx Fifo RTS Trigger level register	0x0000 000e
STATE	R	0x044	Rx/Tx State Machine data	-
-	-	0x038-0xFF8	Reserved	-
PERID	RO	0xFFC	Peripheral ID Register	0x0112 07e1

The following UART signals can be connected to GPIO pins using the IO Configuration peripheral. *X* being A or B (port).

Table 27 – UART Signals

Name	Description
UARTX_TX	UART Transmit for port *X*
UARTX_RX	UART Receive for port *X*
UARTX_RTSn	UART RTSn for port *X*
UARTX_CTSn	UART CTSn for port *X*

4.8.1 UART Transactions

The UART controller is used to transmit and receive UART transaction.

A typical sequence for configuring the UART controller is summarized below:

- Configure the Baud rate with the CLKSCALE register
- Configure the CTRL register as need (such as parity, word size, RTSn/CTSn, etc.)
- Configure the IO Configuration Peripheral to enable UART pins as needed
- If using interrupts:
  - Load the IRQ_ENB register
  - Configure the IRQ Selector Peripheral
  - Set the NVIC priority of the interrupt
  - Enable the NVIC interrupt
- Set the RXENABLE or TXENABLE bit in ENABLE register to enable the interface

4.8.2 DATA Register

The DATA provides access to the UART input or output FIFOs. Any data written to the register is loaded into the transmit FIFO. Reads from the register return an item from the receive FIFO.

Bit	Symbol	Description	Reset value
7:0	VALUE	UART data value	0x00
14:8		Reserved, Read as 0	
15	DPARITY	For transmit in manual parity mode, this bit is used to set the out-going parity bit. For receive in manual parity mode, this bit contains the XOR of the received parity with the PAREVEN value (of the control register).	0
31:16		Reserved, Read as 0	

4.8.3 ENABLE Register

The ENABLE register provide enable/disable control of the UART. When the Receiver is disabled it will ignore all data on the Rx input pin. When the Transmitter is disabled it will not transmit pending data in the Tx FIFO. When the Receiver or Transmitter enable changes from Enabled to Disabled, any current byte receive/transmit in progress will be completed, but new bytes will not be started after that.

The peripheral must have its clock enabled and the receiver enabled to send UART transactions; and have its clock enabled and the transmitter enabled to transmit UART transactions. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. When either the RXENABLE bit or the TXENABLED bit is changed to zero, the corresponding interface will become disabled when the interface is idle (completes any pending transaction). Similarly, disabling the clock from the System Configuration Peripheral will not disable the clock until any pending transactions have completed.

Bit	Symbol	Description	Reset value
0	RXENABLE	Receiver Enable Bit	0
1	TXENABLE	Transmit Enable Bit	0
31:2	-	Reserved, Read as 0	

4.8.4 CTRL Register

The CTRL register provides configuration data for the UART.

Bit	Symbol	Description	Reset value
0	PAREN	Parity Enable	0
1	PAREVEN	When parity is enabled: 1 selects even parity, 0 selects odd parity	0
2	PARMAN	Enables manual parity mode. When enabled the transmitter uses the DPARITY bit from the data word as the parity bit to transmit. When enabled and not in 9-bit mode, the receiver computes the XOR of the received parity and the PAREVEN value (of the control register); this value is stored in DPARITY bit of the data word, and is used as the parity error interrupt. When enabled and in 9-bit mode, the receiver uses received parity values to store in DPARITY bit of the data word, and as the parity error interrupt. This allows interrupt generation on address match.	0
3	STOPBITS	Select the number of stop bits: 0 is 1 stop bit, 1 is 2 stop bits.	0
5:4	WORDSIZE	Selects the word size: 0x0 – 5 bits 0x1 – 6 bits 0x2 – 7 bits 0x3 – 8 bits	0x3
6	LOOPBACK	Loopback mode. When 1, then the Receiver input is connected to the Transmitter output and CTSn input is connected to RTSn output.	0
7	LOOPBACKBLK	Loopback block mode. When 1 and LOOPBACK is 1, then the Transmitter output (to the IO block) is held high during loopback mode. In addition the RTSn signal is held at the DEFRTS value.	0
8	AUTOCTS	Enable auto CTS Mode. When enabled the Transmitter is paused if CTSn is high, and a transmit would normally start (data is ready in FIFO).	0
9	DEFRTS	This specifies the value for the RTSn signal when AUTORTS is not enabled or when the Receiver is not enabled.	1
10	AUTORTS	Enable auto RTS Mode. When enabled the RTSn signal is auto generated from the FIFO level and the RXFIFORTSTRG register. When RX FIFO Count >= RXFIFORTSTRG level, then RTSn will be High,	0

		otherwise it will be Low. When not enabled the DEFRTS value is output on RTSn.	
11	BAUD8	When 0, a standard 16x baud clock is used. When 1 an 8x baud clock is used.	0
31:12	-	Reserved, Read as 0	

4.8.5 CLKSCALE Register

The CLKSCALE register configures the UART clock generator. This value contains a fractional divide number that is used to generate the 16X baud clock. The CLKSCALE register should only be loaded with a new value when the UART is inactive (Not enabled, and no transactions in progress). Any write to this register will reset the clock rate divider.

Bit	Symbol	Description	Reset value
5:0	FRAC	Fractional divide value. This is the fraction divide value in 1/64 units.	0x00
23:6	INT	Integer divide value. This is the integer divide value	0x0 0000
31:24	-	Reserved, Read as 0	

The register value parts can be calculated as follows:

- X = ClockFrequency/(BaudRate*BaudMode)
- BaudMode = 16 or 8 based on BAUD8 control bit
- INT = integer_part_of(X)
- FRAC = integer_part_of(64 * (X - INT) + 0.5)
- REGISTER = INT * 64 + FRAC

Example: 115,200 Baud Rate with 50MHz clock

- X = 50,000,000 / (115,200 * 16) => 27.126736111...
- INT = 27
- FRAC = integer_part_of (64 * .126736111... + 0.5) => integer_part_of (8.611...) => 8
- REGISTER => 27 * 64 + 8 => 1736 => 0x6C8

Table 28 - UART Baud Rate Values

Baud Rate	16X Baud Clock (kHz)	Divide Value (for 50MHz system clock)	Register Value
2,000,000	32000.0	1 36/64	0x000064
1,000,000	16000.0	3 8/64	0x0000c8
115,200	1843.2	27 8/64	0x0006c8
57,600	921.6	54 16/64	0x000d90
38,400	614.4	81 24/64	0x001458
19,200	307.2	162 49/64	0x0028b1
9,600	153.6	325 33/64	0x005161
4,800	76.8	651 3/64	0x00a2c3
2,400	38.4	1302 5/64	0x014585
1,200	19.2	2604 11/64	0x028b0b
300	4.8	10416 43/64	0x0a2c2b

4.8.6 RXSTATUS Register

The RXSTATUS register provides read-only access to UART receiver status.

Bits 7-5 reflect that status of the next data in the data register and can only be non-zero when there is valid data in the DATA register (as indicated by bit 0). Thus a parity error means that the next data word to be read from the Data register had a parity error. Break means that next data word is a break (and will have 0 for the DATA value). Once data is read from the DATA register the status reflects the next data word.

Bit	Symbol	Description	Reset value
0	RDAVL	Read Data Available (1 indicates read data is ready in the receive FIFO)	0
1	RDNFULL	Read Not Full (1 indicates the read data FIFO is NOT FULL)	0
2	RXBUSY	Receiver Busy (1 indicates the receiver is currently receiving a byte).	0
3	RXTO	Receiver Timeout. Indicates that there is data in the Receive FIFO and there has been no receiver FIFO activity (read or write) for 4 character times.	
4	RXOVR	A 1 indicates the receive FIFO overflowed	0
5	RXFRM	A 1 indicates there has been a receive framing error	0

6	RXPAR	A 1 indicates there has been a receive parity error, or an address match in 9-bit mode	0
7	RXBRK	A 1 indicates there has been a receive break condition	
8	RXBUSYBRK	A 1 indicates that a break has been received, but the receiver is still busy waiting from the break to end.	0
9	RXADDR9	The 9-bit Address match register. A 1 indicates there has been an address match in 9-bit mode.	0
14:10	Reserved	Reserved, read as 0	
15	RXRTSN	Output value of RTSn signal	-
31:16	Reserved	Reserved, read as 0	

4.8.7 TXSTATUS Register

The TXSTATUS register provides read-only access to UART transmitter status.

Bit	Symbol	Description	Reset value
0	WRRDY	Write Ready (1 indicates the transmit FIFO is not full)	0
1	WRBUSY	Write Busy (1 indicates the transmit FIFO is not empty)	0
2	TXBUSY	Transmitter Busy (1 indicates the transmitter is currently sending a byte)	0
3	WRLOST	A 1 indicates the transmitter FIFO overflowed	0
14:4	-	Reserved	
15	TXCTSN	Input value of CTSn signal	
31:16	-	Reserved	0

4.8.8 FIFO_CLR Register

The FIFO_CLR register provides write access for clearing status conditions. The FIFO status conditions are cleared implicitly by resetting the FIFO.

Bit	Symbol	Description	Reset value
0	RXFIFO	Empty out the Receive FIFO	0
1	TXFIFO	Empty out the Transmit FIFO	0
31:2	-	Reserved	0

4.8.9 TXBREAK Register

The TXBREAK register provides a method of transmitting a break. The value written to this register determines the length of the break generated. The break request is loaded into the

transmit FIFO similar to data requests. The break duration is measured in character periods, and will be the BRKCNT value plus about 1 3/4.

Using a BRKCNT value of 0x7f will result in a continuous break. In this mode the transmitter sends a break of one-character length; then it goes idle while leaving the Transmit value at 0 (break level). To exit continuous break mode another break must be sent that is not continuous. After that valid data can be sent again. Note that in continuous break mode, the software is responsible for the duration of the break, since it continues until another break request is loaded in the transmitter. Data should not be sent while in continuous break mode, as it will exit continuous break mode, but the receiver will most likely get a framing error.

Bit	Symbol	Description	Reset value
6:0	BRKCNT	Specifies the break duration. The values 0 to 0x7e are timed breaks. The value 0x7f enters continuous break mode.	0
31:7		Reserved, Read as 0	

4.8.10 ADDR9 Register

The ADDR9 register provides the receive match address for 9-bit mode. 9-bit mode is enabled by bit ENB9BIT being 1 and the control register bits PAREN and PARMAN both being enable. In 9-bit mode the parity bit is treated as the address/data indicator (1 being address, 0 being data).

In 9-bit mode the parity bit of each received word is check to determine if the word is address or data. If the word is an address it is checked against the 9-bit receive match address using the ADDR9MASK (bits that are 1 in the ADDR9MASK register determines which address bits are compared). If the address matches, the AddressMatch register is set; if it does not match, the AddressMatch register is cleared. If a received word is determined to be data, it is ignored if the address match register is currently clear, and it is loaded into the FIFO as normal data if the address match register is currently set.

Note, that the matched address is loaded into the receive FIFO with the DPARITY bit set. This allows the actual received address to be examined when the ADDR9MASK has been used to match multiple addresses. Having the DPARITY bit set acts as a parity error; so the parity interrupt can be used to detect new address matches.

Enabling 9-bit mode does not affect the UART transmitter. To Transmit in 9-bit mode the PAREN and PARMAN bits should be set in the control register. Then a word is transmitted as address or data by using the DPARITY bit in the data register.

Bit	Symbol	Description	Reset value
7:0	DATA	receive match address for 9-bit mode	0
14:0	-	Reserved	0
15	ENB9BIT	Enable 9-bit mode	0
31:16	-	Reserved, Read as 0	

4.8.11 ADDR9MASK Register

The ADDR9MASK register provides the mask register for checking addresses in 9-bit mode. The mask determines which bits of the received address are checked against the ADDR9 value. Bits set to 1, enable checking. The default value after reset is all 1's which requires a match of all bits.

Bit	Symbol	Description	Reset value
7:0	DATA	address mask for 9-bit mode	0xff
31:8	-	Reserved, Read as 0	

4.8.12 IRQ_ENB Enable Register

The IRQ_ENB enable register configures the UART interrupts.

Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

Bit	Symbol	Description	Reset value
0	IRQ_RX	Receiver FIFO interrupt enable. Generates interrupt when the FIFO is at least half full. Half full is defined as FIFO count >= RXFIFOIRQTRG	0
1	IRQ_RX_STATUS	Receiver interrupt enable for status conditions (RXOVR, RXFRM, RXPAR, RXBRK)	0
2	IRQ_RX_TO	Receiver interrupt enable for timeout conditions (RXTO)	0
3	–	Reserved, Read as 0	0
4	IRQ_TX	Transmitter interrupt enable. Generates interrupt when the FIFO is at least half empty. Half empty is defined as FIFO count < TXFIFOIRQTRG	0
5	IRQ_TX_STATUS	Rec Transmitter interrupt enable for status conditions (WRLOST)	

6	IRQ_TX_EMPTY	Transmitter interrupt flush enable. Generates an interrupt when the transmit FIFO is empty and TXBUSY is 0.	0
7	IRQ_TX_CTS	Transmitter interrupt enable when CTSn changes value.	0
31:7	-	Reserved, Read as 0	0

4.8.13 IRQ_RAW Register

The IRQ_RAW register provides read only access to the current value of the Raw Interrupt status. The bits correspond to the bits in IRQ Enable Register.

4.8.14 IRQ_END Register

The IRQ_END register provides read only access to the current value of the Enable Interrupt status. This is the logical and of the IRQ_RAW status and the IRQ_ENB register. The bits correspond to the bits in IRQ Enable Register.

4.8.15 IRQ_CLR Register

The IRQ_CLR register provides write only access for clearing the Interrupt status. Writing ones to the given bits will clear the corresponding Raw interrupt status bits.

Note only the Overflow status bits can be cleared. All other interrupt bits depend on data levels in the FIFOs, or Read Status bits that are cleared by reading the data word.

4.8.16 RXFIFOIRQTRG Register

The RXFIFOIRQTRG register configures the RX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Receive FIFO	0x08
31:5	-	Reserved, Read as 0	

4.8.17 TXFIFOIRQTRG Register

The TXFIFOIRQTRG register configures the TX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Transmit FIFO	0x08
31:5	-	Reserved, Read as 0	

4.8.18 RXFIFORTSTRG Register

The RXFIFORTSTRG register configures the TX FIFO level for AUTORTS mode.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the AUTORTS trigger level for the Receive FIFO	0x0e
31:5	-	Reserved, Read as 0	

4.8.19 STATE Register

The STATE register reports debug information on the Rx and Tx State Machines for test purposes. Only the RxFifo and TxFifo counts would be of general use.

Bit	Symbol	Description	Reset value
7:0	RXSTATE	Rx State Machine coding	-
12:8	RXFIFO	Rx FIFO data count	-
15:13	–	Reserved	-
23:16	TXSTATE	Tx State Machine coding	-
28:24	TXFIFO	Tx FIFO data count	-
31:29	–	Reserved	-

4.8.20 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0112 07e1).

4.9 SPI Peripheral (Software label = SPIA, SPIB & SPIC)

The following diagram shows a general block diagram of the SPI Interface.

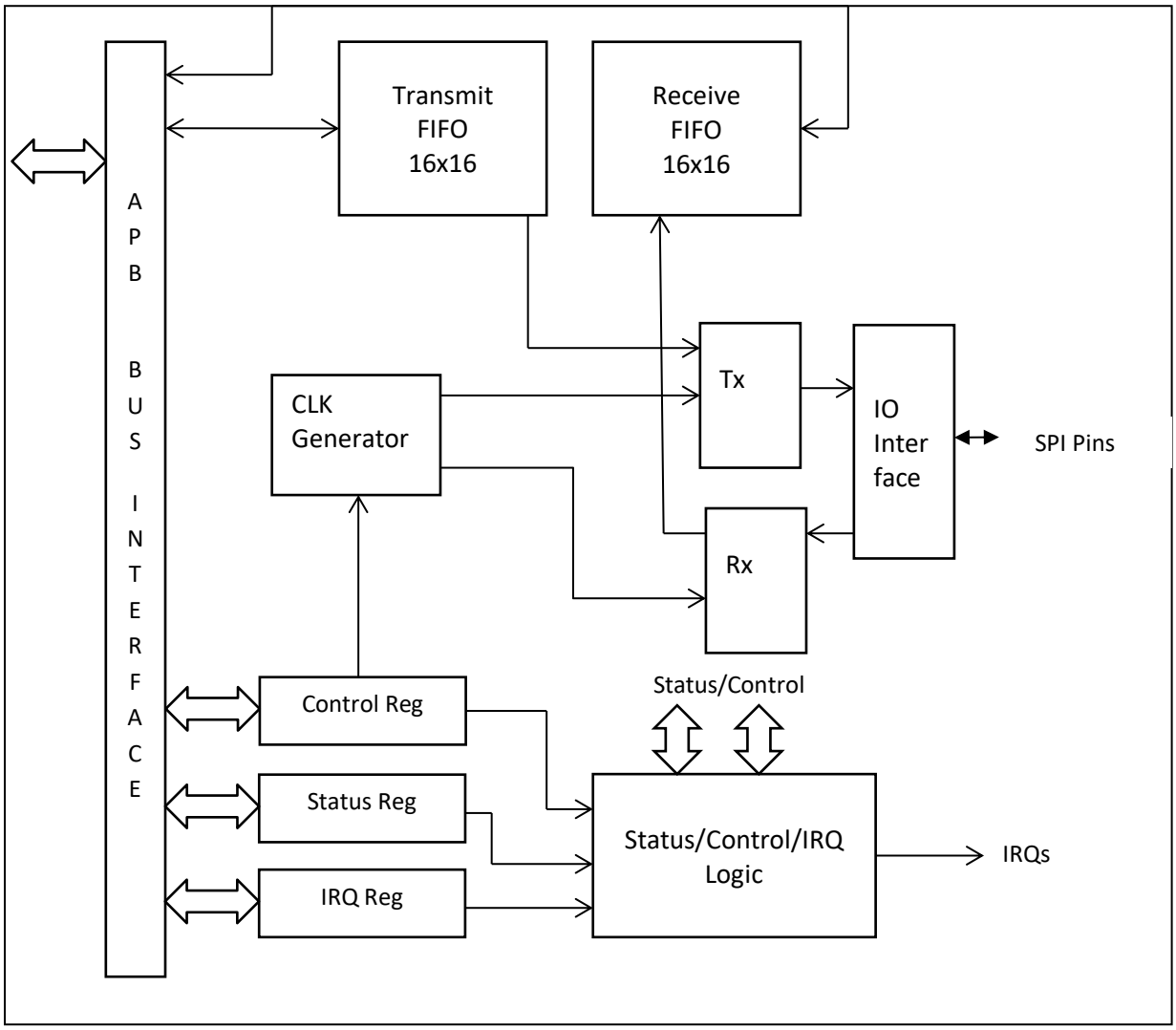


Figure 11 - SPI Block Diagram

The SPI peripheral contains 3 SPI banks. The first 2 SPI banks provide a general SPI that can be configured as a master or slave. The third SPI bank can only be configured as a master, and uses the pins associated with the SPI ROM boot port. The SPI ROM boot process uses its own unique SPI controller; then after the boot process completes the pins are connected to the processor SPI controller.

Each SPI contains a linked Transmit and Receive section, each with a 16 word FIFO.

To have the SPI connected to input/output pins, the IO Configuration peripheral must be properly configured to connect the SPI to some pins. It is recommended that the pins be configured in the IO Configuration peripheral before enabling a given SPI.

SPI2 is always connected to the ROM_SCK (as SPI_SCKC), ROM_CSn (as SPI_SSELCn[0]), ROM_SI (as SPI_MISOC), and ROM_SO (as SPI_MOSIC) pins. Additionally, several more chip selects can be configured with the IO Configuration peripheral.

SPI data is transmitted and received Most-Significant-Bit first.

Table 29 – SPI Bank Location

Name	Address offset	Description
SPI0	0x0000-0x0FFC	SPI 0 – SPIA
SPI1	0x1000-0x1FFC	SPI 1 – SPIB
SPI2	0x2000-0x2FFC	SPI 2 – SPIC
	0x3000-0xFFFC	Reserved

Table 30 - SPI Registers

Name	Access	Address offset	Description	Reset value
CTRL0	RW	0x000	Control register 0	0x0000 0000
CTRL1	RW	0x004	Control register 1	0x0000 0000
DATA	RW	0x008	Data register	0x0000 0000
STATUS	RW	0x00C	Status register	0x0000 0083
CLKPRESCALE	RW	0x010	Clock Prescale register	0x0000 0004
IRQ_ENB	RW	0X014	Interrupt Enable	0x0000 0000
IRQ_RAW	R	0x018	Raw Interrupt Status	0x0000 0008
IRQ_END	R	0x01C	Enabled Interrupt Status	0x0000 0000
IRQ_CLR	W	0x020	Clear Interrupt Status	0x0000 0000
RXFIFOIRQTRG	RW	0x024	Rx FIFO Trigger level register	0x0000 0008

TXFIFOIRQTRG	RW	0x028	Tx FIFO Trigger level register	0x0000 0008
FIFO_CLR	W	0x02C	Clear FIFO Register	-
STATE	R	0x030	Rx/Tx State Machine data	-
-	-	0x030-0xFF8	Reserved	
PERID	RO	0xFFC	Peripheral ID Register	0x0113 07e1

The following SPI signals can be connected to GPIO pins using the IO Configuration peripheral. *X* being A, B or C (port).

Table 31 – SPI Signals

Name	Description
SPI_SCK*X*	SPI Clock for port *X*
SPI_MISO*X*	SPI Master Input Slave Output for port *X*
SPI_MOSI*X*	SPI Master Output Slave Input for port *X*
SPI_SSELXn[0]	SPI Chip Select 0 for port *X* (active low)
SPI_SSELXn[1]	SPI Chip Select 1 for port *X* (active low)
SPI_SSELXn[2]	SPI Chip Select 2 for port *X* (active low)
SPI_SSELXn[3]	SPI Chip Select 3 for port *X* (active low)
SPI_SSELXn[4]	SPI Chip Select 4 for port *X* (active low)
SPI_SSELXn[5]	SPI Chip Select 5 for port *X* (active low)
SPI_SSELXn[6]	SPI Chip Select 6 for port *X* (active low)
SPI_SSELXn[7]	SPI Chip Select 7 for port *X* (active low)

4.9.1 SPI Transactions

The SPI controller is used to control SPI transaction. The interface can be configured as a Master that can initiate transaction, or as a slave that responds to transactions.

4.9.1.1 Configuration

A typical sequence for configuring the SPI controller is summarized below:

- Set the MS bit in CTRL1 as desired for Master/Slave (don't ENABLE interface).
- Configure the IO Configuration Peripheral to enable SPI pins as needed.
- Configure CTRL0, CTRL1, and CLKPRESCALE as desired for transaction type (don't ENABLE interface)
- If using interrupts:
    - Load the IRQ_ENB register

- Configure the IRQ Selector Peripheral
- Set the NVIC priority of the interrupt
- Enable the NVIC interrupt

- Set the ENABLE bit in the CTRL1 register to enable the interface

4.9.1.2 BLOCKMODE Master Transaction

A typical sequence for an SPI Master transaction in BLOCKMODE is summarized below:

- Set the MTXPAUSE bit in CTRL1 (in case we are interrupt while loading data)
- Set the proper CS enable in CTRL1
- Load the desired data into the DATA (FIFO) register
- Clear the MTXPAUSE bit in CTRL1
- Read/Poll the STATUS register until transaction is completed
- Read the returned SPI data from the DATA (FIFO) register
  - If data is to be ignored, the FIFO could just be cleared with the FIFO_CLR register

4.9.2 CTRL0 Register

The CTRL0 register configures the SPI.

Bit	Symbol	Description	Reset value
3:0	SIZE	Size of words (0x03 => 4 bits, 0x0f=> 16 bits)	0x0
5:4	-	Reserved, Must be 0x00	0x0
6	SPO[1]	SPI Clock Polarity	0
7	SPH[1]	SPI Clock Phase	0
15:8	SCRDV[2]	Serial Clock Rate Divider +1 Used with the CLKPRESCALE register to determine the SPI clock rate in master mode.	0x00
31:16	-	Reserved, Read as 0	

Notes:

1. SPO and SPH determine the SPI Clock Polarity and Phase. When SPO is 0 the off state of the clock is 0; when SPO is 1 the off state of the clock is 1. When SPH is 0 data changes on the falling edge of the clock and is captured on the rising edge of the clock; when SPH is 1 data changes on the rising edge of the clock and is captured on the falling edge of the clock.
2. See the CLKPRESCALE register for a description of the SCRDV use.

The following diagrams show the different SPI modes of SPO and SPH, for an 8 bit SIZE.

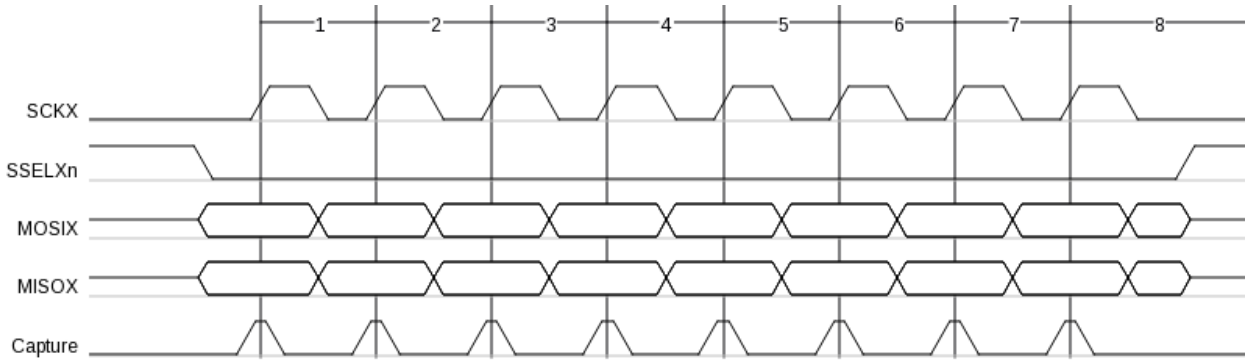


Figure 12 - SPO=0 and SPH=0

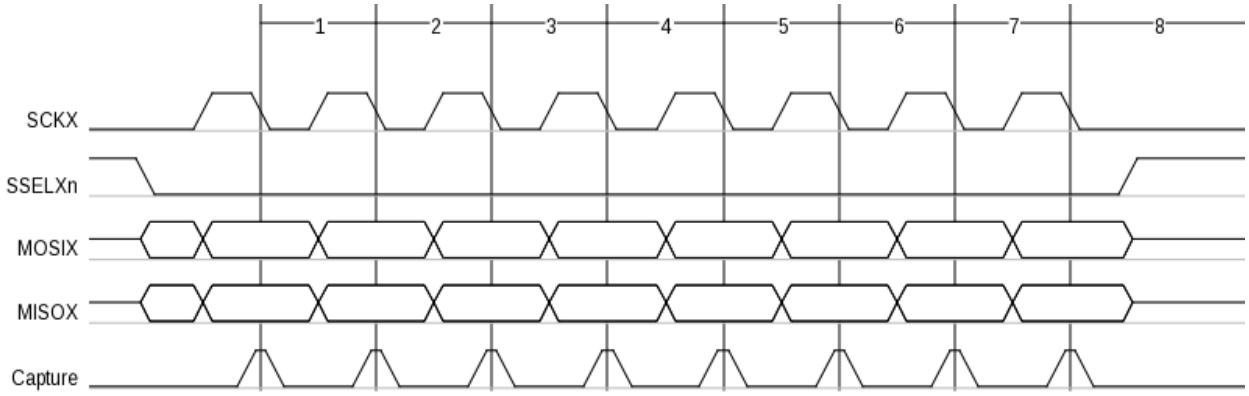


Figure 13 - SPO=0 and SPH=1

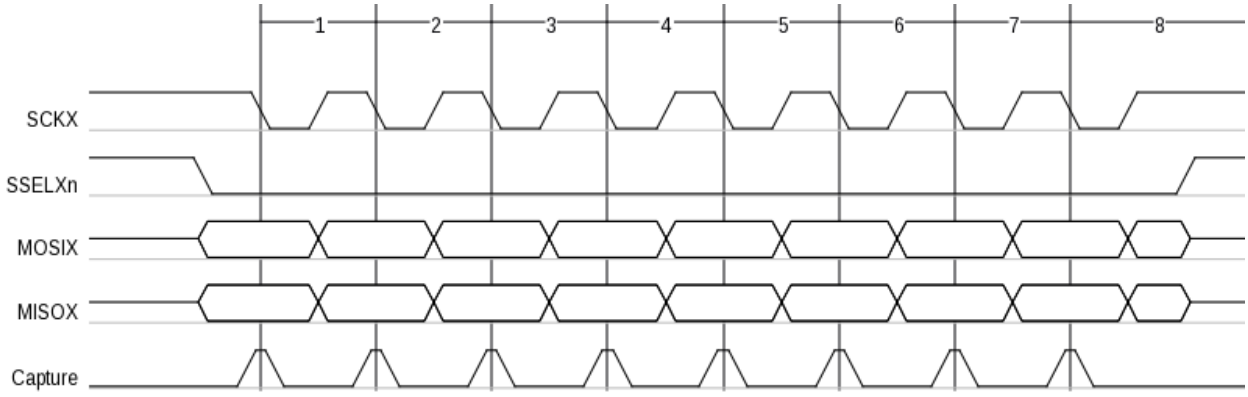


Figure 14 - SPO=1 and SPH=0

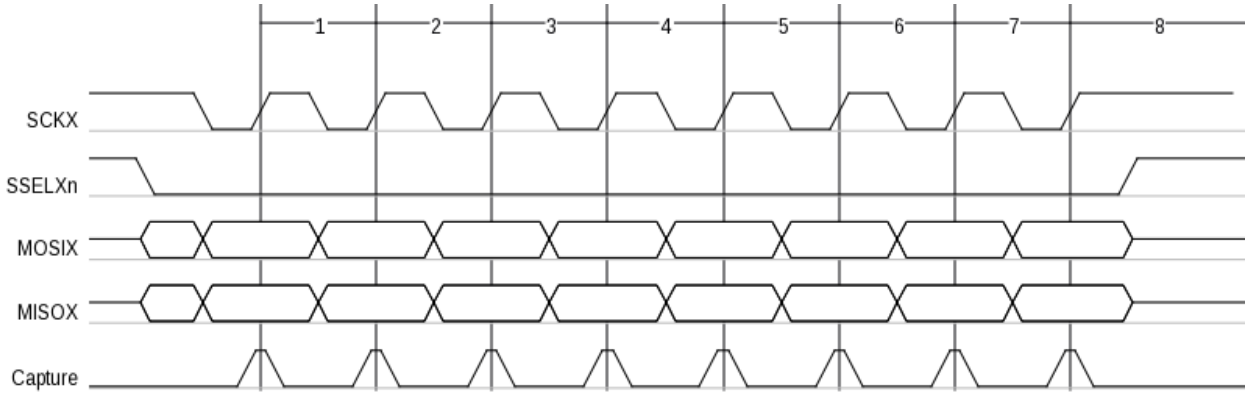


Figure 15 - SPO=1 and SPH=1

4.9.3 CTRL1 Register

The CTRL1 register configures the SPI. When configuring the SPI peripheral, the ENABLE bit should be set last, that is after all the other configuration changes have been made. This allows the logic to initialize the Master/Slave choice (which controls I/O port enabling) before the channel is enabled. Failure to do this can cause output glitches on I/O signals lines when enabling the channel.

The peripheral must have its clock enabled and the interface enabled to send or receive SPI transactions. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. When the ENABLE bit is changed to zero, the interface will become disabled when the controller is idle (completes any pending transaction). Similarly, disabling the clock from the System Configuration Peripheral will not disable the clock until any pending transactions have completed.

Bit	Symbol	Description	Reset value
0	LBM	Loopback Mode	0
1	ENABLE	Enable SPI interface	0
2	MS	Master(0) or Slave(1)	0
3	SOD	Slave Output Disable (All CSn are 1, when disabled)	0
6:4	SS	Slave Select Value. Decoded to generate up to 8 slave select outputs. These are SPI_SSELXn[0-7].	0x0
7	BLOCKMODE	Enable Block Mode. When enabled all data in the FIFO is transmitted as a single SPI frame, unless the BMSTOP bit is set on a data word. An SPI frames is defined as CSn being active for the duration of multiple data words.	0
8	BMSTART	Enable Block Mode Start status. When enabled, bit 31 in the DATA in word will indicate the RXDATAFIRST status. This signifies which Data word is the first received byte in BLOCKMODE.	0
9	BMSTALL	Enable Block Mode Stall. When transmitting in BLOCKMODE and the Transmitter FIFO is empty this enables stalling of the SCK until data is loaded in the FIFO. Note that to end a transmission when BMSTALL is active the BMSTOP bit must be set in the DATA word. This bit only applies to Master/Transmit mode.	0
10	MDLYCAP	Enable Master Delayed Capture mode. This mode is only available when configured as a Master. When enabled this delays the capture of SI data by half a clock cycle, so the data is captured on the same edge as the SO data changes. This can be used to run the SPI faster, provided the external device holds data this long (which is normally the case). Internal part timing is such that data will always be captured on rising SCK before SCK rises external to the chip.	0
11	MTXPAUSE	Pause the Master Transmitter. This allows loading of the Tx FIFO, without transmitting until pause is released.	0
31:12	-	Reserved, Read as 0	

4.9.4 DATA Register

The DATA register that provides access to the SPI input or output FIFOs. Any data written to the register is loaded into the transmit FIFO. Reads from the register return an item from the receive FIFO.

Bit	Symbol	Description	Reset value
15:0	VALUE	SPI data value	0x0000
29:16		Reserved, Read as 0	
30	BMSKIPDATA	This bit is only used for Transmitted data in BLOCKMODE. Setting this bit to 1 along with the BMSTOP bit will end an SPI frame without any additional data to be transmitted. Note, if this is set on the first word of a frame, no transaction is generated. If BMSTOP is not set, this bit is ignored.	0
31	BMSTART/BMSTOP	This bit is only used for BLOCKMODE. For Received data (when BMSTART is enabled in CTRL1), this bit indicates that the data was the first word after ChipSelect went active. For Transmitted data, setting this bit to 1 will end an SPI frame (i.e. ChipSelect) after the specified data word.	0

4.9.5 STATUS Register

The STATUS register provides read-only access to SPI status.

Bit	Symbol	Description	Reset value
0	TFE	Transmit FIFO Empty	1
1	TNF	Transmit FIFO Not Full	1
2	RNE	Receive FIFO Not Empty	0
3	RFF	Receive FIFO Full	0
4	BUSY	Transmit/Receive Busy	0
5	RXDATAFIRST	Indicates that the next Receive FIFO Data word is the first received byte in BLOCKMODE.	0
6	RXTRIGGER	Receive FIFO is above or equals the trigger level	0
7	TXTRIGGER	Transmit FIFO is below the trigger level	1
31:6	-	Reserved, Read as 0	

4.9.6 CLKPRESCALE Register

The CLKPRESCALE register configures the SPI 2x clock generator. Together with SCRDV (bits 15:8 in the CTRL0 register) these define 2 divide counts that are used to build the final SPI clock in master mode.

The SPI clock is primarily used in Master mode. In Slave mode the input SCK clock is used to time transaction. The Slave uses versions of the input signals that are synchronized to the system clock. As a result, the SPI slave is limited to running at 1/12 of the system clock frequency.

The SPI clock will be: SYSCLK / (CLKPRESCALE * (SCRDV + 1))

Where:

- CLKPRESCALE is the CLKPRESCALE register value. If this value is 0, it acts as 1.
- SCRDV is bits 15:8 of CTRL0 register.

Bit	Symbol	Description	Reset value
	VALUE	This bit is always 0	0
7:1	VALUE	Clock divide value.	0x02
31:8	-	Reserved, Read as 0	

4.9.7 IRQ_ENB Register

The IRQ_ENB register provides configuration of the SPI interrupts.

Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

Bit	Symbol	Description	Reset value
0	RORIM	Receive Overrun of the Receive FIFO	0
1	RTIM	Receive Timeout[1]	0
2	RXIM	Receive FIFO is at least half full. Half full is defined as FIFO count >= RXFIFOIRQTRG.	0
3	TXIM	Transmit FIFO is at least half empty. Half empty is defined as FIFO count < TXFIFOIRQTRG.	0
31:4	-	Reserved, Read as 0	

Notes:

1. The Receive Timeout occurs, when the receive FIFO has not been read within 32 clocks ticks (of the SPICLKx2 clock) of receive FIFO being not empty. Reading data from receive FIFO resets the timeout counter. Clearing the RXIM in interrupt status register also resets the time counter. The SPICLKx2 used to clock the timeout register is twice the configured SPI clock rate. Note that when the SPI interface is configured as a slave, the configured SPI clock rate it not used for normal operation (as data is clocked by the SPI clock); but the timeout counter is still clocked by the configured clock rate.

4.9.8 IRQ_RAW Register

The IRQ_RAW register provides read-only access to the raw interrupt status.

Bit	Symbol	Description	Reset value
0	RORIM	Receive Overrun	0
1	RTIM	Receive Timeout	0
2	RXIM	Receive FIFO is at least half full	0
3	TXIM	Transmit FIFO is at least half empty	0
31:4	-	Reserved, Read as 0	

4.9.9 IRQ_END Register

The IRQ_END register provides read-only access to the enabled interrupt status.

Bit	Symbol	Description	Reset value
0	RORIM	Receive Overrun	0
1	RTIM	Receive Timeout	0
2	RXIM	Receive FIFO is at least half full	0
3	TXIM	Transmit FIFO is at least half empty	0
31:4	-	Reserved, Read as 0	

4.9.10 IRQ_CLR Register

The IRQ_CLR status register provides write-only access to clear SPI interrupt status.

Bit	Symbol	Description	Reset value
0	RORIM	Receive Overrun	0
1	RTIM	Receive Timeout	0
31:2	-	Reserved, Read as 0	

4.9.11 RXFIFOIRQTRG Register

The RXFIFOIRQTRG register configures the RX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Receive FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.9.12 TXFIFOIRQTRG Register

The TXFIFOIRQTRG register configures the TX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Transmit FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.9.13 FIFO_CLR Register

The FIFO_CLR register provides access to clearing the Rx or Tx FIFOs.

Bit	Symbol	Description	Reset value
0	RXFIFO	Clear the RX FIFO	0
1	TXFIFO	Clear the TX FIFO	0
31:2	-	Reserved, Read as 0	

4.9.14 STATE Register

The STATE register reports debug information on the TxRx State Machine. Only the RxFifo and TxFifo counts would be of general use.

Bit	Symbol	Description	Reset value
7:0	RXSTATE	TxRx State Machine coding	-
15:8	RXFIFO	Rx FIFO data count	-
23:16	ZERO	Value read as Zero	-
31:24	TXFIFO	Tx FIFO data count	-

4.9.15 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0113 07e1).

4.10 I²C Peripheral (Software label = I2CA and I2CB)

The following diagram shows a general block diagram of the I²C Interface.

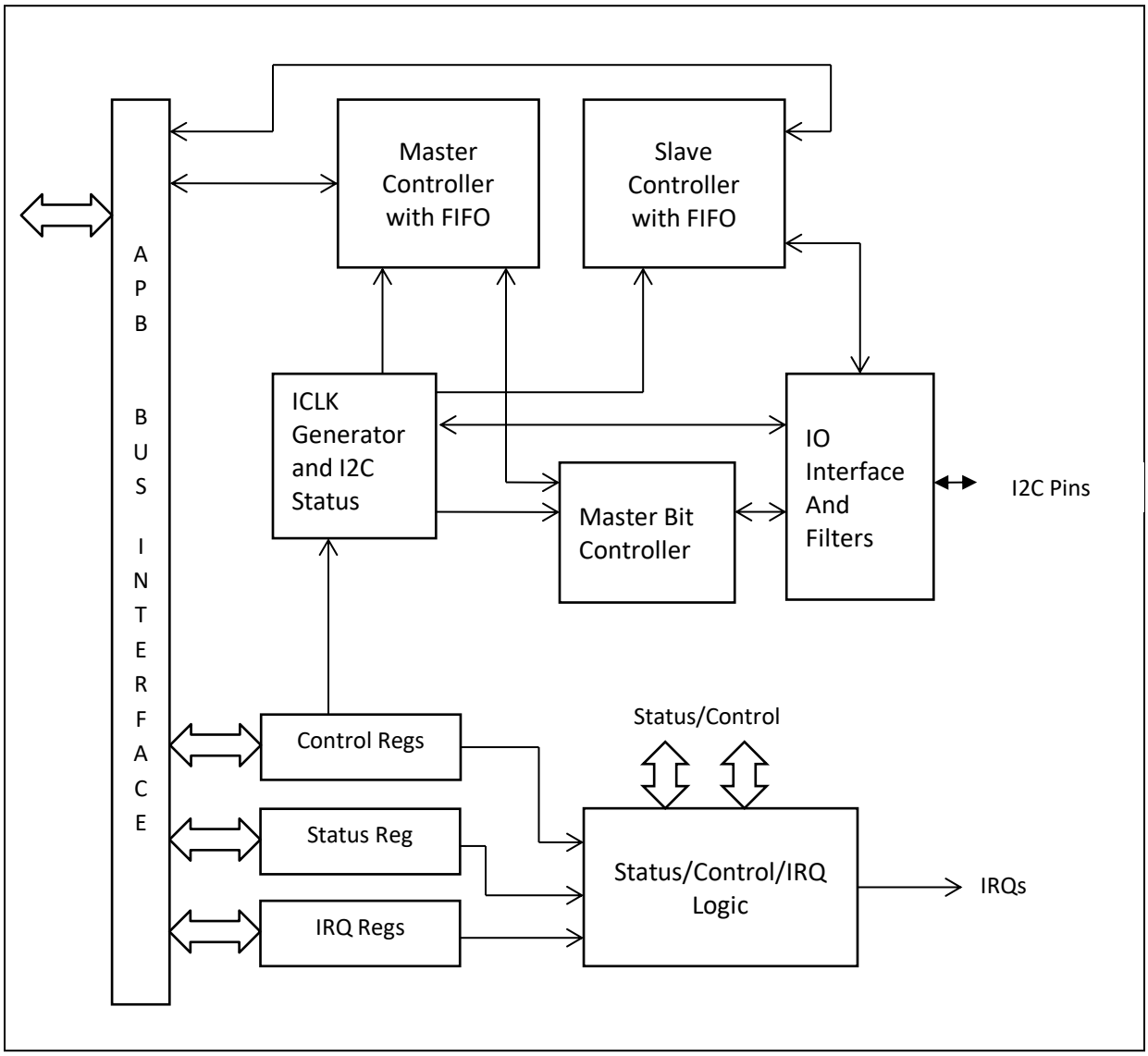


Figure 16 – I²C Block Diagram

The I²C peripheral contains 2 I²C banks.

Each I²C contains a linked Transmit and Receive section, each with a 16 word FIFO.

Both the Master and the Slave support clock stretching under conditions when data it not available for transmitting. See the CTRL and S0_ CTRL registers for details.

The I²C interfaces have dedicated pins that are not shared for other purposes.

Table 32 – I²C Bank Location

Name	Address offset	Description
I2C0	0x0000-0x0FFC	I2C 0 – I2CA
I2C1	0x1000-0x1FFC	I2C 1 – I2CB
	0x2000-0xFFFC	Reserved

Table 33 – I²C Registers

Name	Access	Address offset	Description	Reset value
CTRL	RW	0x000	Control register	0x0000 0000
CLKSCALE	RW	0x004	Clock Scale register	0x0000 0000
WORDS	RW	0x008	Word Count register	0x0000 0001
ADDRESS	RW	0x00C	Address register	0x0000 0000
DATA	RW	0x010	Data register	0x0000 0000
CMD	W	0x014	Command register	0x0000 0000
STATUS	R	0x018	I²C Controller Status register	-
STATE	R	0x01C	Controller State register	-
TXCOUNT	R	0x020	TX Count register	-
RXCOUNT	R	0x024	Rx Count register	-
IRQ_ENB	RW	0X028	Interrupt Enable	0x0000 0000
IRQ_RAW	R	0x02C	Raw Interrupt Status	-
IRQ_END	R	0x030	Enabled Interrupt Status	-
IRQ_CLR	W	0x034	Clear Interrupt Status	-
RXFIFOIRQTRG	RW	0x038	Rx Fifo Trigger level register	0x0000 0008
TXFIFOIRQTRG	RW	0x03C	Tx Fifo Trigger level register	0x0000 0008
FIFO_CLR	W	0x040	FIFO Clear register	-
TMCONFIG	RW	0x044	Timing Config register	0x0000 0000
CLKTOLIMIT	RW	0x048	Clock timeout register	0x0000 0000

	-	0x04c-0xFC	Reserved	
S0_CTRL	RW	0x100	Slave0 Control register	0x0000 0000
S0_MAXWORDS	RW	0x104	Slave0 MaxWord Count register	0x0000 0000
S0_ADDRESS	RW	0x108	Slave0 Address register	0x0000 0000
S0_ADDRESSMASK	RW	0x10C	Slave0 Address Mask register	0x0000 0000
S0_DATA	RW	0x110	Slave0 Data register	0x0000 0000
S0_LASTADDRESS	R	0x114	Slave0 Last Address register	-
S0_STATUS	R	0x118	Slave0 I^2C Controller Status register	-
S0_STATE	R	0x11C	Slave0 Controller State register	-
S0_TXCOUNT	R	0x120	Slave0 TX Count register	-
S0_RXCOUNT	R	0x124	Slave0 Rx Count register	-
S0_IRQ_ENB	RW	0X128	Slave0 Interrupt Enable	0x0000 0000
S0_IRQ_RAW	R	0x12C	Slave0 Raw Interrupt Status	-
S0_IRQ_END	R	0x130	Slave0 Enabled Interrupt Status	-
S0_IRQ_CLR	W	0x134	Slave0 Clear Interrupt Status	-
S0_RXFIFOIRQTRG	RW	0x138	Slave0 Rx Fifo Trigger level register	0x0000 0008
S0_TXFIFOIRQTRG	RW	0x13C	Slave0 Tx Fifo Trigger level register	0x0000 0008
S0_FIFO_CLR	W	0x140	Slave0 FIFO Clear register	-
S0_ADDRESSB	RW	0x144	Slave0 Address B register	0x0000 0000
S0_ADDRESSMASKB	RW	0x148	Slave0 Address B Mask register	0x0000 0000
PERID	RO	0xFFC	Peripheral ID Register	0x0014 07e1

4.10.1 I^2C Master Transactions

The Master controller is used to initiate an I^2C transaction as a master on the bus. This is done by configuring the master registers, and then issuing a command to the CMD register. The

status registers can be monitored to determine completion, or interrupts can be used. Once completed the status needs to be checked to see if the transaction completed as expected.

4.10.1.1 Master Configuration

A typical sequence for configuring the master I²C controller is summarized below:

- Load the CLKSCALE register based on clock and I²C mode (STD/FAST).
- Load the CTRL register as desired
- Set the ENABLE bit in the CTRL register

4.10.1.2 Master Write Transaction

A typical sequence for an I²C write transaction is summarized below:

- Load the WORDS register with the size of the transaction
- Load the ADDRESS register with the destination address and Write active
- Load the required data into the DATA (FIFO) register. (Optionally load DATA in the interrupt subroutine)
- Load the CMD register with 0x03 (Start with Stop)
- If using interrupts:
    - Disable the NVIC interrupt
    - Clear TX FIFO
    - Setup the IRQ_ENB register with only requested sources (Beware that some interrupts may be pending prior to the FIFO being loaded)
    - If using the Half Empty interrupt, set the half empty level in TXFIFOIRQTRG
    - Configure the IRQ Selector Peripheral
    - Set the NVIC priority of the interrupt
    - Enable the NVIC interrupt
- If not using interrupts: Read/Poll the STATUS register until transaction is completed
- Verify transaction completed as desired, potential errors:
    - Negative acknowledge of address
    - Negative acknowledge of data prior to completion
    - Arbitration lost
- If the transaction completed abnormally, clear the transmit FIFO of pending data

4.10.1.3 Master Read Transaction

A typical sequence for an I²C read transaction is summarized below:

- Load the WORDS register with the size of the transaction
- Load the ADDRESS register with the destination address and Read active

- Load the CMD register with 0x03 (Start with Stop)
- If using interrupts:
  - Disable the NVIC interrupt
  - Clear RX FIFO
  - Setup the IRQ_ENB register with only requested sources (Beware that some interrupts may be pending prior to the FIFO being loaded)
  - If using the Half Full interrupt, set the half full level in RXFIFOIRQTRG
  - Configure the IRQ Selector Peripheral
  - Set the NVIC priority of the interrupt
  - Enable the NVIC interrupt
- If not using interrupts: Read/Poll the STATUS register until transaction is completed
- Verify transaction completed as desired, potential errors:
  - Negative acknowledge of address
  - Insufficient data received
  - Arbitration lost
- If the transaction completed abnormally, clear the receive FIFO of unwanted data

4.10.1.4 Master Write-Read Transaction

A typical sequence for an I^2C write-read transaction is summarized below:

- Load the WORDS register with the size of the write transaction
- Load the ADDRESS register with the destination address and Write active
- Load the required data into the DATA (FIFO) register
- Load the CMD register with 0x01 (Start without Stop)
- Read/Poll the STATUS register until transaction is completed
- Verify transaction completed as desired, potential errors:
  - Negative acknowledge of address
  - Negative acknowledge of data prior to completion
  - Arbitration lost
- If the transaction completed abnormally:
  - Clear the transmit FIFO of pending data
  - Load the CMD register with 0x02 (to stop the bus if still the owner)
  - Exit flow
- Load the WORDS register with the size of the read transaction
- Load the ADDRESS register with the destination address and Read active
- Load the CMD register with 0x03 (Start [this will be a ReStart] with Stop)
- Read/Poll the STATUS register until transaction is completed

- Verify transaction completed as desired, potential errors:
  - Negative acknowledge of address
  - Insufficient data received
- If the transaction completed abnormally, clear the receive FIFO of unwanted data

4.10.2 I²C Master Registers

4.10.2.1 CTRL Register

The CTRL register configures the I²C interface.

The peripheral must have its clock enabled and the master interface enabled to generate I²C commands. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. When the clock is enabled but the master interface is not enabled the I²C bus status (BUSY/IDLE) is still maintained. This status is needed to know when it is valid to start transactions.

When the ENABLE bit is changed to zero, the interface will be disabled (ENABLED going to zero) when the master controller is idle (completes any pending transaction). Similarly, the CLKENABLED bit will go to zero when the interface clock is disabled and both the master and slave interfaces are idle. While the clock is disabled, all registers in the peripheral are blocked from updates, and the I²C IO buffers are disabled.

Bit	Symbol	Description	Reset value
0	CLKENABLED	Clock Enabled Status (from System Configuration Peripheral). (ReadOnly)	0
1	ENABLED	Master interface enabled status. (ReadOnly)	0
2	ENABLE	Enable master interface.	0
3	TXFEMD	Transmit on Empty FIFO Mode: 0 – Stall - The I2C clock is stretched until data is available. 1 – End Transaction This bit determines what happens during a transmit operation when the transmit FIFO is empty. Either the transaction is stalled or the transaction is ended. Note that if the transmit FIFO is empty at the start of the transaction (the first data byte) the transaction is always stalled.	0
4	RXFFMD	Receive on Full FIFO Mode:	0

		0 – Stall - The I2C clock is stretched until data is available. 1 – Negative Acknowledge This bit determines what happens during a receive operation when the receive FIFO is full. Either the transaction is stalled or the transaction is ended with a negative acknowledge. Note that if the receive FIFO is full at the start of the transaction (the first data byte) the transaction is always stalled.	
5	ALGFILTER	Enable the analog delay glitch filter	0
6	DLGFILTER	Enable the digital glitch filter	0
7	Reserved	Reserved	0
8	LOOPBACK	LoopBack Mode – Disconnect I^2C interface from any IO pins and directly connect the Master to the Slave.	0
9	TMCONFIGENB	Enable Timing Config register	0
31:10	-	Reserved, Read as 0	

4.10.2.2 CLKSCALE Register

The CLKSCALE register configures the I^2C clock generator (ICLK). This defines the clock divide count used to generate a 20X/25X ICLK clock for the I^2C controller. Thus the system clock must be at least 20X or 25X of the I^2C clock rate. For Standard mode a 20X clock used, which must have a minimum period of 500ns. For Fast mode a 25X clock is used, which must have a minimum period of 100ns.

The divide value needs to be configured to match the proper I^2C bus rates. This divide value must satisfy the follow formula:

$$I2C_{period} = (1+VALUE) * Multiplier * SystemClock_{period}$$

Where Multiplier is 20 for Standard mode, and 25 for Fast mode.

Table 34 – Clock Scale Register

Bit	Symbol	Description	Reset value
7:0	VALUE	Clock divide value.	0x18
30:8	-	Reserved, Read as 0	
31	FASTMODE	Enable Fast Mode. Changes internal timing slightly.	0

Table 35 – Clock Scale Values

System Clock Rate	System Clock Period	Standard-Mode (100kHz) 20X Period = 500ns	Fast-Mode Divide Value (400kHz) 25X Period = 100ns
50MHz	20ns	24	4
40MHz	25ns	19	3
30MHz	33.3ns	14	2
20MHz	50ns	9	1
15MHz	66.7ns	7 (~94kHz)	1 (~300kHz)
10MHz	100ns	4	0
8MHz	125ns	3	None[1]
6MHz	166.7ns	2	None[1]
5MHz	200ns	2 (~83kHz)	None[1]
4MHz	250ns	1	None[1]
2MHz	500ns	0	None[1]

Notes:

1. There is no recommended divide value, that will work for full speed 400kHz Fast Mode.

4.10.2.3 WORDS Register

The WORDS register controls how many data words are included in the I^2C transaction. This value can range from 1 to 0x7fe for receive operation and 0 to 0x7fe for transmit operations. For transmit operations, the transaction will complete after the required number of words have been sent, or a word is negative acknowledged. For receive operations, the desired number of words are received, with the last word being negative acknowledged (so the slave knows it is the end of the transaction).

For transmit operation, if the Word value is set to 0x7ff the transaction will not end until the value is set to a value below the current TXCOUNT. For receive operation, the module never enters into continuous operation if Word value is set to 0x7ff.

Bit	Symbol	Description	Reset value
10:0	VALUE	Word Count	1
31:11		Reserved, Read as 0	

4.10.2.4 ADDRESS Register

The ADDRESS register controls the address used for the I²C transaction. This also includes the direction bit as bit 0 of the register. The number of address bits used it determined by the A10MODE setting.

Bit	Symbol	Description	Reset value
0	DIRECTION	Transaction direction. 0 is Send, 1 is Receive	0
10:1	ADDRESS	Address value	0x000
14:11	Reserved	Reserved, Read as 0	
15	A10MODE	Enable 10Bit address mode	0
31:16		Reserved, Read as 0	0

4.10.2.5 DATA Register

The DATA register that provides access to the I²C input or output FIFOs. Any data written to the register is loaded into the transmit FIFO. Reads from the register return an item from the receive FIFO.

Bit	Symbol	Description	Reset value
7:0	VALUE	I²C data value	-
31:8		Reserved, Read as 0	

4.10.2.6 CMD Register

The CMD register controls starting of transfers by the I²C master controller.

Writes to this register are used to initiate I²C transactions by the controller. The START bit is set to start a transaction, if the STOP is not also set, then a STOP will not be sent after the transaction, allowing subsequent transaction to be started without releasing the bus. If the bus is not released it can be released at the end of the next transaction, or by setting the STOP bit alone.

The CANCEL bit is used to abort a transaction that has been started. Following a CANCEL operation, the FIFOs should be reset by the software to clear out any remaining data.

Bit	Symbol	Description	Reset value
0	START	Include an I²C START with the transaction	0
1	STOP	Include an I²C STOP with the transaction	0
2	CANCEL	Cancel a currently started transaction	0

31:3	-	Reserved, Read as 0	

4.10.2.7 STATUS Register

The STATUS register provides read-only access to I²C controller status.

Bit	Symbol	Description	Reset value
0	I2CIDLE	Indicates if the I²C bus is busy(0) or idle(1)	1
1	IDLE	Indicates if the Controller is busy(0) or idle(1)	1
2	WAITING	Indicates if the Controller is waiting(1). Waiting indicates that the transaction has completed, but that bus has not been released (by a STOP)	0
3	STALLED	Indicates that the Controller is stalled waiting for the data FIFOs; either a send is in progress and the FIFO is empty, or a receive is in progress and the FIFO is full.	0
4	ARBLOST	Indicates that Arbitration was lost on the last transaction	0
5	NACKADDR	Indicates that an Address had a Negative Acknowledge	0
6	NACKDATA	Indicates that a Data word had a Negative Acknowledge	0
7	Reserved		0
8	RXNEMPTY	The Receive FIFO is not empty	0
9	RXFULL	The Receive FIFO is full	0
10	Reserved		0
11	RXTRIGGER	The Receive FIFO is above or equals the trigger level	0
12	TXEMPTY	The Transmit FIFO is empty	0
13	TXNFULL	The Transmit FIFO is Not full	0
14	Reserved		0
15	TXTRIGGER	The Transmit FIFO is below the trigger level	0
29:16	-	Reserved, Read as 0	
30	RAW_SDA	Raw I²C Data value	-
31	RAW_SCL	Raw I²C Clock value	-

4.10.2.8 STATE Register

The STATE register provides read-only access to I²C Controller state. This is primarily for test purposes, and the details of the Controller State/Step values are not documented. The Rx/Tx FIFO levels can be used to determine the amount of data in each FIFO.

Bit	Symbol	Description
3:0	STATE	Controller State
7:4	STEP	Controller Step
12:8	RXFIFO	Rx FIFO Level
13	0	Reserved
18:14	TXFIFO	Tx FIFO Level
19	0	Reserved
28:20	BITSTATE	Bit Controller State
31:29	-	Reserved

4.10.2.9 TXCOUNT Register

The TXCOUNT register provides read-only access to the count of the number of data words sent during the last transaction. If the last word sent was negative acknowledged it is still included in the count. If the counter reaches a count of 0x7fe it stops counting.

Bit	Symbol	Description	Reset value
10:0	VALUE	Count Value	0x000
31:11	-	Reserved, Read as 0	

4.10.2.10 RXCOUNT Register

The RXCOUNT register provides read-only access to count of the number of data words received during the last transaction. If the last word received was negative acknowledged it is still included in the count. If the counter reaches a count of 0x7fe it stops counting.

Bit	Symbol	Description	Reset value
10:0	VALUE	Count Value	0x000
31:11	-	Reserved, Read as 0	

4.10.2.11 IRQ_ENB Register

The IRQ_ENB register provides configuration of the I²C interrupts.

Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

Bit	Symbol	Description	Reset value
6:0	STATUS	Triggered by rising edge of values in STATUS register (See lower 7 bits of STATUS register for bit definitions)	0x00
7	CLKLOTO	Clock Low timeout	0
9:8	-	Reserved	0x0
10	TXOVERFLOW	Tx FIFO Overflow	0
11	RXOVERFLOW	Rx FIFO Overflow	0
12	TXREADY	Tx FIFO is ready for more data (Uses trigger level status)	0
13	RXREADY	Rx FIFO has data ready (Uses trigger level status)	
14	TXEMPTY	Tx FIFO is empty	0
15	RXFULL	Rx FIFO is full	0
31:16	-	Reserved, Read as 0	

4.10.2.12 IRQ_RAW Register

The IRQ_RAW register provides read-only access to the I^2C interrupt status.

Bit	Symbol	Description	Reset value
31:0	VALUE	See I^2C IRQ_ENB	0x0000 0000

4.10.2.13 IRQ_END Register

The IRQ_END register provides read-only access to the I^2C interrupt enabled status.

Bit	Symbol	Description	Reset value
31:0	VALUE	See I^2C IRQ_ENB	0x0000 0000

4.10.2.14 IRQ_CLR Register

The IRQ_CLR register provides write-only access to clear some of the pending I^2C interrupts. Writing a 1 to a given bit in the register clears the associated interrupt.

Bit	Symbol	Description	Reset value
6:0	STATUS	Writing a 1 to these bits has no impact. These bits are always indicators of the peripheral state. (See lower 7 bits of STATUS register for bit definitions)	0x00
7	CLKLOTO	Clock Low timeout	0
9:8	-	Reserved	0x0
10	TXOVERFLOW	Tx FIFO Overflow	0

11	RXOVERFLOW	Rx FIFO Overflow	0
31:12	–	Reserved	0x0000 0000

4.10.2.15 RXFIFOIRQTRG Register

The RXFIFOIRQTRG register configures the RX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Receive FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.10.2.16 TXFIFOIRQTRG Register

The TXFIFOIRQTRG register configures the TX FIFO half empty level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half empty level for the Transmit FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.10.2.17 FIFO_CLR Register

The FIFO_CLR register provides access to clearing the Rx or Tx FIFOs.

Bit	Symbol	Description	Reset value
0	RXFIFO	Clear the RX FIFO	0
1	TXFIFO	Clear the TX FIFO	0
31:2	-	Reserved, Read as 0	

4.10.2.18 TMCONFIG Register

The TMCONFIG register provides access to an alternate timing configuration settings for the I^2C state machine. The use of this register is controlled by the TMCONFIGENB bit in the CTRL register. The reset value of this register is the default values for Standard Mode timing.

Bit	Symbol	Description	Reset value
3:0	TR	Tr time	0x2
7:0	TF	Tf time	0x1
11:8	THIGH	Thigh time	0x8
15:12	TLOW	Tlow time	0x9
19:16	TSUsto	Tsu;sto time	0x8

23:20	TSUsta	Tsu;sta time	0xa
27:24	THDsta	Thd;sta time	0x8
31:28	Tbuf	Tbuf time	0xa

4.10.2.19 CLKTOLIMIT Register

The CLKTOLIMIT register provides a timeout limit on the amount of time the I²C clock is seen to be low. When this register is non-zero, the clock lower counter is enabled. The clock low counter counts the number of cycles that clock is seen to be low. It counts using the clock generated by the clock scale register. When the count value is greater than the counter limit, the clock low interrupt is generated.

Bit	Symbol	Description	Reset value
19:0	VALUE	Clock low limit value. Needs to be non-zero to enable	0x0 0000
31:20	-	Reserved, Read as 0	

4.10.2.20 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0014 07e1).

4.10.3 I²C Slave Registers

This section describes the registers associated with the slave side of the I²C interface.

The slave uses data from the following Master side registers: CLKSCALE, CTRL bits 9-5 (Filter controls, loopback, and timing), and TMCONFIG register if it is enabled.

4.10.3.1 S0_CTRL Register

The S0_CTRL register configures the I²C slave interface.

The peripheral must have its clock enabled and the slave interface enabled to receive I²C commands. The clock enable is controlled by the Clock Enable in the System Configuration Peripheral. When the clock is enabled but the salve interface is not enabled the I²C bus status (BUSY/IDLE) is still maintained. This status is needed to know when it is valid to start transactions.

When the ENABLE bit is changed to zero, the interface will be disabled (ENABLED going to zero) when the slave controller is idle (completes any pending transaction). Similarly, the CLKENABLED bit will go to zero when the interface clock is disabled and both the master and slave interfaces are idle. While the clock is disabled, all registers in the peripheral are blocked from updates, and the I²C IO buffers are disabled.

Bit	Symbol	Description	Reset value
0	CLKENABLED	Clock Enabled Status (from System Configuration Peripheral). (Read Only)	0
1	ENABLED	Slave interface enabled status. (Read Only)	0
2	ENABLE	Enable slave interface.	0
3	TXFEMD	Transmit on Empty FIFO Mode: 0 – Stall – The I2C clock is stretched until data is available. 1 – Negative Acknowledge. When this bit is set to 1, and the FIFO is empty at the time of receiving an address match for a transmit operation, the address will be negative acknowledged. Note that data cannot be negative acknowledged when the FIFO is empty, as that would appear as arbitration lost; as the master generates acknowledges during data transmit from slave. Thus when this bit is 1 and a transmit operation is required with the FIFO being empty, a FIFO underflow will result, and a 0xff data word will be transmitted.	0
4	RXFFMD	Receive on Full FIFO Mode: 0 – Stall - The I2C clock is stretched until data is available. 1 – Negative Acknowledge	0
31:5	-	Reserved, Read as 0	

4.10.3.2 S0_MAXWORDS Register

The S0_MAXWORDS register controls the maximum number of words that will be received by the slave before generating a Negative Acknowledge. This is normally set to value in the

range of 0 to 0x7fe. Setting the value to 0x7ff has the same effect as not setting the ENABLE bit, since the RXCOUNT stops counting at 0x7fe.

Bit	Symbol	Description	Reset value
10:0	MAXWORD	MaxWord Count	0x000
30:11		Reserved, Read as 0	
31	ENABLE	Enables the MaxWord count value. When this bit is 0 the MAXWORDS value is ignored.	0

4.10.3.3 S0_ADDRESS and S0_ADDRESSMASK Registers

The S0_ADDRESS and S0_ADDRESSMASK registers control which addresses the slave interface responds to. A received address is compared to the ADDRESS register using the ADDRESSMASK. Those bits with a 1 in the ADDRESSMASK must match for there to be an address match.

Table 36 – S0_ADDRESS Register

Bit	Symbol	Description	Reset value
0	RW	Read/Write Value	0
10:1	ADDRESS	Address value	0x000
14:11	Reserved	Reserved, Read as 0	
15	A10MODE	Enable 10Bit address mode	0
31:16		Reserved, Read as 0	

Table 37 – S0_ADDRESSMASK Register

Bit	Symbol	Description	Reset value
0	RWMASK	Read/Write Mask (Should normally be 0 to match both read and write addresses).	0
10:1	MASK	Address mask value	0x3ff
31:11		Reserved, Read as 1	

4.10.3.4 S0_DATA Register

The S0_DATA register provides access to the I^2C slave input or output FIFOs. Any data written to the register is loaded into the transmit FIFO. Reads from the register return an item from the receive FIFO.

Bit	Symbol	Description	Reset value
7:0	VALUE	I2C data value	0x00
31:8		Reserved, Read as 0	

4.10.3.5 S0_LASTADDRESS Register

The S0_LASTADDRESS register provides read-only access to the last address that was matched by the slave controller.

Bit	Symbol	Description	Reset value
0	DIRECTION	Transaction direction. 0 is Master Send, 1 is Master Receive	0
10:1	ADDRESS	Address value	0x000
31:11		Reserved, Read as 0	

4.10.3.6 S0_STATUS Register

The S0_STATUS register provides read-only access to I^2C slave controller status.

Bit	Symbol	Description	Reset value
0	COMPLETED	Indicates a possible transaction has completed, either from a stop, or restart	0
1	IDLE	Indicates if the Controller is busy(0) or idle(1)	1
2	WAITING	Indicates if the Controller is waiting(1). Waiting indicates that the transaction has completed, but that bus has not been released (by a STOP)	0
3	TXSTALLED	Indicates that the Controller is stalled waiting for transmit data from the Tx FIFOs (The Tx FIFO is empty).	0
4	RXSTALLED	Indicates that the Controller is stalled waiting for the data to the RX FIFO (the Rx FIFO is full).	0
5	ADDRESSMATCH	Indicates an address match against the received address	0
6	NACKDATA	Indicates that Data word had a Negative Acknowledge	0
7	RXDATAFIRST	Indicates that the Pending Data word is the first received byte following an address match	0
8	RXNEMPTY	The Receive FIFO is Not empty	0
9	RXFULL	The Receive FIFO is full	0
10	Reserved		0

11	RXTRIGGER	The Receive FIFO is above or equals the trigger level	0
12	TXEMPTY	The Transmit FIFO is empty	0
13	TXNFULL	The Transmit FIFO is Not full	0
14	Reserved		0
15	TXTRIGGER	The Transmit FIFO is below the trigger level	0
28:16	-	Reserved, Read as 0	
28	RAW_BUSY	I^2C bus is busy	-
30	RAW_SDA	I^2C Data value	-
31	RAW_SCL	I^2C Clock value	-

4.10.3.7 S0_STATE Register

The S0_STATE register provides read-only access to I^2C slave controller state. This is primarily used for test purposes.

Bit	Symbol	Description
2:0	STATE	Controller State
3		Reserved, Reads as 0
7:4	STEP	Controller Step
12:8	RXFIFO	Rx FIFO Level
13	0	Reserved
18:14	TXFIFO	Tx FIFO Level
31:19	-	Reserved

4.10.3.8 S0_TXCOUNT Register

The S0_TXCOUNT register provides read-only access to the count of the number of data words sent during the last transaction. If the last word sent was negative acknowledged it is still included in the count. The count does not include the address bytes. If the counter reaches a count of 0x7fe it stops counting.

Bit	Symbol	Description	Reset value
10:0	VALUE	Count Value	0x000
31:11	-	Reserved, Read as 0	

4.10.3.9 S0_RXCOUNT Register

The S0_RXCOUNT register provides read-only access to count of the number of data words received during the last transaction. If the last word received was negative acknowledged it is

still included in the count. The count does not include the address bytes. If the counter reaches a count of 0x7fe it stops counting.

Bit	Symbol	Description	Reset value
10:0	VALUE	Count Value	0x000
31:11	-	Reserved, Read as 0	

4.10.3.10 S0_IRQ_ENB Register

The S0_IRQ_ENBregister provides configuration of the I²C interrupts.

Enabling interrupts allows the peripheral to generate interrupts. For the processor to see the interrupt it must also be configured in the IRQ Selector Peripheral to one of the processor interrupts, and enabled in the processors NVIC.

Bit	Symbol	Description	Reset value
7:0	STATUS	Triggered by rising edge of values in S0_STATUS register (See S0_STATUS register for bit definitions)	0x00
8	I2C_START	I²C Start Condition detected	0
9	I2C_STOP	I²C Stop Condition detected	0
10	TXUNDERFLOW	Tx FIFO Underflow	0
11	RXOVERFLOW	Rx FIFO Overflow	0
12	TXREADY	Tx FIFO is ready for more data	0
13	RXREADY	Rx FIFO has data ready	0
14	TXEMPTY	Tx FIFO is empty	0
15	RXFULL	Rx FIFO is full	0
31:16	-	Reserved	0

4.10.3.11 S0_IRQ_RAW Register

The S0_IRQ_RAW register provides read-only access to the I²C interrupt status.

Bit	Symbol	Description	Reset value
31:0	VALUE	See I²C Slave IRQ_ENB	0x0000 0000

4.10.3.12 S0_IRQ_END Register

The S0_IRQ_END register provides read-only access to the I²C interrupt enabled status.

Bit	Symbol	Description	Reset value
31:0	VALUE	See I²C Slave IRQ_ENB	0x0000 0000

4.10.3.13 S0_IRQ_CLR Register

S0_IRQ_CLR register provides write-only access to clear I²C interrupt status. Writing a 1 to a given bit in the register clears the associated interrupt.

Bit	Symbol	Description	Reset value
31:0	VALUE	See I²C Slave IRQ_ENB	0x0000 0000

4.10.3.14 S0_RXFIFOIRQTRG Register

The S0_RXFIFOIRQTRG register configures the RX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Receive FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.10.3.15 S0_TXFIFOIRQTRG Register

The S0_TXFIFOIRQTRG register configures the TX FIFO half full level.

Bit	Symbol	Description	Reset value
4:0	LEVEL	Specifies the Half full level for the Transmit FIFO1	0x08
31:5	-	Reserved, Read as 0	

4.10.3.16 S0_FIFO_CLR Register

The S0_FIFO_CLR register provides access to clearing the Rx or Tx FIFOs.

Bit	Symbol	Description	Reset value
0	RXFIFO	Clear the RX FIFO	0
1	TXFIFO	Clear the RX FIFO	0
31:2	-	Reserved, Read as 0	

4.10.3.17 S0_ADDRESSB and S0_ADDRESSMASKB Registers

The S0_ADDRESSB and S0_ADDRESSMASKB registers controls an additional slave address to which the slave interface will responds to. Received addresses are compared to the

ADDRESSB register using the ADDRESSMASKB. Those bits with a 1 in the ADDRESSMASKB must match for there to be an address match.

The second address is enabled with the ADDRESSBEN bit in the S0_ADDRESSB register. This second address uses the A10MODE bit from the primary address register (so both addresses are either in 7-bit or 10-bit mode at the same time).

Table 38 – S0_ADDRESSB Register

Bit	Symbol	Description	Reset value
0	RW	Read/Write Value	0
10:1	ADDRESS	Address value	0x000
14:11	Reserved	Reserved, Read as 0	
15	ADDRESSBEN	Enable Address B	0
31:16		Reserved, Read as 0	

Table 39 – S0_ADDRESSMASKB Register

Bit	Symbol	Description	Reset value
0	RWMASK	Read/Write Mask (Should normally be 0 to match both read and write addresses).	0
10:1	MASK	Address mask value	0x3ff
31:11		Reserved, Read as 1	

4.10.4 PERID Register

This is a read-only register that returns the ID of the peripheral (0x0014 07e1).

5 JTAG controller and eFuse block

The VA108X0 ARM® Cortex®-M0 Processor chip is designed and built with specific test related features. These features can be used to test features in the devices that are not accessible through normal functional operation of the part. Most of these test features are accessed using the IEEE 1149.1 standard test port (JTAG).

The JTAG port is also used to read and program the on-chip eFuse block that controls the power-up sequence of the part. The eFuse can be programmed up to 30 times. Each of these 30 programming cycles can change the 32-bit boot configuration or a 32-bit unique device ID number. The unique device ID is loaded from the eFuse memory as the part boots and is available in user space in the EF_ID register in the System Configuration peripheral.

The JTAG port can also be used to send encapsulated SPI commands to access an SPI ROM attached to the devices SPI ROM Boot port. This can be used to do in-system programming of the SPI ROM through the JTAG port without adding additional logic to the board to provide access to the SPI ROM.

The JTAG port on the part consists of pins: TCK, TMS, TRSTn, TDI, and TDO.

Internal to the part there are 2 JTAG controllers. These 2 controllers are connected in series such that the TDO of the first controller is connected to TDI of the second controller. The first controller (connected to the TDI pin) is used to access the debug port of the ARM® Cortex®-M0 processor. The second controller in the sequence is the controller used to access chip level configuration and test.

5.1 ARM® Cortex®-M0 JTAG Controller

The ARM® Cortex®-M0 JTAG controller is the standard Debug controller from ARM®. It is used to access the debug port on the ARM® Cortex®-M0.

One special note, when the chip level JTAG controller is used to enable SCAN mode, the ARM® controller is removed from the TDI/TDO path. The part acts as if only one JTAG controller exists at that point. This is done so that the ARM® controller can be switched to run off the system clock and be included in scan testing. There is 1-bit flip-flop added in the TDI/TDO path when in this mode; so that it acts like a normal BYPASS register still exists for the missing controller. After exiting SCAN mode, the TRSTn should be asserted to reset the JTAG controllers; as the ARM® controller will most likely be in an unknown state.

5.1.1 Instruction Codes

The ARM® controller contains a 4-bit instruction register. The implemented instruction codes are listed below:

Instruction Register Code	Instruction Name	Description
4'b1000	ABORT	Select Abort Chain
4'b1010	DPACC	Selects DP Chain
4'b1011	APACC	Selects AP Chain
4'b1110	IDCODE	Standard JTAG Instruction
4'b1111	BYPASS	Standard JTAG Instruction

5.2 Chip Level JTAG C8ontroller

The chip level JTAG controller provides access to the following features:

- IEEE 1149.1 Boundary Scan (Vorago test access only)
- Built-In Self-Test (BIST) of internal memories (Vorago test access only)
- Scan Testing (Vorago test access only)
- IDDQ Testing (Vorago test access only)
- Parametric NAND Tree (Vorago test access only)
- Power-On-Reset testing (Vorago test access only)
- Reduced default ROM startup delay
- Reduced default ROM size loading
- Device resets
- Internal eFuse access
- SPI ROM access

5.2.1 Instruction Codes

The chip level JTAG controller contains a 5-bit instruction register. The implemented instruction codes are listed below:

Instruction Register Code	Instruction Name	Description
5'h00	ALTBYPASS	Implemented as BYPASS
5'h01	EXTEST	Standard JTAG Instruction
5'h02	CLAMP	Standard JTAG Instruction
5'h03	HIGHZ	Standard JTAG Instruction

5'h04	SAMPLE/PRELOAD	Standard JTAG Instruction
5'h05	PULL	Pull Mode (Vorago test access only)
5'h06	IDCODE	Standard JTAG Instruction
5'h07	PULLEN	Pull Enable Register (Vorago test access only)
5'h08	TEST_CFG	Test Configuration Register (Vorago test access only)
5'h09	CLK_CFG	Clock Configuration Register (Vorago test access only)
5'h0a	BIST_MODE	BIST Mode Register (Vorago test access only)
5'h0b	BIST_RST	BIST Reset (Vorago test access only)
5'h0c	BIST_CFG	BIST Configuration Register (Vorago test access only)
5'h0d	BIST_STAT	BIST Status Register (Vorago test access only)
5'h0e	BIST_RUN	BIST Run (Vorago test access only)
5'h0f	MEM_MRG	Memory Margin Register (Vorago test access only)
5'h10	BOOT_CFG	Boot Configuration Register
5'h11	RESET_CFG	Reset Configuration Register
5'h12		Reserved
5'h13		Reserved
5'h14	EF_ADDR	eFuse Address Register
5'h15	EF_WDATA	eFuse Write Data Register
5'h16	EF_RDATA	eFuse Read Data Register
5'h17	ER_TIMING	eFuse Timing Register
5'h18	EF_CMD	eFuse Command Register
5'h19	EF_STATUS	eFuse Status Register
5'h1a	SPI_CONFIG	SPI Encapsulation Configuration Register
5'h1b	SPI_ENCAP	SPI Encapsulation Register
5'h1c		Reserved
5'h1d	HBO_CMD	HBO Command Register (Vorago test access only)
5'h1e	HBO_STATUS	HBO Status Register (Vorago test access only)
5'h1f	BYPASS	Standard JTAG Instruction

5.2.2 BOOT_CFG Register

The BOOT_CFG register is a 32-bit register defined below.

Bit	Reset State	Name	Description
1:0	0x1	ROM_SPEED	This value specifies the speed of ROM_SCK: 0 – CLK divide by 2 (@50MHz => 25MHz) 1 – CLK divide by 6 (@50MHz => 8.33MHz) 2 – CLK divide by 12 (@50MHz => 4.2MHz) 3 – CLK divide by 52 (@50MHz => 962kHz)

4:2	0x0	ROM_SIZE	This value specifies how much of the full 128K byte address space is loaded from the external SPI memory after a reset. The values are: 0 – Full 128k Bytes of address space 1 – First 64K bytes of address space 2 – First 32K bytes of address space 3 – First 16K bytes of address space 4 – First 8K bytes of address space 5 – First 4K bytes of address space The part of the address space not loaded is initialized to zero.
5	0	ROM_NOCHECK	When set to 1, the ROM check is skipped. When set to 0, the ROM check is enabled. In ROM check mode, each 128 byte block of data from the SPI ROM is read twice. If the two reads get different results, the process is repeated until it read the same twice.
8:6	0x4	BOOT_DELAY	This value specifies the boot delay from the end of the Power-On-Sequence to the release of Reset. The delay is based on cycles of the internal nominal 1MHz oscillator. 0 – 1 cycles (~ 0ms) 1 – 1000 cycles (~1ms) 2 – 3000 cycles (~3ms) 3 – 10000 cycles (~10ms) 4 – 30000 cycles (~30ms) 5 – 100000 cycles (~100ms) 6 – 300000 cycles (~300ms) 7 – 500000 cycles (~500ms)
16:9	0x03	ROM_READ	SPI ROM read instruction code.
21:17	0x00	ROM_LATENCY	Number of bits of latency from Address until data from the SPI ROM.
23:22	0x1	ROM_ADDRESS	Rom Address Mode. Specifies the number of bits/bytes to use for addressing the SPI ROM. 0 – 16 bit / 2 bytes 1 – 24 bit / 3 bytes 2 – 32 bit / 4 bytes
24	1	ROM_DLYCAP	ROM SPI Delayed capture. When 1, the ROM_SI data is captured on clock falling edge instead of the normal SPI mode of the rising edge. This allows almost a full SCK clock cycle for the SPI ROM to respond from address to data. This mode allows a faster SPI cycle time.

25	0	ROM_STATUS	In this mode, the first data byte from the SPI ROM following an address is taken as a status byte. A Zero status data byte indicates valid data follows. A non-zero status data byte indicates the device is busy, and that the SPI transaction should be restarted.
26-30	0x2		These bits control internal read timing and must be maintained at this value (0x2).
31	0	ENABLE	Enables the config settings. When this bit is 0, the other bits in this register are ignored. When this bit is 1, the other bits in this register are used.

5.2.3 RESET_CFG Register

The RESET_CFG register is a 9-bit register that is used to control internal reset events.

Bit	Reset State	Description
2:0	0x0	While set to 1, the Power-On-Reset is manually activated. There are three bits; 1 bit for each of the 3 POR blocks.
3	0	While set to 1, the EXTRESET is manually activated.
4	0	While set to 1, the internal Power-On-Reset is disabled.
5	0	While set to 1, the EXTRESET reset is disabled.
6	0	While set to 1, the system boot process is disabled.
7	0	While set to 1, the internal nominal 1MHz oscillator is disabled
8	1	While set to 1, the Power-On-Sequence is clocked from the CLK pin instead of the nominal 1MHz oscillator.

5.2.4 EF_ADDR Register

The EF_ADDR register is a 5-bit register used as the word address during eFuse access. The Address must not be changed while an eFuse command is active.

Bit	Reset	Description
4:0	0x00	eFuse Address

5.2.5 EF_WDATA Register

The EF_WDATA register is a 32-bit register used for eFuse write operations. The Write data must not be changed while an eFuse command is active.

This data is also used as the read-test-data during a read operation. For normal reads, the ER_WDATA registers should be set to 0. When bits in this registers are set to non-zero, the eFuse controller will read the matching bit as if it is programmed.

Bit	Reset	Description

31:0	0x0000 0000	eFuse Write Data

5.2.6 EF_RDATA Register

The EF_RDATA register is a 32-bit register that contains the result from an eFuse read operation.

Bit	Reset	Description
31:0	0x0000 0000	eFuse Read Data. Alternatively this returns the HBO frequency counter value, when the HBO_CMD[2] is high.

5.2.7 EF_CMD Register

The EF_CMD register is a 3-bit register used to control eFuse operations. The Start bit must be changed from 0 to 1 to start an operation. The start bit should then be returned to 0, and watch the status for a completed status.

See section 5.3 for details on the proper operation sequences for accessing the eFuse.

Bit	Reset	Description
0	0	Internal nominal 1MHz oscillator enable. This must be enabled before a Start command can be issued.
1	0	eFuse Write/Read selection. 1 is Write, 0 is Read.
2	0	eFuse START Operations

5.2.8 EF_STATUS Register

The EF_STATUS register is a 2-bit register used to sense eFuse operation status.

Bit	Reset	Description
0	0	Status of nominal 1MHz oscillator. Is 1 when the oscillator is running.
1	0	eFuse Controller Busy. 1 is operation in progress.

5.2.9 EF_TIMING Register

The EF_TIMING register is a 25-bit register used to configure eFuse Write timing. Normally the eFuse is timed from the nominal 1MHz clock. If it is configured (by RESET_CFG) to run of the CLK, then these values should be adjusted accordingly.

Bit	Reset	Description
15:0	0x0009	eFuse Write pulse width (in nominal 1MHz clock periods) minus 1. Should be set to >10us.
20:16	0x00	eFuse Write gap width (in nominal 1MHz clock periods) minus 1. Should be set to >100ns.

24:21	0x0	eFuse Read pulse width (in nominal 1MHz clock periods) minus 1. Should be set to >20ns.

5.2.10 SPI_CONFIG Register

The SPI_CIONFIG register is a 24-bit register used to configure SPI Encapsulation operation.

See SPI_ENCAP command 5.2.11 for a description of SPI Encapsulation.

Bit	Reset	Description
15:0	0x0000	SPI Packet size in bits
23:16	0x00	SPI Leading bit count

5.2.11 SPI_ENCAP Register

The SPI_ENCAP and SPI_CONFIG registers are used for SPI Encapsulation (or tunneling). This allows SPI commands to be used to interact with the SPI ROM interface by encapsulating them in JTAG commands.

The SPI_ENCAP instruction does not have a standard JTAG register, but it is used to generate an SPI packet on the ROM SPI Boot interface (with the ROM interface acting as the Master). This allows interaction with the external SPI ROM via the JTAG port.

The external ChipSelect (ROM_CSn) is set active (low) while this instruction is active and the JTAG controller is in the SHIFT_DR state. There is a one-bit register, similar to a by-pass register that is inserted on the TDI data before it goes out the ROM_SO pin. Similarly, there is a one-bit register on the ROM_SI pin data before it goes out TDO. These two registers stage the data to reduce the long timing paths directly through the chip.

The SPI_CONFIG register is used to determine when during the SHIFT_DR state, the TCK is passed through to the ROM_SCK (SPI Clock) pin. Since other JTAG devices (in bypass mode) could be in the TDI/TDO chain, the SPI Leading bit count specifies the number of bypass bits in the chain before this controller. Note that the ARM® Cortex®-M0 debug port is in the chain before this controller; so the SPI Leading bit count would usually be set to one (if no other devices are in the chain before it). The SPI Leading bit count should not include the built-in bit (between TDI and ROM_SO), it is accounted for by the hardware. Application software needs to account for these extra bits when sending JTAG data. The SPI Packet count is used to specify the total number of clocks generated on the ROM_SCK pin (from the TCK pin) after the leading bit count has been satisfied. Any additional clocks after the SPI Packet count would be used to keep shifting data out the TDI/TDO JTAG chain, but won't generate ROM_SCK clocks.

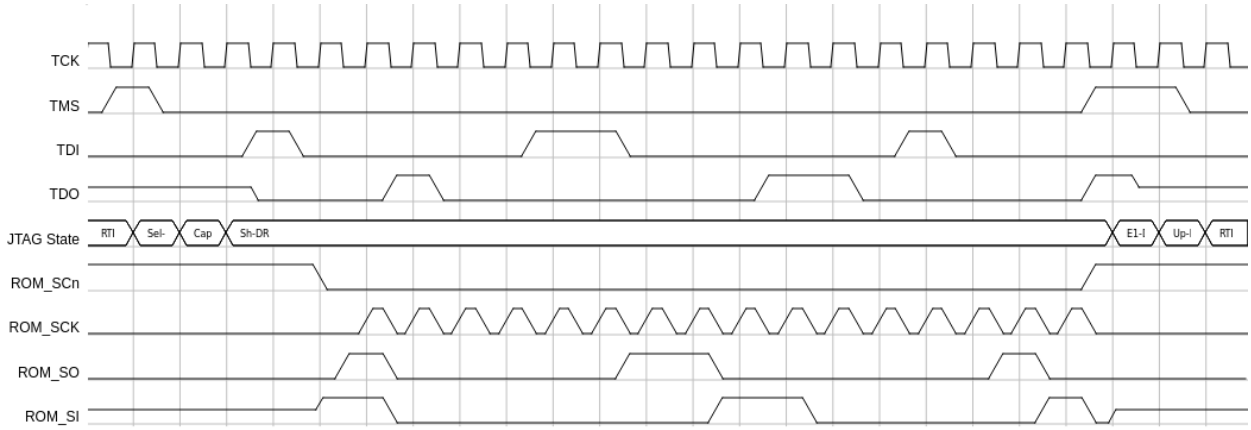
Note that JTAG data is considered to be sent Least-Significant-Bit first and the SPI data is considered to be sent Most-Significant-Bit first. The encapsulation hardware does not change the bit order. The SPI bit order needs to be accounted for when generating the JTAG data stream.

Shown below is a sample JTAG transaction and the related SPI transaction. The assumed starting state of the JTAG controller is as follows:

- JTAG Controllers in state: Run-Test/Idle
- JTAG Instruction register is loaded with: SPI_ENCAP
- ARM® Cortex®-M0 Controller loaded with: BYPASS
- SPI_CONFIG loaded with 0x010010 (Leading bit Count:1, Packet size: 16)

The SPI transaction will be as follows:

- 16 bit long
- Master transmit value: 0x8302 (SPI bit order, MSB first)
- Slave transmit value: 0x80c1 (SPI bit order, MSB first)



5.2.1 HBO_CMD Register

The HBO_CMD register is a 3-bit register used to control HBO frequency counter operations. The HBO frequency counter can be used to measure the frequency of the internal nominal 1MHz oscillator clock.

To measure the internal nominal 1MHz oscillator clock frequency release the Counter reset (setting bit 0 to 1). Then start the counters by setting bit 1 to 1. Then release the start bit (setting bit 1 to 0). Once started, the TCK clock is counted (in a 20 bit counter) until it reaches the limit value (taken from EF_TIMING[19:0]). The internal nominal 1MHz oscillator clock is

counted in a (16 bit counter) until the TCK clock count reaches its limit. The status of the counters being busy is given in bit 0 of the HBO_STATUS register.

The HBO Counter value is tested to see if it is >= EF_WDATA[15:0] and <=ER_WDATA[31:16]; if in this range, the HBO_STATUS bit 1 will report as 1. When HBO_CMD[2] is 1, the final value of the HBO Counter can be read from the EF_RDATA register.

Bit	Reset	Description
0	0	HBO Counter Resetn value. Counters are held in reset while this signal is low
1	0	HBO Counter Start. Setting this to 1 start the HBO/TCK counters.
2	0	HBO Counter value can be read on EF_RDATA when this bit is 1.

5.2.2 HBO_STATUS Register

The HBO_STATUS register is a 2-bit register used to sense eFuse operation status.

Bit	Reset	Description
0	0	HBO/TCK counters are active when this is 1.
1	0	HBO Counter compare value. This is 1 when the counter is not active, and the HBO counter value is >= EF_WDATA[15:0] and <=ER_WDATA[31:16].

5.3 eFuse

The internal eFuse is used to store default configuration data and part ID data. This data is read from the eFuse following a reset event before the SPI ROM data is loaded.

The eFuse data is organized as 32 words of 32-bits of data. All bits in the eFuse are initially zero before programming. Programming can be used to change individual bits to one (but never back to zero). The 32 words provide 32 addressable locations in the eFuse memory. Data is read from the eFuse after a reset. Address locations 0 and 1 contain an index of where to find the configuration data and ID data in the rest of the address space (addresses 2 to 31).

Both configuration data and ID data have 32-bits.

Addresses are used as follows:

Address	Description
0	ID Index. This encodes the address of where the ID data is stored.
1	Configuration index. This encodes the address of where the Configuration data is stored.

2-31	ID or CONFIG data as determined by the related index.

Data words at address 0 and 1 are used as follows:

Bit	Example Value	Description
31:0		Used as a 32-bit priority encoded value of the address+2. Each bit position corresponds to 1 address location. Starting from bit 29 the first bit with a 1 in it corresponds to the address+2 of the location to use for the data. If bits 31 or 30 are set to 1, then this is the same as if all of bits 29:0 are 0, and no data is read (Once bits 31 or 30 are set to 1, the part can no longer be programmed).
	0x00000001	The valid data is at address 2
	0x00000003	The valid data is at address 3
	0x00000007	The valid data is at address 4
	0x1fffffff	The valid data is at address 30
	0x3fffffff	The valid data is at address 31

The bits of the configuration data word are defined as follows:

Bit	Value	Description
25:0		Default values for BOOT_CFG[25:0]. Supplies the boot configuration settings.
26		Reserved
30:27		Default values for MEM_MRG[3:0]. Supplies the memory margin values.
31		Signifies bits 30:0 are valid and should be used.

As the addresses index uses a priority encoding, the lowest addresses should always be used first in programming the eFuse. When it is desired to replace the current configuration data, the next available address location should be used, and the index updated accordingly.

Note that updating the eFuse requires that the index values be read to find the next available address. Then the data needs to be written, and the index needs to be updated. Data should always be read after it was written to insure that it was written correctly.

Before programming configuration settings, these settings should be verified by loading the equivalent values into the proper JTAG registers and resetting the part with JTAG commands. This can validate that the chosen values will work correctly in the system.

5.3.1 eFuse read procedure

- Load EF_WDATA register with 0 (Clear Test Read Data)
- Load EF_ADDR register with the desired read address (Load Address)
- Load EF_CMD registers with 0x1 (Interface Enable)
- Poll/Read EF_STATUS for Bit 0 being 1 (Oscillator is running)
- Load EF_CMD registers with 0x5 (Issue Read Command)
- Read EF_RDATA register (Result of EFuse read)
- Load EF_CMD registers with 0x0 (Interface Disable)

5.3.2 eFuse write procedure

- Load EF_ADDR register with the desired write address (Load Address)
- Load EF_WDATA register with data (Load Data)
- Load EF_TIMING register with proper timing data
- Load EF_CMD registers with 0x3 (Interface Enable with Write Mode)
- Poll/Read EF_STATUS for Bit 0 being 1 (Oscillator is running)
- Bring EFUSE_BURN_ENn pin low (Enable Write)
- Load EF_CMD registers with 0x7 (Issue Write Command)
- Poll/Read EF_STATUS register for Bit 1 being 0 (Check/Wait for write complete)
- Bring EFUSE_BURN_ENn pin high (Disable Write)
- Load EF_CMD registers with 0x0 (Interface Disable)
- Do eFuse Read procedure to verify result

5.3.3 Sample eFuse first time write procedure

- Read all the address locations in the eFuse to verify that it has not been programmed.
- Write the desired ID data to address 2
- Write the value 0x0000 0001 to address 0 (signifies that ID data can be found at address 2)
- Write the desired Configuration data to address 3
- Write the value 0x0000 0003 to address 1 (signifies that Configuration data can be found at address 3)

5.3.4 Sample eFuse second time write procedure

- Read all the address locations in the eFuse. Verify that address locations 4-31 are 0x0000 0000 and that locations 0 and 1 have values below 0x000 0004.
- Write the desired ID data to address 4
- Write the value 0x0000 0007 to address 0 (signifies that ID data is now found at address 4)
- Write the desired Configuration data to address 5

- Write the value 0x0000 000f to address 1 (signifies that Configuration data is now found at address 5)

6 Porting Notes

This section summarizes what is different between the VA108X0 part and the PA32KAS part.

6.1 Memory System

- In EDAC mode, there are unique syndrome bits for each 8-bit byte in a 32-bit address word
- The ROM (Code Memory) is increased to 128K bytes
- The RAM (Data Memory) is increased to 32K bytes

6.2 System Configuration Peripheral

- The previous RAM_TRAP and ROM_TRAP registers, have been split into RAM_TRAP_ADDR, RAM_TRAP_SYND, ROM_TRAP_ADDR, and ROM_TRAP_SYND.
- Added REFRESH_CONFIG, TIM_RESET, TIM_CLK_ENABLE, PERIPHERAL_RESET, and PERIPHERAL_CLK_ENABLE registers.
- RST_CNTRL_ROM and RST_CNTRL_RAM include control bit for Memory Error Reset

6.3 IRQ Selector Peripheral

- GPIO[0-31] replaced with PORTA[0-31] and PORTB[0-11]
- TIM[0-31] reduced to TIM[0-23]
- SBM_* removed
- RAM_* and ROM_* renamed INT_RAM_* and INT_ROM_*
- TXEV_TXIM renamed TXEV
- UART[*].RX and UART[*].TX combined to UART[*]
- SPI[*].* combined to SPI[*]
- Added I2C_MS[*] and I2C_SL[*]
- Added MERESET configuration register

6.4 IO Configuration Peripheral

- All names replaced to match current IO names: PORTA*, and PORTB*
- Additional configuration bits added to registers:
    - Bit 12 – PWOA
    - Bits 14:13 – FUNSEL

6.5 Utility Peripheral

- New peripheral

6.6 General Purpose IO Peripheral

- Significantly different peripheral

- Banks changed from 4 banks of size 8 to 2 banks with sizes 32 and 24
- Split DATA into DATAIN and DATAOUT registers
- Data register in no longer in a memory mapped address space. To write to specific bits a DATAMASK register is provided.
- Alternate access to the DATAOUT register is provided with SETOUT, CLROUT, and TOGOUT.
- Added registers: PULSE, PULSEBASE, DELAY1 and DELAY2
- Removed TIM_SEL register (this function is replaced by the FUNSEL in the IO Configuration Peripheral)

6.7 Timer/Counter Peripheral

- The number of counters was reduced from 32 to 24
- CTRL register has extra bits for Status output selection
  - New modes involving PMWA and PWMB values
- CTRL register has a new REQ_STOP bit
- CSD_CNTRL registers has new bits:
  - CSDEN2, CSDTRG1 and CSDINV2
  - Cascade0/1 trigger mode
- New CASCADE2 register
- PWM_VALUE register renamed to PWMA_VALUE (old name will still work as well)
- New PWMB_VALUE register

6.8 UART Peripheral

- Significantly different peripheral
- STATUS register replaced by RXSTATUS and TXSTATUS
- Added ENABLE register
- CNTRL changed to configure word size, parity, stop, and loopback
- Added IRQ_ENB, IRQ_RAW, IRQ_END, and IRQ_CLR registers
- Added CLRSTAT register
- Added TXBREAK register
- Added RXFIFOIRQTRG and TXFIFOIRQTRG registers

6.9 SPI Peripheral

- Added a third bank, that is a master only controller to access the SPI Boot ROM interface pins
- Added RXFIFOIRQTRG, TXFIFOIRQTRG and FIFO_CLR registers
- Added bits to the STATUS register: RXDATAFIRST, RXTRIGGER, and TXTRIGGER

- Added bits to CTRL1: BMSTART and BMSTALL
- Added bit to DATA: BMSTART/BMSTOP

6.10 I²C Peripheral

- New Peripheral

7 Revision History

Version	Date	Description
1	2/18/2015	Initial Release Revision
1.01	3/5/2015	Minor updates to Feature Summary section. Added description of Interrupt Status registers in IRQ Selector. Updated TIM description of ENABLE and ACTIVE concepts. Updated UART to include setup description. Updated SPI description to include bit order. Updated SPI to include setup description. Updated JTAG-SPI_ENCAP mode to point out the difference in bit ordering between JTAG and SPI. Updated each peripheral with a description of its clock gating behavior.
1.02	6/17/2015	Updated Param Sig mode description. Updated JTag SPI Encapsulation description.
1.03	7/10/2015	Updated JTag command: EF_TIMING Added JTag commands: HBO_CMD, HBO_STATUS
1.04	8/3/2015	Updated Power-Up Sequence description Updated EFuse Read/Write procedure description Updated IO Configuration Function Selection description
1.05		Update Names column in TIM Group table to properly show 24 counters. Updated description of the Utility Peripheral.
1.06	9/2/2015	Updates for reviewers. Cleanup of descriptions and reset values.
1.07	9/16/2015	Style updates for Vorago template
1.08	10/8/2015	Updated Block Diagram to show Timer/Counters to GPIOs. Corrected Memory Map to show 3 SPI ports.
1.09	10/21/2015	Updated part numbers: PA03CSA -> PA21N PA04DNA -> PA22S
1.10	10/28/2015	Diagram updates
1.11	12/16/2015	Updated part numbers to VA10800 and VA10820
1.12	2/24/2016	Numerous spelling and abbreviation changes. Mhz -> MHz, RTC -> TIM, EANBLE -> ENABLE, etc. Corrected RTC block diagram with cascade 2 labels.
1.13	3/30/2016	SPI Status register RESET values updated to match device operation.

1.14	4/30/2016	• IOCONFIG replace IOMGR in clock and reset registers • New company address on last page • Removed reference in 5.2.13 JTAG section to unavailable documents. • Removed "company proprietary" in footer • Removed "Preliminary" Watermark
1.15	6/10/2016	• Updates to JTAG and eFuse chapter. Removed Vorago only information on test modes and clarified boot configuration information. • Adding details to EF_CONFIG and EF_ID register descriptions.
1.16	8/2/2016	• Timer chapter updates – additional information and new example section. • IRQ select chapter – add diagram
1.17	11/21/2016	• Corrected cross reference error in 5.2.7 and 8.2.22 • Consolidated EFUSE_BURN_ENn and EFUSE_BURN_ENn to EFUSE_BURN_ENn to match DS. • eFuse capitalization standardized

1.18	2/4/2017	• CASTRG2 description updated
		• EF_CONFIG and EF_ID descriptions updated.
		• TIM bit name changes
		  ○ AUTO_DEACTIVE > AUTO_DEACTIVATE
		  ○ CASxyz > CSDxyz (CSD_CTRL register)
		• Added note to PERIPHERAL_CLK_ENABLE Register

1.19	11/13/2018	• UART CLKSCALE register. Removed bit 31 RESET definition. • UART FIFO_CLR register. Removed RXSTS and TXSTS bits. Shifted other bits to positions 1 & 0. • UART IRQ_ENB register bit 3 removed. • UART STATE register bit numbering corrected • I2C IRQ_CLR register – clarified which bits can be cleared • I2C WORDS register – clarified that continuous receive operation is not supported. • GPIO EDGE_STATUS register – Register is RW with bits cleared by write operation.
1.20	6/24/2020	• Removed Section 6.6 External Memory Peripheral • Corrected reset values in Table 30 – SPI Registers

		• Added VA10830 and corresponding information concerning internal NVM device

VA108X0

VORAGO Technologies
1501 S MoPac Expressway, Suite 350,
Austin, TX 78746
www.voragotech.com

Email: info@voragotech.com
Phone: (512) 347-1800

8 JTAG controller and eFuse block

The VA10800/VA10820 ARM® Cortex®-M0 Processor chip is designed and built with specific test related features. These features can be used to test features in the devices that are not accessible through normal functional operation of the part. Most of these test features are accessed using the IEEE 1149.1 standard test port (JTAG).

The JTAG port is also used to read and program the on-chip eFuse block that controls the power-up sequence of the part. The eFuse can be programmed up to 30 times. Each of these 30 programming cycles can change the boot configuration or a unique device ID number.

The JTAG port can also be used to send encapsulated SPI commands to access an SPI ROM attached to the devices SPI ROM Boot port. This can be used to do in-system programming of the SPI ROM through the JTAG port without adding additional logic to the board to provide access to the SPI ROM.

The JTAG port on the part consists of pins: TCK, TMS, TRSTn, TDI, and TDO.

Internal to the part there are 2 JTAG controllers. These 2 controllers are connected in series such that the TDO of the first controller is connected to TDI of the second controller. The first controller (connected to the TDI pin) is used to access the debug port of the ARM® Cortex®-M0 processor. The second controller in the sequence is the controller used to access chip level configuration and test.

8.1 ARM® Cortex®-M0 JTAG Controller

The ARM® Cortex®-M0 JTAG controller is the standard Debug controller from ARM®. It is used to access the debug port on the ARM® Cortex®-M0.

One special note, when the chip level JTAG controller is used to enable SCAN mode, the ARM® controller is removed from the TDI/TDO path. The part acts as if only one JTAG controller exists at that point. This is done so that the ARM® controller can be switched to run off the system clock and be included in scan testing. There is 1-bit flip-flop added in the TDI/TDO path when in this mode; so that it acts like a normal BYPASS register still exists for the missing controller. After exiting SCAN mode, the TRSTn should be asserted to reset the JTAG controllers; as the ARM® controller will most likely be in an unknown state.

8.1.1 Instruction Codes

The ARM® controller contains a 4-bit instruction register. The implemented instruction codes are listed below:

Instruction Register Code	Instruction Name	Description
4'b1000	ABORT	Select Abort Chain
4'b1010	DPACC	Selects DP Chain
4'b1011	APACC	Selects AP Chain
4'b1110	IDCODE	Standard JTAG Instruction
4'b1111	BYPASS	Standard JTAG Instruction

8.2 Chip Level JTAG Controller

The chip level JTAG controller provides access to the following features:

- IEEE 1149.1 Boundary Scan
- Built-In Self-Test (BIST) of internal memories
- Scan Testing
- IDDQ Testing
- Parametric NAND Tree
- Power-On-Reset testing
- Reduced default ROM startup delay
- Reduced default ROM size loading
- Device resets
- Internal eFuse access
- SPI ROM access

8.2.1 Instruction Codes

The chip level JTAG controller contains a 5-bit instruction register. The implemented instruction codes are listed below:

Instruction Register Code	Instruction Name	Description
5'h00	ALTBYPASS	Implemented as BYPASS
5'h01	EXTEST	Standard JTAG Instruction
5'h02	CLAMP	Standard JTAG Instruction
5'h03	HIGHZ	Standard JTAG Instruction
5'h04	SAMPLE/PRELOAD	Standard JTAG Instruction
5'h05	PULL	Pull Mode
5'h06	IDCODE	Standard JTAG Instruction
5'h07	PULLEN	Pull Enable Register

5'h08	TEST_CFG	Test Configuration Register
5'h09	CLK_CFG	Clock Configuration Register
5'h0a	BIST_MODE	BIST Mode Register
5'h0b	BIST_RST	BIST Reset
5'h0c	BIST_CFG	BIST Configuration Register
5'h0d	BIST_STAT	BIST Status Register
5'h0e	BIST_RUN	BIST Run
5'h0f	MEM_MRG	Memory Margin Register
5'h10	BOOT_CFG	Boot Configuration Register
5'h11	RESET_CFG	Reset Configuration Register
5'h12		Reserved
5'h13		Reserved
5'h14	EF_ADDR	eFuse Address Register
5'h15	EF_WDATA	eFuse Write Data Register
5'h16	EF_RDATA	eFuse Read Data Register
5'h17	ER_TIMING	eFuse Timing Register
5'h18	EF_CMD	eFuse Command Register
5'h19	EF_STATUS	eFuse Status Register
5'h1a	SPI_CONFIG	SPI Encapsulation Configuration Register
5'h1b	SPI_ENCAP	SPI Encapsulation Register
5'h1c		Reserved
5'h1d	HBO_CMD	HBO Command Register
5'h1e	HBO_STATUS	HBO Status Register
5'h1f	BYPASS	Standard JTAG Instruction

8.2.2 Boundary Scan Pins

All device level pins except those listed below are included in Boundary Scan:

- JTAG Pins: TCK, TMS, TRSTn, TDI, and TDO
- Supply Pins: VSS, VDDC, DVDD
- eFuse Pin: EFUSE_BURN_ENn

8.2.3 EXTEST Mode

This is a standard Boundary Scan mode. The output pins are driven from the value in the boundary scan register. The boundary scan register is selected as the shift path in this mode.

The order of pins in the boundary scan register is given below from TDI to TDO order:

Pins	Bits	Bit Order per Pin
CLK	1	DataIn

PORTA[19] – PORTA[23]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[17] – PORTB[19]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTA[24] – PORTA[31]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[0] – PORTB[8]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[20] – PORTB[23]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[9]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
ROM_SCK	2	DataOut, DataOutEnable[1]
ROM_CSn	2	DataOut, DataOutEnable[1]
ROM_SO	2	DataOut, DataOutEnable[1]
ROM_SI	1	DataIn
EXTRESETn	1	DataIn
I2CA_SCL	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
I2CA_SDA	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
I2CB_SCL	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
I2CB_SDA	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTA[0] – PORTA[1]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[10] – PORTB[13]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTA[2] – PORTA[18]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
PORTB[14] – PORTB[16]	3 bits for each pin	DataIn, DataOut, DataOutEnable[1]
Total:	**187**	

Notes:

1. DataOutEnable is the enable for the DataOut and is active low.

8.2.4 CLAMP Mode

This is a standard Boundary Scan mode. The output pins are driven from the value in the boundary scan register. The BYPASS register is selected as the shift path in this mode.

8.2.5 HIGHZ Mode

This is a standard Boundary Scan mode. The output pins are all placed in a HighZ state. The BYPASS register is selected as the shift path in this mode.

8.2.6 SAMPLE/PRELOAD Mode

This is a standard Boundary Scan mode. The output pins are in normal functional mode. The current value on the pins is captured in the boundary scan register during the CAPTURE_DR state. The boundary scan output register is updated on the UPDATE_DR state. The boundary scan register is selected as the shift path in this mode.

8.2.7 PULL Mode

This mode is similar to the normal Boundary Scan HIGHZ mode. All output pins are placed in the HIGHZ state. Pins that support a configurable Pull-up or Pull-down resistor will have the pull resistor enabled while in this state. The direction of the pull will be determined by the value of the boundary-scan register (similar to CLAMP mode). The BYPASS register is selected in this mode.

8.2.8 PULLEN Register

The PULLEN register is a 1-bit register defined below.

Bit	Reset State	Description
0	1	When set to 0, this disables normal operation of pull-up and pull-down resisters from their internal control logic. When set to 1, this enables normal operation of pull-up and pull-down resisters from their internal control logic. Note: When TRSTn is active (low), or the controller is in Test-Logic-Reset, then pull-up and pull-down resistors are enabled by default.

8.2.9 TEST_CFG Register

The TEST_CFG register is a 5-bit register that configures the test features as defined below:

Bit	Reset State	Value	Description
4:0	00000		Mode Selection
		00000	No special mode enabled
		00001	IDDQ mode.
		00010	Parametric mode
		00011	PARAM_SIG mode for DSTPOR
		00100	PARAM_SIG mode for TCK
		00101	PARAM_SIG mode for TRSTn
		00110	PARAM_SIG mode for TMS

		00111	PARAM_SIG mode for TDI
		01000	PORMON mode
		01001	I2CMON mode
		01010	Reserved
		01011	HBOMON mode
		011xx	Reserved
		1xxxx	SCAN mode

8.2.9.1 IDDQ Mode

In this mode the ROM_SI pin is configured as an IDDQ enable. When IDDQ is active, all the chip output pins will be disabled, and the input pins (except TCK, TRSTn, and ROM_SI) will be disabled.

8.2.9.2 SCAN Mode

In this mode the ROM_SI pin is configured as a SCAN_EN pin.

When SCAN_EN is active, the chip will be in scan shift mode. Pins PORTA[15:0] will be in input mode and will be used as inputs to the 16 scan chains. Pins PORTA[31:16] will be in output mode and will be used as outputs from the 16 scan chains.

When SCAN_EN is inactive, (but SCAN mode is active), the part will clock logic as in normal functional mode, but with the following exceptions:

- The internal memories will be in Test Data Bypass mode, which by-passes normal operation.
- Special scan observation and control flops are enabled.
- Software configurable pull-downs on GPIO pins are all disabled.
- Software configurable pull-ups on GPIO pins are all enabled.
- I^2C input/outputs are disabled.
- The eFuse block is in by-pass mode.

8.2.9.3 Parametric Mode

In this mode the parametric nand tree is enabled. When this is enabled the output of the parametric nand trees is connected to the ROM_SO pin. The parametric nand tree connects all the Non-JTAG input and bidirectional pins in a chain.

The order of the Parametric nand tree is given in the following table. The order is from the pin farthest from the ROM_SO pin, to the pin closest to the ROM_SO pin.

Pins	Bits
CLK	1
PORTA[19] – PORTA[23]	1 bit for each pin
PORTB[17] – PORTB[19]	1 bit for each pin
PORTA[24] – PORTA[31]	1 bit for each pin
PORTB[0] – PORTB[8]	1 bit for each pin
PORTB[20] – PORTB[23]	1 bit for each pin
PORTB[9]	1 bit for each pin
ROM_SI	1
EXTRESETn	1
I2CA_SCL	1 bit for each pin
I2CA_SDA	1 bit for each pin
I2CB_SCL	1 bit for each pin
I2CB_SDA	1 bit for each pin
PORTA[0] – PORTA[1]	1 bit for each pin
PORTB[10] – PORTB[13]	1 bit for each pin
PORTA[2] – PORTA[18]	1 bit for each pin
PORTB[14] – PORTB[16]	1 bit for each pin
Total:	**62**

8.2.9.4 PARAM_SIG Modes

In this mode the selected TEST input pin is configured to drive the ROM_SO pin. There is a mode to enable each of the following pins:

- DSTPOR
- TCK
- TRSTn
- TMS
- TDI

8.2.9.5 PORMON Mode

In this mode the status of the internal Power-On-Reset logic can be monitored on the PORTB pins.

The mode monitor status is defined below:

PortB Bit	Description
2:0	POR33n –Observation of the 3 POR blocks on the 3.3v supply.
5:3	POR15n –Observation of the 3 POR blocks on the 1.5v supply.
8:6	PORGENn –Observation of the 3 POR generate signals. This is a combination of the POR33n, POR15n, and POR trigger from the TAP controller

11:9	OBS_POR33– Observation Flops on the 3.3V PORs. These are asynchronously reset by the PORs, and set on rising CLK.
14:12	OBS_POR15 – Observation Flops on the 1.5V PORs. These are asynchronously reset by the PORs, and set on rising CLK.
17:15	SYNC_PORn – Output of the 4 stage Synchronizer for each of the TMR POR detectors. These are asynchronously reset by the PORs. Each stage of this synchronizer will be set by subsequent rising CLK edges
18	SYNC_PORESETn – Voted version of SYNC_PORn
19	Reserved – Currently Reads as 0

8.2.9.6 I2CMON Mode

In this mode the I2C IO cells can be monitored and controlled on the PORTB pins.

The mode monitor status is defined below:

PortB Bit	Direction	Description
3:0	Output	Monitor of value received on pins: I2CB_SDA, I2CB_SCL, I2CA_SDA, I2CA_SCL directly from the input buffer.
7:4	Output	Monitor of value received on pins: I2CB_SDA, I2CB_SCL, I2CA_SDA, I2CA_SCL after the input digital glitch filter.
17:8	Outputs	Reserved – Currently Reads as 0
21:18	Inputs	Values to output on pins: I2CB_SDA, I2CB_SCL, I2CA_SDA, I2CA_SCL.
22	Inputs	Enable for bits 7:4, 1=> Use bits 7:4 as pin values, 0=> Use internal values.
23	Inputs	Enable FastRx mode of input receiver

8.2.9.7 HBOMON Mode

In this mode the nominal 1MHz oscillator can be monitored and controlled on the PORTB pins.

The mode monitor status is defined below:

PortB Bit	Direction	Description
0	Output	Oscillator output
1	Output	Oscillator complement output
2	Output	Oscillator enabled output
3	Output	Oscillator detector output
4	Output	Oscillator detector complement output
5	Output	Oscillator buffered output
6	Output	Oscillator enable observation

7	Output	Clock Select observation
8	Output	Oscillator disable observation
9	Output	End Power-On-Sequence observation
10	Output	eFuse Read observation
11	Output	eFuse Busy observation
12	Output	eFuse Reset observation
13	Output	Boot Delay Done [0] observation
14	Output	Boot Delay Done [1] observation
15	Output	Boot Delay Done [2] observation
17:16	Output	Reserved – Currently Reads as 0
18	Input	Oscillator EN_OSC_N control
19	Input	Oscillator IDAC[0] control
20	Input	Oscillator IDAC [1] control
21	Input	Oscillator TST_CLK_N control
22	Input	Oscillator TST_DET control
24	Input	Reserved

8.2.10 CLK_CFG Register

The CLK_CFG register is a 3-bit register that allows switching of the internal clock to be from CLK or TCK as defined below.

Bit	Reset State	Description
0	0	When set to 1, the internal clock to the chip is taken from TCK instead of CLK.
1	0	Select the control mechanism used to switch the clocks. This bit should be set before changing bit 0. 0 – Direct Mux switching. The clocks are directly switched. This should be done synchronously (with both TCK and CLK off) to avoid problems. 1 – Synchronous clock switching. This requires that both clocks be running. At least one full clock pulse on each clock after switching the clock source is required before the clock will switch.
2	0	When set to 1, the internal PhasesReset signals are all enabled to be 1 at the same time.

The synchronous clock switching logic is shown below:

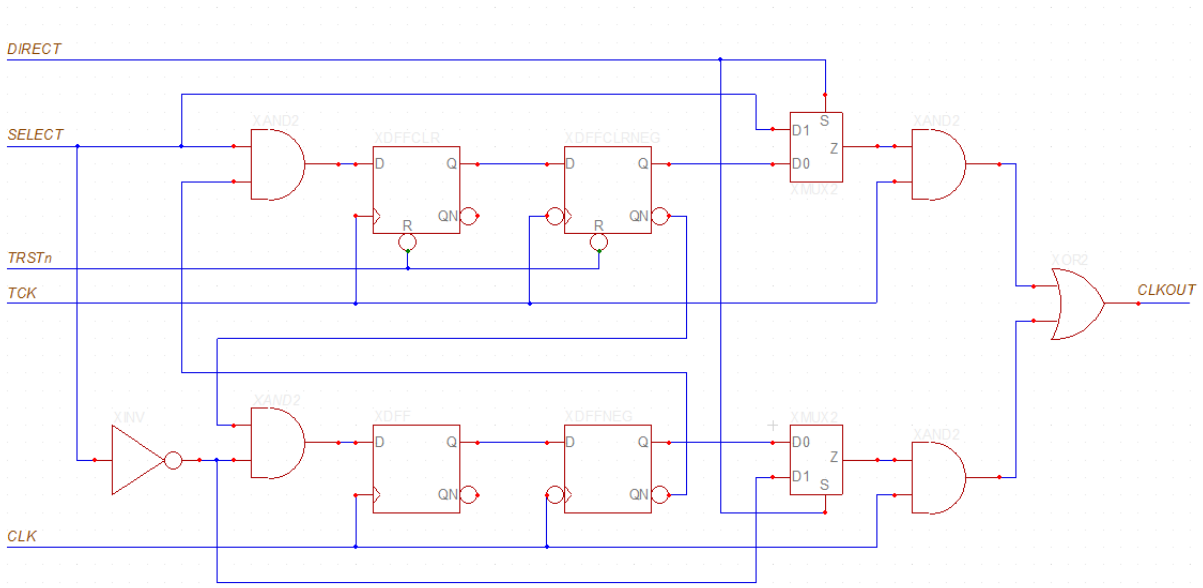


Figure 17 - Synchronous Clock Switching

8.2.11 BIST_MODE Register

The BIST_MODE register is a 1-bit register defined below.

Bit	Reset State	Description
0	0	When set to 1, the memories are placed in BIST mode. This enables clocking of the BIST logic in the memory BIST wrappers, and disables normal functional access.

8.2.12 BIST_RST

The BIST_RST instruction does not access a register. While this instruction is active (loaded into the JTAG controller instruction register), a reset signal is issued to the BIST logic; which does the following:

- Resets the BIST_STAT register
- Resets the BIST_CFG register
- Disables the clock to the BIST logic in the memory BIST wrappers

8.2.13 BIST_CFG Register

The BIST_CFG register provides access to the serial BIST configuration register in the memory BIST wrappers. This is a non-standard JTAG data register in that it does not implement capture or update operations. Data is directly shifted into the memory BIST configuration

register during the JTAG shift operation. Note that there is one-bit dummy register added to the beginning of the BIST scan chain. [There is also a 1-bit BYPASS register from the ARM® M0 Debug controller].

8.2.14 BIST_STAT Register

The BIST_STAT register is a 2-bit register defined below.

Bit	Name	Description
0	BIST_READY	Status indication of BIST completion. A 1 value indicates that BIST has finished.
1	BIST_PASSFAIL	Status indication of the Pass/Fail status of the last BIST operation. A 1 value indicates Pass, and a 0 value indicates Fail.

8.2.15 BIST_RUN

The BIST_RUN instruction does not access a register. While this instruction is active (loaded into the JTAG controller instruction register), the BIST start operation is enabled; this starts the configured BIST test. Once the BIST_RUN has been started, the normal practice is to load the BIST_STAT instruction and monitor the status until BIST_READY indicates that the run is completed.

8.2.16 MEM_MRG Register

The MEM_MGR register is a 5-bit register that controls the memory margin pins of the internal memories. These bits are only used for testing purposes and should not be changed by the user.

Bit	Reset State	Description
0	0	WM
1	0	RM
2	1	RWM
3	0	OLM
4	0	ENABLE – Enables the memory Overrides. When this bit is 0, the values in bits 3-0 are ignored. When this bit is 1, the value in bits 3-0 are applied to the memories.

8.2.17 BOOT_CFG Register

The BOOT_CFG register is a 27-bit register defined below.

Bit	Reset State	Name	Description
1:0	0x1	ROM_SPEED	This value specifies the speed of ROM_SCK: 0 – CLK divide by 2 (@50MHz => 25MHz) 1 – CLK divide by 6 (@50MHz => 8.33MHz) 2 – CLK divide by 12 (@50MHz => 4.2MHz) 3 – CLK divide by 52 (@50MHz => 962kHz)
4:2	0x0	ROM_SIZE	This value specifies how much of the full 128K byte address space is loaded from the external SPI memory after a reset. The values are: 0 – Full 128k Bytes of address space 1 – First 64K bytes of address space 2 – First 32K bytes of address space 3 – First 16K bytes of address space 4 – First 8K bytes of address space 5 – First 4K bytes of address space The part of the address space not loaded is initialized to zero.
5	0	ROM_NOCHECK	When set to 1, the ROM check is skipped. When set to 0, the ROM check is enabled. In ROM check mode, each 128 byte block of data from the SPI ROM is read twice. If the two reads get different results, the process is repeated until it read the same twice.
8:6	0x4	BOOT_DELAY	This value specifies the boot delay from the end of the Power-On-Sequence to the release of Reset. The delay is based on cycles of the internal nominal 1MHz oscillator. 0 – 1 cycles (~ 0ms) 1 – 1000 cycles (~1ms) 2 – 3000 cycles (~3ms) 3 – 10000 cycles (~10ms) 4 – 30000 cycles (~30ms) 5 – 100000 cycles (~100ms) 6 – 300000 cycles (~300ms) 7 – 500000 cycles (~500ms)
16:9	0x03	ROM_READ	SPI ROM read instruction code.
21:17	0	ROM_LATENCY	Number of bits of latency from Address until data from the SPI ROM.

23:22	1	ROM_ADDRESS	Rom Address Mode. Specifies the number of bits/bytes to use for addressing the SPI ROM. 0 – 16 bit / 2 bytes 1 – 24 bit / 3 bytes 2 – 32 bit / 4 bytes
24	1	ROM_DLYCAP	ROM SPI Delayed capture. When 1, the ROM_SI data is captured on clock falling edge instead of the normal SPI mode of the rising edge. This allows almost a full SCK clock cycle for the SPI ROM to respond from address to data. This mode allows a faster SPI cycle time.
25	0	ROM_STATUS	In this mode, the first data byte from the SPI ROM following an address is taken as a status byte. A Zero status data byte indicates valid data follows. A non-zero status data byte indicates the device is busy, and that the SPI transaction should be restarted.
26	0	ENABLE	Enables the config settings. When this bit is 0, the other bits in this register are ignored. When this bit is 1, the other bits in this register are used.

8.2.18 RESET_CFG Register

The RESET_CFG register is a 9-bit register that is used to control internal reset events.

Bit	Reset State	Description
2:0	0x0	While set to 1, the Power-On-Reset is manually activated. There are three bits; 1 bit for each of the 3 POR blocks.
3	0	While set to 1, the EXTRESET is manually activated.
4	0	While set to 1, the internal Power-On-Reset is disabled.
5	0	While set to 1, the EXTRESET reset is disabled.
6	0	While set to 1, the system boot process is disabled.
7	0	While set to 1, the internal nominal 1MHz oscillator is disabled
8	1	While set to 1, the Power-On-Sequence is clocked from the CLK pin instead of the nominal 1MHz oscillator.

8.2.19 EF_ADDR Register

The EF_ADDR register is a 5-bit register used as the word address during eFuse access. The Address must not be changed while an eFuse command is active.

Bit	Reset	Description
4:0	0x00	eFuse Address

8.2.20 EF_WDATA Register

The EF_WDATA register is a 32-bit register used for eFuse write operations. The Write data must not be changed while an eFuse command is active.

This data is also used as the read-test-data during a read operation. For normal reads, the ER_WDATA registers should be set to 0. When bits in this registers are set to non-zero, the eFuse controller will read the matching bit as if it is programmed.

Bit	Reset	Description
31:0	0x0000 0000	eFuse Write Data

8.2.21 EF_RDATA Register

The EF_RDATA register is a 32-bit register that contains the result from an eFuse read operation.

Bit	Reset	Description
31:0	0x0000 0000	eFuse Read Data. Alternatively this returns the HBO frequency counter value, when the HBO_CMD[2] is high.

8.2.22 EF_CMD Register

The EF_CMD register is a 3-bit register used to control eFuse operations. The Start bit must be changed from 0 to 1 to start an operation. The start bit should then be returned to 0, and watch the status for a completed status.

See section 8.3 for details on the proper operation sequences for accessing the eFuse.

Bit	Reset	Description
0	0	Internal nominal 1MHz oscillator enable. This must be enabled before a Start command can be issued.
1	0	eFuse Write/Read selection. 1 is Write, 0 is Read.
2	0	eFuse START Operations

8.2.23 EF_STATUS Register

The EF_STATUS register is a 2-bit register used to sense eFuse operation status.

Bit	Reset	Description
0	0	Status of nominal 1MHz oscillator. Is 1 when the oscillator is running.
1	0	eFuse Controller Busy. 1 is operation in progress.

8.2.24 EF_TIMING Register

The EF_TIMING register is a 25-bit register used to configure eFuse Write timing. Normally the eFuse is timed from the nominal 1MHz clock. If it is configured (by RESET_CFG) to run of the CLK, then these values should be adjusted accordingly.

Bit	Reset	Description
15:0	0x0009	eFuse Write pulse width (in nominal 1MHz clock periods) minus 1. Should be set to >10us.
20:16	0x00	eFuse Write gap width (in nominal 1MHz clock periods) minus 1. Should be set to >100ns.
24:21	0x0	eFuse Read pulse width (in nominal 1MHz clock periods) minus 1. Should be set to >20ns.

8.2.25 SPI_CONFIG Register

The SPI_CIONFIG register is a 24-bit register used to configure SPI Encapsulation operation.

See SPI_ENCAP command 5.2.11 for a description of SPI Encapsulation.

Bit	Reset	Description
15:0	0x0000	SPI Packet size in bits
23:16	0x00	SPI Leading bit count

8.2.26 SPI_ENCAP Register

The SPI_ENCAP and SPI_CONFIG registers are used for SPI Encapsulation (or tunneling). This allows SPI commands to be used to interact with the SPI ROM interface by encapsulating them in JTAG commands.

The SPI_ENCAP instruction does not have a standard JTAG register, but it is used to generate an SPI packet on the ROM SPI Boot interface (with the ROM interface acting as the Master). This allows interaction with the external SPI ROM via the JTAG port.

The external ChipSelect (ROM_CSn) is set active (low) while this instruction is active and the JTAG controller is in the SHIFT_DR state. There is a one-bit register, similar to a by-pass register that is inserted on the TDI data before it goes out the ROM_SO pin. Similarly, there is a one-bit register on the ROM_SI pin data before it goes out TDO. These two registers stage the data to reduce the long timing paths directly through the chip.

The SPI_CONFIG register is used to determine when during the SHIFT_DR state, the TCK is passed through to the ROM_SCK (SPI Clock) pin. Since other JTAG devices (in bypass mode) could be in the TDI/TDO chain, the SPI Leading bit count specifies the number of bypass bits in the chain before this controller. Note that the ARM® Cortex®-M0 debug port is in the chain

before this controller; so the SPI Leading bit count would usually be set to one (if no other devices are in the chain before it). The SPI Leading bit count should not include the built-in bit (between TDI and ROM_SO), it is accounted for by the hardware. Application software needs to account for these extra bits when sending JTAG data. The SPI Packet count is used to specify the total number of clocks generated on the ROM_SCK pin (from the TCK pin) after the leading bit count has been satisfied. Any additional clocks after the SPI Packet count would be used to keep shifting data out the TDI/TDO JTAG chain, but won't generate ROM_SCK clocks.

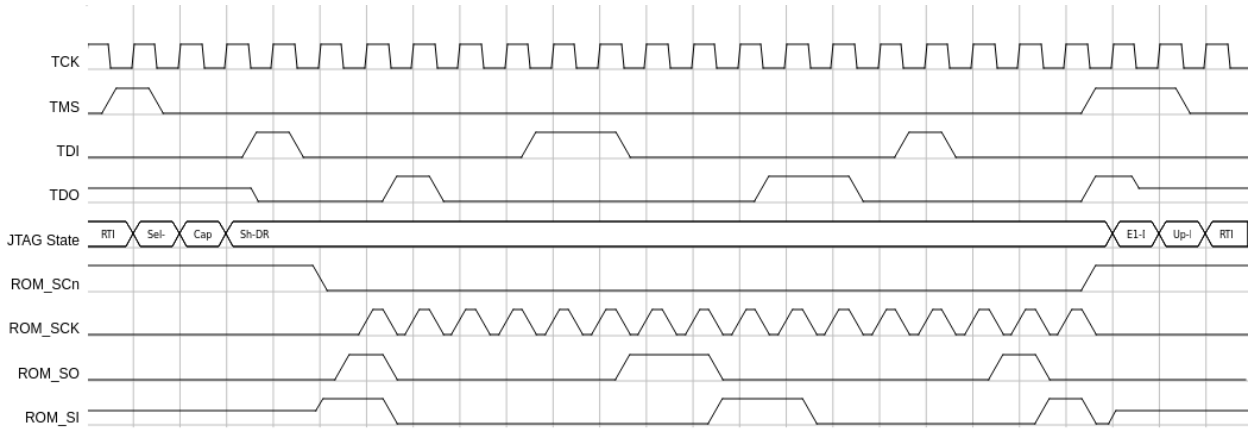
Note that JTAG data is considered to be sent Least-Significant-Bit first and the SPI data is considered to be sent Most-Significant-Bit first. The encapsulation hardware does not change the bit order. The SPI bit order needs to be accounted for when generating the JTAG data stream.

Shown below is a sample JTAG transaction and the related SPI transaction. The assumed starting state of the JTAG controller is as follows:

- JTAG Controllers in state: Run-Test/Idle
- JTAG Instruction register is loaded with: SPI_ENCAP
- ARM® Cortex®-M0 Controller loaded with: BYPASS
- SPI_CONFIG loaded with 0x010010 (Leading bit Count:1, Packet size: 16)

The SPI transaction will be as follows:

- 16 bit long
- Master transmit value: 0x8302 (SPI bit order, MSB first)
- Slave transmit value: 0x80c1 (SPI bit order, MSB first)



8.2.27 HBO_CMD Register

The HBO_CMD register is a 3-bit register used to control HBO frequency counter operations. The HBO frequency counter can be used to measure the frequency of the internal nominal 1MHz oscillator clock.

To measure the internal nominal 1MHz oscillator clock frequency release the Counter reset (setting bit 0 to 1). Then start the counters by setting bit 1 to 1. Then release the start bit (setting bit 1 to 0). Once started, the TCK clock is counted (in a 20 bit counter) until it reaches the limit value (taken from EF_TIMING[19:0]). The internal nominal 1MHz oscillator clock is counted in a (16 bit counter) until the TCK clock count reaches its limit. The status of the counters being busy is given in bit 0 of the HBO_STATUS register.

The HBO Counter value is tested to see if it is >= EF_WDATA[15:0] and <=ER_WDATA[31:16]; if in this range, the HBO_STATUS bit 1 will report as 1. When HBO_CMD[2] is 1, the final value of the HBO Counter can be read from the EF_RDATA register.

Bit	Reset	Description
0	0	HBO Counter Resetn value. Counters are held in reset while this signal is low
1	0	HBO Counter Start. Setting this to 1 start the HBO/TCK counters.
2	0	HBO Counter value can be read on EF_RDATA when this bit is 1.

8.2.28 HBO_STATUS Register

The HBO_STATUS register is a 2-bit register used to sense eFuse operation status.

Bit	Reset	Description
0	0	HBO/TCK counters are active when this is 1.
1	0	HBO Counter compare value. This is 1 when the counter is not active, and the HBO counter value is >= EF_WDATA[15:0] and <=ER_WDATA[31:16].

8.3 eFuse

The internal eFuse is used to store default configuration data and part ID data. This data is read from the eFuse following a reset event before the SPI ROM data is loaded.

The eFuse data is organized as 32 words of 32 bits of data. All bits in the eFuse are initially zero before programming. Programming can be used to change individual bits to one (but never back to zero). The 32 words provide 32 addressable locations in the eFuse memory. Data is read from the eFuse after a power-on-reset. Address locations 0 and 1 contain an

index of where to find the configuration data and ID data in the rest of the address space (addresses 2 to 31).

Both configuration data and Id data have 32 usable bits.

Addresses are used as follows:

Address	Description
0	ID Index. This encodes the address of where the ID data is stored.
1	Configuration index. This encodes the address of where the Configuration data is stored.
2-31	ID or CONFIG data as determined by the related index.

Data words at address 0 and 1 are used as follows:

Bit	Example Value	Description
31:0		Used as a 32-bit priority encoded value of the address+2. Each bit position corresponds to 1 address location. Starting from bit 29 the first bit with a 1 in it corresponds to the address+2 of the location to use for the data. If bits 31 or 30 are set to 1, then this is the same as if all of bits 29:0 are 0, and no data is read (Once bits 31 or 30 are set to 1, the part can no longer be programmed).
	0x00000001	The valid data is at address 2
	0x00000003	The valid data is at address 3
	0x00000007	The valid data is at address 4
	0x1fffffff	The valid data is at address 30
	0x3fffffff	The valid data is at address 31

The bits of the configuration data word are defined as follows:

Bit	Value	Description
25:0		Default values for BOOT_CFG[25:0]. Supplies the boot configuration settings.
26		Reserved
30:27		Default values for MEM_MRG[3:0]. Supplies the memory margin values.
31		Signifies bits 30:0 are valid and should be used.

As the addresses index uses a priority encoding, the lowest addresses should always be used first in programming the eFuse. When it is desired to replace the current configuration data, the next available address location should be used, and the index updated accordingly.

Note that updating the eFuse requires that the index values be read to find the next available address. Then the data needs to be written, and the index needs to be updated. Data should always be read after it was written to insure that it was written correctly.

Before programming configuration settings, these settings should be verified by loading the equivalent values into the proper JTAG registers and resetting the part with JTAG commands. This can validate that the chosen values will work correctly in the system.

8.3.1 eFuse read procedure

- Load EF_WDATA register with 0 (Clear Test Read Data)
- Load EF_ADDR register with the desired read address (Load Address)
- Load EF_CMD registers with 0x1 (Interface Enable)
- Poll/Read EF_STATUS for Bit 0 being 1 (Oscillator is running)
- Load EF_CMD registers with 0x5 (Issue Read Command)
- Read EF_RDATA register (Result of eFuse read)
- Load EF_CMD registers with 0x0 (Interface Disable)

8.3.2 eFuse write procedure

- Load EF_ADDR register with the desired write address (Load Address)
- Load EF_WDATA register with data (Load Data)
- Load EF_TIMING register with proper timing data
- Load EF_CMD registers with 0x3 (Interface Enable with Write Mode)
- Poll/Read EF_STATUS for Bit 0 being 1 (Oscillator is running)
- Bring EFUSE_BURN_ENn pin low (Enable Write)
- Load EF_CMD registers with 0x7 (Issue Write Command)
- Poll/Read EF_STATUS register for Bit 1 being 0 (Check/Wait for write complete)
- Bring EFUSE_BURN_ENn pin high (Disable Write)
- Load EF_CMD registers with 0x0 (Interface Disable)
- Do eFuse Read procedure to verify result

8.3.3 Sample eFuse first time write procedure

- Read all the address locations in the eFuse to verify that it has not been programmed.
- Write the desired ID data to address 2

- Write the value 0x0000 0001 to address 0 (signifies that ID data can be found at address 2)
- Write the desired Configuration data to address 3
- Write the value 0x0000 0003 to address 1 (signifies that Configuration data can be found at address 3)

8.3.4 Sample eFuse second time write procedure

- Read all the address locations in the eFuse. Verify that address locations 4-31 are 0x0000 0000 and that locations 0 and 1 have values below 0x000 0004.
- Write the desired ID data to address 4
- Write the value 0x0000 0007 to address 0 (signifies that ID data is now found at address 4)
- Write the desired Configuration data to address 5
- Write the value 0x0000 000f to address 1 (signifies that Configuration data is now found at address 5)