QUICKLOGIC CORPORATION

S3 Technical Reference Manual (TRM)

S3 Architecture

Revision 1.01a

Notice of Disclaimer

QuickLogic is providing this design, product or intellectual property "as is." By providing the design, product or intellectual property as one possible implementation of your desired system-level feature, application, or standard, QuickLogic makes no representation that this implementation is free from any claims of infringement and any implied warranties of merchantability or fitness for a particular purpose. You are responsible for obtaining any rights you may require for your system implementation. QuickLogic shall not be liable for any damages arising out of or in connection with the use of the design, product or intellectual property including liability for lost profit, business interruption, or any other damages whatsoever. QuickLogic products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use QuickLogic products in these types of equipment or applications.

QuickLogic does not assume any liability for errors which may appear in this document. However, QuickLogic attempts to notify customers of such errors. QuickLogic retains the right to make changes to either the documentation, specification, or product without notice. Verify with QuickLogic that you have the latest specifications before finalizing a product design.

Copyright and Trademark Information

This document is intended to be used by system hardware engineers and low-level software/firmware developers.

Contents

List of Figures

List of Tables

Introduction

Chapter 1. EOS S3 Sensor Processing Platform key features

The EOS S3 is a multi-core, ultra-low power sensor processing system. It is designed for mobile market applications including smartphones, wearables and hearables as well as Internet of Things (IoT) devices. The core of the EOS S3 is QuickLogic's proprietary Flexible Fusion Engine (FFE). To complement the FFE, the EOS S3 also includes an ARM® Cortex-M4-F sub-system to enable higher level processing allowing the application processor to offload additional computation requirements to EOS S3. The multi-core approach along with multiple power islands allows the EOS S3 to process sensor data and run algorithms in the most processing and power efficient manner possible.

1.1 Features List

Multi-Core Design

- Ultra-low power µDSP-like Flexible Fusion Engine (FFE) for always-on, real-time sensor fusion algorithms, an ARM® Cortex® M4-F floating point processor for general purpose processing, and on-chip programmable logic for flexibility and integration of additional logic functions to a single device
- Multiple, concurrent cores enables algorithm partitioning capability to achieve the most power and computationally efficient sensor processing system-on-a-chip (SoC) in the market

Cortex-M4-F Processor

- Up to 80 MHz operating speed
- Up to 512 KB SRAM with multiple power modes, including deep sleep (128 KB of this memory can be used for HiFi sensor batching)
- Ideal for computationally intensive sensor processing algorithms (continuous heart rate monitor, indoor navigation, always-on voice triggering, etc.)

Third-Generation Flexible Fusion Engine (FFE)

- Up to 10 MHz operating frequency
- 50 KB control memory
- 16 KB data memory
- µDSP-like architecture for efficient mathematical computations
- Ideal for always-on, real-time sensor fusion algorithms (such as pedometer, activity classification, gesture recognition, and others)

Sensor Manager

- 1.5 KB x 18-bit memory
- Completely autonomous (zero load on the M4-F) initialization and sampling of sensors through hard-wire I2C or configurable I2C/SPI

Communication Manager

- Communicates with host applications processor through its SPI Slave interface.
- Up to 20 MHz

Dedicated Voice Support

- Audio support for Pulse Density Modulation (PDM) or I2S microphones
- Optional hardware PDM bypass path to forward microphone data to application processor or Voice CODEC
- Dedicated logic for PDM to Pulse Code Modulation (PCM) conversion
- Dedicated hard logic integration of Sensory Low Power Sound Detect (LPSD) for on-chip voice Recognition

Programmable Fabric (FPGA)

- 2,400 effective logic cells with 64 Kbit RAM available
- Eight RAM FIFO controllers
- Provides capability to add logic functions or augment existing logic functions

Additional Features

- On-device circuit to support 32.768 kHz clock or crystal oscillator
- Dual Low-Dropout (LDO) regulators for on-chip regulation
- Power Management Unit (PMU) for minimizing power in all conditions (idle, deep sleep, and shut down)
- SPI Master, SPI Slave, I^2S Master, I^2S Slave, and I^2C interfaces
- Audio support for PDM or I2S microphones
- Internal codec available for Pulse Density Modulation (PDM) to Pulse Code Modulation (PCM) conversion
- 12-bit ΔΣ Analog-to-Digital Converter (ADC) for battery monitoring
- 2-pin Serial Wire Debug (SWD) port
- System DMA engine for efficient data movement
- eFuse memory for storing AES Key, voltage trim and other configurable information

Operating System Support

- Android OS; Contact QuickLogic's FAE for details
- Real-Time Operating System (RTOS) compatible

The following top-level features enable EOS S3 to support Companion and Host use cases.

Table 1-1: EOS S3 Supported features

Feature	Details
M4-F Sub-System	• Cortex M4-F controller with floating point unit support (M4-F) • Embedded SRAM (up to 512 KB) for code and data memory • Vectored interrupt support • Wakeup interrupt controller • 2-pin SWD port
Flexible Fusion Engine	• 50 KB control memory • 16 KB data memory • Single cycle MAC
Digital Microphone Support	• I2S microphone • PDM microphone • On-chip PDM-to-PCM conversion • Hardware bypass path for PDM interface to Host Application Processor and/or Voice CODEC • Integrated Low Power Sound Detector from Sensory, Inc.
Packet FIFOs Batching Memory	• 128 KB of M4-F SRAM can be used as HiFi sensor batching memory • Multiple packet FIFOs to support the FFE to application processor/M4-F data transfers: - 8 KB packet FIFO with ring-buffer mode support - 256 x 32 packet FIFO and two 128 x 32 packet FIFOs
Power Management Unit	• Low-power mode with fast wake-up • Programmable power modes (deep sleep, sleep with retention, and active) • Multiple power domains • Power sequencing for sleep and wake-up entry and exit • Software and hardware initiated sleep entry • Wake-up triggers via internal and external events • Internal LDO support
Programmable FPGA Fabric	• 2,400 effective logic cells with 64 Kbit of RAM, 8 RAM FIFO controllers and 2 GPIO banks • Supports SPI slave configuration • Supports reconfiguration from M4-F • Supports five clocks

Feature	Details
32 kHz Oscillator with Real-Time Clock (RTC)	• 32 kHz crystal oscillator (external crystal required) with bypass option • 1 Hz clock generation with compensation register • RTC function with one alarm register • Start time of 350µs
High Frequency Clock Source	• Programmable frequency (2 MHz to 80 MHz) for better frequency resolution • Calibrated output (using 32 kHz input) • Startup time of 410µs • Clock divider can be programmed in 12 bits
System DMA	• 16 channel DMA allows efficient data movement between processing elements
SPI Slave	• SPI slave application processor communication of up to 20 MHz
Time Stamping	• Automatic hardware time stamp on every sensor read in the interrupt mode • Up to eight sensor interrupt captured time-stamps (8-bit) • Main time stamp of 30 bits for M4-F processor and 24 bits for FFE • Resolution of 1ms
I2C Master and Configurable I2C/SPI Interface	• I2C master and SPI master with programmable clock pre-scalar • Option to disable multi-master support and slave-inserted wait for shorter SCL cycles • Configurable for two I2C Masters or one I2C Master and one SPI Master
Other Interfaces	• SPI master for interfacing with serial flash memories and other external SPI-based peripherals of up to 20 MHz • I2S Slave Transmitter for downloading audio samples to Host Application Processor
UART	• Serial support for M4-F debug and code development • Communication with UART-based external peripherals
Other Peripherals	• Timers • Watchdogs • GPIO controllers
ADC	• Low sampling rate ΔΣ 12-bit
LDOs	• On-chip LDO for system logic • Separate on-board LDO for memory
eFuse	• On-chip eFuse for storing fuse data for in-chip performance tuning and security key bits

Feature	Details
Integrated Software Debug Interface	• 2-pin SWD port for access to the following memory mapped resources: - M4-F internal registers and memories - FFE and Sensor Manager memories - FFE control registers - On-chip programmable logic memories - On-chip programmable logic designs through generic AHB bus - All memory map peripherals such as timers, WDT, SPI master, etc. - I2C master used for I2C sensor debug - Multiplexed dedicated parallel debug interface
Packaging Options	• 42-ball WLCSP (2.66 mm x 2.42 mm x 0.51 mm) (27 user I/O, 2 VCCIO banks) • 64-ball BGA (3.5 mm x 3.5 mm x 0.71mm) (46 user I/O, 2 VCCIO banks)

This section discusses the S3 general hardware architecture. Sub-systems and the functional architecture features are covered in their own separate sections. For example, power, Sensor processing, and voice / audio are discussed in detail in later sections.

Chapter 2. The S3 high-level architecture

The S3 is comprised of four major subsystems together with a number of system top-level modules. The sub-systems are:

- Cortex-M4-F sub-system – has ARM Cortex-M4-F processor, SRAMs, and local APB peripherals
- Voice / audio processing sub-system – capture and detect voice/sound via PDM or I2S interface
- Sensor Processing sub-system – QuickLogic Flexible Fusion Engine (FFE) and sensor managers capture and process external sensor data, with two I2C ports and an SPI master port
- FPGA sub-system – user configurable logic module

(Continued on next page)

2.1 S3 block diagram with main sub-systems

The block diagram illustrates the S3 internal architecture with color coded sub-system groups. (These are not color coded by class of functions, nor power domain.)

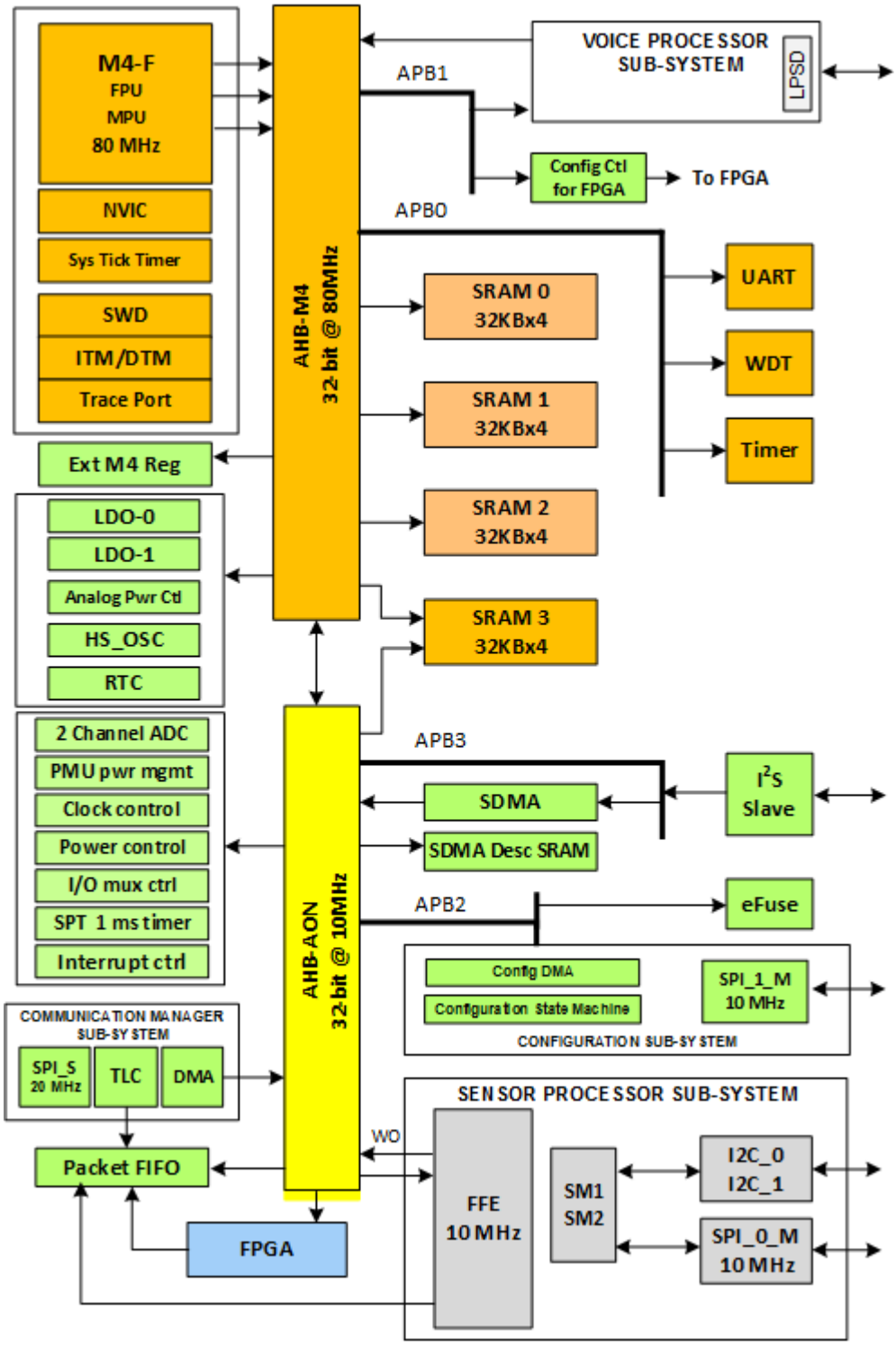

Figure 2-1: S3 high-level block diagram, with five main sub-system groups color coded

2.2 The top-level internal system elements and peripherals in the S3

This section identifies all top-level modules that are part of the main system but not in the major sub-systems. The buses are identified separately in Section 2.4.

2.2.1 List in alphabetical order

This is a list of the top-domain functional blocks in the block diagram. These include system elements such as buses, SRAM, and SDMA

2 channel ADC	The S3 includes a two-channel, accurate high resolution analog-to-digital converter that can be used for voltage monitoring such as a battery check.
AHB (AON) bus	AHB always-on bus, in a power domain separate from that of the M4 and its AHB bus
AHB (M4) bus	AHB bus in the same power domain as the M4
APC	Analog power control block includes the LDOs, RTC
Clock control block	The Clock Reset Unit (CRU) in this block controls the various clock dividers
Config for FPGA	Configuration control for the FPGA
Communication Manager	This subsystem supports communication with a Host / Application Processor in system designs of that type. Its key elements are: SPI_slave interface; TLC; CM DMA.
DMA for the CM	A DMA controller dedicated for use by the TLC
eFuse	System component used to customize certain OEM features
Ext M4 Reg	A group of various ARM M4 registers that are normally optional, and which have been implemented in the S3 M4
FFE	Flexible Fusion Engine for sensor management
FPGA	The S3 contains a user-configurable FPGA logic module
HS_OSC	Internal high-frequency clock generator that can be used to provide the HS_clock, for designs not having an external system-provided high-speed clock signal
I/O mux control	The IOMux handles multiplexing of signals for the IO pads
I^2C x2	There are two I^2C ports, named I2C_0 and I2C_1
I^2S master	I^2S master in Voice Processor
I^2S slave	Interface usable by the Voice / Audio Processing sub-system
Interrupt control	Interrupt control block containing Wake-up Interrupt Controller
ITM/DTM	Instrumentation Trace Macrocell
LDO-0	On-board power regulator 0
LDO-1	On-board power regulator 1
M4-F	Cortex M4 CPU with floating point unit
NVIC	Nested Vectored Interrupt Controller
Packet FIFO	The Packet FIFO Bank handles storage for the FFE
Power control	Control (switching) for the power islands
Power mgmt	The Power Management Unit or PMU provides power scheme management.

RTC	Realtime clock
SDMA	System DMA
SDMA desc SRAM	Description SRAM for SDMA
SM	Sensor Manager. There are two sensor managers.
SPI_0_M	SPI master port for use by the Sensor Processor subsystem
SPI_1_M	SPI master port for initial program load from flash and other SPI slave devices
SPI_S	SPI slave port for connection from Application Processor. A top-level component that is part of the Communication Manager sub-system.
SPT / 1 msec timer	a simple tick timer in its own power domain
SRAM	M4 SRAM - four static RAM blocks supporting the M4 – total of 512KB
SWD	Serial wire debug port
SysTick Timer	A timer for the M4. It is in the Ext M4 register group.
M4 Timer1	The M4 timer1 is within the M4 registers.
TLC	Top-Level Controller. A top-level component that is part of the Communication Manager sub-system. Its operation is tightly integrated with the CM DMA.
Trace Port	a port for delivering trace data stream to an external Trace Port Analyzer (TPA)
UART	Universal Asynchronous Receiver/Transmiter
Voice / audio processing sub-system with LPSD	The integrated Voice / audio processing sub-system is designed to support always-on voice capability. It has been optimized to work with Sensory's TrulyHandsfree™ Voice Control voice recognition software.
WDT	Watchdog timer

2.2.2 List by functional groups

The functional modules and elements listed here are described in detail in the chapters in the section of this document titled "Top-level peripherals".

Power-related functions

- LDO
- PMU / Reset

Clocking and timing

- clock control
- clock oscillators
- Clocking and Timing Elements
- RTC
- M4 Timer1
- Watchdog timer
- other timers in the system (SysTick; SPT)

System control

- Interrupt Controller including Wake-up Interrupt Controller (WIC)
- NVIC

Intra-module connection and data transfer

- AHB bus in M4 power domain
- AON AHB bus / bus matrix
- AHB2APB bus bridges (four)
- System DMA (SDMA) controller

I/O functions

- Master Serial Peripheral Interface (SPI Master) 0 and 1
- Slave Serial Peripheral Interface (SPI slave). Integrated into the Communication Manager for use with a Host
- I^2C master
- I^2S slave
- I^2S master (in Voice Processor) for I2S microphone
- GPIO
- UART
- Analog to Digital Converter (ADC)

Communications and data transfer to / from external systems

- TLC - Top-Level Controller for interfacing with an AP
- Packet FIFO supporting use with external sensors data batching

Storage and buffers

- M4 sub-system SRAM; FPGA SRAM; Audio SRAM
- Packet FIFO

Misc

- Debug and Test (Trace port; Serial Wire Debug port)
- eFuse
- FPGA configuration control module
- Configuration State Machine for initial program load from flash (includes control of DMA for this operation)

2.2.3 Accessing elements and peripherals

Accessing a component in the S3 involves more than just reading or writing to it. Access depends on whether the element power state is on or off, and whether the conduit to it (a bus or a bridge) is powered on or off. Many system elements can sleep to save power but then be powered up based on an interrupt and the action of the Power Management Unit.

To understand component access, refer to the sections of this manual covering power domains and also clocking, and the relevant section for a particular component or sub-system.

2.3 Key blocks in the main sub-systems

2.3.1 Cortex-M4-F main modules

Have ARM Cortex-M4-F processor, the M4 AHB, SRAMs, and related APB peripherals.

The build configuration of the M4-F is:

- ARM Cortex M4F process – processor (ARM IP)
- Memory – power-manageable memory blocks within Cortex-M4-F sub-system
- Cortex M4F Core Peripherals – processor core peripherals (ARM IP)
- Integrated Configurable Debug – for debugging (ARM IP)
- Instrumentation Trace Macrocell (ITM) – for debugging (ARM IP)
- Data Watchpoint and Trace (DWT) – for debugging (ARM IP)
- Flash Patch and Breakpoint Unit (FB) – for debugging and patching (ARM IP)
- Trace Port Interface Unit (TPIU) – for debugging (ARM IP)
- Nested Vectored Interrupt Controller (NVIC) – interrupt controller module (ARM IP)
- Memory Protection Unit (MPU) – memory protection module (ARM IP)
- AHB2APB Bus Bridge – bus bridge from AHB to APB peripherals modules (ARM IP); there are four of these in the S3.
- M4 Timer – timer module (ARM IP, r1p0)
- WDT – watchdog timer module (ARM IP, r1p0)
- UART – asynchronous serial port module (ARM IP: PL011, r1p5)
- AHB bus matrix (ARM IP) operating at up to 80 MHz. There is also a separate AHB bus operating at up to 10 MHz and which is in a different power domain from this one.

These modules are discussed in later chapters.

2.3.2 Voice / audio processor main modules

For detection and capture of voice/sound via PDM or I^2S interface

- Memory/FIFO - There are 6 SRAMs in the voice sub-system.
- Pulse Density Modulation (PDM) Interface - allows connecting a digital microphone with PDM output to this interface.
- Inter-IC Sound (I2S) Interface - allows connecting a digital microphone with I^2S output.
- PDM2PCM Core - an internal core to convert PDM audio data to PCM audio data.
- Low-Power Sound Detect (LPSD) - this module is acoustic activity detection IP from Sensory Inc. It can generate an interrupt to wake-up the M4 when a speech pattern is detected.
- DMA to transfer audio data to system memory

These modules are discussed in chapter 20.

2.3.3 Sensor processor main modules

This subsystem's Sensor Manager captures external sensor data and pass to Flexible Fusion Engine (FFE) for processing. Data can come in through one or both I^2C ports, or the SPI_1 master port. This subsystem's elements are covered in chapter 21.

2.3.4 FPGA main modules

This sub-system includes the FPGA, its configuration module, and internal RAM, The FPGA provides user-configurable glue logic and a small amount of storage. See Chapter 27 for details.

2.3.5 Communication Manager Modules

This sub-system includes the SPI_Slave interface for use with a Host; the TLC, and the DMA for the CM.

2.4 Buses in the S3: AHB, secondary buses, and bridges

The S3 bus hierarchy is designed to support optimum functional partitioning with power management in mind. The hierarchy allows the S3 the ability to support system power at fine granularity.

Main buses

There are two AHB buses in the S3: the M4 AHB which is tightly coupled to the M4 processor and which is in the same power domain, and the AON AHB which is always on.

Each AHB bus has several bus bridges attached to it to connect to various APB modules. These allow separating the high-speed core from slower elements, aiding power savings capability

Bridges between main buses

The AHB2AHB Bridge 0 connects the M4 AHB bus to the AON AHB bus, which is always on. This allows independent power operation of the AHB buses. This bridge also connects to the Ext M4 reg block.

The AHB2AHB Bridge 1 connects the M4 AHB bus to the AON AHB bus but also supports connection to SRAM.

Secondary bridges from buses

The secondary bridges are:

- AHB2APB Bridge 0 supports smaller modules like the UART and some timers.
- AHB2APB Bridge 1 mainly supports the Voice / audio processor sub-system.
- AHB2APB Bridge 2 supports the eFuse system component which is for factory use.
- AHB2APB bridge 3 supports SDMA and the I2S slave module.

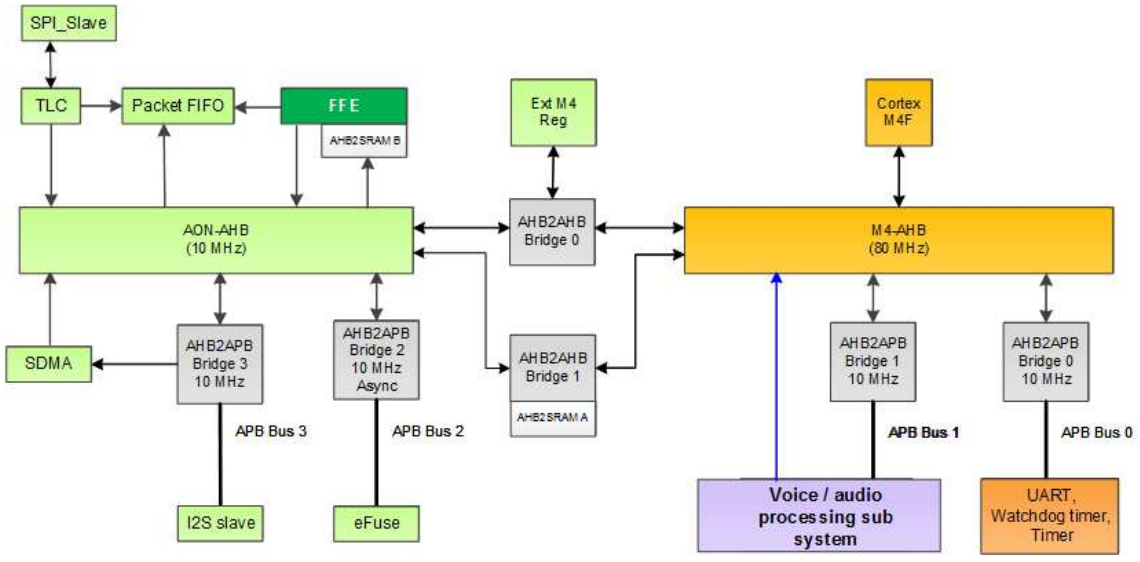

Figure 2-2: Bus architecture in S3

2.5 Memory concepts in the S3 architecture

The S3 gives designers several ways to store data:

1. SRAM banks
2. Packet FIFOs (PKFB SRAM)
3. SRAM within the FPGA fabric

2.5.1 SRAM banks and their usage scenarios

There are four SRAM banks for the M4. Initial program load from external flash will go into the RAM banks.

The first three of the banks are in the M4 power domain and the fourth one is in its own power domain.

Special system design aspects of the fourth SRAM bank

Because the fourth bank is in its own power domain, this allows storing data even when the M4 is asleep.

Also, the fourth bank is accessed from a bus arbiter spanning both the M4 AHB bus and the AON AHB bus. This allows other system elements to access this RAM at any time by means of the AON AHB bus. For example, HiFi sensor data batching can be done to this always-on 128 KB RAM (as well as to other storage such as the Packet FIFO or RAM in FPGA.

2.5.2 Packet FIFOs

There are four packet FIFOs of varying sizes of SRAM. These can be used to handle I/O streams from eternal sensors or to or from an AP.

- 8 KB packet FIFO with ring-buffer mode support
- 256 x 32 packet FIFO
- Two 128 x 32 packet FIFOs

This RAM is in the SRAM PF power domain.

2.5.3 SRAM in the FPGA

The FPGA area contains some dedicated SRAM, which is in the FB power domain. Users may apply it in designs as needed.

2.5.4 Memory power domains and sleep modes

In the S3 there are 31 major power domains (which include the M4 SRAMs) and 19 other, separate, SRAM power domains.

Each of the memory types (M4 SRAM, FIFO SRAM, and FPGA SRAM) has its own available types of power modes. This gives designers flexibility in creating application functions.

These characteristics are discussed in various chapters that related to device Power modes, controls as well as the chapters for the topics of SRAM, Packet FIFO, and FPGA.

2.6 Support for designs with an Applications Processor

- The S3 design features a low-power Communication Manager sub-system for efficient communication with an AP
- The CM's TLC can talk to an AP through a SPI slave interface that can operate at up to 20 MHz
- The TLC can use the Packet FIFO buffer
- S3 interrupts can be routed to the AP
- The AP reboot interrupt input, can inform the S3 of host status

2.7 Support for standalone designs (Host Mode operation)

- The SPI_1_Master module enables the S3 to support Host Mode designs, through its ability to:
  - Retrieve boot code from an external Flash storage device
  - Load this code into the M4-F memory
  - Enable the M4-F to begin code execution
  - Once booted, the M4-F can use SPI_1_Master for SPI communication with any SPI device that is connected to this port. Alternatively, the M4-F may load the on-chip programmable logic with IP that can use the SPI_1_Master interface.
    Note that the Sensor Managers in the FFE block cannot access the SPI_1_Master. Instead, the Sensor Managers use the SPI_0_Master interface for their sensor fusion operations.
- The S3 offers flexibility through alternative ways to establish system clocking (crystal or external signal source)

Chapter 3. ARM Cortex-M4F Sub-System

This chapter and the next one describe the M4 sub-system which contains the ARM IP: Cortex-M4F processor, debugging, bus, and peripherals, as well as 512K SRAM blocks. These sections are included:

- Section 3.1 Description – overall description
- Section 3.2 ARM Cortex-M4-F IP Configuration
- Section 3.3 ARM Cortex-M4F Processor – processor (ARM IP)
- Section 3.4 Cortex-M4F Core Peripherals – processor core peripherals (ARM IP)
- Section 3.5 Buses – system buses
- Section 3.6 Memory – memory blocks within Coretex-M4F sub-system
- Section 3.7 Memory Protection Unit (MPU) – memory protection module (ARM IP)
- Section 3.8 Nested Vectored Interrupt Controller (NVIC) – interrupt controller module (ARM IP)
- Section 3.9 M4 Timer – timer module (ARM IP, r1p0)

3.1 Description

The Cortex-M4F sub-system is one of the primary computation blocks of the EOS platform with the following features

- ARM Cortex-M4F 32-bit CPU with floating point arithmetic
- 2 MHz to 80 MHz operating speed
- Up to 512 KB SRAM with multiple power modes, including deep sleep (128 KB of this memory can be used for HiFi sensor batching)
- Ideal for computationally intensive sensor processing algorithms (continuous heart rate monitor, indoor navigation, always-on voice triggering, etc.)
- ARM IP features such as NVIC, flash patch, SysTick timer, Memory Protection Unit, Bit-Banding, and Integrated debug features.
- Power island to turn on/off M4F as needed
- 512 KB SRAM (16 blocks of 32 KB; each block with its own power island)
- AHB bus muxes (AHB bus matrices and AHB slave muxes)
- Multiple asynchronous bridges to interface with the programmable fabric and the FFE sub-system (AHB-to-AHB to the programmable fabric and AHB-to-SRAM to the FFE)
- Peripheral bus (APB) to:
  - UART
  - Watchdog Timer
  - S3 M4 Timer

The following diagram illustrates the blocks within the Cortex-M4-F sub-system.

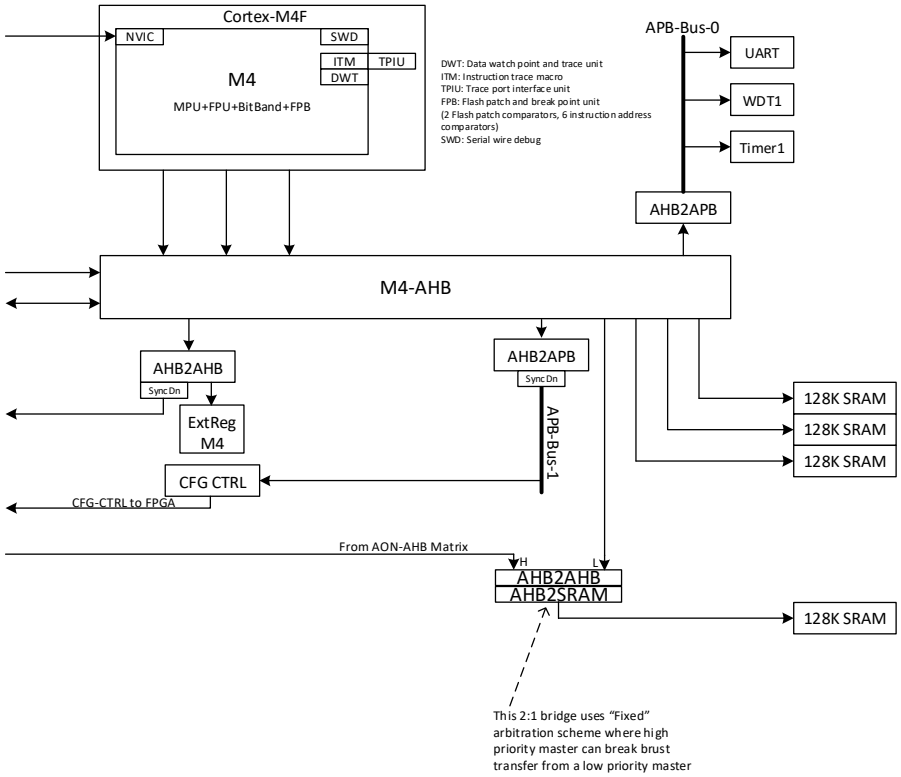

Figure 3-1: Cortex-M4-F sub-system Block Diagram

3.2 ARM Cortex-M4-F IP Configuration

The IP from ARM for Cortex-M4-F can be implemented with various optional components.

Table 3-1 summarizes the options QuickLogic has selected for the EOS S3.

Table 3-1: QuickLogic-implemented ARM IP configuration options

Feature	Description	Feature Implemented
MPU	Memory Protection Unit.	Yes
FPB	Flash Patch and Breakpoint Unit	Yes, 8 breakpoints
DWT	Data Watchpoint and Trace Unit	Yes
ITM	Instrumentation Trace Macrocell Unit	Yes, 32 ports
ETM	Embedded Trace Macrocell	NO
AHB-AP	Memory Access Port (MEM-AP)	Yes (shown in CPU diagram)
HTM interface	AHB Trace Macrocell interface	NO (ETM needed for HTM)
TPIU	Trace Port Interface Unit	Yes
WIC	Wake-up Interrupt Controller	Yes
Debug Port	Debug port AHB-AP interface (SW-DP)	Yes (shown in CPU diagram)
FPU	Floating Point Unit	Yes
Bit-banding	Bit-banding	Yes. Includes atomic bit-band read and write operations

3.3 ARM Cortex-M4-F Processor

3.3.1 Cortex-M4-F Processor Block Diagram

The following diagram illustrates the Cortex-M4-F processor and IP blocks provided by ARM.

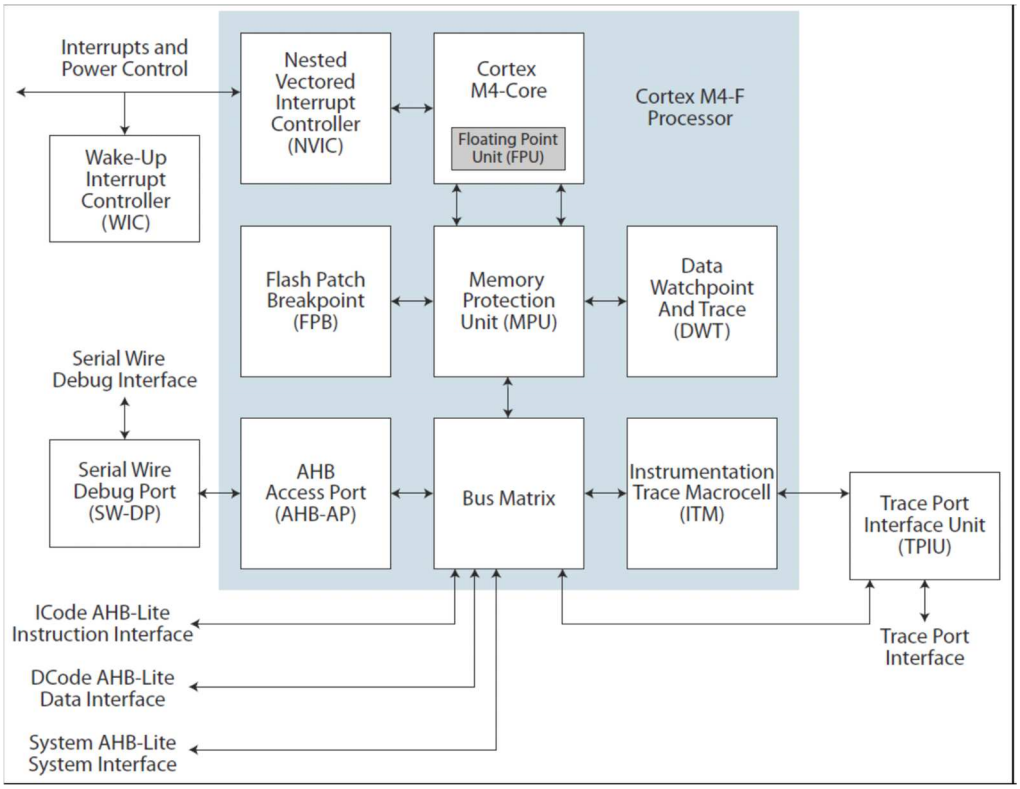

Figure 3-2: Cortex-M4-F Block Diagram

3.3.2 FPU

The Cortex-M4 FPU is an implementation of the single precision variant of the ARMv7-M Floating-Point Extension (FPv4-SP). It provides floating-point computation functionality that is compliant with the *ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*, referred to as the IEEE 754 standard. The FPU supports all single-precision data-processing instructions and data types described in the *ARM Architecture Reference Manual*.

3.4 Cortex-M4-F Core Peripherals

Core peripherals in the M4 sub-system that are provided by ARM and included in the EOS S3 fall into two categories: debugging and peripheral IP:

Debug

- Instrumentation Trace Macrocell (ITM) Unit
- Flash Patch and Breakpoint (FPB) Unit
- Data Watchpoint and Trace (DWT) Unit
- Trace Port Interface Unit (TPIU)
  Note: refer to the Cortex M4 Debug section for more details

Peripherals

- Watchdog Timer (WDT)
- M4 Timer
- NVIC and SysTick timer
- Universal Asynchronous Receiver/Transmitter (UART)
- System DMA
  Note: for UART and System DMA, see Special Function sections

3.4.1 Peripheral ID, Component ID, Designer ID, and Part and Revision Number

Peripheral ID, component ID, designer ID, and revision ID numbers are used by debugger to automatically identify what module and features are available in the device.

Table 3-2: ROM Table ID values: Peripheral IDs, Component IDs and derived values

		Base Address:	0x4001_0xxx	0x4001_2xxx	0x4001_3xxx	0x4000_Cxxx	0xE000_1xxx
Offset	Name	Description	UART	WDT	TIMER	SDMA	DWT
0xFD0	PID4	Peripheral ID 4	–	0x04	0x04	0x04	0x04
0xFD4	PID5	Peripheral ID 5	–	0x00	0x00	–	0x00
0xFD8	PID6	Peripheral ID 6	–	0x00	0x00	–	0x00
0xFDC	PID7	Peripheral ID 7	–	0x00	0x00	–	0x00
0xFE0	PID0	Peripheral ID 0	0x11	0x24	0x22	0x30	0x02
0xFE4	PID1	Peripheral ID 1	0x10	0xB8	0xB8	0xB2	0xB0
0xFE8	PID2	Peripheral ID 2	0x34	0x1B	0x1B	0x0B	0x3B
0xFEC	PID3	Peripheral ID 3	0x00	0x00	0x00	0x00	0x00
0xFF0	CID0	Component ID 0	0x0D	0x0D	0x0D	0x0D	0x0D
0xFF4	CID1	Component ID 1	0xF0	0xF0	0xF0	0xF0	0xe0
0xFF8	CID2	Component ID 2	0x5	0x05	0x05	0x05	0x05
0xFFC	CID3	Component ID 3	0xB1	0xB1	0xB1	0xB1	0xB1
Part Number (PID1[3:0],PID0[7:0])			0x011	0x824	0x822	0x230	0x002
Designer (JEDEC) ID (PID2[6:4],PID1[7:4])			0x41	0x3B	0x3B	0x3B	0x3B
Revision (PID2[7:4])			0x3	0x1	0x1	0x0	0x3
ECO Revision (PID3[7:4])			0x0	0x0	0x0	0x0	0x0
Customer modification number (PID3[3:0])			0x0	0x0	0x0	0x0	0x0
Component ID (CID3,CID2,CID1,CID0)			0xB105_F00D	0xB105_F00D	0xB105_F00D	0xB105_F00D	0xB105_E00D

The designer ID as specified in JEDEC JEP 106 [1]is 0x3B (in bank five) for ARM. Modules WDT, TIMER, DWT, and SDMA have these designer IDs. Peripheral ID 4 has continuation code of 0x04. UART (PrimeCell PL011) has JEDEC_USED=0, and designer ID is 0x41 (non JEDEC ID used by ARM). Peripheral IDs 4-7 are not defined.

Peripheral ID numbers, PID0, PID1, PID2, and PID3 are defined as follows:

Table 3-3: PID0 Register Bit Description

0xFE0	PID0	bit	R/W/C	Default	Description
	partNum7_0	7:0	RO	–	Part_number[7:0]

Table 3-4: PID1 Register Bit Description

0xFE4	PID1	bit	R/W/C	Default	Description
	REVISION	7:4	RO	–	Design_ID[3:0]
	partNum11_8	3:0	RO	–	Part_number[11:8]

Table 3-5: PID2 Register Bit Description

0xFE8	PID2	bit	R/W/C	Default	Description
	REVISION	7:4	RO	–	Revision number
	JEDEC_USED	3	RO	–	0=does not follow JEDEC JEP106 1=JEDEC JEP106 is used
	ID6_4	2:0	RO	–	Designer_ID[6_4]

Table 3-6: PID3 Register Bit Description

0xFEC	PID3	bit	R/W/C	Default	Description
	ECO_REVISION	7:4	RO	–	ECO Revision number
	CUST_MOD_NUM	3:0	RO	–	Customer modification number

See ARM *CoreSight Architecture Spec*[2] for more information.

[1] JEDEC JEP 106
[2] ARM IHI0029D CoreSight Arch. Spec.

The following PIDs and CIDs are from ARM Cortex-M4 TRM[3].

Table 3-7: ROM Table ID values: Peripheral IDs, Component IDs and derived values (from ARM)

		Base Address:	0xE000_0xxx	0xE000_2xxx	0xE004_0xxx	0xE000_Exxx	0xE00F_Fxxx
Offset	**Name**	**Description**	**ITM**	**FPB**	**TPIU**	**SCS Coresight**	**DAP**
0xFD0	PID4	Peripheral ID 4	0x04	0x04	0x04	0x04	0x04
0xFD4	PID5	Peripheral ID 5	0x00	0x00	0x00		0x00
0xFD8	PID6	Peripheral ID 6	0x00	0x00	0x00		0x00
0xFDC	PID7	Peripheral ID 7	0x00	0x00	0x00		0x00
0xFE0	PID0	Peripheral ID 0	0x01	0x03	0xA1	0x0C†	0xC4
0xFE4	PID1	Peripheral ID 1	0xB0	0xB0	0xB9	0xB0	0xB4
0xFE8	PID2	Peripheral ID 2	0x3B	0x2B	0x0B	0x0B	0x0B
0xFEC	PID3	Peripheral ID 3	0x00	0x00	0x00	0x00	0x00
0xFF0	CID0	Component ID 0	0x0D	0x0D	0x0D	0x0D	0x0D
0xFF4	CID1	Component ID 1	0xE0	0xE0	0x90	0xE0	0x10
0xFF8	CID2	Component ID 2	0x05	0x05	0x05	0x05	0x05
0xFFC	CID3	Component ID 3	0xB1	0xB1	0xB1	0xB1	0xB1
Part Number (PID1[3:0],PID0[7:0])			0x001	0x003	0x9A1	0x00C	0x4C4
Designer (JEDEC) ID (PID2[6:4],PID1[7:4])			0x3B	0x3B	0x3B	0x3B	0x3B
Revision (PID2[7:4])			0x3	0x2	0x0	0x0	0x0
ECO Revision (PID3[7:4])			0x0	0x0	0x0	0x0	0x0
Customer modification number (PID3[3:0])			0x0	0x0	0x0	0x0	0x0
Component ID (CID3,CID2,CID1,CID0)			0xB105_E00D	0xB105_E00D	0xB105_900D	0xB105_E00D	0xB105_100D

Note:

† SCS CoreSight part number = 0x000 if no FPU, 0x00C if FPU present

3.4.2 CPUID register

CPUID register is at 0xE000_ED00. The value should be 0x410F_C240 (from ARM DDI0439B)

Table 3-8: CPUID Register Bit Description

0xD00	CPUID	bit	R/W/C	Default	Description
	IMPLEMENTER	31:24	RO	0x41	Implementer: 0x41 = ARM
	VARIANT	23:20	RO	0x0	Processor revision: 0x0 = Rev 0
	CONSTANT	19:16	RO	0xF	Read as 0xF
	PARTNO	15:4	RO	0xC24	Part number: 0xC24 = Cortex-M4
	REVISION	3:0	RO	0x0	Patch release: 0x0 = Patch 0

[3] ARM DDI0439B Cortex-M4, r0p0, TRM

3.5 Buses

3.5.1 M4 AHB

In power domain: M4

The M4 AHB is the main bus for the M4. It is power-switched and in the same domain as the M4.

The bus location in the architecture is outlined in Figure 2-1.

Notes:

- The M4 AHB has the ability to simultaneously access all four of the M4 SRAM blocks via independent AHB buses.
- During times when the M4 AHB is powered down, the S3 can still access the fourth block of M4 SRAM. by means of the AON AHB bus.

3.5.2 AHB2APB bus bridges

In power domain: M4, AON, A1

The four AHB2APB bus bridges connect AHB buses to specific modules.

[This is clocked by Clock 11. See CRU CLK_CONTROL_D_0 Register]

3.6 Memory

3.6.1 Memory Map

The Cortex-M4-F memory map is defined as follows:

Table 3-9: Cortex M4-F memory map

Address	Name	Device Type	XN ?	Description
0x0000_0000 – 0x1FFF_FFFF	Code (Dcode/ ICode)	Normal		Typically ROM or flash memory. Memory required from address 0x0 to support the vector table for system boot code on reset
0x2000_0000 – 0x3FFF_FFFF	SRAM	Normal		SRAM region typically used for on-chip RAM.
0x4000_0000 – 0x5FFF_FFFF	Peripheral	Device	XN	On-chip peripheral address space.
0x6000_0000 – 0x7FFF_FFFF	Reserved			
0x8000_0000 – 0x9FFF_FFFF	Reserved			
0xA000_0000 – 0xBFFF_FFFF	Reserved			
0xC000_0000 – 0xDFFF_FFFF	Reserved			
0xE000_0000 – 0xFFFF_FFFF	System	See Note	XN	System segment for the PPB and vendor system peripherals,

Note:

- XN = Execute Never
- System Region, 0xE000_0000 - 0xFFFF_FFFF, is subdivided as
  - Private Peripheral Bus (PPB): 0xE000_0000 – 0xE00F_FFFF
    - Internal 0xE000_0000 – 0xE003_FFFF
    - External 0xE004_0000 – 0xE00F_FFFF
  - Vendor system region (Vendor_SYS): 0xE010_0000 – 0xFFFF_FFFF (not used in EOS S3)

Table 3-10: SCS address space regions

System Control Space, address range 0xE000E000 to 0xE000EFFF		
Group	**Address range**	**Notes**
System control and ID registers	0xE000E000-0xE000E00F	Includes the Interrupt Controller Type and Auxiliary Control registers
	0xE000ED00-0xE000ED8F	System control block
	0xE000EDF0-0xE000EEFF	Debug registers in the SCS
	0xE000EF00-0xE000EF8F	Includes the SW Trigger Interrupt Register
	0xE000EF90-0xE000EFCF	IMPLEMENTATION DEFINED
	0xE000EFD0-0xE000EFFF	Microcontroller-specific ID space
SysTick	0xE000E010-0xE000E0FF	System Timer, see *The system timer, SysTick* on page B3-744
NVIC	0xE000E100-0xE000ECFF	External interrupt controller, see *Nested Vectored Interrupt Controller, NVIC* on page B3-750
MPU	0xE000ED90-0xE000EDEF	Memory Protection Unit, see *Protected Memory System Architecture, PMSAv7* on page B3-761

Table 3-11: PPB Debug related regions

Debug resource	Address range	See
Instrumentation Trace Macrocell (ITM)	0xE0000000-0xE0000FFF	*The Instrumentation Trace Macrocell* on page C1-843
Data Watchpoint and Trace (DWT) block	0xE0001000-0xE0001FFF	*The Data Watchpoint and Trace unit* on page C1-856
Flash Patch and Breakpoint (FPB) block	0xE0002000-0xE0002FFF	*Flash Patch and Breakpoint unit* on page C1-904
SCS	0xE000ED00-0xE000EFFF	*System Control Space (SCS)* on page B3-708
System Control Block (SCB)	0xE000ED00-0xE000ED8F	*About the System Control Block* on page B3-709
Debug Control Block (DCB)	0xE000EDF0-0xE000EEFF	*Debug register support in the SCS* on page C1-828
Trace Port Interface Unit (TPIU)[a]	0xE0040000-0xE0040FFF	*Trace Port Interface Unit* on page C1-897

Table 3-12: ARMv7 DAP accessible ROM table

Offset	Value	Name	Description
0x000	0xFFF0F003	ROMSCS	Points to the SCS at 0xE000E000.
0x004	0xFFF02002 or 0xFFF02003	ROMDWT	Points to the Data Watchpoint and Trace block at 0xE0001000. Bit [0] is set to 1 if a DWT is fitted.
0x008	0xFFF03002 or 0xFFF03003	ROMFPB	Points to the Flash Patch and Breakpoint block at 0xE0002000. Bit [0] is set to 1 if an FPB is fitted.
0x00C	0xFFF01002 or 0xFFF01003	ROMITM[a]	Points to the Instrumentation Trace block at 0xE0000000. Bit [0] is set to 1 if an ITM is fitted.
0x010	0xFFF41002 or 0xFFF41003	ROMTPIU[b]	Points to the Trace Port Interface Unit. Bit [0] is set to 1 if a TPIU is fitted and accessible to the processor on its PPB.
0x014	0xFFF42002 or 0xFFF42003	ROMETM[b]	Points to the Embedded Trace Macrocell block. Bit [0] is set to 1 if an ETM is fitted and accessible to the processor on its PPB.
0x018	0x00000000	End	End-of-table marker. It is IMPLEMENTATION DEFINED whether the table is extended with pointers to other system debug resources. The table entries always terminate with a null entry.

3.6.2 Memory Address Decoding

The following table summarizes the Cortex-M4-F memory address decoding.

Table 3-13: Memory Address Decoding Table (with reference to Excel tab)

Block	[this column reserved for future use]	M4 Base Address	Last Address	Size †
M4_mem0		0x0000_0000		128KB
M4_mem1		0x0002_0000		128KB
M4_mem2		0x0004_0000		128KB
M4_mem3_aon		0x0006_0000		128KB
RESERVED (code address space)		**0x0008_0000**	**0x1FFF_FFFF**	**–**
M4_mem0 (Mirror)	–	0x2000_0000		128KB
M4_mem1 (Mirror)	–	0x2002_0000		128KB
M4_mem2 (Mirror)	–	0x2004_0000		128KB
M4_mem3_aon (Mirror)	–	0x2006_0000		128KB
RESERVED (SRAM address space)		**0x2008_0000**	**0x3FFF_FFFF**	**–**
M4_Regs		0x4000_0000		4KB
Packet FIFO Bank		0x4000_2000		8KB
Clock		0x4000_4000		1KB
PMU - Power Management Unit		0x4000_4400		1KB
Intr_Ctrl (WIC)		0x4000_4800		1KB
IO_Mux		0x4000_4C00		1KB
Misc		0x4000_5000		1KB
AIP (Analog IP includes RTC, OSC, and LDO)		0x4000_5400		1KB
RESERVED		**0x4000_5800**	**0x4000_59FF**	**–**
JTM (ADC)		0x4000_5A00		512
SPT (RTC)		0x4000_5C00		512
RESERVED		**0x4000_5E00**		**–**
A1_Regs		0x4000_6000		4KB
SPI_MS		0x4000_7000		1KB
DMA_SPI_MS		0x4000_7400		1KB
eFuse		0x4000_8000		4KB
RESERVED		**0x4000_9000**	**0x4000_AFFF**	**–**
I2S_Slave		0x4000_B000		4KB
SDMA		0x4000_C000		4KB
SDMA_Bridge		0x4000_D000		4KB

Block		[this column reserved for future use]	M4 Base Address	Last Address	Size †
SDMA_SRAM			0x4000_F000		4KB
UART			0x4001_0000		4KB
WDT			0x4001_2000		4KB
Timer			0x4001_3000		4KB
CFG_CTL_TOP (PIF Controller)			0x4001_4000		4KB
AUD_Slave			0x4001_5000		4KB
RAMFIFO0 (FPGA)			0x4001_8000		4KB
RAMFIFO1 (FPGA)			0x4001_9000		4KB
RAMFIFO2 (FPGA)			0x4001_A000		4KB
RAMFIFO3 (FPGA)			0x4001_B000		4KB
Fabric (FPGA)			0x4002_0000		128KB
FFE			0x4004_0000	0x4004_FFFF	128KB
	DM0 (data memory)		_0000		16KB
	SM0 (sensor manager)		_4000		16KB
	SM1(sensor manager)		_8000		8KB
	ExtRegsFFE		_A000		8KB
CM (COMM MGR)			0x4005_0000	0x4007_FFFF	192KB
RESERVED (peripheral address space)			**0x4008_0000**	**0x5FFF_FFFF**	**–**
RESERVED			**0x6000_0000**	**0xDFFF_FFFF**	**–**
M4_ITM Instrumentation Trace Macrocell			0xE000_0000	0xE0000_0FFF	4KB
M4_DWT Data Watchpoint Unit			0xE000_1000	0xE0000_1FFF	4KB
M4_FPB Flash Patch and Breakpoint			0xE000_2000	0xE0000_2FFF	4KB
Reserved			**0xE000_3000**	**0xE000_DFFF**	**–**

Block		[this column reserved for future use]	M4 Base Address	Last Address	Size †
M4_SCS System Control Space			0xE000_E000	0xE000_EFFF	4KB
M4_SCS	Interrupt Controller Type (ICTR) and Auxiliary Control Registers (ACTLR)		0xE000_E000	0xE000_E00F	–
	SysTick		0xE000_E010	0xE000_E01F	–
	Reserved	**–**	**0xE000_E020**	**0xE000_E0FF**	–
	M4_NVIC Nested Vectored Interrupt Controller	SCS	0xE000_E100 0xE000_E200 0xE000_E300 0xE000_E400		
	Reserved		**0xE000_E500**	**0xE000_ECFF**	–
	M4_CPUID ‡		0xE000_ED00	0xE000_ED03	–
	Misc SCS Registers		0xE000_ED04	0xE000_ED8F	–
	M4_MPU		0xE000_ED90	0xE000_EDFF	–
	Reserved		**0xE000_EE00**	**0xE000_EEFF**	–
	Misc SCS Registers		0xE000_EF00	0xE000_EF33	–
	M4_FPU		0xE000_EF34	0xE000_EF47	–
	Reserved		**0xE000_EF48**	**0xE000_EFCF**	
	M4_SCS ID registers ^		0xE000_EFD0	0xE000_EFFF	–
Reserved			**0xE000_F000**	**0xE003_FFFF**	**–**
M4_TPIU Trace Port Interface Unit			0xE004_0000	0xE004_0FFF	–
Reserved			**0xE005_0000**	**0xE00F_EFFF**	**–**
M4_DAP Debug Access Port			0xE00F_F000	0xE00F_FFFF	–
Vendor_SYS: not used			0xE010_0000	0xFFFF_FFFF	–

Notes:

- See ARM TRM for details for ARM IP block

‡ CPUID described in Section Cortex M4-F Core Peripherals

^ ID registers described in Section M4-F sleep mode

3.6.3 M4 SRAM

The M4-F processor sub-system has up to 512 KB of embedded SRAM. This 512KB of SRAM is divided into 4 blocks of 128KB each, with each accessible simultaneously via independent AHB busses.

Three of the 128KB blocks (total of 384 KB) of the SRAM memory are accessible from the CPU AHB bus, and in order to access this memory space, the CPU sub-system must be powered on. Each of these 128KB blocks is addressed as four 32KB segments (12 segments in total). An interrupt can be triggered when any of these 32KB memory segments is accessed when the memory is in a lower power state (deep sleep or shut down).

The last 128KB SRAM block is accessible from AON AHB bus, and thus referred to as AON SRAM, however, it is possible to turn off power domain of individual 32KB segment. . It can be accessed regardless of the state of CPU sub-system power by AON AHB master. When M4 AHB master and AON AHB master request need arbitration, a 2:1 bridge (AHB2SRAM arbiter) uses "fixed" arbitration scheme where higher priority master can break burst transfer from a lower priority master.

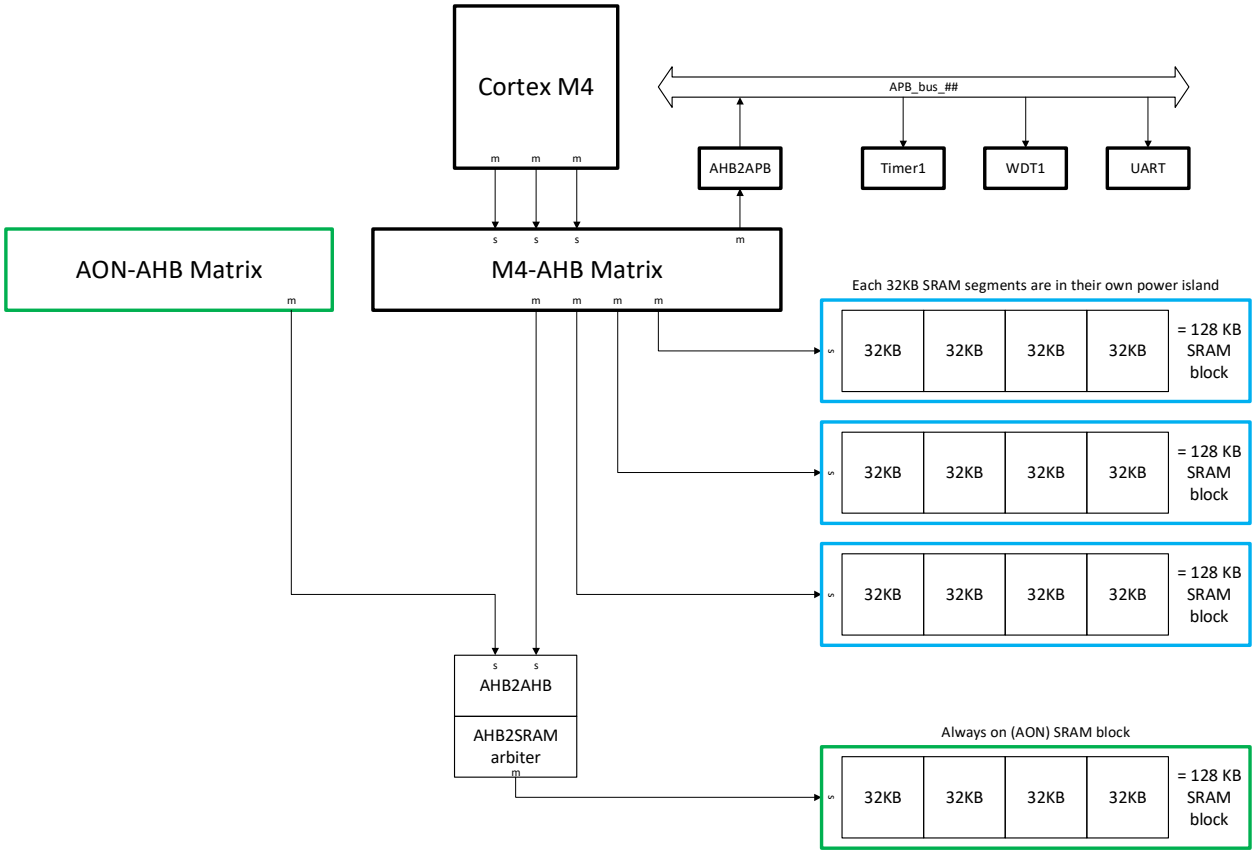

Figure 3-3: Cortex-M4-F SRAM groups

The 384 KB SRAM can be accessed by the following AHB masters:

- M4's system bus
- M4's I-code bus
- M4's D-code bus
- AP through SPI_Slave via the TLC
- DMA for SPI_Master
- DMA for PDM in Voice / audio processor

Table 3-11, SRAM Memory Map and Instance Name Cross Reference, summarizes the M4 memory map. The SRAM block number and corresponding instance name used in register descriptions are also included.

Table 3-14: SRAM Memory Map and Instance Name Cross Reference

SRAM Block	I-Code/D-Code Bus Access Address Range	M4 System Bus Access Address Range	M4SRAM Instance Name	Note
Mem0_0	0x0000_0000 - 0x0000_7FFF	0x2000_0000 - 0x2000_7FFF	M4S0	
Mem0_1	0x0000_8000 - 0x0000_FFFF	0x2000_8000 - 0x2000_FFFF	M4S1	
Mem0_2	0x0001_0000 - 0x0001_7FFF	0x2001_0000 - 0x2001_7FFF	M4S2	
Mem0_3	0x0001_8000 - 0x0001_FFFF	0x2001_8000 - 0x2001_FFFF	M4S3	
Mem1_0	0x0002_0000 - 0x0002_7FFF	0x2002_0000 - 0x2002_7FFF	M4S4	
Mem1_1	0x0002_8000 - 0x0002_FFFF	0x2002_8000 - 0x2002_FFFF	M4S5	
Mem1_2	0x0003_0000 - 0x0003_7FFF	0x2003_0000 - 0x2003_7FFF	M4S6	
Mem1_3	0x0003_8000 - 0x0003_FFFF	0x2003_8000 - 0x2003_FFFF	M4S7	
Mem2_0	0x0004_0000 - 0x0004_7FFF	0x2004_0000 - 0x2004_7FFF	M4S8	
Mem2_1	0x0004_8000 - 0x0004_FFFF	0x2004_8000 - 0x2004_FFFF	M4S9	
Mem2_2	0x0005_0000 - 0x0005_7FFF	0x2005_0000 - 0x2005_7FFF	M4S10	
Mem2_3	0x0005_8000 - 0x0005_FFFF	0x2005_8000 - 0x2005_FFFF	M4S11	
Mem3_0	0x0006_0000 - 0x0006_7FFF	0x2006_0000 - 0x2006_7FFF	M4S12	in AON power domain
Mem3_1	0x0006_8000 - 0x0006_FFFF	0x2006_8000 - 0x2006_FFFF	M4S13	in AON power domain
Mem3_2	0x0007_0000 - 0x0007_7FFF	0x2007_0000 - 0x2007_7FFF	M4S14	in AON power domain
Mem3_3	0x0007_8000 - 0x0007_FFFF	0x2007_8000 - 0x2007_FFFF	M4S15	in AON power domain

Any access not within the mapped boundaries will return a zero.

The SRAM clocks are dynamically controlled. When there is no activity on the memory, the clocks will be gated off to ensure lower power consumption.

3.6.3.1 SRAM Configuration

SRAM has following configuration options:

- LPMH – Low Power Memory Header
  To reduce bitcell leakage in Deep Sleep mode, enable this feature. See CONFIG_MEM1 register bit description.
- LPMF – Low Power Memory Footer
  To reduce leakage current, enable this Transparent Light Sleep (TLS) core feature. See CONFIG_MEM1 register bit description.
- RM – read and write margin control

3.6.3.2 Registers

Base address = 0x4000_0000

Table 3-15: ExtM4Regs Registers for SRAM

Offset	Name/Field	Default	Note
0x200	CONFIG_MEM0	0x00010840	Affects SRAM instance M4S0, M4S1, M4S2, M4S3
0x204	CONFIG_MEM1	0x00010840	Affects SRAM instance M4S4, M4S5, M4S6, M4S7
0x208	CONFIG_MEM2	0x00010840	Affects SRAM instance M4S8, M4S9, M4S10, M4S11
0x21C	M4_MEM_INTR	0x00000000	Flag indicating there was an access to SRAM while in deep sleep or shutdown mode.
0x220	M4_MEM_INTR_EN	0x00000000	Interrupt enable for M4_MEM_INTR

For AON MEM3, refer to Misc register space.

Table 3-16: CONFIG_MEM0 Register Bit Description

0x200	CONFIG_MEM0	31:0	R/W/C	Default	Description
	MEM3_32K_DST	19	RW	0x0	'Disable-Self-Time'. When asserted high, overrides the self-timed circuitry and causes the read margin to be controlled by the falling clk edge. Requires MEM3_32K_RM (read margin) to be set to 2'b00. See bits 16:15. Caution: This pin is intended for debug/FA purposes only.
	Reserved	18	RO	0x0	
	Reserved	17	RO	0x0	
	MEM3_32K_RM	16:15	RW	0x2	Read and write margin control. Recommended setting is 2'b10. 2'b00 provides the most margin (slowest speed). 2'b11 provides the least margin (fastest speed) memory. This setting is required for VDDMIN operation.
	MEM2_32K_DST	14	RW	0x0	Similar to MEM3_32K_xxx above
	Reserved	13	RW	0x0	
	Reserved	12	RW	0x0	
	MEM2_32K_RM	11:10	RW	0x2	
	MEM1_32K_DST	9	RW	0x0	Similar to MEM3_32K_xxx above
	Reserved	8	RW	0x0	
	Reserved	7	RW	0x0	
	MEM1_32K_RM	6:5	RW	0x2	
	MEM0_32K_DST	4	RW	0x0	Similar to MEM3_32K_xxx above
	Reserved	3	RW	0x0	
	Reserved	2	RW	0x0	
	MEM0_32K_RM	1:0	RW	0x2	

Table 3-17: CONFIG_MEM1 Register Bit Description

0x204	CONFIG_MEM1	31:0	R/W/C	Default	Description
					CONFIG_MEM1 details are identical to CONFIG_MEM0 above.

Table 3-18: CONFIG_MEM2 Register Bit Description

0x208	CONFIG_MEM2	31:0	R/W/C	Default	Description
					CONFIG_MEM2 details are identical to CONFIG_MEM0 above.

The following register indicates whether there was an access to SRAM block while that block is in deep sleep or shut down mode.

Table 3-19: M4_MEM_INTR Register Bit Description

0x21C	M4_MEM_INTR	31:0	R/W/C	Default	Description
	MEM2_INTR3	11	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 2 32KB_3 or M4S11) while in deep sleep or shut down mode
	MEM2_INTR2	10	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 2 32KB_2 or M4S10) while in deep sleep or shut down mode
	MEM2_INTR1	9	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 2 32KB_1 or M4S9) while in deep sleep or shut down mode
	MEM2_INTR0	8	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 2 32KB_0 or M4S8) while in deep sleep or shut down mode
	MEM1_INTR3	7	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 1 32KB_3 or M4S7) while in deep sleep or shut down mode
	MEM1_INTR2	6	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 1 32KB_2 or M4S6) while in deep sleep or shut down mode
	MEM1_INTR1	5	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 1 32KB_1 or M4S5) while in deep sleep or shut down mode
	MEM1_INTR0	4	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 1 32KB_0 or M4S4) while in deep sleep or shut down mode
	MEM0_INTR3	3	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 0 32KB_3 or M4S3) while in deep sleep or shut down mode
	MEM0_INTR2	2	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 0 32KB_2 or M4S2) while in deep sleep or shut down mode
	MEM0_INTR1	1	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 0 32KB_1 or M4S1) while in deep sleep or shut down mode
	MEM0_INTR0	0	RW1C	0x0	Interrupt caused by a SRAM access (M4 SRAM segment 0 32KB_0 or M4S0) while in deep sleep or shut down mode

The following register enables interrupt generated for an access to SRAM block while that block is in deep sleep or shut down mode.

Table 3-20: M4_MEM_INTR_EN Register Bit Description

0x21C	M4_MEM_INTR	31:0	R/W/C	Default	Description
	MEM2_INTR3_EN	11	RW	0x0	Interrupt enable (M4 SRAM segment 2 32KB_3, or M4S11)
	MEM2_INTR2_EN	10	RW	0x0	Interrupt enable (M4 SRAM segment 2 32KB_2, or M4S10)
	MEM2_INTR1_EN	9	RW	0x0	Interrupt enable (M4 SRAM segment 2 32KB_1, or M4S9)
	MEM2_INTR0_EN	8	RW	0x0	Interrupt enable (M4 SRAM segment 2 32KB_0, or M4S8)
	MEM1_INTR3_EN	7	RW	0x0	Interrupt enable (M4 SRAM segment 1 32KB_3, or M4S7)
	MEM1_INTR2_EN	6	RW	0x0	Interrupt enable (M4 SRAM segment 1 32KB_2, or M4S6)
	MEM1_INTR1_EN	5	RW	0x0	Interrupt enable (M4 SRAM segment 1 32KB_1, or M4S5)
	MEM1_INTR0_EN	4	RW	0x0	Interrupt enable (M4 SRAM segment 1 32KB_0, or M4S4)
	MEM0_INTR3_EN	3	RW	0x0	Interrupt enable (M4 SRAM segment 0 32KB_3, or M4S3)
	MEM0_INTR2_EN	2	RW	0x0	Interrupt enable (M4 SRAM segment 0 32KB_2, or M4S2)
	MEM0_INTR1_EN	1	RW	0x0	Interrupt enable (M4 SRAM segment 0 32KB_1, or M4S1)
	MEM0_INTR0_EN	0	RW	0x0	Interrupt enable (M4 SRAM segment 0 32KB_0, or M4S0)

Base address = 0x4000_5000

Table 3-21: Misc Registers for SRAM

Offset	Name/Field	Note
0x210	CONFIG_MEM128_AON	This is effectively CONFIG_MEM4. Affects the fourth M4 SRAM bank; these are SRAM instances M4S12, M4S13, M4S14, M4S15 in AON CONFIG_MEM128_AON bit definitions are identical to CONFIG_MEM0 described previously.

3.7 Memory Protection Unit (MPU)

In power domain: M4

The Memory Protection Unit has:

- Eight memory regions.
- Sub Region Disable (SRD), enabling efficient use of memory regions.
- The ability to enable a background region that implements the default memory map attributes.

The MPU is part of the ARM core.

Use of the MPU is controlled by the MPU Config register:

Base address = see section 3.6.2

Table 3-22: MPU Config Register in ExtM4Reg space

0x014	CONFIG2	31:0	R/W/C	Default	Description
	MPU_DISABLE	1	RW	0x0	If asserted the MPU is invisible and unusable. Set HIGH to disable the MPU. Set LOW to enable the MPU.

3.7.1 MPU Registers

The page numbers in the Description column refer to the ARMv7-M Architecture Reference Manual.

Table 3-23: MPU Registers

Address	Name	Type	Reset	Description
0xE000ED90	MPU_TYPE	RO	The reset state will be identified in a later rev of this manual.	MPU Type Register, MPU_TYPE on page B3-766
0xE000ED94	MPU_CTRL	RW	0x00000000	MPU Control Register, MPU_CTRL on page B3-767
0xE000ED98	MPU_RNR	RW		MPU Region Number Register, MPU_RNR on page B3-769
0xE000ED9C	MPU_RBAR	RW		MPU Region Base Address Register, MPU_RBAR on page B3-770
0xE000EDA0	MPU_RASR	RW		MPU Region Attribute and Size Register, MPU_RASR on page B3-771
0xE000EDA4	MPU_RBAR_A1	RW		Alias 1 of MPU_RBAR, see MPU alias register support on page B3-775
0xE000EDA8	MPU_RASR_A1	RW		Alias 1 of MPU_RASR, see MPU alias register support on page B3-775
0xE000EDAC	MPU_RBAR_A2	RW		Alias 2 of MPU_RBAR, see MPU alias register support on page B3-775
0xE000EDB0	MPU_RASR_A2	RW		Alias 2 of MPU_RASR, see MPU alias register support on page B3-775
0xE000EDB4	MPU_RBAR_A3	RW		Alias 3 of MPU_RBAR, see MPU alias register support on page B3-775
0xE000EDB8	MPU_RASR_A3	RW		Alias 3 of MPU_RASR, see MPU alias register support on page B3-775
0xE000EDBC - 0xE000EDEC				Reserved

3.8 Nested Vectored Interrupt Controller (NVIC)

In power domain: M4

The Nested Vectored Interrupt Controller (NVIC) is closely integrated with the processor core to achieve low latency interrupt processing. The system design includes the Wake Up Interrupt controller which can stay on while the processor and NVIC can be put into a very low-power sleep mode, leaving the WIC to identify and prioritize interrupts and awaken items in the CPU power domain.

NVIC Features include:

- 25 External interrupts
- 8 levels of priority
- Dynamic reprioritization of interrupts.
- Priority grouping. This enables selection of preempting interrupt levels and non-preempting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
- Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.
- QuickLogic has integrated the NVIC operation with the Wake-up Interrupt Controller (WIC) module, providing ultra-low power sleep mode support where the WIC, through the PMU, can wake up a ARM core from sleep state.

The location of the NVIC in the system architecture is shown in figure below.

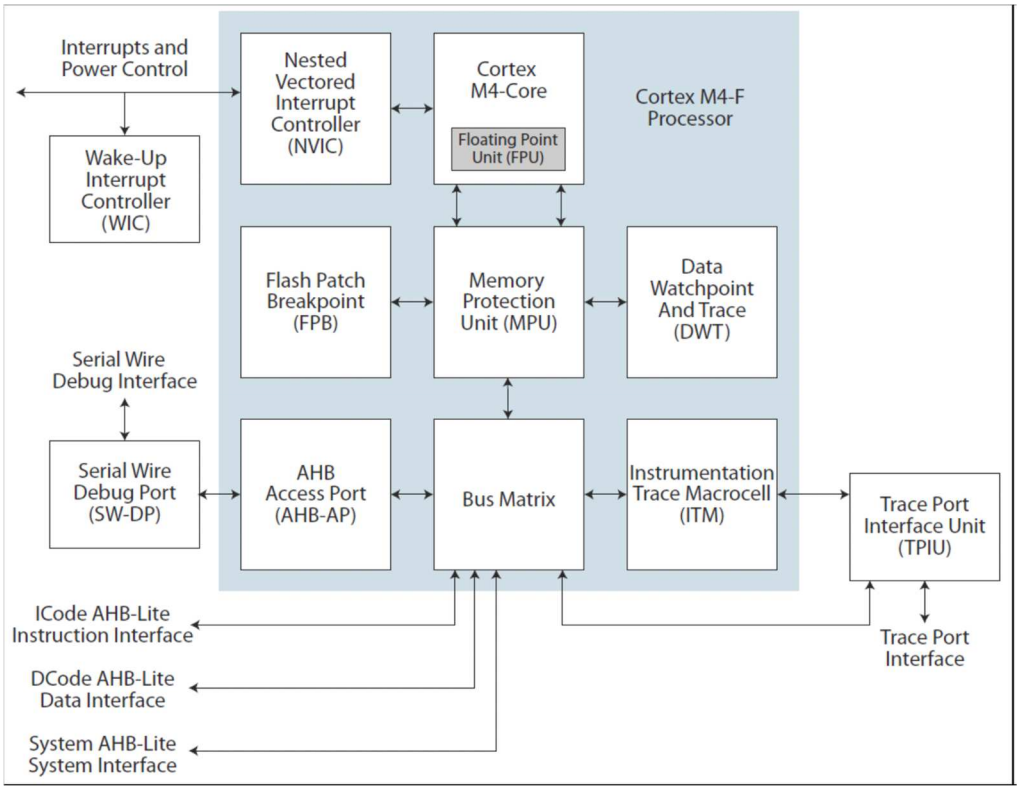

Figure 3-4: NVIC in S3 system architecture

3.8.1 List of interrupts

The following table summarizes interrupt sources.

Table 3-24: Interrupt Sources

Source	Description
Top – sensor / GPIO	Total of 8 pins
A0	AP reboot, SYS_RSTn, ADC, PMU timer, software, LDO power goo
A1	Configuration DMA (CfgDMA), SPI_1_Master
SDMA	SDMA done, SDMA error
PKFB	FIFO overflow, underflow, count threshold, access during sleep, and collision
M4 sub-system	FPU, bus timeout, UART, M4 Timer (also known as Timer1), Watchdog Timer (WDT), SRAM
Sensor / FFE sub-system	Total of 8 FFE messages, total of 16 FFE sub-system interrupts
Voice sub-system	LPSD Voice Detect, DMIC Voice Detect, DMIC Voice Off, LPSD Voice Off, DMAC0 Block Done, DMAC0 Buf Done, DMAC1 Block Done, DMAC1 Buf Done, AP PDM Clock ON, AP PDM Clock OFF, FIFO_TRIG_INT, I2S Interrupt
FPGA Fabric	Total of 4 outputs from FPGA fabric can be used.

3.8.2 NVIC Registers

Base address = 0xE000_E000

Table 3-25: NVIC Registers

Offset	Name	Description
0x100	NVIC_ISER_0	Interrupt enable for interrupts 0 - 31
0x104	NVIC_ISER_1	Interrupt enable for interrupts 32 - 63
0x180	NVIC_ICER_0	Interrupt clear for interrupts 0 - 31
0x184	NVIC_ICER_1	Interrupt clear for interrupts 32 - 63
0x200	NVIC_ISPR_0	Interrupt pending set for interrupts 0 - 31
0x204	NVIC_ISPR_1	Interrupt pending set for interrupts 32 - 63
0x280	NVIC_ICPR_0	Interrupt pending clear for interrupts 0 - 31
0x284	NVIC_ICPR_1	Interrupt pending clear for interrupts 32 - 63
0x300	NVIC_IABR_0	Reads status of active interrupts 0 - 31
0x304	NVIC_IABR_1	Reads status of active interrupts 32 - 63
0x400	NVIC_IPR_0	Interrupt priority interrupts 0 - 31
0x404	NVIC_IPR_1	Interrupt priority interrupts 32 - 63

3.9 Timers

3.9.1 M4 Timer

Power domain: M4

The M4 Timer is also known as Timer1. It is in the M4 power domain. For a timer in a separately controlled power domain, refer to the SPT.

The M4 Timer is a 32-bit down counter connected to the M4 APB and it has the following features:

- Generates an interrupt request signal, TIMERINT, when the counter reaches 0. The interrupt request is held until it is cleared by writing to the INTCLEAR Register.
- Uses the zero to one transition of the external input signal, EXTIN, as a timer enable.
- If the APB timer count reaches 0 and, at the same time, the software clears a previous interrupt status, the interrupt status is set to 1.

The external clock, EXTIN, must be slower than half of the peripheral clock because it is sampled by a double flip-flop and then goes through edge-detection logic when the external inputs act as a clock.

The following diagram illustrates the M4 Timer block architecture.

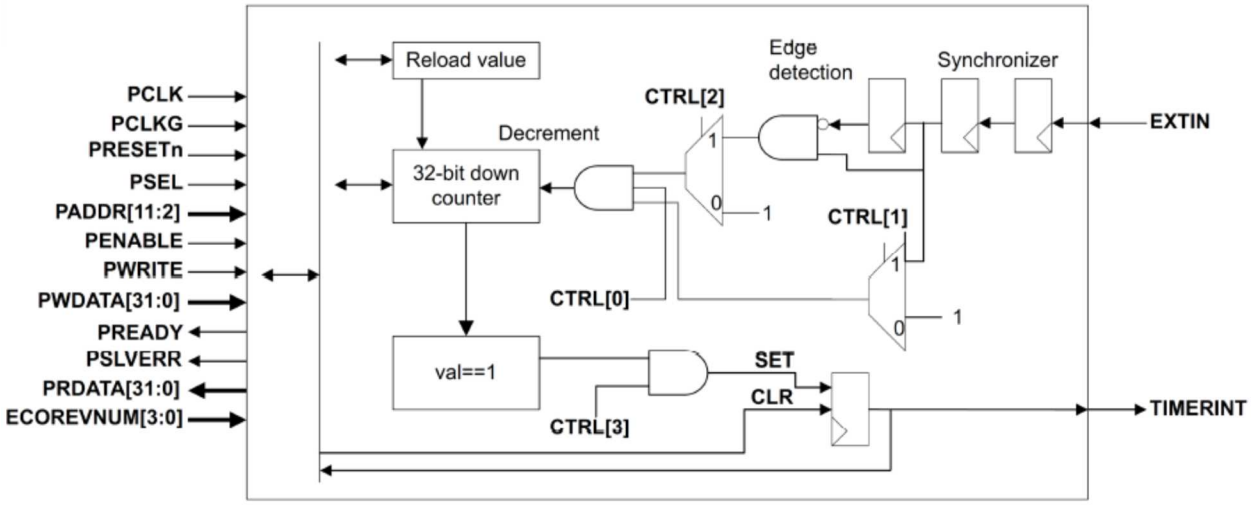

Figure 3-5: M4 Timer Block Diagram

3.9.2 M4 Timer Registers

There are 15 registers in the M4 Timer.

Base address = 0x4001_3000

Table 3-26: Timer Registers

Offset	Name	Description
0x000	CTRL	
0x004	VALUE	
0x008	RELOAD	
0x00C	INTSTATUS_INTCLEAR	
0xFD0	PID4	
0xFD4	PID5	
0xFD8	PID6	
0xFDC	PID7	
0xFE0	PID0	
0xFE4	PID1	
0xFE8	PID2	
0xFEC	PID3	
0xFF4	CID1	
0xFF8	CID2	
0xFFC	CID3	

See ARM Cortex-M System Design Kit for more information.

3.9.3 SysTick Timer

Power domain: M4

The M4 has its own system timer, the SysTick register. It is in the Ext M4 registers group. Note that the S3 includes a separate equivalent timer, the SPT, which is in its own power domain and which therefore can be kept on even when the CPU is powered down. For SysTick Timer use, refer to ARM documentation.

3.9.4 SysTick Register

Table 3-27: SysTick registers

In: EXTM4REGS (0x4000_0000)

Offset	Name	Description
0x000	SYSTICK_REG	ARM SysTick register

3.9.5 Watchdog Timer (WDT)

In power domain: M4

The Watchdog Timer provides a programmable timer interrupt.

Software can enable the watchdog timer to begin counting down to zero. At 0 count, an interrupt will be triggered. If the interrupt is not cleared, a reset is triggered.

Its interrupts can be used as wake-up events for the M4-F.

3.9.6 WDT Registers

Offset	Name/Field	Bits	Type	Default	Description
0x000	WDOGLOAD	31:0	RW	0xFFFF FFFF	The WDOGLOAD Register contains the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for WDOGLOAD is 1.
0x004	WDOGVALUE	31:0	RO	0xFFFF FFFF	The WDOGVALUE Register gives the current value of the decrementing counter.
0x008	WDOGCONTROL	31:0			The WDOGCONTROL Register enables the software to control the watchdog unit.
	RESEN	1	RW	0x0	Enable watchdog reset output, WDOGRES. Acts as a mask for the reset output. Set HIGH to enable the reset or LOW to disable the reset.
	INTEN	0	RW	0x0	Enable the interrupt event, WDOGINT. Set HIGH to enable the counter and the interrupt, or LOW to disable the counter and interrupt. Reloads the counter from the value in WDOGLAND when the interrupt is enabled, after previously being disabled.
0x00c	WDOGINTCLR	31:0	WO	X	A write of any value to the WDOGINTCLR Register clears the watchdog interrupt, and reloads the counter from the value in WDOGLOAD.
0x010	WDOGRIS	0:0	RO	0x0	The WDOGRIS Register indicates the raw interrupt status from the counter. This value is ANDed with the interrupt enable bit from the control register to create the masked interrupt, which is passed to the interrupt output pin.
0x014	WDOGMIS	0:0	RO	0x0	The WDOGMIS Register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupts status with the INTEN bit from the control register, and is the same value that is passed to the interrupt output pin. Enabled interrupt status from the counter.

Offset	Name/Field	Bits	Type	Default	Description
0xC00	WDOGLOCK	31:0	RW	0x0	The WDOGLOCK Register disables write accesses to all other registers. This is to prevent rogue software from disabling the watchdog functionality. Writing a value 0x1ACCE551 enables write access to all other registers. Writing any other value disables write accesses. A read from this register returns only the bottom bit: 0 Indicates that write access is enabled. not locked. 1 Indicates that the write access is disabled, locked
0xF00	WDOGITCR	0	RW	0x0	The WDOGITCR Register enables integration test mode. When in this more, the test output register directly controls the masked interrupt output, WDOGINT, and reset output, WDOGRES. Integration Test mode Enable [0] When set HIGH, places the watchdog into integration test mode.
0xF04	WDOGITOP	1:0	WO	0x0	Watchdog Integration Test Output Set Register When the WDOGITOP Register is in integration test mode, the values in this register directly drive the enabled interrupt output and reset output. [1] Integration Test WDOGINT value [0] Integration Test WDOCRES value
0xFD0	WDOGPERIPHID4	7:0	RO	0x04	Peripheral ID Register 4: [7:4] Block count. [3:0] jep106_c_code.
0xFD4	WDOGPERIPHID5	7:0	RO	0x0	Peripheral ID Register 5.
0xFD8	WDOGPERIPHID6	7:0	RO	0x0	Peripheral ID Register 6.
0xFDC	WDOGPERIPHID7	7:0	RO	0x0	Peripheral ID Register 7.
0xFE0	WDOGPERIPHID0	7:0	RO	0x24	Peripheral ID Register 0. [7:0] Part number[7:0].
0xFE4	WDOGPERIPHID1	7:0	RO	0xB8	Peripheral ID Register 1. [7:4] jep106_id_3_0. [3:0] Part number [11:8].
0xFE8	WDOGPERIPHID2	7:0	RO	0x1B	Peripheral ID Register 2. [7:4] Revision. [3] jedec_used. [2:0] jep106_id_6_4.
0xFEC	WDOGPERIPHID3	7:0	RO	0x00	Peripheral ID Register 3. [7:4] ECO revision number. [3:0] Customer modification number.
0xFF0	WDOGPCELLID0	7:0	RO	0x0D	Component ID Register 0.
0xFF4	WDOGPCELLID1	7:0	RO	0xF0	Component ID Register 1.
0xFF8	WDOGPCELLID2	7:0	RO	0x05	Component ID Register 2.
0xFFC	WDOGPCELLID3	7:0	RO	0xB1	Component ID Register 3.

Chapter 4. System Buses

4.1 M4 AHB bus

In power domain: M4

The M4 AHB bus is one of two AHB buses in the S3. It is a 32-bit bus operating at 80 MHz.

The M4 AHB bus connects to the system elements shown in Figure 2-1

4.2 AON AHB bus

In power domain: A0 (always on)

The AHB AON bus is one of two AHB buses in the S3. It is a 32-bit bus operating at 10 MHz.

The AHB AON bus connects to the system elements shown in Figure 2-1.

4.3 AHB2APB bus bridges

There are four separate AHB2APB bus bridges, connecting the AHB buses to APB peripherals.

They are:

Bridge #	Power domain	Bridge to these modules:
AHB2APB bridge 0	M4	UART, Watchdog timer, and Timer1 for M4
AHB2APB bridge 1	M4	Voice / audio processor sub-system
AHB2APB bridge 2	AON	eFuse
AHB2APB bridge 3	AON	I2S slave, SDMA

Chapter 5. Memory

5.1 SRAM

The S3 contains multiple blocks of SRAM.

1. The M4 SRAM is implemented in 128 Kb blocks: MEM0, MEM1, MEM2, and MEM3.
   Each 128 Kb block consists of four 32K instances, which is in the A0 always-on domain.
2. SRAM in the FPGA - see discussion of FPGA module.

5.2 FIFOs

There are several FIFOs in the S3:

1. The Packet FIFO
2. Voice / audio processor left and right channel FIFOs

This chapter covers the Packet FIFO; the Voice / audio processor FIFOs are covered in Chapter 20.

5.3 Packet FIFO

In power domain: PKFIFO

Note: The Packet FIFO is also known as the Packet FIFO Bank

Function and purpose:

The Packet FIFO interface enables the FFE to pass sensor data in the form of packets to the EOS S3 system. These packets can contain either data resulting from Sensor Fusion processing or can contain unprocessed sensor data. The format and content of each packet is entirely determined by the algorithm running on the FFE.

Its internal structure includes:

- 128 KBytes of M4-F SRAM can be used as HiFi sensor batching memory
- Multiple packet FIFOs to support the FFE to application processor/M4-F data transfers:
  - 8 KBytes packet FIFO with ring-buffer mode support
  - one 256x32 packet FIFO and two 128x32 packet FIFOs

The four packet FIFOs can generate a combined interrupt for the following exception events: overflow, underflow, count threshold, access during sleep, and collision.

Packet FIFO power domains

SRAM PF	PF_SRAM_0	Packet FIFO SRAM Instance 0	256x32 packet FIFO
	PF_SRAM_1	Packet FIFO SRAM Instance 1	128x32 packet FIFO
	PF_SRAM_2	Packet FIFO SRAM Instance 2	128x32 packet FIFO
	PF_SRAM_8K	Packet FIFO SRAM Instance 8K	8 KBytes packet FIFO with ring-buffer mode support

5.3.1 Registers

Offset	Name/Field	Bits	Type	Default	Description
0x000	FIFO_ctrl	31:0		0x0	pktFIFO Control
	pf0_en	0	RW	0x0	0 : disable, 1 : enable
	pf0_push_mux	1	RW	0x0	0 : M4, 1 : FFE
	pf0_pop_mux	2	RW	0x0	0 : M4, 1 : AP
	pf0_push_int_mux	3	RW	0x0	0 : M4, 1 : AP
	pf0_pop_int_mux	4	RW	0x0	0 : M4, 1 : AP
	pf0_ffe_sel	5	RO	0x0	Reserved
	pf1_en	8	RW	0x0	0 : disable, 1 : enable
	pf1_push_mux	9	RW	0x0	0 : M4, 1 : FFE
	pf1_pop_mux	10	RW	0x0	0 : M4, 1 : AP
	pf1_push_int_mux	11	RW	0x0	0 : M4, 1 : AP
	pf1_pop_int_mux	12	RW	0x0	0 : M4, 1 : AP
	pf1_ffe_sel	13	RO	0x0	Reserved
	pf2_en	16	RW	0x0	0 : disable, 1 : enable
	pf2_push_mux	17	RW	0x0	0 : M4, 1 : FFE
	pf2_pop_mux	18	RW	0x0	0 : M4, 1 : AP
	pf2_push_int_mux	19	RW	0x0	0 : M4, 1 : AP
	pf2_pop_int_mux	20	RW	0x0	0 : M4, 1 : AP
	pf2_ffe_sel	21	RO	0x0	Reserved
	pf8k_en	24	RW	0x0	0 : disable, 1 : enable
	pf8k_push_mux	25	RW	0x0	0 : M4, 1 : FFE
	pf8k_pop_mux	26	RW	0x0	0 : M4, 1 : AP
	pf8k_push_int_mux	27	RW	0x0	0 : M4, 1 : AP
	pf8k_pop_int_mux	28	RW	0x0	0 : M4, 1 : AP
	pf8k_ffe_sel	29	RO	0x0	Reserved
0x004	FIFO_sram_ctrl_0	31:0		0x0	SRAM Test Control 0 – **Reserved for Testing only**
	pf0_test1a	0	RW	0x0	0: test enable, 1: disable
	pf0_rmea	1	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf0_rma	5:2	RW	0x0	0: SRAM adjust timing value
	pf0_test1b	8	RW	0x0	0: test enable, 1: disable

Offset	Name/Field	Bits	Type	Default	Description
	pf0_rmeb	9	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf0_rmb	13:10	RW	0x0	0: SRAM adjust timing value
	pf1_test1a	16	RW	0x0	0: test enable, 1: disable
	pf1_rmea	17	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf1_rma	21:18	RW	0x0	0: SRAM adjust timing value
	pf1_test1b	24	RW	0x0	0: test enable, 1: disable
	pf1_rmeb	25	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf1_rmb	29:26	RW	0x0	0: SRAM adjust timing value
0x008	FIFO_sram_ctrl_1	31:0		0x0	SRAM Test Control 1 - **Reserved for Testing only**
	pf2_test1a	0	RW	0x0	0: test enable, 1: disable
	pf2_rmea	1	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf2_rma	5:2	RW	0x0	0: SRAM adjust timing value
	pf2_test1b	8	RW	0x0	0: test enable, 1: disable
	pf2_rmeb	9	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf2_rmb	13:10	RW	0x0	0: SRAM adjust timing value
	pf8k_test1a	16	RW	0x0	0: test enable, 1: disable
	pf8k_rmea	17	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf8k_rma	21:18	RW	0x0	0: SRAM adjust timing value
	pf8k_test1b	24	RW	0x0	0: test enable, 1: disable
	pf8k_rmeb	25	RW	0x0	0: SRAM timing adjust enable, 1: disable
	pf8k_rmb	29:26	RW	0x0	0: SRAM adjust timing value
0x00C	FIFO_status	31:0		0x0B0B0B0B	FIFO Status
	pf0_sram_sleep	1:0	RO	0x3	SRAM sleep status
					00 - active
					01 - Light Sleep
					10 - Deep Sleep
					11 - Shutdown
	pf0_push_int_over	2	RW1C	0x0	0 : no interrupt, 1 : interrupt set overflow (pktFIFO or FFE FIFO)
	pf0_push_int_thresh	3	RO	0x1	0 : no interrupt, 1 : interrupt set push threshold
	pf0_push_int_sleep	4	RW1C	0x0	0 : no interrupt, 1 : interrupt set push on SRAM sleep
	pf0_pop_int_under	5	RW1C	0x0	0 : no interrupt, 1 : interrupt set underflow

Offset	Name/Field	Bits	Type	Default	Description
	pf0_pop_int_thresh	6	RO	0x0	0 : no interrupt, 1 : interrupt set pop threshold
	pf0_pop_int_sleep	7	RW1C	0x0	0 : no interrupt, 1 : interrupt set pop on SRAM sleep
	pf1_sram_sleep	9:8	RO	0x3	SRAM sleep status
					00 - active
					01 - Light Sleep
					10 - Deep Sleep
					11 - Shutdown
	pf1_push_int_over	10	RW1C	0x0	0 : no interrupt, 1 : interrupt set overflow (pktFIFO or FFE FIFO)
	pf1_push_int_thresh	11	RO	0x1	0 : no interrupt, 1 : interrupt set push threshold
	pf1_push_int_sleep	12	RW1C	0x0	0 : no interrupt, 1 : interrupt set push on SRAM sleep
	pf1_pop_int_under	13	RW1C	0x0	0 : no interrupt, 1 : interrupt set underflow
	pf1_pop_int_thresh	14	RO	0x0	0 : no interrupt, 1 : interrupt set pop threshold
	pf1_pop_int_sleep	15	RW1C	0x0	0 : no interrupt, 1 : interrupt set pop on SRAM sleep
	pf2_sram_sleep	17:16	RO	0x3	SRAM sleep status
					00 - active
					01 - Light Sleep
					10 - Deep Sleep
					11 - Shutdown
	pf2_push_int_over	18	RW1C	0x0	0 : no interrupt, 1 : interrupt set overflow (pktFIFO or FFE FIFO)
	pf2_push_int_thresh	19	RO	0x1	0 : no interrupt, 1 : interrupt set push threshold
	pf2_push_int_sleep	20	RW1C	0x0	0 : no interrupt, 1 : interrupt set push on SRAM sleep
	pf2_pop_int_under	21	RW1C	0x0	0 : no interrupt, 1 : interrupt set underflow
	pf2_pop_int_thresh	22	RO	0x0	0 : no interrupt, 1 : interrupt set pop threshold
	pf2_pop_int_sleep	23	RW1C	0x0	0 : no interrupt, 1 : interrupt set pop on SRAM sleep

Offset	Name/Field	Bits	Type	Default	Description
	pf8k_sram_sleep	25:24	RO	0x3	SRAM sleep status
					00 - active
					01 - Light Sleep
					10 - Deep Sleep
					11 - Shutdown
	pf3_push_int_over	26	RW1C	0x0	0 : no interrupt, 1 : interrupt set overflow (pktFIFO or FFE FIFO)
	pf3_push_int_thresh	27	RO	0x1	0 : no interrupt, 1 : interrupt set push threshold
	pf3_push_int_sleep	28	RW1C	0x0	0 : no interrupt, 1 : interrupt set push on SRAM sleep
	pf3_pop_int_under	29	RW1C	0x0	0 : no interrupt, 1 : interrupt set underflow
	pf3_pop_int_thresh	30	RO	0x0	0 : no interrupt, 1 : interrupt set pop threshold
	pf3_pop_int_sleep	31	RW1C	0x0	0 : no interrupt, 1 : interrupt set pop on SRAM sleep
0x010	pf0_push_ctrl	31:0		0x0	FIFO 0 Push Control
	pf0_push_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf0_push_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf0_push_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] overflow
					[2] threshold
					[3] push on SRAM sleep
	pf0_push_thresh	24:16	RW	0x0	PUSH counter threshold (x32 count)
0x014	pf0_pop_ctrl	31:0		0x0	FIFO 0 Pop Control
	pf0_pop_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf0_pop_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf0_pop_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] underflow
					[2] threshold
					[3] pop on SRAM sleep
	pf0_pop_thresh	24:16	RW	0x0	POP counter threshold (x32 count)
0x018	pf0_cnt	31:0		0x01008000	FIFO 0 Count
	pf0_pop_cnt	8:0	RO	0x0	FIFO 0 Pop Count (x32 count) Number of available entries for pop

Offset	Name/Field	Bits	Type	Default	Description
	pf0_empty	15	RO	0x1	FIFO 0 Empty
	pf0_push_cnt	24:16	RO	0x100	FIFO 0 Push Count (x32 count) Number of available entries for push
	pf0_full	31	RO	0x0	FIFO 0 Full
0x01C	pf0_data_reg	31:0		0x0	FIFO 0 Push/POP Data Register
	pf0_data_reg	31:0	RW	0x0	FIFO 0 Push/POP Data Register
0x020	pf1_push_ctrl	31:0		0x0	FIFO 1 Push Control
	pf1_push_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf1_push_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf1_push_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] overflow
					[2] threshold
					[3] push on SRAM sleep
	pf1_push_thresh	23:16	RW	0x0	PUSH counter threshold (x32 count)
0x024	pf1_pop_ctrl	31:0		0x0	FIFO 1 Pop Control
	pf1_pop_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf1_pop_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf1_pop_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] underflow
					[2] threshold
					[3] pop on SRAM sleep
	pf1_pop_thresh	23:16	RW	0x0	POP counter threshold (x32 count)
0x028	pf1_cnt	31:0		0x00808000	FIFO 1 Count
	pf1_pop_cnt	7:0	RO	0x0	FIFO 1 Pop Count (x32 count) Number of available entries for pop
	pf1_empty	15	RO	0x1	FIFO 1 Empty
	pf1_push_cnt	23:16	RO	0x80	FIFO 1 Push Count (x32 count) Number of available entries for push
	pf1_full	31	RO	0x0	FIFO 1 Full
0x02C	pf1_data_reg	31:0		0x0	FIFO 1 Push/POP Data Register
	pf1_data_reg	31:0	RW	0x0	FIFO 1 Push/POP Data Register
		0			
0x030	pf2_push_ctrl	31:0		0x0	FIFO 2 Push Control
	pf2_push_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable

Offset	Name/Field	Bits	Type	Default	Description
	pf2_push_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf2_push_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] overflow
					[2] threshold
					[3] push on SRAM sleep
	pf2_push_thresh	23:16	RW	0x0	PUSH counter threshold (x32 count)
0x034	pf2_pop_ctrl	31:0		0x0	FIFO 2 Pop Control
	pf2_pop_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf2_pop_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf2_pop_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] underflow
					[2] threshold
					[3] pop on SRAM sleep
	pf2_pop_thresh	23:16	RW	0x0	POP counter threshold (x32 count)
0x038	pf2_cnt	31:0		0x00808000	FIFO 2 Count
	pf2_pop_cnt	7:0	RO	0x0	FIFO 2 Pop Count (x32 count) Number of available entries for pop
	pf2_empty	15	RO	0x1	FIFO 2 Empty
	pf2_push_cnt	23:16	RO	0x80	FIFO 2 Push Count (x32 count) Number of available entries for push
	pf2_full	31	RO	0x0	FIFO 2 Full
0x03C	pf2_data_reg	31:0		0x0	FIFO 2 Push/POP Data Register
	pf2_data_reg	31:0	RW	0x0	FIFO 2 Push/POP Data Register
0x040	pf8k_push_ctrl	31:0		0x0	FIFO 8k Push Control
	pf8k_push_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf8k_push_sleep_type	1	RW	0x0	0 : DS, 1 : SD
	pf8k_push_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] overflow
					[2] threshold
					[3] push on SRAM sleep
	pf8k_push_thresh	28:16	RW	0x0	PUSH counter threshold (x16 count)
0x044	pf8k_pop_ctrl	31:0		0x0	FIFO 8k Pop Control
	pf8k_pop_sleep_en	0	RW	0x0	0 : sleep disable, 1 : sleep enable
	pf8k_pop_sleep_type	1	RW	0x0	0 : DS, 1 : SD

Offset	Name/Field	Bits	Type	Default	Description
	pf8k_pop_int_en	4:2	RW	0x0	0 : mask, 1 : enable
					[1] underflow
					[2] threshold
					[3] pop on SRAM sleep
	pf8k_fifo_pkt_mode	5	RW	0x0	FIFO Packet Mode
	pf8k_ring_buff_mode	6	RW	0x0	Ring buffer mode
	pf8k_pop_thresh	28:16	RW	0x0	POP counter threshold (x16 count)
0x048	pf8k_cnt	31:0		0x10008000	FIFO 8k Count
	pf8k_pop_cnt	12:0	RO	0x0	FIFO 8k Pop Count (x16 count) Number of available entries for pop
	pf8k_empty_busy	15	RO	0x1	FIFO 8k Empty / Busy
	pf8k_push_cnt	28:16	RO	0x1000	FIFO 8k Push Count (x16 count) Number of available entries for push
	pf8k_full	31	RO	0x0	FIFO 8k Full
0x04C	pf8k_data_reg	31:0		0x0	FIFO 8k Push/POP Data Register
	pf8k_data_reg	16:0	RW	0x0	FIFO 8k Push/POP Data Register in ringbuffer mode, [16] is treated as SOP (start-of-packet) by autodrain logic
	pf8k_push_eop	17	WO	0x0	FIFO 8k Push EOP (end-of-packet)
0x050	fifo_collision_intr	31:0	RO		**Reserved**
0x054	fifo_collision_intr_en	31:0	RO		**Reserved**

Chapter 6. DMA

There are several DMA controllers in the S3:

- System DMA controller - for system DMA transfers over the AHB buses
- VP DMAC - the DMA controller in the voice / audio sub-system, used to transfer data in and out of the left and right channel FIFOs in that module
- CM DMAC - the DMA controller in the Communication Manager, dedicated to use by the TLC.

6.1 System DMA Controller (SDMA)

In power domain: SDMA

SDMA is an implementation of the ARM DMA Controller IP with SDMA_BRIDGE "wrapper" and SRAM block that may be powered down.

6.1.1 SDMA Registers

Base address = 0x4000_C000

Offset	Name	Bits	Type	Default	Description
0x000	dma_status	31:0	RO	0x0	
	test_status	32:28	RO	0x1	To reduce the gate count, you can configure the Controller to exclude the integration test logic. Read as: 0x0 = Controller does not include the integration test logic. 0x1 = Controller includes the integration test logic. 0x2-0xF = undefined.
	Reserved	27:21	RO	0x0	Undefined
	chnis_minus	20:16	RO	0xf	Number of available DMA channels minus one, for example: b00000 = Controller configured to use 1 DMA channel b00001 = Controller configured to use 2 DMA channels b00010 = Controller configured to use 3 DMA channels . . . b11111 = Controller configured to use 32 DMA channels
	Reserved	15:8	RO	0x0	Undefined
	state	7:4	RO	0x0	Current state of the Control State Machine, and the state can be one of the following: b0000 = Idle b0001 = Reading channel Controller data b0010 = Reading source data end pointer b0011 = Reading destination data end pointer b0100 = Reading source data b0101 = Writing destination data b0110 = Waiting for DMA request to clear b0111 = Writing channel Controller data b1000 = Stalled b1001 = Done
	Reserved	3:1	RO	0x0	Undefined

Offset	Name	Bits	Type	Default	Description
	master_enable	0	RO	0x0	Status of the Controller: 0 = Controller is disabled 1 = Controller is enabled
0x004	dma_cfg	31:0	WO		
	Reserved	31:8	WO		Undefined; write as zeros
	chnl_prot_ctrl	7:5	WO		Sets the AHB-Lite protection by controlling the HPROT[3:1] signal levels as follows: Bit [7] Controls HPROT[3] to indicate if a cacheable access is occurring. Bit [6] Controls HPROT[2] to indicate if a bufferable access is occurring. Bit [5] Controls HPROT[1] to indicate if a privileged access is occurring. Note When bit [n] = 1 then the corresponding HPROT is HIGH. When bit [n] = 0 then the corresponding HPROT is LOW.
	Reserved	4:1	WO		Undefined; write as zeros
	master_enable	0	WO		Enable for the controller: 0 = disables the controller 1 = enables the controller
0x008	ctrl_base_ptr	31:0	RW	0x0	
	ctrl_base_ptr	31:9	RW	0x0	Pointer to the base address of the primiary data structure
	Reserved	8:0	RO		Undefined; writes as zero
0x00C	alt_ctrl_base_ptr	31:0	RO	0x0	
	alt_ctrl_ptr	31:0	RO	0x100	Base address of the alternate data structure
0x010	dma_waitonreq_status	31:0	RO	0x0	
	Reserved	31:16	RO	0x0	Channel 31-16: not used
	dma_waitonreq_status	15:0	RO	0x0	Channel wait on request status. Read as: Bit [C] = 0 dma_waitonreq[C] is LOW. Bit [C] = 1 dma_waitonreq[C] is HIGH.
0x014	chnl_sw_request	31:0	WO		
	Reserved	31:16	WO		Channel 31-16 not used
	chnl_sw_request	15:0	WO		Set the appropriate bit to generate a software DMA request on the corresponding DMA channel. Write as: Bit [C] = 0 Does not create a DMA request for channel C. Bit [C] = 1 Creates a DMA request for channel C. Writing to a bit where a DMA channel is not implemented does not create a DMA request for that channel.

Offset	Name	Bits	Type	Default	Description
0x018	chnl_useburst_set	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_useburst_set	15:0	RW1S	0x0	Returns the useburst status, or disables dma_sreq[C] from generating DMA requests. Read as: Bit [C] = 0 DMA channel C responds to requests that it receives on dma_req[C] or dma_sreq[C]. The controller performs 2R, or single, bus transfers. Bit [C] = 1 DMA channel C does not respond to requests that it receives on dma_sreq[C]. The controller only responds to dma_req[C] requests and performs 2R transfers. Write as: Bit [C] = 0 No effect. Use the chnl_useburst_clr Register to set bit [C] to 0. Bit [C] = 1 Disables dma_sreq[C] from generating DMA requests. The controller performs 2R transfers. Writing to a bit where a DMA channel is not implemented has no effect.
0x01C	chnl_useburst_clr	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_useburst_clr	15:0	RW1S	0x0	Set the appropriate bit to enable dma_sreq[] to generate requests. Write as: Bit [C] = 0 No effect. Use the chnl_useburst_set Register to disable dma_sreq[] from generating requests. Bit [C] = 1 Enables dma_sreq[C] to generate DMA requests. Writing to a bit where a DMA channel is not implemented has no effect
0x020	chnl_req_mask_set	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used

Offset	Name	Bits	Type	Default	Description
	chnl_req_mask_set	15:0	RW1S	0x0	Returns the request mask status of dma_req[] and dma_sreq[], or disables the corresponding channel from generating DMA requests. Read as: Bit [C] = 0 External requests are enabled for channel C. Bit [C] = 1 External request is disabled for channel C. Write as: Bit [C] = 0 No effect. Use the chnl_req_mask_clr Register to enable DMA requests. Bit [C] = 1 Disables dma_req[C] and dma_sreq[C] from generating DMA requests. Writing to a bit where a DMA channel is not implemented has no effect.
0x024	chnl_req_mask_clr	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_req_mask_clr	15:0	RW1S	0x0	Set the appropriate bit to enable DMA requests for the channel corresponding to dma_req[] and dma_sreq[]. Write as: Bit [C] = 0 No effect. Use the chnl_req_mask_set Register to disable dma_req[] and dma_sreq[] from generating requests. Bit [C] = 1 Enables dma_req[C] or dma_sreq[C] to generate DMA requests. Writing to a bit where a DMA channel is not implemented has no effect.
0x028	chnl_enable_set	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_enable_set	15:0	RW1S	0x0	Returns the enable status of the channels, or enables the corresponding channels. Read as: Bit [C] = 0 Channel C is disabled. Bit [C] = 1 Channel C is enabled. Write as: Bit [C] = 0 No effect. Use the chnl_enable_clr Register to disable a channel. Bit [C] = 1 Enables channel C. Writing to a bit where a DMA channel is not implemented has no effect.
0x02C	chnl_enable_clr	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used

Offset	Name	Bits	Type	Default	Description
	chnl_enable_clr	15:0	RW1S	0x0	Set the appropriate bit to disable the corresponding DMA channel. Write as: Bit [C] = 0 No effect. Use the chnl_enable_set Register to enable DMA channels. Bit [C] = 1 Disables channel C. Writing to a bit where a DMA channel is not implemented has no effect.
0x030	chnl_pri_alt_set	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_pri_alt_set	15:0	RW1S	0x0	Returns the channel control data structure status, or selects the alternate data structure for the corresponding DMA channel. Read as: Bit [C] = 0 DMA channel C is using the primary data structure. Bit [C] = 1 DMA channel C is using the alternate data structure. Write as: Bit [C] = 0 No effect. Use the chnl_pri_alt_clr Register to set bit [C] to 0. Bit [C] = 1 Selects the alternate data structure for channel C. Writing to a bit where a DMA channel is not implemented has no effect.
0x034	chnl_pri_alt_clr	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_pri_alt_clr	15:0	RW1S	0x0	Set the appropriate bit to select the primary data structure for the corresponding DMA channel. Write as: Bit [C] = 0 No effect. Use the chnl_pri_alt_set Register to select the alternate data structure. Bit [C] = 1 Selects the primary data structure for channel C. Writing to a bit where a DMA channel is not implemented has no effect.
0x038	chnl_priority_set	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used

Offset	Name	Bits	Type	Default	Description
	chnl_priority_set	15:0	RW1S	0x0	Returns the channel priority mask status, or sets the channel priority to high. Read as: Bit [C] = 0 DMA channel C is using the default priority level. Bit [C] = 1 DMA channel C is using a high priority level. Write as: Bit [C] = 0 No effect. Use the chnl_priority_clr Register to set channel C to the default priority level. Bit [C] = 1 Channel C uses the high priority level. Writing to a bit where a DMA channel is not implemented has no effect.
0x03C	chnl_priority_clr	31:0			
	Reserved	31:16	RO	0x0	Channel 31-16 not used
	chnl_priority_clr	15:0	RW1S	0x0	Set the appropriate bit to select the default priority level for the specified DMA channel. Write as: Bit [C] = 0 No effect. Use the chnl_priority_set Register to set channel C to the high priority level. Bit [C] = 1 Channel C uses the default priority level. Writing to a bit where a DMA channel is not implemented has no effect.
0x040	err_clr	31:0			
	Reserved	31:1	RO	Ox0	Underfined
	err_clr	0	WO		Returns the status of dma_err, or sets the signal LOW. Read as: 0 = dma_err is LOW 1 = dma_err is HIGH. Write as: 0 = No effect, status of dma_err is unchanged. 1 = Sets dma_err LOW. For test purposes, use the err_set register to set dma_err HIGH.
0xFD0	Periph_id_4	31:0	RO	0x4	Peripheral Identification 4
0xFE0	Periph_id_0	31:0	RO	0x30	Peripheral Identification 0
0xFE4	Periph_id_1	31:0	RO	0xB2	Peripheral Identification 1
0xFE8	Periph_id_2	31:0	RO	0xB	Peripheral Identification 2

Offset	Name	Bits	Type	Default	Description
0xFEC	Periph_id_3	31:0	RO	0x0	Peripheral Identification 3
0xFF0	Pcell_id_0	31:0	RO	0xD	Primecell identification 0
0xFF4	Pcell_id_1	31:0	RO	0xF0	Primecell identification 1
0xFF8	Pcell_id_2	31:0	RO	0x5	Primecell identification 2
0xFFC	Pcell_id_3	31:0	RO	0xB1	Primecell identification 3

6.1.2 SDMA_BRIDGE Registers

Base address = 0x4000_D000

Table 6-1: SDMA_BRIDGE Registers

Offset	Name	Description
0x000	DMA_REQ_REG	Send Single or Burst Requests to SDMA Register
0x004	DMA_WAITONREQ_REG	Send WAITONREQ Signal to SDMA Register
0x008	DMA_ACTIVE_REG	SDMA DMA_ACTIVE Signal Status Register
0x00C	SDMA_PWRDN_CNT	SDMA Power down Event Threshold Register
0x010	SDMA_SRAM_CTRL	SDMA SRAM Timing Adjust Register

Table 6-2: SDMA_Bridge Registers Descriptions

Offset	BIT FIELD	bit	R/W/C	Default	Description
0x000	Dma_req_reg	31:0			
	Reserved	31:27	RO	0x0	
	dma_sreq	26:16	WO		Single DMA request send to SDMA
	Reserved	15:11	RO	0x0	
	dma_req	10:0	WO		Burst DMA request send to SDMA
0x004	dma_waitonreq_reg	31:0			
	Reserved	31:11	RO	0x0	
	dma_waitonreq_reg	10:0	RW		Wait on Request signal send to SDMA
0x008	dma_active_reg	31:0			
	Reserved	31:11	RO	0x0	
	dma_active	10:0	RO	0x0	Dma_active signal status from SDMA
0x00C	dma_pwrdwn_cnt	31:0			
	Reserved	31:16	RO	0x0	
	dma_pwrdwn_cnt	15:0	RW	0xffff	sdma power down event threshold. If sdma stays in idle cycles longer than the threshold, sdma will be automaticlly put into power down to save power.
0x010	sdma_sram_ctrl	31:0			**Reserved**
	Reserved	31:6	RO	0x0	
	sdma_sram_rm	5:2	RW	0x0	SRAM adjust timing value
	sdma_sram_rme	1	RM	0x0	1: SRAM timing adjust enable; 0 = disable
	Sdma_sram_test1	0	RO		Reserved

6.2 VP DMAC

The diagram below shows the physical location of the VP DMAC in the Voice / audio processor sub-system.

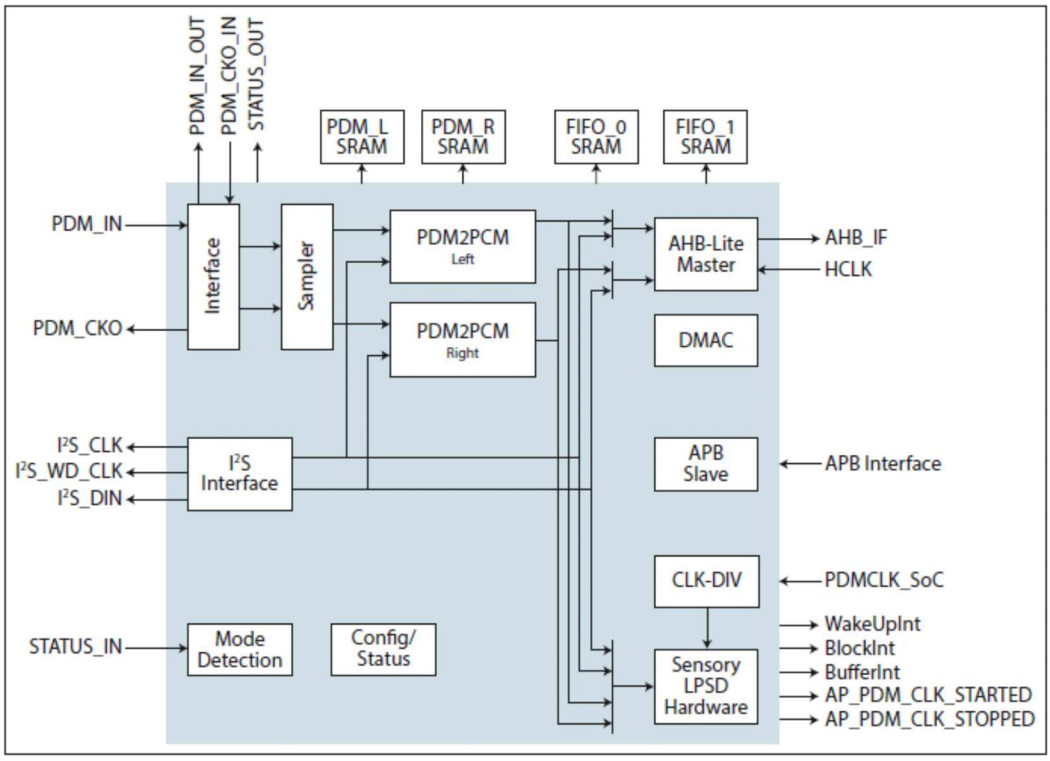

Figure 6-1: VP DMAC in the Voice / audio processor

6.2.1 VP DMAC registers

Offset	Name/Field	Bits	Type	Default	Description	Notes
0x000	DMA_ctrl	31:0		0x0	DMA Control : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.	
	dma_start	0	RW1S	0x0	write a 1: Enable, write a 0: no affect, reads dma_enb	
	dma_stop	1	RW1C	0x0	write a 1: Stop DMA and disable, clears DMA_DONE, write a 0: no affect, reads dma_done	
	dma_ahb_sel	2	RW	0x0	0: DMA to AHB, 1: DMA to header register	
	dma_hsel	3	RO	0x0	1: AHB hsel is asserted, 0: not asserted	
	dma_htrans_0	4	RO	0x0	1: AHB htrans[0] is asserted, 0: not asserted	
	dma_htrans_1	5	RO	0x0	1: AHB htrans[1] is asserted, 0: not asserted	
	dma_hready	6	RO	0x1	1: AHB hready is asserted, 0: not asserted	
	dma_xfr_pending	7	RO	0x0	1: DMA transfer is pending, 0: nothing pending	
	bridge_xfr_pending	8	RO	0x0	1: AHB bridge transfer is pending, 0: nothing pending	
0x004	DMA_destination_address	31:0		0x0	DMA destination address : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.	1
	dma_dest_addr	31:0	RW	0x0	DMA output data address : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.	1
0x008	DMA_transfer_count	31:0		0x0	DMA transfer count in frames (8 bit) (minus 1) : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.	1
	dma_xfr_cnt	25:0	RW	0x0	number of frames for DMA to transfer (minus 1). Max SPI transfer count is 2^16=64k frames	1
0x00C	cfg_flash_header	31:0		0x0	Header values read from EEPROM : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.	

Offset	Name/Field	Bits	Type	Default	Description	Notes
	dma_boot_xfr_size	15:0	RO	0x0	number of double words (8 bytes) for the SPI to transfer (minus 1)	
	dma_spi_clk_divide	23:16	RO	0x0	SPI data clock out divides the ssi_clk (value*2)	
	dma_device_id	31:24	RO	0x0	Device ID	
0x010	DMA_intr	31:0		0x0	DMA interrupts	
	dma_herror	0	RW1C	0x0	1: hresp=1, 0: hresp didn't go to 1, write one to clr	
	rx_data_available	1	RO	0x0	1: rx threshold was hit, 0:threshold was not hit. This is before external mask bit.	
	ahb_bridge_fifo_overflow	2	RW1C	0x0	1: A ahb FIFO bridge overflow occurred, 0: no overflow occurred	
	spim_ssi_txe_intr	3	RO	0x0	SPIM Transmit FIFO empty	
	spim_ssi_txo_intr	4	RO	0x0	SPIM Transmit FIFO overflow	
	spim_ssi_rxf_intr	5	RO	0x0	SPIM Receive FIFO threshold	
	spim_ssi_rxo_intr	6	RO	0x0	SPIM Receive FIFO overflow	
	spim_ssi_rxu_intr	7	RO	0x0	SPIM Receive FIFO underflow	
	spim_ssi_mst_intr	8	RO	0x0	SPIM master interrupt	
0x014	DMA_intr_mask	31:0		0x7	DMA interrupt mask	
	dma_herror_mask	0	RW	0x1	1: disable interrupt, 0:enable interrupt	
	rx_data_available_mask	1	RW	0x1	1: mask rx data available 0:don't mask	
	ahb_bridge_fifo_overflow mask	2	RW	0x1	1: Mask the ahb FIFO bridge overflow, 0: interrupts are enabled	
0x018	Config_State_Machine_Delay	31:0			This is the delay value used in the config state machine. It is used for both deep sleep wakeup delay and between retries.	
	delay_reg	15:0	RW	0x1F4	Delay value used in the config SM state machine. This is clocked at the APB pclk frequency. Default to decimal 500.	
Notes:						
	1) Values are in bytes.					

Chapter 7. Interrupt hardware

This chapter covers the WIC, which is integrated with the PMU and is a top-level system component which is in the AON domain. For discussion of the Nested Vectored Interrupt Controller which is part of the (power-switched) ARM core.

7.1 Wake-up Interrupt Controller (WIC)

In power domain: M4

The Wakeup Interrupt Controller (WIC) peripheral can detect an interrupt and wake the M4 processor or the FFE from deep sleep mode.

The WIC is not programmable; it does not have any registers and it operates entirely driven from hardware signals.

When the WIC is enabled and the processor enters deep sleep mode, the PMU can power down most of the M4 processor. When the WIC receives an interrupt, a number of clock cycles are required to wake up the processor and restore its state so it can process the interrupt. Because of this, interrupt latency is increased in deep sleep mode.

Please refer to ARM Cortex-M4 design notes related to the WIC operation.

7.1.1 Registers

The WIC's functionality is hardwired and not programmable. There is no direct register control needed.

Chapter 8. On-chip LDO power regulation

8.1 LDO

There are two Low Drop-Out (LDO) linear regulators integrated in the EOS S3. They provide flexible input voltage ranges and programmable voltage outputs.

See LDO section for register description.

8.1.1 Design case: Internal Voltages Supplied by Two LDOs

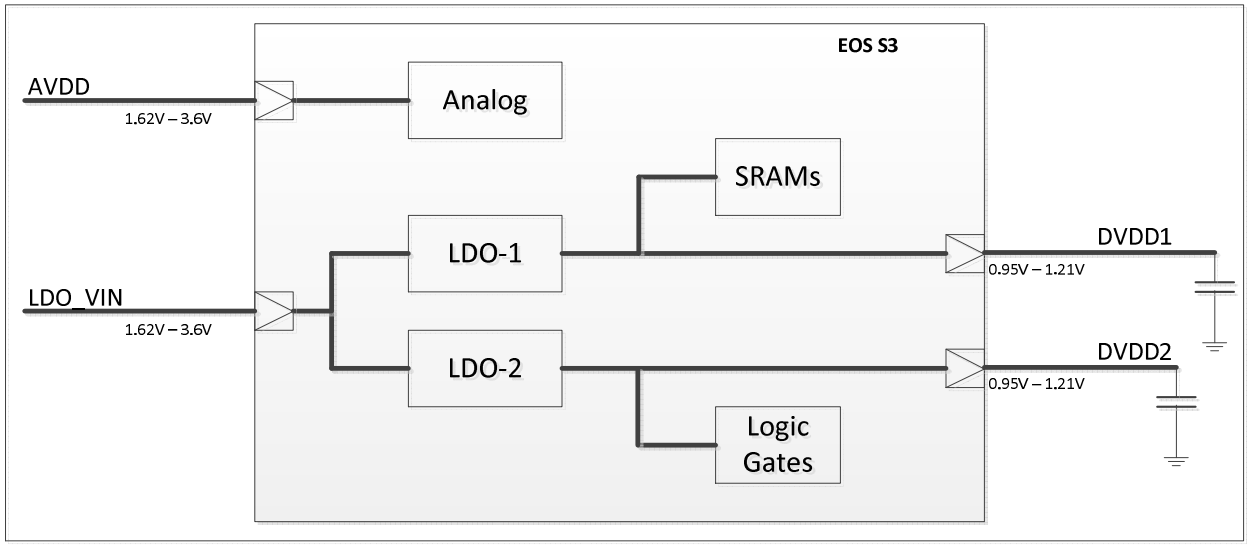

Figure 8-1: LDO configuration option 1

In this design case, all SRAMs will be supplied by LDO-1 and logic gates will be provided by LDO-2.

8.1.2 Design case: Internal Voltages Supplied by Single LDO

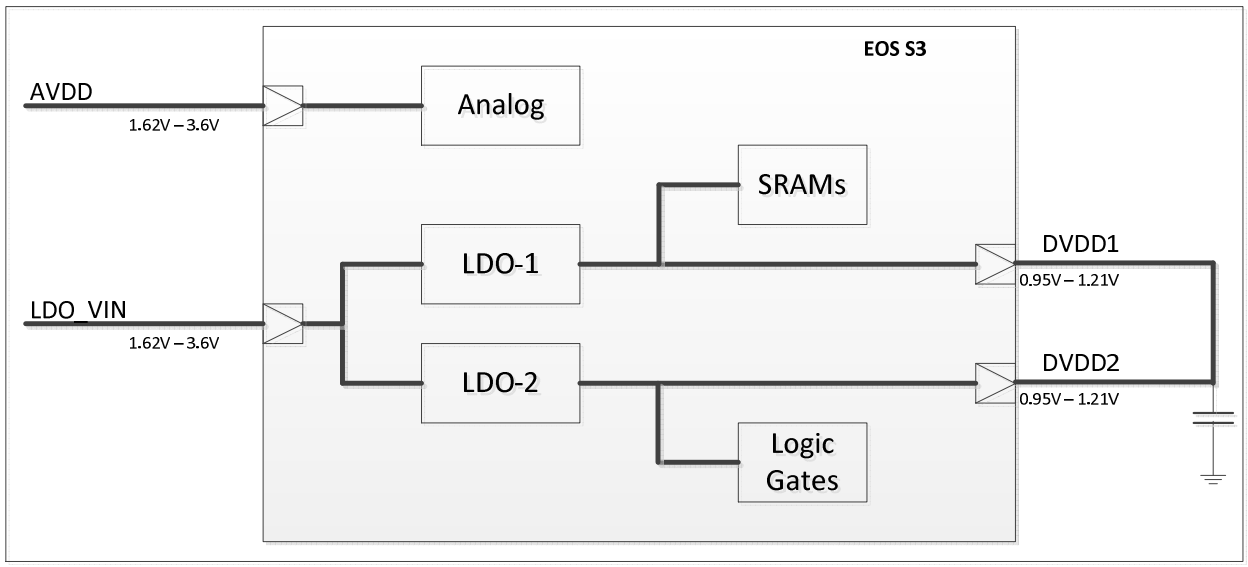

Figure 8-2: LDO configuration option 3

The VDD1 and VDD2 pads will be tied together on the board and the SRAMs, and logic gates will be provided by LDO-1.

After Power on, both LDO-1 and LDO-2 are enabled. LDO-2 must be disabled by firmware once the system boots up.

8.1.3 Design case: Internal Voltages Supplied by External Source

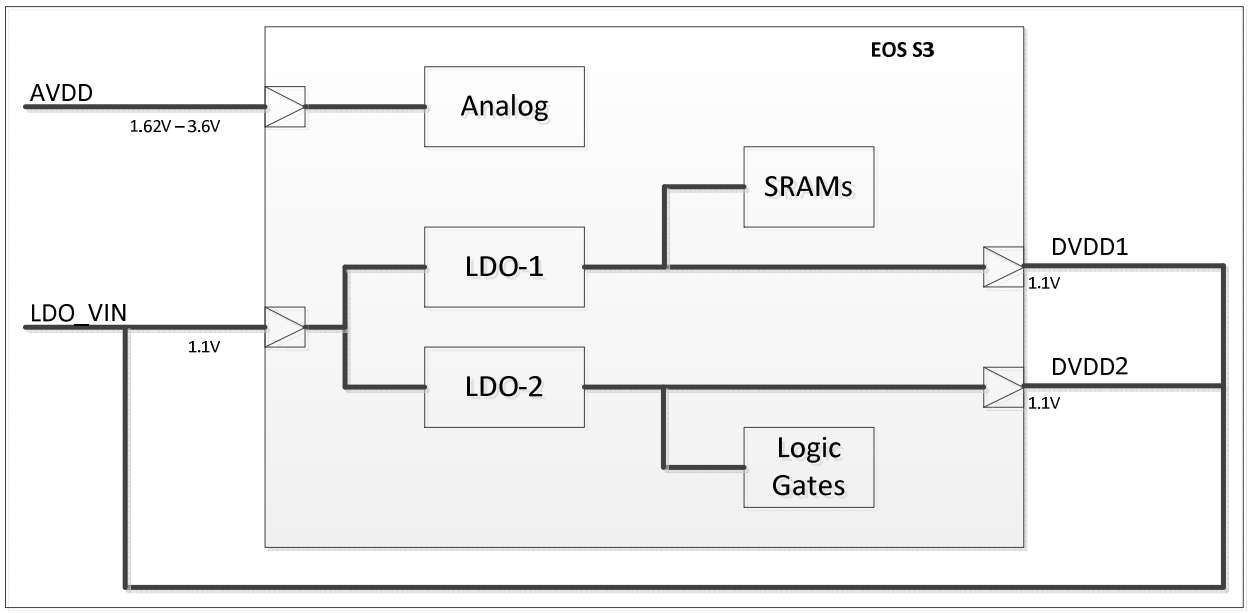

Figure 8-3: LDO configuration option 3

The SRAM and logic gates power will be supplied through LDO_VIN instead from internal LDO-1 or LDO-2 modules. The VDD1, VDD2 and LDO_VIN must be connected to the same external supply voltage. In other words, LDO_VIN = VDD1 = VDD2 = 1.1V.

Internal LDO-1 and LDO-2 are dis-able in this design case.

Note: Applied voltage to LDO_VIN must start at 1.1V to bring the S3 output of reset; the voltage can be adjusted to 1V after the device is configured to reduce power consumption.

Chapter 9. Power Domains in the S3

An S3 power domain is a grouping of circuits that may be turned on / off by firmware or hardware event. Items in a domain each have their own power configuration, status, entry and exit trigger conditions. There are 31 major power domains (including the M4 group of SRAMs).

9.1 Main power domains in the S3

The following 31 major power domains are available in the S3.

Table 9-1: Power Domains in the S3

Power Domain		Description
AON A0		Always ON power domain (does not turn off). The top-level elements in this group are: the AP interface (Communication Manager's TLC and SPI_Slave), SYS_RSTn, GPIO, ADC, PMU (including WIC), and LDO.
A1		The top-level elements in the A1 domain are power switchable (not always-on). It encompasses these top-level components: Configuration DMA, SPI_1_Master.
AD0		Audio DMA
AD1		PDM_LEFT
AD2		PDM_RIGHT
AD3		LPSD
AD4		I2S_MASTER
AD5		Voice APB interface
FB		Fabric (FPGA) including RAM within the FPGA
FFE		Flexible Fusion Engine (Sensor Processing Sub-System)
FUSE		eFuse
I2S		Inter IC Sound Slave (I2S_Slave)
M4		ARM Cortex-M4-F processor and "local" peripherals (SRAM not included)
(M4SRAM)		SRAM in the Cortex-M4-F sub-system, as follows:
M4SRAM	M4S0	M4 SRAM Instance M4S0
	M4S1	M4 SRAM Instance M4S1
	M4S2	M4 SRAM Instance M4S2
	M4S3	M4 SRAM Instance M4S3
	M4S4	M4 SRAM Instance M4S4
	M4S5	M4 SRAM Instance M4S5
	M4S6	M4 SRAM Instance M4S6
	M4S7	M4 SRAM Instance M4S7
	M4S8	M4 SRAM Instance M4S8
	M4S9	M4 SRAM Instance M4S9
	M4S10	M4 SRAM Instance M4S10
	M4S11	M4 SRAM Instance M4S11
	M4S12	M4 SRAM Instance M4S12 (Attached to AON bus) †
	M4S13	M4 SRAM Instance M4S13 (Attached to AON bus) †
	M4S14	M4 SRAM Instance M4S14 (Attached to AON bus) †
	M4S15	M4 SRAM Instance M4S15 (Attached to AON bus) †
PF		Packet FIFO logic
SDMA		System DMA

† M4S[15:12] are attached to AON bus, however, these power domains may be turned off individually.

9.1.1 AON A0 Always ON power domain

The AON A0 domain is always ON. The top level modules that do not have head switches (for shutoff) belong to this power domain.

System elements in it are:

- PMU (including WIC; however. WIC is only activated when M4 processor is in deep sleep state)
- the AP interface (the TLC)
- SYS_RSTn
- GPIO
- ADC
- LDO_0
- LDO_1
- SPI_Slave

9.1.2 A1 domain

System elements in this domain are top-level system components but which have head switches enabling their being turned off. These include:

- Configure DMA (CfgDMA)
- Configure SM (serial wire debug port)
- SPI_1_Master (Flash Controller)

9.1.3 M4 power domain

This comprises the M4 and its closely-related functional elements

9.1.4 M4 SRAMs power domain

This comprises the four M4 SRAMs (128KB). Each SRAM has four banks (32KB each) and each bank is in its own power domain. There are thus 16 M4 SRAM power domains.

The first three SRAMs are attached to the M4 AHB bus. The fourth SRAM is connected to both the M4 AHB bus and the AON AHB bus. This means that it can be used even when the M4 and its bus are asleep.

Note that other SRAM in the system, such as SRAM in the FPGA, or SRAM in the Voice / audio processor, is in its own separate SRAM domain.

9.2 SRAM Power Domains

In addition to the 31 major power domains which contain the M4 SRAM domains, there are 19 other, separate, SRAM power domains (6 SRAMs in Voice/Audio subsystem, 8 SRAMs in FFE, 4 SRAMs in packet FIFO, and 1 SRAM in SDMA) with their own shutdown and deep sleep control.

Table 9-2: SRAM Power Domains

SRAM Power Domain		Description
Voice / Audio subsystem PDM	AD1_SRAM_R0	Audio PDM Left Channel SRAM Instance 0
	AD1_SRAM_R1	Audio PDM Left Channel SRAM Instance 1
	AD1_SRAM_R2	Audio PDM Left Channel SRAM Instance 2
	AD2_SRAM_R0	Audio PDM Right Channel SRAM Instance 0
	AD2_SRAM_R1	Audio PDM Right Channel SRAM Instance 1
	AD2_SRAM_R2	Audio PDM Right Channel SRAM Instance 2
FFE	FFE_SRAM_CM0	FFE SRAM (8Kx40)
	FFE_SRAM_CM1	FFE SRAM (2Kx40)
	FFE_SRAM_DM0	FFE SRAM for (4Kx32)
	FFE_SRAM_DM1	FFE SRAM for (Virtual of DM0)
	FFE_SRAM_SM0	FFE SRAM for SM0
	FFE_SRAM_SM1	FFE SRAM for SM1
PF	PF_SRAM_0	Packet FIFO SRAM Instance 0
	PF_SRAM_1	Packet FIFO SRAM Instance 1
	PF_SRAM_2	Packet FIFO SRAM Instance 2
	PF_SRAM_8K	Packet FIFO SRAM Instance 8K
SDMA		System DMA SRAM

Chapter 10. Clock oscillators, system clocks, and timers

10.1 Introduction

This chapter describes hardware aspects of system master clocks and timers dependent on these clocks.

10.2 Oscillators

There can be three sources for the S3 'master' clock:

1. The "Slow oscillator" - a circuit using an external tuning-fork type 32768 Hz crystal to generate the 32768 Hz “slow” clock for the Real-Time Clock (RTC).
2. An external clock source can be used instead of a crystal for the slow oscillator source.
3. The high-speed oscillator (alternatively called either the high speed oscillator, or HSO, or HS_Osc in this manual) can either be derived from the slow oscillator, or an external source can be used.

10.2.1 The Slow Oscillator

10.2.1.1 Design using crystal

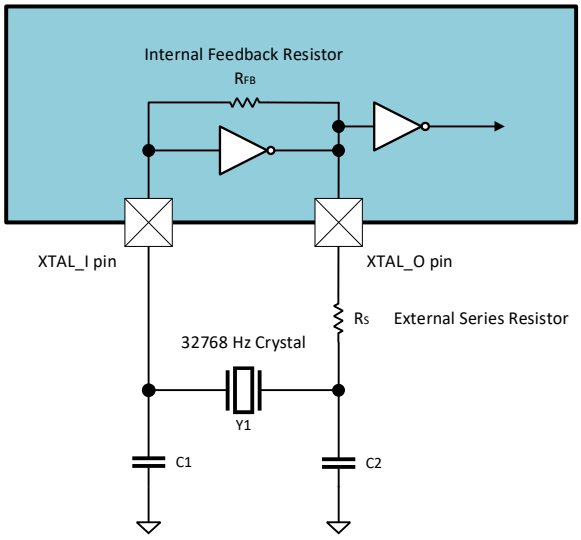

Figure 10-1: 32768 Hz crystal oscillator circuit

Follow crystal manufacturer’s recommended value for external capacitors C1 and C2 taking into consideration stray capacitance due to PCB traces and contact. A fine tuning step may be necessary to obtain stable and accurate oscillation. An external series resistor may be needed to limit the drive level of the crystal to force the oscillator to start at the fundamental frequency rather than at overtones. An internal feddback resistor (Rs) is implemented to start the oscillation.

10.2.1.2 Design using external clock instead of crystal

An external square wave clock source may be used instead of a crystal. A single ended external clock may be supplied to XTAL_I or XTAL_O. A differential clock (180 degrees out of phase) may be supplied to XTAL_I and XTAL_O. There is a register option to select either 32768 Hz or 16384 Hz. Refer to the DC specification for using external OSC.

10.2.2 The Fast Oscillator

10.2.2.1 Frequency selection

The fast oscillator source provides the internal HSO_Clock signal. The fast oscillator (known as the High Speed Oscillator, or HSO, or HS_Osc in this manual and in the software) can be generated internally from the slow oscillator or sourced from an external driver. When generated internally it is calibrated by a Delay-Locked Loop (DLL) with reference frequency of 32768 (or 16384) Hz.

The internal high speed oscillator starts up at 76.97 MHz when referenced to a slow oscillator of 32768 Hz. When 16384 Hz is supplied for the slow clock, the HSO initial frequency is 38.49 MHz

Program settings must ensure a value from 2.195 MHz to 80 MHz for the HSO_Clock.

See AC Electrical Specifications for more details.

See Oscillator Control 0 regiser and Oscillator Control 1 register for programming information.

10.2.2.2 Selection of source for internal HSO_Clock

Start-up configuration options allow the system design to select use of either the internal HS_Osc module or an external FCLK source to source the S3 internal HSO_Clock. See Figure 10-2.

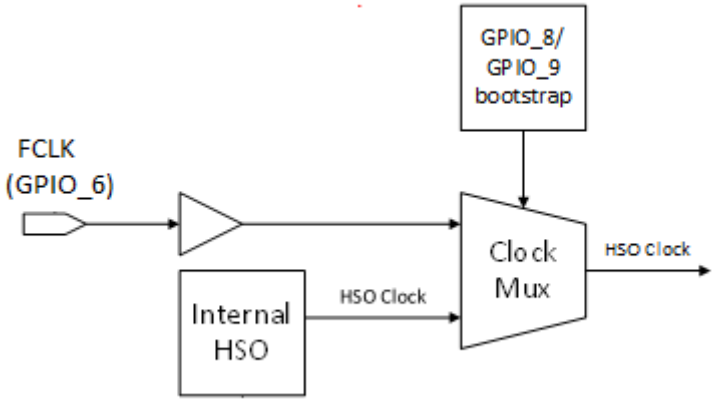

Figure 10-2: HSO clock source

Chapter 11. Clock domains, clock chains, and the CRU

11.1 Clock sources

On chip there are total 19 clock domains, and most clock domains (clock paths) that can be driven by three different clock sources:

- Real-Time Clock (RTC - 32 KHz) – using external 32768Hz clock
- Oscillator Clock (OSC – 2MHz - 80MHz)
- Fast Clock Driving from GPIO 6 (FCLK)

Note: The FCLK mode is selected by bootstrap GPIO 9. This will replace the oscillator as a clock source. This is a static configuration and cannot be changed until the next chip reset. Please refer to System Consideration section for bootstrap details.

Clock domains are numbered in sequence (e.g., C00, C01, C02, and so on). Individual clock paths in a single clock domain are numbered (e.g., P0_A0, P1_A0, and so on). There can be similar clock path names, but they must be in different clock domains (e.g., C00 P0_A0, C01 P0_A0). Table 11-1 shows the maximum frequency per clock, the clock domain the clock is in, and clock paths derived from each clock.

Three serial interfaces have clock inputs.

- SPI Slave interface's clock is driven in on PAD 16. This clock is C00.
- Serial Wire Debug (SWD) clock is driven in on PAD 14 (when bootstrap PAD 8 = 0) or PAD 45 (when bootstrap PAD 8 = 1). This clock is CS.
- I2S Slave Clock is in on PAD 31. This clock is C32.

Each clock path can be gated and divided independently by programming their clock gating and dividing registers (0x40004000 - 0x40004144) in CRU.

Slow clock may be either 32768 Hz or 16384 Hz. See SPT_CFG register.

11.2 Clock divider chains and the CRU

The EOS S3 platform contains 19 clock domains, and most clock domains have their own register-controlled divider. Each clock domain has one or more clock paths that it supports. Clock paths can be individually gated. See table below for a full listing of the clocks domains.

Table 11-1: Clock Listing

Block Name(s)	Max Frequency**	Clock Name	Clock Path	Notes
Always-On Domain				
AP to SPI_Slave, SPI_Slave to TLC, TLC to PKT_FIFO clock	20 MHz	C00	P0_A0	
TLC clock to AHB Switch (AHB clock)	10 MHz	C01	P0_A0	AHB clock must be greater than or equal to one half of the SPI slave clock (in line above)
AHB Switch, Reg Bank, other blocks connected to the switch on the Always-on power domain	10 MHz	C01	P0_A0	
A1 Domain				
CfgSM, CfgDMA, SPI_Master (APB clock)	40 MHz	C02	P0_A1	
CfgDMA (AHB clock)	10 MHz	C01	P4_A1	
SPI_Master serial data clock	20 MHz	C02	n/a	The SPI_Master serial clock frequency is one half of the C02 clock frequency.
I2S Domain				
I2S Slave (DA pin)	10 MHz	C32	P0_I2S	
I2S slave APB interface	10 MHz	C01	P5_I2S	
efuse Domain				
Efuse (APB clock)	40 MHz	C02	P2_FUSE	
SDMA Domain				
AHB2APB, SDMA (AHB clock)	10 MHz	C01	P6_SDMA	
SDMA SRAM Domain				
SDMA SRAM	10 MHz	C01	P1_A0	
FFE Domain				
AHB switch	10 MHz	C01	P3_FFE	
X1 clk	10 MHz	C08	X1_P0_FFE	
X4 clk	40 MHz	C08	X4_P0_FFE	
For A0	10 MHz	C08	X1_P2_A0	
for PKT FIFO	10 MHz	C08	X1_P3_PF	
Packet FIFO Domain				
PKT FIFO (AHB clock)	10 MHz	C01	P2_PF	
PKT FIFO (TLC clock)	20 MHz	C00	P0_A0	

Block Name(s)	Max Frequency**	Clock Name	Clock Path	Notes
PKT FIFO (FPGA clock)	20 MHz	C41	n/a	
PKT FIFO (FFE clock)	10 MHz	C08	X1_P3_PF	
M4 Subsystem Domain				
M4-Complex: M4 subsystem, M4-AHB switch (AHB clock)	80 MHz	C10	HCLK_P0_M4 FCLK_P0_M4	
M4-Complex: UART, WDT1, Timer1 (APB clock 0)	10 MHz	C11	P0_M4	
M4-Complex: to Voice SS and CFG_CTL (APB clock 1)	80 MHz	C10	FCLK_PS_AD0	
M4 CFG_CTL to FPGA (APB clock)	10 MHz	C09	P2_FB	
A0 (AHB clock M4)	80 MHz	C10	FCLK_P6_A0	
M4 SWD (DA pin)	20 MHz	CS	P0_M4	
M4 SRAM Domain				
M4 SRAM 0 (AHB clock)	80 MHz	C10	FCLK_P1_MS0	
M4 SRAM 1 (AHB clock)	80 MHz	C10	FCLK_P2_MS1	
M4 SRAM 2 (AHB clock)	80 MHz	C10	FCLK_P3_MS2	
M4 SRAM 3 (AHB clock)	80 MHz	C10	FCLK_P4_MS3	
Voice Subsystem Domain				
Voice SS (AHB clock)	80 MHz	C10	FCLK_P5_AD0	The ratio between the Voice SS AHB and APB clock must be an integer ratio, such as 1-1, 1-2, 1-4
Voice SS (APB clock)	10 MHz	C09	P0_AD5	
PDM left clock	5 MHz	C30	P0_AD1	
PDM right clock	5 MHz	C30	P1_AD2	
I2S clock	5 MHz	C30	P2_AD4	
LPSD clock	1 MHz	C31	P3_AD3	
FPGA Domain				
FPGA	72 MHz	C16	P0_FB	
FPGA	72 MHz	C21	P0_FB	
AHB2WB for FPGA clock	10 MHz	C40	P0_FB	
FPGA to Packet FIFO clock	20 MHz	C41	P0_FB	

Notes:

- Slow clock is 32768 Hz (or 16384 Hz)
- C20 is always clocked by slow clock
- Max Frequency is ± 10% with DVDD @ 1.1V

On EOS S3, the CRU register block (0x40004000 - 0x40004144) controls the following clock paths functions:

- Clock Sources – Select which clock source.
- Clock Gating – Enable or disable clock gating.
- Clock Divider – Divide down the clock from clock source.

Bootstrapping GPIO 9 = 1 is used to select the FCLK from GPIO 6. This means setting GPIO 9 = 1 during SYS_RSTN assertion and de-assertion.

There are 3 different types CRU programming registers:

1. **CLK_Control_x_1** and **CLK_SWITCH_FOR_x** registers
    a. Select the clock source between oscillator clock and 32 KHz clock.
    b. Also there are three SYNC down clock sources are C10, C08X1, and C31.
    c. These registers are marked in figure 11-1.
    d. *When bootstrap PAD9 = 1, FCLK replaces the oscillator clock as source. External input clock will come from PAD6.*
2. **CLK_Control_x_0** registers
    a. Used for the clock divider ratio (1~512).
    b. These registers are shown in figure 11-1.
    c. The SYNC down clocks have their own clock divider ratio registers:
        i. CRU_C01_CLK_DIV (1~16)
        ii. CRU_C09_CLK_DIV (1~16)
        iii. CRU_C31_CLK_DIV (1~16)
        iv. Divided by 4 (fixed) for C08X1 clock from C08X4
3. **CLKxx_GATE** registers
    a. For clock path gating control.
    b. Set 0 to turn off the clock. Or set 1 to turn on clock. Please refer to EOS S3 Register map for exact bit field.
    c. These registers are marked red in figure 11-1.

Below in Figure 11-1 is an overview of the clock tree diagram. The CRU clock path registers shown below are the offset addresses from 0x40004000. The clock domains are numbered on the right, alongside their maximum frequency.

Please note from Figure 11-1, the following clock domain relationships.

- Clocks C01 and C09 are derived from C10.
- C08X1 is divided by 4 of C08X4. This is a fixed relationship is for FFE clocks.
- C31 (LPSD clock) is divided down from C30. This LPSD clock is typically 512 KHz.

Changing the OSC frequency will impact all clock domains which select OSC as a clock source. Some or all clocks require specific frequencies for proper operation, so keep that in mind when reprogramming the OSC frequency. Please refer to later section on how to change OSC frequency.

Fabric (FPGA) has 3 inputs clocks (C16, C21, C02) and 2 outputs clocks (C40, C41). FPGA must be configured by Host or M4 before it can use the input clocks or drive clock outputs. Please refer to setup FPGA section for more details.

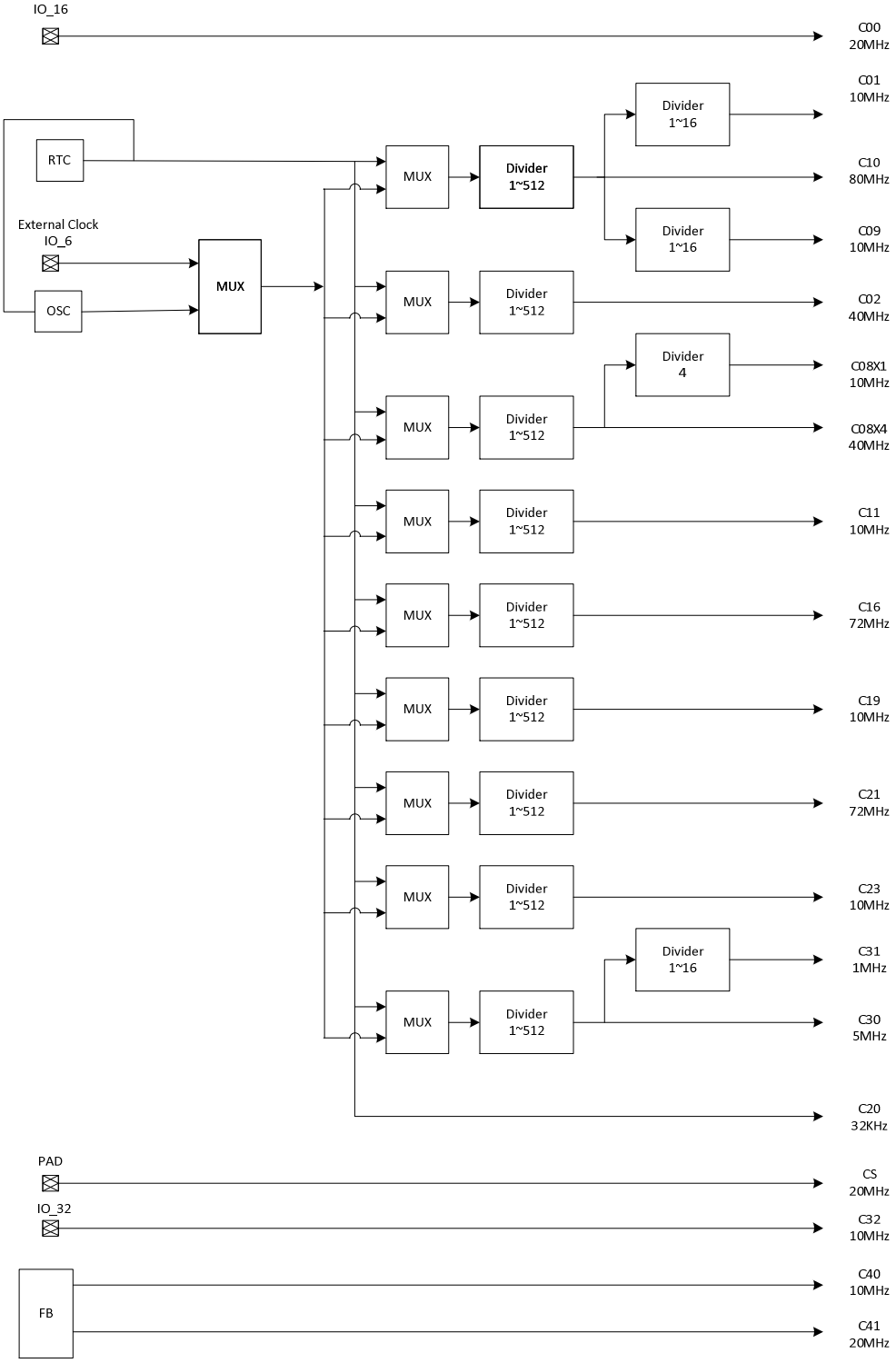

Figure 11-1: Clock tree

Table 11-2: External Clock Source Clock Domains

Name	CRU Alpha Name	Slow Clock Select	Modules/Comments
C00	–	–	Path 0: to Slave interface; clock source: Slave SPI clock from pin
C32	–	–	Path 0: I2S Slave clock from pin
CS	S	–	Clock S: SWD clock from pin

Table 11-3: Clock Domains Generated by FB

Number	CRU Alpha Name	Modules/Comments
C40	–	Clock output from FB to AHB-WB Bridge
C41	–	Clock output from FB to Packet FIFO ASYNC FIFO 1

Dividers

The Clock Dividers can divide the source clock frequency from 1/2 to 1/512. (2, 3, 4… 512) and the output clock duty cycle is 50%.

Each clock domain has its corresponding reset path. All of them will be asynchronously asserted and sync released except the reset path for the Communication Manager SPI/TLC modules since there is no SPI clock running when system reset is asserted. The reset for M4 core will not be de-asserted until clock is toggling for at least four 4 cycles.

11.2.1 Clock Gating

The basic philosophy for power reduction through clock gating is: Turn off clock when that clock is not used to reduce dynamic power consumption.

11.3 Clock Reset Unit

The Clock Reset Unit (CRU) generates clock domains used by all sub-systems.

SW reset may be asserted for following blocks by writing to appropriate register bits.

11.3.1 CRU Control Register Background Information

This section summarizes functions and background information for the clock divider, clock source, and gating controls.

Letter name aliases (A, B, C, and so on) used in the following section are identified in Table 11-1 and also repeated in Table 11-4.

CLK_CONTROL_x

The CLK_CONTROL / Cxx_CLK_DIV register selects:

- Divider on / bypass (where x is A_0, B_0, C_0, D_0, F_0, G_0, H_0, I_0, or PMU)
- Divider ratio (where x is A_0, B_0, C_0, D_0, F_0, G_0, H_0, I_0, or PMU)
- Clock source (where x is F_1, or I_1)

Cxx_CLK_DIV

The Cxx_CLK_DIV register selects:

- Secondary divider ratio (where xx is 01, 09, or 31)

Cxx_CLK_GATE

The Cxx_CLK_GATE (CRU_GENERAL or CS_CLK_GATE) register enables / disables clock gating control for each clock path in a clock domain Cxx

Clock domains are: C01, C02, C08, C09, C10_FCLK, C11, C16, C19, C21, C30_C31, or CS

CLK_DIVIDER_CLK_GATING

The CLK_DIVIDER_CLK_GATING register enables / disables clock for dividers:

- Dividers are: A, B, C, D, F, G, H I, or J

CLK_SWITCH_FOR_x / CLK_CONTROL_y_1

The CLK_SWITCH_FOR_x (where x is B, C, D, G, H, or J) register selects clock source: either high speed clock (OSC) or 32768 Hz clock.

The CLK_CONTROL_y_1 (where y is A, F, or I) register selects clock source: either high speed clock (OSC) or 32768 Hz clock.

In general, high speed clock is selected by default except clock domain C20 is always slow clock.

The following table summarizes the registers to control the clock source selection, divider, and gating if supported.

Table 11-4: Registers for Clock Source Selection, Divider, and Gating

Name	Divider Name	Slow Clock Select Register	Divider Register		Clock Gating Register
			Ratio	Clock Gating*	
C00	–	–	–	–	CRU_GENERAL ‡
C01	(A)	SAME AS C10	c01_clk_div[3:0] ^	_DIV[4] ^	C01_CLK_GATE
C02	B	CLK_SWITCH_FOR_B †	CLK_Control_B_0	[1]	C02_CLK_GATE
C08	C	SAME AS C08X4	1/4	[2]	C08_X1_CLK_GATE
		CLK_SWITCH_FOR_C	CLK_Control_C_0		C08_X4_CLK_GATE
C09	(A)	SAME AS C10	c09_clk_div[3:0] ^	_DIV[4] ^	C09_CLK_GATE
C10	A	CLK_CONTROL_A_1	CLK_Control_A_0	[0]	C10_FCLK_GATE %
C11	D	CLK_SWITCH_FOR_D	CLK_Control_D_0	[3]	C11_CLK_GATE
C16	F	CLK_Control_F_1	CLK_Control_F_0	[5]	C16_CLK_GATE
C19	H	CLK_SWITCH_FOR_H †	CLK_Control_H_0	[7]	C19_CLK_GATE
C20	–	Always slow	–	–	–
C21	I	CLK_Control_I_1	CLK_Control_I_0	[8]	C21_CLK_GATE
C23	J	CLK_SWITCH_FOR_J	CLK_Control_PMU	[9]	none
C30	G	CLK_SWITCH_FOR_G	CLK_Control_G_0	[6]	C30_C31_CLK_GATE
C31	(G)	SAME AS C31	c13_clk_div[3:0] ^	_DIV[4] ^	
C32	–	–	–	–	–
CS	S	–	–	–	CS_CLK_GATE

Note:

* Bit field for CLK_DIVIDER_CLK_GATING register.

† Do not use CLK_CONTROL_B_1 or CLK_CONTROL_H_1 for clock source selection.

‡ Internal SPI Clock (C00) will be gated if SPI chip select pin (CS) is de-asserted.

^ C01_CLK_DIV, C09_CLK_DIV, and C31_CLK_DIV are 4 bit divider. Divider is gated by bit [4] of this register.

% Clock gating is for FCLK paths 0 – 6. HCLK path does not have clock gating control.

11.3.2 CRU Registers

Base address = 0x4000_4000

Table 11-5: CRU Registers

Offset	Name	Description
0x000	CLK_CONTROL_A_0	Clock 10 divider control
0x004	CLK_CONTROL_A_1	Clock 10 clock source control
0x008	CLK_CONTROL_B_0	Clock 2 divider control
0x010	CLK_CONTROL_C_0	Clock 8 divider control
0x014	CLK_CONTROL_D_0	Clock 11 divider control
0x018	CLK_CONTROL_E_0	Clock 12 divider control
0x020	CLK_CONTROL_F_0	Clock 16 divider control
0x024	CLK_CONTROL_F_1	Clock 16 clock source control
0x028	CLK_CONTROL_G_0	Clock 30 divider control
0x02C	CLK_CONTROL_H_0	Clock 19 divider control
0x034	CLK_CONTROL_I_0	Clock 21 divider control
0x038	CLK_CONTROL_I_1	Clock 21 clock source control
0x03C	HIGH_SPEED_CLOCK_SOURCE	Read only: status of high speed clock source selection by bootstrap pin.
0x040	C01_CLK_GATE	Clock 1 gating control
0x044	C02_CLK_GATE	Clock 2 gating control
0x048	C08_X4_CLK_GATE	Clock 8 X4 gating control
0x04C	C08_X1_CLK_GATE	Clock 8 X1 gating control
0x050	C10_FCLK_GATE	Clock 10 gating control
0x054	C11_CLK_GATE	Clock 10 gating control
0x05C	CS_CLK_GATE	Clock S gating control
0x064	C16_CLK_GATE	Clock 16 gating control
0x06C	C19_CLK_GATE	Clock 19 gating control
0x070	C21_CLK_GATE	Clock 21 gating control
0x080	PF_SW_RESET	Software reset for PF block (FIFO_0, FIFO_1, FIFO_2, FIFO_8K, ASYNC_FIFO_0, and Peripheral)
0x084	FFE_SW_RESET	Software reset for FFE block
0x088	FB_SW_RESET	Software reset for FB block (C02, C09, C16, and C21 clock domains)
0x08C	A1_SW_RESET	Software reset for A1 block (SPT and CfgSM: Master SPI, FIFO, DMA, and AHB Master)
0x090	AUDIO_MISC_SW_RESET	Software reset for AUDIO MISC block (AD0, AD1, AD2, AD3, AD4, AD5, DMA, and I2S)
0x094	FB_MISC_SW_RST_CTL	Enable software reset for FB MISC block (AHBWB and PFAFIFO1)
0x100	CLK_CONTROL_PMU	PMU clock gating control divider control
0x104	CRU_GENERAL	SPICLK_ALWAYS_ON control
0x108	CRU_DEBUG	Debug select control
0x110	C01_CLK_DIV	Clock 1 divider control
0x114	C09_CLK_DIV	Clock 9 divider control
0x118	C31_CLK_DIV	Clock 31 divider control
0x11C	C09_CLK_GATE	Clock 9 gating control
0x120	C30_C31_CLK_GATE	Clocks 30 and 31 gating control

Offset	Name	Description
0x124	CLK_DIVIDER_CLK_GATING	Clock Dividers (A, B, C, D, F, G, H, I, and J) clock gating control
0x130	CLK_SWITCH_FOR_B	Clock 2 (eFuse, FB, A1 (Including CFGSM)) clock source control
0x134	CLK_SWITCH_FOR_C	Clock 8 X4 (FFE X4 clk) clock source control
0x138	CLK_SWITCH_FOR_D	Clock 11 (To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER) clock source control
0x13C	CLK_SWITCH_FOR_H	Clock 19 (JTM) clock source control
0x140	CLK_SWITCH_FOR_J	Clock 23 (PMU clk gating control) clock source control
0x144	CLK_SWITCH_FOR_G	Clock 30 (PDM LEFT/RIGHT Clk, I2S_MASTER) clock source control

Table 11-6: CLK_CONTROL_A_0 Register

OFFSET	NAME				
	Bit Field	bit	R/W/C	Default	Description
0x000	CLK_CONTROL_A_0	31:0			For Clock 10 (SYNC Up on A0 and AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix and Trace block)
	RESERVED	31:10			
	DISABLE_CLOCK_	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK__RATIO	8:0	RW	0x4	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 6.

Table 11-7: CRU CLK_CONTROL_A_1 Register

OFFSET	CLK_CONTROL_A_1				
	Bit Field	bit	R/W/C	Default	Description
0x004	CLK_CONTROL_A_1	31:0			For Clock 10 (SYNC Up on A0 and AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix and Trace block)
	RESERVED	31:10			
	CLOCK_SOURCE_SELECTION	1:0	RW	0x0	2'b00 → High Speed/Divided Clock 2'b01 → 32KHz Others → Reserved Please Note: If Reserved Value has been programmed, then the clock will be turn off and system cannot be recovered without System Reset or POR.

Table 11-8: CRU CLK_CONTROL_B_0 Register

OFFSET	CLK_CONTROL_B_0				
	Bit Field	bit	R/W/C	Default	Description
0x008	CLK_CONTROL_B_0	31:0			For Clock 2 (eFuse, FB, A1 (Including CFGSM))
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0x4	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 6.

Table 11-9: CRU CLK_CONTROL_C_0 Register

OFFSET	CLK_CONTROL_C_0				
	Bit Field	bit	R/W/C	Default	Description
0x010	CLK_CONTROL_C_0	31:0			For Clock 8 X4 (FFE X4 clk)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0x4	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 6.

Table 11-10: CRU CLK_CONTROL_D_0 Register

OFFSET	CLK_CONTROL_D_0				
	Bit Field	bit	R/W/C	Default	Description
0x014	CLK_CONTROL_D_0	31:0			For Clock 11 (To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0xE	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 16.

Table 11-11: CRU CLK_CONTROL_F_0 Register

OFFSET	CLK_CONTROL_F_0				
	Bit Field	bit	R/W/C	Default	Description
0x020	CLK_CONTROL_F_0	31:0			For Clock 16 (FB)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0xE	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 16.

Table 11-12: CRU CLK_CONTROL_F_1 Register

OFFSET	CLK_CONTROL_F_1				
	Bit Field	bit	R/W/C	Default	Description
0x024	CLK_CONTROL_F_1	31:0			For Clock 16 (FB)
	RESERVED	31:2			
	CLOCK_SOURCE_SELECTION	1:0	RW	0x0	2'b00 → High Speed/Divided Clock 2'b01 → 32KHz Others → Reserved Note: If Reserved Value has been programmed, then the clock will be turn off and system cannot be recovered without System Reset or POR.

Table 11-13: CRU CLK_CONTROL_G_0 Register

OFFSET	CLK_CONTROL_G_0				
	Bit Field	bit	R/W/C	Default	Description
0x028	CLK_CONTROL_G_0	31:0			For Clock 30 (PDM LEFT/RIGHT clk, I2S Master clk)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0x0E	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x1E, Divided by 32. Note: This Default value is 0x0E, but the Clock Divider output is 1/32 of source clock by default. The issue only happens on default value.

Table 11-14: CRU CLK_CONTROL_H_0 Register

OFFSET	CLK_CONTROL_H_0				
	Bit Field	bit	R/W/C	Default	Description
0x02C	CLK_CONTROL_H_0	31:0			For Clock 19 (JTM)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0xE	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 16.

Table 11-15: CRU CLK_CONTROL_I_0 Register

OFFSET	CLK_CONTROL_I_0				
	Bit Field	bit	R/W/C	Default	Description
0x034	CLK_CONTROL_I_0	31:0			For Clock 21 (FB - additional clk)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0xE	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default 0x4, Divided by 16.

Table 11-16: CRU CLK_CONTROL_I_1 Register

OFFSET	CLK_CONTROL_I_1				
	Bit Field	bit	R/W/C	Default	Description
0x038	CLK_CONTROL_I_1	31:0			For Clock 21 (FB - additional clk)
	RESERVED	31:2			
	CLOCK_SOURCE_SELECTION	1:0	RW	0x0	2'b00 → High Speed/Divided Clock 2'b01 → 32KHz Others → Reserved Please Note: If Reserved Value has been programmed, then the clock will be turn off and system cannot be recovered without System Reset or POR.

Table 11-17: CRU HIGH_SPEED_CLOCK_SOURCE Register

OFFSET	HIGH_SPEED_CLOCK_SOURCE				
	Bit Field	bit	R/W/C	Default	Description
0x03C	HIGH_SPEED_CLOCK_SOURCE	31:0			
	RESERVED	31:1			
	CLOCK_SOURCE_SELECTION	0	RO	0x0	1'b0 High Speed Clock Source is from OSC 1'b1 the source is from PAD ==> Now Only OSC is support. Keep it 0 ==> The PAD clock can be selected by Boot Strap pin.

Table 11-18: CRU C01_CLK_GATE Register

OFFSET	C01_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x040	C01_CLK_GATE	31:0			
	RESERVED	31:10			
	PATH_9_GATING_CONTROL	9	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running For SPT.
	PATH_8_GATING_CONTROL	8	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To Debug controller
	RESERVED	7	RW	0x1	RESERVED
	PATH_6_GATING_CONTROL	6	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To SDMA
	PATH_5_GATING_CONTROL	5	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To I2S module Inside A1
	PATH_4_GATING_CONTROL	4	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To AHB2APB Bridge /CFG DMA Bridge inside A1 , Allow M4 to configure SPI Master to load the code
	PATH_3_GATING_CONTROL	3	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FFE
	PATH_2_GATING_CONTROL	2	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To Packet FIFO
	PATH_1_GATING_CONTROL	1	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To SDMA SRAM
	PATH_0_GATING_CONTROL	0	RO	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To A0

Table 11-19: CRU C02_CLK_GATE Register

OFFSET	C02_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x044	C02_CLK_GATE	31:0			
	RESERVED	31:2			
	PATH_2_GATING_CONTROL	2	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To eFuse
	PATH_1_GATING_CONTROL	1	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FB
	PATH_0_GATING_CONTROL	0	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To A1 (Including CFGSM)

Table 11-20: CRU C08_X4_CLK_GATE Register

OFFSET	C08_X4_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x048	C08_X4_CLK_GATE	31:0			
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FFE X4 clk

Table 11-21: CRU C08_X1_CLK_GATE Register

OFFSET	C08_X1_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x04C	C08_X1_CLK_GATE	31:0			
	RESERVED	31:5			
	RESERVED				
	PATH_3_GATING_CONTROL	3	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To PF ASYNC FIFO 0
	PATH_2_GATING_CONTROL	2	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To A0
	PATH_1_GATING_CONTROL	1	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FB
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FFE X1 clk

Table 11-22: CRU C10_FCLK_GATE Register

OFFSET	C10_FCLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x050	C10_FCLK_GATE	31:0			
	RESERVED	31:7			
	PATH_6_GATING_CONTROL	6	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To the SYNC Up on A0 and AHB Interface of Batching Memory
	PATH_5_GATING_CONTROL	5	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To AUDIO DMA
	PATH_4_GATING_CONTROL	4	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To M4 SRAM Instance, M4S12~M4S15
	PATH_3_GATING_CONTROL	3	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To M4 SRAM Instance, M4S8~M4S11
	PATH_2_GATING_CONTROL	2	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To M4 SRAM Instance, M4S4~M4S7
	PATH_1_GATING_CONTROL	1	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To M4 SRAM Instance, M4S0~M4S3
	PATH_0_GATING_CONTROL	0	RO	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To M4 Bus Matrix and Trace block

Table 11-23: CRU C11_CLK_GATE Register

OFFSET	C11_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x054	C11_CLK_GATE	31:0			Need to program MISC 0x310 bit first to be able to program this bit.
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER

Table 11-24: CRU CS_CLK_GATE Register

OFFSET	CS_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x05C	CS_CLK_GATE	31:0			
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To SWD Clk from PIN

Table 11-25: CRU C16_CLK_GATE Register

OFFSET	C16_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x064	C16_CLK_GATE	31:0			
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FB

Table 11-26: CRU C19_CLK_GATE Register

OFFSET	C19_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x06C	C19_CLK_GATE	31:0			
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To JTM

Table 11-27: CRU C21_CLK_GATE Register

OFFSET	C21_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x070	C21_CLK_GATE	31:0			
	RESERVED	31:1			
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FB (additional clk)

Table 11-28: CRU PF_SW_RESET Register

OFFSET	PF_SW_RESET				
	Bit Field	bit	R/W/C	Default	Description
0x080	PF_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:7			
	RESERVED	6	RW	0x0	Reserved, Used as General Purpose Register
	PF_PERPHERIAL_SW_RESET	5	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually.
	PF_ASYNC_FIFO_0_SW_RESET	4	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. (R08_P3 as well)
	PF_FIFO_8K_SW_RESET	3	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually.
	PF_FIFO_2_SW_RESET	2	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually.
	PF_FIFO_1_SW_RESET	1	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually.
	PF_FIFO_0_SW_RESET	0	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually.

Table 11-29: CRU FFE_SW_RESET Register

OFFSET	FFE_SW_RESET				
	Bit Field	bit	R/W/C	Default	Description
0x084	FFE_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:2			
	FFE_0_X1_SW_RESET	1	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. (R01_P3_FFE as well)
	FFE_0_X4_SW_RESET	0	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. (R01_P3_FFE as well)

Table 11-30: CRU FB_SW_RESET Register

OFFSET	FB_SW_RESET				
	Bit Field	bit	R/W/C	Default	Description
0x088	FB_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:6			
	FB_C21_DOMAIN_SW_RESET	5	RW	0x1	1'b1: Enable the Software Reset. FW needs to disable it manually.
	FB_C16_DOMAIN_SW_RESET	4	RW	0x1	1'b1: Enable the Software Reset. FW needs to disable it manually.
	RESERVED	3	RW	0x1	Reserved, Used as General Purpose Register
	FB_C09_DOMAIN_SW_RESET	2	RW	0x1	1'b1: Enable the Software Reset. FW needs to disable it manually.
	RESERVED	1	RW	0x1	Reserved, Used as General Purpose Register
	FB_C02_DOMAIN_SW_RESET	0	RW	0x1	1'b1: Enable the Software Reset. FW needs to disable it manually.

Table 11-31: CRU A1_SW_RESET Register

OFFSET	A1_SW_RESET				
	Bit Field	bit	R/W/C	Default	Description
0x08C	A1_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:3			
	CFGSM_SW_RESET	2	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. -> This is used to Reset the CfgSM/SPI Master and Related FIFO, DMA and AHB Master
	RESERVED	1	RW	0x0	Reserved, Used as General Purpose Register
	SPT_SW_RESET	0	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. -> This is used to Reset the SPT

Table 11-32: CRU AUDIO_MISC_SW_RESET Register

OFFSET	AUDIO_MISC_SW_RESET				
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
0x090	AUDIO_MISC_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:8			
	I2S_SW_RESET	7	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For I2S Power Domain. Note: It will only reset the AHB interface R01, but it will not reset R32 path. Suggest to power down, then power on I2S if Software Reset is needed
	DMA_SW_RESET	6	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For DMA Power Domain.
	AD5_SW_RESET	5	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD5 Power Domain.
	AD4_SW_RESET	4	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD4 Power Domain.
	AD3_SW_RESET	3	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD3 Power Domain.
	AD2_SW_RESET	2	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD2 Power Domain.
	AD1_SW_RESET	1	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD1 Power Domain.
	AD0_SW_RESET	0	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For AD0 Power Domain

Table 11-33: CRU FB_MISC_SW_RST_CTL Register

OFFSET	FB_MISC_SW_RST_CTL				
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
0x094	FB_MISC_SW_RST_CTL	31:0			
	RESERVED	31:2			
	PFAFIFO1_SW_RESET	1	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For R41
	AHBWB_SW_RESET	0	RW	0x0	1'b1: Enable the Software Reset. FW needs to disable it manually. ==> For R40

`

Table 11-34: CRU CLK_CONTROL_PMU Register

OFFSET	CLK_CONTROL_PMU				
	Bit Field	bit	R/W/C	Default	Description
0x100	CLK_CONTROL_PMU	31:0			This Clock is used to delay the Clock gating control signals from PMU. The PMU will monitor the feedback/delayed Clock Gating Control signals to ensure the clocks are OFF before jump to next state. The Firmware needs to Configure this Divider to ensure there delay time is longer enough. C23 Clock needs to be 2/3 of the lowest clock frequency of other clocks. For Example, if the Lowest clock frequency of other clocks is 5 MHz, then C23 should be lower than 3.33 MHz (Or the clock frequency of other clocks should be at least 1.5 times faster than C23.)
	RESERVED	31:10			
	DISABLE_CLOCK_DIVIDER	9	RW	0x1	1'b1 Clock Divider is ON 1'b0 Clock Divider is OFF, Output the Source Clock Directly
	CLOCK_DIVIDER_RATIO	8:0	RW	0x3E	High Speed Clock Divider Ratio, 0x0 → Divide by 2 0x1 → Divide by 3 ... 0x1FE → Divide by 512 Others → Reserved Default Divided by 64.

Table 11-35: CRU CRU_GENERAL Register

OFFSET	CRU_GENERAL				
	Bit Field	bit	R/W/C	Default	Description
0x104	CRU_GENERAL	31:0			
	RESERVED	31:8			
	GENERAL	7:1	RW	0x0	General Purpose Register
	SPICLK_ALWAYS_ON	0	RW	0x0	1’b0: Internal SPI Clock (C00) will be gated off if SPI CS is de-asserted even SPI Clock on the PAD is still running. 1’b1: Internal SPIC Clock (C00) is running if SPI Clock on the PAD is toggling regardless of SPI CS value.

Table 11-36: CRU CRU_DEBUG Register

OFFSET	CRU_DEBUG				
	Bit Field	bit	R/W/C	Default	Description
0x108	CRU_DEBUG	31:0			
	RESERVED	31:5			
	CRU_DEBUG_SELECT	4:0	RW	0x0	5'd0: None. 5'd1: C00 5'd2: C01 5'd3: C02 5'd4: C08X4 5'd5: C08X1 5'd6: C09 5'd7: C10 5'd8: C11 5'd9: CS 5'd10: C16 5'd11: C19 5'd12: C20/C32 5'd13: C21 5'd14: C23 5'd15: C30/C31
0x110	C01_CLK_DIV	31:0			Source Clock is C10 (CLK to Debug controller, eFuse, SDMA,I2S module Inside A1, AHB2APB Bridge /CFG DMA Bridge inside A1 , FFE, Packet FIFO,SDMA,A0) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C01_CLK_DIV_CG	4	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running This bit is used to turn off the clock for the SYNC down Divider
	C01_CLK_DIV	3:0	RW	0x1	4'h0: The Source Clock Is Divided by 1, No division 4'h1: Divided by 2, … 4'hf: Divided by 16. The input clock frequency will be Divided and generate the corresponding clock output.

Table 11-37: CRU C09_CLK_DIV Register

OFFSET	C09_CLK_DIV				
	Bit Field	bit	R/W/C	Default	Description
0x114	C09_CLK_DIV	31:0			Source Clock is C10 (CLK to Voice APB interface, PIF, FB) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C09_CLK_DIV_CG	4	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running This bit is used to turn off the clock for the SYNC down Divider
	C09_CLK_DIV	3:0	RW	0x1	4'h0: The Source Clock Is Divided by 1, No division 4'h1: Divided by 2, … 4'hf: Divided by 16. The input clock frequency will be Divided and generate the corresponding clock output.

Table 11-38: CRU C31_CLK_DIV Register

OFFSET	C31_CLK_DIV				
	Bit Field	bit	R/W/C	Default	Description
0x118	C31_CLK_DIV	31:0			Source Clock is C30 (LPSD CLK) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C31_CLK_DIV_CG	4	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running This bit is used to turn off the clock for the SYNC down Divider
	C31_CLK_DIV	3:0	RW	0x3	4'h0: The Source Clock Is Divided by 1, No division 4'h1: Divided by 2, … 4'hf: Divided by 16. The input clock frequency will be Divided and generate the corresponding clock output.

Table 11-39: CRU C09_CLK_GATE Register

OFFSET	C09_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x11C	C09_CLK_GATE	31:0			
	RESERVED	31:3			
	PATH_2_GATING_CONTROL	2	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To FB
	PATH_1_GATING_CONTROL	1	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To PIF
	PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To voice APB Interface

Table 11-40: CRU C30_C31_CLK_GATE Register

OFFSET	C30_C31_CLK_GATE				
	Bit Field	bit	R/W/C	Default	Description
0x120	C30_C31_CLK_GATE	31:0			
	RESERVED	31:4			
	C31_PATH_0_GATING_CONTROL	3	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To LPSD clk
	C30_PATH_2_GATING_CONTROL	2	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To I2S Master Clk
	C30_PATH_1_GATING_CONTROL	1	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To PDM RIGHT CLK
	C30_PATH_0_GATING_CONTROL	0	RW	0x0	1'b0: Clock is Stop. 1'b1: Clock is running To PDM LEFT CLK

Table 11-41: CRU CLK_DIVIDER_CLK_GATING Register

OFFSET	CLK_DIVIDER_CLK_GATING				
	Bit Field	bit	R/W/C	Default	Description
0x124	CLK_DIVIDER_CLK_GATING	31:0			This bit is used to turn off the clock for Clock Divider
	RESERVED	31:10			
	CLK_DIVIDER_J_CG	9	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C23 (PMU clk gating control)
	CLK_DIVIDER_I_CG	8	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C21 (FB)
	CLK_DIVIDER_H_CG	7	RW	0x1	1'b0: Clock is Stop. 1'b1: Clock is running To C19 (JTM)
	CLK_DIVIDER_G_CG	6	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C30, C31 (PDM LEFT/RIGHT Clk, I2S_MASTER clk, LPSD CLK)
	CLK_DIVIDER_F_CG	5	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C16 (FB)
	RESERVED	4	RW	0x1	Reserved, Used as General Purpose Register
	CLK_DIVIDER_D_CG	3	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C11 (M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)
	CLK_DIVIDER_C_CG	2	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C08 (FFE X4, X1)
	CLK_DIVIDER_B_CG	1	RW	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C02 (eFuse, FB, A1 (Including CFGSM))
	CLK_DIVIDER_A_CG	0	RO	0x1	1'b0: Clock is stopped. 1'b1: Clock is running To C10,C01,C09 (SYNC Up on A0, AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix , M4 Trace block, Debug controller, eFuse, SDMA,I2S module Inside A1, AHB2APB Bridge /CFG DMA Bridge inside A1 , FFE, Packet FIFO, SDMA, A0, Voice APB interface, PIF, FB)

Table 11-42: CRU CLK_SWITCH_FOR_B Register

OFFSET	NAME				
	Bit Field	bit	R/W/C	Default	Description
0x130	CLK_SWITCH_FOR_B	31:0			For Clock 2 (eFuse, FB, A1 (Including CFGSM))
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Table 11-43: CRU CLK_SWITCH_FOR_C Register

OFFSET	CLK_SWITCH_FOR_C				
	Bit Field	bit	R/W/C	Default	Description
0x134	CLK_SWITCH_FOR_C	31:0			For Clock 8 X4 (FFE X4 clk)
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Table 11-44: CRU CLK_SWITCH_FOR_D Register

OFFSET	CLK_SWITCH_FOR_D				
	Bit Field	bit	R/W/C	Default	Description
0x138	CLK_SWITCH_FOR_D	31:0			For Clock 11 (To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Table 11-45: CRU CLK_SWITCH_FOR_H Register

OFFSET	CLK_SWITCH_FOR_H				
	Bit Field	bit	R/W/C	Default	Description
0x13C	CLK_SWITCH_FOR_H	31:0			For Clock 19 (JTM)
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Table 11-46: CRU CLK_SWITCH_FOR_J Register

OFFSET	CLK_SWITCH_FOR_J				
	Bit Field	bit	R/W/C	Default	Description
0x140	CLK_SWITCH_FOR_J	31:0			For CLK 23 (PMU clk gating control)
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Table 11-47: CRU CLK_SWITCH_FOR_G Register

OFFSET	CLK_SWITCH_FOR_G				
	Bit Field	bit	R/W/C	Default	Description
0x144	CLK_SWITCH_FOR_G	31:0			To C30(PDM LEFT/RIGHT Clk, I2S_MASTER)
	RESERVED	31:1	RO	0x0	
	CLOCK_SOURCE_SELECTION	0	RW	0x0	1'b1: 32KHz Clock 1'b0: High Speed Clock

Core Special IO Functions & GPIO

Chapter 12. Core Special IO Functions & GPIO Overview

The following blocks are parts of the S3 Core Special IO functions:

- I2C: see Sensor sub-system section for details
- SPI Master for Sensor: see Sensor sub-subsystem section for details
- Configuration StateMachine & SPI Master for Flash device (CfgSM): see System section for details
- Communication Manager (CM)
- SPI Slave
- Analog Digital Conversion (ADC)
- System Analog IP (AIP)
- UART
- System Timers & Counters

Chapter 13. Communication Manager (CM) sub-system

The Communication Manager sub-system provides functions for communication with a host when S3 is in Companion mode. In this chapter it will be abbreviated CM.

13.1 CM Architecture

The CM sub-system is comprised of:

- SPI_Slave interface for host access
- Top Level Controller, the CM controller for data transfers
- the CM DMA controller, dedicated to use by this sub-system

A directly related S3 top-level component not within this sub-system is the Packet FIFO. Operation of the Packet FIFO is treated in a separate section but this chapter discusses in relation to operation of the CM.

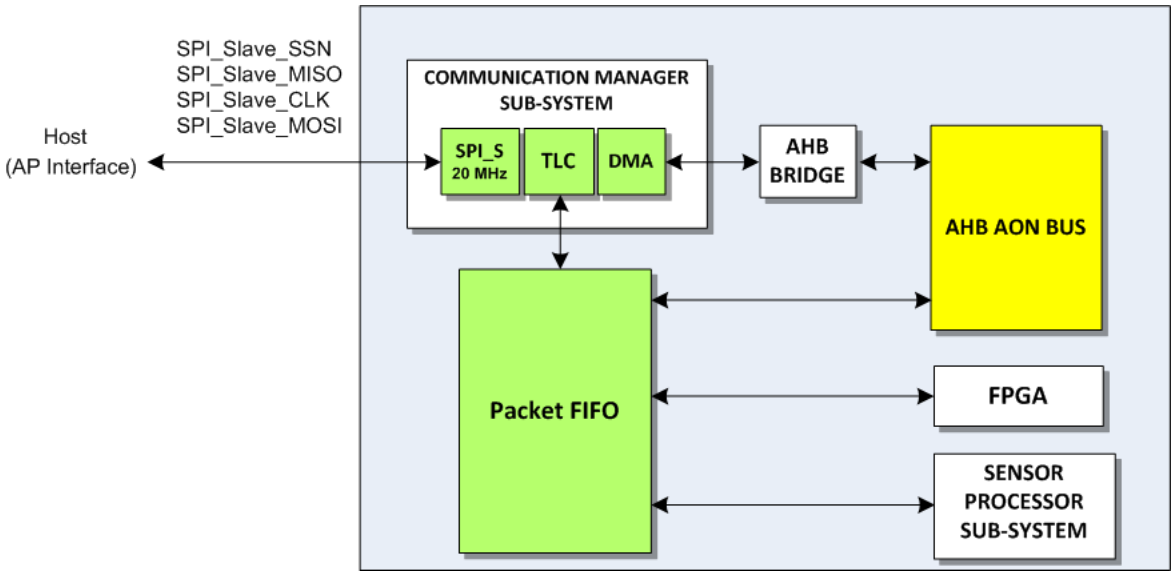

Figure 13-1: Communication Manager Architecture

Serial Peripheral Interface (SPI_Slave)

In power domain: AON A0

Provides interface between the TLC and an Applications Processor (AP).

TLC

In power domain: AON A0

The TLC provides the ability to interface to a Host, via the SPI_Slave interface.

DMA for CM

In power domain: AON A0

DMA dedicated for use by TLC.

13.1.1 Top Level Controller in the CM

Controller for data transfers

[topic to be added in future]

13.1.2 Communication Manager DMA controller

Dedicated to use by this sub-system

[topic to be added in future]

13.2 Communication Manager Theory of Operation

[topic to be added in future: Using the CM to communication with a Host AP]

Chapter 14. SPI_Slave interface in the CM

14.1 Introduction

For systems operating in AP Mode (use of a host), the SPI_Slave interface provides the means for communication between a host and the S3. It operates together with the Top Level Controller (TLC) module to achieve this function.

14.2 Architecture

Figure 14-1 shows the relation of the SPI_Slave interface to other system components.

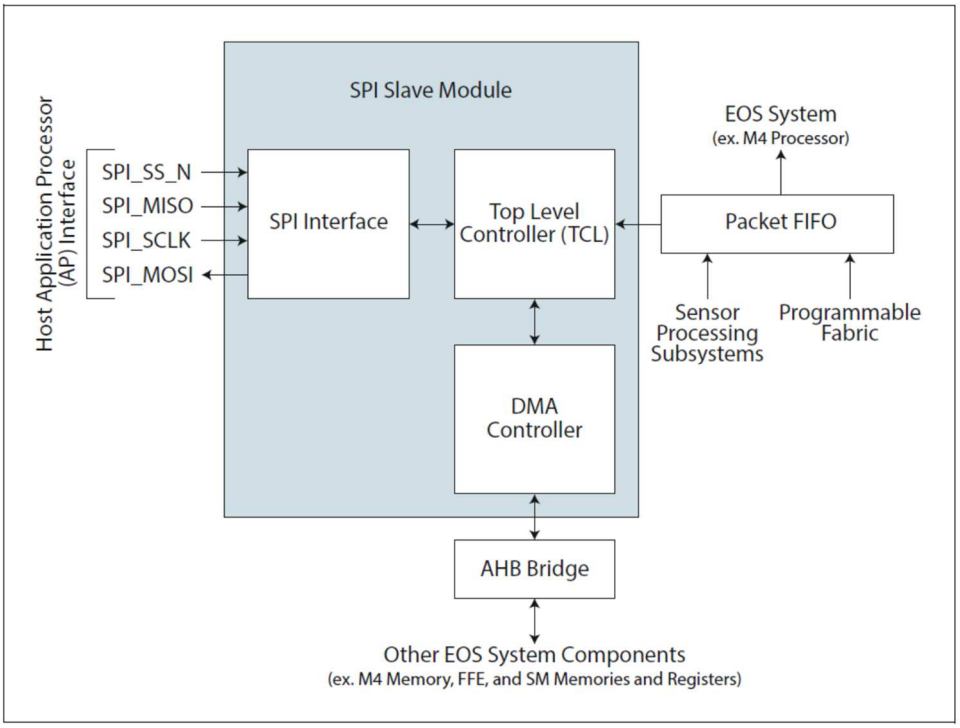

Figure 14-1: SPI_Slave within the Communication Manager

It is important to note that the SPI_Slave module does not communicate directly with other S3 functional blocks. Rather, it must perform transfers through one or more AHB bridges in order to access any register or memory within the EOS S3 system. For optimum performance of this interface during the "Smart Phone" mode, the EOS S3 system should avoid using common paths through the AHB infrastructure that are used by the SPI_Slave interface.

14.3 Usage Roles

In, AP Mode or Host / standalone Mode, the SPI_Slave can be used in two key roles:

- in setting up and debugging the EOS S3 system
- in retrieving run-time data

Use of SPI_Slave in environment with a host

Setup: When non-debug setup operations are performed in this type of system design, the host system uses the SPI_Slave module to transfer (write) a pre-compiled binary file to the S3 device's M4 memory. In addition, the host also writes pre-compiled binary files to other memories such as those used by the FFE and the Sensor Managers. Once that is complete, the host system enables the S3 M4 processor to execute the newly written binary code.

Data retrieval: During normal S3 system processing, the host uses the SPI_Slave interface to retrieve the results of the S3 system's sensor fusion processing. The host may then pass this data to the corresponding application.

Use of SPI_Slave in debugging operation

During S3 system debugging operations, the host uses the SPI Slave interface to load debug code, enable the M4 to execute this code, and retrieve the results. In addition, the host can elect to only enable certain subsystems to diagnose. For example, the host can configure the FFE and Sensor Manager with diagnostic code, enable these subsystems to do sensor fusion processing, and retrieve the result of this processing all without enabling the S3 system's M4 processor.

14.4 SPI Interface Protocol for the SPI_Slave block

The SPI_Slave Interface block supports SPI Mode 0 only:

- CPOL = 0, the base value (idle state) of the clock is 0.
- CPHA = 0, data is captured on the rising edge of the clock, and driven on the falling edge of the clock.

A transaction consists of SPI_SS_N being driven low (active) by the SPI Master, and then driven high after all of the desired bytes have been transferred. The SPI Interfaces assume that all transfers will consist of complete bytes. Any incomplete bytes at the end of a transaction are ignored by the hardware.

The S3 System SPI_Slave Interface protocol supports several different operations. The following sections will outline these operations.

14.5 Basic Read/Write Transfers

For basic Read/Write transfers to the TLC's registers, the S3 SPI_Slave Interface protocol requires that the Address Byte be transmitted first. The Address Byte includes a single Direction Bit representing the direction for the transfer (write vs. read). The Direction Bit is positioned in the most-significant bit of the Address Byte. The value of the Direction Bit is 1 for write transactions, and 0 for read transactions. The remaining 7 bits represent the register address within the TLC module. This address is unique to the TLC module and should not be confused with M4 addresses.

Examples:

- Read starting from TCL address 0x05 → Address Byte = 0x05.
- Write starting to TLC address 0x03 → Address Byte = 0x83.

During write transactions, the first byte received on the SPI_MISO pin corresponds to the Address Byte. The next bytes received correspond to Data Bytes. No valid data is driven on the MOSI pin. If the SPI clock is not free-running (i.e. it does not toggle while SPI_SS_N is inactive), the hardware requires that there be at least 2 extra rising clock edges on the SPI clock following the completion of a write transfer. This ensures that the write data reaches its final destination. The simplest way to generate these extra clock cycles is to perform a read from the TLC's "Scratch Register" (refer to section covering Transfers to TLC Local Registers).

During read transactions, the first byte received on the SPI_MISO pin corresponds to the Address Byte. The SPI_Slave interface ignores all subsequent data bits on the SPI_MISO pin. Instead, following the Address Byte, the SPI_Slave interface begins to drive data on the SPI_MOSI pin. This begins by outputting two dummy bytes followed by the first Data Byte. Thereafter, each byte transmitted on the SPI_MOSI pin will correspond to valid Data Bytes. Unlike write transfers, read transfers do not require extra SPI clock cycles after each transfer.

Figure 14-2 shows an example of the SPI Interface protocol. In each case, the MSB bit is shifted out first.

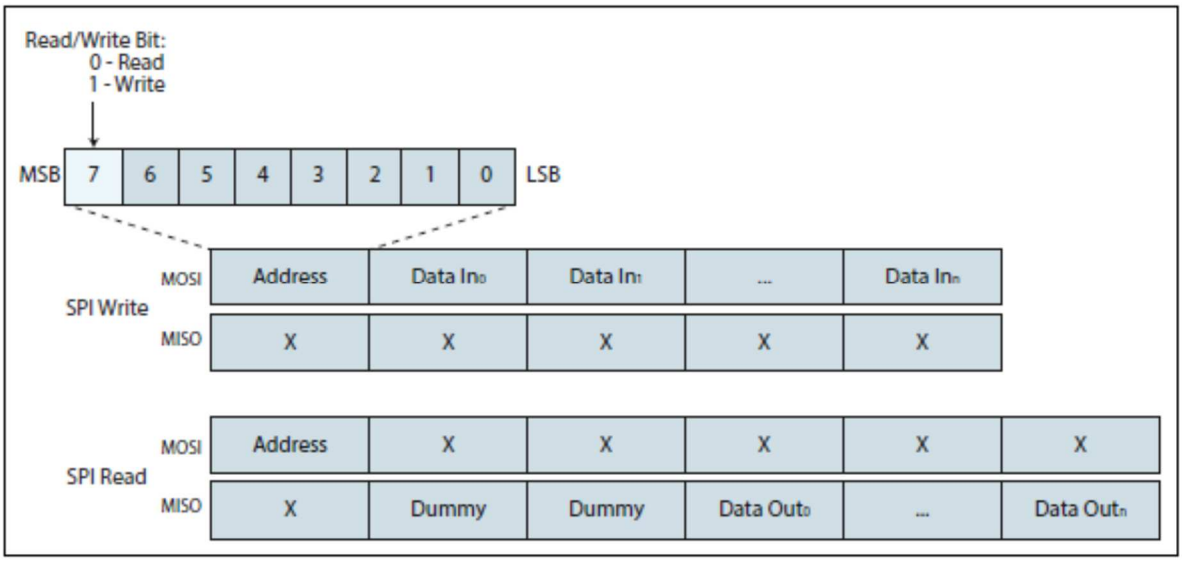

Figure 14-2: SPI_Slave Protocol Diagram

14.5.1 Device ID Read

The Device ID transfer cycle is a special protocol cycle for indentifying the EOS device to the host SPI controller. Unlike the write cycle described earlier, the address value of 0xFF will indicate to the SPI interface that an ID read cycle is underway. In response, the SPI Slave will return an ID value 0x20 on the MISO pin one-byte after the address phase. The following figure shows an example:

Figure 14-3: SPI Slave ID Read Protocol Diagram

14.5.2 Transfer Types

There are three basic transfer types. These are:

- Transfers to TLC local registers
  - These transfers access TLC local registers alone and do not produce any activity on the AHB interface.
- Transfers from Packet FIFOs.
  - These transfers also access the TLC local registers. However, the goal of these accesses is to read one or more bytes from the Packet FIFOs.
  - These read transactions can be conducted as a single or burst transfer.
- Transfers to resources in the M4 Memory address space.
  - These transfers also access the TLC local registers. However, the goal of these accesses is to conduct a transfer using the AHB interface.
  - These transfers provide for the following operations:
    - Single, 4-byte read transfers to the M4's Memory space.
    - Burst write transfers to the M4's Memory space.
    - Burst read transfers from the M4's Memory space.

14.5.2.1 Transfers to TLC Local Registers

These transfers will depend upon the type of TLC register being accessed. The default TLC response is to automatically increment the TLC register address (not the M4 Memory space address) after each byte access. An example would be a read from the M4 Memory Address registers. The SPI protocol's address phase will use the address for the "Memory Address Byte 0" register. Once this transfer has completed, the TLC will then automatically increment its register address to "Memory Address Byte 1". Similarly, once this transfer has completed, the TLC will then automatically increment its register address to "Memory Address Byte 2". This sequence will repeat until all bytes have been transferred or a TLC register address is reached that prevents this auto-incrementing operation. TLC register types that prevent auto-incrementing will be discussed in later sections.

It is also important to note that this auto-incrementing operation does not prevent any special features in these registers from being triggered. For example, writes to "Memory Address Byte 1" may cause a value to automatically be read from the M4 Memory space. A later section will discuss this transfer in more detail.

14.5.2.2 Transfers from Packet FIFOs

Packet FIFOs accessible from within TLC can only be read and not written. Therefore, writes to the Packet FIFOs are ignored. Conversely, since these are FIFOs, a burst read from these FIFOs requires that the same TLC register address be accessed multiple times. For this reason, the TLC does not increment its register address when accessing any Packet FIFO address.

14.5.2.3 Transfers to M4 Memory Address Space

The following sections will outline the transfer types and restrictions when accessing the M4's Memory Address space through the AHB Bridge interface.

14.5.3 Basic AHB Transfer Restrictions

The TLC restricts AHB Memory transfers to 4 bytes per transfer cycle. No other transfer size is currently supported. The following sections will expand on additional transfer specific restrictions.

14.5.3.1 AHB Memory Read

An AHB Memory read transfer is automatically triggered when writing to the TLC's "Memory Address Byte 1" except when both of the following is true:

- Bits [1:0] = 0x3 of the TLC's "Memory Address Byte 0"
- Bits [1:0] = 0x3 of the TLC's "Memory Access Control"

When both of these cases are true, a write to the TLC's "Memory Address Byte 1" will store the address value but not generate a read from the M4's Memory address space.

When a read is triggered, the data read from the M4's Memory Address space is stored in the TLC's "Memory Data Byte 0 – 3". A read from these registers do not automatically trigger an increment in the TLC's "Memory Address Byte 0 – 3" registers. These registers must be manually updated for each read operation.

In addition, the read operation must use AHB resources. This may require some time to complete. As such, the AHB Status register should be read to determine that the read transfer has completed prior to reading from the TLC's "Memory Data Byte 0 – 3" registers.

14.5.3.2 AHB Memory Burst Write

All AHB write operations through the TLC are treated the same way. For example, a single write transfer is treated as a burst of one 32-bit word. All burst write transfers to the M4's Memory Address space begins by disabling the automatic read trigger described in the last section. In doing so, this prevents an unnecessary read transfer through the AHB bus from interfering with the burst write operations.

The Host can now write the target address into the TLC's "Memory Address Byte 0 – 3" registers. This is followed by writing the data value into the TLC's "Memory Data Byte 0 – 3". Upon writing to the "Memory Data Byte 3" register, the "Memory Address Byte 0 – 1" registers are automatically incremented. In addition, the TLC's register address loops back to point to the first data byte register, "Memory Data Byte 0". By doing so, a block of data may be written from a single SPI data stream (i.e. one address phase followed by a series of data phases representing the burst data.). Figure 14-4 shows an example of a burst write sequence.

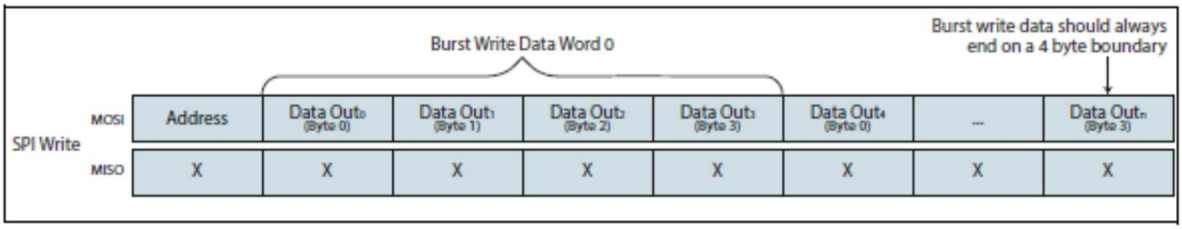

Figure 14-4: SPI Master Burst Write Sequence

AHB Memory write transfer block sizes are currently limited to 64K bytes (i.e. 16K, 32-bit words). In addition, as mentioned in an earlier section, access through the AHB Bridge requires each transfer to consist of 4 bytes. As such, writes to the TLC's "Memory Data Byte 3" register will automatically trigger an increment by 4 of the TLC's "Memory Address Byte 0 – 1" registers, as well as, trigger the write of data through the AHB interface. The TLC's "Memory Address Bytes 2 – 3" registers are not affected by writes to "Memory Data Byte 3". Therefore, burst transfers that exceed the 64K byte boundary will automatically wrap back to the beginning.

14.5.3.3 AHB Memory Burst Read

To complement the Burst Write transfer of the last section, the EOS device provides a Burst Read transfer operation. Unlike the Burst Write, the EOS device implements its Burst Read transfer as a DMA operation.

Prior to initiating Burst Read transfers, software must first check the following:

- There is no Burst Read transfer underway.
  - All of the data from the previous Burst Read transfer must be read out through the SPI Interface prior to initiating another Burst Read transfer.
- There is data in the Burst Read transfer FIFO.
  - This can be checked by reading the "Burst FIFO Status" register.

Once both of these conditions are true, the Host can configure the Burst Read transfer by writing the target address into the TLC's "Burst Read AHB Byte Address Byte 0 – 3" registers. This is followed by writing the data value into the TLC's "Burst Size Byte 0" register. The Burst Read begins with writing to the "Burst Size Byte 1" register. There needs to be two "dummy" byte cycles following the write to the "Burst Size Byte 1" register. The simplest way to accomplish this is to read the "Burst FIFO Status" register. This status should be used to determine that there is at least one byte available to be read from the "Burst Read Data" register. Once at least one Burst Read data byte is available, the burst read operation begins by reading from the "Burst Read Data" register. The TLC supports the burst transfer operation by not increment its register address pointer when reading from the "Burst Read Data" register. Figure 14-5 shows an example of the expected sequence.

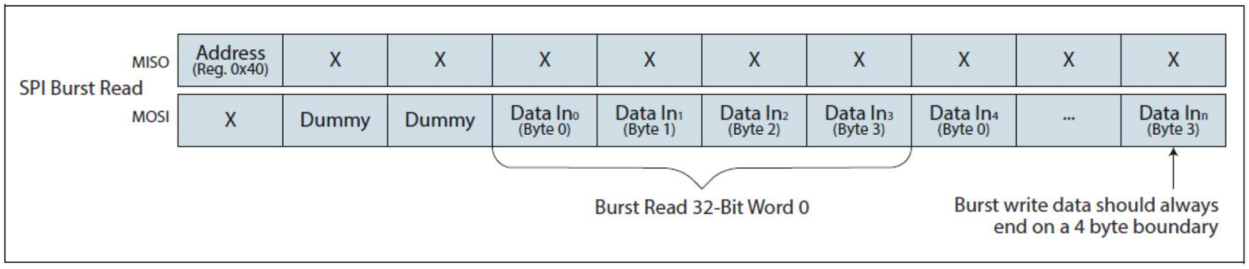

Figure 14-5: Example Burst Read Sequence

14.6 Communication Manager Components Registers

Offset	Name/Field	Bits	Type	Default	Description
0x09	CM_FIFO_0_Data	7:0	RO		Packet FIFO Bank FIFO 0 Read Port
0x0A	CM_FIFO_1_Data	7:0	RO		Packet FIFO Bank FIFO 1 Read Port
0x0B	CM_FIFO_2_Data	7:0	RO		Packet FIFO Bank FIFO 2 Read Port
0x0C	CM_FIFO_8k_Data	7:0	RO		Packet FIFO Bank FIFO 8K Read Port
0x20	MemAddrByte0	7:0	RW	0x0	Memory Address, representing AHB Byte Address bit [7:0]. Bit 7:0 R/W, Default All 0 Bit [1:0] will be ignored since only Support Word Access
0x21	MemAddrByte1	7:0	RW	0x0	AHB Memory Address, representing AHB Byte Address bit [15:8]. Bit 7:0 R/W, Default All 0 Once write to this SFR, an AHB memory Read Could be Trigger. See "TLC AHB Memory Read Trigger" worksheet for detail.
0x22	MemAddrByte2	7:0	RW	0x0	AHB Memory Address, representing AHB Byte Address bit [23:16]. Bit 7:0 R/W, Default All 0
0x23	MemAddrByte3	7:0	RW	0x0	AHB Memory Address, It is representing AHB Byte Address bit [31:24]. Bit 7:0 R/W, Default All 0
0x28	MemDataByte0	7:0	RW	0x0	First Byte (LSB) of AHB memory data
0x29	MemDataByte1	7:0	RW	0x0	Second byte of AHB memory data
0x2A	MemDataByte2	7:0	RW	0x0	Third byte of AHB memory data
0x2B	MemDataByte3	7:0	RW	0x0	Forth byte of AHB memory data, once write to this SFR 1. Trigger a AHB memory Write 2. Auto Increment the AHB memory address (MemAddrByte0/MemAddrByte1) by 4 since AHB memory Address is in Byte Granularity. (offset 0x20~0x21, 64KB range)
0x2F	AhbStatus	7:0			
	NoPendingAhbReq	0	RO	0x0	1'b0 No Pending AHB Memory Read/Write request (Read Only)
	AhbReadDataValid	1	RO	0x0	1'b1 AHB Read Data is Ready (Read Only) This bit is Auto Clear Once a new AHB read Access kick off and Auto Set once AHB read data is valid.
	AhbReqFIFOFull	2	RO	0x0	1'b1 AHB Address/Data/Command FIFO is Full. (Read Only)
	AhbReqFIFOhalfEmpty	3	RO	0x1	1'b1 AHB Address/Data/Command FIFO is less than half full.
	Reserved	5:4	RO	0x0	Reserved, Read Only

Offset	Name/Field	Bits	Type	Default	Description
	AhbReqFIFOSize	7:6	RO		2'b00 : FIFO is 4 Entries 2'b01 : FIFO is 8 Entries Other's : Reserved
0x30	AhbAccessCtl	7:0			
	AhbReadReqMode	1:0	RW	0x0	AHB Memory Read Option 2'b11 : If MemAddrByte[1:0] is 2'b11, then AHB Memory Read will not be automatically triggered once MemAddrByte1 is written Other : Once MemAddrByte1 is written, AHB Memory read is triggered.
	scratch0	7:2	RW	0x0	General Purpose Registers, R/W, Default 0
0x31	ScratchByte	7:0	RW	0x0	General Purpose Registers, R/W, Default 0
0x32	S3GeneralStatus	7:0			
	M4PowerStatus	1:0	R	0x0	Identifies the M4 Power Status 2'b00: Active 2'b01: Deep Sleep 2'b10: Shut Down 2'b11: Clock off
	M4ResetStatus	2	R		1'b1 : M4 Subsystem reset is NOT released 1'b0: M4 Subsystem reset is released (Not the M4 Core Reset)
	M4RebootReq	3	R	0x0	1'b1 Need to re-load the code (if system is used in AP mode) 1'b0 do not reload the code on reboot This bit reflects the value on RFUPMU offset 3EC It will not set for cold start.
	PORiniCond	7:4	R	0xE	0xE value if POR trigger, It reflects the value of RFUMPU offset 00C
0x36	DmaDebugCtl0	7:0	RW	0x0	
	Reserved	7:1	RO	0x0	Reserved, Read Only
	DmaFifoClear	0	RW	0x0	Write 1'b1 to clear the DMA FIFO. Firmware can only set this bit to 1 after set DmaClear bit to 1 and Program this bit to 0 after clear DmaClear bit
0x37	DmaDebugCtl1	7:0	RW	0x0	
	Reserved	7:1	RO	0x0	Reserved, Read Only
	DmaClear	0	RW	0x0	Write 1'b1 to Reset the DMA engineer. Firmware needs to clear it before kick off the new DMA Transfer. Need to do a dummy Read on this SFR after program this bit
0x38	DmAAddr0	7:0	RW	0x0	DMA Starting Address; represents AHB Byte Address bit [7:0]. Bit [1:0] will be ignored since the hardware only supports Word Access
0x39	DmAAddr1	7:0	RW	0x0	DMA Starting Address; represents AHB Byte Address bit [15:8].
0x3A	DmAAddr2	7:0	RW	0x0	DMA Starting Address; represents AHB Byte Address bit [23:16]

Offset	Name/Field	Bits	Type	Default	Description
0x3B	DmAAddr3	7:0	RW	0x0	DMA Starting Address; represents AHB Byte Address bit [31:24]
0x3C	DmaBurstSize0	7:0	RW	0x0	DMA transfer size indicates the number of bytes to be read out (X). The minimum transfer size is 4 bytes. Program the value for number of bytes to read minus 4 bytes (X -4), into these two registers: DmaBurstSize0 register - represents the BurstSize[7:0]. DmaBurstSize1 register - represents the BurstSize[15:8]. Lower 2 bits are ignored by hardware, since it only supports Word Access. This means only multiples of 4 are supported. For example: To read 4 bytes, you would write: DmaBurstSize0 = 0, DmaBurstSize1 = 0 To read 8 bytes, you would write: DmaBurstSize0 = 4, DmaBurstSize1 = 0 … To read 256 bytes, you would write: DmaBurstSize0 = FC, DmaBurstSize1 = 0 To read 260 bytes, you would write: DmaBurstSize0 = 0, DmaBurstSize1 = 1 ... and so on, etc.
0x3D	DmaBurstSize1	7:0	RW	0x0	MSB Byte of DMA transfer size. it is representing BurstSize[15:8]. Max transfer size is 64KB, Once it is written, DMA will be kickoff unless DmaBurstSize0[1:0] = 2'b10. Min. Transfer Size will be 4 Bytes once DMA is Kick Off ({DmaBurstSize0, DmaBurstSize1} == 0x0}
0x3E	Reserved	7:0	WO	0x0	For Dummy Write purpose
0x3F	DmaStatus	7:0	RW	0x1	Write will be ignore.
	Reserved	7:2	WO	0x0	Write will be ignore.
	DmaFIFO_underflow	1	RHW	0x0	1'b1: DMA FIFO hit underflow condition. This bit will be clear once there is a new DMA request
	DmaFIFO_Empty	0	RO	0x1	1'b1 : DMA FIFO is Empty 1'b0 : DMA is NOT Empty. Write to this bit has no effect
0x40	DmaRdData	7:0	RO		DMA Read Data Port
0x7F	DeviceIDByte	7:0	RO	0x21	Device ID, Read Only. 0x21, the Protocol of accessing this SFR is different, See Device ID read Page for detail

Chapter 15. UART

In power domain: M4

The UART provides serial communications support for M4-F debug and code development and communication with serial (UART-based) external peripherals.

The UART can generate transmit FIFO, receive FIFO, receive timeout, modem status, and error condition interrupts.

Its interrupts can be used as wake-up events for the M4-F.

15.1 UART registers

Offset	Name/Field	Bits	Type	Default	Description
0x000	DR	11:0			UART Data Register
	OE	11	RW	X	Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.
	BE	10	RW	X	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.
	PE	9	RW	X	Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UARTLCR_H on page 3-12 select. In FIFO mode, this error is associated with the character at the top of the FIFO.
	FE	8	RW	X	Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.
	DATA	7:0	RW	X	Receive (read) data character. Transmit (write) data character.
0x004	RSR_ECR	3:0	RW		UART is the receive status register/error clear register.
	OE	3	WC	X	Overrun error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by a write to UARTECR. The FIFO contents remain valid because no more data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data, to empty the FIFO.

Offset	Name/Field	Bits	Type	Default	Description
	BE	2	WC	X	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity, and stop bits). This bit is cleared to 0 after a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.
	PE	1	WC	X	Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UARTLCR_H on page 3-12 select. This bit is cleared to 0 by a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO.
	FE	0	WC	X	Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO.
0x018	TFR	8:0			UART Flag Register
	RI	8	RO	X	Ring indicator. This bit is the complement of the UART ring indicator, nUARTRI, modem status input. That is, the bit is 1 when nUARTRI is LOW.
	TXFE	7	RO	1	Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the Line Control Register, UARTLCR_H on page 3-12. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.
	RXFF	6	RO	0	Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H Register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.
	TXFF	5	RO	0	Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H Register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.

Offset	Name/Field	Bits	Type	Default	Description
	RXFE	4	RO	1	Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H Register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.
	BUSY	3	RO	0	UART busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.
	DCD	2	RO	X	Data carrier detect: This bit is the complement of the UART data carrier detect, nUARTDCD, modem status input. That is, the bit is 1 when nUARTDCD is LOW.
	DSR	1	RO	X	Data set ready. This bit is the complement of the UART data set ready, nUARTDSR, modem status input. That is, the bit is 1 when nUARTDSR is LOW.
	CTS	0	RO	X	Clear to send. This bit is the complement of the UART clear to send, nUARTCTS, modem status input. That is, the bit is 1 when nUARTCTS is LOW.
0x020	ILPR_ILPDVSR	7:0	RW	0x0	8-bit low-power divisor value. These bits are cleared to 0 at reset.
0x024	IBRD	15:0	RW	0x0	The integer baud rate divisor. These bits are cleared to 0 on reset.
0x028	FBRD	5:0	RW	0x0	The fractional baud rate divisor. These bits are cleared to 0 on reset.
0x02C	LCR_H	7:0			UART Line Control Register. This register accesses bit 29 to 22 of the UART Line Control Register, UARTLCR.
	SPS	7	RW	0x0	Stick parity select. 0 = stick parity is disabled 1 = either: • if the EPS bit is 0 then the parity bit is transmitted and checked as a 1 • if the EPS bit is 1 then the parity bit is transmitted and checked as a 0. This bit has no effect when the PEN bit disables parity checking and generation. See Table 3-11 on page 3-14 for the parity truth table.
	WLEN	6:5	RW	0x0	Word length. These bits indicate the number of data bits transmitted or received in a frame as follows: b11 = 8 bits b10 = 7 bits b01 = 6 bits b00 = 5 bits.

Offset	Name/Field	Bits	Type	Default	Description
	FEN	4	RW	0x0	Enable FIFOs: 0 = FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers 1 = transmit and receive FIFO buffers are enabled (FIFO mode).
	STP2	3	RW	0x0	Two stop bits select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
	EPS	2	RW	0x0	Even parity select. Controls the type of parity the UART uses during transmission and reception: 0 = odd parity. The UART generates or checks for an odd number of 1s in the data and parity bits. 1 = even parity. The UART generates or checks for an even number of 1s in the data and parity bits. This bit has no effect when the PEN bit disables parity checking and generation. See Table 3-11 on page 3-14 for the parity truth table.
	PEN	1	RW	0x0	Parity enable: 0 = parity is disabled and no parity bit added to the data frame 1 = parity checking and generation is enabled. See Table 3-11 on page 3-14 for the parity truth table.
	BRK	0	RW	0x0	Send break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.
0x030	CR	15:0			UART Control Register
	CTSEn	15	RW	0x0	CTS hardware flow control enable: If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted.
	RTSEn	14	RW	0x0	RTS hardware flow control enable: If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received.
	Out2	13	RW	0x0	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. That is, when the bit is programmed to a 1, the output is 0. For DTE this can be used as Ring Indicator (RI).
	Out1	12	RW	0x0	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. That is, when the bit is programmed to a 1 the output is 0. For DTE this can be used as Data Carrier Detect (DCD).
	RTS	11	RW	0x0	Request to send. This bit is the complement of the UART request to send, nUARTRTS, modem status output. That is, when the bit is programmed to a 1 then nUARTRTS is LOW.
	DTR	10	RW	0x0	Data transmit ready. This bit is the complement of the UART data transmit ready, nUARTDTR, modem status output. That is, when the bit is programmed to a 1 then nUARTDTR is LOW.

Offset	Name/Field	Bits	Type	Default	Description
	RXE	9	RW	0x1	Receive enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for either UART signals or SIR signals depending on the setting of the SIREN bit. When the UART is disabled in the middle of reception, it completes the current character before stopping.
	TXE	8	RW	0x1	Transmit enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for either UART signals, or SIR signals depending on the setting of the SIREN bit. When the UART is disabled in the middle of transmission, it completes the current character before stopping.
	LBE	7	RW	0x0	Loopback enable. If this bit is set to 1 and the SIREN bit is set to 1 and the SIRTEST bit in the Test Control Register, UARTTCR on page 4-5 is set to 1, then the nSIROUT path is inverted, and fed through to the SIRIN path. The SIRTEST bit in the test register must be set to 1 to override the normal half-duplex SIR operation. This must be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to 0 when loopback testing is finished. This feature reduces the amount of external coupling required during system test. If this bit is set to 1, and the SIRTEST bit is set to 0, the UARTTXD path is fed through to the UARTRXD path. In either SIR mode or UART mode, when this bit is set, the modem outputs are also fed through to the modem inputs. This bit is cleared to 0 on reset, to disable loopback.
	SIRLP	2	RW	0x0	SIR low-power IrDA mode. This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active high pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width that is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances.
	SIREN	1	RW	0x0	SIR enable: 0 = IrDA SIR ENDEC is disabled. nSIROUT remains LOW (no light pulse generated), and signal transitions on SIRIN have no effect. 1 = IrDA SIR ENDEC is enabled. Data is transmitted and received on nSIROUT and SIRIN. UARTTXD remains HIGH, in the marking state. Signal transitions on UARTRXD or modem status inputs have no effect. This bit has no effect if the UARTEN bit disables the UART.
	UARTEN	0	RW	0x0	UART enable: 0 = UART is disabled. If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping. 1 = the UART is enabled. Data transmission and reception occurs for either UART signals or SIR signals depending on the setting of the SIREN bit.
0x034	IFLS	6:0			Interrupt FIFO Level Select Register.

Offset	Name/Field	Bits	Type	Default	Description
	RXIFLSEL	5:3	RW	0x2	Receive interrupt FIFO level select. The trigger points for the receive interrupt are as follows: b000 = Receive FIFO becomes ≥ 1/8 full b001 = Receive FIFO becomes ≥ 1/4 full b010 = Receive FIFO becomes ≥ 1/2 full b011 = Receive FIFO becomes ≥ 3/4 full b100 = Receive FIFO becomes ≥ 7/8 full b101-b111 = reserved.
	TXIFLSEL	2:0	RW	0x2	Transmit interrupt FIFO level select. The trigger points for the transmit interrupt are as follows: b000 = Transmit FIFO becomes ≤ 1/8 full b001 = Transmit FIFO becomes ≤ 1/4 full b010 = Transmit FIFO becomes ≤ 1/2 full b011 = Transmit FIFO becomes ≤ 3/4 full b100 = Transmit FIFO becomes ≤ 7/8 full b101-b111 = reserved.
0x038	IMSC	10:0			Interrupt Mask Set/Clear Register
	OEIM	10	RW	0x0	Overrun error interrupt mask. A read returns the current mask for the UARTOEINTR interrupt. On a write of 1, the mask of the UARTOEINTR interrupt is set. A write of 0 clears the mask.
	BEIM	9	RW	0x0	Break error interrupt mask. A read returns the current mask for the UARTBEINTR interrupt. On a write of 1, the mask of the UARTBEINTR interrupt is set. A write of 0 clears the mask.
	PEIM	8	RW	0x0	Parity error interrupt mask. A read returns the current mask for the UARTPEINTR interrupt. On a write of 1, the mask of the UARTPEINTR interrupt is set. A write of 0 clears the mask.
	FEIM	7	RW	0x0	Framing error interrupt mask. A read returns the current mask for the UARTFEINTR interrupt. On a write of 1, the mask of the UARTFEINTR interrupt is set. A write of 0 clears the mask.
	RTIM	6	RW	0x0	Receive timeout interrupt mask. A read returns the current mask for the UARTRTINTR interrupt. On a write of 1, the mask of the UARTRTINTR interrupt is set. A write of 0 clears the mask.
	TXIM	5	RW	0x0	Transmit interrupt mask. A read returns the current mask for the UARTTXINTR interrupt. On a write of 1, the mask of the UARTTXINTR interrupt is set. A write of 0 clears the mask.
	RXIM	4	RW	0x0	Receive interrupt mask. A read returns the current mask for the UARTRXINTR interrupt. On a write of 1, the mask of the UARTRXINTR interrupt is set. A write of 0 clears the mask.
	DSRMIM	3	RW	0x0	nUARTDSR modem interrupt mask. A read returns the current mask for the UARTDSRINTR interrupt. On a write of 1, the mask of the UARTDSRINTR interrupt is set. A write of 0 clears the mask.

Offset	Name/Field	Bits	Type	Default	Description
	DCDMIM	2	RW	0x0	nUARTDCD modem interrupt mask. A read returns the current mask for the UARTDCDINTR interrupt. On a write of 1, the mask of the UARTDCDINTR interrupt is set. A write of 0 clears the mask.
	CTSMIM	1	RW	0x0	nUARTCTS modem interrupt mask. A read returns the current mask for the UARTCTSINTR interrupt. On a write of 1, the mask of the UARTCTSINTR interrupt is set. A write of 0 clears the mask.
	RIMIM	0	RW	0x0	nUARTRI modem interrupt mask. A read returns the current mask for the UARTRIINTR interrupt. On a write of 1, the mask of the UARTRIINTR interrupt is set. A write of 0 clears the mask.
0x03C	RIS	11:0			Raw Interrupt Status Register.
	OERIS	10	RO	X	Overrun error interrupt status: Returns the raw interrupt state of the UARTOEINTR interrupt.
	BERIS	9	RO	X	Break error interrupt status: Returns the raw interrupt state of the UARTBEINTR interrupt.
	PERIS	8	RO	X	Parity error interrupt status: Returns the raw interrupt state of the UARTPEINTR interrupt.
	FERIS	7	RO	X	Framing error interrupt status: Returns the raw interrupt state of the UARTFEINTR interrupt.
	RTRIS	6	RO	X	Receive timeout interrupt status: Returns the raw interrupt state of the UARTRTINTR interrupt
	TXRIS	5	RO	X	Transmit interrupt status: Returns the raw interrupt state of the UARTTXINTR interrupt.
	RXRIS	4	RO	X	Receive interrupt status: Returns the raw interrupt state of the UARTRXINTR interrupt.
	DSRRMIS	3	RO	X	nUARTDSR modem interrupt status: Returns the raw interrupt state of the UARTDSRINTR interrupt.
	DCDRMIS	2	RO	X	nUARTDCD modem interrupt status: Returns the raw interrupt state of the UARTDCDINTR interrupt.
	CTSRMIS	1	RO	X	nUARTCTS modem interrupt status: Returns the raw interrupt state of the UARTCTSINTR interrupt.
	RIRMIS	0	RO	X	nUARTRI modem interrupt status: Returns the raw interrupt state of the UARTRIINTR interrupt.
0x040	MIS	10:0			Masked Interrupt Status Register
	OEMIS	10	RO	X	Overrun error masked interrupt status: Returns the masked interrupt state of the UARTOEINTR interrupt.
	BEMIS	9	RO	X	Break error masked interrupt status: Returns the masked interrupt state of the UARTBEINTR interrupt.
	PEMIS	8	RO	X	Parity error masked interrupt status: Returns the masked interrupt state of the UARTPEINTR interrupt.

Offset	Name/Field	Bits	Type	Default	Description
	FEMIS	7	RO	X	Framing error masked interrupt status: Returns the masked interrupt state of the UARTFEINTR interrupt.
	RTMIS	6	RO	X	Receive timeout masked interrupt status: Returns the masked interrupt state of the UARTRTINTR interrupt.
	TXMIS	5	RO	X	Transmit masked interrupt status: Returns the masked interrupt state of the UARTTXINTR interrupt.
	RXMIS	4	RO	X	Receive masked interrupt status: Returns the masked interrupt state of the UARTRXINTR interrupt.
	DSRMMIS	3	RO	X	nUARTDSR modem masked interrupt status: Returns the masked interrupt state of the UARTDSRINTR interrupt.
	DCDMMIS	2	RO	X	nUARTDCD modem masked interrupt status: Returns the masked interrupt state of the UARTDCDINTR interrupt.
	CTSMMIS	1	RO	X	nUARTCTS modem masked interrupt status: Returns the masked interrupt state of the UARTCTSINTR interrupt.
	RIMMIS	0	RO	X	nUARTRI modem masked interrupt status: Returns the masked interrupt state of the UARTRIINTR interrupt.
0x044	ICR	10:0			Interrupt Clear Register.
	OEIC	10	WO	X	Overrun error interrupt clear: Clears the UARTOEINTR interrupt.
	BEIC	9	WO	X	Break error interrupt clear: Clears the UARTBEINTR interrupt.
	PEIC	8	WO	X	Parity error interrupt clear: Clears the UARTPEINTR interrupt.
	FEIC	7	WO	X	Framing error interrupt clear: Clears the UARTFEINTR interrupt.
	RTIC	6	WO	X	Receive timeout interrupt clear: Clears the UARTRTINTR interrupt.
	TXIC	5	WO	X	Transmit interrupt clear: Clears the UARTTXINTR interrupt
	RXIC	4	WO	X	Receive interrupt clear: Clears the UARTRXINTR interrupt
	DSRMIC	3	WO	X	nUARTDSR modem interrupt clear: Clears the UARTDSRINTR interrupt
	DCDMIC	2	WO	X	nUARTDCD modem interrupt clear: Clears the UARTDCDINTR interrupt
	CTSMIC	1	WO	X	nUARTCTS modem interrupt clear: Clears the UARTCTSINTR interrupt
	RIMIC	0	WO	X	nUARTRI modem interrupt clear: Clears the UARTRIINTR interrupt
0x080	TCR	2:0			Test Control Register
	SIRTEST	2	WO	X	SIR Test Enable
	TESTFIFO	1	WO	X	Test FIFO enable

Offset	Name/Field	Bits	Type	Default	Description
	ITEN	0	WO	X	Integration test enable
0x084	ITIP	7:0			Integration test input register
	UARTTXDMACLR	7	RW	X	
	UARTRXDMACLR	6	RW	X	
	nUARTRI	5	RO	X	
	nUARTDCD	4	RO	X	
	nUARTCTS	3	RO	X	
	nUARTDSR	2	RO	X	SIR Test Enable
	SIRIN	1	RO	X	Test FIFO enable
	UARTRXD	0	RO	X	Integration test enable
0x088	ITOP	15:0	RW	X	Integration test output register
0x08c	TDR	10:0	RW	X	Test data register
0xFE0	PeriphID0	7:0	RO	0x11	UART PeriphID0 register
0xFE4	PeriphID1	7:0	RO	0x10	UART PeriphID1 register
0xFE8	PeriphID2	7:0	RO	0x34	UART PeriphID2 register
0xFEC	PeriphID3	7:0	RO	0x0	UART PeriphID3 register
0xFF0	PCellID0	7:0	RO	0xD	UART PCellID0 register
0xFF4	PCellID1	7:0	RO	0xF0	UART PCellID1 register
0xFF8	PCellID2	7:0	RO	0x5	UART PCellID2 register
0xFFC	PCellID3	7:0	RO	0xB1	UART PCellID3 register

Chapter 16. Clocking and Timing Elements

16.1 RTC (Real Time Clock)

In power domain:

The real-time clock is one of three clock sources that can be used as master for the clock chain.

16.1.1 RTC Registers

The RTC and SPT registers are in the same block. Please see Section 0

16.2 SPT (Simple Periodic Timer)

In power domain: AON

The Simple Periodic Timer (SPT) provides 24-bit counter timer with 1mS resolution.

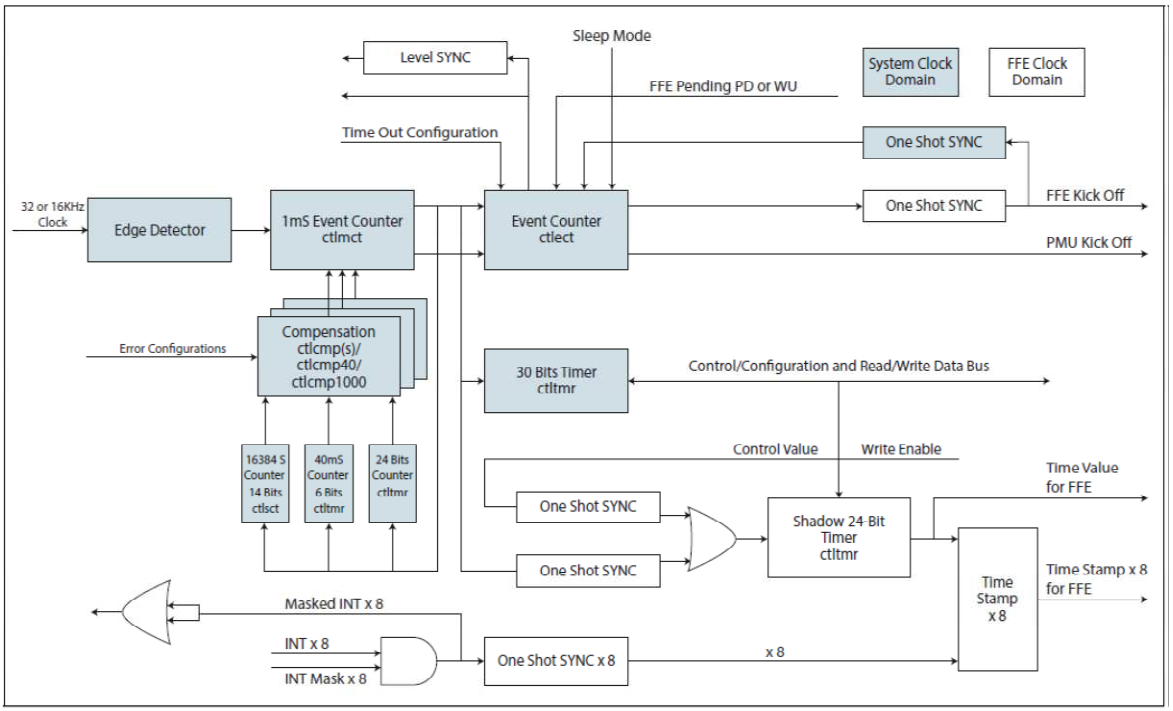

Figure 16-1: SPT Block Diagram

The EOS S3 system supports several counters that count the clock event to generate the 1ms event as well as time out events for waking up FFE and FFE power domain. It also provides the 24-bit timers and 8 time stamps for FFE and 30-bit timer for software use. The clock event is generated on both edges of reference clock. The period of 1mS time out event can be adjusted by configuring the trim bits.

- Clock Event Generator: Generate the Clock Event base on the edge of reference clock. Reference Clock could be either 16 KHz or 32 KHz with certain PPM error.
- 1ms event Counter: The resolution is 1 Clock Event.
- For 32 KHz Reference Clock
    - 1ms event is alternating between 65 and 66 counts of clock event in the following sequence
        - 65 → 66 → 65 → 66 → 65 → 66 → 65…….

  For 16 KHz Reference Clock
    - 1ms event is alternating between 32 and 33 counts of clock event in the following sequence
        - 32 → 33 → 33 → 33 → 32 → 33 → 33 → 33 → 32 → 33→ 33….
- 30/24 Bits Timer: The resolution is 1mS.
- Time Stamp: Total 8, Timers LSB (Lower 16 bits of timers) is latched once corresponding Interrupt triggers.
- 5mS time out event with less than 1% error in two hours period.

16.2.1 Error Correction for 1mS Timer

The 1ms event counter implementation will accumulate around -550us error for 1 Second period without error compensation. So Error Correction circuit is implemented to compensate the error of design and reference clock. See the *QuickLogic EOS S3 Sensor Processing Platform User Guide* for more details.

Multiple layers of compensation scheme is used:

- Compensation every 40ms – Increase 1 or not
- Compensation every 1 Second – Increase or Decrease 1
- Compensation every 2 Second – Increase or Decrease 1
- Compensation every 4 Seconds – Increase or Decrease 1
- Compensation every 8 Seconds – Increase or Decrease 1
- Compensation every 16 Seconds – Increase or Decrease 1
- Compensation every 32 Seconds – Increase or Decrease 1
- Compensation every 64 Seconds – Increase or Decrease 1
- Compensation every 128 Seconds – Increase or Decrease 1
- Compensation every 256 Seconds – Increase or Decrease 1
- Compensation every 512 Seconds – Increase or Decrease 1
- Compensation every 1024 Seconds – Increase or Decrease 1
- Compensation every 2048 Seconds – Increase or Decrease 1
- Compensation every 4096 Seconds – Increase or Decrease 1
- Compensation every 8192 Seconds – Increase or Decrease 1
- Compensation every 16384 Seconds – Increase or Decrease 1

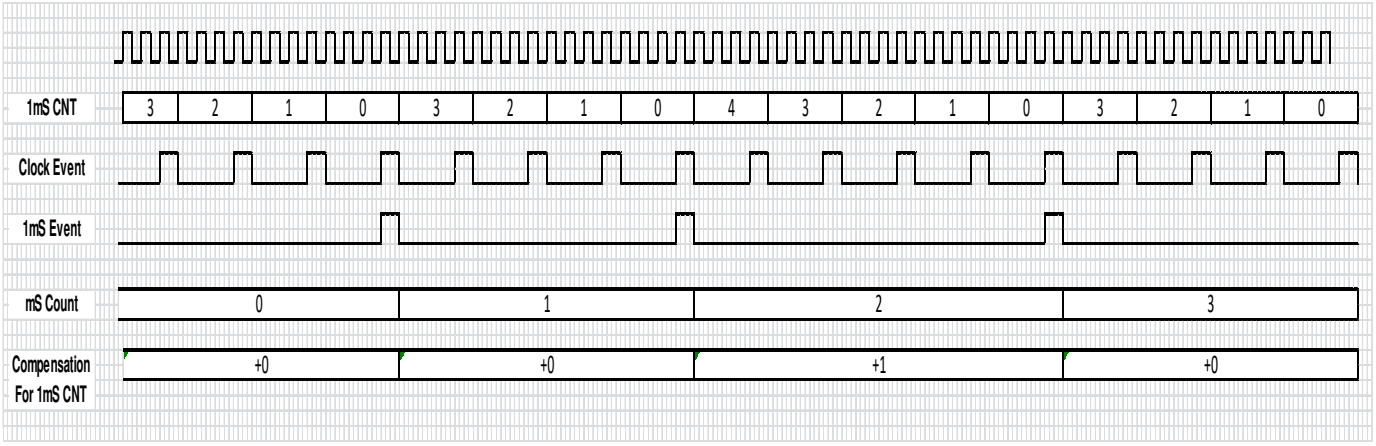

Figure 16-2: 1ms Count and 1ms Counter Relationship

16.2.2 Timeout Event Counter

This counter will count the 1ms event and the time out period is from 4ms to 100ms base on configuration.

16.2.3 Time Stamp Counters

There are total 8 time stamps. Each time stamp has 16 bits and the corresponding interrupt source which could be individual mask out. If the interrupt trigger, the lower 16 bits of 24 bits timer will load to the corresponding time stamp.

16.2.4 PMU and FFE Wakeup

This timer allows Power Management Unit to power up the FFE. Once the FFE power domain is on, a kick-off event is sent to the FFE. There are register controls which control the timing relationship between the power-up of FFE and the kick-off event to FFE. This allows timing flexibility for software writers. Please refer to User Guide for more details.

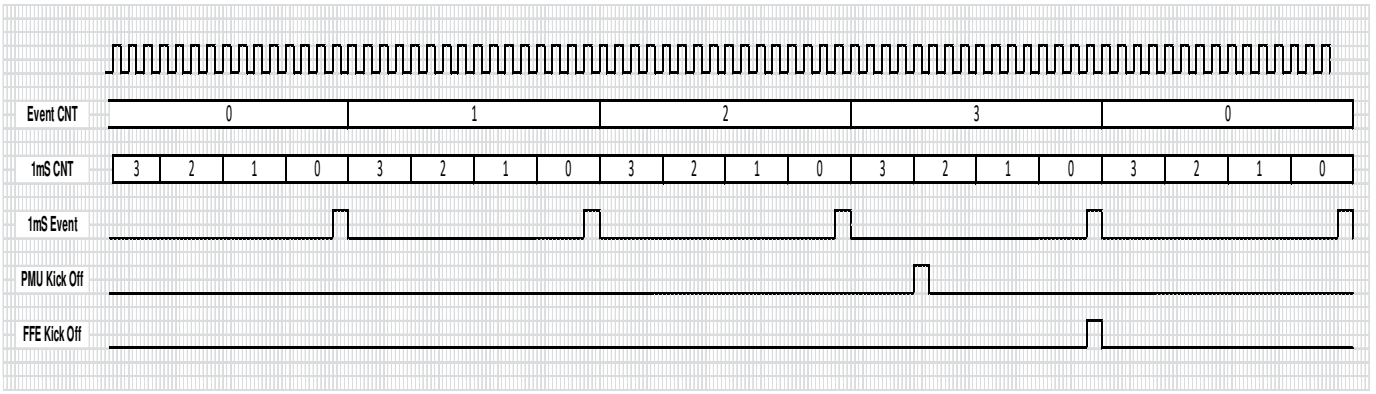

Figure 16-3: PMU and FFE Timing Waveform

16.2.5 Registers for SPT and RTC

Base address = 0x4000_5C00

Table 16-1: SPT and RTC Registers

Offset	Name/Field	Description
0x000	SPT_CFG	SPT Configuration Register
0x004	SLEEP_MODE	Block PMU and FFE Kick Off Signal Control Register
0x008	ERROR_CMP_40M	40 msec Increment Error Compensation Register
0x00C	ERROR_CMP_1S_0	1 sec Increment Error Compensation 0 Register
0x010	ERROR_CMP_1S_1	1 sec Increment Error Compensation 1 Register
0x014	ERROR_CMP_1S_2	1 sec Increment Error Compensation 2 Register
0x018	ERROR_CMP_1S_3	1 sec Increment Error Compensation 3 Register
0x01C	ERROR_CMP_RTC_0	2, 4, 6, 8, 16 sec Increment Error Compensation Register
0x020	ERROR_CMP_RTC_1	32, 64, 128, 256 sec Increment Error Compensation Register
0x024	ERROR_CMP_RTC_2	512, 1024, 2048, 4096 sec Increment Error Compensation Register
0x028	ERROR_CMP_RTC_3	8192, 16384 sec Increment Error Compensation Register
0x02C	UPDATE_TIMER_VALUE	Update 30-bit Timer Value Register
0x030	SPARE	Spare 8-bit Register
0x034	TIMER_VALUE	Current 30-bit Timer Value Register
0x038	EVENT_CNT_VALUE	Current 8-bit Event Counter Value Register
0x03C	MS_CNT_VALUE	Current 8-bit 1 msec Counter Value Register

Table 16-2: SPT_CFG Register

0x000	SPT_CFG				
	BIT FIELD	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	31:22			
	PMU_TO_PERIOD	21:18	RW	0x0	PMU Kick Off (Time out Period) Configuration PMU Kick Off signal will assert N cycles before FFE kick signal asserting 0x1: 1 Clock Event before FFE Kick off 0x2: 2 Clock Event before FFE kick off 0x3: 3 Clock Event before FFE Kick off … 0xF: 15 Clock Event before FFE Kick Off NOTE: For 0x0, FFE Time out Event will be used to wake up FFE power domain.
	FFE_TO_PERIOD	17:10	RW	0x0	FFE Kick Off (Time Out) Period Configuration 0x4: 4msec 0x5: 5msec … 0x64: 100msec Others: Reserved NOTE: If 0x0, The Event Counter will be turn off and all the Signals to PMU will be de-asserted. NOTE: If PMU_TO_PERIOD is 0x0, FFE Kick off signal will not trigger until FFE is alive.
	INT_MASK_N	9:2	RW	0x0	1'b0: INT Trigger signal will be masked. 1'b1: INT Trigger signal is used. Bit 2: For INT source 0 Bit 3: For INT source 1 … Bit 9: For INT source 7
	CLK_SRC_SEL	1	RW	0x0	1'b0: Clock Source is 32KHz 1'b1: Clock Source is 16KHz
	SPT_EN	0	RW	0x0	1'b0: all counters will be reset to 0 except 30-bit counter. 1'b1: Turn on Counter/Timer.

Table 16-3: SLEEP_MODE Register

0x004	SLEEP_MODE				
	BIT FIELD	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	31:1			
	SLEEP_MODE	0	RWHC	0x0	1'b1: The PMU and FFE kick off signal will be blocked. This bit will be cleared by HW if any of non-masked interrupts are triggered

Table 16-4: ERROR_CMP_40M

0x008	ERROR_CMP_40M	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_40M	19:0	RW	0x0	1'b1: Increase # of counter of the corresponding 1msec event by 1. Bit 0: For 1st 1msec event for 40msec period Bit 1: For 3rd 1msec event for 40msec period Bit 2: For 5th 1msec event for 40msec period … Bit 18: for 37th 1msec event for 40msec period Bit 19: For 39th 1msec event for 40msec period

Table 16-5: ERROR_CMP_1S_0

0x00C	ERROR_CMP_1S_0	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_1S_0	31:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 1st 1msec Event for 1S period Bit 3:2: For 41th 1msec Event for 1S period Bit 5:4: For 81th 1msec Event for 1S period Bit 7:6: For 121th 1msec Event for 1S period … Bit 31:30: For 601th 1msec Event for 1S period

Table 16-6: ERROR_CMP_1S_1

0x010	ERROR_CMP_1S_1	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_1S_1	17:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 641th 1msec Event for 1S period Bit 3:2: For 681th 1msec Event for 1S period Bit 5:4: For 721th 1msec Event for 1S period Bit 7:6: For 761th 1msec Event for 1S period … Bit17:16: For 961th 1msec Event for 1S period

Table 16-7: ERROR_CMP_1S_2

0x014	ERROR_CMP_1S_2	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_1S_2	23:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 80th 1msec Event for 1S period Bit 3:2: For 160th 1msec Event for 1S period Bit 5:4: For 240th 1msec Event for 1S period Bit 7:6: For 320th 1msec Event for 1S period … Bit23:22: For 960th 1msec Event for 1S period

Table 16-8: ERROR_CMP_1S_3

0x018	ERROR_CMP_1S_3	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_1S_3	11:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 80th 1msec Event for 1S period Bit 3:2: For 160th 1msec Event for 1S period Bit 5:4: For 240th 1msec Event for 1S period Bit 7:6: For 320th 1msec Event for 1S period … Bit23:22: For 960th 1msec Event for 1S period

Table 16-9: ERROR_CMP_RTC_0

0x01C	ERROR_CMP_RTC_0	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_RTC_0_2	27:24	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 25:24: For 1002th 1msec Event for 2S Period Bit 27:26: For 1450th 1msec Event for 2S Period
	ERROR_CMP_RTC_0_4	23:16	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 17:16: For 2002th 1msec Event for 4S Period Bit 19:18: For 2450th 1msec Event for 4S Period Bit 21:20: For 3002th 1msec Event for 4S Period Bit 23:22: For 3450th 1msec Event for 4S Period
	ERROR_CMP_RTC_0_8	15:8	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 9:8: For 4002th 1msec Event for 8S Period Bit 11:10: For 4450th 1msec Event for 8S Period Bit 13:12: For 6002th 1msec Event for 8S Period Bit 15:14: For 6450th 1msec Event for 8S Period
	ERROR_CMP_RTC_0_16	7:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 4002th 1msec Event for 16S Period Bit 3:2: For 4450th 1msec Event for 16S Period Bit 5:4: For 6002th 1msec Event for 16S Period Bit 7:6: For 6450th 1msec Event for 16S Period

Table 16-10: ERROR_CMP_RTC_1

0x020	ERROR_CMP_RTC_1	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_RTC_0_32	31:24	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 25:24: For 16002th 1msec Event for 32S Period Bit 27:26: For 16450th 1msec Event for 32S Period Bit 29:28: For 24450th 1msec Event for 32S Period Bit 31:30: For 24450th 1msec Event for 32S Period
	ERROR_CMP_RTC_0_64	23:16	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b10: Decrease # of counter of the corresponding 1msec event by 1. 2'b11/2'b00: No change on the counter value Bit 17:16: For 32002th 1msec Event for 64S Period Bit 19:18: For 32450th 1msec Event for 64S Period Bit 21:20: For 48002th 1msec Event for 64S Period Bit 23:22: For 48450th 1msec Event for 64S Period
	ERROR_CMP_RTC_0_128	15:8	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 9:8: For 64002th 1msec Event for 8S Period Bit 11:10: For 64450th 1msec Event for 128S Period Bit 13:12: For 96002th 1msec Event for 128S Period Bit 15:14: For 96450th 1msec Event for 128S Period
	ERROR_CMP_RTC_0_256	7:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 128002th 1msec Event for 256S Period Bit 3:2: For 128450th 1msec Event for 256S Period Bit 5:4: For 192002th 1msec Event for 256S Period Bit 7:6: For 192450th 1msec Event for 256S Period

Table 16-11: ERROR_CMP_RTC_2

0x024	ERROR_CMP_RTC_2	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_RTC_0_512	31:24	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 25:24: For 256002th 1msec Event for 512S Period Bit 27:26: For 256450th 1msec Event for 512S Period Bit 29:28: For 384002th 1msec Event for 512S Period Bit 31:30: For 384450th 1msec Event for 512S Period
	ERROR_CMP_RTC_0_1024	23:16	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 17:16: For 512002th 1msec Event for 1024S Period Bit 19:18: For 512450th 1msec Event for 1024S Period Bit 21:20: For 768002th 1msec Event for 1024S Period Bit 23:22: For 768450th 1msec Event for 1024S Period
	ERROR_CMP_RTC_0_2048	15:8	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 9:8: For 1024002th 1msec Event for 2048S Period Bit 11:10: For 1024450th 1msec Event for 2048S Period Bit 13:12: For 1536002th 1msec Event for 2048S Period Bit 15:14: For 1536450th 1msec Event for 2048S Period
	ERROR_CMP_RTC_0_4096	7:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 2048002th 1msec Event for 4096S Period Bit 3:2: For 2048450th 1msec Event for 4096S Period Bit 5:4: For 3072002th 1msec Event for 4096S Period Bit 7:6: For 3072450th 1msec Event for 4096S Period

Table 16-12: ERROR_CMP_RTC_3

0x028	ERROR_CMP_RTC_3	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	ERROR_CMP_RTC_0_8192	15:8	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 9:8: For 4096002th 1msec Event for 8192S Period Bit 11:10: For 4096450th 1msec Event for 8192S Period Bit 13:12: For 6144002th 1msec Event for 8192S Period Bit 15:14: For 6144450th 1msec Event for 8192S Period
	ERROR_CMP_RTC_0_16384	7:0	RW	0x0	2'b01: Increase # of counter of the corresponding 1msec event by 1. 2'b11: Decrease # of counter of the corresponding 1msec event by 1. 2'b10/2'b00: No change on the counter value Bit 1:0: For 8192002th 1msec Event for 160384S Period Bit 3:2: For 8192450th 1msec Event for 160384S Period Bit 5:4: For 12288002th 1msec Event for 160384S Period Bit 7:6: For 12288450th 1msec Event for 160384S Period

Table 16-13: UPDATE_TIMER_VALUE

0x02C	UPDATE_TIMER_VALUE	31:0			
	Bit Field	bit	R/W/C	Default	Description
	UPDATE_TIMER_VALUE	29:0	RW	0x0	Update the 30-Bit Timer once write Note: Please program SPT_EN (offset 0x000, bit 0) to 0 before writing to this register to avoid any potential issue

Table 16-14: SPARE_BITS

0x030	SPARE_BITS	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	SPARE_BITS	7:0	RW	0x0	Spare

Table 16-15: TIMER_VALUE

0x034	TIMER_VALUE	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	TIMER_VALUE	29:0	RO	0x0	Return the Value of 30-bits, in 1msec resolution. This is the RTC value

Table 16-16: EVENT_CNT_VALUE

0x038	EVENT_CNT_VALUE	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	EVENT_CNT_VALUE	7:0	RO	0x0	Return the Value of the Event counter generating FFE Time out event It is an up-count counter, in 1msec resolution.

Table 16-17: MS_CNT_VALUE

0x03C	MS_CNT_VALUE	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	MS_CNT_VALUE	7:0	RO	0x40	Return the Value of the 1msec counter which is generating the 1msec event. It is a downcount counter. Default is 0x40 (65-1) For 32KHz input, the resolution is ~15uS For 16KHz input, the resolution is ~30uS

Chapter 17. Analog IP (AIP) block

The Analog IP block consists of the RTC, OSC, and LDO modules. Items in this block relate to control of oscillators, timers, or DC voltage regulators.

17.1 Real time clock

discussion to be added

For RTC registers, see Section 17.5.

17.2 HS_OSC

discussion to be added

17.3 APC

discussion to be added

17.4 LDO

discussion to be added

17.5 AIP group registers

Base address = 0x4000_5400

Table 17-1: RTC_CTRL_1 Register

0x004	RTC_CTRL_1	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:5			
	CLKDIV	4:0	RW	0x0	5'b00000 : 32.768 Khz

Table 17-2: RTC_CTRL_2 Register

0x008	RTC_CTRL_2	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	TEST_CTRL	2	RW	0x0	RTC test[4:3] control for rtc bypass mode 0: test[4:3] will be forced to 2'b11 when pad8 is strapped to 1, and forced to 2'b00 when pad8 is strapped to 0 1: normal mode; test[4:0] controlled from 0x1C
	BYP16K	1	RW	0x0	Changes internal clock division for 16384 Hz bypass compatibility 1'b0 : xtal is 32KHz 1'b1 : xtal is 16KHz
	CLKE	0	RW	0x1	1'b1 RTC Clock Output Enable (No SYNC Needed)

Table 17-3: RTC_CTRL_3 Register

0x00C	RTC_CTRL_3	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:3			
	CE	2:0	RW	0x0	3'b000 Normal Function Please refer RTC datasheet for detail

Table 17-4: RTC_CTRL_4 Register

0x010	RTC_CTRL_4	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	WR	0	RW	0x0	Write Pulse to program the RTC internal Register Please refer RTC datasheet for detail

Table 17-5: RTC_CTRL_5Register

0x014	RTC_CTRL_5	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	C	0	RW	0x0	Serial Input Clock

Table 17-6: RTC_CTRL_6 Register

0x018	RTC_CTRL_6	31:0			
	Bit Field	bit	R/W/C	Default	Description
	PI	31:0	RW	0x0	Parallel Input data Please refer RTC datasheet for detail

Table 17-7: RTC_CTRL_7 Register

0x01C	RTC_CTRL_7	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:5			
	TEST	4:0	RW	0x0	Please refer RTC datasheet for detail

Table 17-8: RTC_STA_0 Register

0x020	RTC_STA_0	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:4			
	oscok	3	RHW	0x0	Please refer RTC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	digtestbus	2	RHW	0x0	Please refer RTC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	testreq	1	RHW	0x0	Please refer RTC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	alarm	0	RHW	0x0	Please refer RTC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.

Table 17-9: RTC_STA_1 Register

0x024	RTC_STA_1	31:0			
	Bit Field	bit	R/W/C	Default	Description
	PO	31:0	RHW	0x0	Please refer RTC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.

Table 17-10: Oscillator Control 0 Register Bit Description

0x080	OSC_CTRL_0	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	FREF16K_SEL	1	RW	0x0	1'b0 : reference clock is 32KHz 1'b1 : reference clock is 16KHz
	EN	0	RW	0x1	1'b0 : OSC OFF 1'b1 : OSC ON (NO SYNC needed, OSC guarantee there is no Glitch)

Table 17-11: Oscillator Control 1 Register Bit Description

0x084	OSC_CTRL_1	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:16			
	GENERAL_PURPOS_SFR	15:13	RW	0x00	
	PROG	12:0	RW	0x92D	Please refer OSC datasheet for others (No SYNC Needed) Power On Default Value is 76.97 MHz No Support on "Delta Mode"

Note:

Default value of 0x92D = 2349 decimal. 32768 Hz × 2349 = 76 972 032 Hz or 76.97 MHz

Table 17-12: OSC_CTRL_2 Register

0x088	OSC_CTRL_2	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	DELTA	0	RW	0x0	**Reserved**

Table 17-13: OSC_CTRL_3 Register

0x08C	OSC_CTRL_3	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:8			
	GENERAL_PURPOS_SFR	7:4	RW	0x0	
	ENMON	3	RW	0x1	Turn on Monitor function by default
	RESERVED	**2:1**	**RO**	**0x0**	**Reserved**
	REFOK	0	RW	0x0	If 1'b1, will force the refok pin to 1, otherwise, it is control by the RTC/oscok

Table 17-14: OSC_CTRL_4 Register

0x090	OSC_CTRL_4	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:6			
	WR	5	RW	0x0	Please refer OSC datasheet for detail
	CE	4:3	RW	0x0	Please refer OSC datasheet for detail
	TEST	2:0	RW	0x0	Please refer OSC datasheet for detail

Table 17-15: OSC_CTRL_5 Register

0x094	OSC_CTRL_5	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	SDI	0	RW	0x0	Please refer OSC datasheet for detail

Table 17-16: OSC_CTRL_6 Register

0x098	OSC_CTRL_6	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	SCK	0	RW	0x0	Please refer OSC datasheet for detail

Table 17-17: OSC_STA_0 Register

0x0A0	OSC_STA_0	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:2			
	ANATESTREQ	1	RHW	0x0	Please refer OSC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	LOCK	0	RHW	0x0	Please refer OSC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.

Table 17-18: OSC_STA_1 Register

0x0A4	OSC_STA_1	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	SDO	0	RHW	0x0	Please refer OSC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.

Table 17-19: APC_CTRL_0 Register

The Analog Power Controller group is part of the PMU

0x100	APC_CTRL_0	31:0			
	Bit Field	bit	R/W/C	Default	Description
	RESERVED	31:1			
	DIS	0	RO	0x0	1'b0 : APC ON, Always ON

Table 17-20: APC_CTRL_1 Register

0x104	APC_CTRL_1	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:8**			
	VT	7:3	RW	0x00	Please refer OSC datasheet for others (No SYNC Needed)
	TT	2:0	RW	0x00	Please refer OSC datasheet for others (No SYNC Needed)

Table 17-21: APC_CTRL_2 Register

0x108	APC_CTRL_2	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:7**			
	TEST	6:3	RW	0x0	Please refer OSC datasheet for others (No SYNC Needed)
	IT	2:0	RW	0x0	Please refer OSC datasheet for others (No SYNC Needed)

Table 17-22: APC_STA_0 Register

0x120	APC_STA_0	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:5**			
	TESTREQ	4	RHW	0x0	Please refer APC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	DIGTESTBUS	3	RHW	0x0	Please refer APC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	PORZ	2	RHW	0x1	Please refer APC datasheet for detail, NO SYNC, FW needs to read it twice to ensure the value.
	RESERVED	**1:0**	**RO**	**0x0**	**Reserved**

Table 17-23: RING_OSC Register

0x180	RING_OSC	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:8**			
	GENERAL_PURPOS_SFR	7:1	RW	0x00	
	RING_OSC_EN	0	RW	0x0	1'b1: Turn on the RING OSC, Ring OSC will be always on despite the HW control.

Table 17-24: LDO_30_CTRL_0 Register

0x200	LDO_30_CTRL_0	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:10**			**Reserved**
	LDO_30_DI	9:5	RW	0x11	Output voltage programming DI Vo (V) 00000 0.75 00001 0.77 00010 0.79 00011 0.81 00100 0.83 00101 0.85 00110 0.87 00111 0.89 01000 0.91 01001 0.93 01010 0.95 01011 0.97 01100 0.99 01101 1.01 01110 1.03 01111 1.05 10000 1.07 **10001 1.09** 10010 1.11 10011 1.13 10100 1.15 10101 1.17 10110 1.19 10111 1.21
	LDO_30_IMAX	4:2	RW	0x3	Configures the control for maximum expected current imax current (mA) 000 1 001 2 010 4 **011 8** 100 12 101 16 110 22 111 30
	LDO_30_DISPG	1	RW	0x0	It is used to disable the power good comparator. 1'b0 : Enable 1'b1 : Disable (the pg output is forced to “1”)
	LDO_30_DIS	0	RW	0x0	Used to disable LDO 1'b0 : Enable 1'b1 : Disable

Table 17-25: LDO_30_CTRL_1 Register

0x204	LDO_30_CTRL_1	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:5**			**Reserved**
	LDO_30_TESTREQ	4	RO	0x0	Request of connection of the ANATESTBUS to an external pin for characterization - **Reserved**
	LDO_30_TEST	3:0	RW	0x0	Lab test and internal block characterization test control pins - **Reserved**

Table 17-26: LDO_50_CTRL_0 Register

0x210	LDO_50_CTRL_0	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:10**			**Reserved**
	LDO_50_DI	9:5	RW	0x11	Output voltage programming DI Vo (V) 00000 0.75 00001 0.77 00010 0.79 00011 0.81 00100 0.83 00101 0.85 00110 0.87 00111 0.89 01000 0.91 01001 0.93 01010 0.95 01011 0.97 01100 0.99 01101 1.01 01110 1.03 01111 1.05 10000 1.07 **10001 1.09** 10010 1.11 10011 1.13 10100 1.15 10101 1.17 10110 1.19 10111 1.21
	LDO_50_IMAX	4:2	RW	0x3	Configures the control for maximum expected current imax current (mA) 000 1 001 2 010 4 **011 8** 100 12 101 20 110 30 111 50
	LDO_50_DISPG	1	RW	0x0	It is used to disable the power good comparator. 1'b0 : Enable 1'b1 : Disable (the pg output is forced to “1”)
	LDO_50_DIS	0	RW	0x0	Used to disable LDO 1'b0 : Enable 1'b1 : Disable

Table 17-27: LDO_50_CTRL_1 Register

0x214	LDO_50_CTRL_1	31:0			
	Bit Field	**bit**	**R/W/C**	**Default**	**Description**
	RESERVED	**31:5**			**Reserved**
	LDO_50_TESTREQ	4	RO	0x0	Request of connection of the ANATESTBUS to an external pin for characterization - **Reserved**
	LDO_50_TEST	3:0	RW	0x0	Lab test and internal block characterization test control pins - **Reserved**

Chapter 18. Analog-to-Digital Converter (ADC)

In power domain: AVDD

This module's inputs are configured to I/O pads.

18.1.1 Overview

The ADC is an accurate high resolution ADC for voltage monitoring. To achieve excellent repeatability and high PSRR, the ADC uses a 12-bit advanced fully differential Delta-Sigma ADC. The ADC is specified from T_J = -40°C to +125°C and it is designed to achieve 2.0% overall accuracy. Figure 18-1 shows a general block diagram of the ADC.

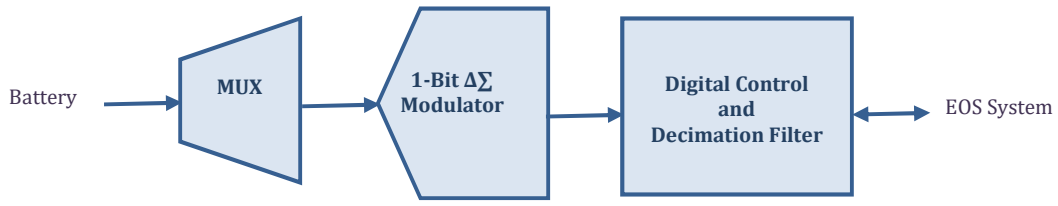

Figure 18-1: ADC Block Diagram

18.1.2 Functional Description

The ADC voltage measurement core includes an input multiplexer, a delta-sigma modulator, and a digital control core. The digital control core controls the analog blocks power up/down sequences, generates the delta-sigma control signals and implements the output decimation.

18.1.3 PCB Layout Recommendations

Please implement the following PCB layout recommendations when using the ADC:

1. Try to minimize the distances between the ADC input and the location for the voltage being measured. Avoid routing close to noise sources, i.e., clock generators, DC/DC converters and data/address buses.
2. Use wide tracks to minimize inductance and reduce series resistance. Use grounded guard traces when possible. A trace with a 10 mil minimum width and spacing is recommended.

18.1.4 Example Application

The ADC accepts a maximum input voltage of 1.6 VDC. This requires certain design elements in order to measure a fully charged battery that produces an output in excess of 4 VDC. For that case, an external analog level shifter (not a power-wasting passive resistive divider!) is needed to scale the battery voltage so that it is compatible with the ADC. The following figure shows one such method:

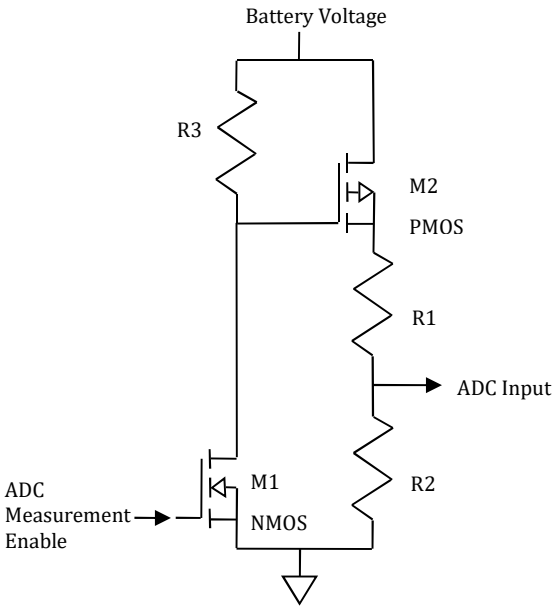

Figure 18-2: Example Voltage Divider Circuit

The voltage divider resistors, R1 and R2 should be chosen according to the amount of voltage scaling required. It is left to software to scale the ADC values in order to determine the corresponding battery voltage level.

In addition, the ADC provides an enable output for the specific purpose of controlling an external voltage divider. In this example, this enable controls components, M1, M2, and R3. These components allow the ADC to disable the voltage divider between ADC measurements. Not doing so will result in a constant current draw on the battery. This current draw can result in reduced battery life. An additional benefit is that these components disable the voltage divider when the EOS device is in a low power state.

18.1.5 ADC Registers

See section 38.8 for details.

Chapter 19. S3 GPIO

19.1 IO Mux and GPIO Introduction

In power domain: VCCIO A / B

The S3 GPIO functions provide IO to the pads. The IO Multiplexer (IOMUX) controls the configuration of the IO and IO operations.

There are total of 46 pads (or IOs) that can be multiplexed to/from various functions on-chip. Each pad can be independently programmed as inputs pads, outputs pads, or bidirectional pads on the chip. Programming the IOMUX registers (0x40004C00 – 0x40004DAC) will control the various on-chip functions which can drive and sample the pad values. The behavior of each register is the same as that for the other 45 registers. Therefore, we provide a discussion of the abstract 'PAD_x CTRL' register as example for all the registers.

NOTE: PAD is the equivalent of IO

19.2 IO Mux Overview

On EOS S3, the IOMUX register block (0x40004C00 – 0x40004DAC) controls what drives/samples the 46 pads for BGA package.

Below is an overview of the block diagram of the implementation.

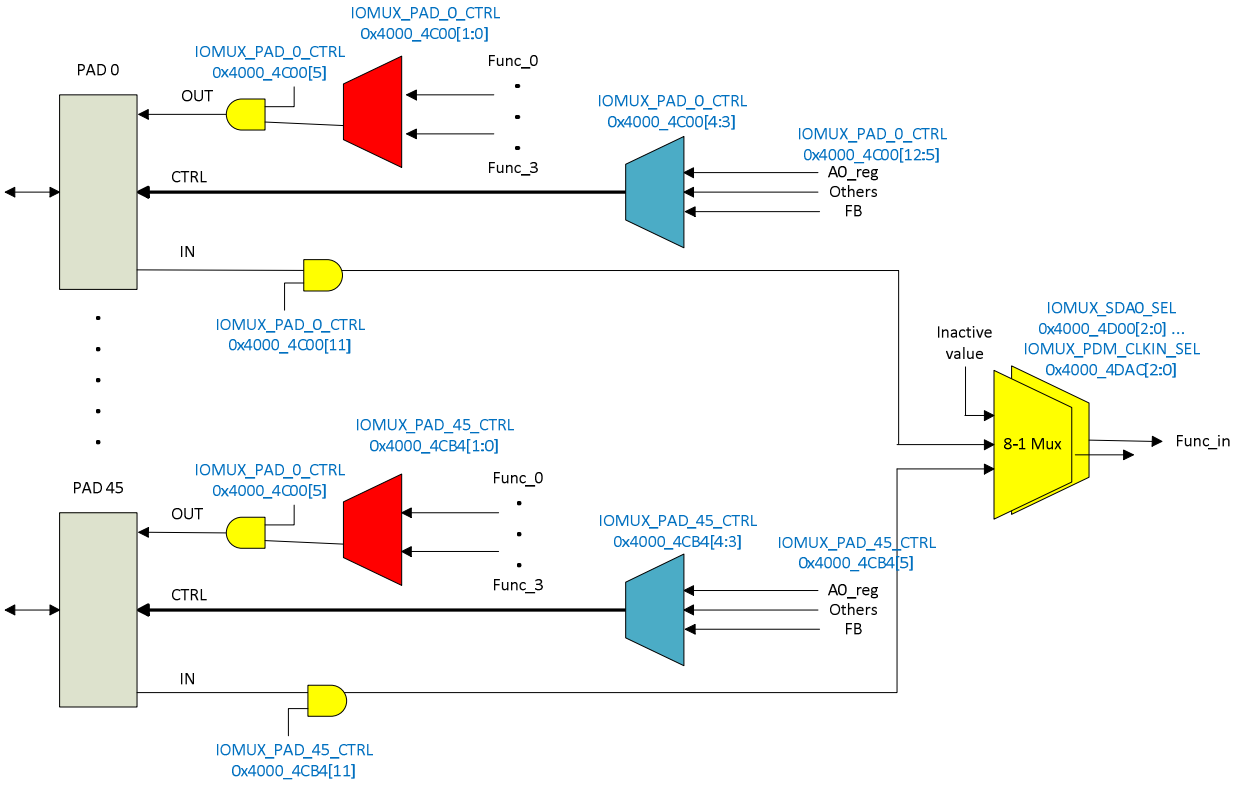

Figure 19-1: IOMux block diagram

From Figure above, the red-colored muxes selects between which output function can drive to a pad. There is 1 red mux for each pad and are controlled by IOMUX_PAD_0_CTRL[1:0].

The blue-colored muxes selects the output pad control, which can come from:

- IOMUX_PAD_x_CTRL[12:5] (register controlled, e.g. A0 register)
- Or Other - controlled for bidirectional signals e.g. SWD, I2C controllers
- Or Fabric (FB) IO controlled. These muxes are for output and/or bidirectional signaling from the chip to system.

The yellow-colored "AND" gate and mux allows the chip to sample pad values to various functional blocks within the chip. This logic is for input and bidirectional signaling to the chip from the system.

Here are some considerations to keep in mind when doing pad allocation in your system design for EOS S3.

- M4, AP (via SPI slave), Debugger (SWD) can read/write the IOMUX registers. The IOMUX registers are located in the Always-ON power domain. If the power is cycled, then the IOMUX register settings will default back to its reset state. If a chip reset is asserted, then also the IOMUX registers will default back to its reset state. Fabric, which is a slave IP, cannot configure the IOMUX registers. The M4, AP, or Debugger is required to setup the IOMUX settings for Fabric to use.
- 8 pads can be used as general purpose IO (GPIOs) which the M4 can drive and sample, using IOMUX_IO_REG_SEL (0x40004D60), MISC_IO_INPUT (0x40005100), MISC_IO_OUTPUT (0x40005104) register controls.
- Fabric can control all 46 pads, assuming the M4 setups the appropriate IOMUX programming.
- 8 of the pads can be used as external interrupts to the system (e.g. sensor interrupt or gpio interrupt) – either as interrupts to M4 or sensor interrupts to FFE.
- The other pads can be used for serial interfaces: SPI slave, multiple SPI masters, Serial Wire Debug (SWD), UART, IRDA, I2C masters, Audio microphone inputs (PDM or I2S), and I2S output to AP.
- 8 of the pads can be used as debug monitoring pins (debug_mon). Some of PMU, CRU, Audio, FFE internal signaling can be brought out to 8 pads. For example, CRU can bring out clocks to a pad to monitor frequency.

Please refer to the Appendix and open the embedded .xlsx. Refer to mode_config tab in EOS S3 IOMUX xls for functions which can be assigned to which pads. This is a useful tab for system design, as it shows all the possible IO assignments.

IMPORTANT: Determine functions for all 46 pads first as per your system design, then program the IOMUX register accordingly.

The next sections refer to how to program IOMUX registers.

19.3 How to Select Output Function

Each IO output can be driven from 1 of 4 different functional outputs. Refer to figure 19-1 for details. Only 1 function can drive to the pad as output.

There are 46 IOMUX_PAD_x_CTRL registers where x goes from 0 to 45, one register per each pad. Program the corresponding register IOMUX_PAD_x_CTRL bits[1:0] will select the function which drives corresponding pad.

Each IOMUX_PAD_x_CTRL register looks similar to the following. Below is an example of such a register.

Table 19-1: IOMUX_PAD_O_CTRL Register

Offset	Name/Field	Bits	Type	Default	Description
0x40004C00	IOMUX_PAD_0_CTRL	31:0			
	pad_0_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	pad_0_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	pad_0_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	pad_0_P	7:6	RW	0x1	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	pad_0_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	pad_0_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	pad_0_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	pad_0_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

See EOS S3 IOMUX Assignments section for control details of IOMUX_PAD_0_CTRL and pad_0_CTRL_SEL.

Here is a partial list of pad 0 – 2 from the EOS S3 IOMUX Assignments section. The first column shows the pad assignment. The second column shows the mux select for outputs. The 3rd column shows the functional output. This table shows the select encoding for the output mux.

Table 19-2: EOS S3 IOMUX Select (partial)

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Pad Type	Default Ctrl
0	0	SCL0	BIDIR	A0_Reg
0	1	FBIO_0	BIDIR	
1	0	SDA0	BIDIR	A0_Reg
1	1	FBIO_1	BIDIR	
2	0	FBIO_2	BIDIR	A0_Reg
2	1	spi_sensor_ssn2	OUT	
2	2	debug_mon[0]	OUT	
2	3	batt_mon	OUT	
2		sensor_interrupt	IN	

Below are examples of possible configuration:

Example 1: Program IOMUX_PAD_0_CTRL[1:0]:

- Set to 0 for I2C0 CLK (SCL0) to drive pad 0
- Set to 1 for Fabric IO 0 (FBIO_0) to drive pad 0
- 2 and 3 are not assigned and are reserved.

Example 2: Program IOMUX_PAD_1_CTRL[1:0]:

- Set to 0 for I2C0 DAT (SDA0) to drive pad 1
- Set to 1 for Fabric IO 1 (FBIO_1) to drive pad 1
- 2 and 3 are not assigned and are reserved.

Example 3: Program IOMUX_PAD_2_CTRL[1:0]:

- Set to 0 for Fabric IO 2 (FBIO_2) to drive PAD2
- Set to 1 for SPI Sensor chip Select 2 (spi_sensor_ssn2) to drive PAD2
- Set to 2 for bring out debug monitor point (debug_mon[0]) to drive PAD2
- Set to 3 for bring out battery monitor IO (batt_mon) to drive PAD2

Note: If the pad is used as input pad, then bits IOMUX_PAD_x_CTRL[1:0] can be ignored or left as is. Pad 2 can also be an input pad assigned to sensor interrupt.

Note: IOMUX_PAD_43_CTRL (0x40004CAC) is protected from accidental writes. In order to program this register, please program MISC_LOCK_EN (0x40005310) to 0x1ACCES551 prior to writing IOMUX_PAD_43_CTRL.

Table 19-3: MISC_LOCK_KEY_CTRL Register Definition

Offset	Name/Field	Bits	Type	Default	Description
0x40005310	MISC_LOCK_KEY_CTRL	31:0			
	LOCK_KEY	31:1	WO	0x0	Enable write access to all below registers by writing 0x1ACCE551. Disable write access by writing any other value. M4 WDT Intr/reset clear 0x4000_4830[4:3] M4 WDT Intr/reset enable AP 0x4000_4834[4:3] Pad #43 (AP_INTR) 0x40004CAC[12:0] M4 Low Power Configuration 0x40004484[1:0] M4 WDT Clock Gate 0x40004054[0]
	LOCK_KEY_EN	0	RW	0x1	0: lock disabled, write to register enabled 1: lock enable, write to register disabled

The pad controls for the pad is controlled by IOMUX_PAD_x_CTRL[4:3]. This is the blue-colored mux seen in Figure 19-1.

Table 19-4: IOMUX_PAD_0_CTRL Register Definition - Bits 4:3

Offset	Name/Field	Bits	Type	Default	Description
0x40004C00	IOMUX_PAD_0_CTRL	31:0			
	pad_0_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric

- Select 0, then output enable from pad is controlled by IOMUX_PAD_x_CTRL[5]. Since IOMUX is located in A0 power domain (always-on power domain), this is consider an A0 register.
    - Program IOMUX_PAD_x_CTRL[5]
        - Set to 0 to be an output pad.
        - Or set to 1 to be an input pad. Output driver within pad is disabled.

Table 19-5: IOMUX_PAD_CTRL Register Definition - Bit 5

Offset	Name/Field	Bits	Type	Default	Description
0x40004C00	IOMUX_PAD_0_CTRL	31:0			
	pad_0_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

- Select 1, if alternate function controls the output enable dynamically. For example, I2C master(s) need bidirectional control of the output enable for the I2C DAT pin (SDAx). For example, Serial Wire Debug (SWD) needs bidirectional control of the output enable for SWD DATA pin (SW_DP_IO).
    - IOMUX_PAD_x_CTRL[5] is ignored.
- Select 2, if Fabric controls the output enable dynamically.
    - IOMUX_PAD_x_CTRL[5] is ignored.

19.4 Selecting an Input Function

Each functional input can be selected from a possible 8 IOs. See the yellow-colored "AND" gate and mux in Figure 19-1. There are 2 steps for setting up input pads.

1. Program IOMUX_PAD_x_CTRL[11] – Receive enable (yellow-colored "AND" gate):
    - Set 1 to be input pad.
    - Also set 0 to IOMUX_PAD_x_CTRL[5] to disable as output pad.
2. Program input mux to route the pad signal to functional block inside chip (yellow-colored mux):
    - Each functional input has IOMUX_<function>_SEL (0x40004d00 - 0x40004dac).

Table 19-6: IOMUX_SDA0_SEL / IOMUX_SDA1_SEL Register Definitions

Offset	Name/Field	Bits	Type	Default	Description
0x40004D00	IOMUX_SDA0_SEL	31:0			
	SDA0_SEL	0	RW	0x0	Sel 0: 1 1: pad #1
0x40004D04	IOMUX_SDA1_SEL	31:0			
	SDA1_SEL	1:0	RW	0x0	Sel 0: 1 1: pad #15 2: pad #32 3: pad #44

Example 1: I2C0 DAT pin (SDA0) can get its input from PAD1.

- Program IOMUX_SDA0_SEL (0x40004D00) = 1.
  - This enables the yellow mux between PAD1 to I2C0 master.
- Program IOMUX_PAD_1_CTRL = 0x803.
  - Bit[1:0] = 0, selects the SDA0 function
  - Bit[4:3] = 1 selects the Other function. This means I2C0 master block will control the output enable.
  - Bit[5] is ignored. Can write 0 for now.
  - Bit[11] =1 , sets receive enable. This is required to I2C0 master to see I2C0 DAT pin change, since this bidirectional pin.

Example 2: I2C1 DAT pin (SDA1) can get its input from 3 possible pads: PAD15, pad 32, PAD44.

Option1 if PAD15 selected:

- Program IOMUX_SDA1_SEL (0x40004D04) = 1 for pad 15.
- Program IOMUX_PAD_15_CTRL (0x40004C3C) = 0x80A
  - Bit[1:0] = 2 to select SDA_1 for PAD15. See Table 7 below.
  - Bit[4:3] = 1 to select Other function. This means I2C1 master block will control the output enable.
  - Bit[5] is ignored. Can write 0 for now.
  - Bit[11] =1, sets receive enable. This is required for I2C1 master to see I2C1 DAT pin change, since this bidirectional pin.

Option2 if PAD32 selected:

- Program IOMUX_SDA1_SEL (0x40004D04) = 2 for pad 32.
- Program IOMUX_PAD_32_CTRL (0x40004C80) = 0x80B
  - Bit[1:0] = 3 to select SDA_1 for PAD32. See Table 7 below.
  - Bit[4:3] = 1 to select Other function. This means I2C1 master block will control the output enable.
  - Bit[5] is ignored. Can write 0 for now.
  - Bit[11] =1, sets receive enable. This is required for I2C1 master to see I2C1 DAT pin change, since this bidirectional pin.

Option 3 if PAD44 is selected:

- Program IOMUX_SDA1_SEL (0x40004D04) = 3 for pad 44.
- Program IOMUX_PAD_44_CTRL (0x40004CB0) =
  - Bit[1:0] = 2 to select SDA_1 for PAD44. See Table 7 below.
  - Bit[4:3] = 1 to select Other function. This means I2C1 master block will control the output enable.
  - Bit[5] is ignored. Can write 0 for now.
  - Bit[11] =1, sets receive enable. This is required for I2C1 master to see I2C1 DAT pin change, since this bidirectional pin.

See IOMUX Assignment table for details.

19.5 IOMux Assignments

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
0	0	SCL0	BIDIR	A0_Reg
0	1	FBIO_0	BIDIR	
1	0	SDA0	BIDIR	A0_Reg
1	1	FBIO_1	BIDIR	
2	0	FBIO_2	BIDIR	A0_Reg
2	1	spi_sensor_ssn2	OUT	
2	2	debug_mon[0]	OUT	
2	3	batt_mon	OUT	
2		sensor_interrupt	IN	
3	0	S_INTR_0	IN	A0_Reg
3	0	FBIO_3	BIDIR	
4	0	FBIO_4	BIDIR	A0_Reg
4	1	spi_sensor_ssn3	OUT	
4	2	debug_mon[1]	OUT	
4	3	sda_1_dpu	OUT	
4		sensor_interrupt	IN	
5	0	FBIO_5	BIDIR	A0_Reg
5	1	spi_sensor_ssn4	OUT	
5	2	debug_mon[2]	OUT	
5	3	sda_0_dpu	OUT	
5		sensor_interrupt	IN	
6	0	FBIO_6	BIDIR	A0_Reg

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
6	1	spi_sensor_mosi	OUT	
6	2	debug_mon[3]	BIDIR	
6	3	gpio[0]	BIDIR	
6		FCLK	IN	
6		IrDA_Sirin	IN	
6		sensor_interrupt	IN	
7	0	FBIO_7	BIDIR	A0_Reg
7	1	spi_sensor_ssn5	OUT	
7	2	debug_mon[4]	OUT	
7	3	SWV	OUT	
7		sensor_interrupt	IN	
8	0	FBIO_8	BIDIR	A0_Reg
8	1	pdm_cko	OUT	
8	2	i2s_cko	OUT	
8	3	IrDA_Sirout	OUT	
8		sensor_interrupt	IN	
8		spi_sensor_miso	IN	
9	0	FBIO_9	BIDIR	A0_Reg
9	1	spi_sensor_ssn1	OUT	
9	2	i2s_wd_cko	OUT	
9	3	gpio[1]	BIDIR	
9		pdm_stat_i	IN	
9		sensor_interrupt	IN	
10	0	FBIO_10	BIDIR	A0_Reg
10	1	spi_sensor_clk	OUT	
10	2	Reserved		

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
10	3	SWV	OUT	
10		sensor_interrupt	IN	
10		i2s_din	IN	
10		pdm_din	IN	
11	0	FBIO_11	BIDIR	A0_Reg
11	1	spi_sensor_ssn6	OUT	
11	2	debug_mon[5]	OUT	
11	3	gpio[2]	BIDIR	
11		sensor_interrupt	IN	
12	0	FBIO_12	BIDIR	A0_Reg
12	1	spi_sensor_ssn7	OUT	
12	2	debug_mon[6]	OUT	
12	3	IrDA_Sirout	OUT	
12		sensor_interrupt	IN	
13	0	FBIO_13	BIDIR	A0_Reg
13	1	spi_sensor_ssn8	OUT	
13	2	debug_mon[7]	OUT	
13	3	SWV	OUT	
13		sensor_interrupt	IN	
14	0	SW_DP_CLK	IN	Others
14	0	FBIO_14	BIDIR	
14	1	IrDA_Sirout	OUT	
14	2	SCL_1	BIDIR	
14	3	gpio[3]	BIDIR	
14		UART_rxd	IN	
14		sensor_interrupt	IN	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
15	0	SW_DP_IO	BIDIR	Others
15	1	FBIO_15	BIDIR	
15	2	SDA_1	BIDIR	
15	3	UART_txd	OUT	
15		sensor_interrupt	IN	
15		IrDA_Sirin	IN	
16	0	SPIs_CLK	IN	A0_Reg
16	0	FBIO_16	BIDIR	A0_Reg
16		UART_rxd	IN	
17	0	SPIs_MISO	OUT	Others
17	1	FBIO_17	BIDIR	
17		nUARTCTS	IN	
18	0	FBIO_18	BIDIR	A0_Reg
18	1	SWV	OUT	
18	2	debug_mon[0]	OUT	
18	3	gpio[4]	BIDIR	
18		sensor_interrupt	IN	
19	0	SPIs_MOSI	IN	A0_Reg
19	0	FBIO_19	BIDIR	A0_Reg
19	1	nUARTRTS	OUT	
20	0	SPIs_SSn	IN	A0_Reg
20	0	FBIO_20	BIDIR	A0_Reg
20	1	UART_txd	OUT	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
21	0	FBIO_21	BIDIR	A0_Reg
21	1	nUARTRTS	OUT	
21	2	debug_mon[1]	OUT	
21	3	gpio[5]	BIDIR	
21		IrDA_Sirin	IN	
21		sensor_interrupt	IN	
22	0	FBIO_22	BIDIR	A0_Reg
22	1	IrDA_Sirout	OUT	
22	2	debug_mon[2]	OUT	
22	3	gpio[6]	BIDIR	
22		sensor_interrupt	IN	
22		nUARTCTS	IN	
23	0	FBIO_23	BIDIR	A0_Reg
23	1	SPIm_SSn2	OUT	
23	2	SWV	OUT	
23	3	gpio[7]	BIDIR	
23		ap_i2s_wd_clk_in	IN	
23		sensor_interrupt	IN	
24	0	FBIO_24	BIDIR	A0_Reg
24	1	ap_i2s_dout	OUT	
24	2	UART_txd	OUT	
24	3	gpio[0]	BIDIR	
24		IrDA_Sirin	IN	
24		sensor_interrupt	IN	
25	0	FBIO_25	BIDIR	A0_Reg
25	1	SPIm_SSn3	OUT	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
25	2	SWV	OUT	
25	3	IrDA_Sirout	OUT	
25			IN	
25		UART_rxd	IN	
25		sensor_interrupt	IN	
26	0	FBIO_26	BIDIR	A0_Reg
26	1	spi_sensor_ssn4	OUT	
26	2	debug_mon[3]	OUT	
26	3	gpio[1]	BIDIR	
26		sensor_interrupt	IN	
27	0	FBIO_27	BIDIR	A0_Reg
27	1	spi_sensor_ssn5	OUT	
27	2	debug_mon[4]	OUT	
27	3	SPIm_SSn2	OUT	
27		sensor_interrupt	IN	
28	0	FBIO_28	BIDIR	A0_Reg
28	1	spi_sensor_mosi	OUT	
28	2	debug_mon[5]	BIDIR	
28	3	gpio[2]	BIDIR	
28		i2s_din	IN	
28		pdm_din	IN	
28		sensor_interrupt	IN	
28		IrDA_Sirin	IN	
29	0	FBIO_29	BIDIR	A0_Reg
29	1	pdm_cko	OUT	
29	2	i2s_cko	OUT	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
29	3	IrDA_Sirout	OUT	
29		sensor_interrupt	IN	
29		spi_sensor_miso	IN	
30	0	FBIO_30	BIDIR	A0_Reg
30	1	spi_sensor_ssn1	OUT	
30	2	i2s_wd_cko	OUT	
30	3	gpio[3]	BIDIR	
30		pdm_stat_i	IN	
30		sensor_interrupt	IN	
31	0	FBIO_31	BIDIR	A0_Reg
31	1	spi_sensor_clk	OUT	
31	2	Reserved		
31	3	gpio[4]	BIDIR	
31		sensor_interrupt	IN	
31		ap_i2s_clk_in	IN	
32	0	FBIO_32	BIDIR	A0_Reg
32	1	spi_sensor_ssn5	OUT	
32	2	debug_mon[6]	OUT	
32	3	SDA_1	OUT	
32		sensor_interrupt	IN	
33	0	FBIO_33	BIDIR	A0_Reg
33	1	spi_sensor_ssn6	OUT	
33	2	debug_mon[7]	OUT	
33	3	SCL_1	OUT	
33		sensor_interrupt	IN	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
34	0	SPIm_CLK	OUT	Others
34	1	FBIO_34	BIDIR	Others
34	2	ap_pdm_stat_o	BIDIR	
34	3	debug_mon[0]	OUT	
34		sensor_interrupt	IN	
35	0	FBIO_35	BIDIR	A0_Reg
35	1	SPIm_SSn3	OUT	
35	2	spi_sensor_ssn7	OUT	
35	3	debug_mon[1]	OUT	
35		sensor_interrupt	IN	
36	0	SPIm_MISO	IN	Others
36	0	FBIO_36	BIDIR	Others
36	1	SWV	OUT	
36	2	spi_sensor_ssn2	BIDIR	
36	3	gpio[5]	BIDIR	
36		sensor_interrupt	IN	
37	0	FBIO_37	BIDIR	A0_Reg
37	1	SDA_2_DPU	OUT	
37	2	spi_sensor_ssn8	OUT	
37	3	debug_mon[2]	OUT	
37		sensor_interrupt	IN	
38	0	SPIm_MOSI	OUT	Others
38	1	FBIO_38	BIDIR	Others
38	2	debug_mon[3]	BIDIR	
38	3	gpio[6]	BIDIR	
38		sensor_interrupt	IN	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
38		ap_pdm_cko_in	IN	
39	0	SPIm_SSn1	OUT	Others
39	1	FBIO_39	BIDIR	Others
39	2	ap_pdm_io	OUT	
39	3	debug_mon[4]	OUT	
39		sensor_interrupt	IN	
40	0	FBIO_40	BIDIR	A0_Reg
40	1	SCL_2	OUT	
40	2	debug_mon[5]	OUT	
40	3	Reserved	BIDIR	
40		sensor_interrupt	IN	
40		IrDA_Sirin	IN	
41	0	FBIO_41	BIDIR	A0_Reg
41	1	SDA_2	OUT	
41	2	debug_mon[6]	OUT	
41	3	IrDA_Sirout		
41		sensor_interrupt	IN	
42	0	FBIO_42	BIDIR	A0_Reg
42	1	SWV	OUT	
42	2	debug_mon[7]	OUT	
42	3	SDA_1_DPU		
42		sensor_interrupt	IN	
43	0	AP_INTR	OUT	A0_Reg
43	1	FBIO_43	BIDIR	

Pad #	Mux Sel for OUT (default=0)	Ball Name/Function	Type	Default Ctrl
44	0	SW_DP_IO	BIDIR	Other
44	1	FBIO_44	BIDIR	A0_Reg
44	2	SDA_1	BIDIR	
44	3	UART_txd	BIDIR	
44		sensor_interrupt	IN	
44		IrDA_Sirin	IN	
45	0	SW_DP_CLK	IN	Other
45	0	FBIO_45	BIDIR	A0_Reg
45	1	IrDA_Sirout	OUT	
45	2	SCL_1	OUT	
45	3	gpio[7]	BIDIR	
45		sensor_interrupt	IN	
45		UART_rxd	IN	

19.6 GPIO Registers

Offset	Name/Field	Bits	Type	Default	Description
0x000	PAD_0_CTRL	31:0			
	PAD_0_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux chapter
	PAD_0_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_0_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_0_P	7:6	RW	0x1	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_0_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_0_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_0_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_0_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x004	PAD_1_CTRL	31:0			
	PAD_1_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_1_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_1_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_1_P	7:6	RW	0x1	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_1_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_1_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_1_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_1_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x008	PAD_2_CTRL	31:0			
	PAD_2_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_2_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_2_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_2_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_2_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_2_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_2_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_2_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x00C	PAD_3_CTRL	31:0			
	PAD_3_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_3_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_3_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_3_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_3_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_3_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_3_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_3_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x010	PAD_4_CTRL	31:0			

Offset	Name/Field	Bits	Type	Default	Description
	PAD_4_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_4_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_4_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_4_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_4_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_4_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_4_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_4_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x014	PAD_5_CTRL	31:0			
	PAD_5_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_5_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_5_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_5_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_5_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_5_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_5_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_5_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x018	PAD_6_CTRL	31:0			
	PAD_6_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_6_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_6_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_6_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_6_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_6_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_6_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_6_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x01C	PAD_7_CTRL	31:0			
	PAD_7_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_7_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric

Offset	Name/Field	Bits	Type	Default	Description
	PAD_7_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_7_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_7_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_7_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_7_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_7_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x020	PAD_8_CTRL	31:0			
	PAD_8_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_8_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_8_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_8_P	7:6	RW	0x2	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_8_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_8_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_8_REN	11	RW	0x1	Receive enable 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
	PAD_8_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x024	PAD_9_CTRL	31:0			
	PAD_9_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_9_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_9_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_9_P	7:6	RW	0x2	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_9_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_9_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_9_REN	11	RW	0x1	Receive enable 0x1: enable 0x0: disable
	PAD_9_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x028	PAD_10_CTRL	31:0			
	PAD_10_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux

Offset	Name/Field	Bits	Type	Default	Description
	PAD_10_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_10_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_10_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_10_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_10_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_10_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_10_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x02C	PAD_11_CTRL	31:0			
	PAD_11_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_11_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_11_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_11_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_11_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_11_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_11_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_11_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x030	PAD_12_CTRL	31:0			
	PAD_12_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_12_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_12_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_12_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_12_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_12_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_12_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_12_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x034	PAD_13_CTRL	31:0			
	PAD_13_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_13_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_13_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_13_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_13_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_13_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_13_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_13_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x038	PAD_14_CTRL	31:0			
	PAD_14_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_14_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_14_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_14_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_14_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_14_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_14_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_14_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x03C	PAD_15_CTRL	31:0			
	PAD_15_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_15_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_15_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_15_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_15_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_15_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_15_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_15_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x040	PAD_16_CTRL	31:0			
	PAD_16_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_16_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_16_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_16_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_16_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_16_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_16_REN	11	RW	0x1	Receive enable 0x1: enable 0x0: disable
	PAD_16_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x044	PAD_17_CTRL	31:0			
	PAD_17_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_17_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_17_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_17_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_17_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_17_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_17_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_17_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x048	PAD_18_CTRL	31:0			
	PAD_18_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_18_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_18_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_18_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_18_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_18_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_18_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_18_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x04C	PAD_19_CTRL	31:0			
	PAD_19_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_19_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_19_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_19_P	7:6	RW	0x2	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_19_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_19_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_19_REN	11	RW	0x1	Receive enable 0x1: enable 0x0: disable
	PAD_19_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x050	PAD_20_CTRL	31:0			
	PAD_20_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_20_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_20_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_20_P	7:6	RW	0x2	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_20_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_20_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_20_REN	11	RW	0x1	Receive enable 0x1: enable 0x0: disable
	PAD_20_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x054	PAD_21_CTRL	31:0			
	PAD_21_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_21_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_21_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_21_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_21_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_21_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_21_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_21_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x058	PAD_22_CTRL	31:0			
	PAD_22_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_22_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_22_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_22_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_22_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_22_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_22_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_22_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x05C	PAD_23_CTRL	31:0			
	PAD_23_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_23_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_23_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_23_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_23_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_23_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_23_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_23_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x060	PAD_24_CTRL	31:0			
	PAD_24_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_24_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_24_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_24_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_24_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_24_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_24_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_24_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x064	PAD_25_CTRL	31:0			
	PAD_25_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_25_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_25_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_25_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_25_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_25_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_25_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_25_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x068	PAD_26_CTRL	31:0			
	PAD_26_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_26_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_26_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_26_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_26_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_26_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_26_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_26_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x06C	PAD_27_CTRL	31:0			
	PAD_27_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_27_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_27_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_27_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_27_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_27_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_27_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_27_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x070	PAD_28_CTRL	31:0			
	PAD_28_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_28_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_28_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_28_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_28_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_28_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_28_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_28_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x074	PAD_29_CTRL	31:0			
	PAD_29_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_29_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_29_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_29_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_29_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_29_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_29_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_29_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x078	PAD_30_CTRL	31:0			
	PAD_30_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_30_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_30_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_30_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_30_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_30_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_30_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_30_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x07C	PAD_31_CTRL	31:0			
	PAD_31_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_31_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_31_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation

Offset	Name/Field	Bits	Type	Default	Description
	PAD_31_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_31_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_31_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_31_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_31_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x080	PAD_32_CTRL	31:0			
	PAD_32_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_32_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_32_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_32_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_32_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA

Offset	Name/Field	Bits	Type	Default	Description
	PAD_32_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_32_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_32_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x084	PAD_33_CTRL	31:0			
	PAD_33_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_33_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_33_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_33_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_33_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_33_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_33_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_33_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Offset	Name/Field	Bits	Type	Default	Description
0x088	PAD_34_CTRL	31:0			
	PAD_34_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_34_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_34_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_34_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_34_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_34_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_34_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_34_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x08C	PAD_35_CTRL	31:0			
	PAD_35_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_35_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE

Offset	Name/Field	Bits	Type	Default	Description
	PAD_35_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_35_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_35_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_35_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_35_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_35_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x090	PAD_36_CTRL	31:0			
	PAD_36_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_36_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_36_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_36_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper

Offset	Name/Field	Bits	Type	Default	Description
	PAD_36_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_36_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_36_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_36_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x094	PAD_37_CTRL	31:0			
	PAD_37_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_37_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_37_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_37_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_37_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_37_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)

Offset	Name/Field	Bits	Type	Default	Description
	PAD_37_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_37_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x098	PAD_38_CTRL	31:0			
	PAD_38_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_38_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_38_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_38_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_38_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_38_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_38_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_38_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x09C	PAD_39_CTRL	31:0			

Offset	Name/Field	Bits	Type	Default	Description
	PAD_39_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_39_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_39_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_39_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_39_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_39_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_39_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_39_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0A0	PAD_40_CTRL	31:0			
	PAD_40_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_40_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE

Offset	Name/Field	Bits	Type	Default	Description
	PAD_40_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_40_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_40_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_40_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_40_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_40_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0A4	PAD_41_CTRL	31:0			
	PAD_41_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_41_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_41_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_41_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper

Offset	Name/Field	Bits	Type	Default	Description
	PAD_41_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_41_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_41_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_41_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0A8	PAD_42_CTRL	31:0			
	PAD_42_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_42_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_42_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_42_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_42_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_42_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)

Offset	Name/Field	Bits	Type	Default	Description
	PAD_42_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_42_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0AC	PAD_43_CTRL	31:0			
	PAD_43_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_43_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_43_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_43_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_43_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_43_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_43_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_43_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0B0	PAD_44_CTRL	31:0			

Offset	Name/Field	Bits	Type	Default	Description
	PAD_44_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_44_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE
	PAD_44_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_44_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_44_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_44_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_44_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_44_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x0B4	PAD_45_CTRL	31:0			
	PAD_45_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_45_CTRL_SEL	4:3	RW	0x1	Control selection for IO output 0x0 = A0 registers (as defined below) * 0x1 = Others 0x2 = Fabric * Exception - when FUNC_SEL=2, the control is from FFE

Offset	Name/Field	Bits	Type	Default	Description
	PAD_45_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_45_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_45_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_45_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_45_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_45_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable
0x100	SDA0_SEL	31:0			
	SDA0_SEL	0	RW	0x0	Sel 0: 1 1: pad #1
0x104	SDA1_SEL	31:0			
	SDA1_SEL	1:0	RW	0x0	Sel 0: 1 1: pad #15 2: pad #32 3: pad #44
0x108	SDA2_SEL	31:0			
	SDA2_SEL	0	RW	0x0	Sel 0: 1 1: pad #41

Offset	Name/Field	Bits	Type	Default	Description
0x10C	SCL0_SEL	31:0			
	SCL0_SEL	0	RW	0x0	Sel 0: 1 1: pad #0
0x110	SCL1_SEL	31:0			
	SCL1_SEL	1:0	RW	0x0	Sel 0: 1 1: pad #14 2: pad #33 3: pad #45
0x114	SCL2_SEL	31:0			
	SCL2_SEL	0	RW	0x0	Sel 0: 1 1: pad #40
0x118	SPIs_CLK_SEL	31:0			
	SPIs_CLK_SEL	0	RW	0x0	Sel 0: pad #16 1: 0
0x11C	SPIs_SSn_SEL	31:0			
	SPIs_SSn_SEL	0	RW	0x0	Sel 0: pad #20 1: 0
0x120	SPIs_MOSI_SEL	31:0			
	SPIs_MOSI_SEL	0	RW	0x0	Sel 0: pad #19 1: 0
0x124	SPIm_MISO_SEL	31:0			
	SPIm_MISO_SEL	0	RW	0x0	Sel 0: pad #36 1: 0
0x128	PDM_DATA_SEL	31:0			

Offset	Name/Field	Bits	Type	Default	Description
	PDM_DATA_SEL	1:0	RW	0x0	Sel 0: 0 1: pad #10 2: pad #28
0x12C	I2S_DATA_SEL	31:0			
	I2S_DATA_SEL	1:0	RW	0x0	Sel 0: 0 1: pad #10 2: pad #28
0x130	FCLK_SEL	31:0			
	FCLK_SEL	2:0	RW	0x0	Sel (Input setting for CLOCK) 7: “H” (1) 6: “L” (0) Sel (PAD# selection) 4: pad #27 3: pad #26 2: pad #10 1: pad #7 0: pad #2
0x134	UART_rxd_SEL	31:0			
	UART_rxd_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #14 2: pad #16 3: pad #25 4: pad #45
0x138	IrDA_Sirin_SEL	31:0			
	IrDA_Sirin_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #6 2: pad #15 3: pad #21 4: pad #24 5: pad #28 6: pad #40 7: pad #44

Offset	Name/Field	Bits	Type	Default	Description
0x13C	S_INTR_0_SEL	31:0			
	S_INTR_0_SEL	0	RW	0x0	Sel 0: 0 1: pad #3
0x140	S_INTR_1_SEL	31:0			
	S_INTR_1_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #2 2: pad #6 3: pad #18 4: pad #24 5: pad #35 6: pad #36
0x144	S_INTR_2_SEL	31:0			
	S_INTR_2_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #4 2: pad #8 3: pad #21 4: pad #25 5: pad #37 6: pad #38
0x148	S_INTR_3_SEL	31:0			
	S_INTR_3_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #5 2: pad #9 3: pad #22 4: pad #28 5: pad #39 6: pad #40
0x14C	S_INTR_4_SEL	31:0			

Offset	Name/Field	Bits	Type	Default	Description
	S_INTR_4_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #7 2: pad #10 3: pad #26 4: pad #29 5: pad #44
0x150	S_INTR_5_SEL	31:0			
	S_INTR_5_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #11 2: pad #14 3: pad #27 4: pad #30 5: pad #45
0x154	S_INTR_6_SEL	31:0			
	S_INTR_6_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #12 2: pad #15 3: pad #31 4: pad #32 5: pad #41
0x158	S_INTR_7_SEL	31:0			
	S_INTR_7_SEL	2:0	RW	0x0	Sel 0: 0 1: pad #13 2: pad #23 3: pad #33 4: pad #34 5: pad #42
0x15C	NUARTCTS_SEL	31:0			
	NUARTCTS_SEL	1:0	RW	0x0	Sel 0: 1 1: pad #17 2: pad #22

Offset	Name/Field	Bits	Type	Default	Description
0x160	IO_REG_SEL	31:0			
	IO_REG_SEL	7:0	RW	0x0	Selects which IO input will be registered Bit Sel Pad# 0 0 6 0 1 24 1 0 9 1 1 26 2 0 11 2 1 28 3 0 14 3 1 30 4 0 18 4 1 31 5 0 21 5 1 36 6 0 22 6 1 38 7 0 23 7 1 45
0x170	SW_CLK_SEL	31:0			
	SW_CLK_SEL	0	RW	0x0	Sel 0: pad #14 / #45 (pad selection will depend on strap #8) 1: 0
0x174	SW_IO_SEL	31:0			
	SW_IO_SEL	0	RW	0x0	Sel 0: pad #15 / #44 (pad selection will depend on strap #8) 1: 0
0x180	FBIO_SEL_1	31:0			
	FBIO_SEL_1	31:0	RW	0x0	Sel 0: 0 1: pad
0x184	FBIO_SEL_2	31:0			
	FBIO_SEL_2	13:0	RW	0x0	Sel 0: 0 1: pad

Offset	Name/Field	Bits	Type	Default	Description
0x190	SPI_SENSOR_MISO_SEL	31:0			
	SPI_SENSOR_MISO_SEL	1:0	RW	0x0	Sel 0: 0 1: pad #8 2: pad #29
0x194	SPI_SENSOR_MOSI_SEL	31:0			
	SPI_SENSOR_MOSI_SEL	1:0	RW	0x0	Sel 0: 0 1: pad #6 2: pad #28
0x1A0	I2S_WD_CLKIN_SEL	31:0			
	I2S_WD_CLKIN_SEL	0	RW	0x0	Sel 0: 0 1: pad #23
0x1A4	I2S_CLKIN_SEL	31:0			
	I2S_CLKIN_SEL	0	RW	0x0	Sel 0: 0 1: pad #31
0x1A8	PDM_STAT_IN_SEL	31:0			
	PDM_STAT_IN_SEL	1:0	RW	0x0	Sel 0: 0 1: pad #9 2: pad #30
0x1AC	PDM_CLKIN_SEL	31:0			
	PDM_CLKIN_SEL	0	RW	0x0	Sel 0: 0 1: pad #38

19.6.1 PAD_x CTRL register Description

This section describes the control actions of any of the 45 PAD control registers.

In the following, all bits are RW.

Table 19-7: PAD Control Register

Name/Field	Bits	Default	Description
PAD_x_CTRL	31:0		Control for Pad x.
PAD_x_FUNC_SEL	1:0	0x0	Functional selection for IO output Refer to IO Mux chapter
PAD_x_CTRL_SEL	4:3	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
PAD_x_OEN	5	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
PAD_0_P	7:6	0x1	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
PAD_0_E	9:8	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA This sets the drive strength for the pad. When designing, be aware that the combined drive capability of the S3 for all output pins is xxx (VALUE TO BE IDENTIFIED)
PAD_0_SR	10	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency) This field sets sensitivity to signal slew rate.
PAD_0_REN	11	0x0	Receive enable 0x1: enable 0x0: disable
PAD_0_SMT	12	0x0	Schmitt Trigger 0x1: enable 0x0: disable This field sets whether the pin handles input with a Schmitt trigger or simply is normal logic level-sensitive.

Chapter 20. Voice / Audio Processing Sub-System

Note: throughout this manual and SDK support code, the terms 'voice' and 'audio' are synonymous.

20.1 Introduction

20.1.1 General characteristics

The S3 includes features to help integrate audio in a design:

- Support for five types of Digital Microphone Interface (only one audio interface type may active at any time,)
  - PDM using internal CODEC
  - PDM using external CODEC
  - PDM using Knowles VoiceQ
  - I2S mic - direct (16/18/24 bits)
  - I2S mic - Sub-Sample
- Optional PDM bypass path allowing forwarding microphone raw data to Application Processor or a Voice CODEC
- Dedicated PDM to PCM (Pulse Code Modulation) conversion hardware
- Dedicated hard logic integration of Sensory Low Power Sound Detect (LPSD) for on-chip voice recognition
- Supports either Mono or Stereo operation; separate logic for left and right channel; channels can be enabled/disabled independently.
- 16 KHz sampling rate, 16 bit PCM data internally
- Can down-sample 32 KHz I2S input data to 16 KHz PCM data.
- Two FIFO SRAMs to buffer data, SRAM can be powered off to save power.

The voice / audio processing sub-system may be used with or without an applications processor (EOS S3 in AP Mode or in Host Mode.

The S3 supports use of Sensory TrulyHandsfree™ software. The design options are a) an OEM may license it from Sensory, or b) obtain it from reseller QuickLogic.

The built-in LPSD hardware can work with I2S Digital Mics and PDM Digital Mics. It can be used to detect voice and wake up the S3.

Mode detection module is enabled only to work with LPSD-featured PDM Digital Mic.

PDM clock output source can either be from voice IP or from AP. When clocked by AP, PDM2PCM Core, FIFOs, LPSD hardware and AHB/DMA will be powered down.

The VP DMAC is used to transfer audio data from FIFOs to S3 memory.

20.1.2 Power

The voice / audio processing sub-system hardware was designed to provide low-power operating features. Any part of the sub-system that is not operational is shut down to reduce power consumption.

Specific elements in the Voice Processing sub-system have been given their own power domains to allow fine-grain control. These are shown in the table below and listed in the Power chapter.

Power Domain	Description
AD0	Audio DMA
AD1	PDM_LEFT
AD2	PDM_RIGHT
AD3	LPSD
AD4	I2S_MASTER
AD5	Voice APB interface

20.1.3 List of operating modes

The S3 voice/audio processing sub-system offers a variety of voice processing modes:

- use PDM Internal CODEC
- use PDM External CODEC
- use PDM VoiceQ
- use I^2S Direct
- use I^2S Sub-Sample

20.1.4 Application example in system with Application Processor

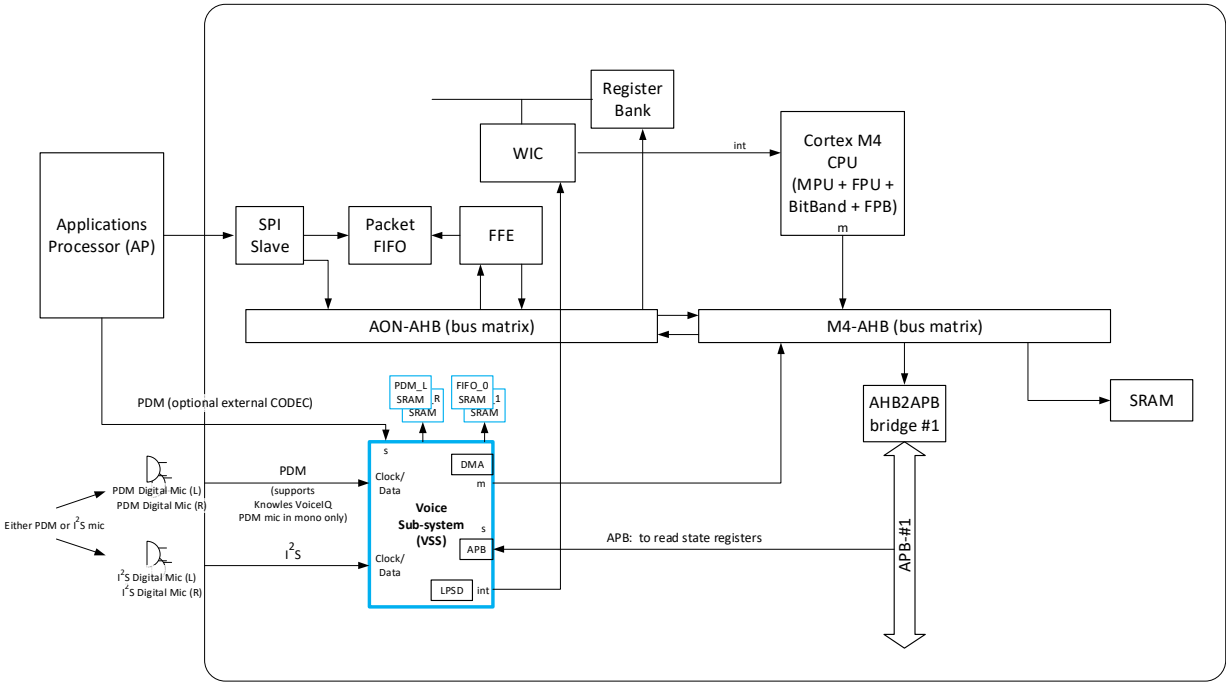

Figure 20-1: Voice / audio processing sub-system used in a design with Applications Processor

20.2 Sub-system Architecture

20.2.1 Voice / Audio Processing sub-system internal block diagram

The following is the voice / audio processing sub-system internal block diagram.

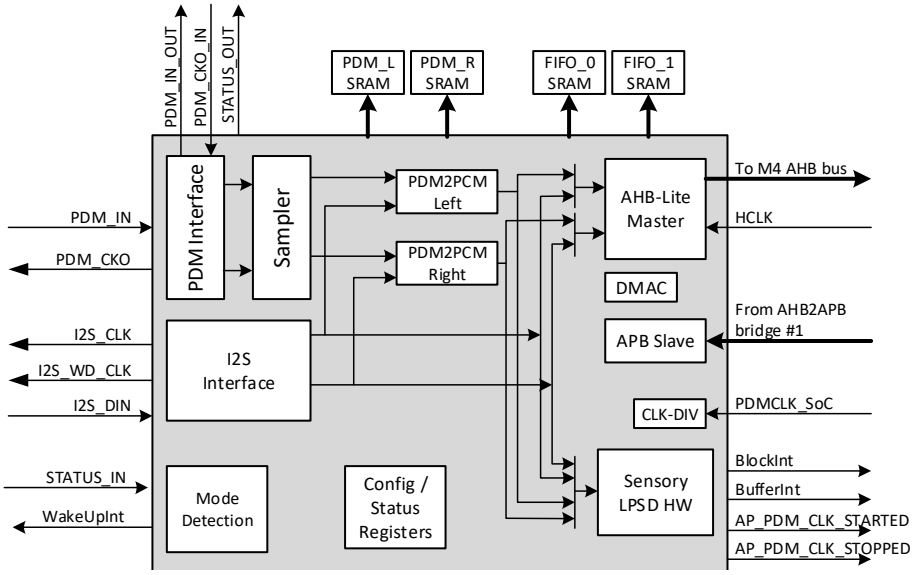

Figure 20-2: Voice / Audio Processing sub-system internal block diagram

The elements in this sub-system are:

Element	Function	Notes
PDM interface	for PDM mics	
sampler		
PDM2PCM converter left	Left channel codec converting PDM to PCM	
PDM2PCM converter right	Right channel codec converting PDM to PCM	
PDM SRAM	SRAM for storage of left and right channel PDM data	
FIFO_0 and FIFO_1 SRAM	data buffers	
I^2S interface	for I^2S mics	This element is in the A1 power domain.
mode detection block	Voice IQ mic input	Use as interrupt to controlmic function.
configuration / status registers	registers for configuration of Voice / audio processing sub-system and status	
DMAC	DMA controller to handle transferring data from this sub system to AHB bus	This element is in the A1 power domain.
AHB-lite master	Lightweight bus master for transferring voice data to the M4 AHB bus.	
APB slave	Target for AHB2APB bridge #1	
CLK-DIV	clock divider	
Sensory LPSD-hardware block for low-power sound detection	To minimize power associated with always on voice processing, the EOS S3 platform supports acoustic activity detection using LPSD hardware. When enabled, the EOS S3 platform can send PCM samples from left or right microphone to the LPSD hardware. The logic inside the LPSD hardware is designed to detect human voices and wake up the rest of the EOS S3 device.	

20.2.1.1 Pulse Density Modulation (PDM) Interface

Supports connecting a digital microphone with PDM output to this interface.

See Figure 20-2.

20.2.1.2 Inter-IC Sound (I^2S) Interface

You can connect a digital microphone with I^2S output to this interface.

See Figure 20-2.

20.2.1.3 Low-Power Sound Detect (LPSD)

To provide always on voice processing but also minimize power use, the EOS S3 supports acoustic activity detection using LPSD hardware. This allows normal PDM or I2S microphones from any vendor to get power savings associated with the LPSD hardware. When enabled, the EOS S3 platform can send PCM samples from left or right microphone to the LPSD hardware. The logic inside the LPSD hardware is designed to detect human voice and wakeup the rest of the EOS S3 device. The LPSD hardware is best used with PDM or I2S microphones and should not be enabled when using microphones that have dedicated acoustic detection capability.

The LPSD can be configured to generate an interrupt to wake up the M4 when a speech pattern is detected. This allows most of the S3 system to sleep until a voice command is detected.

20.2.2 PDM Internal CODEC Mode

Figure 20-1 illustrates the operating mode in a design which uses the PDM internal CODEC hardware. Areas of circuit not used in this mode are grayed out. These areas are powered down.

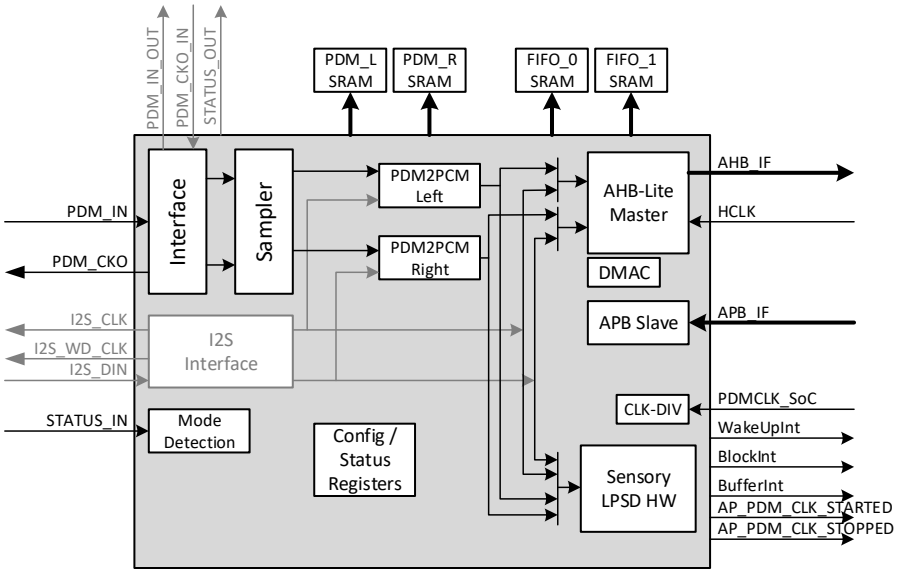

Figure 20-3: Voice / audio processing sub-system PDM Internal CODEC Mode

20.2.3 PDM External CODEC Mode

The following diagram illustrates the operating mode which uses an external PDM CODEC. Areas of circuit not used in this mode are grayed out. These areas are powered down.

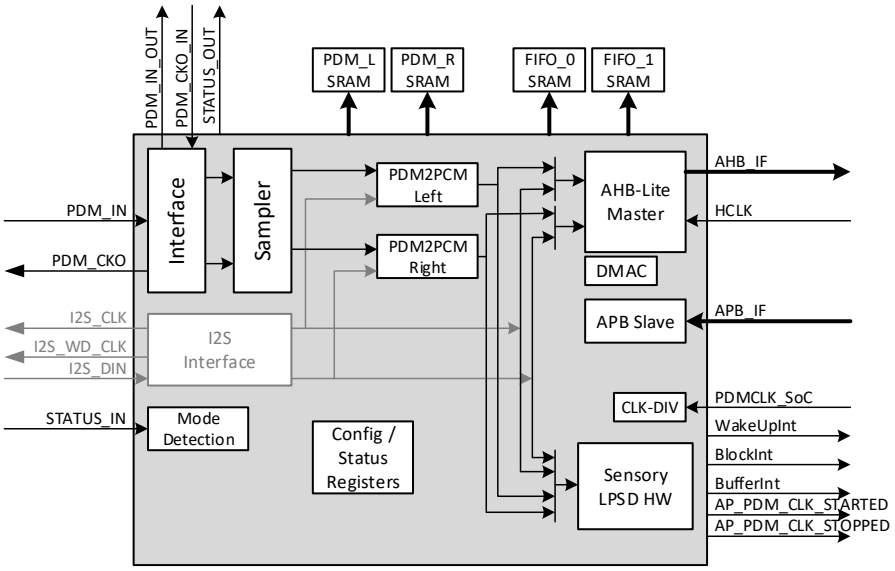

Figure 20-4: Voice / audio processing sub-system PDM External CODEC Mode

The digital microphone signal is sent to the Applications Processor for processing.

20.2.4 PDM VoiceQ Mode

The following diagram illustrates the operating mode in a design that uses PDM VoiceQ[4]. Areas of circuit not used in this mode are grayed out. These areas are powered down.

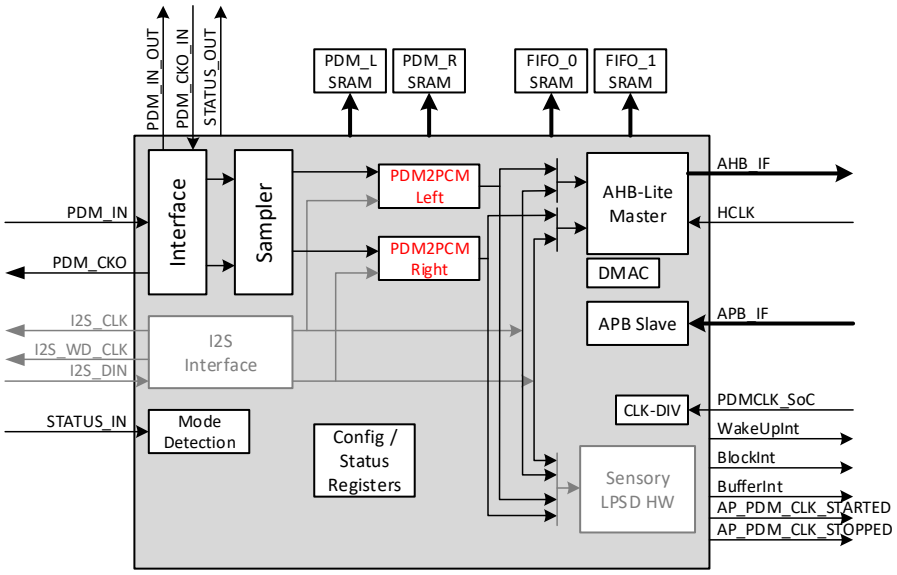

Figure 20-5: Voice / audio processing sub-system PDM VoiceQ Mode

[4] VoiceQ is a voice sensing technology by Knowles Corporation (http://www.knowles.com/)

20.2.5 I²S Direct Mode

The following diagram illustrates a design using the I²S direct mode for operation. Areas of circuit not used in this mode are grayed out. These areas are powered down.

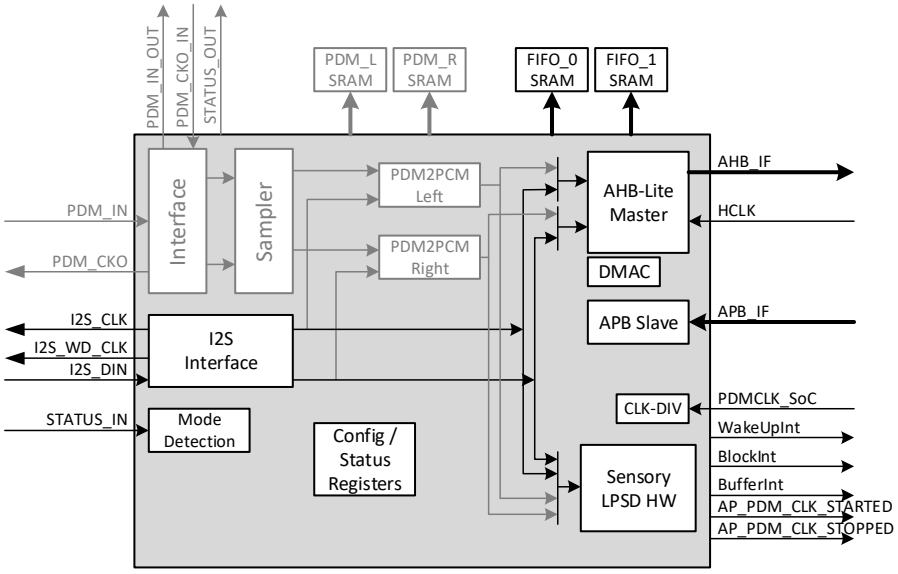

Figure 20-6: Voice / audio processing sub-system I²S Direct Mode

20.2.6 I²S Sub-Sample Mode

The following diagram illustrates the PDM I2S sub-sample operating mode. Areas of circuit not used in this mode are grayed out. These areas are powered down.

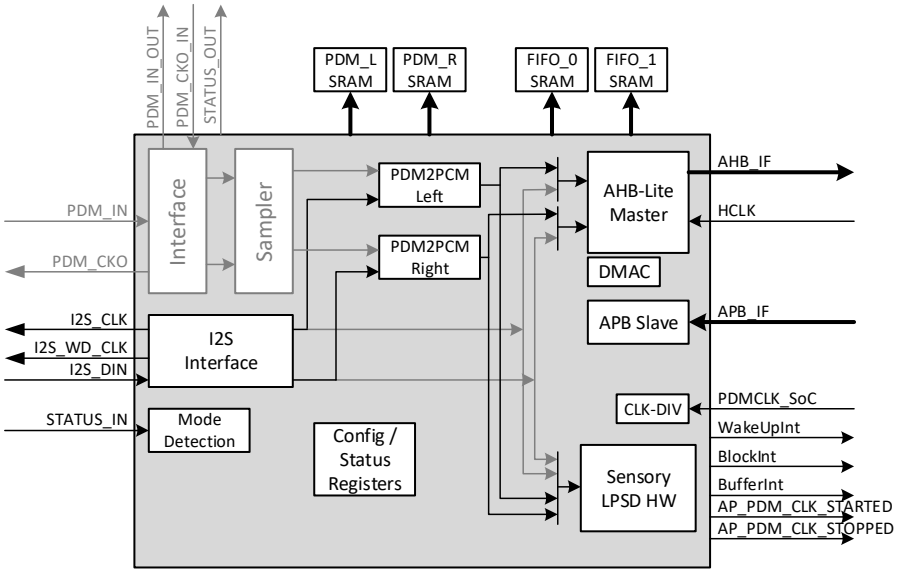

Figure 20-7: Voice / audio processing sub-system I²S Sub-Sample Mode

20.3 Voice / audio processing sub-system registers

The following is the master list of registers in this sub-system.

Offset	Name/Field	Bits	Type	Default	Description
0x000	VOICE_CONFIG	31:0			Audio system configuration register
	DMIC_SEL	0	RW	0x0	1: I2S, 0: PDM
	LPSD_SEL	1	RW	0x0	1: use external LPSD, 0: use internal sensory LPSD
	MODE_SEL	2	RW	0x0	1: stereo mode, 0: mono mode
	MONO_CHN_SEL	3	RW	0x0	1: right channel in mono mode, 0: left channel
	I2S_DS_EN	4	RW	0x0	1: enable 0: disable
	PDM_VOICE_SCENARIO	7:5	RW	0x0	000: scenario1 001: scenario2 010: scenario3
	PDM_MIC_SWITCH_TO_AP	8	RW	0x0	1: PDM DMIC switch to AP 0: no switch
	LPSD_USE_DC_BLOCK	9	RW	0x0	1: enable 0: disable
	LPSD_MUX	10	RW	0x0	1: right channel PCM data to LPSD in stereo mode 0: left channel PCM data to LPSD in stereo mode
	LPSD_NO	11	RW	0x0	0: enable HW LPSD 1: disable HW LPSD
	I2S_PGA_EN	12	RW	0x0	1: enable 0: disable
	Reserved	14:13	RO	0x0	
	DIV_AP	17:15	RW	0x2	AP_PDM_CKO_IN frequency divide-down ratio for AP clock detection
	DIV_WD	23:18	RW	0x10	AP_PDM_CKO_IN clock detection window range
	FIFO_0_CLEAR	24	RW	0x0	1: flush FIFO, 0: don't flush FIFO
	FIFO_1_CLEAR	25	RW	0x0	1: flush FIFO, 0: don't flush FIFO
	LPSD_VOICE_DETECTED_MASK	26	RW	0x0	1: disable, 0: enable
	DMIC_VOICE_DETECTED_MASK	27	RW	0x0	1: disable, 0: enable
	DMAC_BLK_DONE_MASK	28	RW	0x0	1: disable, 0: enable
	DMAC_BUF_DONE_MASK	29	RW	0x0	1: disable, 0: enable
	AP_PDM_CLK_ON_MASK	30	RW	0x0	1: disable, 0: enable
	AP_PDM_CLK_OFF_MASK	31	RW	0x0	1: disable, 0: enable
0x004	LPSD_CONFIG	31:0			LPSD config register
	LPSD_THD	15:0	RW	0x4b0	LPSD threshold parameter
	LPSD_RATIO_STOP	23:16	RW	0x58	LPSD stop ratio
	LPSD_RATIO_RUN	31:24	RW	0x4d	LPSD run ratio

Offset	Name/Field	Bits	Type	Default	Description
0x008	VOICE_DMAC_CONFIG	31:0			Audio DMAC configure register
	DMAC_EN	0	RW	0x0	1: enable, 0: disable
	DMAC_START	1	WO	0x0	1: start DMA, 0: no effect
	DMAC_STOP	2	RW	0x0	1: stop DMA, 0: no action
	AHB_RDY	3	RW	0x0	1: AHB clock ready; 0: AHB clock not ready
	AHB_BURST_LENGTH	5:4	RW	0x1	00: 1w, 01: 4w, 10: 8w, 11: 16w
	PINGPONG_MODE	6	RW	0x0	1: use pingpong buffer, 0: use single buffer
	STEREO_DUAL_BUF_MODE	7	RW	0x0	1: use separate buffers to store left/right channel data 0: use same buffer to store left/right channel data (interleaved)
	VOICE_DMAC_BURST_SPD	15:8	RW	0x0	AHB cycles between two consecutive AHB bursts
0x00C	VOICE_DMAC_LEN	31:0			Audio DMAC length register
	DMAC_BLK_LEN	15:0	RW	0x78	DMA block transfer length in words
	DMAC_BUF_LEN	31:16	RW	0x4b0	DMA buffer transfer length in words
0x010	VOICE_DMAC_FIFO	31:0			Audio DMAC Buffer offset
	RESERVED	15:0	RO	0x0	Reserved
	DMAC_BUF_OFFSET	31:16	RW	0x0	buffer offset address in dual buffer mode
0x014	VOICE_DMAC_DST_ADDR0	31:0	RW	0x0	DMA0 dest address for the first buffer
0x018	VOICE_DMAC_DST_ADDR1	31:0	RW	0x0	DMA1 dest address for the second buffer
0x01c	PDM_CORE_CONFIG	31:0			PDM2PCM core configure register
	PDM_CORE_EN	0	RW	0x1	1: enable, 0: disable
	SOFT_MUTE	1	RW	0x0	1: enable, 0: disable
	DIV_MODE	2	RW	0x0	1: use PDM_RIGHT_CLK in sampler, 0: use PDM_LEFT_CLK in sampler
	S_CYCLES	5:3	RW	0x1	set number of PDM_CLK during gain setting changes or soft mute
	HP_GAIN	9:6	RW	0xB	Adjust high pass filter coefficients
	ADCHPD	10	RW	0x1	1: disable, 0: enable high pass filter
	M_CLK_DIV	12:11	RW	0x0	PDM_CLK frequency divisor
	SINC_RATE	19:13	RW	0x18	SINC decimation rate
	PGA_L	24:20	RW	0x8	Left channel PGA gain

Offset	Name/Field	Bits	Type	Default	Description
	PGA_R	29:25	RW	0x8	Right channel PGA gain
	DMICK_DLY	30	RW	0x0	input data sampling with PDM clock cycle delay
	DIV_WD_MODE	31	RW	0x0	Status IN detection window
0x20	VOICE_STATUS	31:0			Audio Status register
	FIFO_0A_EMPTY	0	RO	0x1	1: FIFO empty, 0: not empty
	FIFO_0A_FULL	1	RW1C	0x0	1: FIFO full, 0: not full
	FIFO_0A_OVERFLOW	2	RW1C	0x0	1: FIFO overflow, 0: not overflow
	Reserved1	3	RO	0x0	
	FIFO_0B_EMPTY	4	RO	0x1	1: FIFO empty, 0: not empty
	FIFO_0B_FULL	5	RW1C	0x0	1: FIFO full, 0: not full
	FIFO_0B_OVERFLOW	6	RW1C	0x0	1: FIFO overflow, 0: not overflow
	Reserved2	7	RO	0x0	
	FIFO_1A_EMPTY	8	RO	0x1	1: FIFO empty, 0: not empty
	FIFO_1A_FULL	9	RW1C	0x0	1: FIFO full, 0: not full
	FIFO_1A_OVERFLOW	10	RW1C	0x0	1: FIFO overflow, 0: not overflow
	Reserved3	11	RO	0x0	
	FIFO_1B_EMPTY	12	RO	0x1	1: FIFO empty, 0: not empty
	FIFO_1B_FULL	13	RW1C	0x0	1: FIFO full, 0: not full
	FIFO_1B_OVERFLOW	14	RW1C	0x0	1: FIFO overflow, 0: not overflow
	Reserved3	15	RO	0x0	
	DMIC_VOICE_DETECTED_REG	16	RO	0x0	1: detected, 0: not detected
	LPSD_VOICE_DETECTED_REG	17	RO	0x0	1: detected, 0: not detected
	AP_PDM_CLK_OFF_REG	18	RO	0x1	1: off, 0: on
	AP_PDM_CLK_ON_REG	19	RO	0x0	1: on, 0: off
	DMAC1_BUF_DONE_REG	20	RW1C	0x0	1: done, 0: not done
	DMAC1_BLK_DONE_REG	21	RW1C	0x0	1: done, 0: not done
	DMAC0_BUF_DONE_REG	22	RW1C	0x0	1: done, 0: not done
	DMAC0_BLK_DONE_REG	23	RW1C	0x0	1: done, 0: not done
	Reserved5	31:24	RO	0x0	
0x24	I2S_CONFIG	31:0	RW		I2S master configure register
	I2S_LRCDIV	11:0	RW	0x40	I2S_CLK divisor for WD_CLK generator
	I2S_BCLKDIV	17:12	RW	0x2	I2S_MASTER_CLK divisor for I2S_CLK generator
	I2S_CLK_INV	18	RW	0x0	1: inverting, 0: no inverting

Offset	Name/Field	Bits	Type	Default	Description
	I2S_IWL	20:19	RW	0x0	input sample data bit shift
0x28	FIFO_SRAM_CFG	31:0	RW		FIFO SRAM configure register
	SRAM_0A_TEST1	0	RW	0x0	Test pin to bypass self-timed circuit
	SRAM_0A_RME	1	RW	0x0	Read-Write margin Enable Input
	SRAM_0A_RM	5:2	RW	0x0	Read-Write margin Input for Left Channel 8KB FIFO
	SRAM_0B_TEST1	6	RW	0x0	Test pin to bypass self-timed circuit
	SRAM_0B_RME	7	RW	0x0	Read-Write margin Enable Input
	SRAM_0B_RM	11:8	RW	0x0	Read-Write margin Input for Left Channel 512B FIFO
	SRAM_1A_TEST1	12	RW	0x0	Test pin to bypass self-timed circuit
	SRAM_1A_RME	13	RW	0x0	Read-Write margin Enable Input
	SRAM_1A_RM	17:14	RW	0x0	Read-Write margin Input for Right Channel 8KB FIFO
	SRAM_1B_TEST1	18	RW	0x0	Test pin to bypass self-timed circuit
	SRAM_1B_RME	19	RW	0x0	Read-Write margin Enable Input
	SRAM_1B_RM	23:20	RW	0x0	Read-Write margin Input for Right Channel 512B FIFO
0x2C	PDM_SRAM_CFG	31:0	RW		PDM core SRAM configure register
	PDM_SRAM_L_TEST1	0	RW	0x0	Test pin to bypass self-timed circuit
	PDM_SRAM_L_RME	1	RW	0x0	Read-Write margin Enable Input
	PDM_SRAM_L_RM	5:2	RW	0x0	Read-Write margin Input for Right Channel PDM SRAM
	PDM_SRAM_R_TEST1	6	RW	0x0	Test pin to bypass self-timed circuit
	PDM_SRAM_R_RME	7	RW	0x0	Read-Write margin Enable Input
	PDM_SRAM_R_RM	11:8	RW	0x0	Read-Write margin Input for Left Channel PDM SRAM
0x30	DBG_MUX_CFG	31:0	RW		Audio Debug register

Offset	Name/Field	Bits	Type	Default	Description
	DBG_MUX	3:0	RW	0x0	0000 => dbg_tp[7:0] = dbg_fifo_0a_wptr[7:0] 0001 => dbg_tp[7:0] = dbg_fifo_0a_rptr[7:0] 0010 => dbg_tp[7:0] = {1'b0, dbg_fifo_0a_rptr[10:8], 1'b0, dbg_fifo_0a_wptr[10:8]} 0011=> dbg_tp[7:0] = dbg_fifo_1a_wptr[7:0] 0100 => dbg_tp[7:0] = dbg_fifo_1a_rptr[7:0] 0101 => dbg_tp[7:0] = {1'b0, dbg_fifo_1a_rptr[10:8], 1'b0, dbg_fifo_1a_wptr[10:8]} 0110 => dbg_tp[7:0] = {2'd0, dbg_fifo_0b_wptr[5:0]} 0111 => dbg_tp[7:0] = {2'd0, dbg_fifo_0b_rptr[5:0] } 1000 => dbg_tp[7:0] = {2'd0, dbg_fifo_1b_wptr[5:0]} 1001 => dbg_tp[7:0] = {2'd0, dbg_fifo_1b_rptr[5:0] } 1010 => dbg_tp[7:0] = {2'd0, dbg_afifo_0_rptr[1:0], 2'd0, dbg_afifo_0_wptr[1:0]} 1011 => dbg_tp[7:0] = {2'd0, dbg_afifo_1_rptr[1:0], 2'd0, dbg_afifo_1_wptr[1:0]} 1100 => dbg_tp[7:0] = {PCM_DATA_L[7:1],VALID} 1101 => dbg_tp[7:0] = {PCM_DATA_L[15:9],VALID} 1110 => dbg_tp[7:0] = {PCM_DATA_R[7:1],VALID} 1111 => dbg_tp[7:0] = {PCM_DATA_R[15:9],VALID}

Chapter 21. Sensor Processing Hub Sub-system

21.1 Introduction

The S3 includes the versatile Sensor Processing sub-system (also known as the Sensor Hub) that performs a variety of sensor fusion activities while supporting low power usage.

The ultra-low power Flexible Fusion Engine (FFE) in this sub-system is a processor that allows using power management together with real-time sensor fusion algorithms. Operating independently of the M4 or an AP which can be asleep, the FFE too can sleep until a sensor or timer event causes the Power Management Unit to awaken it.

21.2 Sensor Processing Hub Sub-System Architecture

21.2.1 Block diagram

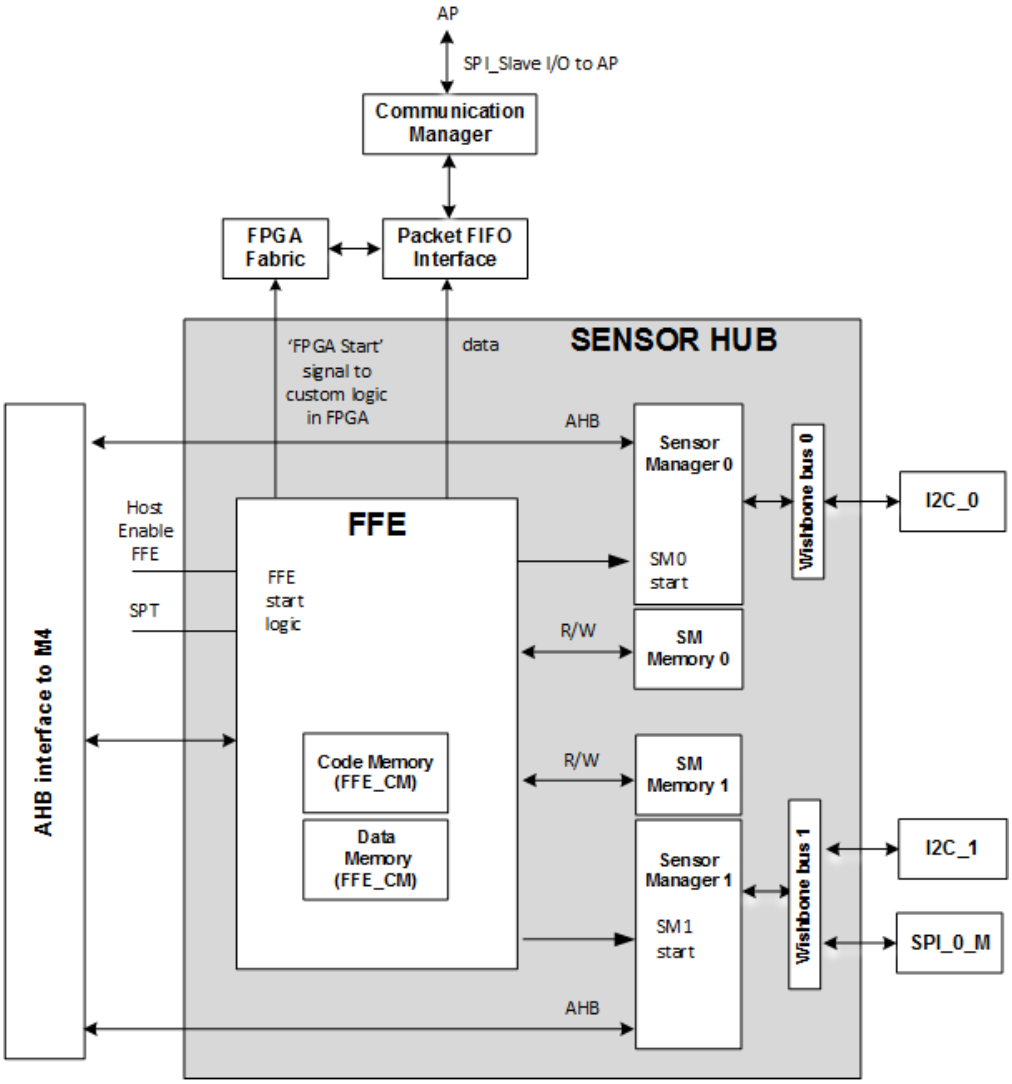

Figure 21-1: Sensor Processing Hub sub-system's elements

21.2.2 Key elements in the Sensor Processing Hub sub-system

- **Flexible Fusion Engine (FFE)**
  This module is responsible for coordinating the retrieval of sensor data by each Sensor Manager and using this data for sensor fusion operations. It contains memory available for processing operations.
- **Sensor Manager 0 and 1**
  The Sensor Managers control their attached sensors, retrieve data, and put data in the sensor memory where it can be accessed by the FFE.
- **Sensor Processor I^2C Interfaces**
  The Sensor Processing sub system has two I^2C Master interfaces for connection to I^2C-enabled sensors.
- **Sensor Processor Serial Peripheral Interface Master (SPI_0_Master)**
  This interface allows the Sensor Processor to be the master for SPI-enabled sensor devices.
- **AHB master bridge for the Sensor Processor**
  The AHB Master Bridge allows this sub system to use S3 system resources such as memory and DMA.
- **Wishbone Bus (WB)**
  WB provides flexible connections from Sensor Managers and the I^2C and SPI ports for this sub-system.

Major elements are described further in following sections.

21.2.2.1 AHB master bridge for the Sensor Processing Hub

The AHB Master Bridge for the Sensor Processing Hub allows this sub system to use S3 resources such as memory and DMA. Users do not need to specifically program bridge operation in order to achieve access to resources. The following is shown merely for general orientation.

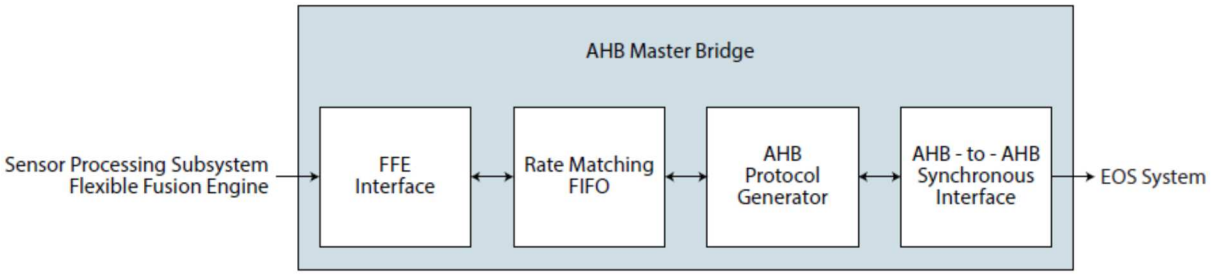

Figure 21-2: AHB Master Bridge for Sensor Processing Hub - Block Diagram

The main purpose of the AHB Master Bridge is to enable the FFE to do the following operations:

- Update the FFE memories, using the M4-F SDMA controller. This requires the M4-F to configure the SDMA controller in advance of the FFE DMA request.
- Use sections of the M4-F memories as a large FIFO. This enables storage of more processed sensor data than is available via other hardware paths.

21.2.3 Related system components: Packet FIFO

The Packet FIFO interface enables the FFE to pass sensor data in the form of packets to the EOS S3 platform. These packets can contain either data resulting from Sensor Fusion processing or unprocessed sensor data. The format and content of each packet is determined by the algorithm running on the FFE.

21.3 General characteristics

21.3.1 Key functional characteristics

Major functionality that the user can control includes:

- Wishbone Bus access to multiple I2C and SPI Interfaces
- Access to the FFE and Sensor Manager (SM) memories
- Debug resources for both the FFE and SM
- Execution control and status for both the FFE and SM
- Interrupt resources for the Sensor Processing Subsystem

FFE

- Up to 10 MHz operating frequency
- 50 KB control memory (code memory)
- 16 KB data memory
- µDSP-like architecture for efficient mathematical computations
- Ideal for always-on, real-time sensor fusion algorithms (such as pedometer, activity classification, gesture recognition, and others)

Sensor Managers

- Dedicated Microcontroller Unit (MCU)
- 1.5 KB x 18-bit memory for command and data memory and control (Mailbox) bits
- Completely autonomous (zero load on M4-F) initialization and sampling of sensors through hard-wire I2C or configurable I2C/SPI interface
- Dramatically lowers the power consumption associated with sensor data acquisition
- Time Stamping
    - Automatic hardware time stamp on every sensor read in the interrupt mode
    - Up to eight sensor interrupt captured time-stamps (8-bit)
    - Main time stamp of 30 bits for M4-F processor and 24 bits for FFE
    - Resolution of 1 msec

I^2C interfaces

The following features are available in the I^2C interfaces:

- Compatible with Philips I^2C standard
- Multi Master operation
- Software programmable clock frequency
- Clock Stretching and Wait state generation
- Software programmable acknowledge bit
- Interrupt or bit-polling driven byte-by-byte data-transfers
- Arbitration lost interrupt, with automatic transfer cancelation
- Start/Stop/Repeated Start/Acknowledge generation
- Start/Stop/Repeated Start detection
- Bus busy detection
- Supports 7 and 10 bit addressing mode; the interface supports 10 bit slave addresses by generating two address transfers (refer to Phillips I^2C specifications for details)
- Operates from a wide range of input clock frequencies

SPI_0_Master

The main purpose of the SPI_0_Master is to support connection to SPI-based sensors.

21.3.2 Power

This sub-system is in the FFE power domain.

Chapter 22. Sensor Processing Hub Theory of Operation

During EOS S3 system non-debug setup operations in Hosted designs, the Host system uses the Configuration Manager's SPI Slave module to write a pre-compiled binary file to the S3 device's M4 memory. In addition, the Host also writes pre-compiled binary files to other memories such as those used by the FFE and the Sensor Manager. Once that is complete, the Host enables the S3 M4 processor to execute the newly written binary code. At that time, the Sensor Hub also can be enabled to do its intended operations.

22.1 Control and Flow

22.1.1 Operating flow

Programming SensorHub subsystem involves:

- enable fast clock to the Hub elements
- write the FFE code (algorithm) into the FFE Code Memory
- write Sensor Manager code into the code section of the SM Memory
- write constants and variables with specific initial values into the FFE Data Memory
- enable slow clock to the Hub
- enable the FFE to run
- Host configures the SPT and enables this timer

On the next time tick, the Sensor Hub will then run.

Thereafter, a typical SM operating cycle is.

- Timer wakes the FFE
- The FFE wakes the SM.
- SM reads sensor
- SM writes data to SM. sets semaphore, then sleeps
- FFE reads semaphore, reads data from SM memory and batches it, clears the memory, then goes to sleep

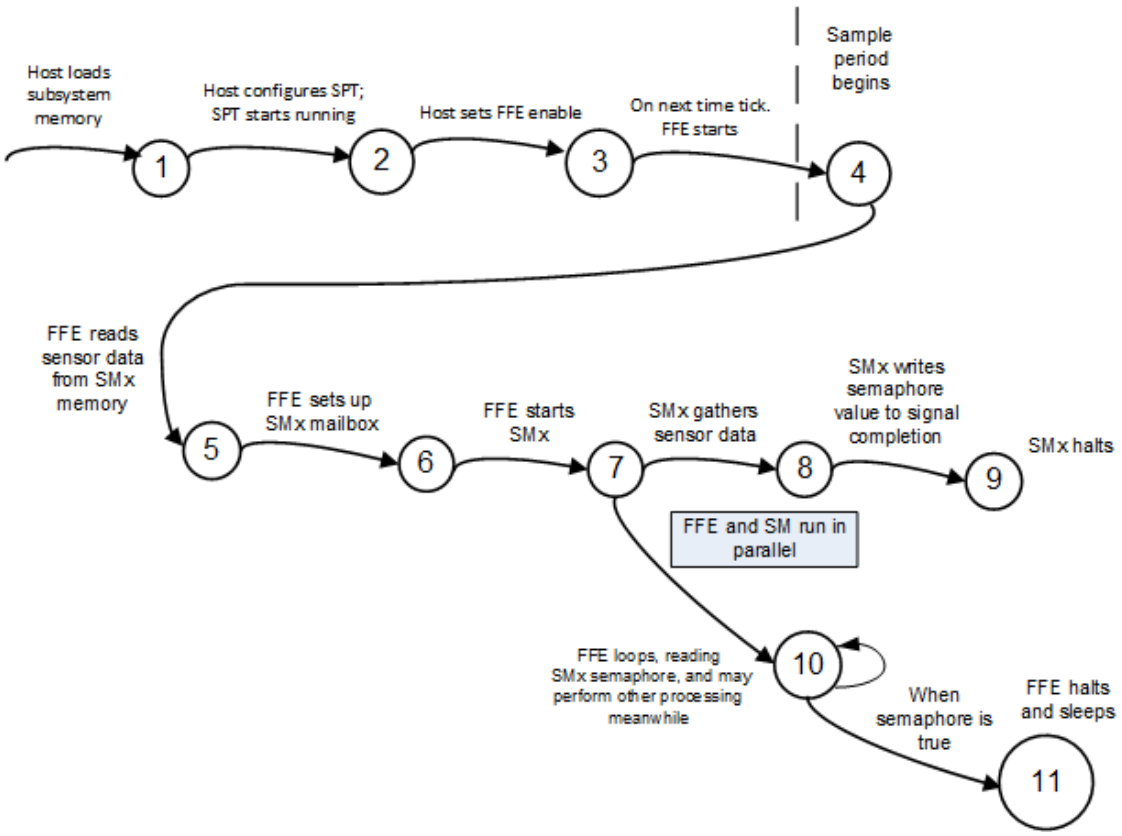

Figure 22-1: General flow of states in the Sensor Hub

The symbol '->' means 'the following action transitions to this numbered state"

-> 1 Host loads FFE CM and DM, and loads SM code area with session code

-> 2 Host configures SPT; SPT starts running, generating time ticks

-> 3 Host sets FFE enable; this allows FFE to start on the next SPT time tick

-> 4 FFE starts when ((FFE enable) && (SPT time tick output)) == true,

Sample period now begins

-> 5 FFE reads sensor data from SMx memory, then saves it to a designated memory, then clears the SMx memory

-> 6 FFE sets up SMx mailbox bits governing session code use by SM

-> 7 FFE starts SMx to run its session

FFE and SMx are now running at the same time.

-> 8 The selected SM runs its code segments, as determined by the controlling Mailbox bits; the SM gathers sensor data and writes it to SMx data memory area

-> 9 After SMx reaches its final code segment, it writes a semaphore value to SMx memory to inform FFE of completion status, then halts

7 -> 10 FFE loops, reading semaphore value until semaphore == true. While it loops, the FFE may be performing other processing tasks too.

-> 11 When semaphore == true is detected, FFE halts and goes to sleep

22.1.2 SM Mailboxes

As shown in Section 24.1.1.1, each Sensor Manager's memory is partitioned into three areas:

1. Mailbox bits. Each of these bits determines whether the Sensor Manager runs the code segment corresponding to that Mailbox bit. The FFE controls the value of these bits and thus controls which code segments get run. The Host sets the value in each code segment and thus determines what code operations can be performed.

2. Data region. This area of SM memory is where the SM writes sensor data.

3. Code region. This area of SM memory is where the Host writes code segment values/

22.2 Sampling and Timing

22.2.1 Time Stamping

- Automatic hardware time stamp on every sensor read in the interrupt mode
- Up to eight sensor interrupt captured time-stamps (8-bit)
- Main time stamp of 30 bits for M4-F processor and 24 bits for FFE
- Resolution of 1 msec

Chapter 23. Flexible Fusion Engine (FFE)

This module is responsible for coordinating the retrieval of sensor data by each Sensor Manager and using this data for sensor fusion operations.

The FFE is responsible for the following:

- Coordinating the operation of the Sensor Manager(s)
- Retrieval of sensor data retrieved by the Sensor Manager(s)
- Sensor fusion calculations
- Transferring the results of the sensor fusion calculations to the EOS S3 platform
- Coordinating FFE operations with on-chip programmable logic IP

The general operating flow for the FFE is:

1. To reduce power usage, the Sensor Processor sub-system sleeps until needed.

2. Then, typically when triggered by a timer such as the SPT, the FFE is awakened and it either:

a) Reads data from the Sensor Memory, or
b) Rhe FFE awakens long enough to trigger a Sensor Manager to awaken and get data to put in the Sensor Memory, or
c) The FFE awakens and performs a sensor processing algorithm.

23.1 Architecture of the FFE

The FFE consists of three basic blocks:

- µDSP-Like Processor
- Instruction Memory (also known as Code Memory or CM in the register descriptions)
- Data Memory (also known as DM in the register descriptions)

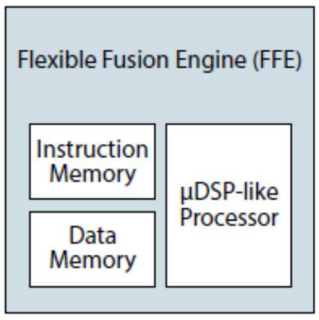

Figure 23-1: FFE block diagram

23.1.1 µDSP general functions

The µDSP provides the main operation of the FFE. The µDSP retrieves instructions from the Instruction Memory along with data values stored in the Data Memory. In addition the µDSP performs the following operations:

- Waiting for a Start signal from the EOS S3 platform to begin processing
- Receiving Mailbox values from the EOS S3 platform to direct FFE processing
- Writing Mailbox values to the Sensor Manager Memory — The Mailbox values determine which sensors will be contacted by the Sensor Manager during each sampling period.
- Reading sensor data values from the Sensor Manager Memory prior to starting a new Sensor Manager session.
- Starting each Sensor Manager session to retrieve a new set of sensor data
- In parallel with the Sensor Manager session, performing Sensor Fusion calculations based on the sensor data values retrieved from Sensor Manager Memory.
- Sending the results of the Sensor Fusion calculations to the EOS S3 platform — The FFE can use either the Packet FIFO interface or the AHB Master port to pass packets of sensor data to the EOS S3 platform
- Coordinating FFE operations with on-chip programmable logic-based IP. This IP waits for a Start signal from the EOS S3 system prior to beginning processing.

QuickLogic provides a variety of software support, in the form of APIs, and programming guides for the FFE. Documentation for these is separate from this hardware manual. However, this manual includes register information.

23.1.2 Instruction Memory

The Instruction Memory contains the instructions used by the µDSP for performing the Sensor Fusion operations. The EOS S3 platform loads this memory prior to the first FFE operation. At the start of each FFE processing session, µDSP reads instructions from this memory starting at address 0 and continuing until a Stop instruction is read.

The EOS S3 platform may elect to alter the Sensor Fusion operation on a session-by-session basis by passing Mailbox data from the Control Registers module. If the EOS S3 platform does this, it does not need to modify the Instruction or Data Memories to support multiple Sensor Fusion processing modes. It is important to note that the Harvard architecture FFE cannot write to its own Instruction Memory, and no other module within the Sensor Processing Subsystem can read or write to the Instruction Memory. FFE code must be loaded by action by the M4.

23.1.3 Data Memory

The Data Memory contains the data portion of the µDSP program execution. The EOS S3 platform loads this memory prior to the beginning of the first FFE session. The Instruction Memory determines which portions of Data Memory the Microcontroller reads or writes thereafter. Values stored in Data Memory can include:

- Constant values
- Variable values
- Sensor data values

It is important to note that both the EOS S3 platform and FFE can write to this memory. However, this does not extend to any other module in the Sensor Processing Subsystem.

23.2 Theory of operation for the FFE

In practice, FFE µDSP programs are compiled using the FFEAT application. See FFEAT User Guide for details.

23.2.1 Power control

After the Wakeup Interrupt Controller is triggered by sensor or timer events, the WIC triggers the PMU to awaken the FFE if it is asleep. See discussion in Sensor Processing (FFE) Sub-System Low Power Modes.

23.2.2 Mailboxes

There are several registers implemented as mailboxes for information exchange between M4/Host and FFE.

23.2.3 Data handling

See QL Application Note 127 EOS S3 Sensor Processing Platform Software Architecture Overview.pdf

Chapter 24. Sensor Managers

The Sensor Processing sub-system has two Sensor Managers. These modules handle managing the external sensors, sensor data transfers, and certain S3 internal inter-module data transfers.

24.1 Sensor Manager Internal Architecture

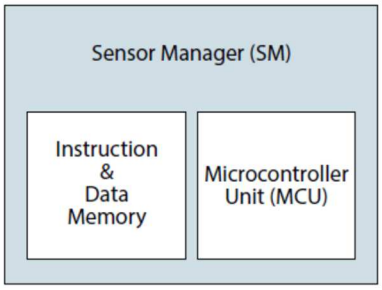

Figure 24-1: Sensor Manager block diagram

The architecture of each Sensor Manager consists of these parts:

- Microcontroller Unit - responsible for the operation of the Sensor Manager.
- Sensor Manager Memory block, partitioned into:
  - Instruction Memory -- segments loaded by the M4 or the AP hold sensor management routines
  - Data Memory for sensor data

Additionally, each Sensor Manager connects to external interface blocks via a Wishbone bus as shown in Figure 21-1, Sensor Processing Hub sub-system. Sensor Manager 0 can operate with I^2C port 0 via Wishbone Bus 0. Sensor Manager 1 can operate with I^2C port 1 and SPI_0_Master via Wishbone Bus 1.

24.1.1 Sensor Manager Memory

Each Sensor Manager has a dedicated block of memory that it and the FFE can access. The block is used to hold code, data, and Mailbox bits that control code operation.

24.1.1.1 Structure of Sensor Manager memory

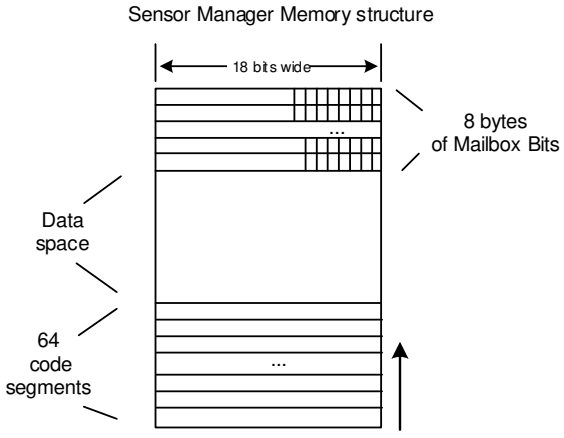

Figure 24-2: Sensor Manager Memory Structure

24.2 Related system elements

24.2.1 Wishbone Bus

The Sensor Processor Hub sub-system includes two Wishbone buses. Wishbone Bus 0 connects Sensor Manager 0 with I2C Master 0. Wishbone Bus 1 connects Sensor Manager 1 with I2C Master interface 1 and SPI_0_Master interface. Before a Sensor Manager can connect to an I/O interface, the relevant Wishbone bus must be programmatically selected using an FFE register for this.

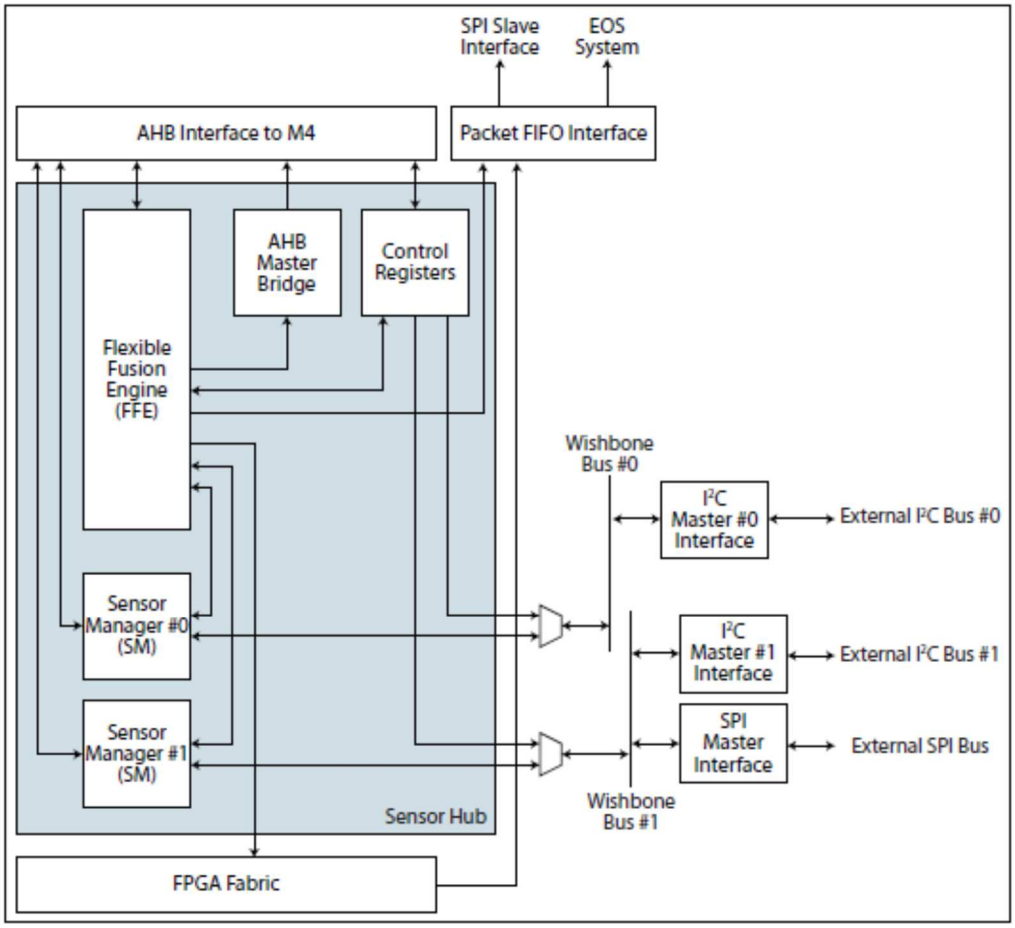

Figure 24-3: SensorHub subsystem with Wishbone buses

To perform an operation with the one of the Sensor Hub IO devices, a Sensor Manager must first select the Wishbone bus through which the device is reached. This is done via the FFE WB_ADDR register at 0x000.

24.2.2 I2C interfaces

The I2C interfaces are outside the Sensor Manager blocks but are highly integrated into their operation. Note that the I2C interfaces can be accessed by other system elements such as the M4 CPU but normally there is no need to, since the purpose of having the Sensor Managers is to offload sensor access from the CPU and thereby reduce power consumption.

The following block diagram shows the internals of the two I2C interfaces.

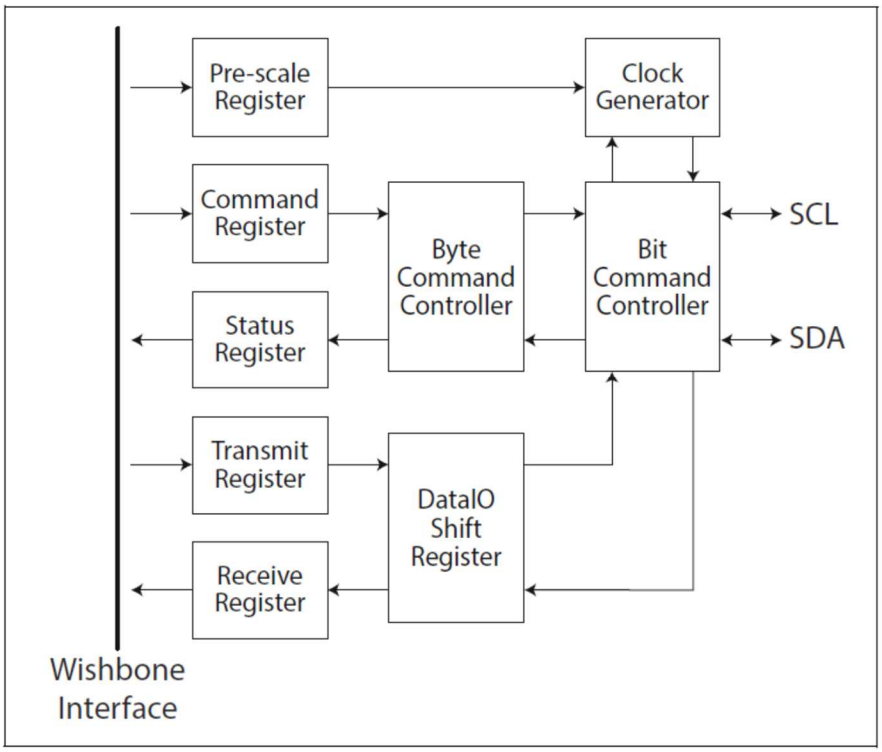

Figure 24-4: I²C interface internals

24.2.2.1 I²C Master 0

I²C Master 0 is controlled by Sensor Manager 0 and communications are over the Wishbone Bus 0. See Figure 21-1.

See the FFE register descriptions section dfor details on the registers.

See GPUIO section on IOMUX and GPIO control.

See the Sensor Manager Programming Manual for details on creating Sensor Manager programs.

Power domain: FFE

24.2.2.2 I²C Master 1

I²C Master 1 is controlled by Sensor Manager 1 and communications are over the Wishbone Bus 1. See Figure 21-1.

See the FFE register descriptions section dfor details on the registers.

See GPUIO section on IOMUX and GPIO control.

See the Sensor Manager Programming Manual for details on creating Sensor Manager programs.

Power domain: FFE

24.2.3 SPI_0_Master

Sensor Manager 1 can access the SPI_0_Master interface (but Sensor Manager 0 cannot). This interface is discussed in detail in a later chapter.

Chapter 25. SPI_0_Master

This is one of two SPI Master in the system. Its registers are in the group for this sub-system.

SPI_0_Master is controlled by Sensor Manager 1 and communications are over the Wishbone Bus 1. See Figure 21-1.

For control register descriptions, see the FFE register descriptions section in this chapter.

For information on assigning signals for this interface to I/O pads, and signal control, see section on IOMUX and GPIO control.

Power domain: FFE

25.1 Architecture and Operation

The following diagram shows the internal S3 connections to this interface (from Wishbone Bus 1) and the external connections leading to the IOMUX multiplexer. By means of the IOMUX, users can program mappings of the external connections to I/O pads on the S3. For details see the discussions in the IOMUX section of the HRM.

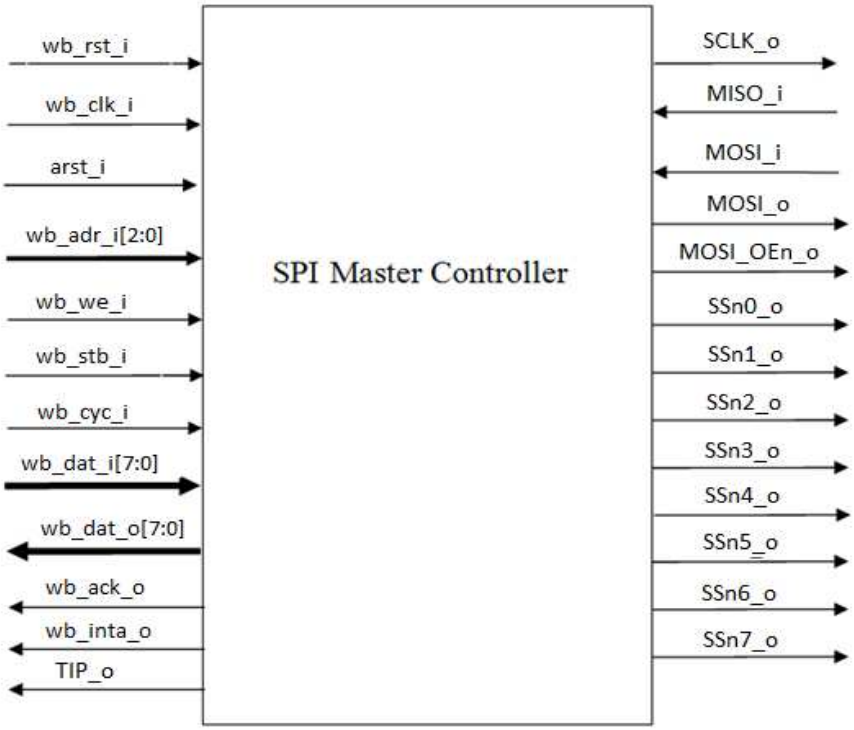

Figure 25-1: SPI_0_Master block connections

25.2 I/O signals for the SPI_0_Master block

Signal	I/O	Description
SPI Interface		
SCLK_o	O	SPI CLOCK
MOSI_o	O	Master Out Slave In: Output from the master is connected to slave
MOSI_OEn_o	O	Output Enable, active low
MOSI_i	I	Input to Master, in 3-wire SPI configuration mode
MISO_i	I	Master In Slave Out: Output from the slave is connected to master
SSn0_o	O	Slave Select 0: Active low output from master to select slave
SSn1_o	O	Slave Select 1: Active low output from master to select slave
SSn2_o	O	Slave Select 2: Active low output from master to select slave
SSn3_o	O	Slave Select 3: Active low output from master to select slave
SSn4_o	O	Slave Select 4: Active low output from master to select slave
SSn5_o	O	Slave Select 5: Active low output from master to select slave
SSn6_o	O	Slave Select 6: Active low output from master to select slave
SSn7_o	O	Slave Select 7: Active low output from master to select slave
Wishbone Interface		
wb_clk_i	I	Master Clock
wb_rst_i	I	Synchronous Reset, Active High
arst_i	I	Asynchronous Reset, Active High
wb_adr_i [2:0]	I	Lower address bits
wb_dat_i [7:0]	I	Input data bus
wb_dat_o [7:0]	O	Output data bus
wb_we_i	I	Write Enable
wb_stb_i	I	Strobe signal/Core select input
wb_cyc_i	I	Valid bus cycle input
wb_ack_o	O	Bus cycle acknowledge output
wb_inta_o	O	Interrupt signal output
tip_o	O	Transfer In Progress output 1: SPI Transaction in progress 0: SPI Transaction complete

25.3 Master / slave clocking

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device, slave devices that are not selected do not interfere with SPI bus activities. The diagram below illustrates the expected connection.

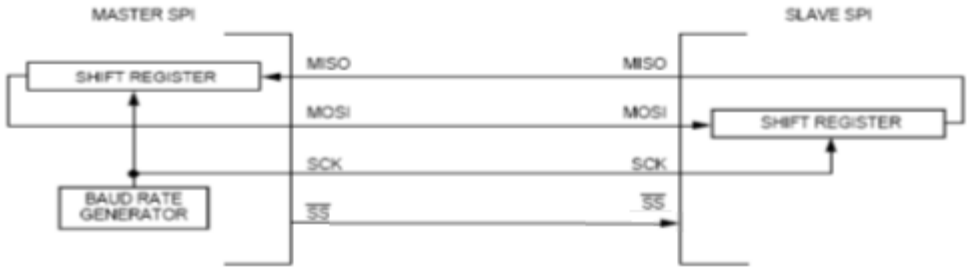

Figure 25-2: SCK between master and slave

25.4 SPI Transactions

25.4.1 SPI Write Cycle

Basic SPI write operation with mode 11 (CPOL = 1, CPHA = 1)

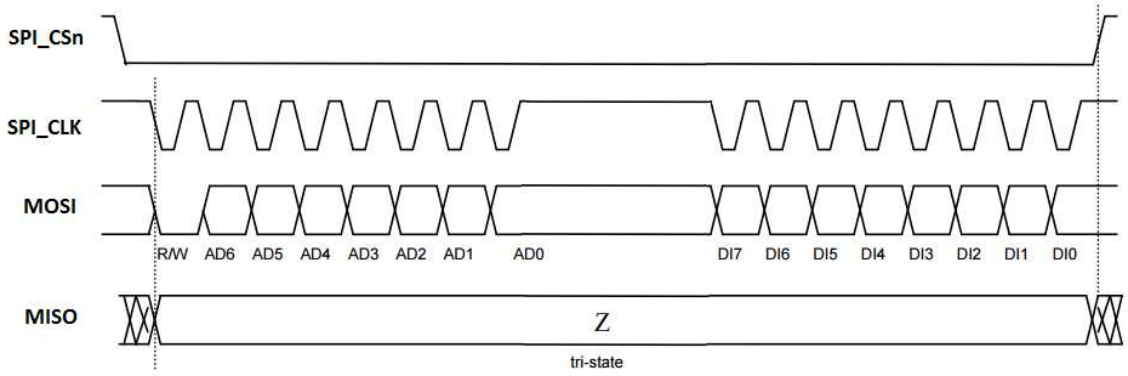

Figure 25-3: Basic SPI Write Operation (mode 11)

25.4.2 SPI Read Cycle

Basic SPI read operation with mode 11 (CPOL = 1, CPHA = 1)

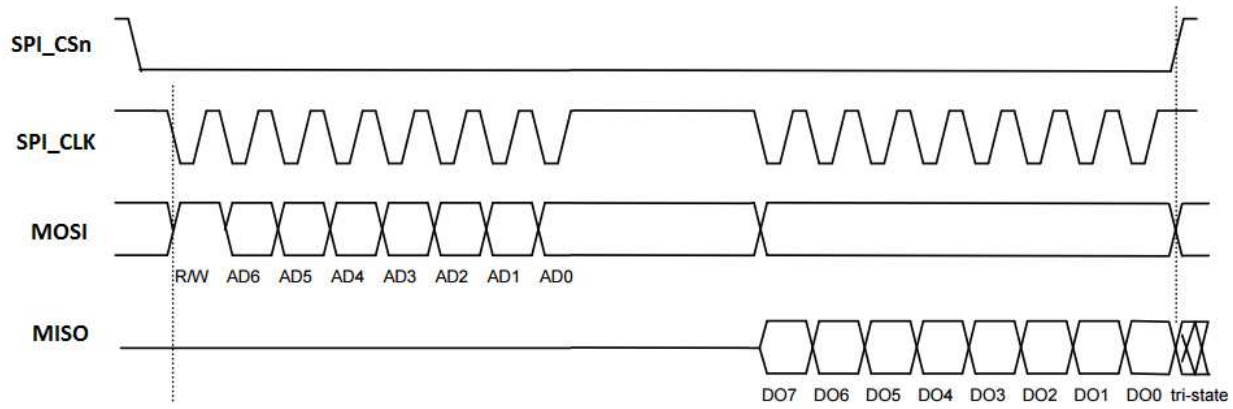

Figure 25-4: Basic SPI Read Operation (mode 11)

25.4.3 SPI Multiple Read Cycle

SPI Multiple read operation with mode 11 (CPOL = 1, CPHA = 1).

Multiple read operations are possible by keeping the SPI_CSn low and continuing the data transfer. Only the first register address needs to be written. Addresses are automatically incremented after each read as long as the SPI_CSn line is held low.

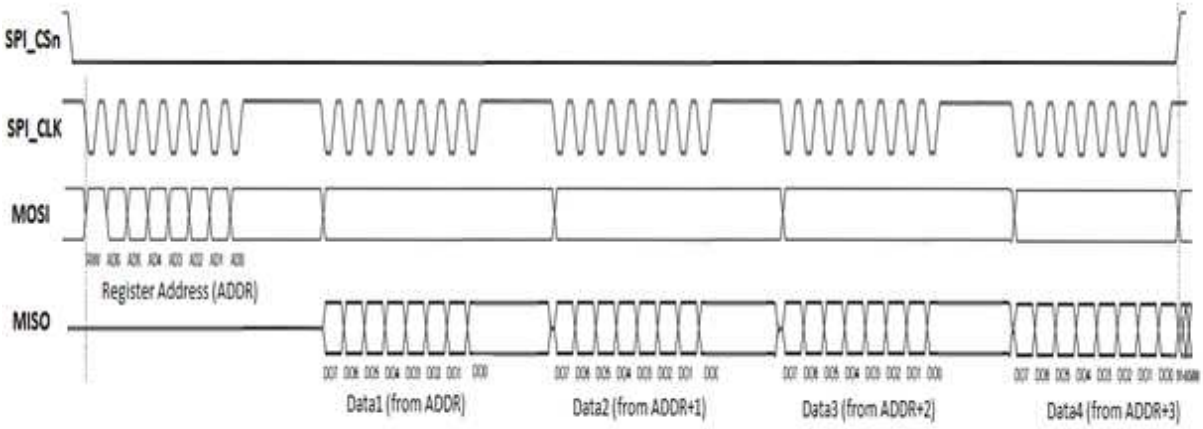

Figure 25-5: SPI Multiple Read Operation (mode 11)

25.4.5 SPI 3 wire configuration

This is the basic SPI read/write operation with mode 11 (CPOL = 1, CPHA = 1) in 3 wire configuration:

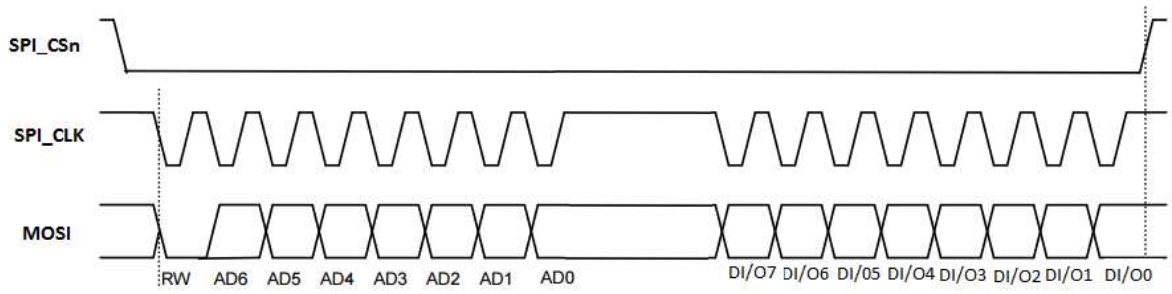

Figure 25-6: 3-wire basic SPI read / write sequence (mode 11)

25.4.6 SPI corner cases

To accommodate certain sensors that lack 8-bit data alignment, the SPI_0_Master interface includes these capabilities

- Capability to shift data based on 1 to 8 shift clock cycles.
- Chip Select signal is kept low between SPI transfer cycles.
- The SPI shift clock can run for a preset number of cycles after Chip Select has been de-asserted. Some devices require additional shift clock cycles after Chip Select is removed in order to achieve, for example, low power states or to complete the requested operation.

25.4.7 Clock Phase and Polarity Controls

Using two bits in the SPI Control Register1, software selects one of four combinations of serial clock phase and polarity.

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

Clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

25.4.7.1 Transfer Format for CPHA = 0

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after SS has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the shift register, depending on LSBFE bit.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer and is transferred to the parallel SPI Data Register after the last bit is shifted in.

After the 16th (last) SCK edge:

- Data that was previously in the master SPI Data Register should now be in the slave data register and the data that was in the slave data register should be in the master.
- The SPIF flag in the SPI Status Register is set indicating that the transfer is complete.

Figure 25-7 is a timing diagram of a SPI transfer where CPHA = 0. SCK waveforms are shown for CPOL = 0 and CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave and the MOSI signal is the output from the master. The SS pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.

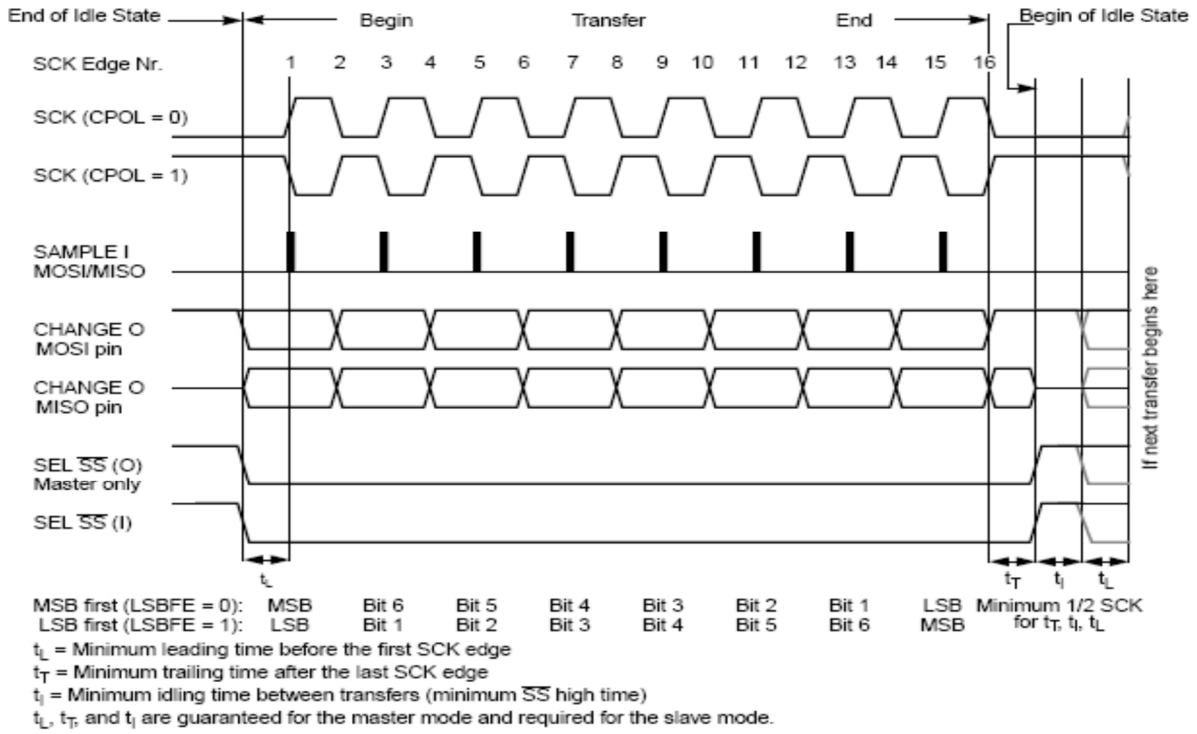

Figure 25-7: SPI Clock Format 0 (CPHA = 0)

25.4.7.2 Transfer Format for CPHA = 1

Some peripherals require the first SCK edge before the first data bit becomes available at the data out pin, the second edge clocks data into the system. In this format, the first SCK edge is issued by setting the CPHA bit at the beginning of the 8-cycle transfer operation.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the LSB or MSB of the SPI shift register, depending on LSBFE bit. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

This process continues for a total of 16 edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

Data reception is double buffered; data is serially shifted into the SPI shift register during the transfer and is transferred to the parallel SPI Data Register after the last bit is shifted in.

After the 16th SCK edge:

- Data that was previously in the SPI Data Register of the master is now in the data register of the slave, and data that was in the data register of the slave is in the master.
- The SPIF flag bit in SPISR is set indicating that the transfer is complete.

Figure 25-8 shows two clocking variations for CPHA = 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The SS line is the slave select input to the slave. The SS pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.

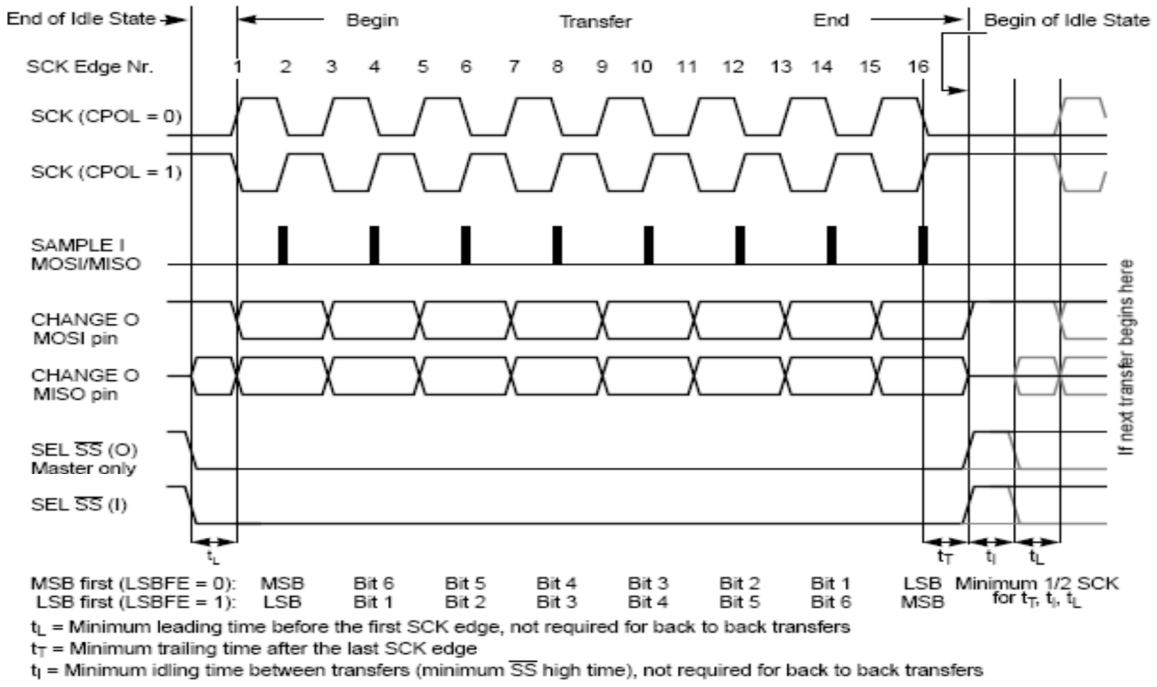

Figure 25-9: SPI Clock Format 1 (CPHA = 1)

25.5 SPI_0_Master Registers

25.5.1 Address Map

The table below shows the memory map of the SPI register set. There are eight registers that control the SPI operation.

The address listed for each register is the offset from the base address of the chip select of the processor.

Table 25-1: SPI_0_Master register address table

Address	Access	Reset Value	Description
0x00	RW	0x01	SPI Baud Register, Divisor LSB
0x01	RW	0x00	SPI Baud Register, Divisor MSB
0x02	RW	0x00	SPI Configuration Register
0x03	W	0x00	Transmit Register
0x03	R	0x00	Receive Register
0x04	W	0x00	Command Register (SPI Transfer Register)
0x04	R	0x00	Status Register
0x05	RW	0x00	Slave Select Register
0x06	RW	0x07	SPI bit / serial clock control
0x07	RW	0x00	Additional serial clock cycles required after CSn de-activated

25.5.2 Register Descriptions

This section provides a detailed description of all the registers in the SPI Host Controller and the corresponding bits.

25.5.2.1 Register: SPI Baud Register LSB (SPIBR LSB) (Offset 0x00)

This is a Read/Write register and it controls the baud rate of the SPI system.

Bits	Type	Default	Description
7:0	R/W	0x01	Divisor register LSB

25.5.2.2 Register: SPI Baud Register MSB (SPIBR MSB) (Offset 0x01)

This is a Read/Write register and it controls the baud rate of the SPI system.

Bits	Type	Default	Description
7:0	R/W	0x00	Divisor register LSB

The Baud Rate divisor can be calculated using the following formula:

Baud Rate Divisor [15:0] = Bus Clock / 2* Baud Rate

Baud Rate = Bus clock/ 2* Baud Rate Divisor [15:0]

25.5.2.3 SPI Configuration Register (offset 0x02)

- Select SCK phase and polarity
- Interrupt flag enable
- Normal/Bidirectional mode of operation

25.5.2.4 SPI Configuration Register (SPICR) (Offset 0x02)

The SPI control register (SPICR) should be written to only when a transaction is not in progress. This register configures the SPI Host Controller. It controls the mode of operation (Master/Slave), the phase and polarity of the clock, and the transfer type (LSB or MSB first).

Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
SPE	SPIE	BIDIROEn	SPC0	CPOL	CPHA	Reserved	LSBFE

Bit 1 is not implemented (reads or writes to these bits are not implemented). After system reset the following register bits are set to the default values.

Bits	Type	Default	Description
0	R/W	0	LSBFE: 1 = LSB is transferred first 0 = MSB is transferred first
1	-	0	Reserved
2	R/W	0	CPHA: SPI Clock Phase Bit. When CPOL = 0: 0 = data are captured on the clock's rising edge and data is propagated on a falling edge 1 = data are captured on the clock's falling edge and data is propagated on a rising edge When CPOL = 1: 0 = data are captured on the clock's falling edge and data is propagated on a rising edge 1 = data are captured on the clock's rising edge and data is propagated on a falling edge CPHA=0 means sample on the leading (first) clock edge, while CPHA=1 means sample on the trailing (second) clock edge, regardless of whether that clock edge is rising or falling. Note that with CPHA=0, the data must be stable for a half cycle before the first clock cycle
3	R/W	0	**CPOL:** SPI clock polarity bit. This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 1 = Active-low clocks selected; SCK idles high. 0 = Active-high clocks selected; SCK idles low.
4	R/W	0	**SPC0:** Serial pin control bit 0. This bit enables bidirectional pin configuration.
5	R/W	0	**BIDIROEn:** Output enable in the bidirectional mode. 0 = Output buffer enabled. 1 = Output buffer disabled.
6	R/W	0	**SPIE:** SPI interrupt enable bit. This bit enables SPI Interrupts each time the IR or IW status flag is set. 1 = SPI interrupts enabled. 0 = SPI interrupts disabled
7	R/W	0	**SPE:** SPI system enable bit. This bit enables the SPI system and dedicates the SPI port pins to SPI system Functions. 1 = SPI port pins are dedicated to SPI functions. 0 = SPI disabled

25.5.2.5 Transmit Register (Offset 0x03) - Write only

Bits	Type	Default	Description
7:0	W	0x00	Next byte transmitted through SPI interface

Note: In most SPI slave protocol, Bit [7] of this register during address transfer represents the RW bit,

'1' = reading from slave

'0' = writing to slave

25.5.2.6 Receive Register (Offset 0x03) -Read Only

Bits	Type	Default	Description
7:0	R	0x00	Last byte received through SPI interface

25.5.2.7 SPI Command (Transfer) Register (Offset 0x04) -Write Only

This register contains the bit to start/stop a transaction and indicates whether it is a read or a write transaction.

Bits	Type	Default	Description
0	W	0	Start: Start a transaction , cleared automatically
1	W	0	Stop: Stop a transaction, cleared automatically
2	W	0	Write: Performs a write transaction, cleared automatically
3	W	0	Read: Performs a read transaction, cleared automatically
[6:4]	-	0	Reserved
7	W	0	IACK, Interrupt Acknowledge. When set, clears a pending interrupt. This bit is cleared automatically.

25.5.2.8 SPI Interrupt / Status Register (Offset 0x04) -Read Only

This SPI Interrupt/ Status Register (SPIIR) contain Interrupt Flags, indicating the completion of transactions.

- Read Interrupt – indicates a read transaction has been completed
- Write Interrupt – indicates a write transaction has been completed
- TIP, Transfer in progress.

Bit[7]	Bit[6]	Bit[5]	Bit[4:3]	Bit[2]	Bit[1]	Bit[0]
Reserved	Reserved	Reserved	Reserved	TIP	IW	IR

Bits	Type	Default	Description
0	RO	0	IR: Interrupt flag that indicates read is done
1	RO	0	IW: interrupt flag that indicates write is done
2	RO	0	TIP, Transfer in progress. '1' when transferring data '0' when transfer complete
[7:3]	-	0	Reserved

25.5.2.9 Slave Select Register (Offset 0x05)

The SPI slave select register can be used to communicate with up to eight slave devices by setting the corresponding bit. This register configures which slave is selected during a transaction.

Bit[7]	**Bit[6]**	**Bit[5]**	**Bit[4]**	**Bit[3]**	**Bit[2]**	**Bit[1]**	**Bit[0]**
S7	S6	S5	S4	S3	S2	S1	S0

Bits	Type	Default	Description
0	R/W	0	S0: Slave 1 select bit
1	R/W	0	S1: Slave 2 select bit
2	R/W	0	S2: Slave 3 select bit
3	R/W	0	S3: Slave 4 select bit
4	R/W	0	S4: Slave 5 select bit
5	R/W	0	S5: Slave 6 select bit
6	R/W	0	S6: Slave 7 select bit
7	R/W	0	S7: Slave 8 select bit

25.5.2.10 SPI bit / clock control register (Offset 0x06)

Bits	Type	Default	Description
2:0	R/W	0x7	SPI bit / serial clock control Determines the number of SPI bits transaction 111 – 8-bit transaction 110 – 7-bit transaction 101 – 6-bit transaction 100 – 5-bit transaction 011 – 4-bit transaction 010 – 3-bit transaction 001 – 2-bit transaction 000 – 1-bit transaction
7:3	-	0	Reserved

25.5.2.11 Number of SPI clocks required after CSn is de-activated (Offset 0x07)

Bits	Type	Default	Description
2:0	R/W	0x0	Number of additional SPI clock cycle required after CSn is disabled 111 – 8 SPI clock cycles 110 – 7 SPI clock cycles 101 – 6 SPI clock cycles 100 – 5 SPI clock cycles 011 – 4 SPI clock cycles 010 – 3 SPI clock cycles 001 – 2 SPI clock cycles 000 – 1 SPI clock cycle
6:3	-	0	Reserved
7	R/W	0	Additional SPI clock enable 0 – disable additional SPI clocks 1 – enable additional SPI clocks

25.6 Programming

25.6.1 SPI Host Operation

This section describes the configuration and operation of the SPI Host Controller and details how the reads and writes are performed with the external slave device.

25.6.2 Read Operation

To perform a master read operation:

1. Program the Divisor register (SPIBR, Offset Address 0x00 & 0x01, 2 Bytes) for setting up the required baud rate bits.
2. Set up the SPI controller by writing to the SPI configuration register (offset 0x02).
   Note: Steps a through e can be performed with a single write to address offset 0x02
   a. Enable the SPI by setting the SPE bit (bit [7]) to "1"
   b. Configure the shift-out direction by setting the LSBFE bit (bit [0])
      If data is to be transferred least significant bit (LSB) first, set LSBFE to '1'
      If data is to be transferred most significant bit (MSB) first, set LSBFE to '0'
   c. Set the SPI clock phase (CPHA, bit [2]) and polarity bits (CPOL, bit [3]) to the desired setting
   d. Enable the SPI interrupts by setting the SPIE bit (bit [6]) to "1"
3. Select an external slave by asserting the corresponding bit in the Slave Select Register (offset 0x05). For example, to select the slave device connected to the SS_bar [7], set the S7 bit (bit [7]) to "1".
4. Write to the Transmit register (offset 0x03) with the desired address.
5. Start address transfer by writing to the SPI Command/ Transfer Register (offset 0x04)
   Note: Steps a and b can be performed with a single write (0x05) to address offset 0x04
6. After the address byte transaction is complete the SPI controller generates an interrupt signal, **IW** (offset 0x04, bit 1).
   Note: This interrupt signal automatically clears the Start bit (Command register, bit [0]) to "0".
   To acknowledge and clear the interrupt, the processor must write to the IACK bit (bit[7]) in the command register (offset 0x04)
   Also, TIP_o output (Status register, bit [2]) can be monitored to see if SPI transaction is complete
7. Start SPI read transaction by writing to the SPI Command/ Transfer Register (offset 0x04).
   Note: Steps a and b can be performed with a single write (0x09) to address offset 0x04
   a. For a read operation, set the Write bit (bit [2]) to "0" and the Read bit (bit [3]) to "1".
   b. To start the read process, set the Start bit (bit [0]) to "1".
8. After the first byte read transaction is completed the SPI controller generates an interrupt signal to the processor, **IR** (offset 0x04, bit 0).
   Note: This interrupt signal automatically clears the Start bit (Command register, bit [0]) to "0".
   To acknowledge and clear the interrupt, the processor must write to the IACK bit (bit[7]) in the command register (offset 0x04)
9. If more bytes need to be read, steps 7 and 8 are repeated.
10. Once the desired numbers of data bytes have been read, to complete the whole SPI transaction write 1 to Stop bit (bit][1]) of the SPI Command/ Transfer Register. This will make the corresponding SPI_CSn line to de-activate (SPI_CSn goes high)

25.6.3 Write Operation

To perform a master read operation:

1. Program the Divisor register (SPIBR, Offset Address 0x00 & 0x01) for setting up the required baud rate bits.
2. Set up the SPI controller by writing to the SPI configuration register (offset 0x02).
   NOTE: Steps a through e can be performed with a single write to address offset 0x02.
   a. Enable the SPI by setting the SPE bit (bit [7]) to "1".
   b. Configure the shift-out direction by setting the LSBFE bit (bit [0])
      If data is to be transferred least significant bit first, set LSBFE to '1'
      If data is to be transferred most significant bit first, set LSBFE to '0'
   c. Set the SPI clock phase (CPHA, bit [2]) and polarity bits (CPOL, bit [3]) to the desired setting
   d. Enable the SPI interrupts by setting the SPIE bit (bit [6]) to "1".
   e. To enable the bidirectional mode of operation, set the SPC0 bit (bit [4]) to "1" and set the BIDIROE bit (bit [5]) control the direction. For normal mode, set both the bits to "0".
3. Select an external slave by asserting the corresponding bit in the Slave Select Register (offset 0x05). For example, to select the slave device connected to the SS_bar [7], set the S7 bit (bit [7]) to "1".
4. Write to the Transmit register (offset 0x03) with the desired address.
5. Start address transfer by writing to the SPI Command/ Transfer Register (offset 0x04).
   Note: Steps 5a and 5b can be performed with a single write (0x05) to address offset 0x04.
   a. For a write operation, set the Write bit (bit [2]) to "1" and the Read bit (bit [3]) to "0".
   b. To start the SPI transaction process, set the Start bit (bit [0]) to "1".
6. After the address byte transaction is complete the SPI controller generates an interrupt signal, IW (offset 0x04, bit 1).
   Note: This interrupt signal automatically clears the Start bit (Command register, bit [0]) to "0".
   To acknowledge and clear the interrupt, the processor must write to the IACK bit (bit[7]) in the command register (offset 0x04)
7. Write to the Transmit register (offset 0x03) with the desired Data.
8. Start data transfer by writing to the SPI Command/ Transfer Register (offset 0x04).
9. After the Data byte transaction is complete the SPI controller generates an interrupt signal, IW (offset 0x04, bit 1).
   Note: This interrupt signal automatically clears the Start bit (Command register, bit [0]) to "0".
   To acknowledge the interrupt, the processor must read the SPI interrupt register (offset 0x04) to clear.
10. If more bytes need to be written, steps 8 and 9 are repeated.
11. Once the desired numbers of data bytes are written, to complete the whole SPI transaction, write 1 to Stop bit (bit][1]) of the SPI Command/ Transfer Register. This will make the corresponding SPI_CSn line to de-activate (SPI_CSn goes high)
12. If additional SPI clock enable bit is set (bit[7] register offset 0x07) then SPI master will drive the SPI clock, for the number of SPI clocks specified (bit (2:0) register offset 0x07), after the SPI_CSn is disabled.

Chapter 26. Sensor sub-system registers

The Sensor Manager is programmed using the QuickLogic compiler for the Sensor Manager instruction set.

Control for the Sensor Hub is also performed through direct operations with FFE registers. The I^2C and SPI interfaces are accessed by indexed write and read operations performed through FFE registers. For example, addressing interfaces on the Wishbone buses is performed by write to the FFE WB_ADDR register.

The following describes the FFE register set.

26.1 FFE registers

Offset	Name/Field	Bits	Type	Default	Description	Retention?
0x000	WB_ADDR	7:0	RW		Wishbone Bus Master address selection	No
	Addr	2:0	RW	0x0	Slave address register via Wishbone Bus master	
	RESERVED	5:3	RW	0x0	Reserved	
	slave_sel	7:6	RW	0x0	The two MSBs select which slave is accessed by Wishbone Bus master 0x0 = I2C_0 0x1 = I2C_1 0x2 = SPI_0	
0x004	WDATA	7:0	RW	0x0	This register is for writes to an I2C slave. It is the I2C slave data register selected via the Wishbone Bus master	No
0x008	CSR	7:0	RW		Control and Status register	No
	spi0_mux_sel	7	RW	0x0	SPI_0_Master wishbone control mux select 1 = Control from Wishbone Bus Master 0 = Control from SM1	
	i2c1_mux_sel	6	RW	0x0	I2C_1 wishbone control mux select 1 = Control from Wishbone Bus Master 0 = Control from SM1	
	i2c0_mux_sel	5	RW	0x0	I2C_0 wishbone control mux select 1 = Control from WB Master 0 = Control from SM0	
	OVFL	4	RO	0x0	ffe_push_overflow	
	BUSY	3	RO	0x0	WB busy	
	mux_wb_sm	2	WO	0x0	Mux select choosing between SM and WB masters 1 = WB master 0 = SM WB master)	
	wb_ms_wen	1	RW	0x0	WB master write enable 1 = Write 0 = Read	
	wb_ms_start	0	RW	0x0	WB master start transactions. This bit is self clear	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
0x00C	RDATA	7:0	RO	0x0	This register is for reads from an I2C slave. Read data from I2C to WB master is registered	No
0x014	SRAM_test_reg1	31:0	RW	0x0	SRAM test control register 1	No
	DM1_RME	31	RW	0x0	DM1_RME control to FFE SRAM module	
	DM1_TEST1	30	RW	0x0	DM1_TEST1 control to FFE SRAM module	
	DM0_RM	29:26	RW	0x0	DM0 RM [3:0] control to FFE SRAM module	
	DM0_RME	25	RW	0x0	DM0_RME control to FFE SRAM module	
	DM0_TEST1	24	RW	0x0	DM0_TEST1 control to FFE SRAM module	
	CM2k_RM	23:20	RW	0x0	CM2k_RM [3:0] control for FFE SRAM	
	CM2k_RME	19	RW	0x0	CM2k_RME control for FFE SRAM	
	CM2k_TEST1	18	RW	0x0	CM2k_TEST1 control for FFE SRAM	
	CM8k_RM	17:14	RW	0x0	CM8k_RM [3:0] control for FFE SRAM	
	CM8k_RME	13	RW	0x0	CM8k_RME control for FFE SRAM	
	CM8k_TEST1	12	RW	0x0	CM8k_TEST1 control for FFE SRAM	
	SM0_RM	11:8	RW	0x0	SM0_RM[3:0] control for FFE SRAM	
	SM0_RME	7	RW	0x0	SM0_RME control for FFE SRAM	
	SM0_TEST1	6	RW	0x0	SM0_TEST1 control for FFE SRAM	
	SM1_RM	5:2	RW	0x0	SM1_RM[3:0 control for FFE SRAM	
	SM1_RME	1	RW	0x0	SM1_RME control for FFE SRAM	
	SM1_TEST1	0	RW	0x0	SM1_TEST1 control for FFE SRAM	
0x018	SRAM_test_reg2	31:0	RW	0x0	SRAM test control register 2	No
	DM3_RM	15:12	RW	0x0	DM3 RM [3:0] control to FFE SRAM module	
	DM3_RME	11	RW	0x0	DM3_RME control to FFE SRAM module	
	DM3_TEST1	10	RW	0x0	DM3_TEST1 control to FFE SRAM module	
	DM2_RM	9:6	RW	0x0	DM2 RM [3:0] control to FFE SRAM module	
	DM2_RME	5	RW	0x0	DM2_RME control to FFE SRAM module	
	DM2_TEST1	4	RW	0x0	DM2_TEST1 control to FFE SRAM module	
	DM1_RM	3:0	RW	0x0	DM1 RM [3:0] control to FFE SRAM module	
0x020	FFE_CSR	31:0	RW	0x0	FFE Control and Status Register	Yes
	i2c2_sel	0	RW	0x0	1: selects i2c 2 0: selects i2c 1	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
	I2c0_dyn_pullup_en	1	RW	0x0	I2c0 dynamic pull-up enable	
	I2c1_dyn_pullup_en	2	RW	0x0	I2c1 dynamic pull-up enable	
	I2c2_dyn_pullup_en	3	RW	0x0	I2c2 dynamic pull-up enable	
	RESERVED	31:4	RO	0x0	Reserved	
0x038	FFE_DEBUG_COMBINED	31:0	RO		Combined FFE debug signals	No
	SM0_SM_debug	7:0	RO	0x0	Sensor memory 0 debug signals	
	SM1_SM_debug	15:8	RO	0x0	Sensor memory 1 debug signals	
	FFE_debug	23:16	RO	0x0	FFE debug signals	
0x100	COMMAND	31:0				No
	RUN_FFE_ONCE	0	WO	0x0	When a '1' is written to this location, causes the FFE to execute one complete run of its algorithm; reads as 0	
	RESERVED	1	WO	0x0	Reserved	
	RUN_SM0_ONCE	2	WO	0x0	When a '1' is written to this location, causes SM0 to run once; reads as 0	
	RUN_SM1_ONCE	3	WO	0x0	When a '1' is written to this location, causes SM1 to run once; reads as 0	
0x108	INTERRUPT	31:0				Yes
	SM_MULT_WR_INTR	0	RW1C	0x0	This bit is set when an FFE tries to write to more than one FIFO simultaneously. The FIFO PUSH value must be one hot, with only one pushd asserted.	
	FFE_OVERRUN	1	RW1C	0x0	This bit is set when the FFE does not complete running its code (the algorithm) by the time the next sample period begins. This bit can only be cleared by issuing a device reset (software reset, or global reset via the Reset pin).	
	RESERVED	2	RO	0x0	Reserved	
	RESERVED	3	RO	0x0	Reserved	
	FFE_SM1_OVERRUN	4	RW1C	0x0	This bit is set when SM1 does not complete the algorithm by the time the next sample period begins. This bit can only be cleared by issuing a device reset (software reset, or global reset via the Reset pin).	
	FFE_SM0_OVERRUN	5	RW1C	0x0	This bit is set when SM0 does not complete the algorithm by the time the next sample period begins. This bit can only be cleared by issuing a device reset (software reset, or global reset via the Reset pin).	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
	I2C_MS_1_ERROR	6	RW1C	0x0	This bit is set when the I2C Master receives a NACK when transmitting a device address. The I2C Master is used by the Sensor Manager to retrieve sensor data.	
	I2C_MS_0_ERROR	7	RW1C	0x0	This bit is set when the I2C Master receives a NACK when transmitting a device address. The I2C Master is used by the Sensor Manager to retrieve sensor data.	
	CM_8k_LP_INTR	8	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	DM0_LP_INTR	9	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	DM1_LP_INTR	10	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	SM0_LP_INTR	11	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	SM1_LP_INTR	12	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	FFE_BP_MATCH_INTR	13	RW1C	0x0	This bit is set when there is a break point match in FFE	
	Reserved	14	RW1C	0x0	Reserved	
	PKFB_OVF_INTR	15	RW1C	0x0	This bit is set when the FFE pushes to the PKFB causing an overflow	
	SM0_BP_MATCH_INTR	16	RW1C	0x0	This bit is set when there is a break point match in SM0	
	SM1_BP_MATCH_INTR	17	RW1C	0x0	This bit is set when there is a break point match in SM1	
	SPI_0_MS_INTR	18	RW1C	0x0	This bit is set when there is an interrupt request from SPI_0_Master for sensor	
	CM_2k_LP_INTR	19	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	DM2_LP_INTR	20	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	DM3_LP_INTR	21	RW1C	0x0	This bit is set when there is an access to the memory while it is in low power (deep sleep or shut down)	
	ahbm_bus_error_intr	22	RW1C	0x0	This bit is set when there is a bus error on the AHB bus (HRESP=1).	
0x10c	INTERRUPT_ENABLE	31:0				Yes
	SM_MULT_WR_INTR_EN	0	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
	FFE_OVERRUN_EN	1	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	RESERVED	2	RW	0x0	Reserved	
	RESERVED	3	RW	0x0	Reserved	
	FFE_SM1_OVERRUN_EN	4	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	FFE_SM0_OVERRUN_EN	5	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	I2C_MS_1_ERROR_EN	6	RW	0x0	Interrupt enabled for I2C_1_Master 0: mask interrupt 1: enable interrupt	
	I2C_MS_0_ERROR_EN	7	RW	0x0	Interrupt enabled for I2C_0_Master 0: mask interrupt 1: enable interrupt	
	CM_8k_LP_INTR_EN	8	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	DM0_LP_INTR_EN	9	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	DM1_LP_INTR_EN	10	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	SM0_LP_INTR_EN	11	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	SM1_LP_INTR_EN	12	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	FFE_BP_MATCH_INTR_EN	13	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	Reserved	14	RW	0x0	Reserved	
	PKFB_OVF_INTR_EN	15	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	SM0_BP_MATCH_INTR_EN	16	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	SM1_BP_MATCH_INTR_EN	17	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	SPI_MS_INTR_EN	18	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
	CM_2k_LP_INTR_EN	19	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	DM2_LP_INTR_EN	20	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	DM3_LP_INTR_EN	21	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
	ahbm_bus_error_intr_en	22	RW	0x0	Interrupt enabled 0: mask interrupt 1: enable interrupt	
0x110	STATUS	31:0				No
	SM0_BUSY	0	RO	0x0	This bit is set whenever the Sensor Manager 0 is busy.	
	SM1_BUSY	1	RO	0x0	This bit is set whenever the Sensor Manager 1 is busy.	
	FFE_BUSY	2	RO	0x0	This bit is set whenever the FFE is busy.	
	Reserved	3	RO	0x0	Reserved.	
	FFE_BG_FLAG	4	RO	0x0	This is the FFE background thread status	
	FFE_FG_FLAG	5	RO	0x0	This is the FFE foreground thread status	
0x114	MAILBOX_TO_FFE	31:0				yes
	MAILBOX_TO_FFE	31:0	RW	0x0	This is the M4 Mailbox register to the FFE. This register can be set by system software to send a message or configuration information to the FFE as it runs its algorithm, thus affecting the algorithm while it is running. A special instruction may be used in the algorithm to read this mailbox register.	
0x120	SM_RUNTIME_ADDR	31:0				yes
	SM_RUNTIME_ADDR	9:0	RW	0x0	SM0/SM1 run time address	
0x124	SM0_RUNTIME_ADDR_CTRL	31:0				No
	SM0_RUNTIME_ADDR_NEW	0	RW	0x0	Write a '1' to Toggle signal used to signal when a new value has been written	
0x128	SM1_RUNTIME_ADDR_CTRL	31:0				No
	SM1_RUNTIME_ADDR_NEW	0	RW	0x0	Write a '1' to Toggle signal used to signal when a new value has been written	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
0x12c	SM0_RUNTIME_ADDR_CUR	31:0				No
	SM0_RUNTIME_ADDR_CUR	9:0	RO	0x0	SM0 current program counter	
0x130	SM1_RUNTIME_ADDR_CUR	31:0				No
	SM1_RUNTIME_ADDR_CUR	9:0	RO	0x0	SM1 current program counter	
0x140	SM0_DEBUG_SEL	31:0				No
	SM0_DEBUG_SEL	7:0	RW	0x0	SM0 Debug selection	
0x144	SM1_DEBUG_SEL	31:0				No
	SM1_DEBUG_SEL	7:0	RW	0x0	SM1 Debug selection	
0x148	FFE_DEBUG_SEL	31:0				
	FFE_DEBUG_SEL_SM0	7:0	RW	0x0	SM0 LS debug selection	Yes
	FFE_DEBUG_SEL_SM1	15:8	RW	0x0	SM1 LS debug selection	
	FFE_DEBUG_SEL_FFE0	23:16	RW	0x0	FFE LS debug selection	
	FFE_TOP_DEBUG_SEL	25:24	RW	0x0	FFE_TOP MS debug selection 0: SM0 1: SM1 2: FFE	
0x150	FFE_BREAK_POINT_CFG	31:0				yes
	FFE_BP_EN	0	RW	0x0	"break point" execution 0 : Disabled 1 : Enabled	
	FFE_FORCE_STOP	1	RW	0x0	This causes the FFE to immediately halt execution.	
	FFE_BreakPt_Sw_Brk	2	RW1T	0x0	Causes the signal to toggle when written with a '1' in this bit position.	
0x154	FFE_BREAK_POINT_CONT	31:0				No
	SM1_BP_CONT	2	RW	0x0	This is a single, host controlled input toggle signal, Break Point Match Continue. Software uses this toggle signal to resume code execution from the Break Point condition.	
	SM0_BP_CONT	1	RW	0x0	This is a single, host controlled input toggle signal, Break Point Match Continue. Software uses this toggle signal to resume code execution from the Break Point condition.	

Offset	Name/Field	Bits	Type	Default	Description	Retention?
	FFE_BP_CONT	0	RW	0x0	This restarts FFE code execution following a pause due to reaching a "break point".	
0x158	FFE_BREAK_POINT_STAT	31:0				No
	SM1_BP_MATCH	2	RO	0x0	SM Break Point Match signal output to the host that notifies the host that the break point condition has been detected.	
	SM0_BP_MATCH	1	RO	0x0	SM Break Point Match signal output to the host that notifies the host that the break point condition has been detected.	
	FFE_BP_MATCH	0	RO	0x0	This signals that a "break point" has been reached and FFE execution is paused.	
0x160	FFE_BP_XPC_0	31:0				Yes
	FFE_BP_XPC_0	13:0	RW	0x0	These registers hold the xPC (program counter) address "break points".	
0x164	FFE_BP_XPC_1	31:0				yes
	FFE_BP_XPC_1	13:0	RW	0x0	These registers hold the xPC (program counter) address "break points".	
0x168	FFE_BP_XPC_2	31:0				Yes
	FFE_BP_XPC_2	13:0	RW	0x0	These registers hold the xPC (program counter) address "break points".	
0x16C	FFE_BP_XPC_3	31:0				Yes
	FFE_BP_XPC_3	13:0	RW	0x0	These registers hold the xPC (program counter) address "break points".	

Chapter 27. FPGA sub-system

27.1 Introduction

This sub system provides capability to add logic for custom operating functions.

Configuration for 2400 effective logic cells is stored in 64 Kbit SRAM.

27.1.1 Power

Power Domain: FPGA

27.2 Sub-system Architecture

The on-chip programmable logic provides flexibility to the EOS S3 platform for implementing additional, customer-determined, functions. The logic block consists of multiplexor based logic cells, built-in RAM modules and FIFO controllers, built-in multipliers, as well as pre-built interfaces with I/O drivers of the EOS S3 device. Key functionality is listed in the following table.

Table 27-1: On-Chip Programmable Logic Major Features

Feature	Parameters
Logic Cells	891
8K RAM Modules (9,216 bits)	8
FIFO Controllers	8
RAM Bits	65,536
Configurable Interface	32
Multiplier	2x 32 x 32 (or 4x 16 x 16)

27.2.1 FPGA sub-system components

- Configuration Memory
- Configuration controller - [description to be added]
- Configurable logic matrix
- Bus Interface
- Eight RAM FIFO Controllers

27.2.2 Functional Description

27.2.2.1 Logic Cell

Each logic cell is a multiplexer-based single register. The cell has a high fan-in, and fits a wide range of functions with up to 22 simultaneous inputs (including register control lines), and four outputs (three combinatorial and one registered). Figure 27-1, Logic Cell Block Diagram, illustrates the logic block structure. The high logic capacity and fan-in of the logic cell accommodates many user functions with a single level of logic delay. The logic cell can implement the following functions:

- Two independent 3-input functions
- Any 4-input function
- 8 to 1 mux function
- Independent 2 to 1 mux function
- Inverted or non-inverted clock signal to flip-flop
- Single dedicated register with active high clock enable, set and reset signals
- Direct input selection to the register, which allows combinatorial and register logic to be used separately
- Combinatorial logic can also be configured as an edge-triggered master-slave D flip-flop

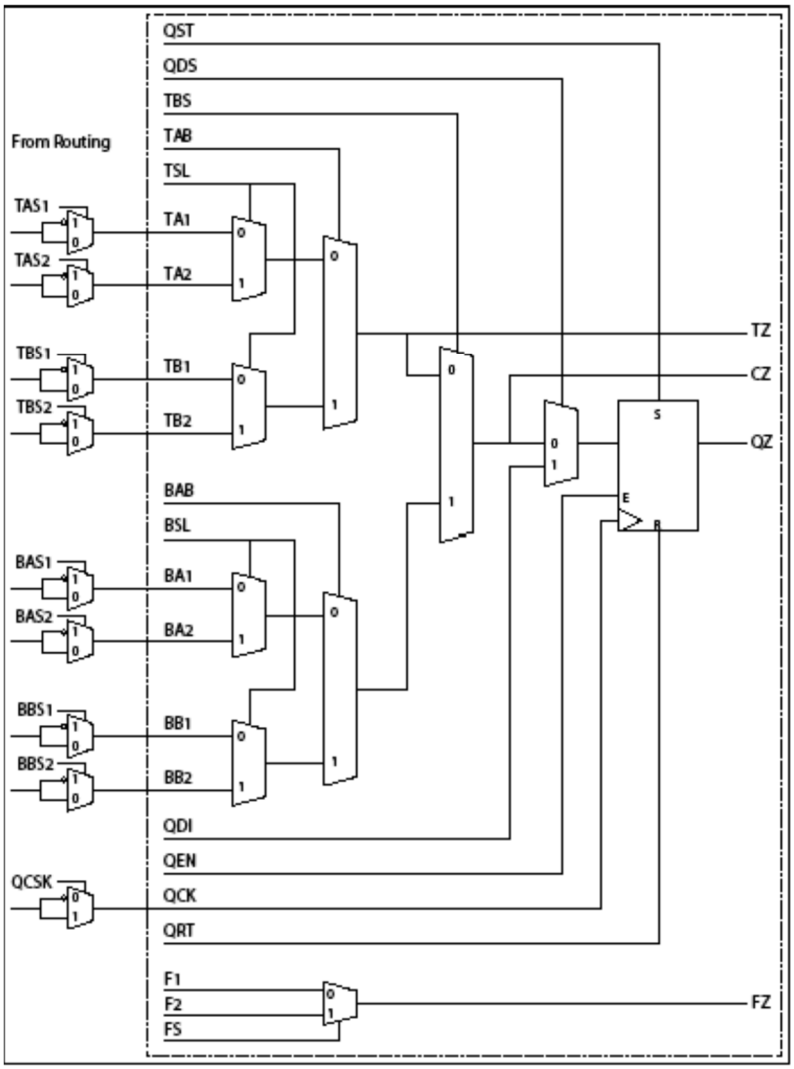

Figure 27-1: Logic Cell Block Diagram

27.2.2.2 RAM/FIFO

The on-chip programmable logic also includes up to eight 8 kilobits (9,216 bits) dual-port RAM modules for implementing RAM and FIFO functions.

RAM features include:

- Independently configurable read and write data bus widths
- Independent read and write clocks
- Inverted or non-inverted clock signals to read and write clock inputs
- Horizontal and vertical concatenation
- Write byte enables
- Selectable pipelined or non-pipelined read data

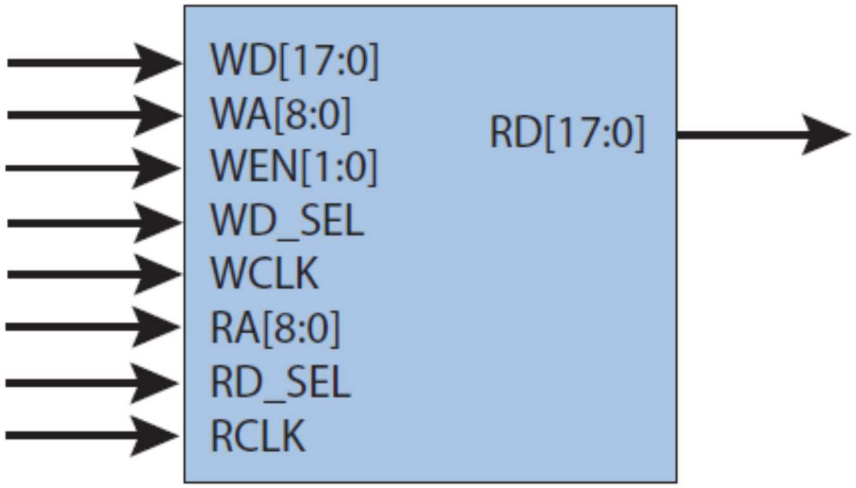

Figure 27-2: 8Kbit Dual Port RAM block in 512x18 configuration

Table 27-2: RAM Interface Signals

Signal Name	Function
Inputs	
WD[17:0]	Write Data
WA[8:0]	Write Address
WEN[1:0]	Write Enable (two 9-bit enables)
WD_SEL	Write Chip Select
WCLK	Write Clock
RA[8:0]	Read Address
RD_SEL	Read Chip Select
RCLK	Read Clock
Output	
RD[17:0]	Read Data Output

The read and write data buses of a RAM block can be arranged to variable bus widths. The bus widths can be configured using the RAM Wizard available in QuickWorks, QuickLogic's development software. The selection of the RAM depth and width determines how the data is addressed.

The RAM blocks also support data concatenation. Designers can cascade multiple RAM modules to increase the depth or width by connecting corresponding address lines together and dividing the words between modules. Generally, this requires the use of additional programmable logic resources. However, when concatenating only two 8-kilobit RAM blocks, they can be concatenated horizontally or vertically without using any additional programmable fabric resources.

For example, two internal dual-port RAM blocks can be concatenated vertically to create a 1024x18 RAM block or horizontally to create a 512x36 RAM block. A block diagram of horizontal and vertical concatenation is displayed in Figure below.

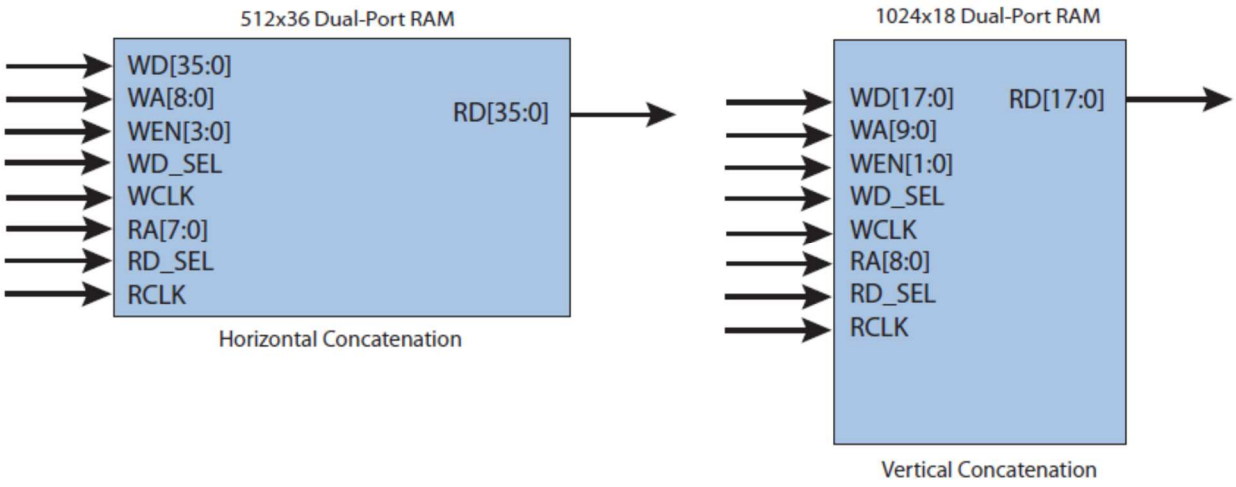

Figure 27-3: Horizontal and Vertical Concatenation Examples

Table 27-3: Available Dual-Port Configuration

Number of RAM Blocks	Depth	Support Widths
1	512	1-18 bits
2	512	1-36 bits
2	1024	1-18 bits
2	2048	1-9 bits

PolarPro 3 Dual-Port RAM modules can also be concatenated to generate True Dual-Port RAMs. The True Dual-Port RAM module's Port1 and Port2 have completely independent read and write ports and separate read and write clocks. This allows Port1 and Port2 to have different data widths and clock domains from each other. It is important to note that there is no circuitry preventing a write and read operation to the same address space at the same time. Therefore, the designer must ensure that the same address is not read from and written to simultaneously, otherwise the data is considered invalid. Likewise, the same address should not be written to from both ports at the same time. However, it is possible to read from the same address.

Figure 27-4 shows an example of a 1024x18 true Dual-Port RAM.

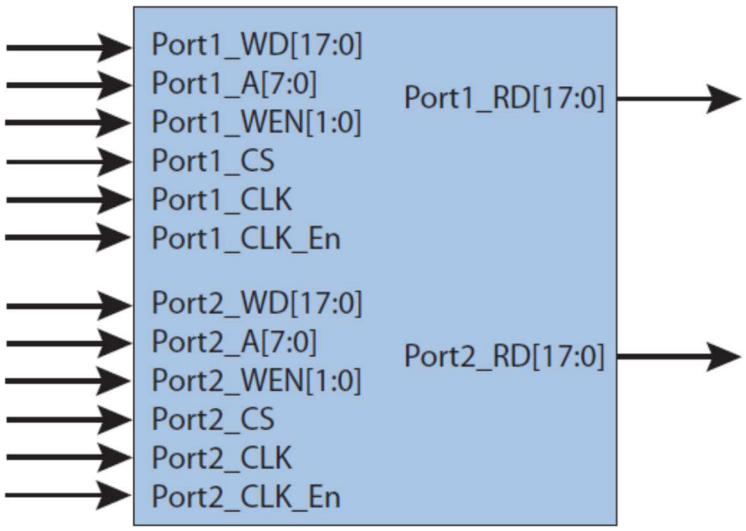

Figure 27-4: 1024x18 True Dual-Port RAM Block

Table 27-4 describes the true dual-port RAM interface signals.

Table 27-4: True Dual-Port RAM Interface Signals

Port	Signal Name	Function
Port1	Inputs	
	Port1_WD[17:0]	Write Data
	Port1_A[9:0]	Write Address
	Port1_WEN	Write Enable
	Port1_CS	Chip Select
	Port1_CLK	Clock
	Output	
	Port1_RD[17:0]	Read Data
Port2	Inputs	
	Port2_WD[17:0]	Write Data
	Port2_A[9:0]	Write Address
	Port2_WEN	Write Enable
	Port2_CS	Chip Select
	Port2_CLK	Clock
	Output	
	Port2_RD[17:0]	Read Data

Table 27-5 lists the true dual-port configurations that are available.

Table 27-5: Available True Dual-Port Configurations

Number of RAM Blocks	Depth	Width
2	1024	1-18
2	2048	1-9

27.2.2.3 FIFO Controller

Every 8-kilobit RAM block can also be implemented as a synchronous or asynchronous FIFO. There are built-in FIFO controllers that allow for varying depths and widths without requiring on-chip programmable logic resources. During asynchronous operation, the FIFO works in a half-duplex fashion such that PUSH is on one clock domain and POP is on another clock domain. The DIR signal allows the FIFO PUSH and POP signal directions to reverse.

FIFO controller features include:

- x9, x18 and x36 data bus widths
- Independent PUSH and POP clocks
- Independent programmable data width on PUSH and POP sides
- Configurable synchronous or asynchronous FIFO operation
- 4-bit PUSH and POP level indicators to provide FIFO status outputs for each port
- Pipelined read data to improve timing
- Option for inverted or non-inverted asynchronous flush input

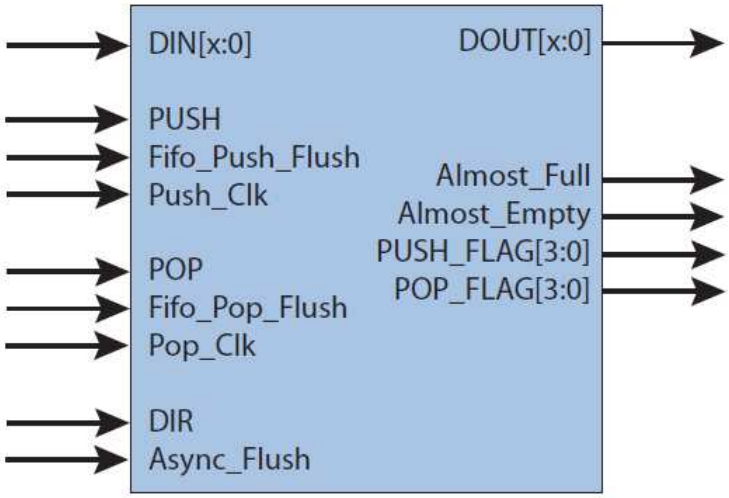

Figure 27-5: eFPGA FIFO Module

Table 27-6: FIFO Interface Signals

Signal Name	Width (bits)	Direction	Function
PUSH Signals			
DIN	1 to 36	Input	Input data bus
PUSH	1	Input	Initiate a data push
Fifo_Push_Flush	1	Input	Empties the FIFO
Push_Clk	1	Input	Push Data Clock
POP Signals			
DOUT	1 to 36	Ouptut	Output Data Bus
POP	1	Input	Initiate a data pop
Fifo_Pop_Flush	1	Input	Empties the FIFO
Pop_Clk	1	Input	Pop Data Clock
Status Flags			
Almost_Full	1	Output	Assert when FIFO has one location available
Almost_Empty	1	Output	Assert when FIFO has one data left
PUSH_FLAG	4	Output	FIFO PUSH level indicator
POP_FLAG	4	Output	FIFO POP level indicator
Asynchronous Signals			
DIR	1	Input	Determine the direction of the PUSH and POP signals: 0: Signals set as normal 1: Reverses the FIFO direction so that the PUSH signals become POP signals and vice versa.
Async_Flush	1	Input	Asynchornous input signal to flush FIFO. Used to reset FIFO logic asynchronously

Table 27-7: Available FIFO Configurations

Number of RAM Blocks	Depth	Supported Widths
1	512	1-18 bits
2	512	1-36 bits
2	1024	1-18 bits
2	2048	1-9 bits

Table 27-8 and 27-9 highlight the corresponding FIFO level indicator for each 4-bit value of the PUSH_FLAG and POP_FLAG outputs.

Table 27-8: FIFO Push Level Indicator Values

Value	Status
0000	Full
0001	Empty
0010	Room for more than ½
0011	Room for more than ¼
0100	Room for less than ¼
1010	Room for 32 to 63
1011	Room for 16 to 31
1100	Room for 8 to 15
1101	Roo for 4 to 7
1110	Room for atleast 2
1111	Room for atleast
Others	Reserved

Table 27-9: FIFO Pop Level Indicator Values

Value	Status
0000	Empty
0001	1 entry in FIFO
0010	Atleast 2 entries in FIFO
0011	Atleast 4 entries in FIFO
0100	Atleast 8 entries in FIFO
0101	Atleast 16 entries in FIFO
0111	Atleast 32 entries in FIFO
1000	Less than ¼ to 64
1101	¼ or more full
1110	½ or more full
1111	Full
Others	Reserved

FIFO Synchornous Flush Procedure

Both PUSH and POP domains are provided with a flush input signal synchronized to their respective clocks. When a flush is triggered from one side of the FIFO, the signal propagates and re-synchronizes internally to the other clock domain. During a flush operation, the values of the FIFO flags are invalid for a specific number of cycles (see Figure 27-6 and Figure 27-7).

As shown in Figure 27-6, when the **Fifo_Push_Flush** asserts, the **Almost_Full** and **PUSH_FLAG** signals become invalid until the FIFO can flush the data with regards to the Push clock domain as well as the Pop clock domain. After the **Fifo_Push_Flush** is asserted, the next rising edge of the Pop clock starts the Pop flush routine.

Figure 27-6 illustrates a FIFO Flush operation. After the **Fifo_Push_Flush** is asserted at 2 (**PUSH_Clk**), two POP clock cycles (**11** and **12**) are required to update the **POP_FLAG**, and **PUSH_FLAG** signals. The **Almost_Empty** signal is asserted to indicate that the push flush operation has been completed. On the following rising edge of the **PUSH_Clk** (7), the **PUSH_FLAG** is accordingly updated to reflect the successful flush operation.

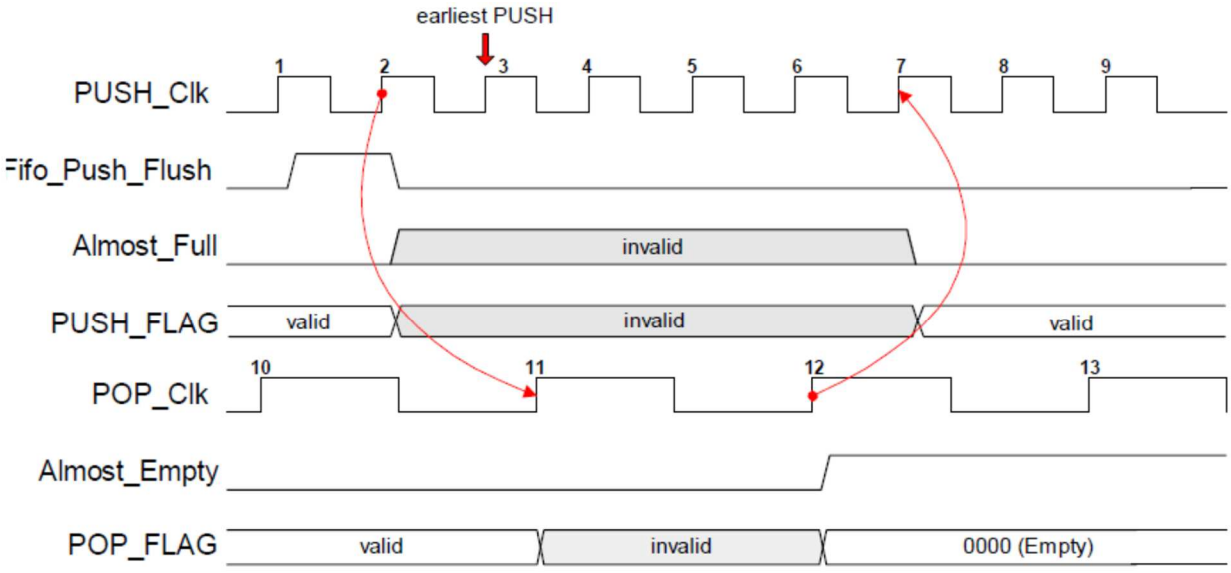

Figure 27-6: FIFO Flush from PUSH side

Figure 27-7 illustrates a POP flush operation. After the **Fifo_Pop_Flush** is asserted at 2 (**POP_Clk**), two PUSH clock cycles (**11** and **12**) are required to update the **POP_FLAG** and **PUSH_FLAG** signals. The **Almost_Empty** signal is asserted to indicate that the pop flush operation has been completed. On the following rising edge of the **POP_Clk** (7), the **POP_FLAG** is updated accordingly to reflect the successful flush operation.

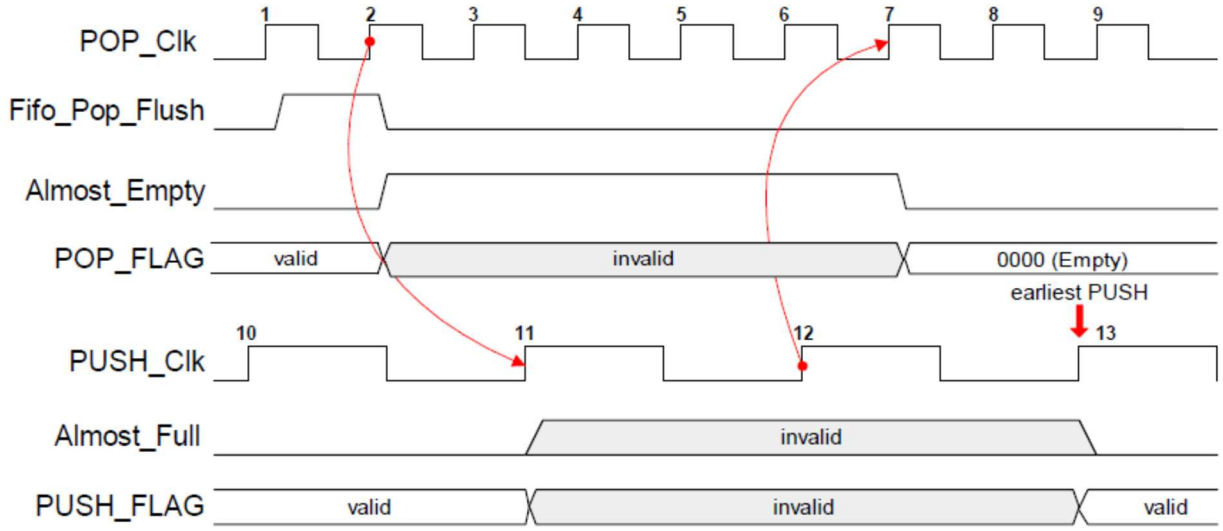

Figure 27-7: FIFO Flush from POP Side

Figure 27-6 and Figure 27-7 are only true for this particular PUSH-POP clock frequency combination. The clock frequency and phase difference between POP_Clk and PUSH_Clk can cause an additional flush delay of one clock cycle in either domain because of the asynchronous relationship between the two clocks.

FIFO Asynchornous Flush Procedure

Aside from the synchronous flush controls, there is an asynchronous flush signal named ASYNC_FLUSH. This signal is tied directly to the PUSH and POP pointers without any deglitching circuitry. The designer can add deglitching circuitry if desired.

27.2.2.4 Distributed Clock Networks

Global Clocks

The FPGA clock network architecture consists of a 2-level H-tree network as shown in Figure 27-8. The first level of each clock tree (high-lighted in red) spans from the clock input pad to the global clock network and to the center of each quadrant of the chip. The second level spans from the quadrant clock network to the column clock network, and then to every logic cell inside that quadrant. There are five global clocks in the global clock network, and five quadrant clocks in each quadrant clock network. All global clocks drive the quadrant clock network inputs.

The quadrant clocks output to column clock buffers, which can be dynamically disabled at the column level. Column clock buffers can be implemented in Verilog, VHDL, and schematic designs by instantiating the column clock buffer macro, CAND. Figure 27-9 shows the schematic representation of the CAND macro. The global clocks can drive RAM block clock inputs and reset, set, enable, and clock inputs to I/O registers. Furthermore, the quadrant clock outputs can be routed to all logic cell inputs.

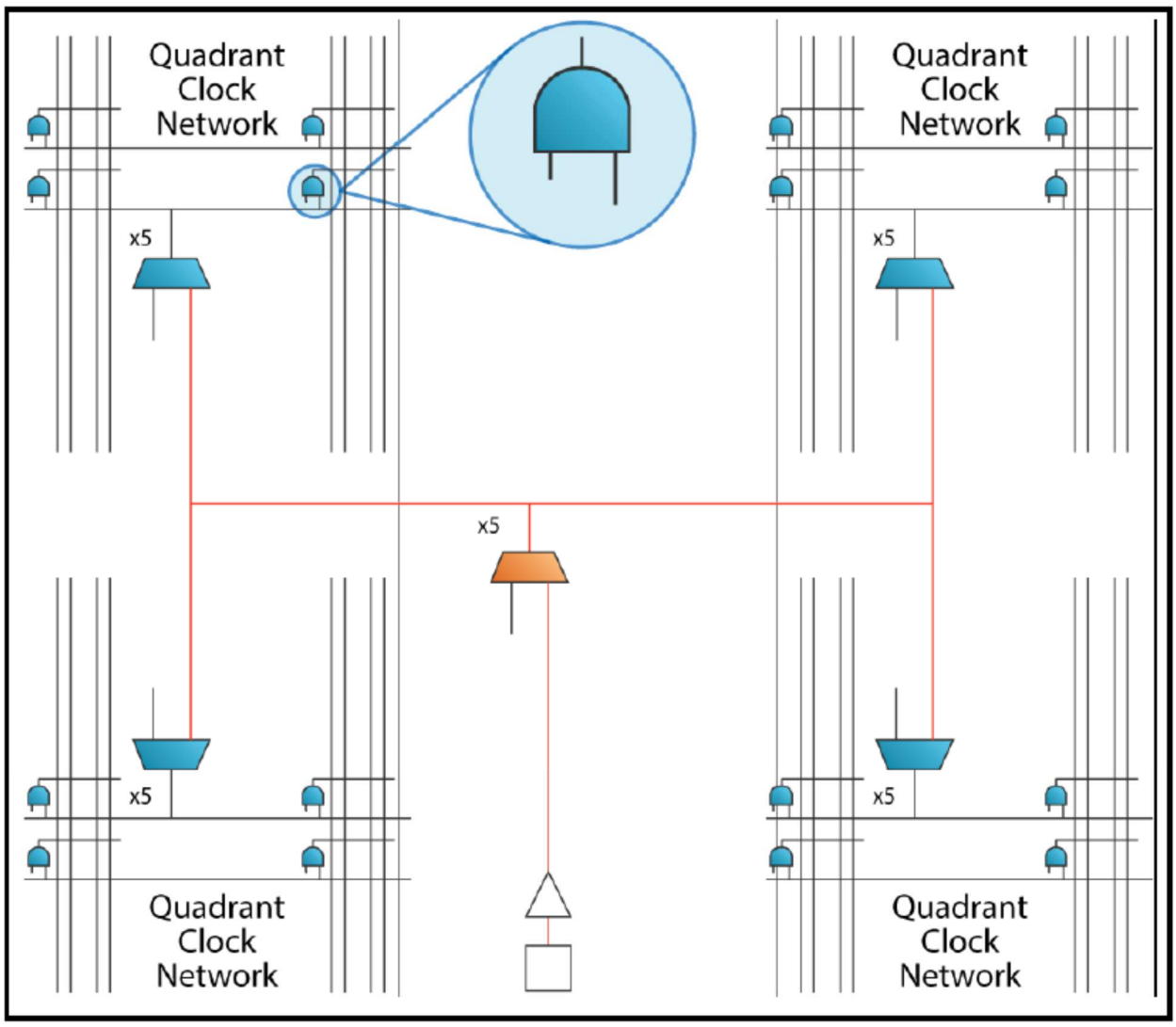

Figure 27-8: FPGA Clock Architecture

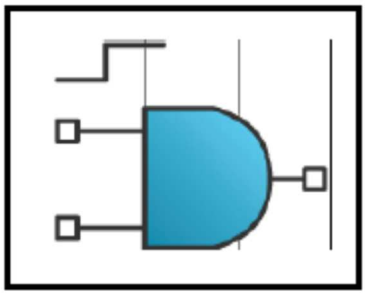

Figure 27-9: CAND Macro

The five global clock networks can either be driven directly by clock pads or internally generated signals. These global clocks go through 2-input global clock muxes located in the middle of the die. A diagram of a 2-input global clock mux is shown in Figure 27-10.

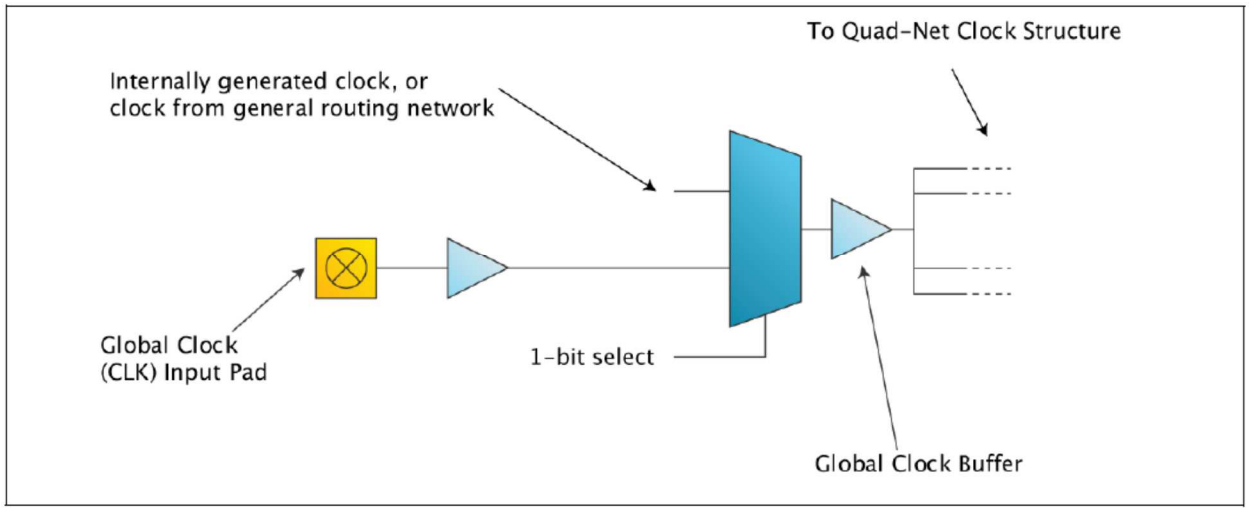

Figure 27-10: Global Clock Structure

Figure 27-11 illustrates the quadrant clock 2-input mux.

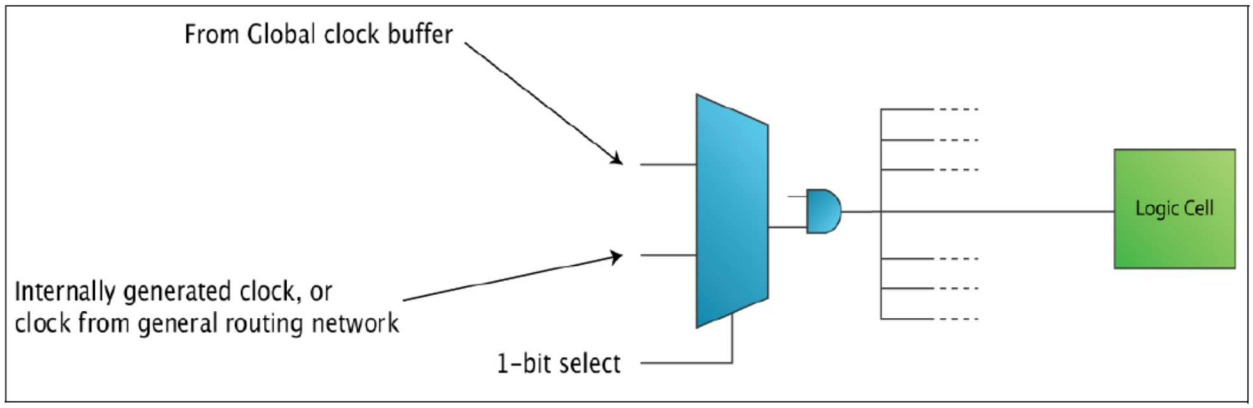

Figure 27-11: Quadrant Clock Structure

Note: Select lines for the global clock and quadrant clock muxes are static signals and can't be changed dynamically during device operation.

27.2.2.5 Configurable Input/Output Signals

Configurable IO interface provides additional functionality prior to driving the actual IO. This additional functionality is comprised of the following:

- Register path versus non register path for IN, OUT, and EN
- FIX_HOLD feature to improve hold time

Figure 27-12, On-Chip Programmable Logic Configurable Input/Output for on-chip programmable logic configurable I/Os, shows the I/O.

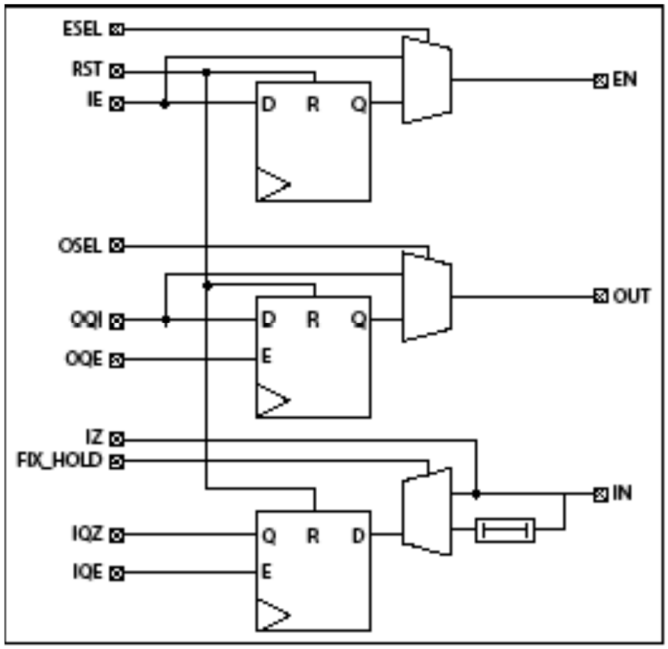

Figure 27-12: On-Chip Programmable Logic Configurable Input/Output

27.2.2.6 Multipliers

Built-in signed multipliers are also available in the on-chip programmable logic. The multiplier relieves the use of logic to implement such functions. There are two instances embedded in the on-chip programmable logic. The multiplier can be configured as one 32x32 bit multiplier or two 16x16 bit multipliers.

A block diagram of the multiplier is shown in Figure 27-13.

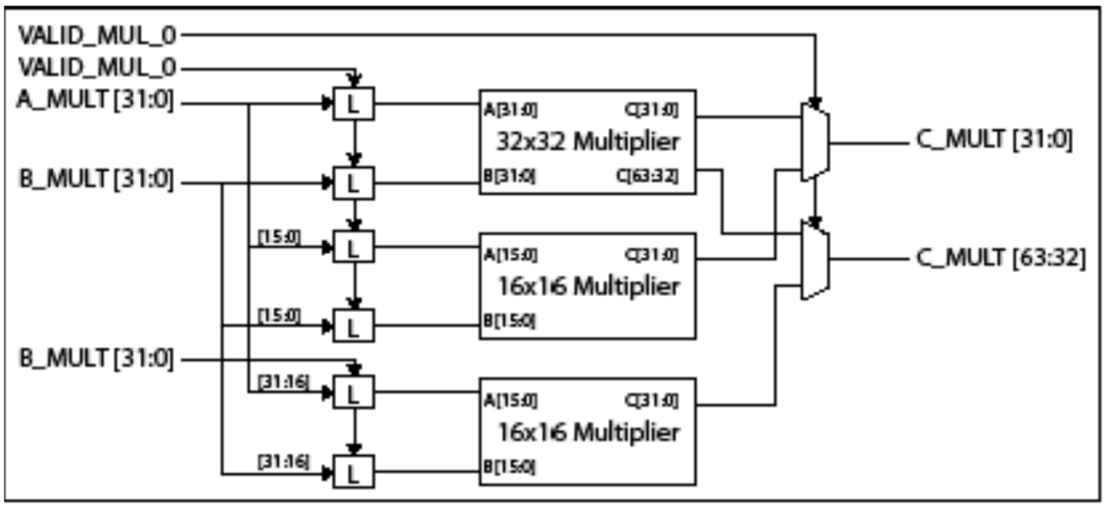

Figure 27-13: On-Chip Programmable Logic Multiplier

27.2.3 Interface to the On-Chip Programmable Logic

IP within the on-chip programmable logic can use the following interfaces to communicate with resources outside of the chip. These include:

- S3 Platform
- SPI Master Interface for System Support
- Sensor Processing Subsystem
- Packet FIFO

These resources help the IP to coordinate its activities with other modules in the EOS S3 platform. Additionally, the IP can call upon external resources to support its processing activities. The following sections will cover what resources are available to IP within the on-chip programmable logic. It is important to note that it is not essential that IP within the on-chip programmable logic use these interfaces.

27.2.4 S3 Platform Interface

The interface between IP in the on-chip programmable logic and the EOS S3 platform consists of the following:

- Data transfer interface via an AHB-to-Wishbone bridge
- SDMA interface
- Interrupt interface

AHB-To-Wishbone Bridge

The AHB-To-Wishbone Bridge provides the means for the S3 platform (M4-F or AP) to access IP within the on-chip programmable logic. Specifically, this interface takes a 32-bit address and data on its AHB port and passes these values to the Wishbone bus. Figure 27-14 illustrates the AHB-to-Wishbone Bridge.

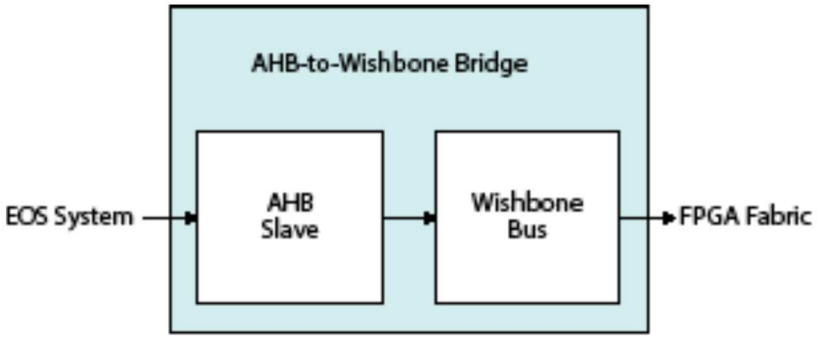

Figure 27-14: AHB-to-Wishbone Bridge

The connection between the AHB Slave and Wishbone Bus supports asynchronous transfers. This allows the on-chip programmable logic-based IP to use a clock frequency appropriate to its operation without the EOS S3 platform losing its ability to communicate with the resources of the IP. For example, most IPs require the use of some type of Start or Stop register bit to control their operations. The asynchronous interface ensures that the EOS S3 platform can still access these registers.

It is important to note that the on-chip programmable logic-based IP does not have the ability to initiate direct transfers to the EOS S3 platform. This is done for two reasons: first, the Wishbone bus cannot support multiple masters (the Wishbone interface only supports Wishbone clients) and second, the AHB Slave interface cannot master the AHB bus.

27.3 FPGA Use

27.3.1 FPGA Configuration Control

In power domain: M4 & FPGA

On boot-up, the FPGA configuration data is downloaded through the SPI_1_Master interface, under control of CFG_CTL, CfgSM, CfgDMA registers.

[discussion to be added]

27.4 FPGA sub-system registers

Mappings of IO functions from the FPGA to pads are controlled by the IOMUX. Refer to the register descriptions for that module.

Debug

Chapter 28. M4-F Development and debug support elements

28.1 Integrated Configurable Debug

The Cortex-M4-F processor can implement a complete hardware debug solution. This provides high system visibility of the processor and memory through a 2-pin SWD port.

QuickLogic has included the debug elements documented in this chapter.

- ITM and DWT functionality is available for trace debugging. For system trace the processor integrates an Instrumentation Trace Macrocell™ (ITM) alongside data watchpoints and a profiling unit.
- To enable simple and cost-effective profiling of the system events these generate, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.
- The Flash Patch and Breakpoint Unit (FPB) provide up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words of the program in the control memory region.

Note: resets affect debug as follows:

POR	Resets entire M4 system (core and debug)
SYSRESETn pin	Reset M4 core only (core but not debug logic)
software reset via write to a register	variable effect - not assessable

28.2 Serial Wire Debug port (SWD)

The EOS S3 features an Integrated Software Debug Interface. Its 2-pin SWD port supports access to the following memory mapped resources:

- M4-F internal registers and memories
- FFE and Sensor Manager memories
- FFE control registers
- On-chip programmable logic memories
- On-chip programmable logic designs through generic AHB bus
- All memory map peripherals such as timers, WDT, SPI Master, etc.
- I^2C Master used for I^2C sensor debug
- Multiplexed dedicated parallel debug interface

Depending on overall EOS S3 system setup and requirements, two different Serial Wire Interfaces can be selected. The external debugger can be connected to either GPIO14/GPIO15 or GPIO44/GPIO45, based on the bootstrap pin GPIO8. The two signals are serial wire clock and serial wire data. The optional serial wire viewer can be selected from several different pins.

28.2.1 Debug Configuration

The recommend external configuration if using ARM DS debugger is shown in Figure 28-1. Other debuggers may have different recommendations.

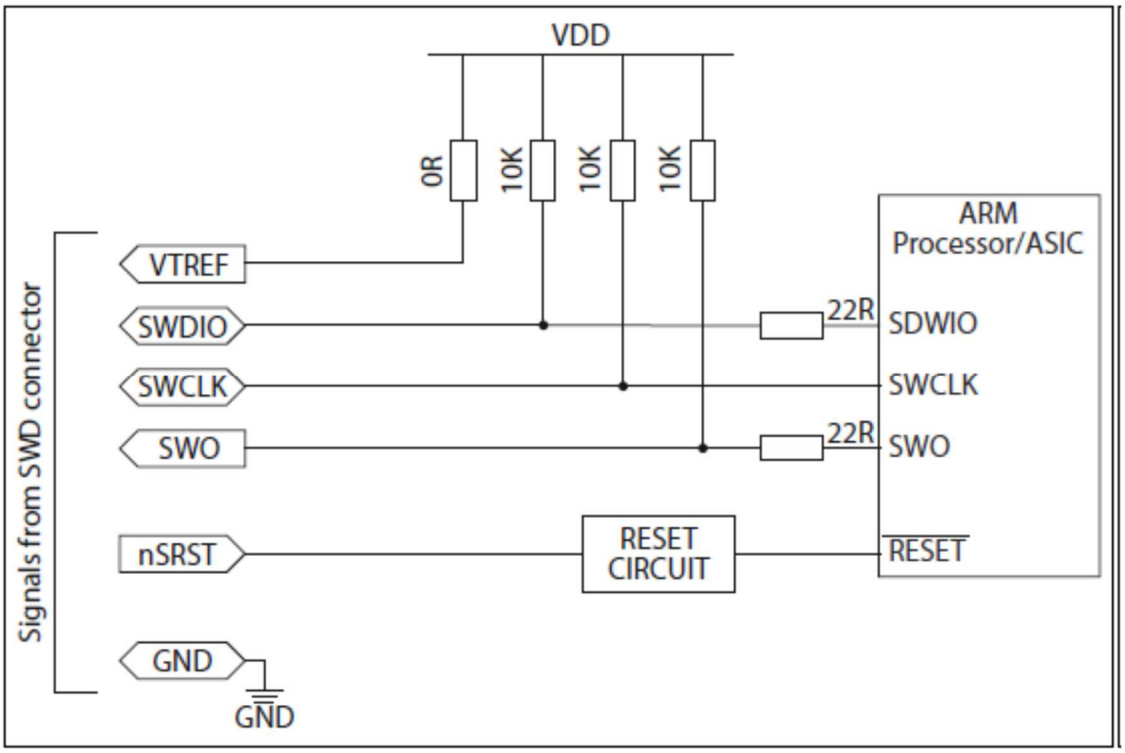

Figure 28-1: Recommended External Debugger Connection for ARM DS Debugger

28.2.2 Debug Bootstrap Configuration

Upon cold boot up, the M4 DAP will be enabled. The M4 DBGEN is register enabled by default, and can be disabled later if not needed. The M4 DAP will only be reset during cold boot up (it is controlled by the POR from APC). The release of the M4 reset depends on the state of bootstrap pin GPIO19. When it is strapped to high, the M4 reset will be released. When strapped to low, the M4 reset release will depend on AP/cfg_sm.

28.2.3 Companion/High-Level O/S Host Configuration (Application Processor in System)

In a system with an application processor present, the application processor must drive bootstrap pin GPIO19 to indicate whether the Debugger is present. In this configuration, GPIO19 is connected to the application processor as part of the SPI interface (the GPIO 19 alternate function is SPIs_MOSI) and it must drive GPIO19 during the deassertion of SYS_RSTn. Driving GPIO19 high will disable debugger support; driving GPIO19 low will enable debugger support.

28.2.4 Host Configuration (the EOS S3 system operating as Host)

In a Host design, bootstrap pin GPIO19 must always be strapped low to allow M4 operation.

28.2.5 DAP accessible ROM table

The Debug Access Port (DAP)-accessible ROM table is defined as follows:

Base address=0xE00F_F000

Table 28-1: ARMv7-M DAP accessible ROM table

Offset	Value	Name	Description
0x000	0xFFF0_F003	ROMSCS	Points to the SCS at 0xE000_E000
0x004	0xFFF0_2003	ROMDWT	Points to the DWT at 0xE000_1000
0x008	0xFFF0_3003	ROMFPB	Points to the FPB at 0xE000_2000
0x00C	0xFFF0_1003	ROMITM	Points to the ITM at 0xE000_0000
0x010	0xFFF4_1003	ROMTPIU	Points to the TPIU at 0xE004_0000
0x014	0xFFF4_2002	rometm	Points to the ETM at 0xE000_1000 (not fitted, thus last nibble = 2)
0x018	0x0000_0000	END	End of table marker of 0x0000_0000

The value in the table (except the last nibble) is a negative pointer relative to ROM table base address of **0xE00F_F000**. The last nibble = 2 if the feature is not fitted; = 3 if the feature is fitted.

For example ROMITM has value of 0xFFF0_1003. Drop the last nibble and take 2's complement:

2's complement of 0xFFF0_1000 = 0x000F_F000

Then subtract from ROM table base address

0xE00F_F000 – 0x000F_F000 = 0xE000_0000

The ITM is located at 0xE000_0000. See section 28.3 for details.

28.2.6 AHB-AP

The AHB-AP is a *Memory Access Port* (MEM-AP) component of the Debug Access Port (DAP) as defined in the *ARM Debug Interface v5 Architecture Specification*[5]. The AHB-AP is a debug access port into the Cortex-M4 system, and provides access to all memory and registers in the system, including processor registers through the SCS. System access is independent of the processor status. SW-DP is used to access the AHB-AP.

The AHB-AP is a master into the Bus Matrix. Transactions are made using the AHB-AP programmer's model, which generates AHB-Lite transactions into the Bus Matrix.

28.3 Instrumentation Trace Macrocell (ITM)

The Instrumentation Trace Macrocell (ITM) provides a memory-mapped register interface to allow applications to write logging / event words to the external Trace Port Interface Unit (TPIU). The ITM also supports control and generation of timestamp information packets. The event words and timestamp information are formed into packets and multiplexed with hardware event packets from the Data Watchpoint and Trace (DWT) block according to the packet protocol described in *ARM®v7-M Architecture Reference Manual*.

28.3.1 ITM Registers

There are 32 stimulus source ports and 3 control/status registers for total of 35 registers in ITM.

Base address = 0xE000_0000

[5] ARM IHI0031x ARM Debug Interface (ADI)

Table 28-2: ITM Registers

Address Offset	Name	Description
0x000	STIM0	Stimulus port register 0
0x004	STIM1	Stimulus port register 1
0x008	STIM2	Stimulus port register 2
0x00C	STIM3	Stimulus port register 3
0x010	STIM4	Stimulus port register 4
0x014	STIM5	Stimulus port register 5
0x018	STIM6	Stimulus port register 6
0x01C	STIM7	Stimulus port register 7
0x020	STIM8	Stimulus port register 8
0x024	STIM9	Stimulus port register 9
0x028	STIM10	Stimulus port register 10
0x02C	STIM11	Stimulus port register 11
0x030	STIM12	Stimulus port register 12
0x034	STIM13	Stimulus port register 13
0x038	STIM14	Stimulus port register 14
0x03C	STIM15	Stimulus port register 15
0x040	STIM16	Stimulus port register 16
0x044	STIM17	Stimulus port register 17
0x048	STIM18	Stimulus port register 18
0x04C	STIM19	Stimulus port register 19
0x050	STIM20	Stimulus port register 20
0x054	STIM21	Stimulus port register 21
0x058	STIM22	Stimulus port register 22
0x05C	STIM23	Stimulus port register 23
0x060	STIM24	Stimulus port register 24
0x064	STIM25	Stimulus port register 25
0x068	STIM26	Stimulus port register 26
0x06C	STIM27	Stimulus port register 27
0x070	STIM28	Stimulus port register 28
0x074	STIM29	Stimulus port register 29
0x078	STIM30	Stimulus port register 30
0x07C	STIM31	Stimulus port register 31
0XE00	TER	Trace enable register
0xE40	TPR	Trace privilege register
0xE80	TCR	Trace control register

Refer to *ARM®v7-M Architecture Reference Manual* for more details.

28.4 Data Watchpoint and Trace (DWT)

The Data Watchpoint and Trace (DWT) component can provide the following features:

- PC sampling, two forms are supported:
  - PC sample trace output as a result of a cycle count event or DWT function match
  - External PC sampling using a PC sample register.
- Comparators to support:
  - Watchpoints – enters Debug state or takes a DebugMonitor exception
  - Data tracing
  - Signaling for use with an external resource, for example an ETM (not implemented)
  - Cycle count matching.
- Exception trace support
- Profiling counter support.

28.4.1 Registers

There are 32 registers in the DWT.

Base address = 0xE000_1000

Table 28-3: DWT Registers

Offset	Name/Field	Description
0x000	DWT_CTRL	Control Register
0x004	CycCnt	Cycle Count Register
0x008	CPICnt	CPI Count Register
0x00C	ExcCnt	Exception Overhead Count Register
0x010	SleepCnt	Sleep Count Register
0x014	LSUCnt	LSU Count Register
0x018	FoldCnt	Folded-instruction Count Register
0x01C	PCSR	Program Counter Sample Register
0x020	COMP0	Comparator Register0
0x024	MASK0	Mask Register0
0x028	FUNCTION0	Function Register0
0x02C	–	Reserved
0x030	COMP1	Comparator Register1
0x034	MASK1	Mask Register1
0x038	FUNCTION1	Function Register1
0x03C	–	Reserved
0x040	COMP2	Comparator Register2
0x044	MASK2	Mask Register2
0x048	FUNCTION2	Function Register2
0x04C	–	Reserved
0x050	COMP3	Comparator Register3
0x054	MASK3	Mask Register3
0x058	FUNCTION3	Function Register3
0xFD0	DWTPERIPHID4	Peripheral identification register 4
0xFD4	DWTPERIPHID5	Peripheral identification register 5
0xFD8	DWTPERIPHID6	Peripheral identification register 6
0xFDC	DWTPERIPHID7	Peripheral identification register 7

Offset	Name/Field	Description
0xFE0	DWTPERIPHID0	Peripheral identification register 0
0xFE4	DWTPERIPHID1	Peripheral identification register 1
0xFE8	DWTPERIPHID2	Peripheral identification register 2
0xFEC	DWTPERIPHID3	Peripheral identification register 3
0xFF0	DWTCID0	Component identification register 0
0xFF4	DWTCID1	Component identification register 1
0xFF8	DWTCID2	Component identification register 2
0xFFC	DWTCID3	Component identification register 3

Refer to *ARM® v7-M Architecture Reference Manual* for more details.

28.5 Flash Patch and Breakpoint Unit (FPB)

ARM M4 Flash Patch and Breakpoint (FPB) capability is a set of address matching tags, which reroute accesses into flash to a special part of SRAM. This permits patching flash locations for breakpointing and quick fixes or changes.

The Flash Patch and Breakpoint component can provide support for:

- Remapping of specific literal locations from the Code region of system memory to an address in the SRAM region.
- Remapping of specific instruction locations from the Code region of system memory to an address in the SRAM region.
- Breakpoint functionality for instruction fetches.

28.5.1 FPB Registers

There are 10 registers in the FPB unit.

Base address = 0xE000_2000

Table 28-4: FPB Registers

Address Offset	Name	Description
0x000	FPB_CTRL	Control Register
0x004	FPB_REMAP	Remap Register
0x008	FPB_COMP0	Comparator 0 Register
0x00C	FPB_COMP1	Comparator 1 Register
0x010	FPB_COMP2	Comparator 2 Register
0x014	FPB_COMP3	Comparator 3 Register
0x018	FPB_COMP4	Comparator 4 Register
0x01C	FPB_COMP5	Comparator 5 Register
0x020	FPB_COMP6	Comparator 6 Register
0x024	FPB_COMP7	Comparator 7 Register

Table 28-5: FPB Control Register Bit Description

0x000	FPB_CTRL	31:0	R/W/C	Default	Description
	–	31:15	RO	–	Reserved
	NUMCODE64	14:12	RO	X	The most significant 3 bits of NUM_CODE
	NUMLIT	11:8	RO	8	The number of literal address comparators supported: 8
	NUMCODE30	7:4	RO	X	The least significant 4 bits of NUM_CODE
	–	3:2	RO	–	Reserved
	KEY	1	RW	0	On any write to FPB_CTRL, the FPB unit ignores the write unless this bit is 1
	ENABLE	0	RW	0	Enable bit for the FPB

Table 28-6: FPB Remap Register Bit Description

0x004	FPB_REMAP	31:0	R/W/C	Default	Description
	–	31:30	RO	–	Reserved
	RMPSPT	29	RO	X	Indicates whether the FPB unit supports flash patch remap
	REMAP	28:5	RW	X	Remap address
	–	4:0	RO	–	Reserved

Table 28-7: FPB Compare 0 Register Bit Description

0x008	FPB_COMP0	31:0	R/W/C	Default	Description
	–	31:30	RW	–	Reserved
	REPLACE	29	RO	X	Defines the behavior when the COMP address is matched for instr addr comparator
	COMP	28:5	RW	X	Address to compare
	ENABLE	4:0	RO	–	Enable bit for this comparator

FPB_COMP1. FPB_COMP7 have identical register bit description as FPB_COMP0.

Refer to *ARM® v7-M Architecture Reference Manual*[6] for more details.

28.6 Trace Port Interface Unit (TPIU)

The Cortex-M4 Trace Port Interface Unit (TPIU) is a component that acts as a bridge between the on-chip trace data from the Instrumentation Trace Macrocell (ITM) to a data stream. The TPIU encapsulates IDs where required, and the data stream is then captured by an external Trace Port Analyzer (TPA). The Cortex-M4 TPIU is specially designed for low-cost debug. It is a special version (subset) of the CoreSight TPIU.

28.6.1 TPIU Registers

There are 10 registers in TPIU.

Base address = 0xE004_0000

Table 28-8: TPIU Registers

Address Offset	Name	Description
0x000		
0x004		
0x008		
0x00C		
0x010		
0x014		
0x018		
0x01C		
0x020		
0x024		

[6] ARM DDI0403C *ARM v7-M Architecture Reference Manual*

Misc. Resources

Chapter 29. eFuse

In power domain: eFUSE

The eFuse module is used for factory configuration including LDO trimming. The module contains user-readable registers.

The EOS S3 System provides an on-chip eFuse allowing for storing fuse data for analog trim and other bits. The eFuse block has 128 bits for programming, internally divided into 2 banks of 64 bits each. Some bits are reserved for analog trim bits.

The eFuse can only be programmed by QuickLogic during manufacturing.

During boot up, the eFuse controller will automatically read the eFuse bits and store all 128 eFuse data bits into registers. The M4 can also power-on eFuse and write a kick off register bit to read the eFuse bits again. The analog trim bits inside eFuse will be latched into the always-on logic. The other eFuse bits can be accessed by reading the eFuse registers.

The eFuse block is contained in its own power domain. It is recommended that after bootup, the eFuse power domain be shut down to save power.

29.1.1 eFuse Registers

Offset	Name/Field	Bits	Type	Default	Description
0x000	LDO30_VT_BITS	31:0	RO		
	RESERVED	31:5	RO	0x0	Reserved
	LDO30_VT_BITS	4:0	RO	0x11	LDO30 output voltage prog bits
0x004	LDO50_VT_BITS	31:0	RO		
	RESERVED	31:5	RO	0x0	Reserved
	LDO50_VT_BITS	4:0	RO	0x11	LDO50 output voltage prog bits
0x008	APC_TRIM_BITS	31:0	RO		
	APC_CU_TRIM_BITS	12:10	RO	0x0	APC current trim bits
	APC_TM_TRIM_BITS	7:5	RO	0x0	APC TEMP prog bits
	APC_VT_TRIM_BITS	4:0	RO	0x0	APC voltage Trim bits
0x00c	M4_SRAM_CFG	31:0	RO		
	M4_SRAM_CFG	1:0	RO	0x0	M4 SRAM size configuration 00: M4 SRAM size is 512KB 01: M4 SRAM size is 384KB 11: M4 SRAM size is 256KB
0x010	LPSD_CFG	31:0	RO		
	VOICE_BUNDLE	16	RO	0x0	voice bundle bit

Offset	Name/Field	Bits	Type	Default	Description
	LPSD_DISABLE	0	RO	0x0	LPSD HW enable bit 1=disable 0=enable
0x14	RESERVED	31:0	RO		
0x18	RESERVED	31:0	RO		
0x1c	RESERVED	31:0	RO		
0x20	RESERVED	31:0	RO		
0x24	SW_RD_KICKOFF	31:0	WO		
	Reserved	31:1	WO	0x0	Reserved
	SW_RD_KICKOFF	0	WO	0x0	SW kick off read eFuse Write 1 to generate a pulse to kick off efuse read
0x28	EFUSE_RD_DONE	31:0	RO		
	TESTER_BITS	3:1	RO	0x0	Tester bits
	EFUSE_RD_DONE	0	RO	0x0	eFuse read is done 1 = done 0 = read is not finished

System Considerations

Chapter 30. System configuration for start-up

This chapter covers basic system configuration items needed at start-up.

Behavior of the S3 after reset depends on which system design mode it is configured for. This behavior is discussed in following sections.

30.1 System design modes

An S3 board design can operate in one of two possible system mode configurations, as were shown in Section 1.2. These modes are:

1. Companion Mode

In this design, the S3 is used as a slave to the application processor. The S3 uses the SPI_Slave interface for communications with the application processor. The host configures the S3 for use.

2. Host Mode

In this design the S3 is stand-alone, and it uses the SPI_1_Master interface to load code and data into memory from an external flash memory.

30.2 Selecting system design mode on boot

The state of bootstrap pin GPIO_20 at reset time configures whether the S3 enters Companion Mode or Host Mode upon reset.

Table 30-1: Configuration for Companion Mode or Host Mode

			Pin functions		
GPIO_20	Mode	Source port for bootload	GPIO pin	SPI I/O Function on GPIO pin	I/O Direction
Pulled Down	Host	SPI_Master for flash	GPIO_34 GPIO_36 GPIO_38 GPIO_39	SCLK MISO MOSI SS1n	Out In Out Out
Pulled Up	Companion	SPI_Slave to AP	GPIO_16 GPIO_17 GPIO_19 GPIO_20	SCLK MISO MOSI CSn	In Out In In

30.3 Configuration for debugging

The EOS S3 supports Cortex M4 system debugging through use of a serial wire debug connection. This involves alternate use of certain GPIO pins. These pins can be assigned to the function by means of the status of two other GPIO pins on startup.

The S3 Data Sheet contains significant additional relevant information in its section 'Debugger Bootstrap Configurations'. Please refer to details there

30.3.1 Configuring to identify that M4 Serial Wire Debugger is present

Configure bootstrap pin GPIO_19 to identify to the S3 whether SWD is present.

Table 30-2: Configuration for use of Serial Wire Debugger

GPIO_19	Debugger State
Pulled Down	Not Present
Pulled Up	Present

30.3.2 Configuring M4 Serial Wire Debug Port pin assignment

The M4 Serial Wire Debug port signals, SW_CLK and SW_IO, are assigned to different package pins depending on state of bootstrap pin GPIO_8.

Table 30-3: M4 Serial Wire Debug Port Bootstrap Configuration

GPIO_8 State	Serial Wire Debug Port Signal	
	SW_CLK	SW_IO
Pulled Down (default)	GPIO_14	GPIO_15
Pulled Up	GPIO_45	GPIO_44

30.4 Configuring clock oscillators

The S3 master clock oscillator can be one of three sources. For details, see discussion in the Clocking chapter.

30.4.1 Boot-time configuration for use of internal HS_Osc or external driver for HS_Osc

Selecting use of either internal HS_Osc or external HS_Osc source is configured by bootstrap pins GPIO_8 and GPIO_9.

Table 30-4: Internal/External HSO Configuration

Pin State		HSO Configuration
GPIO_8	GPIO_9	
Pulled Down (default)	Pulled Down (default)	Internal HSO
	Pulled Up	Internal HSO
Pulled Up	Pulled Down (default)	Internal HSO
	Pulled Up	use External HSO (FCLK) from GPIO_6

The configuration is latched upon de-assertion of the SYS_RSTn pin.

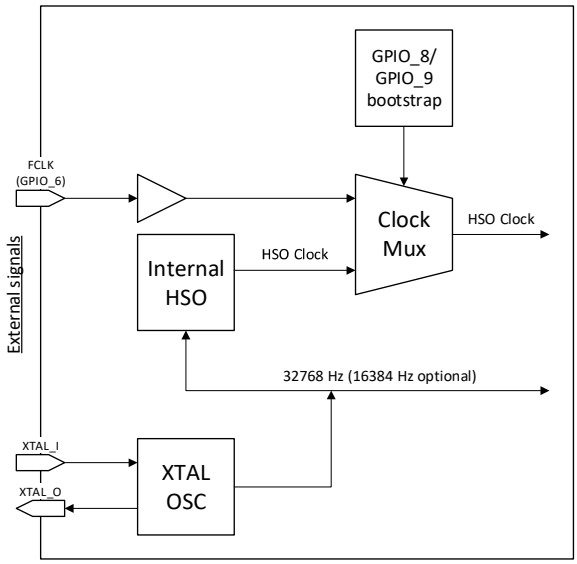

Figure 30-1: Selecting either internal or external High Speed Oscillator (HSO)

30.5 I/O configuration

The EOS device I/O configuration options are selected by pulling special bootstrap pins high or low, which are latched upon de-assertion of the SYS_RSTn pin.

See device default IO chapter for details.

30.6 LDO configuration

LDO architectural configuration is hardwired through the board design. However, voltage tweaks for LDOs are supported via registers settings.

30.7 FPGA configuration

[To be added]

Chapter 31. Reset, start-up, and interrupts

This chapter covers system reset and general handling of interrupts. For discussion of interrupts and the NVIC and the WIC hardware modules for interrupts see:

- Wakeup Interrupt Controller
- Nested Vectored Interrupt Controller (NVIC) and ARM documentation for the NVIC.

31.1 Reset

There are multiple reset sources: hardware (2) and software (1).

- Power-ON-Reset (POR) - automatically performed by the S3 after power is applied
- system reset through SYS_RSTn pin - requires physical signal to the pin
- software reset - performed by a write to a register

After booting up, the firmware can program to block the system reset (SYS_RSTn pin) and treat it as one of the interrupt sources.

The following diagram illustrates power sequence waveform.

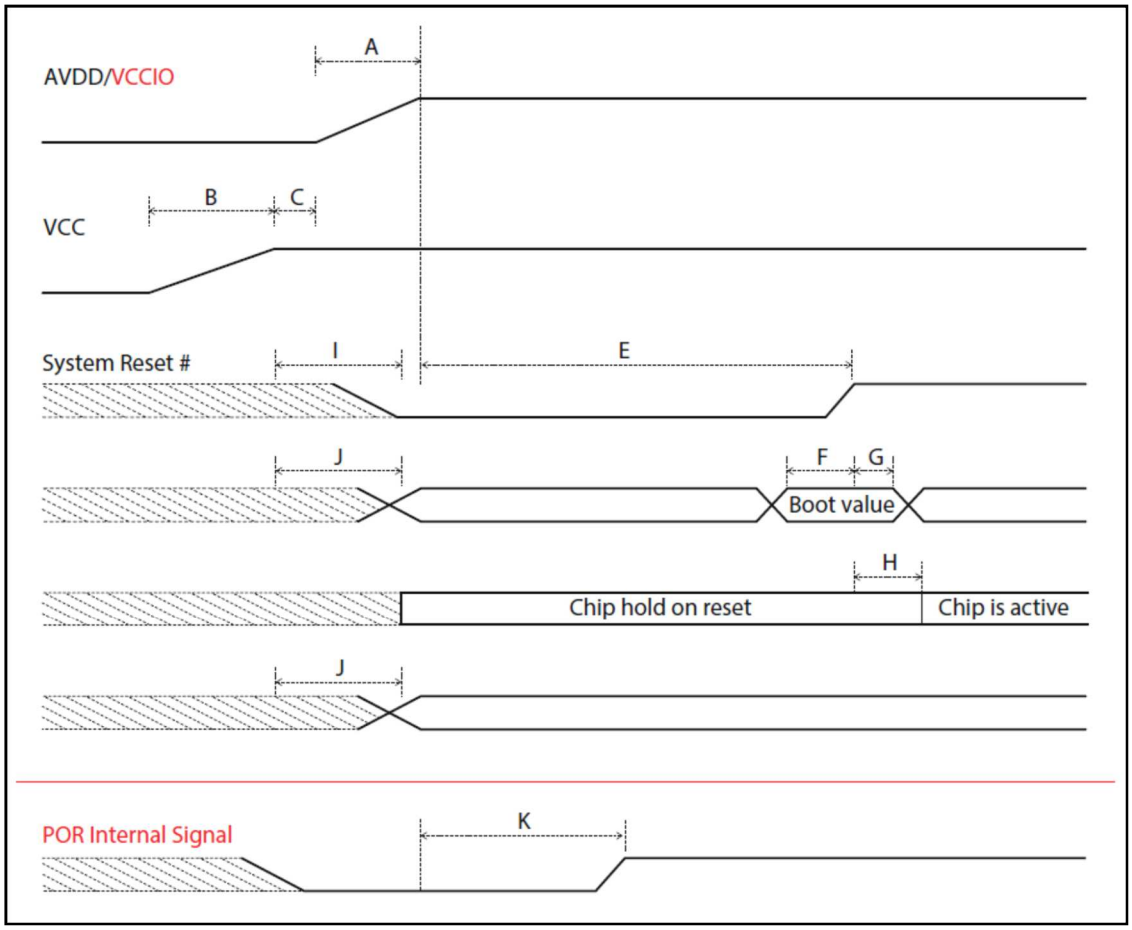

Figure 31-1: Power-up Sequence

For more details see Datasheet for timing details.

31.2 Startup flow for Companion Mode

The S3 on startup examines a GPIO pin's configuration to tell the whether the S3 is a slave in a system with an AP. If so, the S3 prepares the SPI_Slave interface for use to load initial configuration from the host system. It is then the host's responsibility to control and direct this activity.

31.3 Startup flow for Host Mode

31.3.1 Host mode boot load from external flash

For systems configured for Host Mode operation. The Configuration Manager subsystem handles initial program load automatically, by DMA transfer of flash memory data obtained through the SPI_Master interface. For details see the chapter discussing the Configuration Manager and its Configuration State Machine and ConfigDMA modules.

The memory image loads starting from 0x0000_0000 as required by the ARM_v7.

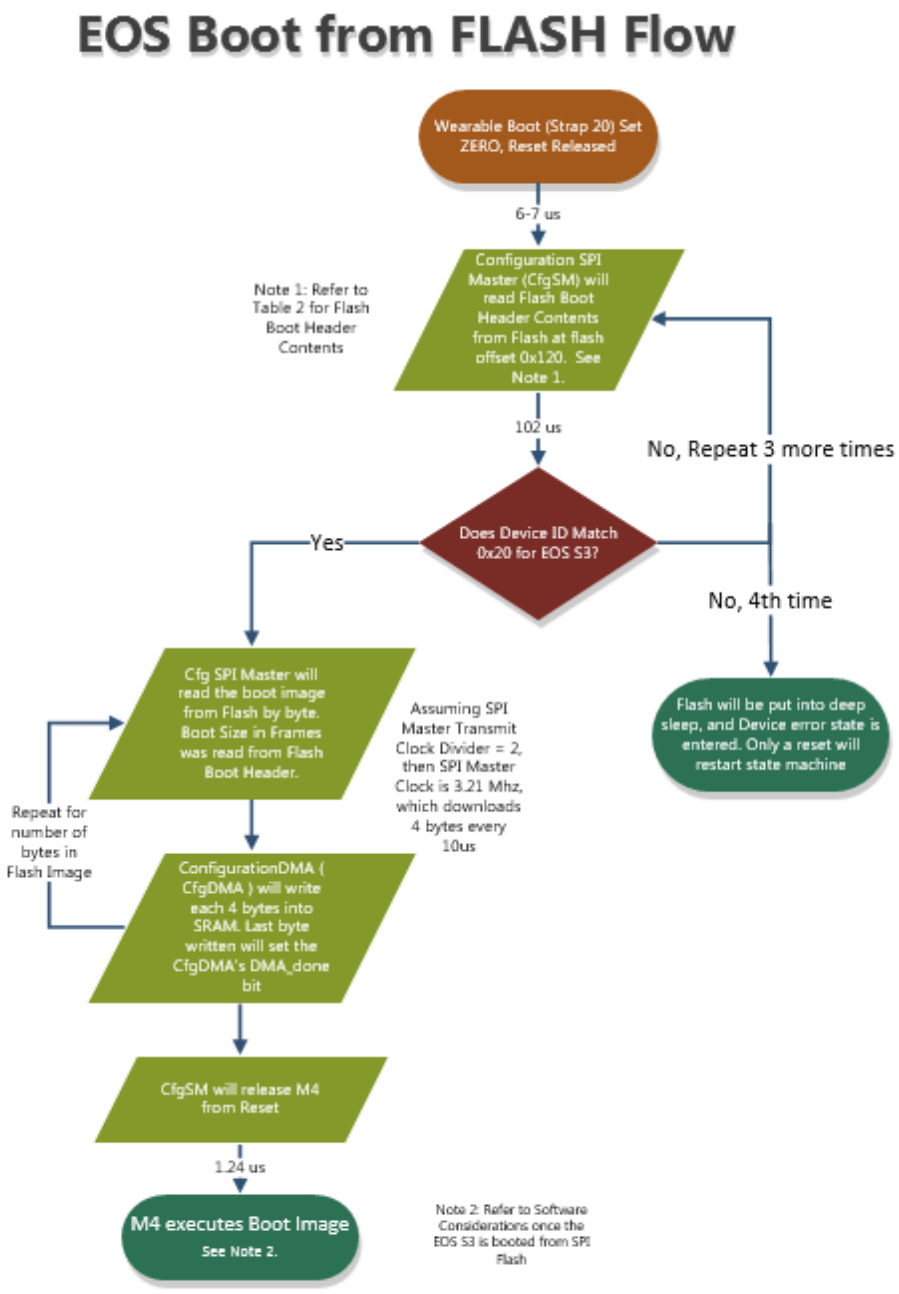

Figure 31-2: Boot from Flash Flow

31.4 Interrupts

Interrupts generated by S3 sub-system events can be routed to an Applications Processor (AP), or to the M4 in a standalone design. When routed to M4 - All interrupts to M4 will be connected to ME's NVIC. There are 2 levels of interrupt masking and clearing. One will be at the source of interrupts, and the other at the top-level interrupt controller. When routed to AP - The interrupt mechanism for the AP is the same as M4's but it has a different mask. All interrupt sources are combined to single combined interrupt before being sent to the AP.

31.4.1 Functions of the NVIC and WIC interrupt controllers

The NVIC is the interrupt controller of the M4. It is in the same power domain and hence is active only when the M4 is active. However, the NVIC can be woken up by means of the Wake-Up Interrupt controller which receives interrupts from important peripherals. For more details see NVIC section for more details.

The WIC is a QuickLogic system component whose purpose is to be on when the system is in deep sleep mode, and to receive interrupts from peripherals and then command the PMU to wake up the M4's NVIC.

31.4.2 Interrupt Sources

Interrupts from each sub-system's functional blocks are combined into one interrupt for each sub-system.

The following table summarizes interrupt sources.

Table 31-1: Interrupt Sources

Source	Description
Top – sensor / GPIO	Total of 8 pins
M4 sub-system	FPU, bus timeout, UART, M4 Timer (also known as Timer1), Watchdog Timer (WDT), SRAM
FFE sub-system	Total of 8 FFE messages, total of 16 FFE sub-system interrupts
A0	AP reboot, SYS_RSTn, ADC, PMU timer, software, LDO power good
A1	Configuration DMA (CfgDMA), SPI Master
Voice sub-system	The voice / audio processing sub-system has these possible interrupt sources: LPSD Voice Detect, DMIC Voice Detect, DMIC Voice Off, LPSD Voice Off, DMAC0 Block Done, DMAC0 Buf Done, DMAC1 Block Done, DMAC1 Buf Done, AP PDM Clock ON, AP PDM Clock OFF, FIFO_TRIG_INT, I2S Interrupt
SDMA	SDMA done, SDMA error
PKFB	The Packet FIFO Bank (PKFB) handles FIFO overflow, underflow, count threshold, access during sleep, and collision
FPGA Fabric	Total of 4 outputs from FPGA fabric can be used.

Chapter 32. Power Modes

This chapter describes power modes in the S3.

32.1 Introduction

The S3 provides main power modes and low power modes.

Main power modes:

1. Shutdown (all power off) also known as SD
2. Run (power on)

Low power modes:

Low power modes reduce system power consumption and are achieved in various ways. The S3 implements several low power modes to reduce static, dynamic, leakage power consumed by the M4 and the CPU sub-system and other subsystems.

Low power modes in the S3 include:

Sleep (As defined for the M4)	
Deep Sleep (DS)	(As defined for the M4) (As defined for SRAM)

The set of power modes supported in a given part of the architecture will vary by sub-system and module. Every sub-system and every module have their own configuration, status, entry and exit trigger conditions.

Dynamic power consumption may be reduced through clock gating. See **Section** 11.2.1 Clock Gating for more details. While clock gating reduces dynamic power consumption for circuits when they are not used, a static circuit consumes some power through leakage current.

A power domain group of circuits may be turned on/off by firmware or hardware event. The VDD power supply to the power domain is removed by a "head switch" when the feature is not used. This is defined as shutdown (SD). This significantly reduces leakage current. If the memory and flip flop state must be retained, most power domains may be placed in a deep sleep (DS) mode to reduce power consumption. In the DS mode, most of the logic and flip-flops (FFs) are turned off while SRAM memory contents are retained. Some SRAMs also support light sleep mode (see **Section 32.4.2** for more details). These modes are collectively referred to as *low power* modes. The top level modules that do not have head switch belong to Always On (AON) power domain.

The Power Management Unit (PMU) controls sequences to enter or exit low power modes. See PMU section for more details.

32.2 Power Modes in the S3

32.2.1 Comparison of Power Modes

Table 32-1: Comparison of Power Modes

Modes:	No power (OFF)	Low Power		Run (normal)
	Shutdown (SD)	Deep Sleep (DS)	Light Sleep	
Power	Off	Partially Off	On	On
Data Retention	Lost	Retained	Retained	Normal
Power Consumption	Lowest	Lower	Low	Normal
Entry / Exit Time	Slowest	Faster than shutdown	Faster than deep sleep	N/A
Access Time	Slow†	Slow†	Slower than run	Normal

Note:

† When a memory or module is in deep sleep or shutdown, it must be woken up; access time is slow.

An SRAM in a sub-system may be placed in low power mode independent of the other circuits in the sub-system. The following diagram illustrates separate power domains for logic and individual SRAM blocks within a given sub-system. When the sub-system is running with all SRAMs turned on, all power mode signals (all DS and SD signals) are de-asserted. That is, all the switches are closed and power is supplied to all circuits within the sub-system. When Deep Sleep (DS) for the sub-system is asserted, the head switch SW1 is opened, and power to the majority of logic section is removed, however, VDDSD provides some power to logic and VDDSRAM provides power to all SRAMS. If SRAM_0_SD is asserted, the head switch inside SRAM_0 block will turn off all power to the SRAM_0 block without affecting power mode of rest of the sub-system.

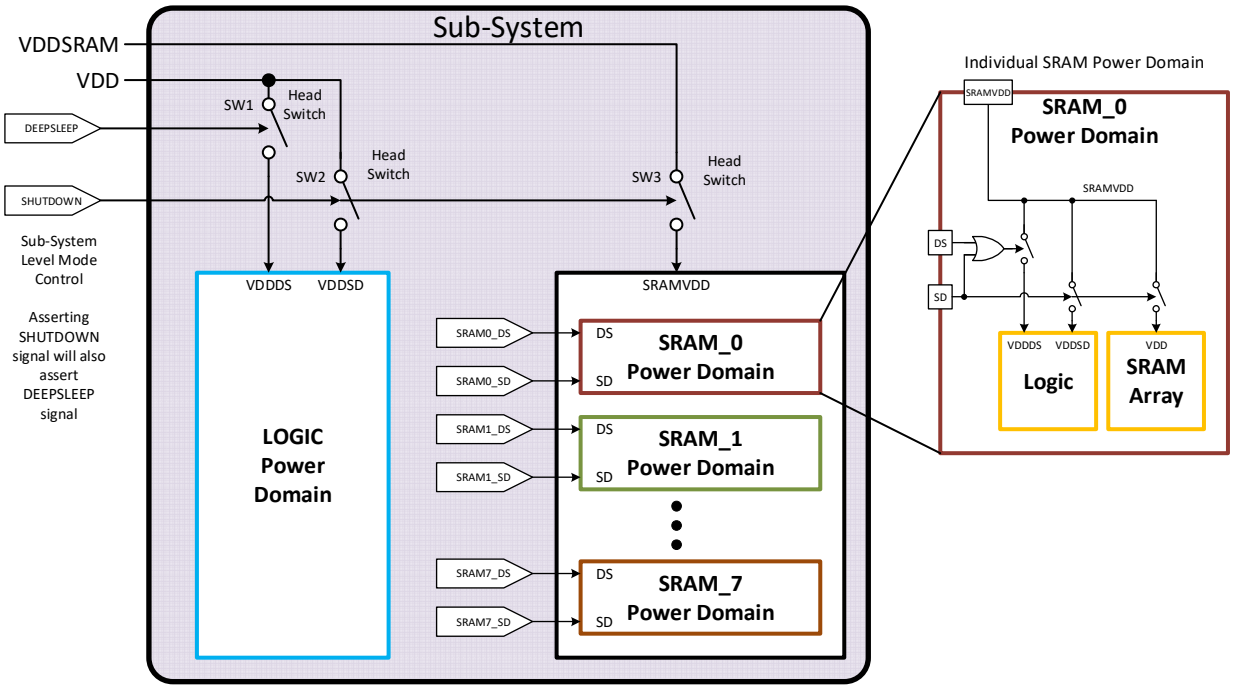

Figure 32-1: Logic and SRAM Power Domains within a Sub-System

Note:

When the shutdown (SD) is asserted at the sub-system level, the PMU will also assert the Deep Sleep (DS) signal at the same time (effectively an OR logic). In addition, head switch SW3 is opened, removing power to SRAM (this gives lower power consumption than simply asserting SD at the individual SRAM level).

32.2.2 Methods for entering Low Power Modes

Each power domain may be placed in a low power mode (shutdown, deep sleep, or light sleep, depending on which are supported by the domain) by M4 firmware or by the TLC writing to a control bit. (See also discussion of Interfacing with an AP).

- See TLC section for details on TLC accessing M4 memory/register space.
- See description on entering low power mode in respective sections.
- Some power domains can be put in a low power mode by PMU hardware driven by an event.
- Some domains only support low power state through shutdown.

32.2.2.1 Software sources for initiating low-power modes

The following table summarizes the software source initiating the low power mode entry for each power domain. Software sources are not maskable.

See also: relevant PMU register fields.

Table 32-2: Software triggering of Low Power Modes

Power Domain	Software Source to Enter Low Power Mode	Alternate Software Source
M4	WFI/WFE ARM instruction code	
FFE	DS or SD: PMU.FFE_FB_PF_SOFTWARE_PD[0] Based on: FFE_POWER_MODE_CFG	
FB	DS or SD: PMU.FFE_FB_PF_SOFTWARE_PD[1] Based on: FB_POWER_MODE_CFG	
PF	DS or SD: PMU.FFE_FB_PF_SOFTWARE_PD[2] Based on: PF_POWER_MODE_CFG	
AD0	SD: PMU.AUDIO_SOFTWARE_PD[0]	
AD1	SD: PMU.AUDIO_SOFTWARE_PD[1]	
AD2	SD: PMU.AUDIO_SOFTWARE_PD[2]	
AD3	SD: PMU.AUDIO_SOFTWARE_PD[3]	
AD4	SD: PMU.AUDIO_SOFTWARE_PD[4]	
AD5	SD: PMU.AUDIO_SOFTWARE_PD[5]	
A1	DS or SD: PMU.MISC_SOFTWARE_PD[6] Based on: A1_POWER_MODE_CFG	
eFuse	SD: PMU.MISC_SOFTWARE_PD[2]	
I2S	SD: PMU.MISC_SOFTWARE_PD[5]	
SDMA	DS or SD: PMU.MISC_SOFTWARE_PD[0] Based on: SDMA_POWER_MODE_CFG	
M4SRAM[15:12]	SD: PMU.M4SRAM_SOFTWARE_PD[15:12]	DS: M4_MEM_CTRL_0 SD: M4_MEM_CTRL_1
M4SRAM[11:0]	SD: PMU.M4SRAM_SOFTWARE_PD[11:0]	DS: M4_MEM_CTRL_0 SD: M4_MEM_CTRL_1
PF SRAM	SD: PMU. PF_MEM_CTRL_1	
FFE SRAM	DS: PMU.FFE_MEM_CTRL_0 SD: PMU.FFE_MEM_CTRL_1	
AD1 (Left) SRAM	DS: PMU.AUDIO_MEM_CTRL_0 SD: PMU.AUDIO_MEM_CTRL_1	
AD2 (Right) SRAM	DS: PMU.AUDIO_MEM_CTRL_0 SD: PMU.AUDIO_MEM_CTRL_1	
SDMA SRAM	DS: PMU.SDMA_MEM_CTRL_0 SD: PMU.SDMA_MEM_CTRL_1	

32.2.2.2 Hardware sources for initiating low-power modes

The following table summarizes for each power domain the hardware sources that can initiate entry to low power mode entry. Some hardware sources are maskable.

Table 32-3: Entering Low Power Modes by Hardware

Power Domain	Hardware Source to Enter Low Power Mode	Maskable?
M4	–	–
FFE	FFE BUSY signal falling edge (See **Section 0**) FB Interface falling edge (See **Section 32.7**)	Yes
FB	FB Interface falling edge (See **Section 32.7**)	Yes
PF	–	–
AD0	–	–
AD1	–	–
AD2	–	–
AD3	–	–
AD4	–	–
AD5	–	–
A1	–	–
eFuse	–	–
I2S	–	–
SDMA	When SDMA is in idle state longer than SDMA_BRIDGE.SDMA_PWRDN_CNT (See **Section 32.9**)	Yes
M4SRAM[15:12]	–	–
M4SRAM[11:0]	M4 SRAM will enter DS mode when M4 enters SD mode † if MEMORY_POWER_DOWN_CONTROL[11:0] = 1	DS: Yes
PF SRAM	PF SRAM will enter DS mode when PF enters low-power mode if MEMORY_POWER_DOWN_CONTROL[17] = 1	DS: Yes SD: No
FFE SRAM	FFE SRAM will enter DS mode when FFE enters low-power mode † if MEMORY_POWER_DOWN_CONTROL[16] = 1	DS: Yes SD: No
AD1 (Left) SRAM	AD1 SRAM will enter DS mode when AD1 enters shutdown mode	No
AD2 (Right) SRAM	AD2 SRAM will enter DS mode when AD2 enters shutdown mode	No
SDMA SRAM	SDMA SRAM will enter DS mode when SDMA enters low-power mode † if MEMORY_POWER_DOWN_CONTROL[18] = 1	Yes

† See MEMORY_POWER_DOWN_CONTROL register description for more details.

32.2.3 Methods for exiting Low Power Modes

Each power domain may be taken out of a low power mode by M4 firmware or by the TLC writing to a control bit. Some power domains exit low power mode by PMU hardware through an event.

32.2.3.1 Software sources for exiting low power mode

The following table summarizes the software source initiating the mode change for each power domain.

Table 32-4: Software sources that can cause exit from low power mode

Power Domain	Software Source to Exit Low Power Mode
M4	–
FFE	PMU.FFE_FB_PF_SOFTWARE_WU[0]
FB	PMU.FFE_FB_PF_SOFTWARE_WU[1]
PF	PMU.FFE_FB_PF_SOFTWARE_WU[2]
AD0	PMU.AUDIO_SOFTWARE_WU[0]
AD1	PMU.AUDIO_SOFTWARE_WU[1]
AD2	PMU.AUDIO_SOFTWARE_WU[2]
AD3	PMU.AUDIO_SOFTWARE_WU[3]
AD4	PMU.AUDIO_SOFTWARE_WU[4]
AD5	PMU.AUDIO_SOFTWARE_WU[5]
A1	PMU.MISC_SOFTWARE_WU[6]
eFuse	PMU.MISC_SOFTWARE_WU[2]
I2S slave	PMU.MISC_SOFTWARE_WU[5]
SDMA	PMU.MISC_SOFTWARE_WU[0]
M4SRAM	PMU.M4SRAM_SOFTWARE_WU[15:0]
PF SRAM	PMU.PF_MEM_CTRL_1
FFE SRAM	PMU.FFE_MEM_CTRL_0 / PMU.FFE_MEM_CTRL_1
AD1 (Left) SRAM	PMU.AUDIO_MEM_CTRL_0 / PMU.AUDIO_MEM_CTRL_1
AD2 (Right) SRAM	PMU.AUDIO_MEM_CTRL_0 / PMU.AUDIO_MEM_CTRL_1
SDMA SRAM	PMU.SDMA_MEM_CTRL_0 / PMU.SDMA_MEM_CTRL_1

32.2.3.2 Hardware sources for exiting low power mode

The following table summarizes for each power domain. These are the hardware sources that can initiate the mode change.

Table 32-5: Hardware sources that can cause exit from low power mode

Power Domain	Hardware Source to Exit Low Power Mode	Maskable?
M4	Interrupts (WIC)	Yes
FFE	Sensor/GPIO INTs will be used to wake the FFE sampling timer, but will not be used to wake up FFE directly	Yes
FB	Sensor/GPIO Interrupts or FFE Sampling Timer	Yes Yes
PF	–	
AD0	LPSD or GPIO/LPSD	Yes Yes
AD1	LPSD or GPIO/LPSD	Yes Yes
AD2	LPSD or GPIO/LPSD	Yes Yes
AD3	LPSD or GPIO/LPSD	Yes Yes
AD4	LPSD or GPIO/LPSD	Yes Yes
AD5	LPSD or GPIO/LPSD	Yes Yes
A1	–	–
eFuse	–	–
I2S	–	–
SDMA	–	–
M4SRAM		–
PF SRAM	PF SRAM will exit DS mode when PF exits low-power mode	No
FFE SRAM	FFE SRAM will exit DS mode when FFE exits low-power mode	No
AD1 SRAM	AD1 SRAM will exit DS mode when AD1 exits low-power mode	No
AD2 SRAM	AD2 SRAM will exit DS mode when AD2 exits low-power mode	No
SDMA SRAM	SDMA SRAM will exit DS mode when SDMA exits low-power mode	No

32.4 M4 Sub-System Low Power Modes

The Cortex-M4F and associated blocks, as shown below, are in a single power domain. In addition, each 32KB SRAM block has its own power domain. Note that power to an Always-ON (AON) power domain is always supplied.

Power domains are: AON (128KB – 4x32KB), M4, M4S0, M4S1, M4S2, M4S3, M4S4, M4S5, M4S6, M4S7, M4S8, M4S9, M4S10, and M4S11.

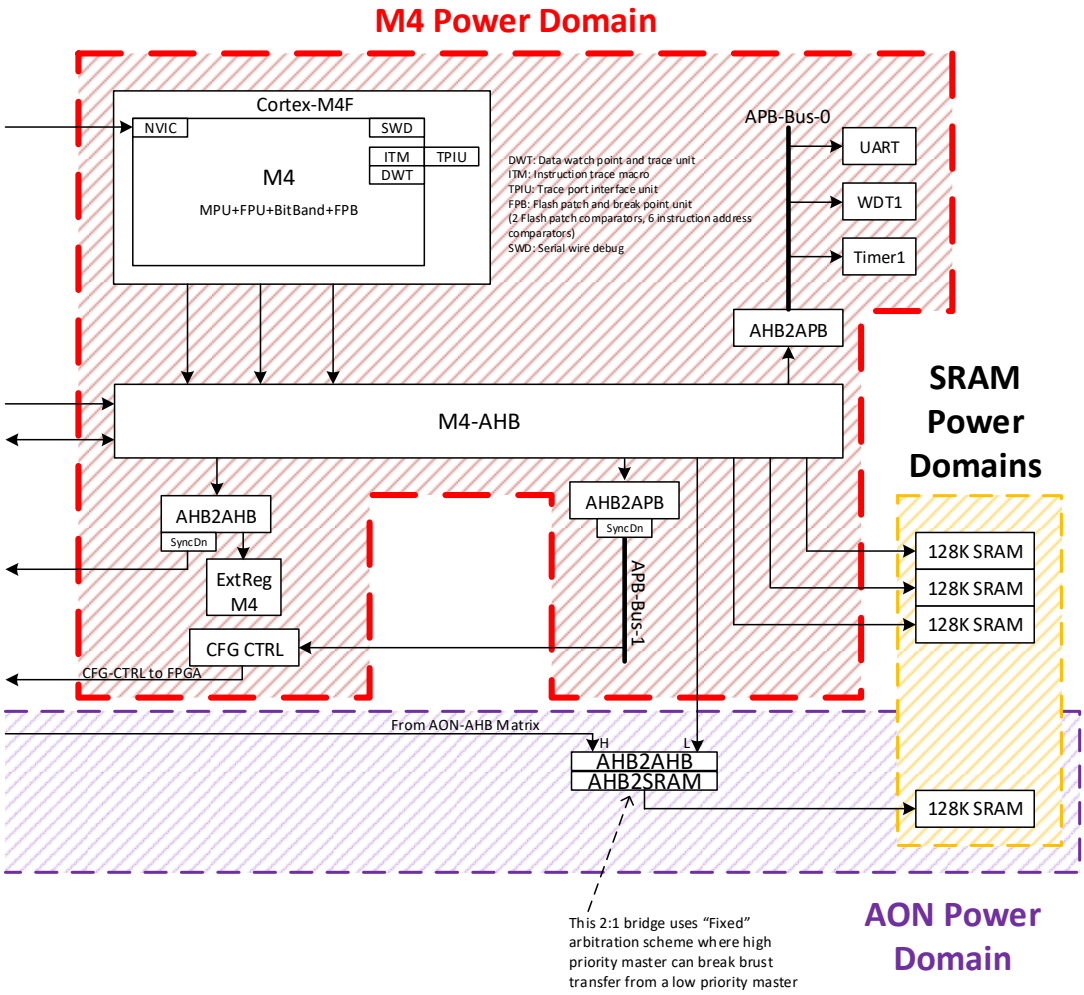

Figure 32-2: M4 and SRAM Power Domains

32.4.1 M4-F Sleep Modes

The Cortex-M4-F CPU and associated modules in the same power domain as described above may be in one of these states:

1. M4 level
   a. SLEEP – most clocks are gated off, small part of the processor is running
   b. DEEP SLEEP – all clocks gated off
2. CPU subsystem
   a. Retention – power down with retention
   b. Shut Down – power down; memory/register content will be lost; power is removed from the block

32.4.2 M4 SRAM power domains and sleep Modes

The SRAM may be in one of these states:

- OFF or Shutdown – memory content will be lost; power is removed from the block.
- Deep Sleep – memory content is kept; power to the block is on.
- Light Sleep – see Section 32.4.3 for more details.
- ON – normal operation.

Twelve 32KB SRAM attached to M4 AHB bus and four 32KB SRAM attached to AON AHB bus (arbitrated with M4 AHB bus) are individually in their own power domain with its own head switch as illustrated below:

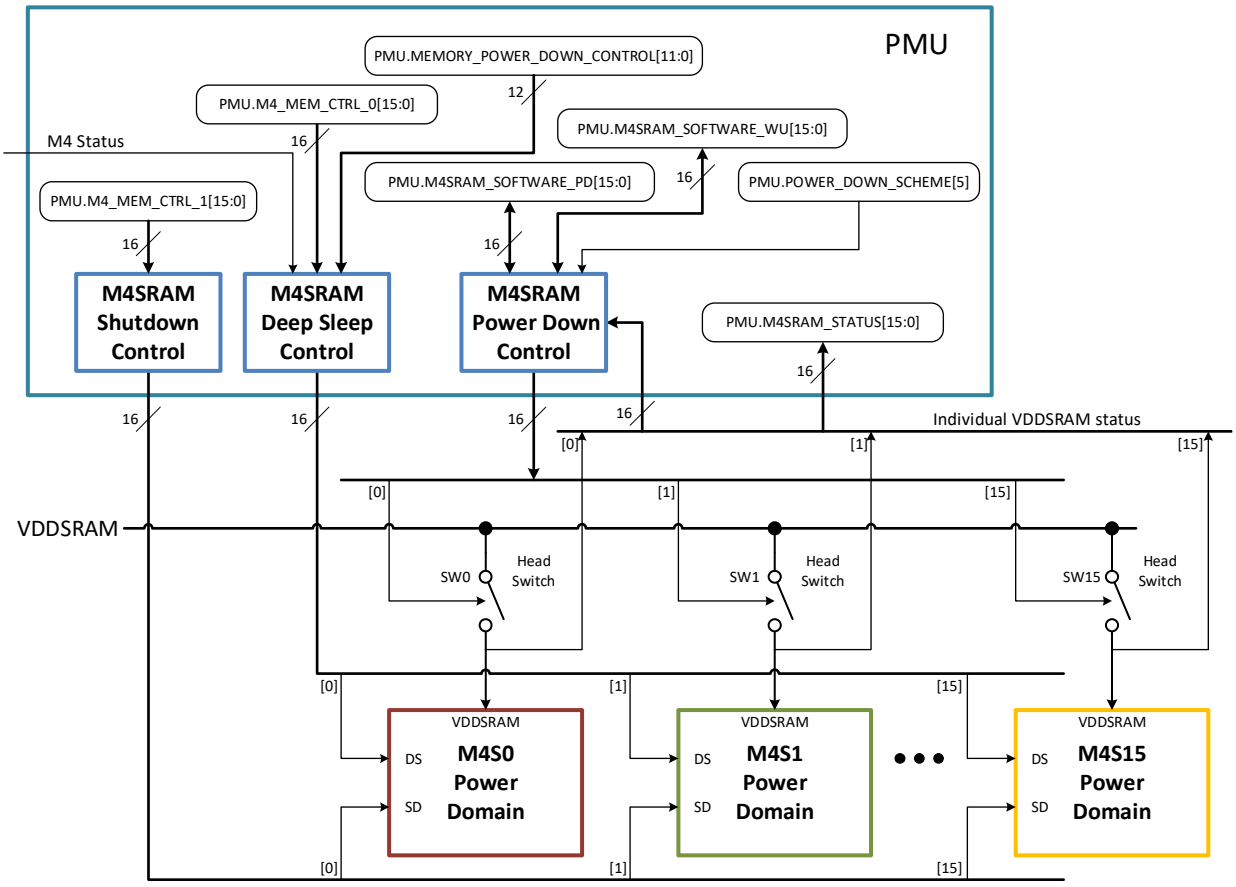

Figure 32-3: M4 SRAM Power Domains

M4 SRAM power domain control and status registers in PMU are described in following sections.

32.4.3 M4 SRAM Sleep Modes: LPMF and LPMH

The M4 SRAMs support Transparent Light Sleep (TLS) mode using LPMF (footer) circuit and Deep Sleep mode using LPMH (header) circuit.

- LPMH – Controls SRAM header to reduce bitcell leakage in Deep Sleep mode
- LPMF – enable Transparent Light Sleep (TLS) SRAM feature to reduce leakage current

Note: These modes are not recommended in a typical system operation. Enable LPMH and LPMF for shipping to achieve lowest power consumption.

32.4.4 M4 Power Domain Configuration

The power state/mode of M4 CPU and modules in the same power domain is controlled (configured) using these registers in PMU:

- Register PMU: WIC_CONTROL – enables wake up interrupt controller (WIC)
- Register PMU: EXTERNAL_WAKING_UP_SOURCE – configure mask for wake up source
- Register PMU: M4_POWER_MODE_CFG – configure M4 low power mode

Prior to initiating M4 CPU low power mode, the wake-up interrupt controller (WIC) must be enabled. Otherwise, the M4 will never wake-up unless the whole chip is reset.

32.4.5 M4 Power Domain Status

The following register indicates the status of M4 power state:

- Register PMU: WIC_STATUS - indicates whether WIC is ready for M4 power saving mode
- Register PMU: M4_STATUS - indicates whether M4 is active or shutdown

32.4.6 M4 Entering Low Power Mode

Low Power mode is entered via M4 instructions: Wait For Interrupt (WFI) or Wait For Event (WFE).

32.4.7 M4 Exiting Low Power Mode

M4 wake-up must be triggered by hardware event.

External wake up sources are WIC interrupt.

These wake-up sources may be masked.

32.4.8 M4 SRAM Power Domain Configuration

The power state/mode of M4 SRAM is configured using these registers in PMU:

- Register PMU: POWER_DOWN_SCHEME - configures concurrent or sequential shutdown of SRAM blocks (other than M4S0)
- Register PMU: MEMORY_POWER_DOWN_CONTROL - configures concurrent M4 and M4SRAM power savings
- Register PMU: M4SRAM_SOFTWARE_LPMF - configures M4 SRAM Light Sleep Footer
- Register PMU: M4SRAM_SOFTWARE_LPMH_MASK_N - configures M4 SRAM Light Sleep Header

M4S0 SRAM Block

The M4S0 block is a special block with additional features that are configured with the following registers:

- Register PMU: M4S0_POWER_MODE_CFG - configures M4S0 32KB memory power mode
- Register PMU: M4S0_PD_SOURCES_MASK_N - configures M4S0 32KB memory power down mask
- Register PMU: M4S0_WU_SOURCES_MASK_N - configures M4S0 32KB memory wake up mask

32.4.9 M4 SRAM Power Domain Status

The following register indicates the status of M4SRAM power state:

- Register PMU: M4S0_SRAM_STATUS - indicates whether M4S0 is active or shutdown
- Register PMU: M4SRAM_STATUS - indicates whether corresponding M4SRAM is active or shutdown

32.4.10 M4 SRAM Entering Low Power Mode

The M4SRAM low power mode may be initiated by PMU hardware or firmware.

To enter low power mode through firmware, these control registers in PMU initiate the power saving mode sequence. Setting these bits will trigger the low power mode sequence.

- Register PMU: M4_MEM_CTRL_0 - controls deep sleep signal of corresponding M4SRAM
- Register PMU: M4_MEM_CTRL_1 - controls shutdown signal of corresponding M4SRAM
- Register PMU: M4SRAM_SOFTWARE_PD - controls power head switch of corresponding M4SRAM

When MEMORY_POWER_DOWN_CONTROL[11:0] is set, corresponding M4SRAM M4S[11:0] will enter deep sleep mode when M4 is shutdown. See MEMORY_POWER_DOWN_CONTROL register description for more details.

32.4.11 M4 SRAM Exiting Low Power Mode

M4 SRAM wake-up may be triggered by hardware event or initiated by software.

To exit low power mode through firmware, these control registers in PMU initiate the wake-up. Set register PMU: M4SRAM_SOFTWARE_WU will trigger the wake-up sequence of corresponding SRAM block.

External wake up sources are WIC interrupt.

These wake-up sources may be masked.

32.5 Voice (Audio) Sub-System Low Power Modes

32.5.1 Entry to a Low Power State

The Voice / audio sub-system logic power domains, AD0, AD1, AD2, AD3, AD4, and AD5, may enter the low power mode Shutdown by means of firmware-generated action. There is no hardware trigger for entry to low power mode.

32.5.2 Exiting from a Low Power State

The Voice / audio sub-system logic power domains, AD0, AD1, AD2, AD3, AD4, and AD5, may exit the Shutdown state by means of firmware-generated action or when initiated by either of these hardware actions:

1. When LPSD hardware is triggered
2. By GPIO activity related to the voice sub-system.

These hardware triggers events may be masked.

LPSD triggering and GPIO activity are described in Voice Sub System section.

32.6 Voice (Audio) Sub-System SRAM Low Power Modes

32.6.1 Entry to a Low Power State

The Voice / audio processing sub-system SRAM power domains are: AD1_SRAM_R0, AD1_SRAM_R1, AD1_SRAM_R2, AD2_SRAM_R0, AD2_SRAM_R1, and AD2_SRAM_R2. These power domains may enter low power (Deep sleep or Shutdown) mode by firmware-generated action. Also, a non maskable hardware trigger can shut down the AD1 or AD2 SRAMs in AD1 or AD2.

32.6.2 Exiting a Low Power State

Firmware to enable power switches for audio blocks.

32.7 FPGA (FB) Sub-System Low Power Modes

The FPGA Fabric (FB) has its own VLP mode power management circuit independent of the PMU. It decides to use VLP mode or not based on PMU register bit.

The following registers in the PMU are related to the low power modes for the FB:

- Register: POWER_DOWN_SCHEME
  - FFEFB_PD – configures concurrent or sequential power down of FFE and FB.
  - FFEFB_WU – configures concurrent or sequential wake up of FFE and FB.
- Register: FB_STATUS - reports whether FB is active, deep sleep or shutdown
- Register: FB_POWERMODE_CFG - configures whether to enter deep sleep or shutdown when PD event occurs
- Register: FB_PD_SOURCES_MASK_N – set the mask for sources that can power down the FPGA
- Register: FB_WU_SOURCES_MASK_N – set the mask for sources that can power on the FPGA
- Register: FFE_FB_PF_SOFTWARE_PD - put FB into low power mode as defined by **FB_POWER_MODE_CFG** register

32.8 Sensor Processing (FFE) Sub-System Low Power Modes

The sensor processing (FFE) sub-system is typically turned off until sensor reading is needed. The sub-system may be woken up by an interrupt or periodically to poll the sensors. The following diagram illustrates the sequence for periodically turning on the FFE power domain.

1. The SPT generates the FFE kick off (wake up) signal with the period FFE_TO_PERIOD.
2. Prior to FFE kick off, PMU turns on the FFE power domain. This is the PMU kick off pulse.
3. FFE will start processing after PMU_TO_PERIOD to wait for the FFE power domain to stabilize.

PMU will turn off FFE power domain once the FFE busy signal is de-asserted.

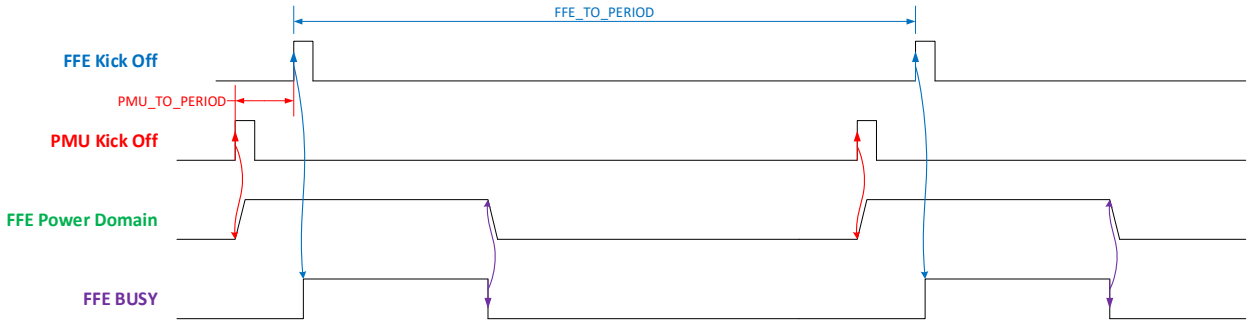

Figure 32-4: FFE Wake up/Shutdown Sequence

32.9 SDMA Low Power Mode

The SDMA_BRIDGE monitors how many cycles SDMA stays in IDLE state. If SDMA stays in IDLE state longer than the SDMA_BRIDGE.SDMA_PWRDN_CNT threshold, SDMA_BRIDGE will send a power down request to PMU to shut down the SDMA power domain.

Chapter 33. The Configuration Manager sub-system

33.1 Introduction

This chapter outlines the purpose and use of this subsystem, and its internal construction.

The S3 Configuration Manager provides support for Host Mode designs that require program loading from external flash memory. In contrast, AP mode designs use the host to do program loading, and they work with another S3 subsystem, the Communication Manager and its SPI_Slave interface, in doing this.

33.2 Configuration sub-system architecture

The Configuration Manager subsystem (also known as the CM) is made up of the Configuration State Machine (CSM), the SPI_1_Master interface, and the Configuration DMA module (ConfigDMA). These blocks are designed to work together having the main function to communicate with SPI based external peripherals. These external peripherals include: Flash, SPI based external peripherals and SPI based sensors. Details of block functions are given in the following sections.

The SPI_1_Master and the CFG_DMA can also be used by the M4 or AP or sensors.

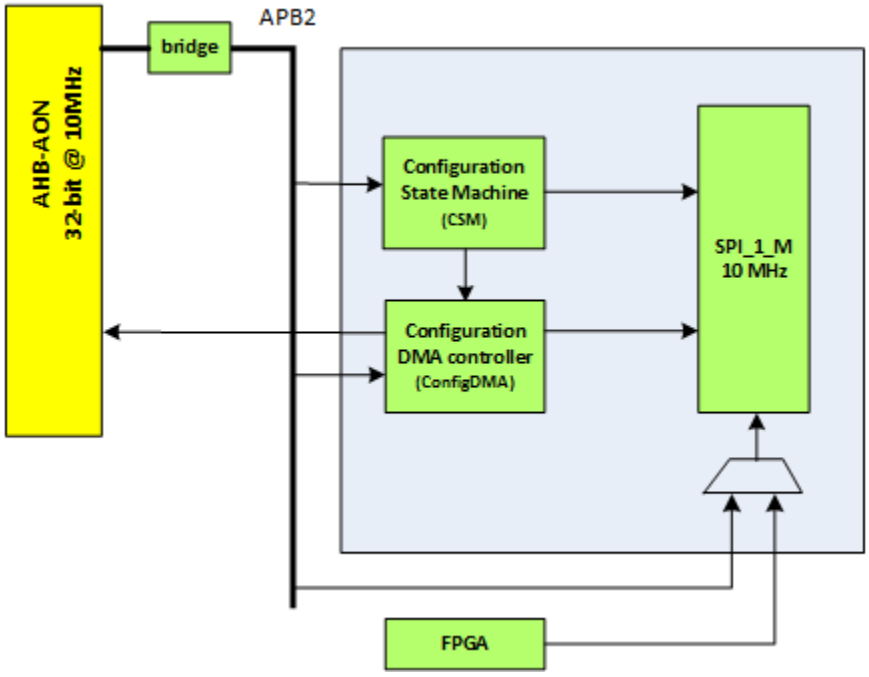

Figure 33-1: Configuration Manager high-level architecture

33.3 Configuration State Machine General Operation

Configuration Logic is responsible for:

- Reading the external flash device
- Configuring its SPI transfer parameters using data stored within the external flash devices
- Confirming that the boot code is compatible with the EOS S3 device using stored values in the external flash.
- Loading the boot code into M4-F memory and enabling the M4-F execution once the boot code transfer has completed.
- Minimizing the elapsed time for booting the M4-F by using DMA transfers of boot code from the external .flash
- Posting status bits to the M4-F that aid in diagnosing the state of the boot process

The Configuration State Machine is a finite-state-machine (FSM) which can execute 4 different microcode sequences.

- Read Header Contents from Flash
- Download Boot Image from Flash
- Deep Sleep Off Mode

Figure 31-2 illustrates how the Configuration State Machine works once the Host bootstrap is selected (GPIO_20 = pull down) and reset is deasserted.

33.4 Read Header Contents from Flash

The CfgSM will download the flash header from the SPI Flash. This header requires the following information and format in table below.

Table 33-1: Flash Boot Header

Flash Boot Header Fields	Bit Field	Value	Description
Device ID	[31:24]	0x20	This value is required for EOS S3.
SPI Transmit Clock Divider	[23:16]	X	This X value multiplied by 2 is used to program the SPI_BAUDR register which controls the SPI Master clock output to SPI Flash. If X = 2, then 4 is programmed to SPI_BAUDR, which is 3.21 Mhz.
Boot Size in Frames	[15:0]	Y	This Y is boot size minus 1 in double words. If Y = 0x1FF, then 4096 bytes will be downloaded from SPI Flash. Frame size is 8 bytes.

The values in the header are stored in a register and used by the CfgSM to setup the various values, like DMA transfer size and SPI clock. The Device ID in the header must match the value 0x20 for EOS S3. **Note:** If the Device ID does not match, the header will be read three more times and checked. If, after the fourth time, the Device ID still does not match, the flash will be put in deep sleep and the device error state will be entered. There is a timer delay between successive reads of the header. Once the CfgSM enters the device error state, it can only be restarted by asserting the reset signal.

The CfgSM downloads the flash header at a slow speed of ~ 643.9 KHz. This slow frequency is to allow download from a wide range of SPI Flash models. The header can be customized to speed up the SPI clock for boot image download.

A possible flash header value could be 0x20021FFF where the clock divider is 2 * 2 = 4 (3.21 MHz) and 0x1FFF for 64 Kbytes are downloaded.

33.5 Boot SPI

The CfgSM will load the boot image once the header has been read successfully. This size of the image was set by the value obtained from the header. The clock divider for the portion of the transfer is also set by the value in the header. The Configuration DMA is used to read data from the flash via the SPI and transfer it to the M4 memory by way of the dma_to_ahb bridge. The transfer will continue until the dma_done signal is asserted. Dma_done is produced when the last frame has been transferred and the dma_to_ahb bridge has finished any pending transfers.

Please refer to software considerations section for details.

Important: SPI Master can support up to 3 slave devices. However, the SPI Flash for booting needs to be attached to the **PAD 39** (**SPIm_SS1**). The microcode in the CfgSm assumes this.

33.6 Deep Sleep Mode

Deep sleep is a flash mode that offers significant power savings. It requires special instructions to enable and enter this mode. This mode will be enabled when the flash header has failed the four read attempts.

33.7 Software Considerations

The CfgSM for boot-up in wearable mode uses the Configuration DMA hardware. Once the boot-up is complete, the DMA_Ctrl's **dma_stop** bit will be set to 1. Refer to table 33-1 for bit field. This needs to be explicitly cleared by software by writing a 1 to this bit, so that future DMAs from SPI Master will work correctly.

Important: Remember to clear the dma_stop bit after Wearable boot mode (bootstrap GPIO 20 = 0), by writing a 1 to clear the bit. Writing a zero has no effect and will not clear it.

Table 33-2: DMA_CTRL Register

Offset	Name/Field	Bits	Type	Default	Description
0x000	DMA_ctrl	31:0		0x0	DMA Control : this register is only accessible when the dma or cfg_sm is not selecting the dmas_mux.
	dma_start	0	RW1S	0x0	write a 1: Enable write a 0: no affect, reads dma_enb
	dma_stop	1	RW1C	0x0	write a 1: Stop DMA and disable, clears DMA_DONE write a 0: no affect, reads dma_done
	dma_ahb_sel	2	RW	0x0	0: DMA to AHB 1: DMA to header register
	dma_hsel	3	RO	0x0	1: AHB hsel is asserted 0: not asserted
	dma_htrans_0	4	RO	0x0	1: AHB htrans[0] is asserted 0: not asserted
	dma_htrans_1	5	RO	0x0	1: AHB htrans[1] is asserted 0: not asserted
	dma_hready	6	RO	0x1	1: AHB hready is asserted 0: not asserted
	dma_xfr_pending	7	RO	0x0	1: DMA transfer is pending 0: nothing pending
	bridge_xfr_pending	8	RO	0x0	1: AHB bridge transfer is pending 0: nothing pending

33.8 Configuration DMA

The Configuration DMA registers are located at address base 0x40007400. There registers will control the DMA behavior. The DMA can only burst read out data from SPI slave devices through the SPI Master.

Additional SPI Master programming is required as well. As mentioned earlier, the SPI Master's address base is 0x4000_7000. The below registers are referred to by their address offset.

33.8.1 How to Start a DMA

To start a DMA read transfer from SPI Flash, perform the following steps.

DMA Boot Setup

1) DMA source address is always SPI_DR0 (0x060) register. This is fixed.
2) Set DMA_DESTINATION_ADDRESS (0x004) to be location in memory (e.g. 0x20060000)
3) Set DMA_TRANSFER_COUNT (0x000) to the frames to transfer minus one.
4) Set DMA_CTRL (0x000) = 0x2 to clear DMA_stop bit if set.

SPI Boot Setup

1) Disable SPI_SSIENR (0x008)=0, to be able to write registers (if not already disabled)
2) Set SPI_CTRL0 (0x000) = 0x307.
    a. Set DFS = 0x7. 8-bit serial data transfer.
    b. Set TMOD = 0x3. EEPROM Read.
3) Set SPI_IMR (0x02c) = 0x1C.
    a. A '1' unmasks an interrupt.
    b. **Important Note:** RXFIM interrupts needs to be unmasked for DMA to read out data correctly from SPI Master's receive FIFO.
4) Set Frames to transfer
    a. Set SPI_CTRLR1 (0x004) = Number of frames to transfer minus 1.
    b. This must be a multiple of 4 to create 32 bit words for the DMA.
5) Set SPI_BAUDR (0x014) to 2 or higher for SPI clock divider.
6) Enable SPI_SSIENR (0x008) = 0x1.
    a. This allows writing data to the FIFOs.
7) Set the opcode (1 byte) and address (3 bytes) into the transmit FIFO SPI_DR0 for reading the flash (DR).
    a. Write SPI_DR0 = 0x3. SPI Flash Read opcode = 0x3 (read).
    b. Write SPI_DR0 = Most significant byte of address (addr[23:16]).
    c. Write SPI_DR0 = Mext byte of address (addr[15:8]).
    d. Write SPI_DR0 = Last byte of address (addr[7:0]).
8) Set DMA_CTRL = 0x1, to enable DMA.
9) Enable the desired slave select (SPI_SER). This will begin the SPI transfer.

The SPI Master will transmit the data in its SPI_DR0 FIFO (transmit FIFO). The SPI Flash model will see the 0x3 Flash Read command and the 3-byte address. It will return byte data to the SPI Master, as long as SPI's chip select is held low. The SPI Master counts the number of bytes it needs to read back. As bytes of data are received, the RXFIM interrupt will trigger with each 2 bytes. This interrupt signals to the DMA to read out the bytes from SPI_DR0 FIFO (receive FIFO). The DMA engine will write the bytes starting at DMA_DESTINATION_ADDRESS, and incrementing the address after every 4 bytes. Once the DMA engine is done, it will signal via a DMA_DONE interrupt and sets the DMA_CTRL's dma_stop bit = 1. This dma_stop bit needs to be cleared by writing a 1 to it, before doing the next DMA operation.

33.8.2 How to Stop an Active DMA Transfer

To stop a currently running DMA transfer, perform the following steps.

1) Set SPI_SSIENR(0x008) = 0 to stop the SPI.
2) Set DMA_CTRL (0x000) = 0x2 to clear DMA_stop bit if set.

This will stop the SPI and DMA transfer. Any pending data in the DMA_TO_AHB bridge FIFO will still be transferred until all the FIFO data has been processed.

Chapter 34. SPI Master

The SPI Master consists of the following features:

- Operates as a Master only
- Supports up to three slaves (SPIm_SS1, SPIm_SS2, SPIm_SS3)
- Operates in mode 0 (This can be reprogrammed by M4-F later)
- Supports a frame size of 8 (This can be reprogrammed by M4-F later)
- Supports a maximum transfer size of 64K Frames
- Supports little-endian data ordering
- Shifts out the most significant bit data first
- Supports DMA transfers
- Supports standard SPI protocol

SPI Master Interface does not provide the following:

- Support for multiple SPI masters
- Support for other serial protocols (such as SSP or Microwire)
- Support for protocols that include DDR, dual, and quad transfers

SPI Master Interface that is accessible by:

- M4-F Processor
- Host Application Processor
- Configuration Logic
- On-chip programmable logic

The SPI register block is at base address 0x40007000.

34.1 SPI Master

Here is a high-level block diagram for the SPI Master. Please refer to Figure below.

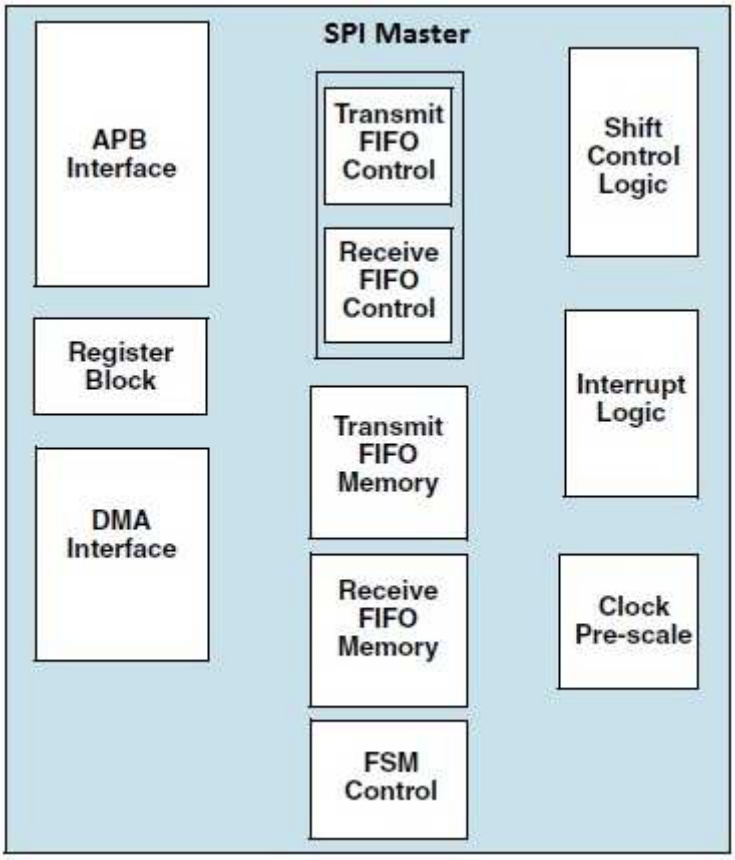

Figure 34-1: SPI Master Block Diagram

Programming the SPI Master go through the APB bus interface. There is a register block inside the SPI Master.

The SPI master can both transmit serial data and receive serial data from SPI slaves. The width of the transmit FIFO and receive FIFOs are 16-bits wide. Transmit FIFO is 131 entries deep. Receive FIFO is 8 entries deep.

Writing SPI_DR0 (0x060) register will push data transmit FIFO. Reading SPI_DR0 register will pop data from the receive FIFO. Use the status bits in SPI_SR register to see when Transmit FIFO is empty (TFE) or when the Receive FIFO is full (RFF). Interrupts can be triggered on these status bits as well.

The SPI_BAUDR register controls the SPI clock to the SPI Slaves. It is recommended that SPI clock be ½ of the C02 clock frequency. Write 0x2 or higher multiples of 2 to SPI_BAUDR.

34.2 SPI Transfer Modes

The transfer mode (TMOD) is set by writing control register SPI_CTRL0.

34.2.1 Transmit and Receive

When TMOD = 2'b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the SPIm_MOSI line to the target device, which replies with data on the SPIm_MISO line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

34.2.2 Transmit Only

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the SPIm_MOSI line to the target device, which replies with data on the SPIm_MISO line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

34.2.3 Receive Only

When TMOD = 2'b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The SPIm_MOSI output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

34.2.4 EEPROM Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the SPI Master is transmitting data on its SPIm_MOSI line, data on the SPIm_MISO line is ignored). The SPI Master continues to transmit data until the transmit FIFO is empty. Therefore, you should ONLY have enough data frames in the transmit FIFO to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO than are needed, then read data is lost. When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (SPIm_MISO) is valid and is stored in the receive FIFO; the SPIm_MOSI output is held at a constant logic level. The serial transfer continues until the number of data frames received by the SPI Master matches the value of the NDF field in the CTRLR1 register + 1.

34.3 SPI Flash Command Write

The SPI Master can be programmed to do SPI Flash command writes. A SPI Flash command is typically 1 byte write (8-bits of data). The following sequence can be used for other SPI slave devices that require 1 byte write.

1) SPI_SSIENR = 0. Disable the SPI, so it can be programmed
2) SPI_CTRLR0 = 0x107.
    a. Set DFS = 0x7. 8-bit serial data.
    b. Set TMOD = 1. Transmit Only.
3) SPI_BAUDR = 0x2.
    a. Divide the SPI clock by 2.
4) SPI_SER = 0. Disable the SPI until transmit fifo is written.
5) SPI_SSIENR = 0x1.
6) Write SPI_DR0 the command byte you want to transmit.
7) Set SPI_SER to the SPI slave you want to address (0x1, 0x2, or 0x4).
8) Wait for SPI transmit to finish. Poll on SPI_SR status bits. See Table 34-1
    a. Wait for SPI_SR [2] TFE = 1. Transmit FIFO is empty.
    b. Wait for SPI_SR [0] BUSY = 0. Wait for Busy is cleared.[1]

Note 1: Data transfers are started by SPI Master. When SPI Master is enabled (SPI_SSIENR = 0x1), at least one valie data entry is present in the transmit FIFO and a serial-slave device is selected (SPI_SER = 0x1, 0x2, or 0x4). When actively transferring data, the busy flag (BUSY) in the status register (SPI_SR) is set. You must wait until the busy flag is cleared before attempting a new serial transfer. The BUSY status is not set when the data are written into the transmit FIFO. This bit gets set only when the target slave has been selected and the transfer is underway. After writing data into the transmit FIFO, the shift logic does not begin the serial transfer until a positive edge of the sclk_out signal is present. The delay in waiting for this positive edge depends on the baud rate of the serial transfer (SPI_BAUDR). Before polling BUSY status (SPI_SR), you should poll the TFE status (waiting for 1) or wait BAUDR * ssi_clk clock cycles.

Table 34-1: SPI_SR Register

Offset	Name/Field	Bits	Type	Default	Description
0x028	SR	6:0			
	DCOL	6	RC	0x0	Data Collision Error. Relevant only when the SPI Master is configured as a master device. This bit is set if the SPI Master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
	TXE	5	RC	0x0	No function for SPI Master. Slave usage only.
	RFF	4	RO	0x0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
	RFNE	3	RO	0x0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO. 0 - Receive FIFO is empty 1 - Receive FIFO is not empty
	TFE	2	RO	0x1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
	TFNF	1	RO	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
	BUSY	0	RO	0x0	SSI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the SPI Master is idle or disabled. 0 – SPI Master is idle or disabled 1 – SPI Master is actively transferring data

34.4 SPI Flash Page Write

The SPI Master can write to SPI Flash model for entire page.

1) Send Write Enable (WREN) command 0x6 to SPI flash.
2) SPI_SSIENR = 0. Disable SPI Master.
3) SPI_CTRLR0 = 0x10F.
    a. Set DFS = 0xF. 16-serial data transfer.
    b. Set TMOD = Transmit Only.
4) SPI_BAUDR = 0x2. Divide SPI Clock by 2.
5) SPI_SER = 0. Disable the SPI select until DR0 is written.
6) SPI_SSIENR = 1. Enable SPI Master.
7) Send the page program command and flash address (flash_addr[23:0]) you want to access.
    a. SPI_DR0 = (0x2 << 8) | flash_addr >> 16.
        i. Send the PAGE Program (PP) 0x2 command
        ii. Send the flash address [23:16] you want to access.
    b. SPI_DR0 = flash_addr[15:0]
        i. Send the flash address [15:0] you want to access.
8) Send the data that you want to write.
    a. Write SPI_DR0 as many times as page size / 2 bytes.
9) Set SPI_SER to the SPI slave you want to address (0x1, 0x2, or 0x4).
10) Wait for SPI transmit to finish. Poll on SPI_SR status bits. See Table 34-1.
    a. Wait for SPI_SR [2] TFE = 1. Transmit FIFO is empty.
    b. Wait for SPI_SR [0] BUSY = 0. Wait for Busy is cleared.

Example if the SPI Flash page is 256 bytes, then SPI_DR0 needs to be written 256 / 2 bytes = 128 times for the entire page to be written. The procedure needs to be repeated to access the next SPI flash page with the flash address incremented by 256.

Sytem Clocks

Chapter 35. Clock Setup

This chapter describes clock setup details.

35.1 Change the Oscillator Frequency

The Oscillator (OSC) controls are located in Analog IP (AIP) registers at address base (0x40005400). Below are the registers required to operate the OSC control.

Table 35-1: AIP_OSC Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x40005480	AIP_OSC_CTRL_0	31:0			
	RESERVED	31:1			
	fref16k_sel	1	RW	0x0	1'b0 : reference clock is 32KHz 1'b1 : reference clock is 16KHz
	en	0	RW	0x1	1'b0 : OSC OFF 1'b1 : OSC ON (NO SYNC needed, OSC guarantee there is no Glitch)
0x40005484	AIP_OSC_CTRL_1	31:0			
	RESERVED	31:16			
	General_Purpose_SFR	15:13	RW	0x00	General Purpose bits
	prog	12:0	RW	0x92D	Please refer OSC datasheet for others (No SYNC Needed) Power On Default Value is 76.97 MHz No Support on "Delta Mode"
0x400054A0	AIP_OSC_STA_0	31:0			
	RESERVED	31:2			
	anatestreq	1	RHW	0x0	Please refer OSC datasheet for detail, NO SYNC, FW need to read it twice to ensure the value.
	lock	0	RHW	0x0	Please refer OSC datasheet for detail, NO SYNC, FW need to read it twice to ensure the value.

By default the OSC is enabled.

- To turn OSC OFF, write AIP_OSC_CTRL_0[0] en = 0.
- To turn OSC ON, write AIP_OSC_CTRL_0[0] en = 1.

EOS S3 OSC ranges from 2.1 MHz to 80 MHz. The register AIP_OSC_CTRL_**1prog** [12:0] controls the frequency. Below is the equation for OSC frequency based on **prog** value. Additional information can be found in next section.

$$OSCfreq = (prog\,[11:0] + 3) * 32{,}768\,Hz$$

Procedure to change the OSC frequency:

- Make sure Oscillator is enabled by AIP_OSC_CTRL_0 bit[0] en = 1.
- Set AIP_OSC_CTRL_1 **prog[11:0]** to change frequency
  - Write 0x40 to get 2.1 MHz OSC output.
  - Write 0x980 to get 79.79 MHz OSC output.
  - Keep **prog** bit[12] to zero for all time.
- Poll the AIP_OSC_STAT **lock** goes 0 first.
  - During this time, the OSC is trying to lock to new **prog** frequency.
- Poll the AIP_OSC_STAT **lock** goes high
  - Read this register twice to ensure **lock**.
  - Once lock = 1, OSC has locked to new prog frequency. Clock is guaranteed to be the programmed frequency.

Typical time for OSC frequency change is 210 us (maximum). OSC startup time to look will be 210 us + 60 us (maximum).

35.2 Oscillator Programming Table

Below is the OSC's programming table for AIP_OSC_CTRL_1 bit[11:0] – prog.

Please keep AIP_OSC_CTRL_1 bit[12] – prog[12] to zero. This is the delta feature, which is not supported in EOS S3.

prog	freq [Hz]	max error [%]	max error [periods]
000001000000	2195456	6.25	4
000010000000	4292608	3.13	4
000011000000	6389760	2.08	4
000100000000	8486912	3.13	8
000101000000	10584064	2.50	8
000110000000	12681216	2.08	8
000111000000	14778368	1.79	8
001000000000	16875520	1.56	8
001001000000	18972672	1.39	8
001010000000	21069824	1.25	8
001011000000	23166976	1.14	8
001100000000	25264128	1.04	8
001101000000	27361280	0.96	8
001110000000	29458432	0.89	8
001111000000	31555584	0.83	8
010000000000	33652736	0.78	8
010001000000	35749888	0.74	8
010010000000	37847040	0.69	8
010011000000	39944192	0.66	8
010100000000	42041344	0.63	8

prog	freq [Hz]	max error [%]	max error [periods]
010101000000	44138496	0.60	8
010110000000	46235648	0.57	8
010111000000	48332800	0.54	8
011000000000	50429952	1.04	16
011001000000	52527104	1.00	16
011010000000	54624256	0.96	16
011011000000	56721408	0.93	16
011100000000	58818560	0.89	16
011101000000	60915712	0.86	16
011110000000	63012864	0.83	16
011111000000	65110016	0.81	16
100000000000	67207168	0.78	16
100001000000	69304320	0.76	16
100010000000	71401472	0.74	16
100011000000	73498624	0.71	16
100100000000	75595776	0.69	16
100101000000	77692928	0.68	16
100110000000	79790080	0.66	16
100111000000	81887232	0.64	16

Figure 35-1: OSC programming controls

35.3 Setup the Clock Source

Program the following Clock Mux Registers to select the clock source for each clock.

IMPORTANT: Bootstrap PAD9 = 1 will select the FCLK as input clock from PAD6. If the register(s) below select the High Speed Clock, then FCLK is the clock source in this case. Since this is a bootstrap option, this can clock selection can only be cleared through system reset.

Table 35-2: C10-CLK_CONTROL_A_1 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x004	CLK_Control_A_1	31:0			For Clock 10 (SYNC Up on A0 and AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix and Trace block)
	RESERVED	31:10			
	Clock_Source_Selection	1:0	RW	0x0	2'b00 --> High Speed/Divided Clock 2'b01 --> 32KHz Others --> Reserved **Please Note : If Reserved Value has been programmed, then the clock will be turn off and cannot be recovery without System Reset.**

Table 35-3: C02 - CLK_SWITCH_FOR_B Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x130	CLK_SWITCH_FOR_B	31:0			For Clock 2 (eFUSE, FB, A1 (Including CFGSM))
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

Table 35-4: C08X4 - CLK_SWITCH_FOR_C Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x134	CLK_SWITCH_FOR_C	31:0			For Clock 8 X4 (FFE X4 clk)
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

Table 35-5: C11 - CLK_SWITCH_FOR_D Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x138	CLK_SWITCH_FOR_D	31:0			For Clock 11 (To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

Table 35-6: C16 - CLK_Control_F_1 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x024	CLK_Control_F_1	31:0			For Clock 16 (FB)
	RESERVED	31:2			
	Clock_Source_Selection	1:0	RW	0x0	2'b00 --> High Speed/Divided Clock 2'b01 --> 32KHz Others --> Reserved **Please Note : If Reserved Value has been programmed, then the clock will be turn off and cannot be recovery without System Reset.**

Table 35-7: C19 - CLK_SWITCH_FOR_H Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x13C	CLK_SWITCH_FOR_H	31:0			**For Clock 19 (ADC)**
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

Table 35-8: C21 - CLK_Control_I_1 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x038	CLK_Control_I_1	31:0			For Clock 21 (FB - additional clk)
	RESERVED	31:2			
	Clock_Source_Selection	1:0	RW	0x0	2'b00 --> High Speed/Divided Clock 2'b01 --> 32KHz Others --> Reserved **Please Note : If Reserved Value has been programmed, then the clock will be turn off and cannot be recovery without System Reset.**

Table 35-9: C23 - CLK_SWITCH_FOR_J Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x140	CLK_SWITCH_FOR_J	31:0			For CLK 23 (PMU clk gating control)
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

Table 35-10: C30 - CLK_SWITCH_FOR_G Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x144	CLK_SWITCH_FOR_G	31:0			To C30(PDM LEFT/RIGHT Clk, I2S_MASTER)
	RESERVED	31:1	RO	0x0	
	Clock_Source_Selection	0	RW	0x0	1'b1 : 32KHz Clock 1'b0 : High Speed Clock

35.4 Setup the Divider

Program the following Clock Divider Control Registers to setup the clock divider ratio for each clock.

Note: The default of bit 9 is 1, Clock Divider is ON. Program 0 to bit 9, the Clock Divider is OFF and the Source Clock will be output directly.

Table 35-11: C10 - CLK_Control_A_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x000	CLK_Control_A_0	31:0			For Clock 10 (SYNC Up on A0 and AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix and Trace block)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0x4**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 6.**

Table 35-12: C02 - CLK_Control_B_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x008	CLK_Control_B_0	31:0			For Clock 2 (eFUSE, FB, A1 (Including CFGSM))
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0x4**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 6.**

Table 35-13: C08X4 - CLK_Control_C_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x010	CLK_Control_C_0	31:0			For Clock 8 X4 (FFE X4 clk)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0x4**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 6.**

Table 35-14: C11 - CLK_Control_D_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x014	CLK_Control_D_0	31:0			For Clock 11 (To M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0xE**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 16.**

Table 35-15: C16 - CLK_Control_F_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x020	CLK_Control_F_0	31:0			For Clock 16 (FB)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0xE**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 16.**
0x024	CLK_Control_F_1	31:0			**For Clock 16 (FB)**
	RESERVED	31:2			
	Clock_Source_Selection	1:0	RW	0x0	2'b00 --> High Speed/Divided Clock 2'b01 --> 32KHz Others --> Reserved **Please Note : If Reserved Value has been programmed, then the clock will be turn off and cannot be recovery without System Reset.**

Table 35-16: C19 - CLK_Control_H_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x02C	CLK_Control_H_0	31:0			For Clock 19 (ADC)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0xE**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 16.**

Table 35-17: C21 - CLK_Control_I_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x034	CLK_Control_I_0	31:0			For Clock 21 (FB - additional clk)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0xE**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x4, Divided by 16.**

Table 35-18: C23 - CLK_Control_PMU Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x100	CLK_Control_PMU	31:0			This Clock is used to delay the Clock gating control signals from PMU. The PMU will monitor the feedback/delayed Clock Gating Control signals to ensure the clocks are OFF before jump to next state. The Firmware needs to Configure this divider to ensure there delay time is longer enough. C23 Clock needs to be 2/3 of the lowest clock frequency of other clocks. For Example, if the Lowest clock frequency of other clocks is 5, then C23 should be lower than 3.33MHz (Or the clock frequency of other clocks should be at least 1.5 times faster than C23.)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	*1'b1 Clock Divider is ON* *1'b0 Clock Divider is OFF, Output the Source Clock Directly*
	Clock_Divider_Ratio	8:0	RW	*0x3E*	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved Default Divided by 64.

Table 35-19: C30 - CLK_Control_G_0 Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x028	CLK_Control_G_0	31:0			For Clock 30 (PDM LEFT/RIGHT clk, I2S Master clk)
	RESERVED	31:10			
	Enable_Clock_Divider	9	RW	0x1	***1'b1 Clock Divider is ON*** ***1'b0 Clock Divider is OFF, Output the Source Clock Directly***
	Clock_Divider_Ratio	8:0	RW	**0x0E**	High Speed Clock Divider Ratio, 0x0 --> Divide by 2 0x1 --> Divide by 3 ... 0x1FE --> Divide by 512 Others --> Reserved **Default 0x1E, Divided by 32.** **Note: This Default value is 0x0E, but the Clock Divider output is 1/32 of source clock by default. The issue only happens on default value.**

Table 35-20: C01_CLK_DIV Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x110	C01_CLK_DIV	31:0			Source Clock is C10 (CLK to eFUSE, SDMA,I2S module Inside A1, AHB2APB Bridge /CFG DMA Bridge inside A1 , FFE, Packet FIFO,SDMA,A0) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C01_CLK_DIV_CG	4	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **This bit is used to turn off the clock for the SYNC down Divider**
	C01_CLK_DIV	3:0	RW	**0x1**	**4'h0 : The Source Clock Is Divided by 1, No division** **4'h1 : Divided by 2,** **...** **4'hf : Divided by 16.** **The input clock frequency will be divided and generate the corresponding clock output.**

Table 35-21: C09_CLK_DIV Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x114	C09_CLK_DIV	31:0			Source Clock is C10 (CLK to Voice APB interface, PIF, FB) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C09_CLK_DIV_CG	4	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **This bit is used to turn off the clock for the SYNC down Divider**
	C09_CLK_DIV	3:0	RW	**0x1**	**4'h0 : The Source Clock Is Divided by 1, No division** **4'h1 : Divided by 2,** **...** **4'hf : Divided by 16.** **The input clock frequency will be divided and generate the corresponding clock output.**

Table 35-22: C31_CLK_DIV Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x118	C31_CLK_DIV	31:0			Source Clock is C30 (LPSD CLK) If Bit 4 is 0, any change on Bit 3:0 will not take effect. And bit 4 and bit 3:0 cannot change at same time.
	RESERVED	31:5			
	C31_CLK_DIV_CG	4	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **This bit is used to turn off the clock for the SYNC down Divider**
	C31_CLK_DIV	3:0	RW	**0x3**	**4'h0 : The Source Clock Is Divided by 1, No division** **4'h1 : Divided by 2,** **...** **4'hf : Divided by 16.** **The input clock frequency will be divided and generate the corresponding clock output.**

35.5 Enable the Clock Gate

On EOS S3, clock gating registers are provided to turn off or enable the clocks. There are two levels of clock gating:

- Clock gating at input of the clock dividers.
- Clock gating at each individual path clock.

For the first level of clock gating, the CLK_DIVIDER_CLK_GATING(0x124) register controls each clock input into each clock divider (A , B, C, D, F, G, H, I, J). This will allow downstream clocks to gate off close to the root clock source. **Important:** Firmware should NOT program CLK_DIVIDER_A_CG bit[0]. This bit should remain 1. Please refer Table 35-23 for details of register.

Table 35-23: CRU_CLK_DIVIDER_CLK_GATING Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x124	CLK_DIVIDER_CLK_GATING	31:0			This register is used to turn off the clocks for Clock Dividers.
	RESERVED	31:10			
	CLK_DIVIDER_J_CG	9	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C23 (PMU clk gating control)**
	CLK_DIVIDER_I_CG	8	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C21 (FB)**
	CLK_DIVIDER_H_CG	7	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C19 (ADC)**
	CLK_DIVIDER_G_CG	6	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C30, C31 (PDM LEFT/RIGHT Clk, I2S_MASTER clk, LPSD CLK)**
	CLK_DIVIDER_F_CG	5	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C16 (FB)**
	RESERVED	4	RW	0x1	Reserved, Used as General Purpose Register
	CLK_DIVIDER_D_CG	3	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C11 (M4 peripherals - AHB/APB bridge, UART, WDT and TIMER)**
	CLK_DIVIDER_C_CG	2	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C08 (FFE X4, X1)**

Address	Register Name	Bit Field	R/W Access	Default	Description
	CLK_DIVIDER_B_CG	1	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C02 (eFUSE, FB, A1 (Including CFGSM))**
	CLK_DIVIDER_A_CG	0	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To C10,C01,C09 (SYNC Up on A0, AHB Interface of Batching Memory, AUDIO DMA, M4 SRAMs,M4 Bus Matrix , M4 Trace block, Debug controller, eFUSE, SDMA,I2S module Inside A1, AHB2APB Bridge /CFG DMA Bridge inside A1 , FFE, Packet FIFO,SDMA,A0, Voice APB interface, PIF, FB)** **Note: Firmware Should NOT program this bit to 0.**

The 2nd level of clock gating is at the path of each clock. For example, C01 clock drives 9 separate clocks C01_P0, C01_P1, C01_P2, C01_P3, C01_P4, C01_P5, C01_P6, C01_P7, and C01_P9. Each path clock can be individually gated. See C01_CLK_GATE register below.

Table 35-24: C00 - CRU_General Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x104	CRU_GENERAL	31:0			
	RESERVED	31:8			
	General	7:1	RW	0x0	General Purpose Register
	SPICLK_ALWAYS_ON	0	RW	0x0	1'b0 : Internal SPI Clock (C00) will be gated off if SPI CS is de-asserted even SPI Clock on the PAD is still running. 1'b1 : Internal SPIC Clock (C00) is running if SPI Clock on the PAD is toggling regardless of SPI CS value.

Table 35-25: C01_CLK_GATE

Address	Register Name	Bit Field	R/W Access	Default	Description
0x040	C01_CLK_GATE	31:0			
	RESERVED	31:10			
	Path_9_Gating_Control	9	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **For SPT.**
	RESERVED	8	RW	0x0	RESERVED
	Path_7_Gating_Control	7	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running To **eFUSE**
	Path_6_Gating_Control	6	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To SDMA**
	Path_5_Gating_Control	5	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To I2S module Inside A1**
	Path_4_Gating_Control	4	RW	**0x1**	1'b0 : Clock is Stop. 1'b1 : Clock is running **To AHB2APB Bridge /CFG DMA Bridge inside A1** , Allow M4 to configure SPI Master to load the code
	Path_3_Gating_Control	3	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To FFE**
	Path_2_Gating_Control	2	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To Packet FIFO**
	Path_1_Gating_Control	1	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To SDMA SRAM**
	Path_0_Gating_Control	0	RO	**0x1**	1'b0 : Clock is Stop. 1'b1 : Clock is running **To A0**

Table 35-26: C10_FCLK_GATE Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x050	C10_FCLK_GATE	31:0			
	RESERVED	31:7			
	Path_6_Gating_Control	6	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To the SYNC Up on A0 and AHB Interface of Batching Memory**
	Path_5_Gating_Control	5	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To AUDIO DMA**
	Path_4_Gating_Control	4	RWHC	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To M4 SRAM Instance, M4S12~M4S15** **This bit will be set if any of the Memories (M4S12~M4S15) been wakeup by Hardware.**
	Path_3_Gating_Control	3	RWHC	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To M4 SRAM Instance, M4S8~M4S11** **This bit will be set if any of the Memories (M4S8~M4S11) been wakeup by Hardware.**
	Path_2_Gating_Control	2	RWHC	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To M4 SRAM Instance, M4S4~M4S7** **This bit will be set if any of the Memories (M4S4~M4S7) been wakeup by Hardware.**
	Path_1_Gating_Control	1	RWHC	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To M4 SRAM Instance, M4S0~M4S3** **This bit will be set if any of the Memories (M4S0~M4S3) been wakeup by Hardware.**
	Path_0_Gating_Control	0	RO	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To M4 Bus Matrix and Trace block**

Important: While the M4SRAM 32KB banks have explicit clock gating controls in C10_FCLK_GATE(0x050) bit[4:1] , it is recommended to NOT enable them explicitly in Firmware. Instead, the auto-wakeup mechanism on the M4SRAM will power up and enable the clock gating automatically when any master (M4 , AP via SPI Slave, Debugger, SDMA, Audio DMA, CfgSM DMA) accesses the M4SRAM banks.

Table 35-27: C09_CLK_GATE Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x11C	C09_CLK_GATE	31:0			
	RESERVED	31:3			
	Path_2_Gating_Control	2	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To FB**
	Path_1_Gating_Control	1	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To PIF**
	Path_0_Gating_Control	0	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To voice APB Interface**

Table 35-28: C081X1_CLK_Gate Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x04C	C08_X1_CLK_GATE	31:0			
	RESERVED	31:4			
	Path_3_Gating_Control	3	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To PF ASYNC FIFO 0**
	Path_2_Gating_Control	2	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To A0**
	RESERVED	1	RW	0x0	Reserved
	Path_0_Gating_Control	0	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To FFE X1 clk**

Table 35-29: C30_C31_CLK_GATE Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x120	C30_C31_CLK_GATE	31:0			
	RESERVED	31:4			
	C31_path_0_Gating_Control	3	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To LPSD clk**
	C30_Path_2_Gating_Control	2	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To I2S Master Clk**
	C30_Path_1_Gating_Control	1	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To PDM RIGHT CLK**
	C30_Path_0_Gating_Control	0	RW	0x0	1'b0 : Clock is Stop. 1'b1 : Clock is running **To PDM LEFT CLK**

Table 35-30: CS_CLK_GATE Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x05C	CS_CLK_GATE	31:0			
	RESERVED	31:1			
	Path_0_Gating_Control	0	RW	0x1	1'b0 : Clock is Stop. 1'b1 : Clock is running **To SWD Clk from PIN**

Chapter 36. Functional Domain Clock Setup

This section talks about clock setup by functional domain. Some important notes on clock setup dependencies are included in this section.

36.1 Setup FFE clocks

The FFE subsystem max frequency is 10 MHz This means C08_X1 cannot not be programmed higher than 10 MHz. C08_X1 is divided down by 4 from C08_X4, which has a max frequency of 40 MHz. Below is the procedure for enabling the clocks required for FFE domain.

Important: Remember that FFE power domain needs to be powered up before enabling the clocks. By default, the FFE power domain is SHUTDOWN out of reset.

1. Setup the Clock Source as OSC.
    a. C01 - CLK_Control_A_1 (0x004) = 2'b00 (default)
    b. C08 – CLK_SWITCH_FOR_C (0x134) = 1'b0 (default)
2. Setup the Divider
    a. C01 – CLK_Control_A_0 (0x000) and C01_CLK_DIV (0x110). [A]
    b. C08 – CLK_Control_C_0 (0x010). This will setup C08_X4.[B]
3. Enable the Clock Gate
    a. CLK_DIVIDER_CLK_GATING (0x124) [2] = 1 (default)
    b. C01 – C01_CLK_GATE(0x040) [3] = 1.[C]
    c. C08X1 –C08_X1_CLK_GATE (0x04C)[8] = 1.[D]
    d. C08X4 –C08_X1_CLK_GATE[0] = 1

- *Note A: CLK_Control_A_0 impacts C10 frequency. And C01 is a divided down clock from C10.*
- *Note B: C08_X1 is divided down by 4 from C08_X4. This divider relationship is fixed.*
- *Note C: If using SPT periodic timer to wake FFE periodically, then SPT clock gate needs to be enabled. Set C01_CLK_GATE[9] =1too. If FFE is pushing sensor data into PacketFifo, then PKFB clock need to be enabled. Set C01_CLK_GATE[2] to 1 too. Accordingly, write C01_CLK_GATE = 0x20D.*
- *Note D: If FFE pushing sensor data into PacketFifo, then FFE to PacketFifo clock needs to be enabled. Set C08_X1_CLK_GATE[2] to 1 too. Accordingly, write C08_X1_CLK_GATE = 0x5.*

36.2 Setup PKFB clocks

The PacketFifo (PKFB) typically receives data from FFE, so PKFB clock enabling is very similar to FFE. PKFB's maximum frequency is 10 MHz

Important: Remember that PKFB Power domain needs to be powered up before enabling the clocks. By default, the PKFB power domain is SHUTDOWN out of reset.

1. Setup the Clock Source. Typically, the High-Speed OSC is selected as source.
    a. C01 - CLK_Control_A_1 (0x004) = 2'b00 (default)
    b. C08 – CLK_SWITCH_FOR_C (0x134) = 1'b0 (default)
2. Setup the Divider
    a. C01 – CLK_Control_A_0 (0x000) and C01_CLK_DIV (0x110).
        - Note that CLK_Control_A_0 impacts C10 frequency, and C01 is a divided down clock from C10.
    b. C08_X4 – CLK_Control_C_0 (0x010)
        - C08_X1 is divided down by 4 from C08_X4. This is fixed.
3. Enable the Clock Gate
    a. C01_CLK_GATE (0x040) [2] = 1. [E]
    b. C08_X1_CLK_GATE (0x04C)[3] = 1.

Note E: If using SPT periodic timer to wake FFE periodically, then SPT clock gate needs to be enabled. Set C01_CLK_GATE[9] =1 too.

SPI slave interface can also pop data from PKFB. SPI clock is from C00 – PAD16, and has a maximum frequency of 20 MHz.

Fabric can also push data into PKFB, which requires C41 to be configured into fabric design.

36.3 Setup Fabric clocks

Fabric (FB) is on-chip configurable logic. FB can configured with different designs once powered up. Once FB is powered down, or chip reset is applied, that design is lost and needs to reconfigure again.

Important: Remember that Fabric Power domain needs to be powered up before enabling the clocks. By default, the Fabric power domain is SHUTDOWN out of reset. Please refer to *EOS S3 Power Management User Guide* for details on how to power Fabric domain on.

Fabric (FB) has 3 inputs clocks. These input clocks are used to drive various FPGA designs as needed. This is dependent on the actual design configured and loaded into Fabric.

- C16 - Sys_Clk0
- C21 - Sys_Clk1
- C02 - Sys_Pclk

Also Fabric has 2 output clocks. These output clocks go from Fabric into the SOC. These clocks drive specific logic within the EOS S3.

- C40 (Fabric to Wishbone clock) – WB_CLK
- C41 (Fabric to PKFB clock) – Sys_Pkfb_Clk

C40 requires Wishbone Slave design loaded into Fabric. This clock drives the Wishbone interface, which allows M4 to write/read the Fabric via that interface. The maximum frequency of this clock is 10 MHz

C41 requires the fabric design which pushes data into the PacketFifo. This clock drives the Fabric to PKFB interface. The maximum frequency of this clock is 10 MHz. Note: Remember to turn off C41 before powering down PacketFifo domain.

Important: C40/C41 is NOT output clock to EOS S3 PADS. General Fabric IO can be configured instead to be an output clock to PAD if needed.

M4 can write Fabric via an AHB to Wishbone Bridge. The Fabric requires to be configured with a fabric design with Wishbone slave design to generate C40.

Enable the Fabric Configuration clocks:

1. Power up the Fabric domain. Please refer to *EOS S3 Power Management User Guide.*
2. Program divider and clock gate for C09
    a. C09_CLK_DIV (0x114) – Divider and root clock gate for C09.
3. Enable the clock gates
    a. C09_CLK_GATE(0x118) = 0x6.

Once the Fabric is configured, enable the Fabric input clocks:

1. Program the dividers and clocks source for fabric input clocks as desired.
    a. C16 – CLK_Control_F_0 (0x020) / CLK_Control_F_1 (0x024).
    b. C21 – CLK_Control_I_0 (0x034) / CLK_Control_F_1 (0x038).
    c. C02 – CLK_Control_B_0 (0x008) / CLK_Control_B_1 (0x00C).
2. Enable the Clock Gates
    a. Ensure that CLK_DIVIDER_CLK_GATING(0x0124) [1] (C02) ,[5] (C16) , [8] (C21) are all set to 1. This is the master clock gate going into each divider (B, F, I).
    b. C16_CLK_GATE(0x0x64)[0] = 1 to enable Sys_Clk0.
    c. C21_CLK_GATE(0x070)[0] = 1 to enable Sys_Clk1.
    d. C02_CLK_GATE (0x044) [1] = 1 to enable Sys_Pclk.

36.4 Setup Voice Subsystem clocks

In the Voice Subsystem, there are six Audio domains – AD0, AD1, AD2, AD3, AD4, and AD5. These Audio power domains are described in more detail in *EOS S3 Power Management User Guide*.

Important: Remember that Audio power domains need to be powered up before enabling the clocks. By default, the Audio power domains are SHUTDOWN out of reset. Please refer to *EOS S3 Power Management User Guide* for details on how to power Fabric domain on.

Each of the Audio domains has the following clocks:

- AD0 - AHB DMA Clock – Audio DMA logic (C10_P5)
- AD1 - PDM Left Clock – PDM-to-PCM conversion filter for left channel (C30_P0)
- AD2 - PDM Right Clock – PDM-to-PCM conversion filter for right channel (C30_P1)
- AD3 - LPSD Clock – Low Power Sound Detect (C30_P3)
- AD4 - I2S Master Clock – I2S Audio data receiver block (C30_P2)
- AD5 - APB Clock for Audio Configuration controls (C09_P0)

1. Setup the Clock Source as OSC.
    a. CLK_Control_A_1 (0x004) = 2'b00 (default) for C10
    b. CLK_SWITCH_FOR_G (0x144) = 0 (default) for C30
2. Setup the Divider based on OSC frequency.
    a. CLK_Control_A_0 (0x000)
    b. C09_CLK_DIV(0x114) – Divider and root clock gate for C09.
    c. CLK_Control_G_0 (0x028)
3. Enable the Clock Gate
    a. C10_FCLK_GATE (0x050)[5] =1
    b. C09_CLK_GATE (0x11C)[0] =1
    c. C30_C31_CLK_GATE (0x120)[3:0] = 0xF

36.5 Setup SDMA clocks

The System DMA (SDMA) domain is used to transfer data between memories on EOS S3. The SDMA can transfer data between M4SRAM, FFE memories, and Fabric. SDMA runs off the C01_P6 clock.

Important: Remember that SDMA power domain needs to be powered up before enabling the clocks. By default, the SDMA power domain is SHUTDOWN out of reset. Please refer to *EOS S3 Power Management User Guide* for details on how to power Fabric domain on.

1. Setup the Clock Source as OSC
    a. CLK_Control_A_1 (0x004) = 2'b00 (default)
2. Setup the Divider
    a. CLK_Control_A_0 (0x000)
    b. C01_CLK_DIV (0x110)
3. Enable the Clock Gate
    a. C01_CLK_GATE (0x040)[1]= 1 to enable SDMA SRAM Clock
    b. C01_CLK_GATE (0x040)[6]= 1 to enable SDMA Clock

If the SDMA is transferring data into Fabric or FFE memories, that those domains need to be powered on and the corresponding clocks enabled.

36.6 Setup M4 clocks

The M4 power domain includes the M4-F processor. The M4 Processor uses the following clocks.

- C10
- CS – Serial Wire Debug Clock for debugger

Out of reset, M4 power domain is powered ON. And the clocks are enabled on by default. The M4-F processor speed is running at 12.8 MHz , which is OSC speed 76.8 MHz divided by 6 (CLK_Control_A_0 (0x000) = 0x204.

1. Setup the Clock Source can be OSC or RTC (32Khz)
    a. CLK_Control_A_1 (0x004) = 2'b00 for OSC or 2'b01 for RTC.
2. Setup the Divider
    a. CLK_Control_A_0 (0x000) = 0x204 (clock source divided by 6)
3. Enable the Clock Gate
    a. C10_FCLK_GATE (0x050)[6, 4:0] set to all 1.

M4 can run from OSC or RTC (32KHZ clock source). The PMU can also program the OSC to automatically shut down when M4 shuts down. *Please refer to EOS S3 Power Management User Guide for details.*

36.7 Setup A1 CfgSM Clocks

A1 domain contains the SPI master and Configuration State Machine(CfgSM). The CfgSM contains a DMA master that controls the SPI master to download of boot code in Wearable mode (bootstrap PAD 20 = 0). After boot, the DMA can be configured to do burst reads from external SPI flash. CfgSM requires the following clocks:

- C01 – configuration clock for SPI Master / CfgSM
- C02 – SPI master clock

Below are the relevant register controls to re-enable the A1 clocks, in case A1 is powered down and clock gated.

1. Setup the Clock Source from OSC
    a. CLK_Control_A_1 (0x004) = 2'b00 (C10 -> C01)
    b. CLK_SWITCH_FOR_B (0x130)= 1'b0 (C02)
2. Setup the Divider
    a. CLK_Control_A_0 (0x000) for C10
    b. C01_CLK_DIV (0x110) for C01
    c. CLK_Control_B_0 (0x008) for C02
3. Enable the Clock Gate
    a. C01_CLK_GATE (0x040)[4] to 1 (default)
    b. C02_CLK_GATE(0x044)[0] to 1 (default)
    c. Assume CLK_DIVIDER_CLK_GATING (0x124)[0] = 1 (default)

The C02 clock (Fssi_clk) can be further divided down to become the SPI Master clock output using the SPI_BAUDR register below. Note that the SPI_BAUDR can divide down the C02 Clock by factors of 2. Remember to enable the IOMUX programming for SPI Master clock pad. Please refer to EOS S3 IOMUX User Guide.

Table 36-1: SPI_BAUDR Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x40007014	SPI_BAUDR	15:0			
	SCKDV_15_1	15:1	RW	0x0	SSI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: Fsclk_out = Fssi_clk/SCKDV where SCKDV is any even value between 2 and 65534. For example: for Fssi_clk = 3.6864MHz and SCKDV =2 Fsclk_out = 3.6864/2 = 1.8432MHz
	SCKDV_0	0:0	RO	0x0	From description of "SCKDV_15_1": "The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. " "SCKDV" is broken into "SCK_DV_15_1" and "SCK_DV_0" purely by ASIC verification for the purpose of setting "RW/RO" property for bit 15_1/0 separately in order to pass test case.

36.8 Setup Analog-to-Digital Convertor

The Analog-to-Digital-Converter (ADC) requires a 1 MHz input clock to operate. C19 clock becomes the input clock to the ADC. The ADC requires a delta-sigma clock frequency of 200 KHz – maximum 2 MHz, with the typical being 1 MHz. Please refer to Table 36-2 below for ADC register details.

Table 36-2: ADC Registers

Address	Register Name	Bit Field	R/W Access	Default	Description
0x40005A00	ADC_Out	31:0			
	OUT	11:0	RO	0x0	12-bit conversion output
0x40005A04	ADC_Status	31:0			
	EOC	0	RO	0x0	End-of-conversion. Rises when data is valid.
0x40005A08	ADC_Control	31:0			
	SOC	0	RW	0x0	Asynchronous start-of-conversion. Needs to rise and be held high for each conversion.
	SEL	1	RW	0x0	Channel Selection 0: CH0 1: CH1
	MEAS_EN	2	RW	0x0	Battery measurement enable

ADC Conversion Procedure:

1. Select which ADC channel input 0 or 1.
    a. Program ADC_Control[1] = 0 or 1 depending on ADC channel selected.
2. Setup a 1MHz clock to ADC (C19).
    a. Select the clock source to be HSOSC. Program 0 to CLK_Control_H_1 (0x030) to select the OSC source.
    b. Based on the OSC frequency, program the clock divider to CLK_Control_H_0 (0x028).
        i. If OSC prog[11:0] = 0x92d (default), then OSC is running at 77.07 Mhz. Please refer to Change the Oscillator Frequency for OSC frequency equation. Then program divider 77 into CLK_Control_H_0 [8:0] = 77 – 2 = 75 = 0x4B, to get 1 MHz ADC input clock (C19).
    c. Enable the clock gates.
        i. Program 1 to CLK_DIVIDER_CLK_GATING (0x124) bit[7]. Enable the C19 root.
        ii. Program 1 to C19_CLK_GATE(0x06C). Enable the path clock to ADC.
3. Write 0 to ADC_Control[0] SOC. This will disable the Start-of-Conversion signal to ADC.
4. Apply the voltage to ADC channel 0 or 1. This should match step 1 above.
5. Write 1 to ADC_Control[0] SOC. This will enable the Start-of-Conversion signal to ADC.
6. Poll until ADC_Status[0] EOC is 0. This signals End-of-Conversion is started when low.
7. Poll until ADC_Status[1] EOC is 1. When 1, this signals that ADC has reached End-of-Conversion of Analog signal into digital value.
8. Read out ADC_Out[11:0] for the digital value.

Important: The ADC's EOC signal will be held high until the SOC goes high again. This is why Step 6 is required, otherwise, if step 6 is skipped, then step 8 will provide a false read out of ADC.

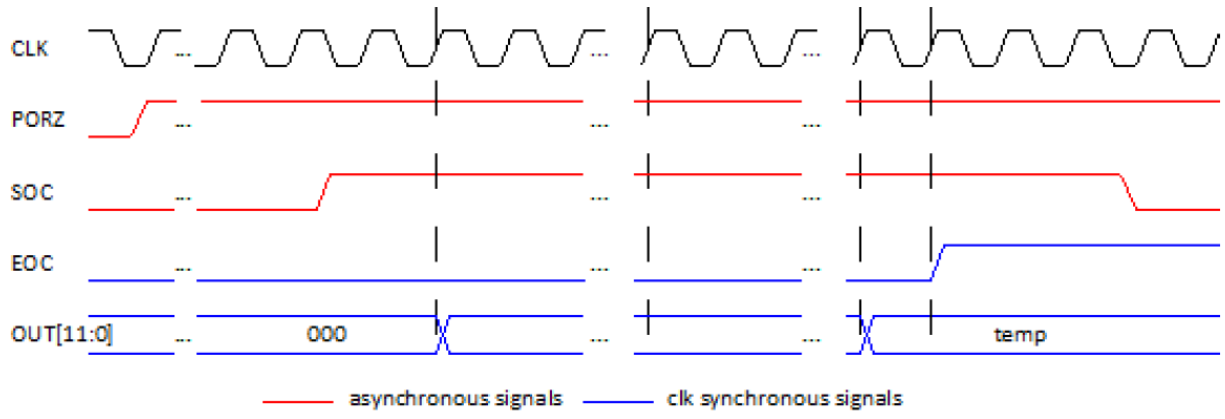

Figure 36-1: ADC Procedure Signaling

36.9 Setup I2S Slave Clock

I2S Slave clock receives its input clock (C32) from PAD 31. Remember that the IOMUX_PAD_31_CTRL/IOMUX_I2S_WD_CLK_IN_SEL needs to configure to receive clock on PAD31. The maximum frequency of C32 is 10 MHz. Also, the I2S is configured by C01. Below is the procedure to enable the I2S clocks.

1. Setup the Clock Source as OSC
    a. CLK_Control_A_1 (0x004)= 2'b00 (default)
2. Setup the Divider
    a. CLK_Control_A_0 (0x000)
3. Enable the Clock Gates
    a. C01_CLK_GATE(0x040)[5] = 1 enables the APB configuration clock to I2S.
4. Setup the IOMUX to allow C32 as input from PAD31
    a. IOMUX_PAD_31_CTRL = 0x220. See

b. Table 36-3: IOMUX_PAD_32_CTRL Register below.
c. IOMUX_I2S_WD_CLKIN_SEL = 1. Table 36-4: IOMUX_I2S_WD_CLKIN_SEL Register

Table 36-3: IOMUX_PAD_32_CTRL Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x40004C7C	IOMUX_PAD_31_CTRL	31:0			
	PAD_31_FUNC_SEL	1:0	RW	0x0	Functional selection for IO output Refer to IO Mux
	PAD_31_CTRL_SEL	4:3	RW	0x0	Control selection for IO output 0x0 = A0 registers (as defined below) 0x1 = Others 0x2 = Fabric
	PAD_31_OEN	5	RW	0x1	Active low output enable 0x1: driver disabled 0x0: normal operation
	PAD_31_P	7:6	RW	0x0	Driver state control 0x0: Z 0x1: pull up 0x2: pull down 0x3: keeper
	PAD_31_E	9:8	RW	0x1	Drive Strength 0x0: 2mA 0x1: 4mA 0x2: 8mA 0x3: 12mA
	PAD_31_SR	10	RW	0x0	Slew Rate 0x1: Fast 0x0: Slow (half frequency)
	PAD_31_REN	11	RW	0x0	Receive enable 0x1: enable 0x0: disable
	PAD_31_SMT	12	RW	0x0	Schmitt Trigger 0x1: enable 0x0: disable

Table 36-4: IOMUX_I2S_WD_CLKIN_SEL Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x40004DA0	IOMUX_I2S_WD_CLKIN_SEL	31:0			
	I2S_WD_CLKIN_SEL	0	RW	0x0	Sel 0: 0 1: pad #23

36.10 Setup M4 Peripheral Clocks

The UART, M4 Timer, Watchdog clocks are driven by C11. The maximum allowed frequency for C11 is 10 MHz

1. Setup the Clock Source as OSC
    a. CLK_SWITCH_FOR_D (0x) = 0 (default)
2. Setup the Divider based on OSC frequency
    a. CLK_Control_D_0 = 0x20E (divide by 16 , default)
3. Unlock the Clock Gate register
    a. MISC_LOCK_KEY_CTRL = 0x1acce551
4. Enable the Clock Gate
    a. C11_CLK_GATE = 1. This register is write-protected. Please do Step 3 first to unlock access to this register.

Chapter 37. How to Bring Clock Out to Debug Pin

EOS S3 can be programmed to bring out internal clocks to a PAD. This feature aids for the debug of clock programming to ensure clock is running and programmed frequency is as expected.

On chip, an 8-bit debug monitoring bus (debug_mon[7:0]) allows certain internal signals for CRU, PMU, Audio, or FFE to be brought to eight output PADs. The debug_mon[7:0] can be brought out to a variety of pads. *Please refer to EOS S3 IOMUX User Guide for more details on PAD assignments for debug monitoring bus.*

Figure below shows the debug muxes for selecting a clock. Notice that the clocks can be selected to multiple pads as seen on the right side of diagram. The bulk of the clocks C00 (SPI Slave Clk), C01, C02, C08x4, C08x1, C09, C10, C11, CS (SWD Clk), C19, C20, C21, C23, C31 can be sent to debug_mon[7] on PAD 13, 33 or 42. C32 and C30 can be output to debug_mon[6] on PAD 12, 32, or 4.

Here is an example procedure to bring the clock C02 out to PAD 13.

1. Program MISC_ SUBSYS_DEBUG_MON_SEL (0x40005004) to 0 which selects A0.
2. Program MISC_A0_DEBUG_MON_SEL (0x40005008) to 1 which selects CRU
3. Program the IO_MUX_PAD_13_CTRL (0x40004c34) bit[1:0] to 0x2. *Please refer to EOS S3 IOMUX user Guide for more details for bit encoding.*
4. Program CRU_DEBUG_SELCT (0x40004108) to 0x3.
5. Check that PAD 13 is toggling like a clock as expected.

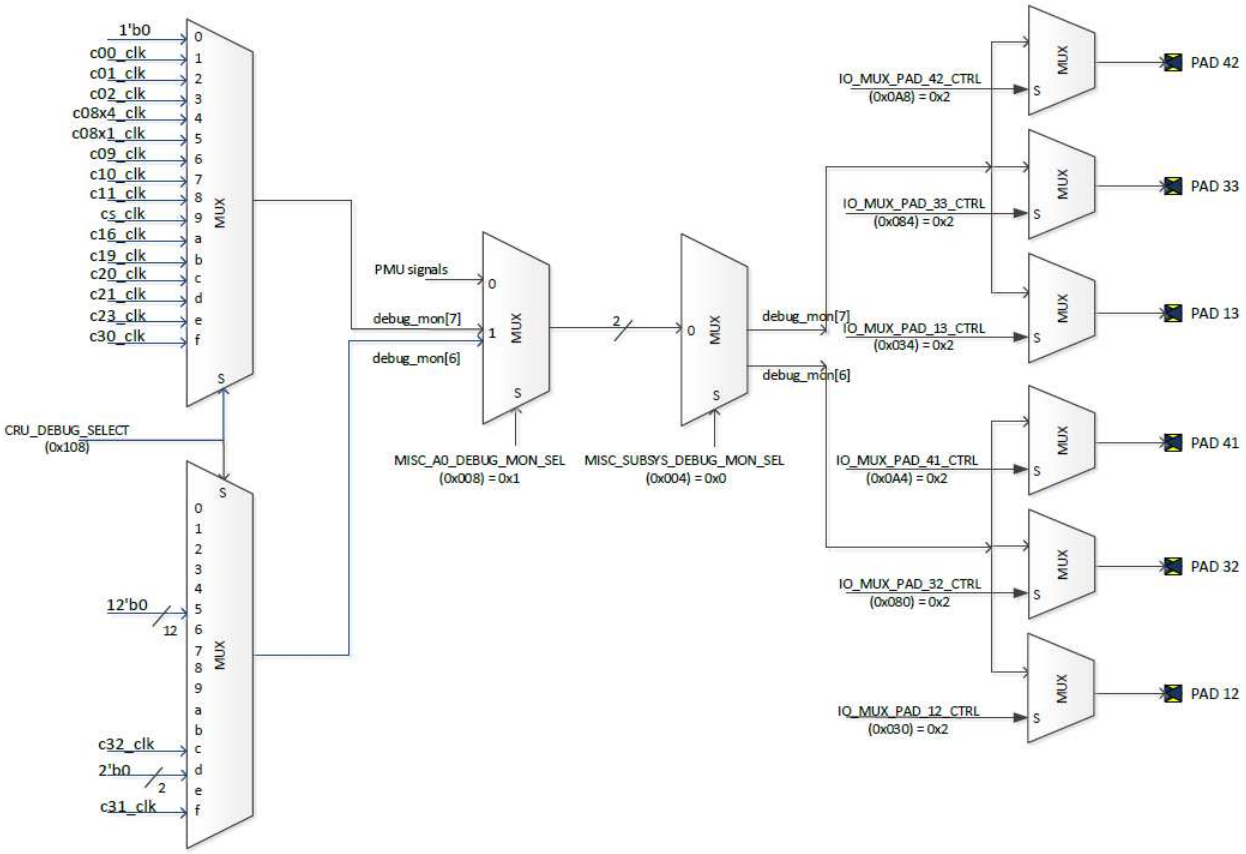

Figure 37-1: Clock Debug Multiplexers

Chapter 38. Software Resets

Each functional domain has a software reset associated with it. CRU contains registers to issue soft resets to those functional domains. Write 1 to the appropriate bit to assert the functional domain reset. Write 0 to appropriate bit to release the reset.

Table 38-1: CRU Software Reset Register

Address	Register Name	Bit Field	R/W Access	Default	Description
0x080	PF_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:7			
	RESERVED	6	RW	0x0	Reserved, Used as General Purpose Register
	PF_Perpherial_SW_RESET	5	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.
	PF_ASYNC_FIFO_0_SW_RESET	4	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. (R08_P3 as well)
	PF_FIFO_8K_SW_RESET	3	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.
	PF_FIFO_2_SW_RESET	2	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.
	PF_FIFO_1_SW_RESET	1	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.
	PF_FIFO_0_SW_RESET	0	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.
0x084	FFE_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:2			
	FFE_0_X1_SW_Reset	1	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually.

Address	Register Name	Bit Field	R/W Access	Default	Description
					(R01_P3_FFE as well)
	FFE_0_X4_SW_Reset	0	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. (R01_P3_FFE as well)
0x088	FB_SW_RESET	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:6			
	FB_C21_Domain_SW_Reset	5	RW	0x1	1'b1 : Enable the Software Reset. FW need to disable it manually.
	FB_C16_Domain_SW_Reset	4	RW	0x1	1'b1 : Enable the Software Reset. FW need to disable it manually.
	RESERVED	3	RW	0x1	Reserved, Used as General Purpose Register
	FB_C09_Domain_SW_Reset	2	RW	0x1	1'b1 : Enable the Software Reset. FW need to disable it manually.
	RESERVED	1	RW	0x1	Reserved, Used as General Purpose Register
	FB_C02_Domain_SW_Reset	0	RW	0x1	1'b1 : Enable the Software Reset. FW need to disable it manually.
0x08C	*A1_SW_RESET*	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:3			
	CfgSM_SW_RESET	2	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. -> This is used to Reset the CfgSM/SPI Master and Related FIFO, DMA and AHB Master

Address	Register Name	Bit Field	R/W Access	Default	Description
	RESERVED	1	RW	0x0	Reserved, Used as General Purpose Register
	SPT_SW_RESET	0	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. -> This is used to Reset the SPT
0x090	*AUDIO_MISC_SW_RESET*	31:0			Once Program the SW Reset Bit to 1, the corresponding reset will be asserted immediately. Once Program the SW Reset Bit to 0, the corresponding reset will be de-asserted synchronous even the corresponding clock is not running. (Turn off by Clock gating cell)
	RESERVED	31:8			
	I2S_SW_RESET	7	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For I2S Power Domain. Note: It will only reset the AHB interface R01, but it will not reset R32 path. Suggest to power down, then power on I2S if Software Reset is needed
	DMA_SW_RESET	6	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For DMA Power Domain.
	AD5_SW_RESET	5	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD5 Power Domain.
	AD4_SW_RESET	4	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD4 Power Domain.
	AD3_SW_RESET	3	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD3 Power Domain.
	AD2_SW_RESET	2	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD2 Power Domain.
	AD1_SW_RESET	1	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD1 Power Domain.

Address	Register Name	Bit Field	R/W Access	Default	Description
	AD0_SW_RESET	0	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For AD0 Power Domain
0x094	*FB_MISC_SW_RST_CTL*	31:0			
	RESERVED	31:2			
	PFAFIFO1_SW_RESET	1	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For R41
	AHBWB_SW_RESET	0	RW	0x0	1'b1 : Enable the Software Reset. FW need to disable it manually. ==> For R40

Appendix A. Terminology and Conventions

A.1. Glossary of terms

The following table explains the terminology, conventions, abbreviations, and acronyms used.

Table: Glossary of Terms

Term	Description
A0N	Always ON power domain (does not turn off) Include AP interface (TLC), SYS_RSTn, GPIO, ADC, PMU, LDO, slave I2S
A1	Configuration DMA, Master SPI
AD0	Audio domain 0 Voice sub-system audio DMA power domain/clock
AD1	Audio domain 1 Voice sub-system PDM_LEFT power domain/clock
AD2	Audio domain 2 Voice sub-system PDM_RIGHT power domain/clock
AD3	Audio domain 3 Voice sub-system LPSD power domain/clock
AD4	Audio domain 4 Voice sub-system I2S_MASTER power domain/clock
AD5	Audio domain 6 Voice sub-system top level (APB I/F) power domain/clock
ADC	Analog to Digital Converter
AHB	ARM Advanced High-performance Bus – typically this bus is running at high speed. Modules such as CPU, DSP, co-processors, memory, FIFO, etc. all interface to this bus.
AIP	Analog IP block
AON	Always ON power domain or circuit inside AON power domain
APB	ARM Advanced Peripheral Bus – typically “slow” peripheral bus for interface with external devices.
APC	Analog power control, a part of the PMU
Bit-banding	Maps a complete word of memory onto a single bit in the bit-band region.
Bootstrap pin	to configure the operating (or pin configuration) mode, tie the pin to high (VCCIO) or low (GND) during period when external reset (SYS_RSTN) is asserted .
CFGSM	Configuration State Machine

Term	Description
CODEC	Coder/Decoder – in EOS, CODEC encodes or decodes digital audio signals
CfgDMA	Configure DMA module
CfgSM	Configuration State Machine group is made up of the CFG_SM, SPI0_Master, and the CFG_DMA controller for DMA transfer from flash memory through the SPI_Master port
CM	Depends on context, Is either 1) Communication Manager 2) Code memory in FFE
Coremark	Benchmark standard.
CoreSight	ARM debugging and trace technology. The infrastructure for monitoring, tracing, and debugging a complete system on chip.
Cortex-M4F	ARM deeply embedded processor with floating point arithmetic functionality; low-power, low gate count, low interrupt latency, and low-cost debug
CRU	Clock Reset Unit
DAP	ARM Debug Access Port – A TAP block that acts as an AMBA, AHB or AHB-Lite, master for access to a system bus. The DAP is the term used to encompass a set of modular blocks that support system wide debug.
DBG	Debug Controller
DM0, DM1	Data memory in FFE
DMA	Direct Memory Access
DST	Disable Self Time
DTM	
DWT	ARM M4 Data Watchpoint and Trace – part of the ARM debug architecture
eFuse	Electronic fuse – options that are factory programmed by QuickLogic
Fabric	Another name for FPGA
FB	Refers to Fabric (FPGA) module, its power domain, its clock
FBIO	I/O belonging to the FPGA fabric.
FFE	Flexible Fusion Engine module/ power domain/clock Ultra-low power µDSP-like processor that performs routine tasks
FCLK	Fast Clock
FPB	ARM M4 Flash Patch and Breakpoint – A set of address matching tags, which reroute accesses into flash to a special part of SRAM. This permits patching flash locations for breakpointing and quick fixes or changes.

Term	Description
FPU	Floating Point Unit
HSO	High Speed Oscillator
I2S	Inter IC Sound
ITM	ARM Instrumentation Trace Macrocell – part of the ARM debug architecture
JTM	Delta-Sigma ADC
Light Sleep	SRAM low power mode with lower leakage advantage but access time penalty.
LPMF	SRAM (Footer) circuit to enable light sleep mode.
LPMH	SRAM (Header) circuit to enable deep sleep mode.
LPSD	Low Power Sound Detect (acoustic activity detection IP from Sensory Inc.) For information refer to http://www.sensory.com/
MPU	Memory Protection Unit – hardware to protect region of memory
MS0	M4 SRAM Instance M4S0–M4S3 power domain/clock
MS1	M4 SRAM Instance M4S4–M4S7 power domain/clock
MS2	M4 SRAM Instance M4S8–M4S11 power domain/clock
MS3	M4 SRAM Instance M4S12–M4S15 power domain/clock
NVIC	Nested Vectored Interrupt Controller – Provides the M4F processor with configurable interrupt handling abilities.
OSC	Oscillator
PCM	Pulse Code Modulation – for digital audio in EOS
PDM	Pulse Density Modulation – for digital audio in EOS
PF	Packet FIFO module/ power domain/clock
PIF	FPGA Programming Interface
PKFB	Packet FIFO Bank
PMU	Power Management Unit
PMUT	Power Management Unit Timer – simple periodic timer
POR	power-on reset
RTC	Real-Time Clock (not calendar)
SCS	The ARMv7-M System Control Space
SDMA	System DMA
SM	The term SM is used in some register descriptions to denote sensor memory.
SM0, SM1	Sensor Manager - there are two in the S3

Term	Description
SPI	Serial Peripheral Interface – synchronous serial interface with separate clock and unidirectional data signals and chip select. A SPI module is either master (generates clock) or slave (receives clock). There are three SPI modules in the S3.
SPI_0_Master	The SPI interface in the Sensor Processor
SPI_1_Master	The SPI interface used to load data from external flash memory
SPI_1_Slave	The SPI interface in the Communications Manager block, used to communicate with an Application Processor host..
SPT	Simple Periodic Timer Generates 1ms interval counter; requires count error compensation for accuracy.
STM	Software Test Mode – pin used to enter test mode
SW-DP, also known as SWD	ARM Serial-Wire Debug Port – An optional external interface for the DAP that provides a serial-wire bidirectional debug interface.
SWV	Serial-Wire Viewer. Debug tool.
TAP	Test Access Port – The collection of four mandatory and one optional terminal that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are **TDI**, **TDO**, **TMS**, and **TCK**. The optional terminal is **TRST**. This signal is mandatory in ARM cores because it is used to reset the debug logic.
Timer1	Also known as the M4 Timer
TLC	Top Level Controller
TLS	Transparent Light Sleep – SRAM low power mode; lower pro: leakage, con: performance penalty to get in/out of TLS mode.
TPA	ARM Trace Port Analyzer – A hardware device that captures trace information output on a trace port. This can be a low-cost product designed specifically for trace acquisition, or a logic analyzer.
TPIU	ARM M4 Trace Port Interface Unit – part of the ARM debug architecture
VoiceQ	VoiceQ is a voice sensing technology by Knowles Corporation http://www.knowles.com/
WIC	wake-up interrupt controller
Wishbone bus	An SoC IP interconnect bus
WU	wake-up

A.2. Structure of a register definition

The following table is a typical register description. The first column of the heading is the address offset within the module. Second column of the heading is the register name, while body is bit field name, Third column of the heading is the bit width of the register (typically 32-bit or [31:0]), while body is bit number(s) of the bit field. Fourth column indicates whether the register can be read (R) or written (W) by M4. More details below. Fifth column is the default value after reset.

Table: Sample Register bit Definition

0x030	Power_Down_ Scheme				
	Bit Field	Bit	R/W/C	Default	Description
	RESERVED	31:8	RO	–	Reserved
	M4M4S0_PD	7	RW	0x0	1'b1: If M4 and M4S0 PD event happen at same time, then M4 and M4S0 will put into power saving mode at same time
	…	6			…
	SRAM_PD	5	RW	0x0	1'b: If more than one SRAM (M4S1 – M4S15, NOT include M4S0) PD event happens at same time, these SRAMs will power down in parallel. 1'b0: If more than one SRAM (M4S1 – M4S15) PD event happens at same time, these SRAMs will power down in the following priority. M4S1→M4S2→…→M4S14→ M4S15 (M4S1 has the highest priority and M4S15 has the lowest priority)
	…	4			…
	M4M4S0_WU	3	RW	0x0	1'b1: If M4 and M4S0 WU events happen at same time, then M4 and M4S0 will wake up at same time
	…	2			…
	SRAM_WU	1	RW	0x0	1'b1: If more than one SRAM(M4S1 – M4S15, NOT include M4S0) WU events happen at same time, these SRAMs will wake up in parallel. 1'b0: If more than one SRAM (M4S1 – M4S15) WU event happens at same time, these SRAMs will wake up in the following priority. M4S1→M4S2→…→M4S14→ M4S15 (M4S1 has the highest priority and M4S15 has the lowest priority)

Table: Read/Write/Clear definitions

R/W/C	Definition	M4 read	M4 write	Hardware
RW	Read / Write	returns value previously written	Control or Data bit	Uses the value
RO	Reserved bit	this bit is reserved or undefined	No effect or must always write 0	Reserved / Undefined
RO	Read Only	Status or value bit	No effect or must always write 0	Value set by hardware
RHW	Read / Hardware Write	Status or value bit	No effect or must always write 0	Value set by hardware
RWHC	Read / Write / Hardware Clear	Control bit returns value previously written, or may be cleared after an hardware event	Control bit	M4 writing 1 will initiate hardware event which will clear this bit after event is done.
RW1C	Read / Write 1 to Clear	Status / flag bit; typically 0 means default state or flag is not set (event has not occurred)	Writing 0 has no effect Writing 1 will clear flag	Value is reflects hardware status/flag M4 writing 1 will clear the flag
RW1S	Read / Write 1 to Set	Status / flag bit; typically 0 means default state	Writing 0 has no effect Writing 1 will set flag or control	Value is reflects hardware status/flag M4 writing 1 will set the flag
WO	Write Only	Undefined	Control bit	Uses the value

Number representation:

2'b11 = 2 bit binary number (decimal equivalent is 3)

4'hf = 4 bit hexadecimal number (decimal equivalent is 15)

0x0FF = 3 nibble hexadecimal number (decimal equivalent is 255)

31:0 = 31 down to 1, typically represents bit index.

Appendix B. Reference Documents

These QuickLogic documents are referenced:

- EOS S3 DS (datasheet)

See ARM documentation for more information:

- DDI0183G UART PL011, r1p5, TRM
- DDI0403C ARMv7-M Architecture Reference Manual, C_errata_v3
- DDI0439B Cortex-M4, r0p0, TRM
- DDI0479C Cortex-M system design kit, r1p0, TRM
- IHI0029D CoreSight Architecture Spec, v2.0
- IHI0031C ARM Debug Interface Architecture Specification ADIv5.x

Other referenced materials:

- JEDEC JEP 106AR (Revision of JEP106AQ), http://www.jedec.org/standards-documents/results/jep106, May 2015