QuickLogic EOS S3 Voice and Sensor Processing Platform Datasheet

Version 1.0

Contact Information

Phone: (408) 990-4000 (US)

+(44) 1932-21-3160 (Europe)

+(886) 26-603-8947 (Taiwan)

+(86) 139-0517-5302 (China)

+(81) 3-5875-0547 (Japan)

+(82) 31-601-4225 (Korea)

E-mail: info@quicklogic.com

Sales: America-sales@quicklogic.com

Europe-sales@quicklogic.com

Asia-sales@quicklogic.com

Japan-sales@quicklogic.com

Korea-sales@quicklogic.com

Support: www.quicklogic.com/support

Internet: www.quicklogic.com

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

Contents

Figures

Tables

QuickLogic EOS S3 Voice and Sensor Processing Platform Highlights

Multi-Core, Ultra Low Power Sensor & Audio Processing Platform Enabling Always-On, Always-Aware Application

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 enable 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 frequency
- 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 recognition, etc.)

Third-Generation Flexible Fusion Engine

- 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 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

Communication Manager

Communicates with host applications processor through SPI Slave interface of 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

On-Chip Programmable Logic

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

Operating System Support

Android 6.0 compliant

Additional Features

- On-device circuit to support 32.768 kHz clock or crystal oscillator
- Thirty-one different power islands for granular management of system power
- Power Management Unit (PMU) for minimizing power in all conditions (idle, deep sleep, and shut down)
- SPI Master, SPI Slave, I2S Master, I2S Slave, and I2C interfaces
- 12-bit Δ Analog-to-Digital Converter (ADC)
- 2-pin Serial Wire Debug (SWD) port
- System DMA engine for efficient data movement
- Dual Low-Dropout (LDO) regulators for on-chip voltage regulation

1. Functional Overview

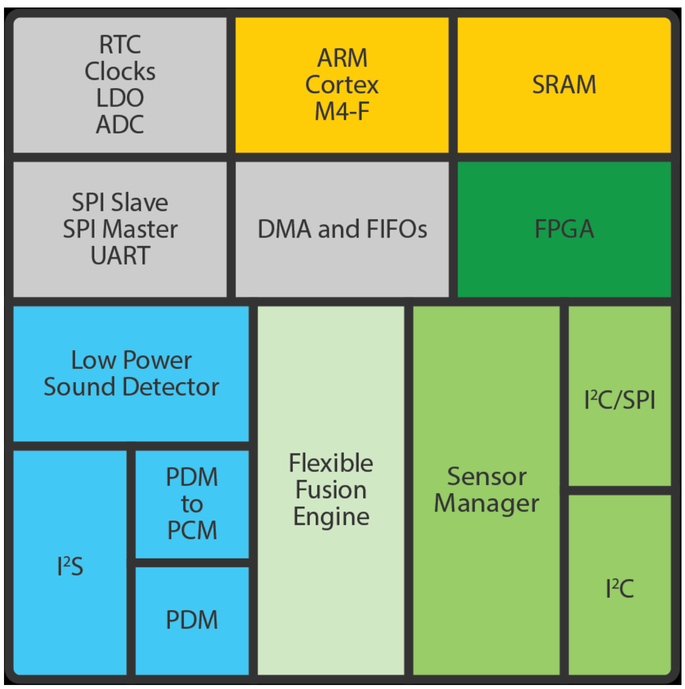
The EOS S3 platform is a multi-core, ultra-low power sensor processing system designed for mobile market applications such as smartphone, wearable, and Internet of Things (IoT) devices. The core of the QuickLogic EOS S3 platform is its proprietary µDSP-like Flexible Fusion Engine (FFE). To complement the FFE, the EOS S3 platform also includes a Cortex M4-F subsystem that enables higher level, general purpose processing.

This multi-core architecture enables smartphone application processors to offload real-time, always-on sensor computational requirements to the EOS S3 platform. The multi-core approach and multiple power islands allow the EOS S3 platform to process sensor data and run sensor fusion algorithms in the most efficient manner possible for both processing and power.

1.1. EOS S3 Voice and Sensor Processing Platform Architecture

The following figure shows a system level architecture diagram that highlights the major functional blocks of the EOS S3 voice and sensor processing platform. More detailed block diagrams are provided later in this data sheet that highlights the available datapath options, clock and power domain partitions.

Figure 1: EOS S3 Voice and Sensor Processing Platform Architecture



The following table lists the top-level features. This feature set enables the EOS S3 platform to support use cases in the smartphone and wearable device markets.

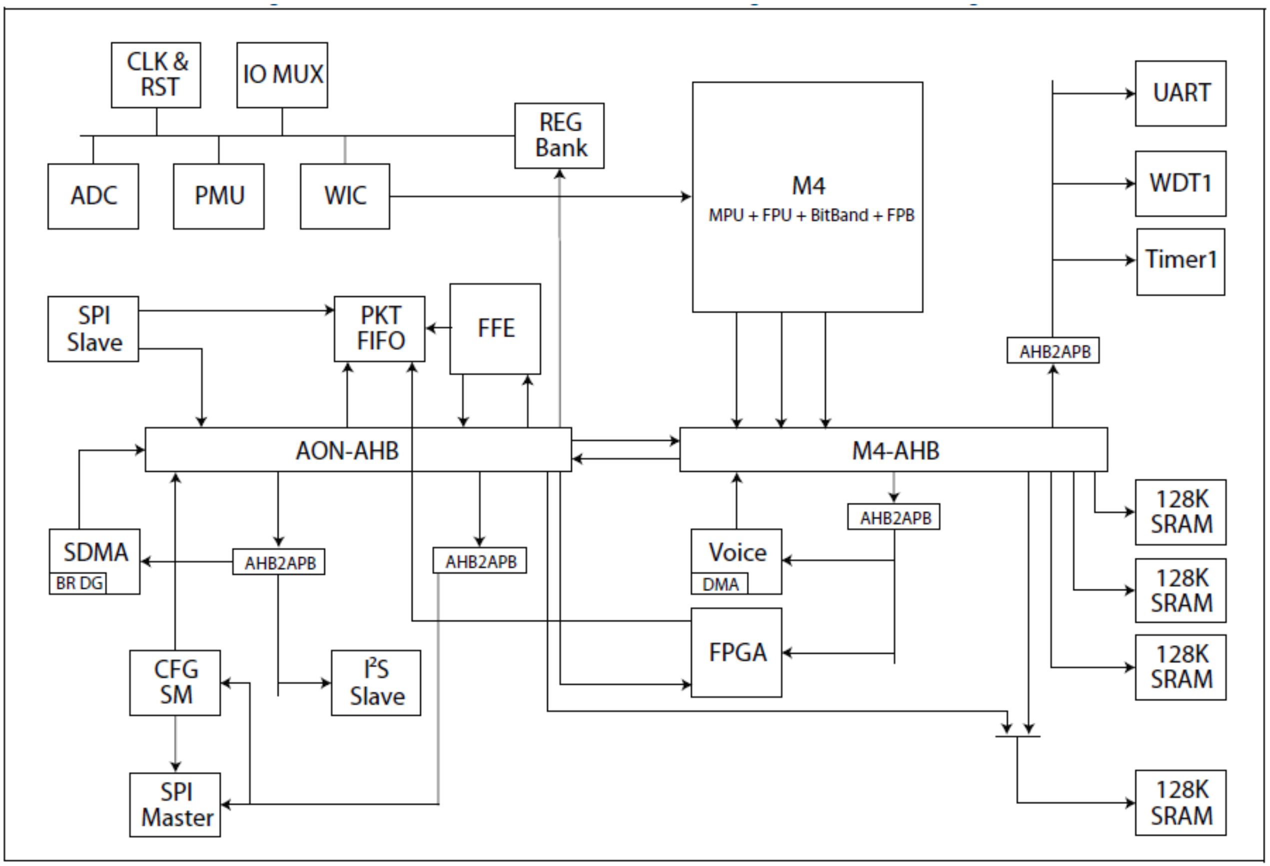
Table 1: EOS S3 Platform Supported Features

Feature	Details
M4-F Subsystem	Cortex M4-F processor with floating point unit support (M4-F) Embedded SRAM (up to 512 KBytes) for code and data memory Vectored interrupt support Wakeup interrupt controller 2-pin SWD port
FFE	50 KB control memory 16 KB data memory Single cycle MAC
Digital Microphone Support	I^2S microphone PDM microphone On-chip PDM-to-PCM conversion Hardware bypass path for PDM interface to host application processor and/or Voice CODEC Integrated LPSD from Sensory
Packet FIFOs Batching Memory	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
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 Firmware and hardware-initiated sleep entry Wake-up triggers via internal and external events
On-chip Programmable Logic	2,400 effective logic cells with 64 Kbits of RAM, 8 RAM FIFO controllers and 2 GPIO banks Supports SPI Slave configuration Supports reconfiguration from M4-F Supports five clocks
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 500 ms
High Frequency Clock Source	Programmable frequency (2 MHz to 80 MHz) for better frequency resolution Calibrated output (using 32 kHz input) Startup time of 410 µs 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 1 msec
I^2C Master and Configurable I^2C/SPI Interface	I^2C Master and SPI Master with programmable clock pre-scaler Option to disable multi-master support and slave-inserted wait for shorter SCL cycles Configurable for two I^2C Masters or one I^2C 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 I^2S 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 8-bit GPIO controller
ADC	Low sampling rate (12-bit)
LDO Regulators	On-chip LDO for system logic Separate on-board LDO for memory
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. I^2C Master used for I^2C 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/Os and 2 VCCIO banks) 64-ball BGA (3.5 mm x 3.5 mm x 0.71 mm) (46 user I/Os and 2 VCCIO banks)

The following figure shows a more detailed view of the datapaths within the EOS S3 voice and sensor processing platform.

Figure 2: EOS S3 Voice and Sensor Processing Platform Block Diagram



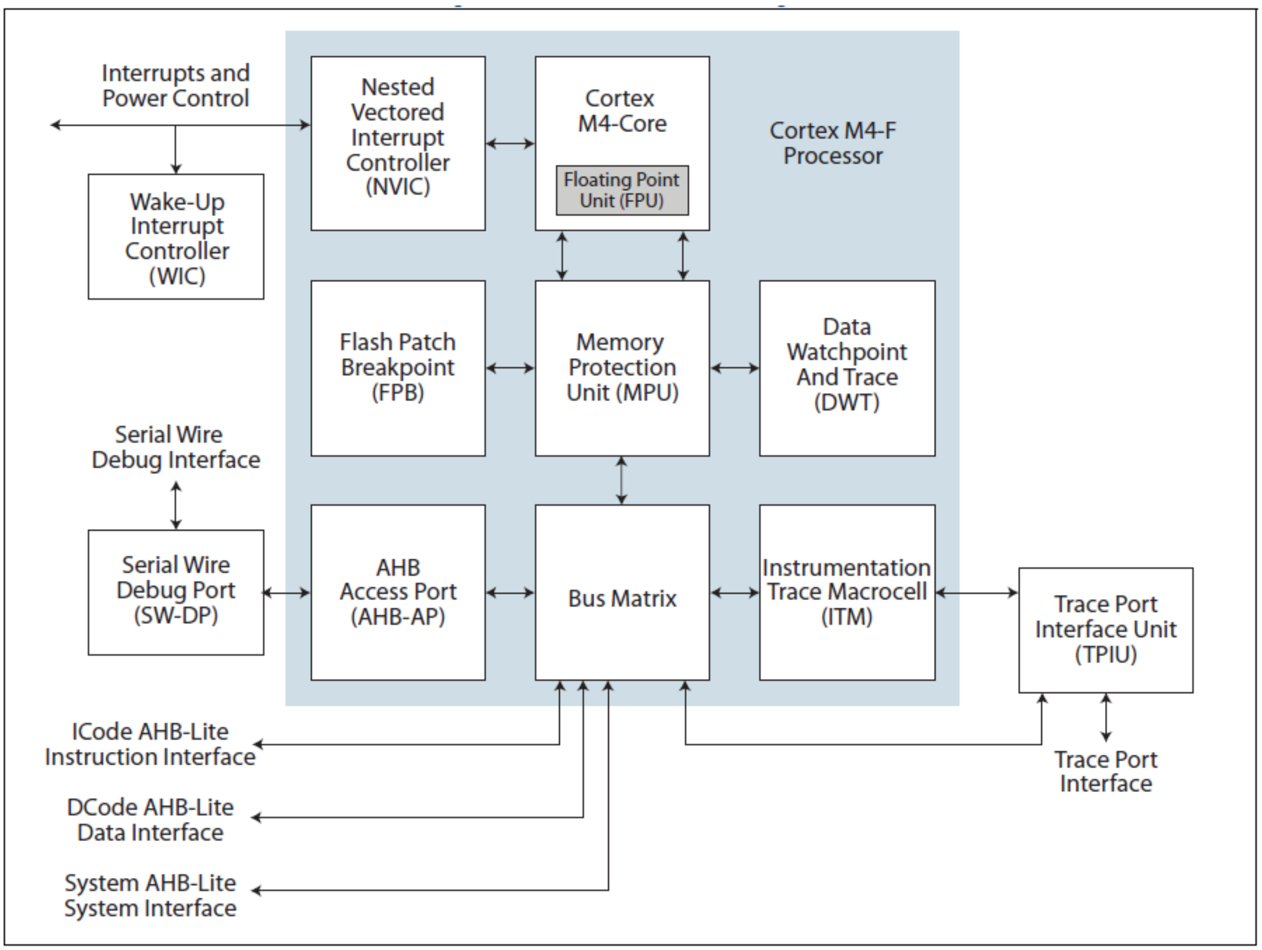
2. M4-F Processor Subsystem

2.1. Subsystem Overview

The M4-F 32-bit processor subsystem is one of the primary computation blocks of the EOS platform (shown in Figure 3) and includes:

- Optional features such as a Nested Vectored Interrupt Controller (NVIC), flash patch, etc.
- Up to 512 KB SRAM
- Peripheral bus incorporating:
  - UART
  - Watchdog Timer
  - Timers

Figure 3: Cortex M4-F Block Diagram



The M4-F processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency using an efficient instruction set and extensively optimized design. This combination provides high-end processing hardware that includes optional IEEE754-compliant single-precision floating-point computation, and a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, that ensure saturating arithmetic and dedicated hardware division.

To aid in designing of cost-sensitive devices, the M4-F processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The M4-F processor implements a version of the Thumb®, an instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The M4-F instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The M4-F instruction set provides the exceptional performance that is expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers. The Cortex M4-F processor closely integrates a configurable NVIC, to deliver industry-leading interrupt performance. The NVIC includes a Non-Maskable Interrupt (NMI) that can provide up to 256 interrupt priority levels.

The tight integration of the processor core and NVIC provides fast execution of Interrupt Service Routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple operations.

Interrupt handlers do not require wrapping in assembler code, removing any code overhead from the ISRs. A tool-chain optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, which includes an optional deep sleep function. This enables the entire device to be rapidly powered down while still retaining program state.

2.1.1. System-Level Interface

The M4-F processor provides multiple interfaces using AMBA® technology to provide high speed, low latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks and thread-safe Boolean data handling.

The M4-F processor has a Memory Protection Unit (MPU) that permits control of individual regions in memory, enabling applications to utilize multiple privilege levels, separating and protecting code, data and stack on a task-by-task basis.

Such requirements are becoming critical in many embedded applications such as automotive.

2.1.2. Integrated Configurable Debug

The 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.

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.

2.1.3. M4-F and Core Peripherals

The M4-F processor provides the following features:

- A low gate count processor core with low latency interrupt processing that includes:
  - A subset of the Thumb instruction set, defined in the *ARMv7-M Architecture Reference Manual*
  - Banked Stack Pointer (SP)
  - Hardware integer divide instructions, SDIV and UDIV
  - Handler and thread modes
  - Thumb and debug states
  - Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency
  - Automatic processor state saving and restoration for low latency Interrupt Service Routine (ISR) entry and exit
  - Support for ARMv6 big-endian byte-invariant or little-endian accesses
  - Support for ARMv6 unaligned accesses
- Floating Point Unit (FPU) providing:

  - IEEE 754-compliant operations on single-precision, 32-bit, floating point values
  - 32-bit instructions for single-precision (C float) data-processing operations
  - Combined Multiply and Accumulative instructions for increased precision (Fused MAC)
  - Hardware support for denormals and all IEEE rounding modes
  - 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers
  - Decoupled three-stage pipeline
- NVIC closely integrated with the processor core to achieve low-latency interrupt processing
  - External interrupts, configurable from 1 to 240
  - Bits of priority, configurable from 3 to 8
  - Dynamic reprioritization of interrupts
  - Priority grouping, enabling selection of preempting and non-preempting interrupt levels
  - Support for tail-chaining and late arrival of interrupts, enabling 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
  - Optional Wake-up Interrupt Controller (WIC), providing ultra-low power sleep mode support
- An optional MPU
  - 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
- Bus interfaces
  - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: I-Code, D-Code, and System bus interfaces
  - Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface
  - Bit-band support that includes atomic bit-band read and write operations
  - Memory access alignment
  - Write buffer for buffering of write data
  - Exclusive access transfers for multiprocessor systems
- Low-cost debug solution that features:
  - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYS_RSTn is asserted
  - Serial Wire Debug Port (SW-DP) debug access
  - Flash Patch Breakpoint (FPB) unit for implementing breakpoints and code patches
  - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf() style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode

2.1.4. Embedded SRAM

The M4-F processor subsystem has up to 512 KB of embedded SRAM that is divided into four sub-blocks of 128 KB each. Each sub-block is accessible simultaneously via four independent AHB busses. Each 128 KB sub-block isfurther divided down in four 32 KB segments (16 segments in total).

Three of the 128 KB sub-blocks (384 KB) of the SRAM memory reside in the processor power domain. The CPU subsystem must be powered on to access this memory space. Each 128 KB sub-block is addressed as four 32 KB segments (12 segments in total). An interrupt can be triggered when any of the 32 KB memory segments are accessed if the memory is in a lower power state (deep sleep or shut down).

The last 128 KB SRAM sub-block resides in the Always-On power domain and can be accessed regardless of the state of the processor subsystem power.

The 512 KB SRAM can be accessed by the following AHB Masters:

- M4-F system bus
- M4-F I-code bus
- M4-F D-code bus
- Application Processor (AP) through the SPI Slave Interface via the TLC
- Configuration DMA for SPI Flash Controller Master
- Voice DMA
- System DMA
- FFE

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

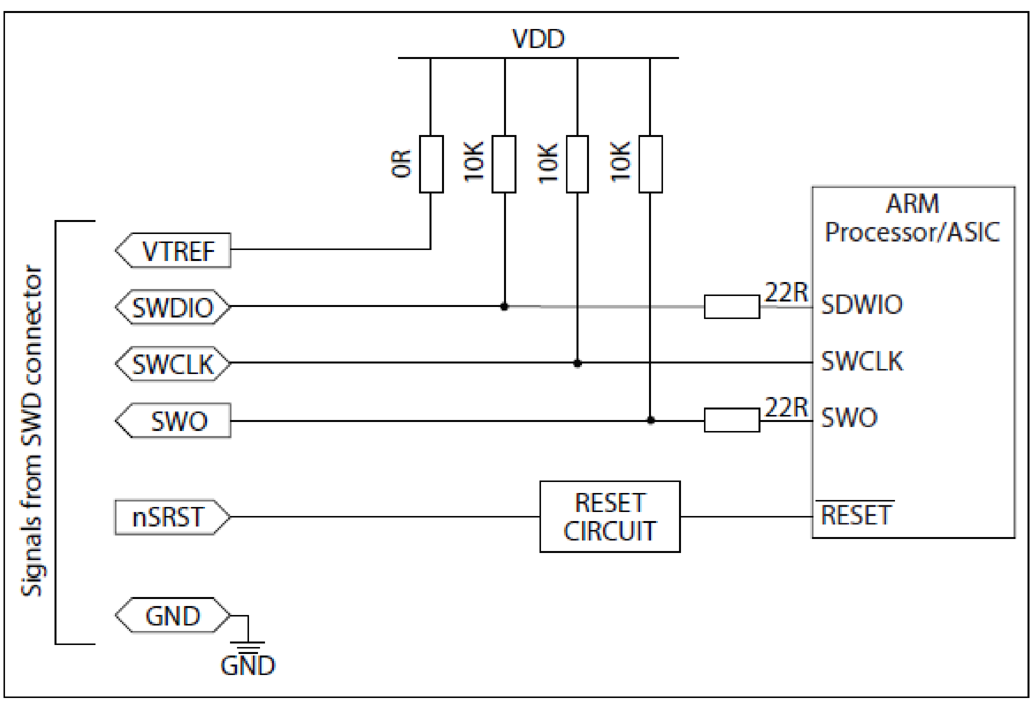
2.1.5. Development Support (Serial Wire Interface)

Depending on overall EOS S3 platform setup and requirements, two different Serial Wire interfaces can be selected. The external debugger can be connected to either IO_14/IO_15 or IO_44/IO_45, based on the bootstrap pin IO_8 (see **Bootstrap Modes** on page 66). The two signals are serial wire clock and serial wire data. The optional serial wire viewer can be selected from several different pins.

2.1.5.1. Debugger Configuration

The recommend external configuration if using ARM DS debugger is shown in **Figure 4**. Other debuggers may have different recommendations.

Figure 4: Recommended External Debugger Connection for ARM DS Debugger



2.1.6. Debugger Bootstrap Configurations

Upon cold boot up, the M4-F DAP is enabled. The M4-F DBGEN is register-enabled by default and can be disabled later if not needed. The M4-F DAP will only be reset during cold boot up (it is controlled by the POR from APC). The release of the M4-F reset depends on the state of bootstrap pin IO_19. When it is strapped to high, the M4-F reset is released. When strapped to low, the M4-F reset release depends on AP `cfg_sm`.

- Smartphone/High-Level O/S Wearable Configuration (Application Processor in System)

  In a system with an application processor present, the application processor must drive bootstrap pin IO_19 to indicate whether the Debugger is present. In this configuration, IO_19 is connected to the application processor as part of the SPI interface (the IO_ 19 alternate function is SPIs_MOSI) and it must drive IO_19 during the de-assertion of SYS_RSTn. Driving IO_19 high enables debugger support by releasing the M4-F from system reset immediately. A debugger can take control of the system. Driving IO_19 low allows the system to boot normally, which disallows the debugger access until after M4-F is released from reset. Once the M4-F is booted, the debugger can be attached.

- Wearable Configuration (EOS S3 platform operating as Host)

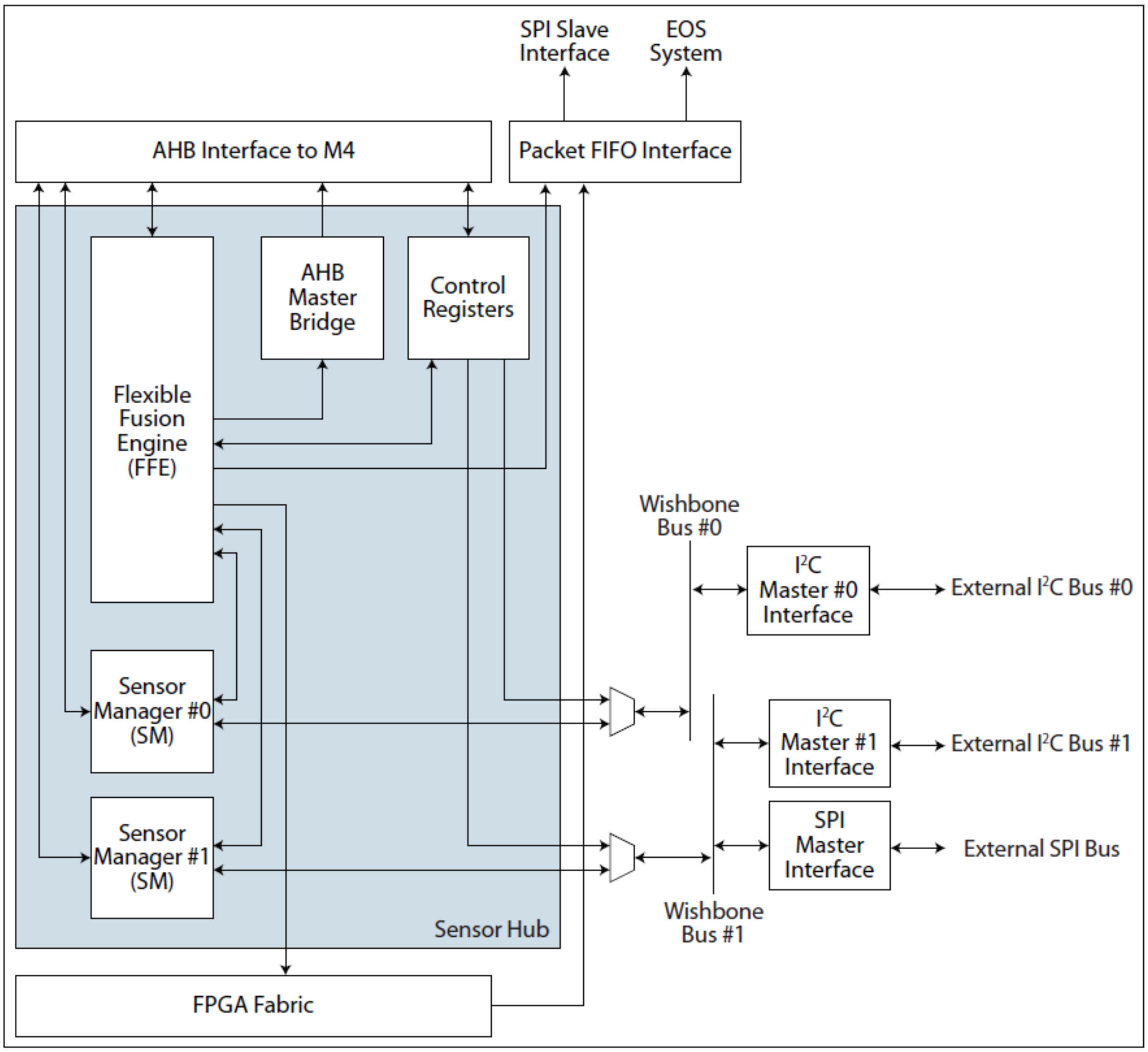
  In a wearable design, bootstrap pin IO_19 must always be strapped low to allow M4-F operation. Once the boot code is downloaded and the M4-F is released from reset, the debugger can be attached to the system.

3. Sensor Processing Subsystem

3.1. Overview

The Sensor Processing Subsystem provides the EOS S3 device with the ability to perform sensor fusion operations while using low overall power. **Figure 5** illustrates the architecture of this module.

Figure 5: Sensor Processing Subsystem Block Diagram



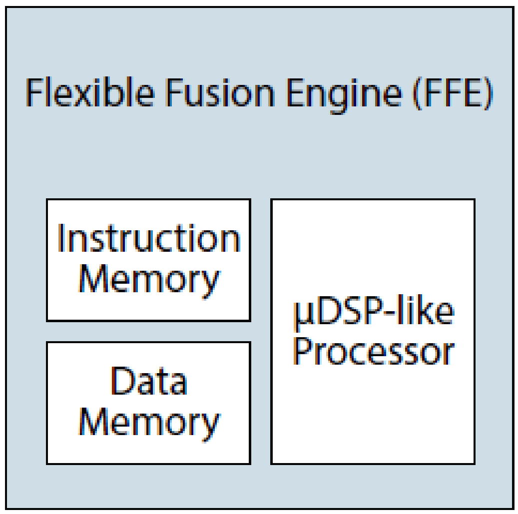
The key to the Sensor Processing Subsystem is the FFE. This block is responsible for coordinating the retrieval of sensor data by each Sensor Manager and using this data for sensor fusion operations.

3.2. Flexible Fusion Engine

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 **Figure 6** illustrates the features of the FFE architecture.

Figure 6: FFE Architecture



The FFE consists of three basic blocks:

- DSP-Like Processor
- Instruction Memory
- Data Memory

3.2.1. µDSP-Like Processor

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 selected 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.

3.2.2. Instruction Memory

The Instruction Memory contains the instructions used by the µDSP for performing the Sensor Fusion operation. The EOS S3 platform loads this memory prior to the beginning of the first FFE session. At the start of each session, µDSPs reading instructions from this memory staring 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 FFE cannot write to its own Instruction Memory. Similarly, no other module within the Sensor Processing Subsystem can read or write to the Instruction Memory.

3.2.3. Data Memory

The Data Memory contains the data portion of the øDSP 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.

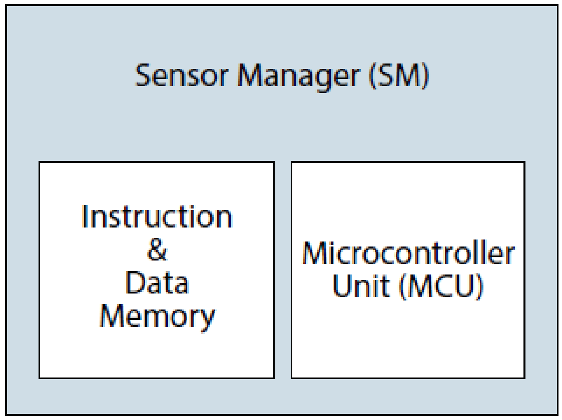
3.3. Sensor Manager

The Sensor Manager is responsible for the following tasks:

- Coordinating its actions with the FFE.
- Using the Wishbone bus interface to access the I2C/SPI Master modules.
- Managing external sensors:
  - ‣ Sensor data retrieval (such as single values, burst transfers, FIFO transfers, etc.)
  - ‣ Sensor configuration
  - ‣ Sensor calibration
  - ‣ Sensor power state management
- Storing sensor data in the proper format for retrieval by the FFE.

Figure 7 illustrates the features of the Sensor Manager architecture.

Figure 7: Sensor Manager Architecture



The Sensor Manager consists of two parts:

- Microcontroller Unit
- Instruction and Data Memory

3.3.1. Microcontroller Unit

The Microcontroller Unit (MCU) is responsible for the operation of the Sensor Manager. The MCU retrieves instructions and data from its memory module and performs operations that include:

- Reading the Mailbox data written by the FFE The Mailbox data defines which sensor handling routines to execute during the current Sensor Manager session.
- Executing the selected sensor handling routines
- Accessing the target sensor through the appropriate I^2C Master or SPI Master interface
- Storing sensor data in the appropriate packet format for retrieval by the FFE

It is important to note that each Sensor Manager session is not necessarily targeted at sensor data retrieval. As mentioned earlier, some sessions may be targeted at configuring sensors, or changing power state. The FFE selects which sensor operations will be active during each Sensor Manager session. Additionally, the Sensor Manager focuses on sensor management. As such, it does not participate in Sensor Fusion calculations.

3.3.2. Instruction and Data Memory

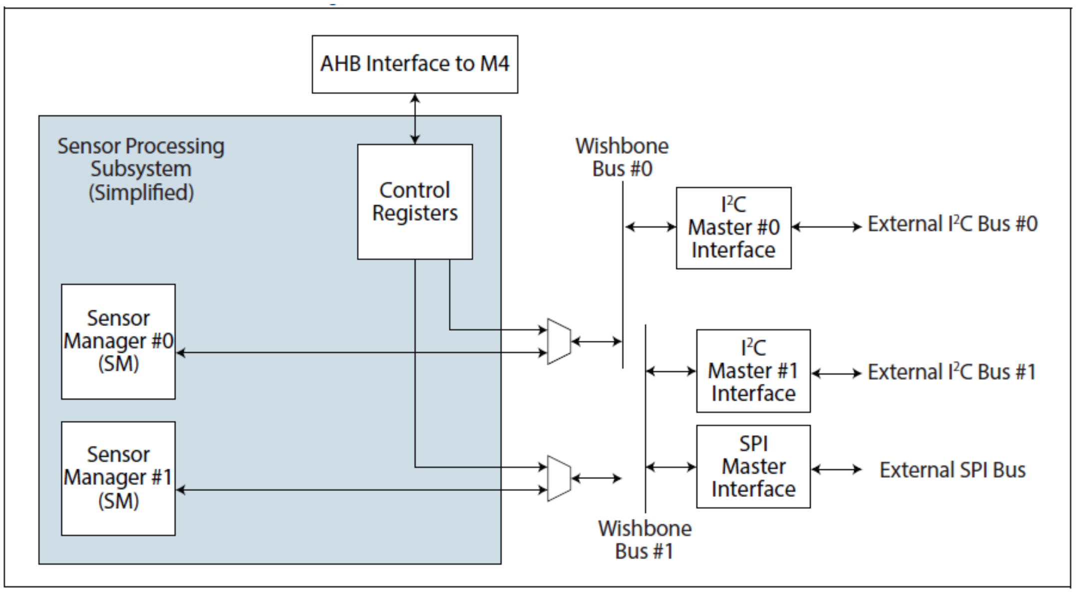
The Instruction and Data Memory holds the information that the Microcontroller uses for each of its processing sessions. Prior to the first processing session, the EOS S3 platform (e.g., M4-F or application processor) loads this memory with a series of sensor management routines. The algorithms loaded into the FFE determine, on a session by session basis, which of these routines the Sensor Manager will use. More specifically, the FFE algorithms write to a Mailbox data structure in the Sensor Manager memory. In response, the Sensor Manager only uses those routines enabled for the current session. The purpose for this approach is to enable the Sensor Manager to sample each sensor at its own rate.

The Sensor Manager stores the retrieved sensor data into the Instruction and Data Memory. The location and format of the resulting data structure helps the FFE to correctly retrieve and process this data. The format of the data structure is not fixed by hardware. Rather, it is left to software to define and implement this structure as needed.

3.4. I²C Master

There are two I²C Master modules in the EOS S3 device, and each one is assigned to a Sensor Manager module. The EOS S3 platform also makes both of these I²C Master modules directly accessible to the EOS S3 platform internal bus system. In each case, the I²C Master module provides the means for accessing devices on the associated I²C bus.

Figure 8: I2C Modules within the EOS S3 Platform



The I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. It is most suitable for applications that require occasional communication over a short distance between many devices.

The I²C standard is a true multi-master bus that includes collision detection and arbitration to prevent data corruption if two or more masters attempt to control the bus simultaneously.

The interface defines three transmission speeds:

- Normal: 100 Kbps
- Fast: 400 Kbps
- High speed: 3.5 Mbps

Only 100 Kbps (Normal) and 400 Kbps (Fast) modes are directly supported. The following features are available in the I²C Master block:

- Compatible with the Philips I²C 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 cancellation
- Start/Stop/Repeated Start/Acknowledge generation
- Start/Stop/Repeated Start detection
- Bus busy detection
- Supports 7-bit and 10-bit addressing mode
- Operates from a wide range of input clock frequencies

- Static synchronous design
- Fully synthesizable

The following sections describe the I^2C system operations.

3.4.1. System Configuration

The I^2C system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to these two signals must have open drain or open collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line on a byte-by-byte basis, and each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; as a result, the SDA line can be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (for details, see **START Signal** on page 18 and **STOP Signal** on page 18).

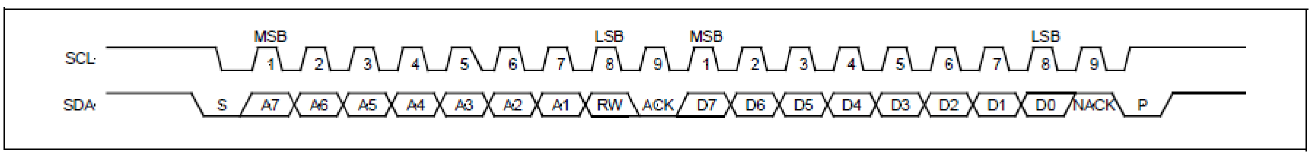
3.4.2. I^2C Protocol

Normally, standard communication consists of the following four parts:

- START signal generation
- Slave address transfer
- Data transfer
- STOP signal generation

Figure 9 illustrates an example of I^2C protocol.

Figure 9: I^2C Protocol Example



3.4.3. START Signal

When the bus is free/idle, this means no master device is engaging it (and both SCL and SDA lines are high), and the master can initiate a transfer by sending a START signal. A START signal, usually referred to as the S-bit, is defined as a high-to-low transition of SDA while SCL is high. The START signal denotes the beginning of a new data transfer.

A Repeated START is a START signal that is sent without first generating a STOP signal. The master uses this method to communicate with another slave or with the same slave in a different transfer direction (for example, changing from writing to a device to reading from a device) without releasing the bus.

The core generates a START signal when the STA-bit in the Command Register is set and the RD or WR bits are set. Depending on the current SCL line status, it generates a START or Repeated START.

3.4.4. Slave Address Transfer

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a 7-bit calling address followed by a RW bit. The RW bit signals to the slave the data transfer direction. No two slaves in the system can have the same address.

Only the slave with an address that matches the one transmitted by the master responds by returning an acknowledge bit, which pulls SDA low at the ninth SCL clock cycle.

NOTE: The core supports 10-bit slave addresses by generating two address transfers. For details, see the Philips I^2C specifications.

The core treats a Slave address transfer like any other write action. The core stores the slave device address in the Transmit Register, sets the WR bit and then transfers the slave address on the bus.

3.4.5. Data Transfer

Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. Each transferred byte is followed by an acknowledge bit on the ninth SCL clock cycle. If the slave signals a No Acknowledge, the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle. If the master, as the receiving device, does not acknowledge the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

To write data to a slave, store the data to be transmitted in the Transmit Register and set the WR bit. To read data from a slave, set the RD bit. During a transfer, the core sets the TIP flag, indicating that a transfer is in progress. When the transfer is done, the TIP flag is reset, the IF flag is set, and when enabled, an interrupt is generated. The Receive Register contains valid data after the IF flag has been set. The user can issue a new write or read command when the TIP flag is reset.

3.4.6. STOP Signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the P- bit, is defined as a low-to-high transition of SDA while SCL is at logical 1.

3.4.7. Arbitration

The I^2C Master block supports multi-master arbitration. However, this feature is not supported by other elements of the EOS S3 platform.

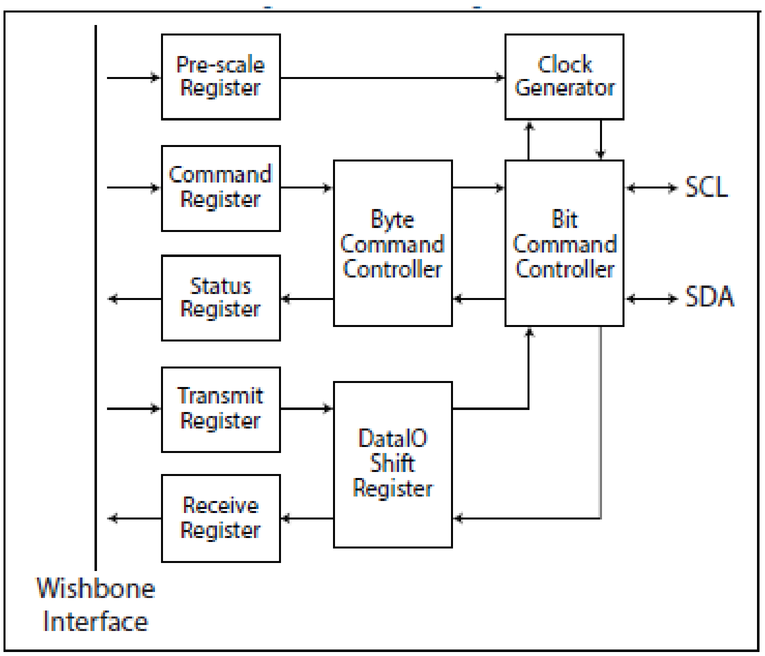
3.4.8. I^2C Core Architecture

The I^2C core is built around the following four primary blocks (as shown in **Figure 10**):

- Clock Generator
- Byte Command Controller
- Bit Command Controller
- DataIO Shift Register.

NOTE: All other blocks are involved with interfacing or for storing temporary values.

Figure 10: I^{2}C Block Diagram



The Sensor Manager uses the Wishbone interface to access the I^{2}C Master during sensor data transfers.

3.4.9. Clock Generator

The Clock Generator generates an internal 4*Fscl clock enable signal that triggers all of the synchronous elements in the Bit Command Controller. In addition, it also handles clock stretching required by some slaves.

3.4.10. Byte Command Controller

The Byte Command Controller handles I^{2}C traffic at the byte level. It takes data from the Command Register and translates it into sequences based on the transmission of a single byte. By setting the START, STOP, and READ bit in the Command Register, the Byte Command Controller performs the following sequence:

- A START signal is generated
- The byte is read from the slave device
- A STOP signal is generated

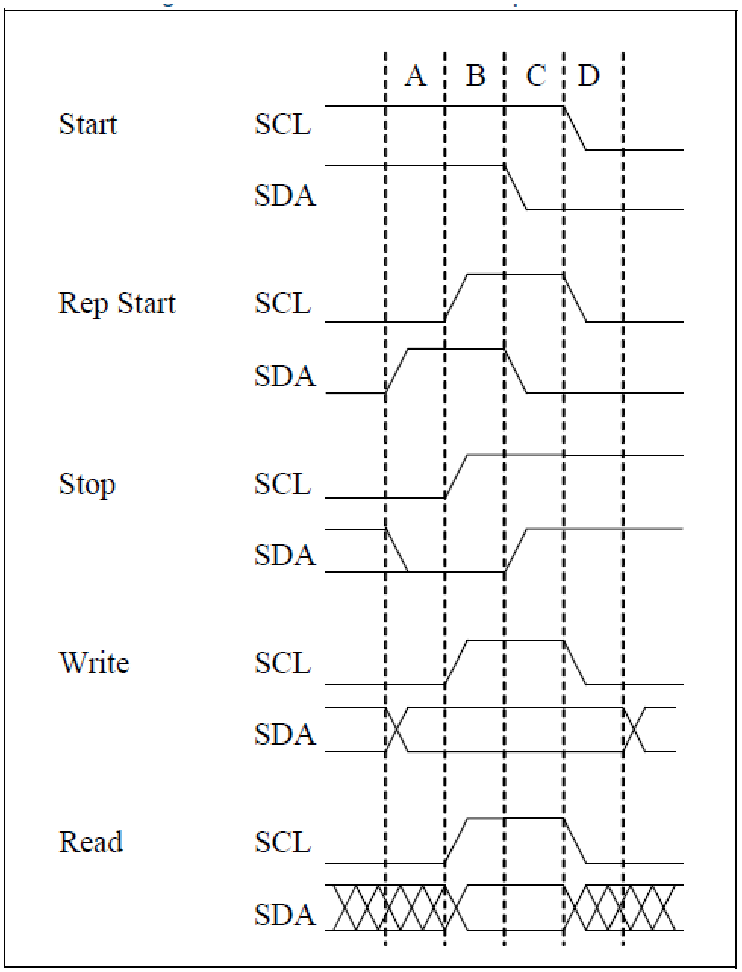
Setting the START, STOP, and READ bits and the Byte Command Controller sequence starts a process that acts to divide each byte operation into separate bit-operations, which are then sent to the Bit Command Controller.

3.4.11. Bit Command Controller

The Bit Command Controller handles the actual transmission of data and the generation of the specific levels for START, Repeated START, and STOP signals by controlling the SCL and SDA lines.

The Byte Command Controller tells the Bit Command Controller which operation needs to be performed. For a single- byte read, the Bit Command Controller receives eight separate read commands. Each bit-operation is divided into five smaller pieces (idle and A, B, C, and D), except for a STOP operation which is divided into four smaller pieces (idle and A, B, and C). **Figure 11** illustrates the I^{2}C bit command sequences.

Figure 11: I²C Bit Command Sequences



3.4.12. Data I/O Shift Register

The DataIO Shift Register contains the data associated with the current transfer. During a read action, data is shifted in from the SDA line. After a byte has been read, the contents are copied into the Receive Register. During a write action, the contents of the Transmit Register are copied into the DataIO Shift Register and are then transmitted onto the SDA line.

3.5. Serial Peripheral Interface (SPI)

The EOS S3 platform relies on three separate SPI interfaces.

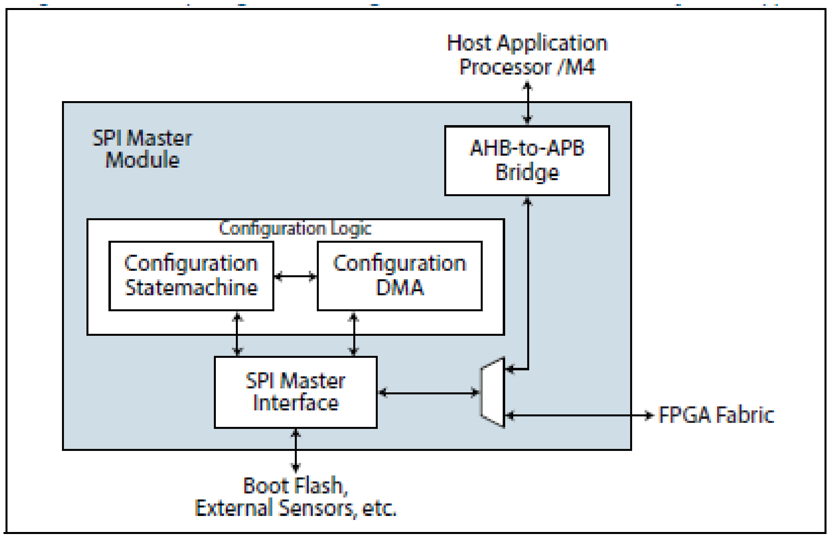
- SPI Master for System Support
- SPI Master for Sensor Subsystem Support
- SPI Slave

The following sections describe each of these interfaces.

3.5.1. SPI Master for System Support

IP within the on-chip programmable logic can directly access the SPI Master for system support interface. **Figure 12** shows this connection.

Figure 12: On-Chip Programmable Logic IP Access to SPI Master for System Support



The ability to directly access the SPI Master provides an IP designer with the option to create on-chip programmable logic IP that can directly access external devices such as Flash Memory or Sensors. In the latter case, the Sensor data values can be used for additional Sensor Fusion operations in parallel with the Sensor Processing Subsystem. In such cases, the on-chip programmable logic IP can use either the Packet FIFO Interface or use the SDMA to move the processed data into the EOS S3 platform for further evaluation or processing.

During the initial boot operation, the SPI Master is exclusively accessed by the Configuration Logic. Once the Configuration Logic completes the initial boot operation, the SPI Master becomes available to the M4-F for additional data retrieval from the external flash device. More specifically, the initial boot code provides a boot loader to the M4-F for the M4-F to complete its retrieval of M4-F code.

Once the boot process completes, either the M4-F or on-chip programmable logic can access the SPI Master Interface and use this to access any device on the SPI bus. This can be additional external flash devices, sensors, or system support devices such as Power Management devices.

3.5.2. SPI Master for System Support Features

The SPI Master interface supports the following operations:

- Single SPI transfers
- DMA transfers of SPI data retrieved from an external flash device. The following features show the operation of this module.
- SPI Master Interface provides the following:
  - Operates as a Master only
  - Supports up to three slaves
  - Operates in mode 0 (this can be reprogrammed by M4)
  - Supports a frame size of 8 (this can be reprogrammed by M4)
  - 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:
  - Host Application Processor
  - Configuration Logic
  - On-chip programmable logic
- 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

3.5.3. Configuration Logic

The configuration logic consists of the following two parts:

- Configuration State Machine
- Configuration DMA logic

The Configuration State Machine provides control over retrieving boot code from an external flash device. To complete this operation, the Configuration State Machine must configure and control the Configuration DMA and SPI Master Interface.

The Configuration State machine programs the Configuration DMA and SPI Master Interface to access an external flash, by performing the following operations:

- Setting the correct SPI clock rate
- Awakening the external Flash device from a low power state (such as a deep sleep modes)
- Examining the boot code parameters to optimize SPI Master transfers
- Initiating DMA loading of the M4-F boot code into M4-F memory
- Enabling M4-F operations

As a part of this process, the Configuration State Machine examines the EOS S3 device ID in the boot flash data. If this device ID is incorrect, the Configuration State Machine halts the boot process, and this boot process can only be restarted by asserting a reset.

3.5.4. SPI Master for Sensor Processing Subsystem Support

The EOS S3 platform SPI Host Controller can communicate with up to eight SPI sensor devices using the Wishbone classic interface.

The SPI Master Interface features include:

- Support for Master configuration (multi-master configuration is *not* supported)
- Connection capability supports up to eight SPI slaves with individual slave select lines
- Interrupt generation capability

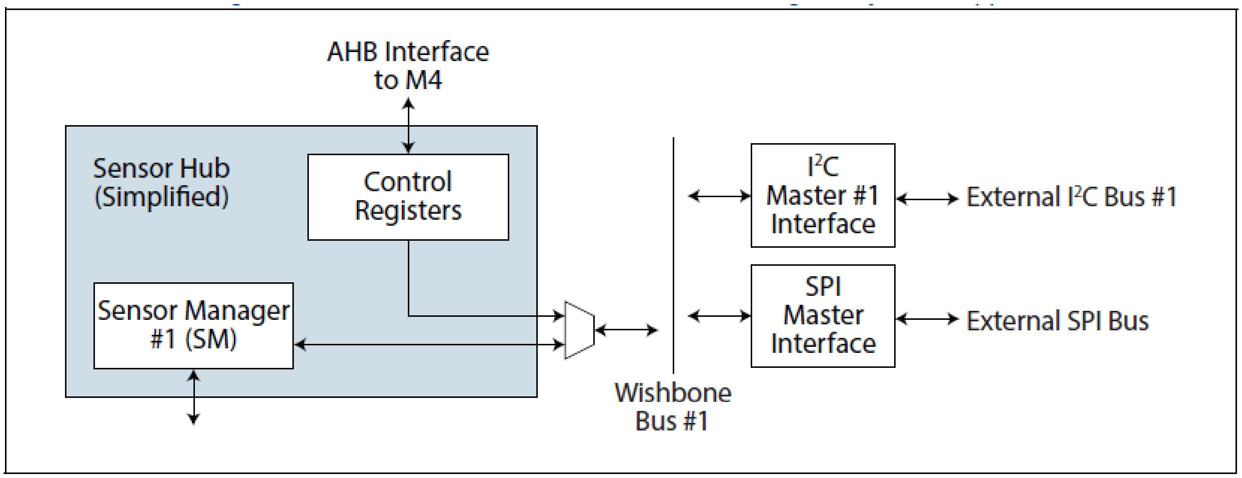
- Serial clock with programmable phase and polarity
- Four programmable transfer formats supported (controlled by CPOL and CPHA)
- Support for all current SM instructions
- Support all operations (identical to the I^2C Master Controller)
- 8-bit Wishbone interface (identical to the I^2C Master Controller)

The SPI Master module used for the Sensor Processing Subsystem support resides on the same Wishbone bus as the I^2C Master module described in **Sensor Manager** on page 1 and **Control Registers** on page 36. Like the I^2C Master module, the SPI Master module enables the Sensor Processing subsystem to communicate with external devices.

The SPI module is accessible to the M4-F processor but is not used for operations such as retrieving boot code from external flash storage devices. The operation is left to the SPI Master used for System Support.

Figure 13 shows basic connections to the SPI Interface that supports the Sensor Processing subsystem.

Figure 13: SPI Interface used for Sensor Processing Subsystem Support



3.5.5. SPI Slave

The SPI Slave module provides the means for communicating between a host system and the EOS S3 platform. This block consists of the SPI Interface and the Top Level Controller (TLC) module.

The SPI Slave module performs the following two roles:

- Setting up and debugging the EOS S3 platform
- Retrieving run-time data

Both of these operations are possible during either *Smart Phone* or *Wearable* EOS S3 modes. However, in a typical *Wearable* application, there is no local Host to use the SPI Slave interface. Therefore, the following description is primarily focused on the *Smart Phone* mode of operation.

During the EOS S3 platform non-debug setup operations, the Host system uses the SPI Slave module to write a precompiled binary file to the EOS S3 device M4-F memory. In addition, the Host also writes precompiled binary files to other memories such as those used by the FFE and the Sensor Manager. Once completed, the Host system enables the EOS S3 platform M4-F processor to execute the newly written binary code.

During normal the EOS S3 platform processing, the Host uses the SPI Slave interface to retrieve the results of the EOS S3 platform sensor fusion processing, and then it passes this data on to the corresponding application. Explanation regarding how the Host knows when to retrieve results is beyond the scope of this section.

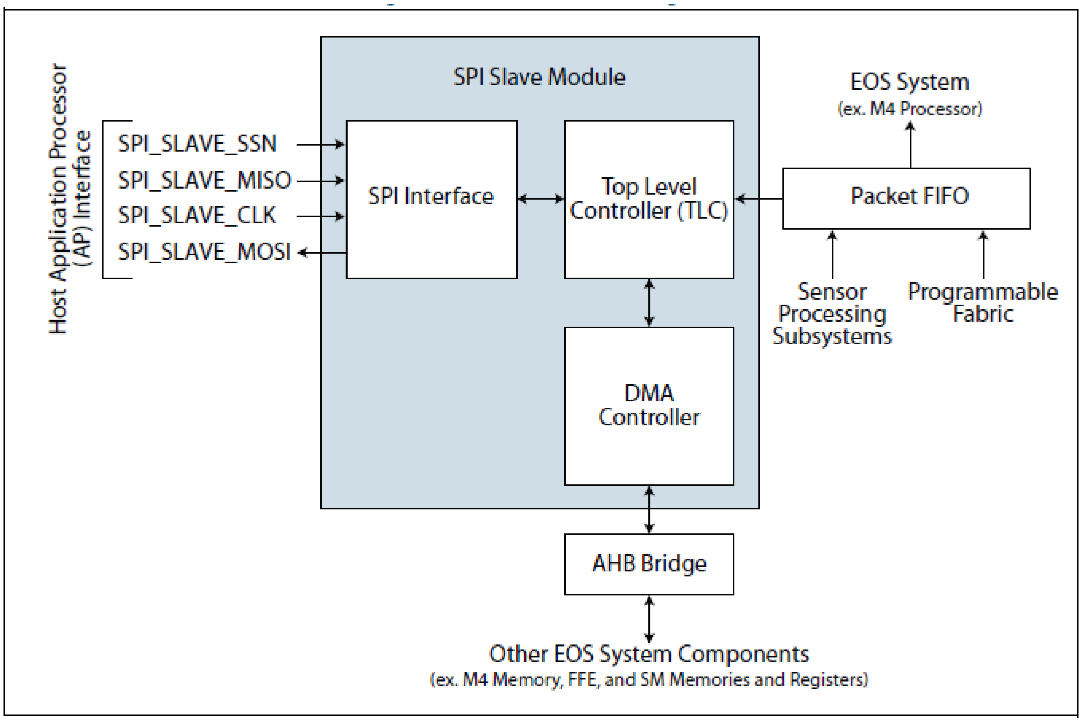
During the EOS S3 platform debugging operations, the Host uses the SPI Slave interface in ways that are similar to the setup and normal processing operations addressed previously. Specifically, the Host uses the SPI Slave interface to perform the following tasks:

- Load debug code
- Enable the M4-F to execute this code
- Retrieve the results

In addition, the Host can elect to only enable specific subsystems to diagnose. For example, the Host can configure the FFE and Sensor Manager with diagnostic code, enable these subsystems to perform sensor fusion processing, and then retrieve the result of this processing all without enabling the EOS S3 platform M4-F processor.

Figure 14 illustrates the SPI Slave interface.

Figure 14: SPI Slave Block Diagram



It is important to note that the SPI Slave module does not communicate directly to other functional blocks. Rather, it needs to perform transfers through one or more AHB bridges for to access any register or memory within the EOS S3 platform. For optimum performance of this interface during the Smart Phone mode, the EOS S3 platform needs to avoid using common paths through the AHB infrastructure that are also used by the SPI Slave interface.

3.5.6. SPI Interface Protocol

The SPI Interface block only supports SPI Mode 0:

- 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 being driven low (active) by the SPI Master, and then driven high after all of the desired bytes have been transferred. The SPI Interface assumes that all transfers consist of complete bytes. Any incomplete bytes at the end of a transaction are ignored by the hardware. The EOS S3 Platform SPI Interface protocol supports several different operations. The following sections describe these operations.

3.5.7. Basic Read/Write Transfers

For basic Read/Write transfers to the TLC registers, the EOS S3 platform SPI Interface protocol requires that the Address Byte be transmitted first. The Address Byte includes a single Direction bit that represents the direction for the transfer (write vs. read). As per SPI protocol, SPI_SLAVE_SSn is also required to be LOW when active SPI transactions are in progress. The rising edge of SPI_SLAVE_CLK captures the data bits on SPI_SLAVE_MOSI and SPI_SLAVE_MISO pins.

The Direction bit is positioned in the Most Significant Bit (MSB) 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, and this address is unique to the TLC module and should not be confused with M4-F addresses.

Examples:

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

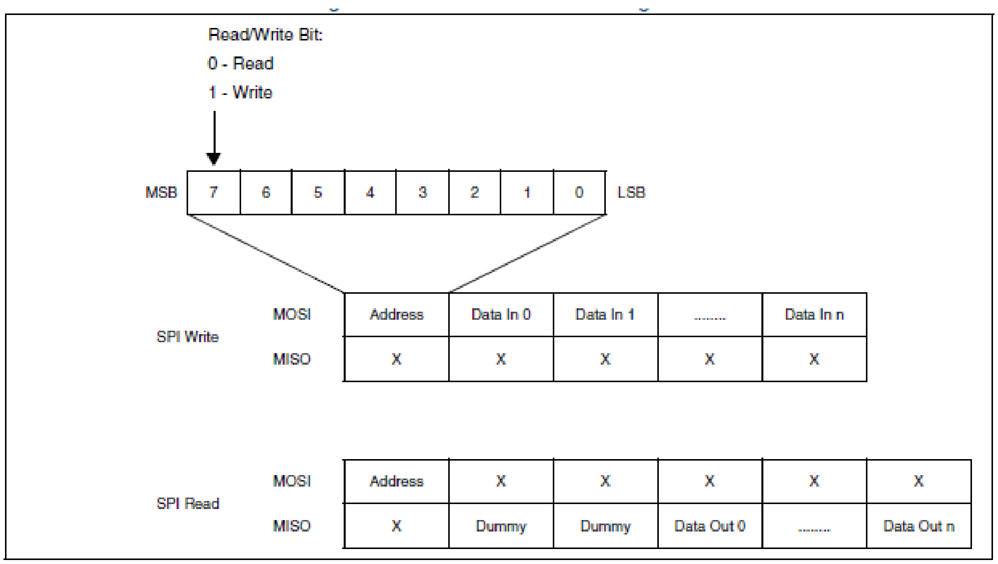
During write transactions, the first byte received on the Master Out Slave In (SPI_SLAVE_MOSI) pin corresponds to the Address Byte. The next bytes received correspond to Data Bytes. No valid data is driven on the Master In Slave Out (SPI_SLAVE_MISO) pin. If the SPI clock is not free-running (for example, it does not toggle while SPI_SLAVE_SSN is inactive), the hardware requires that there be at least two 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 Scratch Byte Register (0x31). Refer to the **Transfers to TLC Local Registers** on page 28.

During read transactions, the first byte received on the SPI_SLAVE_MOSI pin corresponds to the Address Byte. The SPI Interface ignores all subsequent data bits on the SPI_SLAVE_MOSI pin. Instead, following the Address Byte, the SPI Interface begins to drive data on the SPI_SLAVE_MISO pin.

This begins by outputting two dummy bytes followed by the first Data Byte. Thereafter, each byte transmitted on the SPI_SLAVE_MISO pin corresponds to valid Data Bytes. Unlike write transfers, read transfers do not require extra SPI clock cycles after each transfer. **Figure 15** shows an example of the SPI Interface protocol. In that use case, the MSB bit is shifted out first.

Figure 15: SPI Slave Protocol Diagram



3.5.8. Device ID Read

The Device ID transfer cycle is a special protocol cycle for identifying the EOS S3 device to the Host SPI controller. Unlike the write transfer cycle previously described, the address value of 0xFF indicates to the SPI Interface that an ID read cycle is underway. In response, the SPI Interface returns an ID value 0x21 on the SPI_SLAVE_MISO pin one byte after the address phase.

Figure 16 shows an example of the SPI Slave Device ID read protocol.

Figure 16: SPI Slave Device ID Read Protocol

MOSI	0xFF	X	X
MISO	X	Dummy	0x21

3.5.9. Transfer Types

There are three basic transfer types supported:

- Transfers to TLC local registers — This transfer type accesses TLC local registers alone and does not produce any activity on the AHB interface.
- Transfers from Packet FIFOs
  - This transfer type accesses the TLC local registers. However, the goal is to read from the Packet FIFOs.
  - The read transactions can be conducted as a single or as a burst transfer.
- Transfers to resources in the M4-F Memory address space.
  - This transfer type also accesses the TLC local registers. However, the goal is to conduct a transfer using the AHB interface.
  - This transfer type also supports the following operations:

    – Burst write transfers to the M4-F Memory space.

    – Burst read transfers from the M4-F Memory space.

3.5.10. Transfers to TLC Local Registers

This transfer type depends upon the type of TLC register being accessed. The default TLC response is to increment automatically the TLC register address (not the M4-F Memory space address) after each byte accessed (for example, as in a read from the M4-F Memory Address registers). For details see **Figure 15**, where the SPI protocol address phase uses the address for the Memory Address Byte 0 register.

Once this transfer has completed, the TLC automatically increments its register address to Memory Address Byte 1. Similarly, once this transfer has completed, the TLC automatically increments its register address to Memory Address Byte 2. This sequence repeats until all bytes are transferred or a TLC register address is reached that prevents this auto- incrementing operation. For details about TLC register types that prevent auto-incrementing, see **Transfers to M4-F Memory Address Space** on page 28.

NOTE: This auto-incrementing operation does not prevent any special features in these registers from being triggered. For exceptions, see **Basic AHB Transfer Restrictions** on page 28.

3.5.11. Transfers from Packet FIFOs

Packet FIFOs accessible from within TLC can only be read from and cannot be written to. Therefore, any 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. As a result, the TLC does not increment its register address when accessing any Packet FIFO address.

3.5.12. Transfers to M4-F Memory Address Space

The following sections outline the transfer types and restrictions when accessing the M4-F Memory Address space via the AHB Bridge interface.

3.5.13. 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 describe additional transfer specific restrictions.

- AHB Memory Burst Write
- AHB Memory Burst Read

3.5.13.1. AHB Memory Burst Write

All AHB write operations are done though programming TLC registers. A single write transfer is treated as a burst of one 32-bit word. To set up an AHB write operation, the Host needs to write the TLC AHB Access Control to 0x3.

Set up the target AHB address by writing to the TLC Memory Address Byte 3 – 0. Keep the TLC Memory Address Byte 0 as bits[1:0] to 0x3. For example, writing to address 0x20001040 means writing:

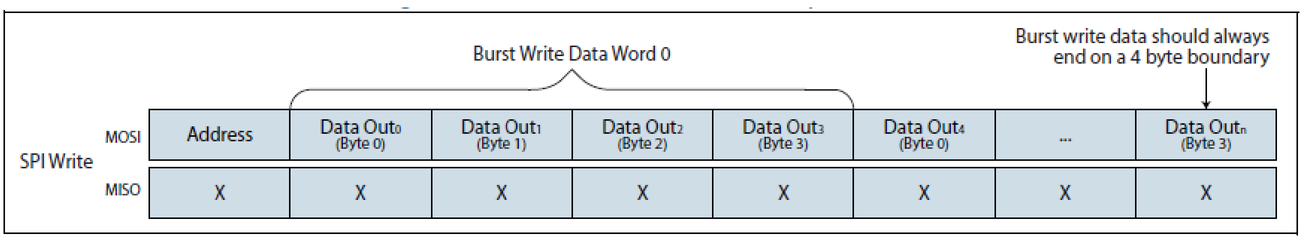
- TLC Memory Address Byte 0 to 0x43
- TLC Memory Address Byte 1 to 0x10
- TLC Memory Address Byte 2 to 0x00
- TLC Memory Address Byte 3 to 0x20

This is followed by writing the data value into the TLC 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 registers address loops back to point to the first data byte register, Memory Data Byte 0. By doing this, a block of data may be written from a single SPI data stream (for example, one address phase followed by a series of data phases representing the burst data).

See **Figure 17** for an example of a burst write sequence.

Figure 17: SPI Master Burst Write Sequence



AHB Memory write transfer block sizes are currently limited to 64K bytes (for example, 16K, 32-bit words). In addition, access restrictions through the AHB Bridge require each transfer cannot consist of more than four bytes. Consequently, writes to the TLC Memory Data Byte 3 register automatically trigger an increment by four in the TLC Memory Address Byte 0 – 1 registers, as well as triggering as data write through the AHB interface. The TLC 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 automatically wrap back to the beginning.

3.5.13.2. AHB Memory Burst Read

To complement the Burst Write transfer in the previous section, the EOS S3 device also provides a Burst Read transfer operation. Unlike the Burst Write, the EOS S3 device implements its Burst Read transfer as a DMA operation. Prior to initiating Burst Read transfers, the software must first check the following conditions:

- If there is no Burst Read transfer underway: All of the data from the previous Burst Read transfer must first be read out through the SPI Interface prior to initiating another Burst Read transfer.
- If there is data remaining in the Burst Read transfer FIFO: Check this by reading the Burst FIFO Status register.

Once both conditions are true, the Host can configure the Burst Read transfer by writing the target address into the TLC Burst

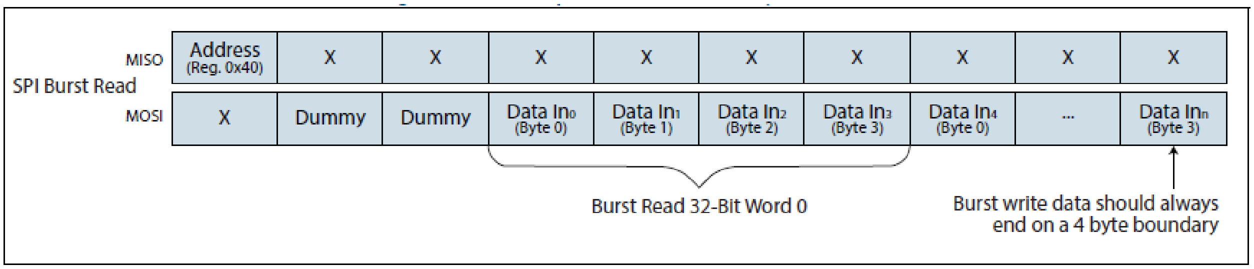
Read AHB Byte Address Byte 0 – 3 registers. This is followed by writing the data value into the TLC Burst Size Byte 0 register. The Burst Read begins with writing to the Burst Size Byte 1 register, and there needs to be two *dummy* byte cycles following the write to the Burst Size Byte 1 register.

The simplest way to do this is by reading the Burst FIFO Status register, where this status is used to determine that there is a minimum of one byte available to be read from the Burst Read Data register. If the minimum (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 incrementing its register address pointer when it reads from the Burst Read Data register. If a single word needs to be read, set both the Burst Size Byte 0 and 1 register to zero.

Figure 18 shows an example of the sequence.

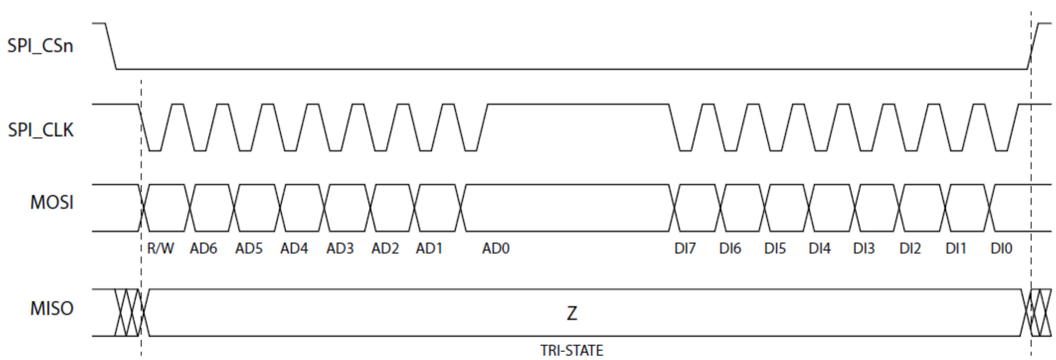
Figure 18: Example Burst Read Sequence



3.5.14. SPI Write Cycle

Figure 19 shows the basic SPI write operation under mode 11 (CPOL = 1, CPHA = 1).

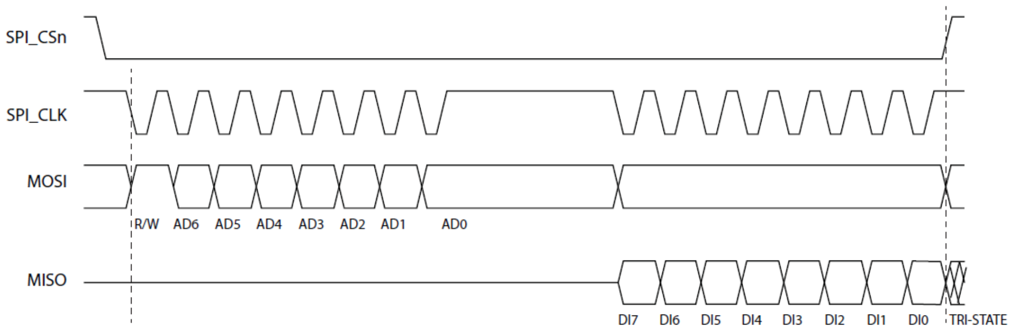
Figure 19: Basic SPI Write Operation (Mode 11)



3.5.15. SPI Read Cycle

Figure 20 shows the basic SPI read operation under mode 11 (CPOL = 1, CPHA = 1).

Figure 20: Basic SPI Read Operation (Mode 11)

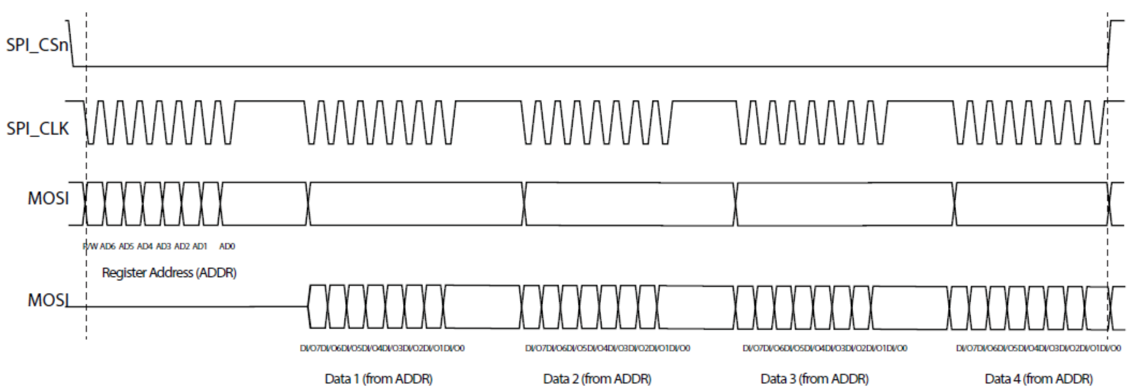


3.5.16. SPI Multiple Read Cycle

Figure 21 shows SPI multiple read operations under 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 providing that the SPI_CSn line is held low.

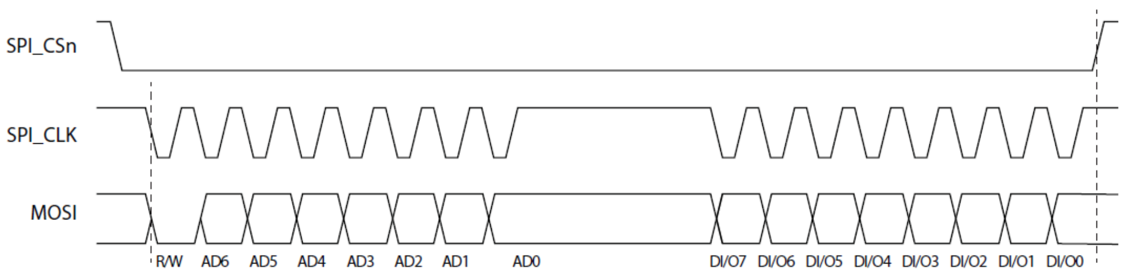
Figure 21: SPI Multiple Read Operation (Mode 11)



3.5.17. SPI 3-Wire Configuration

Figure 22 shows a basic SPI read/write operation under mode 11 (CPOL = 1, CPHA = 1) in a 3-wire configuration.

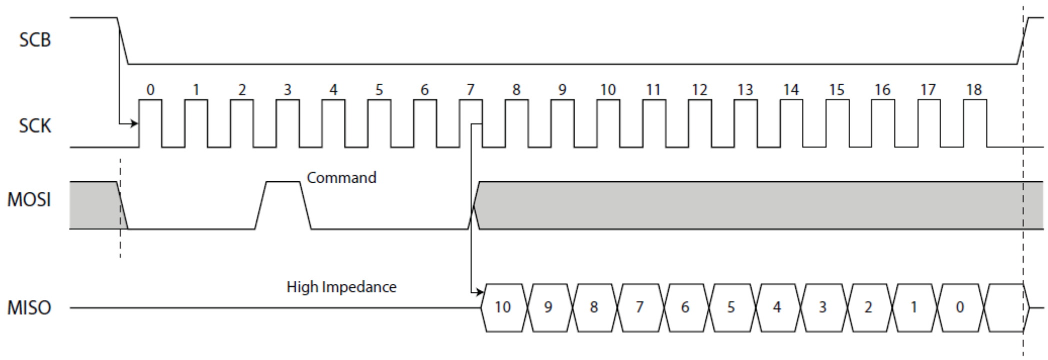
Figure 22: 3-Wire Basic SPI Read/Write Sequence (Mode 11)



3.5.18. SPI Corner Cases

There are cases where certain sensors do not support 8-bit data alignment. In the following example, this transfer sequence comes from a muRata™ SCA100T-D07 2-Axis High Performance Analog Accelerometer, (as shown in **Figure 23**).

Figure 23: muRata Command and 11-bit SPI Acceleration Data Read Sequence



If the SPI interface is intended to support additional devices, such as SPI-based ACDs for measuring the output of additional sensor types, then the number of potential non-byte aligned shift sequences increases significantly.

For example, the SPI transfer sequence corresponding to the AD7091 low power, 12-bit ADC is shown in **Figure 24**.

Figure 24: AD7091 SPI Transfer Sequence

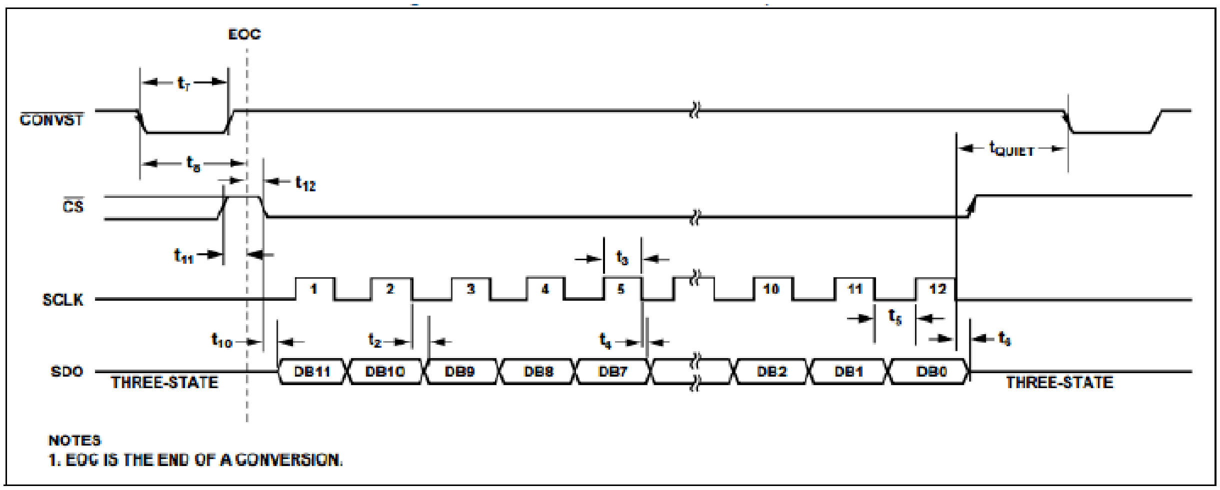


Figure 23 and **Figure 24** show that the data portion of the transfer consists of values that are not a direct multiple of 8-bits. Therefore, the SPI transfer would need to first store the MSBs (e.g., DB[11:8] for the AD7091) using a shift sequence of four SPI clock cycles prior to storing the LSB.

These are just a few of the potential corner cases that SPI-based devices may present. To support such cases, the following capabilities have been added:

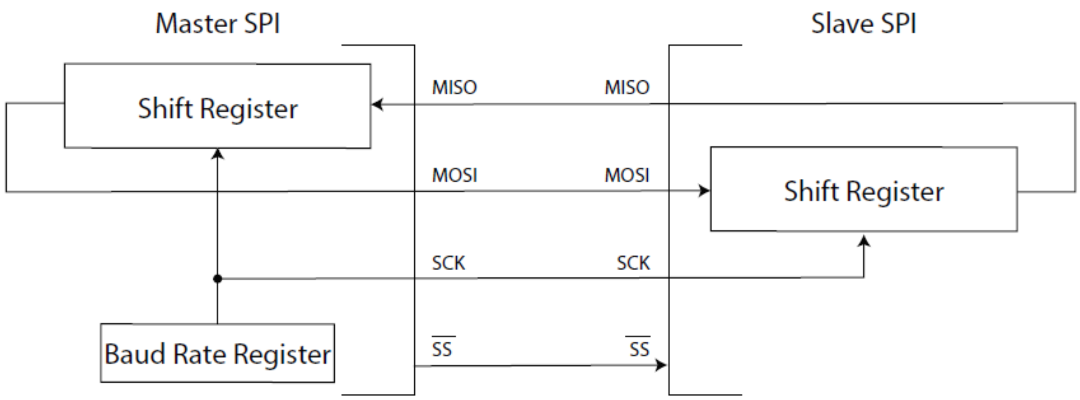
- Shift data based on 1 to 8 shift clock cycles
- Keep the Chip Select signal low between SPI transfer cycles
- Allow for running the SPI shift clock for a preset number of cycles after asserted

Some specific devices require additional shift clock cycles after Chip Select is removed (for example, to allow them to achieve low power states or to complete a requested operation).

3.5.19. Transmission Format

During a SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes both the shifting and the sampling of the information on the two serial data lines. A slave select line allows for the selection of an individual slave SPI device (slave devices that are not selected do not interfere with the ongoing SPI bus activities. **Figure 25** shows the relationship between the SPI Master, SPI Slave, and the related registers.

Figure 25: SPI Block Diagram



3.5.20. Clock Phase and Polarity Controls

Using two bits in the SPI Control Register1, the 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, neither of which significantly affects 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 may be changed between transmissions to allow a master device to communicate with peripheral slaves that have different requirements.

CPHA = 0 Transfer Format

The first edge on the SCK line clocks 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 data is available at the slave 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 Least Significant Bit First Enable (LSBFE).

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 16 edges on the SCK line, with data being latched on odd numbered edges, and shifted on even numbered edges.

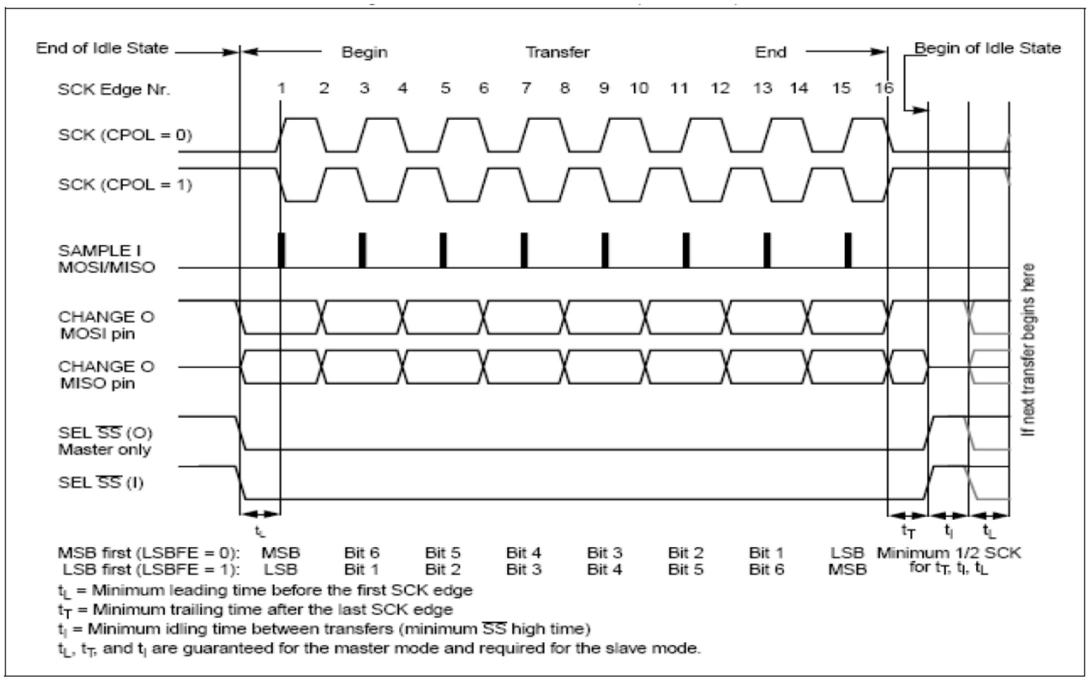
Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer, and is transferred to the parallel SPI data register after the last bit is shifted in. After the sixteenth (the last of the edges) on the SCK line:

- Data that was previously in the master SPI data register should now be in the slave data register, and conversely, the data that was in the slave data register should be in the master SPI data register.
- The SPI Interrupt Flag (SPIF) in the SPI status register is set indicating that the transfer is complete.

Figure 26 shows a timing diagram of a SPI transfer where CPHA = 0, with SCK waveforms shown for CPOL = 0 and CPOL = 1. The diagram can 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 set either high or reconfigured as a general-purpose output that does not affect the SPI.

Figure 26: SPI Clock Format 0 (CPHA = 0)



CPHA = 1 Transfer Format

Some peripherals require the first SCK edge before the first data bit becomes available at the data out pin, and 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, which 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.

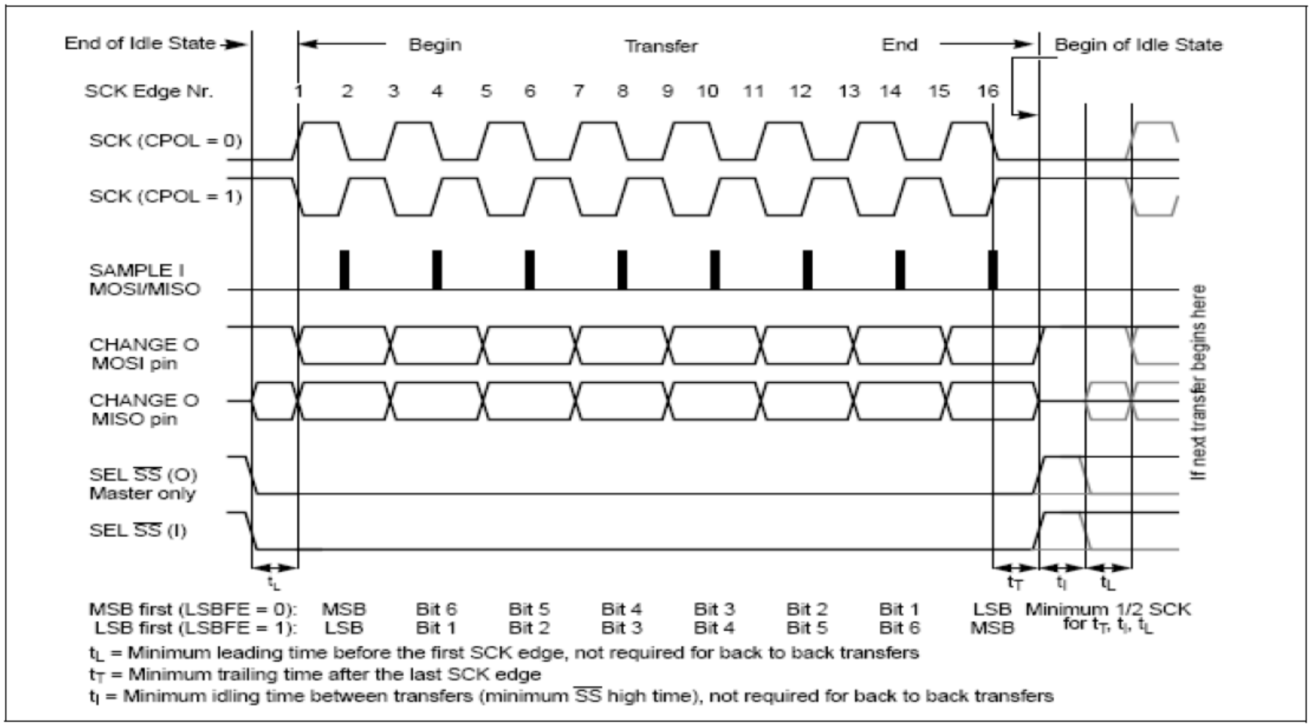
After the sixteenth SCK edge:

- Data that was previously in the SPI data register of the master is now in the data register of the slave, and conversely, 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 27 shows two clocking variations for CPHA = 1. The diagram can 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 that does not affect the SPI.

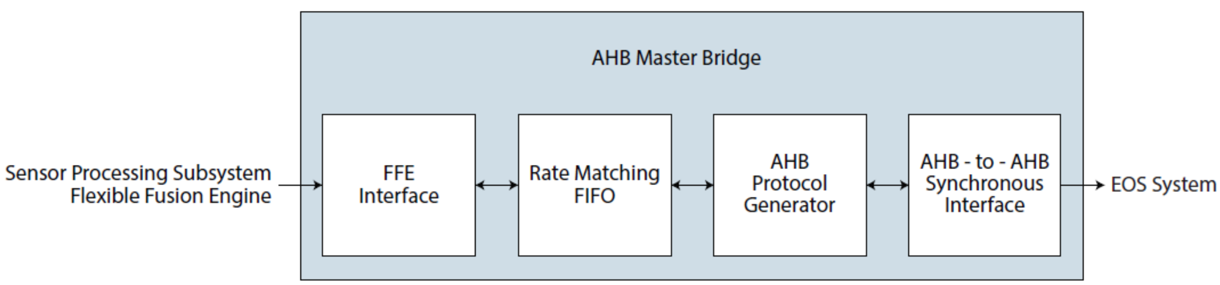
Figure 27: SPI Clock Format 1 (CPHA = 1)



3.6. AHB Master Bridge

The FFE AHB Master Bridge gives the Sensor Processing Subsystem FFE the ability to write directly to some of the EOS S3 platform resources. **Figure 28** shows that this interface is composed of four functional units.

Figure 28: FFE AHB Master Bridge Block Diagram



The primary purpose for the AHB Master Bridge is to enable the Sensor Processing Subsystem FFE to conduct the following operations:

- Enable the Sensor Processing Subsystems FFE to initiate updates of the FFE memories using the M4-F DMA controller. This requires the M4-F to configure the DMA controller in advance of the FFE DMA request.
- Use sections of the M4-F memories as a large FIFO. This enables the storage of a larger amount of processed sensor data than is available via other hardware paths.

3.7. Control Registers

The Control Registers block contains a series of register used for accessing the operations of the Sensor Processing Subsystem. These registers include the following:

- Wishbone Bus access to multiple I^2C 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

3.8. 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.

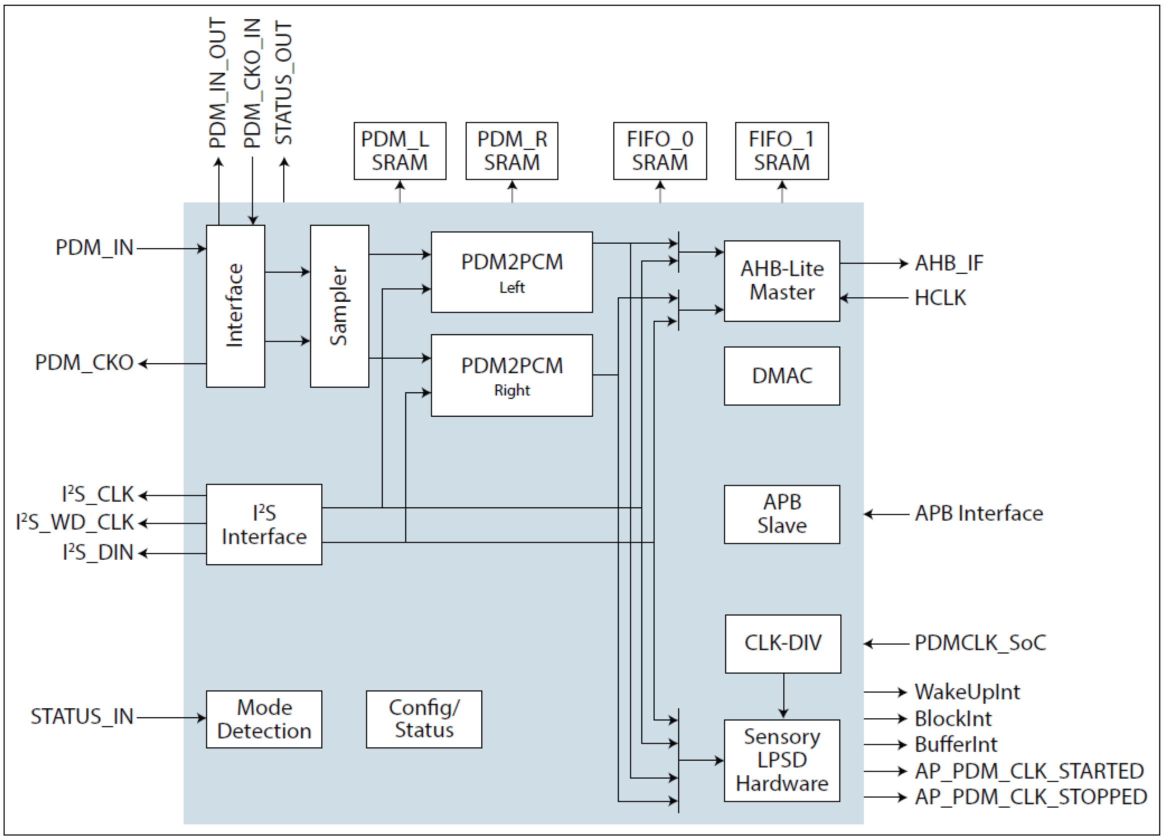
3.9. On-Chip Programmable Logic

The FFE has the capability to pass a Start signal to an IP in the on-chip programmable logic. Similarly, the IP in the on- chip programmable logic can pass a Busy signal to the Sensor Processing Subsystem. The objective is to extend the coordination of Sensor Fusion processing to IP in the on-chip programmable logic.

4. Voice Subsystem

The integrated Voice Subsystem shown in **Figure 29** is designed to support always-on voice capability and has been optimized to work with today's leading-edge voice recognition software. The EOS S3 platform supports two types of digital microphones. Both types of microphones are supported in mono and stereo configuration.

Figure 29: Voice Subsystem Block Diagram



4.1. PDM Microphone

The EOS S3 platform supports PDM microphones in a mono or stereo configuration. As shown in **Figure 29** the incoming PDM data is sampled using the PDM sampler. In case of a mono microphone, software can enable the Left or Right PDM-to-PCM (PDM2PCM) converter. When two microphones are used in stereo configuration, both Left and Right PDM2PCM converters are enabled. The output of the PDM2PCM converters are 16-bit PCM samples at a 16-kHz sample rate. The PCM sample size and sample rate is chosen to support TrulyHandsfree Voice Control voice recognition software.

4.2. I²S Microphones

The EOS S3 platform also supports mono or stereo I²S microphones, as shown in **Figure 29**. The I²S interface inside the EOS S3 platform provides signals needed for interfacing to the microphones, and the outputs are 16-bit PCM samples. The output can be used as is or it can be multiplied by a factor of 2, 4, or 8. In the case where down sampling of the PCM sample is required, the PDM2PCM block can be used to down sample from a 32-kHz sample rate to a 16-kHz sample rate. In situations where incoming PCM samples lack acoustic fidelity, digital gain of up to 34.5 dB can be applied to the PCM samples by PDM2PCM block.

4.3. Low Power Sound Detect Support

To minimize power associated with always on voice processing, the EOS S3 platform supports acoustic activity detection using LPSD hardware. This allows normal PDM or I^2S 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 I^2S microphones and should not be enabled whenusing microphones that have dedicated acoustic detection capability.

4.4. PDM Slave Port for External Codec

As shown in **Figure 29**, the EOS S3 platform provides a PDM Slave port that allows an external application processor to interface to the PDM microphones connected to the EOS S3 device. When this port is enabled, the EOS S3 essentially behaves transparently. The PDM clock driven by application processor, received by the EOS S3 platform on the PDM_CKO_IN input pin is driven out to PDM_CKO output pin of the EOS S3 device. Similarly, PDM data received by the EOS S3 device on its PDM_IN pin from PDM microphones is driven out on the EOS S3 device PDM_IN_OUT pin without modification. There are several software-controlled modes, associated with this port. Depending on the mode and state of clock on PDM_CKO_IN pin, PDM_CKO can be driven by the EOS S3 platform and PDM data can be consumed by the EOS S3 platform.

4.5. DMA and AHB Master Port

PCM samples from the microphones are stored in FIFOs as shown in **Figure 29**. There is a separate FIFO for each microphone. The DMA Controller (DMAC) inside the EOS S3 platform is responsible for transferring the PCM samples from the FIFOs to M4-F SRAM using EOS S3 platform AHB-Lite Master port.

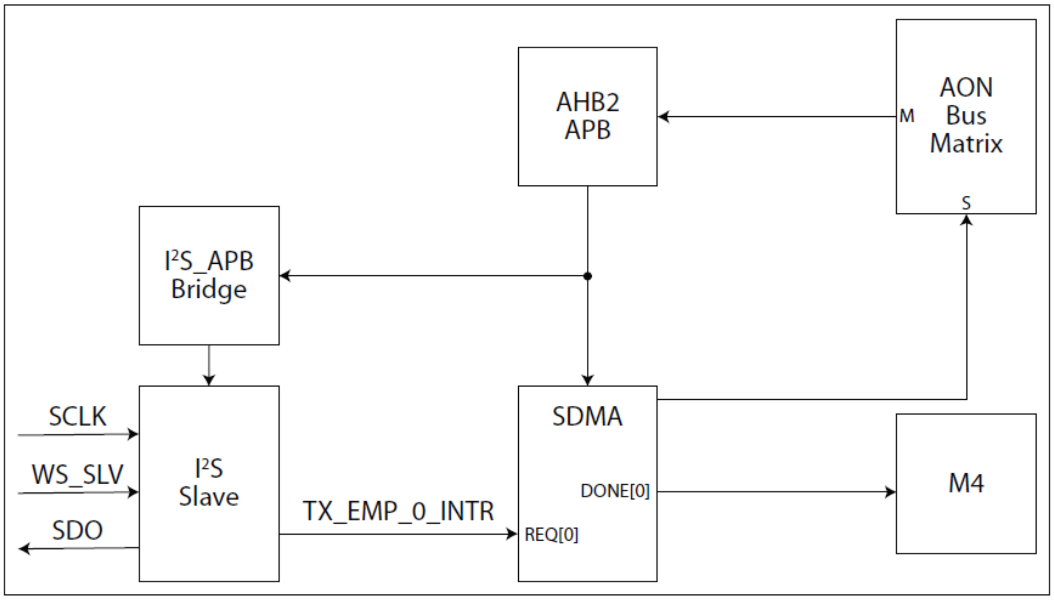
4.6. APB Slave Port

As shown in **Figure 29**, the EOS S3 platform supports an APB Slave port. The M4-F can use this port to access audio configuration registers inside the EOS S3 platform.

4.7. I^2S Slave Port

The EOS S3 platform implements an I^2S Slave port, allowing an external I^2S Master to interface to EOS S3 platform, as shown in **Figure 29**. This port can transfer voice PCM samples from M4-F SRAM to an external I^2S Master, such as an Application Processor or a Codec. Channel 0 of System DMA is allocated for transferring PCM samples from M4-F SRAM to I^2S Slave port. **Figure 30** illustrates connections between external I^2S Master and EOS S3 platform I^2S Slave port. It also shows connections between I^2S Slave port, System DMA (SDMA) and other blocks inside EOS S3 platform.

Figure 30: I²S Slave Port



5. Timing

5.1. I²C Master AC Timing

Figure 31 shows the I²C Master AC timing.

Figure 31: I²C Master AC Timing

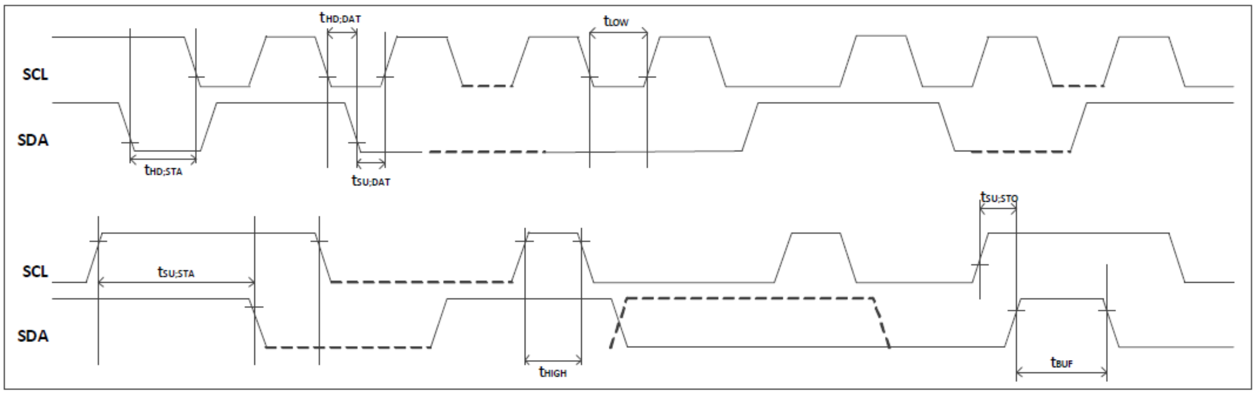


Table 2 describes the I²C Master AC timing parameters.

Table 2: I²C Master AC Timing

Symbol	Description	Standard Mode		Fast Mode		Units
		Min.	Max.	Min.	Max.	
f_{SCL}	Operating frequency.	-	100	-	400	kHz
t_{LOW}	Clock low period.	4.7	-	1.30	-	µS
t_{HIGH}	Clock high period.	4.0	-	0.60	-	µS
$t_{HD;STA}$	Hold time for repeated START condition.	3.10	-	0.60	-	µS
$t_{SU;STA}$	Setup time for repeated START condition.	4.19	-	0.60	-	µS
t_{BUF}	Bus free time between STOP and START condition.	4.7	-	1.3	-	µS
$t_{HD;DAT}$ [a]	Data hold time.	0	-	0	-	µS
$t_{SU;DAT}$	Data setup time.	0.25	-	0.10	-	µS
$t_{SU;STO}$	Setup time for STOP.	4.0	-	0.6	-	µS

a. The receiving device must provide an internal delay of 300 nS for the SDA signal with respect to the SCL signal to bridge the undefined region of the falling edge of SCL.

5.2. I²S Timing

Figure 32 and **Table 3** describe the I²S timing.

Figure 32: I²S Timing Waveform

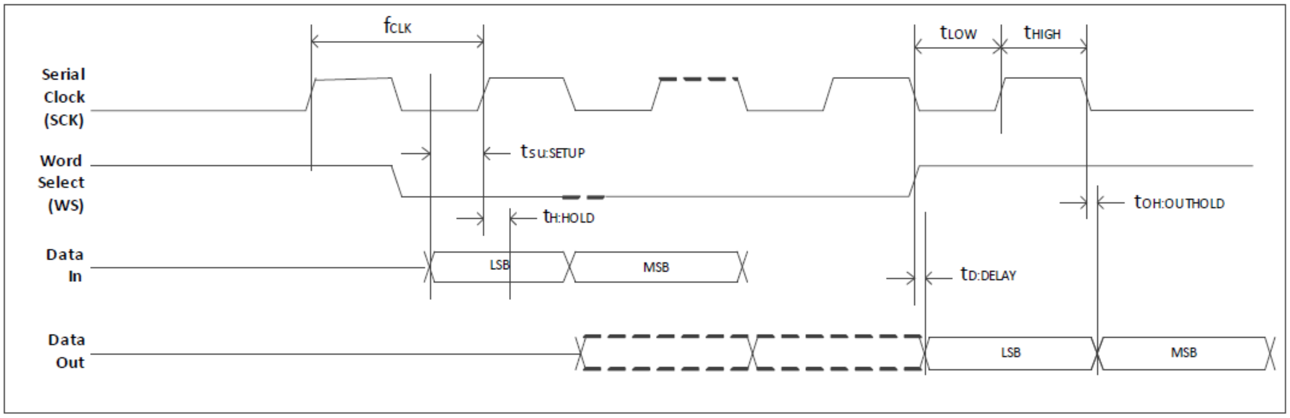


Table 3: I²S Timing

Symbol	Parameter	Min.	Typ.	Max.	Units
f_{CLK}	I²S Clock Frequency	-	-	10	MHz
t_{LOW}	Clock High Period	45	-	-	ns
t_{HIGH}	Clock Low Period	45	-	-	ns
t_{SU}	Data Input Set Up Time	10	-	-	ns
t_H	Data Input Hold Time	1	-	-	ns
t_D	Clock to Data Out Delay	3	-	35	ns
t_{OH}	Clock to Out Hold	2	-	-	ns

5.3. PDM Microphone Timing

Figure 33 and **Table 4** describe the PDM microphone timing.

Figure 33: PDM Microphone Timing

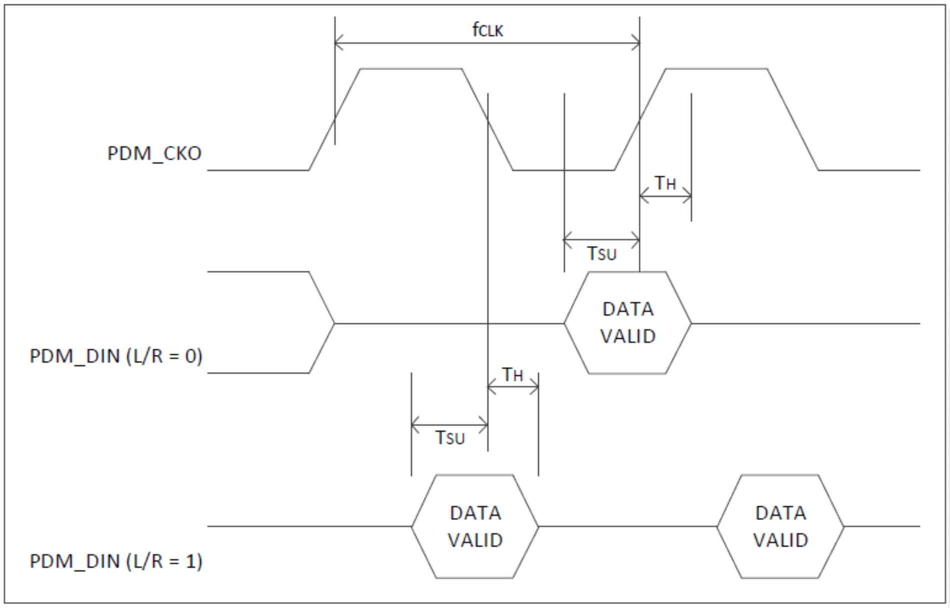


Table 4: PDM Microphone Timing

Symbol	Parameter	Min.	Typ.	Max.	Units
f_{CLK}	PDM Frequency	-	-	10	MHz
t_{SU}	Data Input Set Up Time	10	-	-	ns
t_H	Data Input Hold Time	1	-	-	ns
	f_{CLK} Duty Cycle	48	50	52	%

5.4. SPI Timing

5.4.1. SPI Master

Figure 34 illustrates the SPI Master timing.

Figure 34: SPI Master AC Timing

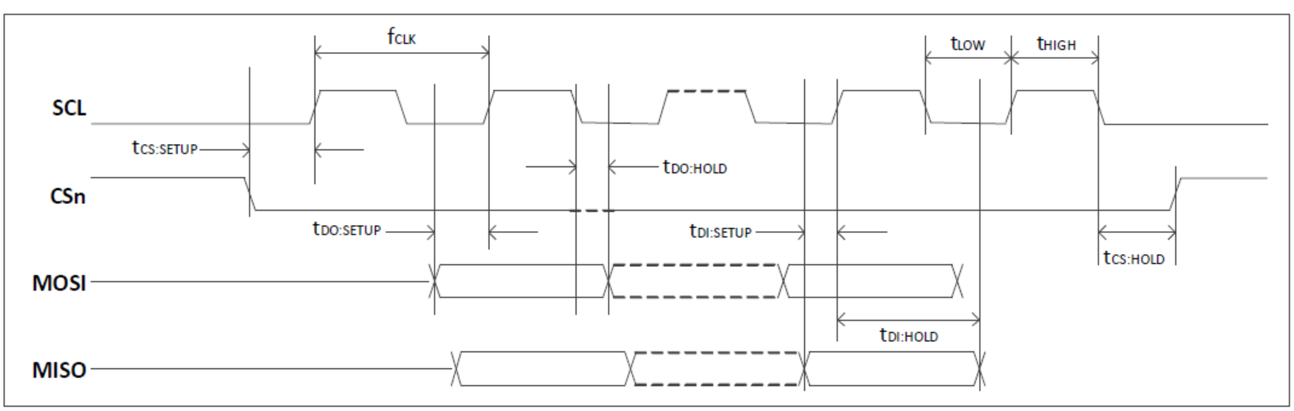


Table 5 describes the SPI Master timing.

Table 5: SPI Master Timing

Symbol	Parameter	Min.	Max.	Units
f_{CLK}	SPI Clock Frequency	-	20	MHz
t_{LOW}	SPI CLK Low Time	24	-	nS
t_{HIGH}	SPI CLK High Time	24	-	nS
$t_{DO:SETUP}$	Data Output Setup Time to slave device	(2/ f_{CLK})-5	-	nS
$t_{DO;HOLD}$	Data Output Hold Time	1.1	4.8	nS
$t_{DI:SETUP}$	Data Input Setup Time	8	-	nS
$t_{DI;HOLD}$	Data Input Hold Time	1	-	nS
$t_{CS:SETUP}$	CS Input Setup Time	50	-	nS
$t_{CS;HOLD}$	CS Input Hold Time	50	-	nS

5.4.2. SPI Slave

Figure 35 illustrates the SPI Slave timing.

Figure 35: SPI Slave Timing

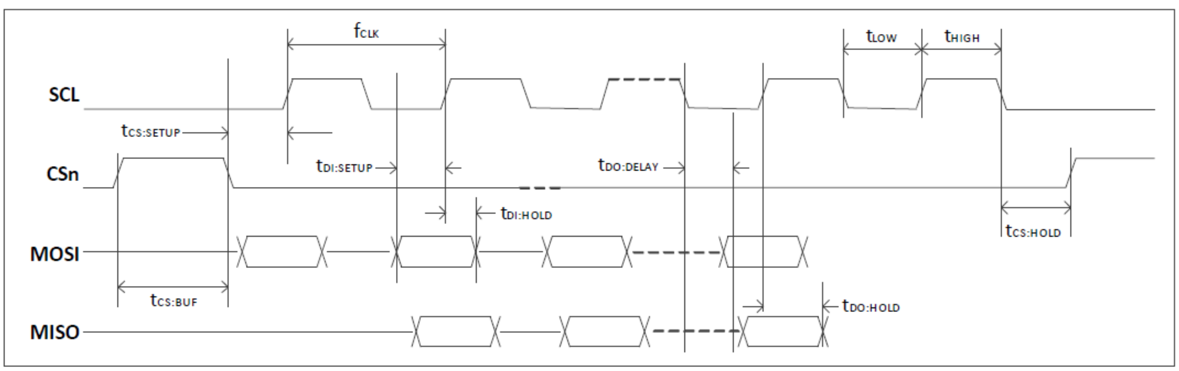


Table 6 describes the SPI Slave timing.

Table 6: SPI Slave Timing

Symbol	Parameter	Min.	Max.	Units
f_{CLK}	SPI Clock	-	20	MHz
t_{LOW}	SPI Clock Low Time	22.5	-	nS
t_{HIGH}	SPI Clock High Time	22.5	-	nS
$t_{DO:SETUP}$	MISO Output Delay from SPI Clock Driving Edge	-	16	nS
$t_{DO;HOLD}$	MISO Output Delay Hold Time	2	-	nS
$t_{DI:SETUP}$	MOSI Input Setup Time	4	-	nS
$t_{DI;HOLD}$	MOSI Input Hold Time	4	-	nS
$t_{CS:SETUP}$	CS Input Setup Time	4	-	nS
$t_{CS;HOLD}$	CS Input Hold Time	4	-	nS
$t_{CS;BUFF}$	CS High Time	50	-	nS

6. On-Chip Programmable Logic

The on-chip programmable logic provides flexibility to the EOS S3 platform for implementing additional functions. The on-chip programmable logic consists of multiplexor-based logic cells, built-in RAM modules and FIFO controllers, built- in multipliers, as well as interfaces with I/O drivers of the EOS S3 device. The major features of the embedded on-chip programmable logic are listed in **Table 7**.

Table 7: On-Chip Programmable Logic Major Features

Feature	EOS S3
Logic Cells	891
8K RAM Modules (512x18 – 9,216 bits)	8
FIFO Controllers	8
RAM Bits	73,728
Configurable Interface	32
Multiplier	2x 32 x 32
	4x 16 x 16

6.1. Functional Description

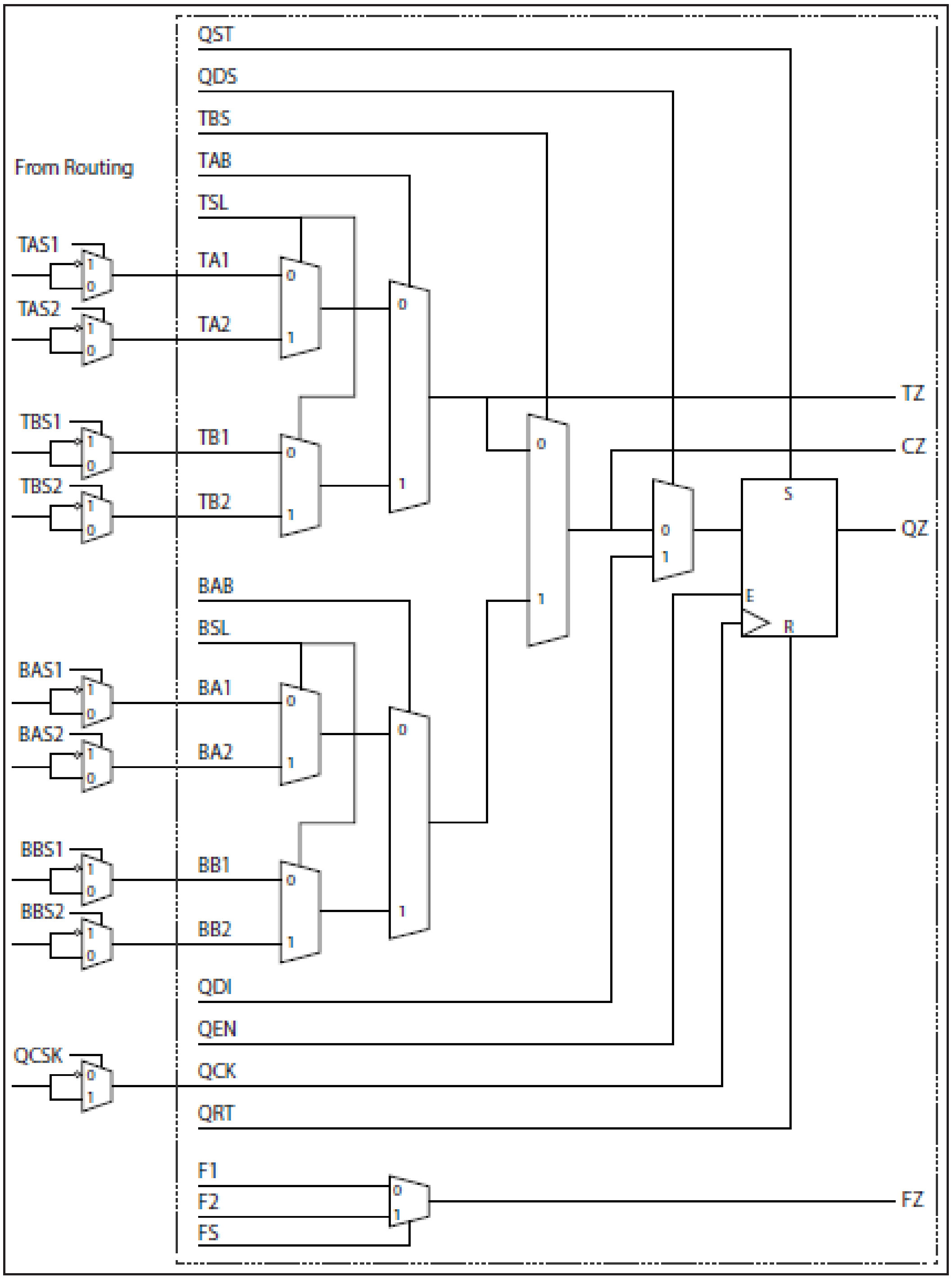
6.1.1. Logic Cell

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 36 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 is capable to 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 36: Logic Cell Block Diagram



6.2. RAM/FIFO

The on-chip programmable logic also includes up to eight instances of 8K (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

6.2.1. FIFO Controller

Every 8K 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

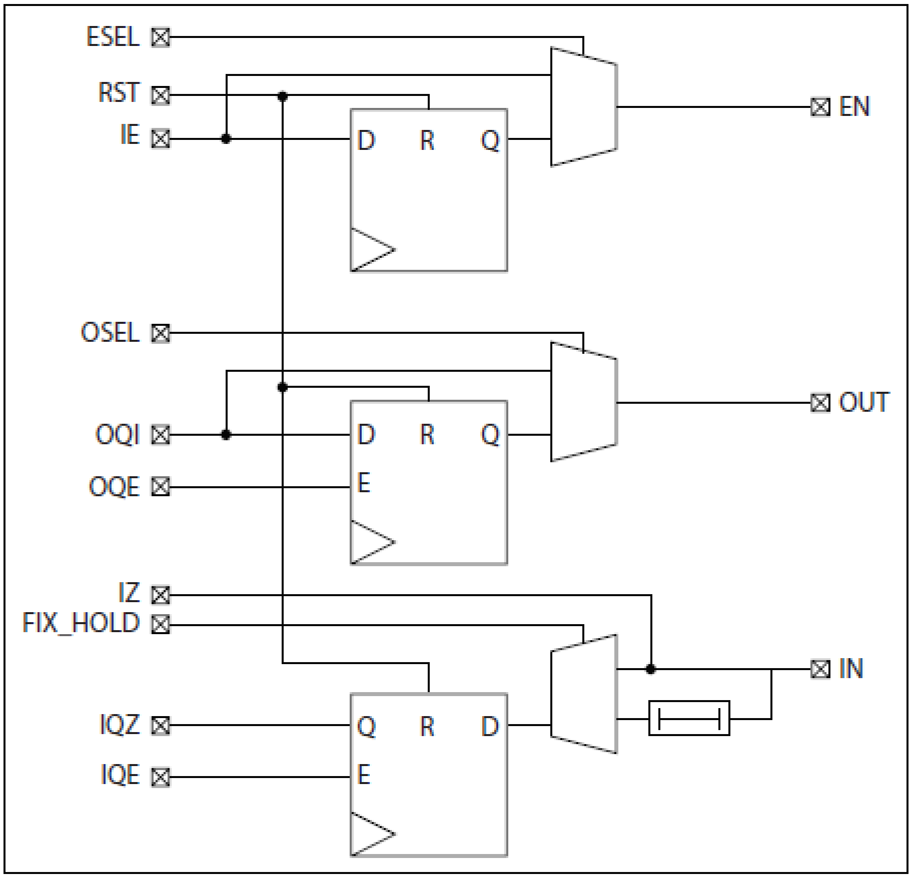
6.2.2. Configurable Input/Output Signals

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

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

See **Figure 37** for on-chip programmable logic configurable I/Os.

Figure 37: On-Chip Programmable Logic Configurable Input/Output

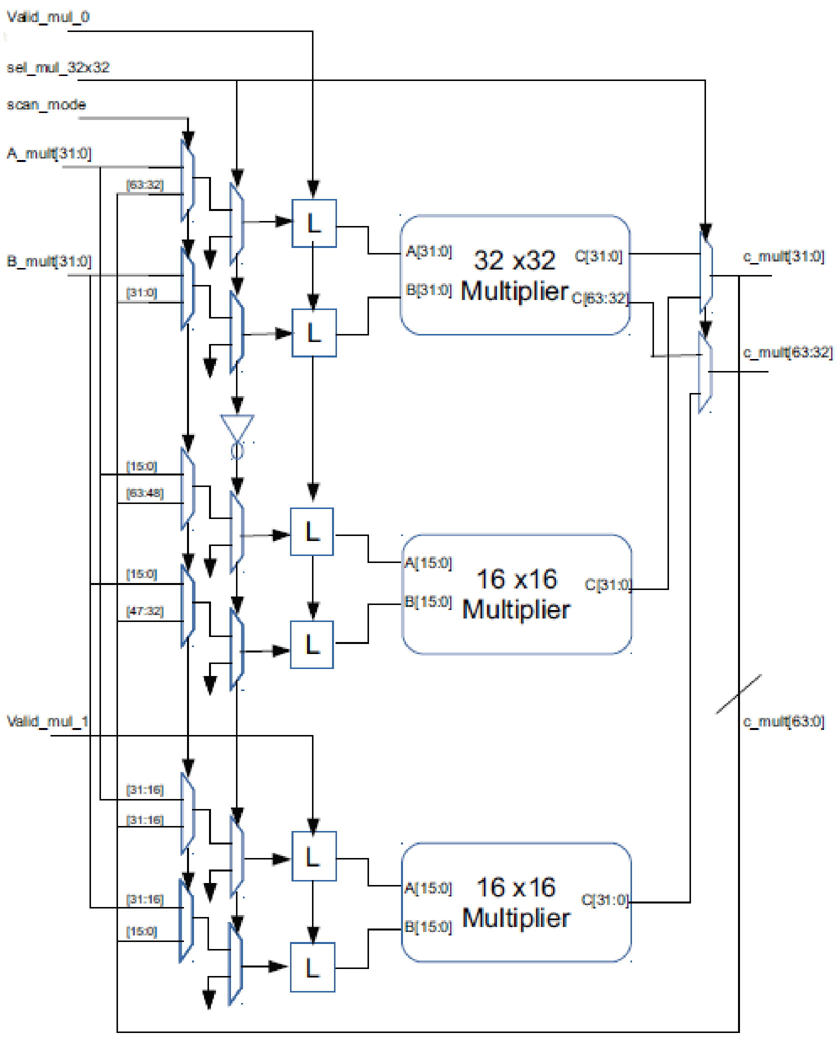


6.2.3. Multipliers

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 38**.

Figure 38: On-Chip Programmable Logic Multiplier



6.3. 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:

- EOS 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.

6.3.1. EOS S3 Platform Interface

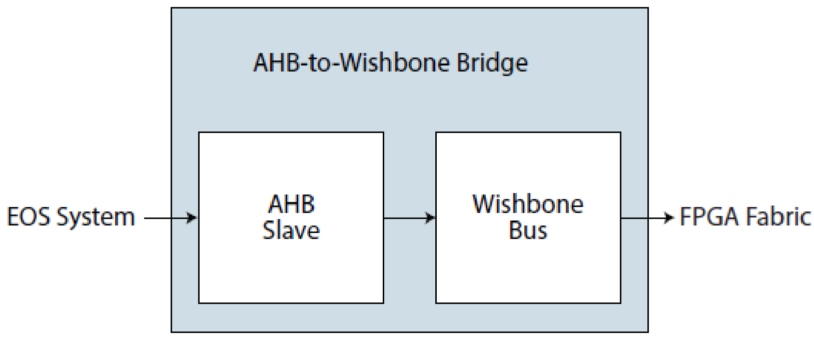
The interface between IP in the on-chip programmable logic and the EOS S3 platform consists of the following:

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

6.3.2. AHB-To-Wishbone Bridge

The AHB-To-Wishbone Bridge provides the means for the EOS 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 39** illustrates the AHB-to-Wishbone bridge.

Figure 39: 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.

6.3.3. SDMA Interface

The EOS S3 platform provides a System DMA (SDMA) module for use by various components. The purpose for the SDMA function is to avoid loading the Host processor with simple data movement operations, and to conduct data movement during periods when the Host Processor is in a low-power state.

For the IP in the on-chip programmable logic, the SDMA interface provides the means to process a block of data and transfer the results to M4-F memory. Examples of this include:

- IP using SDMA to move data from M4-F memory, processing the data, and initiating SDMA to move this finished result back to M4-F memory.
- IP reading data from an external sensor (e.g., through the SPI Master for System Support), processing the data, and using an SDMA operation to save the results to M4-F memory.

Once the SDMA operation completes, an interrupt from the SDMA can signal to the Host that the operation has completed. Alternately, the IP in the on-chip programmable logic can also issue its own interrupt to the EOS S3 platform to signal the completion of its processing operation.

It is important to note that the SDMA module resides at the EOS S3 platform level. Therefore, it can access the M4-F Memory modules and the on-chip programmable logic IP. However, the IP cannot directly access the M4-F Memory modules.

6.3.4. Interrupt Interface

The EOS S3 platform provides a set of interrupt signals to the on-chip programmable logic IP. These interrupt signals enable the IP to signal to the EOS S3 platform that it needs attention. The nature of this attention depends upon the IP and the nature of the required processing.

6.3.5. Sensor Processing Subsystem Interface

The Sensor Processing Subsystem can drive IP residing in the on-chip programmable logic with a Start signal and sample a Busy status signal. These signals enable the IP to coordinate its processing with that of the Sensor Processing Subsystem. This allows both modules to provide processed data to the EOS S3 platform for further evaluation or Sensor Fusion processing.

It is important to note that there is no direct connection between the Sensor Processing Subsystem and the on-chip programmable logic for the passing either processed or raw data. Instead, the data must first be passed to the EOS S3 platform (e.g., the M4-F Memory) before it can be passed to either the Sensor Processing Subsystem (e.g., from the on- chip programmable logic IP) or to the on-chip programmable logic IP (e.g., from the Sensor Processing Subsystem).

6.3.6. Packet FIFO Interface

IP within the on-chip programmable logic can pass data to the EOS S3 platform via the Packet FIFO interface. Like the Sensor Processing Subsystem, the IP logic can write raw or processed data into the Packet FIFO in a software-defined packet format.

It is important to note that the Sensor Processing Subsystem and on-chip programmable logic-based IP do not share the same connection to the Packet FIFO. Rather, each module uses a separate port to connect to the Packet FIFO. However, within the Packet FIFO, each port must be assigned by software to a separate internal FIFO (e.g., the Packet FIFO is composed of sub-FIFO 8K, 0, 1, and 2).

7. Power Management

7.1. Low Dropout Regulators

The EOS S3 platform has two on-chip low dropout regulators (LDOs). One LDO is for SRAM, and the other LDO is for digital logic. By having a separate regulator for the SRAM, the SRAM voltage can be further reduced to save power consumption.

Table 8: LDO Regulators

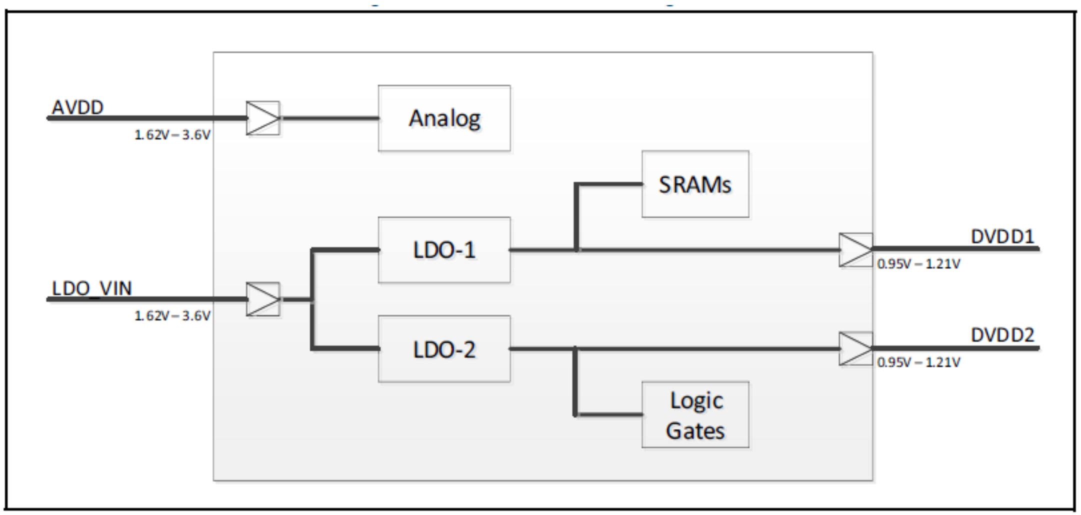
LDO	Maximum Current	Output Voltage Range
LDO-1 for SRAM	50 mA	0.95V ~ 1.21V
LDO-2 for Digital Logic	30 mA	0.95V ~ 1.21V

There are three possible LDO use cases for EOS S3 platform, which allows for flexibility customer in configuration. Each is illustrated in **Figure 40** through **Figure 42**.

7.1.1. Use Case 1: Dual Voltage Rail Supplied by On-chip LDOs

In this example, all SRAMs are supplied by LDO-1 and Logic Gates are provided by LDO-2.

Figure 40: Use Case 1: Dual Voltage Rail

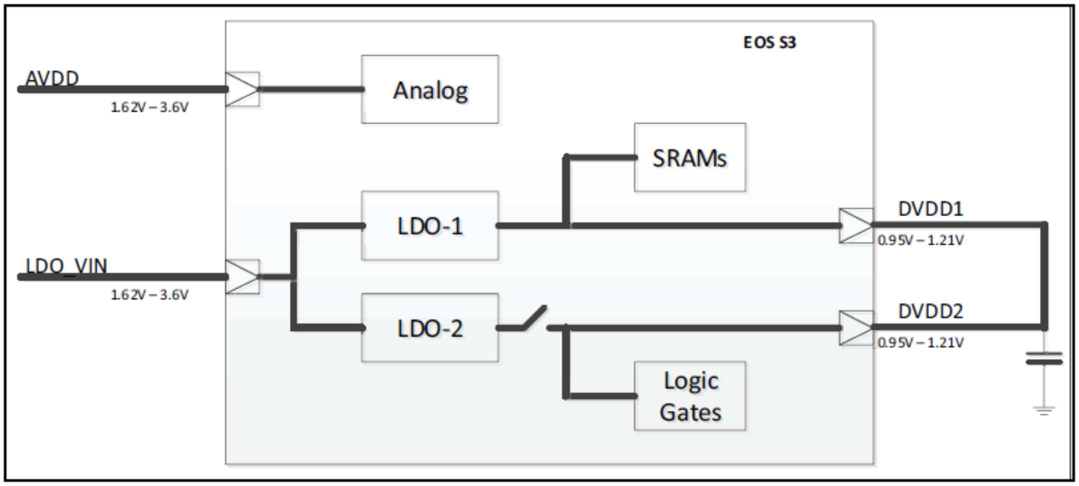


7.1.2. Use Case 2: Single Voltage Rail Supplied by Single On-chip LDOs

In this example, the SRAMs and Logic Gates are provided by LDO-1. The DVDD2 and DVDD1 pads are tied together on the board. After voltage is applied to LDO_VIN, both LDO-1 and LDO-2 are on. To further reduce the device core power consumption, turn off LDO-2 with firmware.

NOTE: LDO-1 supports up to 50 mA loading.

Figure 41: Use Case 2: Single Voltage Rail



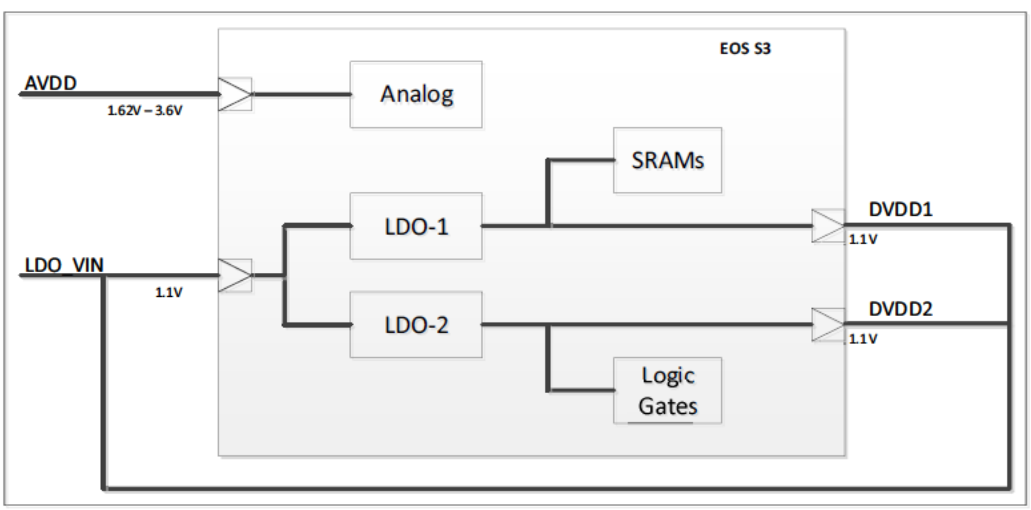
7.1.3. Use Case 3: External Voltage Supplied

In this example, the EOS S3 platform LDOs are bypassed and SRAMs and Logic Gates power are externally supplied through DVDD1 and DVDD2 pads. The LDO_VIN, DVDD2 and DVDD1 pads are tied together on the board and hook up to the supply voltage.

NOTE: In this configuration, an external voltage of 1.1V ±50 mV is required to be applied to LDO_VIN, DVDD1 and DVDD2. The device requires 1.05 V minimum to come out of Power-On reset. See **Table 28** for DC specifications.

For bypass configuration, external power supplies are used. Therefore, turn off the internal LDOs with firmware to further reduce the device core power consumption.

Figure 42: Use Case 3: External Voltage Supplied



7.2. Power-On Sequence

The recommend power-on sequence for the EOS S3 platform is shown in **Figure 43**.

Figure 43: Power-On Sequencing

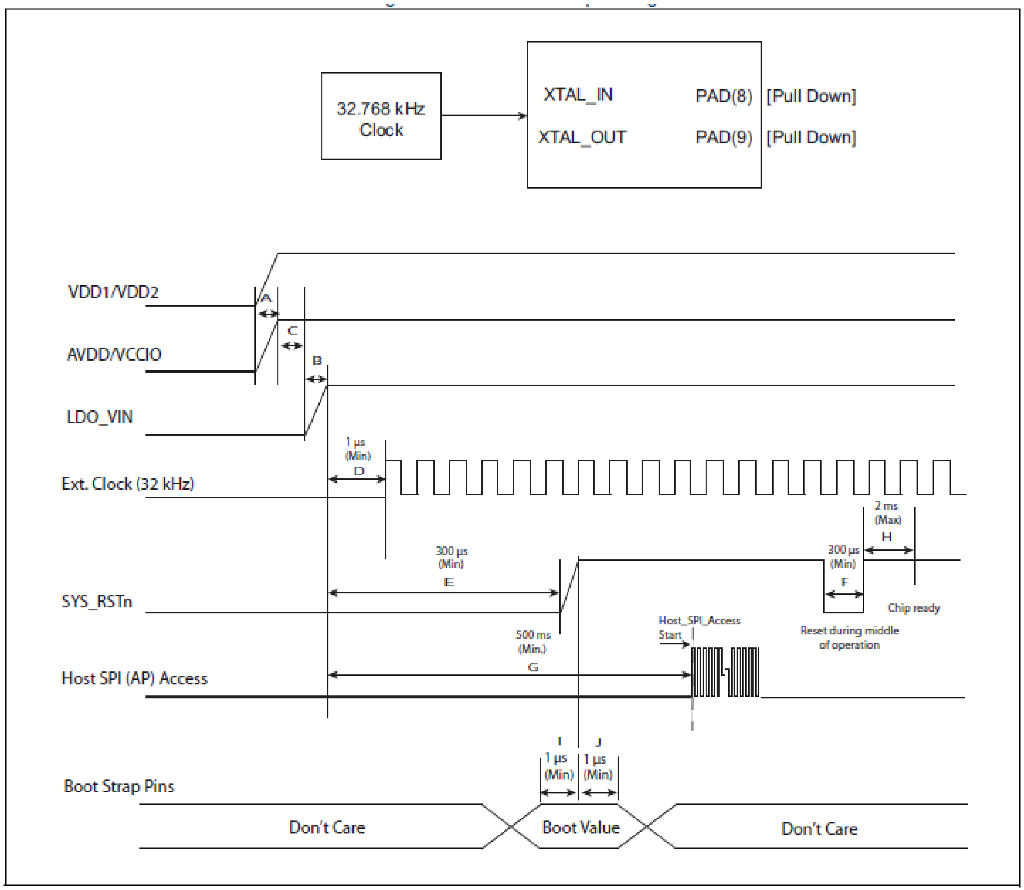


Table 9 shows the power-on sequencing timing parameters.

Table 9: Power-On Sequencing Timing Parameters

Letter	Parameter	Condition	Min.	Typ.	Max.	Unit
A	AVDD Voltage Rising Time	From 0V to Target operating voltage	See Note	See Note	See Note	ms
B	LDO_VIN/VDD1/VDD2 Rising Time	From 0V to Target operating voltage	See Note	See Note	See Note	ms
C	VCCIOA/VCCIOB Rising Time	From 0V to Target operating voltage	See Note	See Note	See Note	µs
D	Voltage Ready to XTAL_IN (CMOS CLK)	All voltage rails at 90% voltage to CMOS CLK	1	-	-	µs
E	Voltage Ready to SYS_RSTn Release	All voltage rails at 90% voltage to SYS_RSTn is released	300	-	-	µs
F	SYS_RSTn Middle of the Operation	Host SPI Access is available 2 ms after SYS_RSTn is released	300	-	-	µs

G	Voltage Ready to HOST_SPI_Access Start	Conditions A through F must be met	500	-	-	ms
H	System Reset Release to Chip Exits Reset State	Chip ready 2 ms after SYS_RSTn release	-	-	2	ms
I	Bootstrap pins setup time	Setup time with respect to SYS_RSTn	1	-	-	µs
J	Bootstrap pins hold time	Hold time with respect to SYS_RSTn	1	-		µs
	CMOS CLOCK is active after AVDD is powered down[a]	CMOS CLOCK is required to be active for a minimum time if AVDD is powered down	2.0	-		ms

a. This is only required for a system that has intermittent power-down interruption. It is not required for AVDD always-on or CMOS CLOCK always-on.

NOTE: It is required to power up AVDD first. AVDD, VDD and VCCIO can be powered up together. However, avoid turning on VDD and VCCIO at the same time. Power on VDD and then Power on VCCIOA/B; check the ramp up time for VCCIO and VDD to ensure that VDD reaches 0.6v before VCCIO reaches 1V. This is to prevent contention with other signals in the system (including SYS_RSTn signal), refer to **Table 27** for more information. To avoid high current during power-on sequence, VDD must not be powered on before AVDD. The recommended power sequence is AVDD–>VDD–>VCCIO.

NOTE: When using internal LDO for VDD, bring up AVDD first, followed by LDO (VDD) and VCCIO. Internal LDO ramps up after AVDD reaches 1.5v and internal LDO response time/ramp up time is 200us.

NOTE: Device initiation can begin after SYS_RSTn timing is met. SYS_RSTn signal is held low before power rails reach 90%. Initialization before SYS_RSTn timing is met can result in bootup failure. There is a weak pull-up inside SYS_RSTn, see **Table 30** for the resistance value.

Figure 44: Current Measurement Scheme with External Power Supplies

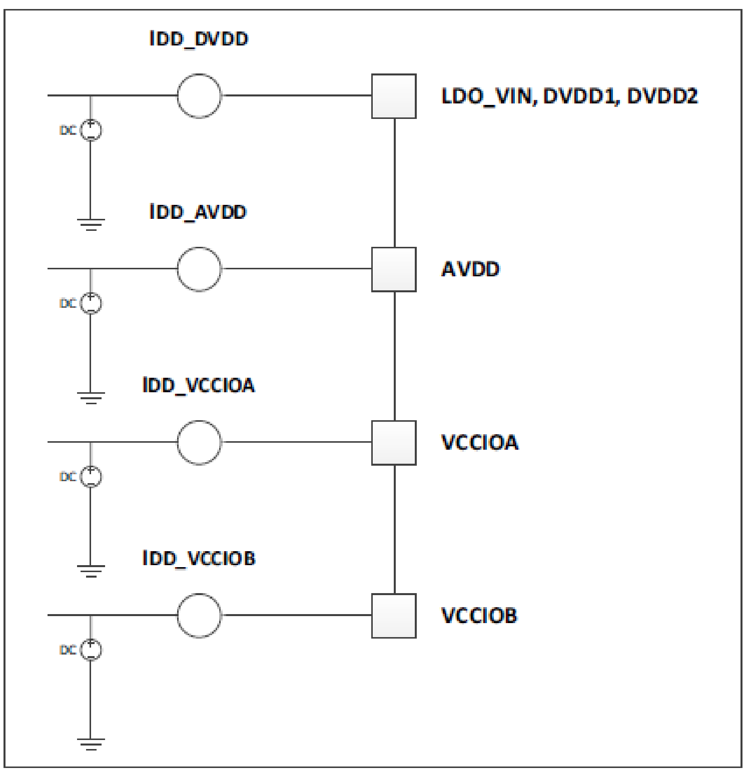


Table 10 and **Table 11** show the current measurement on each supply rail for power-on sequence in LDO bypass mode (external voltage supply). All I/Os and dedicated pins are tri-stated, and all external caps are removed.

Table 10: Current Measurements for Power-On Sequence[a]

Power Sequence	1 DVDD (1.1V)	2 AVDD (1.8V)	3 VCCIOA (1.8V)	4 VCCIOB (1.8)
IDD DVDD	275 µA	147 µA	147 µA	183 µA
IDD VCCIO A	0.01 µA	0.01 µA	0.05 uA	0.03 uA
IDD VCCIO B	0.01 µA	0.01 µA	0.01 uA	0.3 uA
IDD AVDD	-60 µA	19.8 µA	19.8 µA	19.8 µA

a. Typical values are based on 25°C and nominal voltage (VDD1=VDD2=1.1V, VCCIO=1.8V).

Table 11: Current Measurements for Power-On Sequence[a]

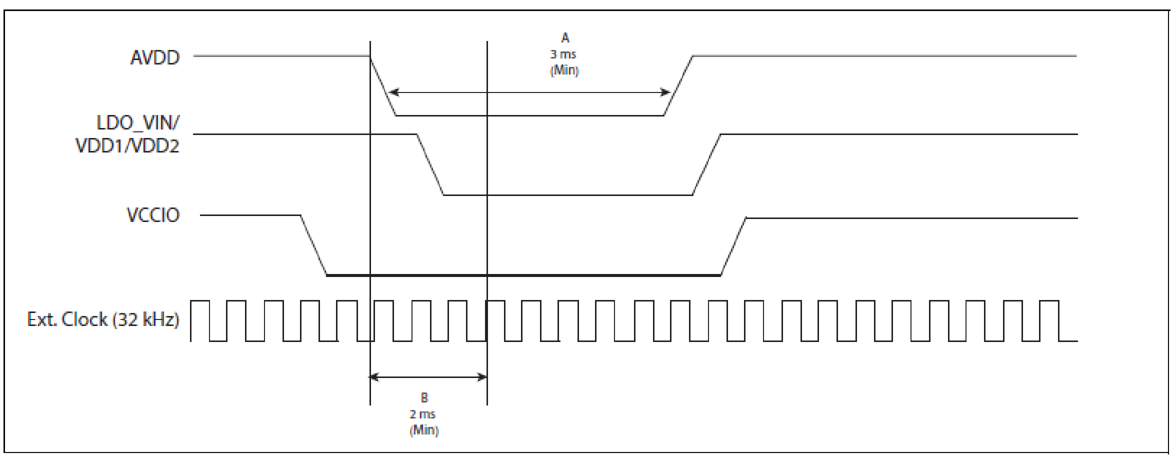
Power Sequence	1 AVDD (1.8V)	2 DVDD (1.1V)	3 VCCIOB (1.8V)	4 VCCIOA @ 1.8V
IDD DVDD	-0.08 µA	147 µA	147 µA	183 uA
IDD VCCIO A	0.01 µA	0.01 µA	0.05 uA	0.03 uA
IDD VCCIO B	0.01 µA	0.01 µA	0.01 uA	0.3 uA
IDD AVDD	20.0 µA	19.8 uA	19.9 uA	19.8 uA

a. Typical values are based on 25°C and nominal voltage (VDD1=VDD2=1.1V, VCCIO=1.8V).

7.3. Power-Down Sequence

The recommend power-down sequence for the EOS S3 platform is shown in **Figure 45**.

Figure 45: Power-Down Sequencing



NOTE: Recommended power down sequence: VCCIO > VDD/AVDD. Power down VCCIO before VDD. Power down all power supplies at same time is OK. Power down AVDD/VCCIO together then VDD is OK.

NOTE: For AVDD ensure CMOS clock is active during AVDD ramp down and after 2ms AVDD is powered down. Refer to **Table 12** for AVDD. This is only required for a system that has intermittent power-down interruption for battery savings. It is not required for AVDD always-on or CMOS CLOCK always-on or when using XTAL. For power-on sequence refer to **Table 9** on page 54.

Table 12 shows the power-down sequencing timing parameters.

Table 12: Power-Down Sequencing Timing Parameters

Letter	Parameter	Condition	Min.	Typ.	Max.	Unit
A	AVDD Voltage Power-Down Duration Time	AVDD Voltage Power-Down Duration Time	3.0	-	-	ms
B	CMOS CLOCK is active after AVDD is powered down[a]	CMOS CLOCK is required to be active for a minimum time if AVDD is powered down	2.0	-	-	ms

a. This is only required for a system that has intermittent power-down interruption. It is not required for AVDD always-on or CMOS CLOCK always-on.

Table 13 shows the current measurement on voltage rail after power down with active CMOS clock.

Table 13: Current Measurements for Power-Down [a]

Power Sequence	AVDD 1.8V	VDD 1.1V	VCCIOB 1.1V	VCCIOA 1.1V
IDD with CMOS clock active	-2.8 mA	0	0	0

b. Typical values are based on 25°C and nominal voltage (VDD1=VDD2=1.1V, VCCIO=1.8V).

Table 14 and **Table 15** shows the typical inrush current for each mode with one voltage rail powerup.

Table 14: LDO Mode Typical Inrush Current[a]

Mode	Data (mA)
AVDD @ 1.8V	0.013
VCCIOA @ 1.8V	3.662
VCCIOA @ 3.3V	65.640
VCCIOB @ 1.8V	51.630
VCCIOB @ 3.3V	114.000
LDOVIN @ 1.1V	4.000

a. All IOs are tri-stated or not driven, including SYS_RSTn.

Table 15: LDO Bypass Mode Typical Inrush Current

Mode	Data (mA)
AVDD @ 1.8V	0.020
AVDD @ 3.3V	0.023
VCCIOA @ 1.8V	7.598
VCCIOA @ 3.3V	77.610
VCCIOB @ 1.8V	42.550
VCCIOB @ 3.3V	269.200
LDOVIN @ 1.1V	4.144

Table 16 shows maximum supply power consumption for DVDD rail.

Table 16: Maximum Supply Power Consumption

Mode	Data (mA)
LDO Mode @ 1.8V	80
LDO Bypass Mode @ 1.1V	80

7.4. Clocks and Resets

7.4.1. Clocks

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 17** for a full listing of the clock domains.

Table 17: Clocks Listing

Block Name(s)	Maximum Frequency[a]	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	The 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	
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	
PKT FIFO (FPGA clock)	10 MHz	C41	n/a	
PKT FIFO (FFE clk)	10 MHz	C08	X1_P3_PF	
M4-F Subsystem Domain				

M4-Complex: M4-F 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	
SRAM Domain				
SRAM 128 Kbyte instance 0 (AHB clock)	80 MHz	C10	FCLK_P1_MS0	
SRAM 128 Kbyte instance 1 (AHB clock)	80 MHz	C10	FCLK_P2_MS1	
SRAM 128 Kbyte instance 2 (AHB clock)	80 MHz	C10	FCLK_P3_MS2	
SRAM 128 Kbyte instance 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	
I^2S clock	5 MHz	C30	P2_AD4	
LPSD clock	1 MHz	C31	P3_AD3	
FPGA Domain				
FPGA	10 MHz	C16	P0_FB	
FPGA	10 MHz	C21	P0_FB	
AHB2WB for FPGA clock	10 MHz	C40	P0_FB	
FPGA to Packet FIFO clock	10 MHz	C41	P0_FB	

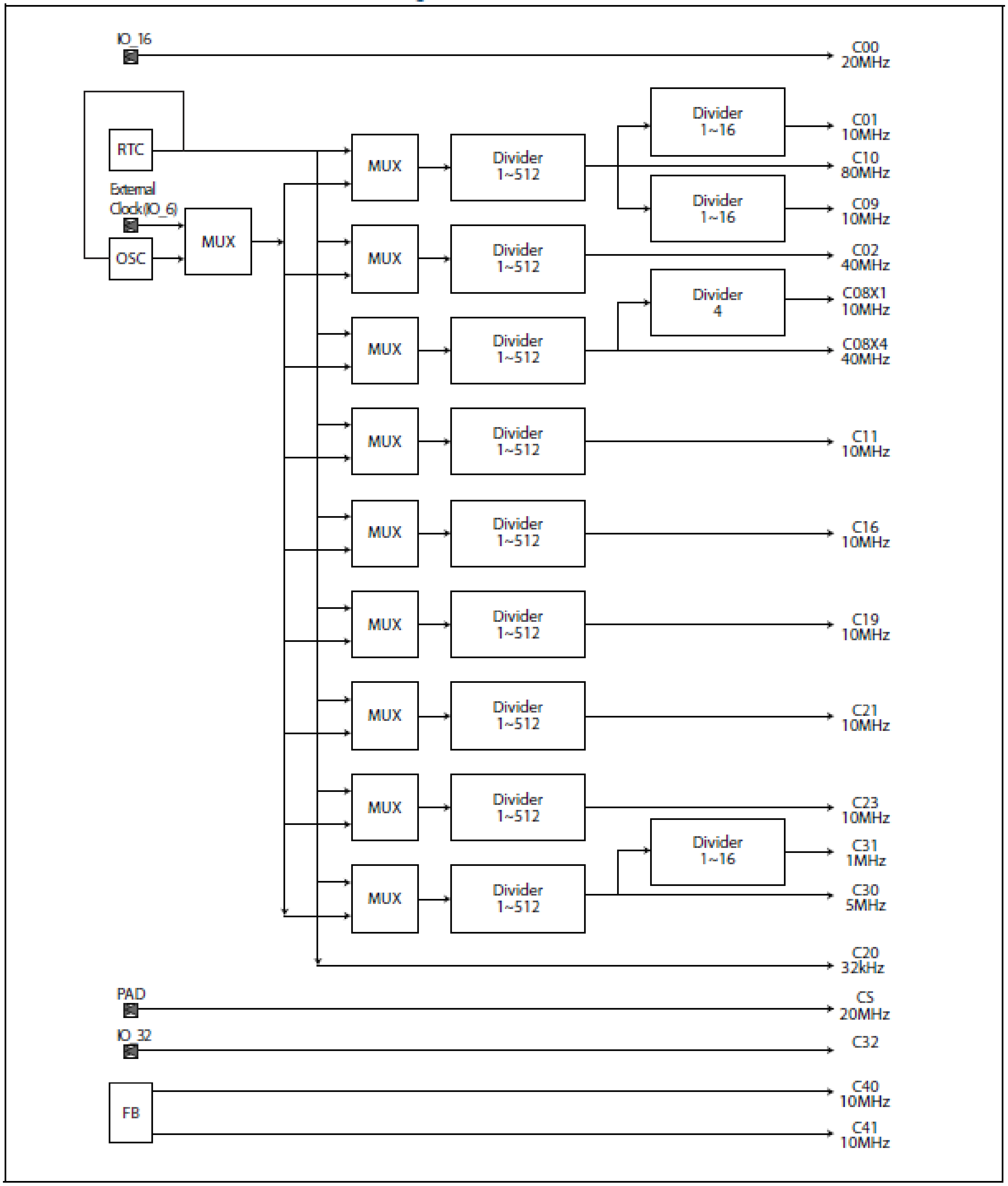
a. Maximum frequency is with VDD = 1.1V ±10%.

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).

For most of the clock domains, there are three possible sources:

- Fast Clock Driving from IO_6 - FCLK
- Real-Time Clock (RTC - 32 kHz)
- Oscillator Clock (OSC - Max. is 80 MHz). See **Figure 46** for details.

Figure 46: Clock Tree



For Clock domains 01, 09 and 10, the clock phase is locked, but the frequency may be different, such as Clock domain C30 and C31. For Clock C08X4 and C08X1, the clock phase is locked and the frequency of C08X1 is always one-fourth of C08X4 clock frequency.

Most Clock paths can be gated by software independently, the exceptions are Clock C00_P0, C01_P0, C10_HCLK_P0, C10_FCLK_P0, C20_P0 and C23_P0. Software can also gate off the clock domains individually. For Clock domains C40 and C41, the clock gating scheme depends on the design inside the on-chip programmable logic.

7.4.2. Resets

Each Clock path has its corresponding Reset Path. Most of them are asynchronous asserted and synchronous released except for the Reset path for non-free running clock domains (e.g., clock paths from IOs, such as C00, C40, C41, CS and C32). The reset for the M4-F core will is not de-asserted until the clock is toggled for a minimum of four cycles.

There are two possible global reset sources:

- Power-ON-Reset
- SYS_RSTn (System Reset)

After booting up, software can program a PMU register to block the SYS_RSTn (System Reset) and treat it as one of the interrupt sources. The software can also reset some of the modules such as Voice Support by programming register bits. For details about the PMU register, see the *QuickLogic E0S S3Registers*.

8. Other EOS S3 Platform Features

8.1. Multi-Function Inputs/Outputs (IOs)

There are 46 I/Os for BGA package and 27 I/Os for WLCSP package that can be muxed for various functions. Each I/O output can have up to 4 different functional outputs. Each functional input can be selected from up to 8 different I/Os. The controls for I/Os (such as output enable, drive strength, etc.) can be controlled from three different sources; the A0 registers, the on-chip programmable logic and other sources (such as M4-F, FFE, etc.). Refer to **Table 26** on page 79 for more IO options. Complete programming examples can be found in the QuickLogic *EOS S3 Sensor Processing Platform Input Output Multiplexor User Guide*.

8.2. General Purpose Inputs/Outputs (GPIOs)

Of the 46 multi-functional IOs, only 8 can be used as GPIOs by M4-F to drive or sample from registers. Each of the 8 GPIOs can be assigned to 2 different IOs. Following are possible IO assignments for each of the 8 GPIOs. Refer to the QuickLogic *EOS S3 Sensor Processing Platform Input Output Multiplexor User Guide* for programming details.

- IO_6 or IO_24 can be GPIO 0
- IO_9 or IO_26 can be GPIO 1
- IO_11 or IO_28 can be GPIO 2
- IO_14 or IO_30 can be GPIO 3
- IO_18 or IO_31 can be GPIO 4
- IO_21 or IO_36 can be GPIO 5
- IO_22 or IO_38 can be GPIO 6
- IO_23 or IO_45 can be GPIO 7

IMPORTANT: When doing system design, not all IOs can be used as M4-F controllable GPIOs.

8.3. Fabric Inputs/Outputs (FBIOs)

Alternately, the 46 multi-functional IOs can be driven by on-chip programmable logic. This is listed as FBIO(x) in **Table 26** on page 79 in the Alternate Function column. Each IO can be driven by on-chip programmable logic as FBIO. For example, IO_0 is FBIO_0, IO_1 is FBIO_1, and so on. Refer to the QuickLogic *EOS S3 Sensor Processing Platform Input Output Multiplexor User Guide* for programming details.

IMPORTANT: While this option gives more flexibility for system design, it does consume more power as the on-chip programmable logic requires to be powered on and configured to utilize this feature.

8.4. Interrupts

Interrupts generated by EOS S3 device subsystem events can be routed to the Application Processor (AP) or the M4-F.

8.4.1. Interrupt Structure

Interrupts in the system can be routed to two different destinations:

- M4-F processor: All interrupts to M4-F connect to M4-F NVIC, with two levels of interrupt masking and clearing, one at the interrupt source, the other at the top-level interrupt controller.

- AP: Interrupt mechanism for the AP is the same as M4, but with a different mask. All interrupt sources are muxed to a single, combined interrupt before being sent to the AP.

8.4.2. Interrupt Sources

Interrupts sourced from each subsystem functional blocks get combined into one interrupt for each subsystem.

- TOP Interrupts

  Sensor/GPIO Interrupts - Eight pins can be used for sensor interrupts or generic GPIO interrupts depending on system requirements. Each interrupt can be configured to use either *edge* detection (configurable to the positive or negative edge) or *level* detection (configurable to a high-level or a low-level setting).

- M4-F Subsystem Interrupts
  - FPU - The Floating Point Unit (FPU) can generate interrupts on floating point events.
  - Bus Timeout - There are bus timeout monitors that prevent AHB/APB Slaves from locking up a system. If there is no response after 1,024 clock cycles, an interrupt can be generated.
  - UART - The UART can generate transmit FIFO, receive FIFO, receive timeout, modem status, and error condition interrupts.
  - Timer - An interrupt can be triggered when the timer counts down to zero.
  - Watchdog Timer - Software can enable the watchdog timer, and after counting down to 0, an interrupt can be triggered. If the interrupt is not cleared, a reset is triggered.
  - SRAM - An interrupt can be triggered when any segment of the 512 KB (16 instances of 32 KB) M4-F memory is accessed when the memory is in a lower power state (e.g., in deep sleep or in shutdown mode).
- FFE Interrupts
  - FFE Message - Eight interrupt messages can be used by FFE for various purposes.
  - FFE Subsystem - Sixteen interrupts are generated from the FFE subsystem, and they are combined into a single interrupt source.
- A0 Interrupts
  - AP Re-Boot -This interrupt is asserted when there is a need for rebooting. This occurs when all the M4-F SRAMs are shut down for power savings, and upon wake up, rebooting is necessary.
  - Reset Interrupt - The SYS_RSTn pin can be used to generate an interrupt.
  - ADC Interrupt - Interrupt generated upon completion of analog to digital conversion.
  - PMU Timer - This 16-bit timer (with 32 kHz clock source) can be used to wake up FFE0 from low power mode before the FFE kickoff timer expires.
  - Software Interrupts - Two software interrupts can be triggered by software for handshaking between AP and the M4-F.
  - LDO Power Good - Independent interrupts from LDO-2 and/or LDO-1 are triggered when the voltage falls below the threshold value.
- A1 Interrupts
  - Configuration DMA - In wearable mode, an interrupt is asserted after the DMA download from flash to M4-F memories is completed.
  - SPI Master Interrupt - The configuration block can generate a combined interrupt, and this includes interrupts from the SPI Master.
- Voice Interrupts
  - LPSD Voice Detect – Interrupt is triggered when voice is detected by LPSD HW.
  - DMIC Voice Detect – Interrupt is triggered when voice is detected by DMIC.
  - DMIC Voice Off – Interrupt is triggered when HW Loss of Voice is detected by DMIC.
  - LPSD Voice Off - Interrupt is triggered when Loss of Voice is detected by LPSD.

    - DMAC0 Block Done – DMAC0 finished transfer of a block size of data.
    - DMAC1 Block Done – DMAC1 finished transfer of a block size of data.
    - DMAC0 Buffer Done – DMAC0 finished transfer of a buffer size of data.
    - DMAC1 Buffer Done – DMAC1 finished transfer of a buffer size of data.
    - AP PDM Clock ON – AP PDM Clock is detected.
    - AP PDM Clock OFF – Loss of AP PDM Clock is detected.
- SDMA Interrupts
    - SDMA Done – Interrupt per each channel of DMA (0-11) when DMA is completed.
    - SDMA Error – Interrupt for SDMA Error
- PKFB Interrupts

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

  Four outputs from the on-chip programmable logic can be used as messages. Each interrupt message can be selected to be either *edge* or *level* detection. For edge detection, it can be configured to be positive or negative edge; similarly, if selected to be level detect, it can be configured to be level high or low.

8.4.3. M4-F Wake-Up Events

The following interrupts sources can also be used as wake-up events for the M4-F. The M4-F can configure the system to wake-up after a certain event occurs (and can even power itself down in the interim to conserve power). The PMU wakes up the M4-F when any of the following interrupts are detected.

- Software interrupts
- FFE interrupts
- On-chip programmable logic interrupts
- Sensor/GPIO interrupts
- M4-F SRAM sleep interrupt
- UART
- TIMER
- WDOG Interrupt/Reset
- Bus Timeout
- FPU
- PKFB
- I^2S
- Audio
- Configuration DMA
- Configuration SPI Master
- PMU Timer
- ADC Done
- RTC Alarm
- Reset Interrupt
- FFE Message
- FFE Combined

- AP Boot
- LDOs Power Good Interrupts
- SRAM Timeout
- LPSD Voice Detect
- DMIC Voice Detect
- SDMA DONE Channel 1–11

NOTE: SDMA Channel 0 (I^2S Slave) does not wake up the M4-F.

8.5. Bootstrap Modes

The EOS S3 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. For timing details, see **Figure 47** on page 66.

Figure 47: Bootstrap Timing

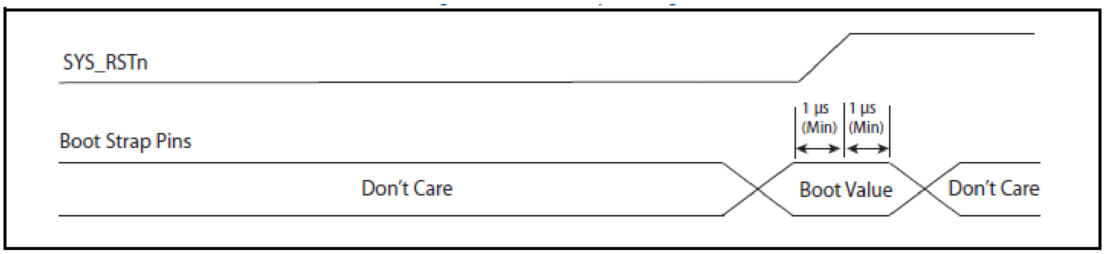


Table 18 shows the bootstrap timing during power-on sequence.

Table 18: Bootstrap Timing During Power-On Sequence

Symbol	Parameter	Min.	Typ.	Max.	Unit
T_{BSU}	Bootstrap pins setup time	1	-	-	µs
T_{BH}	Bootstrap pins hold time	1	-		µs

8.6. M4-F Serial Wire Debug Port Configuration

IO_19 assert high is used for Serial Wire Debug with pin IO_8 which configures I/Os used for the M4-F Serial Wire Debug.

Table 19: M4-F Serial Wire Debug Port Bootstrap Configuration

Serial Wire Debug Port Signal	IO_8 Pulled Down (Default)	IO_8 Pulled Up
SW_CLK	IO_14	IO_45
SW_IO	IO_15	IO_44

8.6.1. Internal/External HSO Configuration

The configuration of internal or external High-Speed Oscillator (HSO) is configured by bootstrap pins IO_8 and IO_9. When selecting the External HSO, the external clock is provided on IO_6.

Table 20: Internal/External HSO Configuration

IO_8	IO_9	Clock Source	HSO Configuration
Pulled Down (default)	Pulled Down (default)	Crystal or 32 kHz CMOS Clock at XTAL_IN	Internal HSO
Pulled Down (default)	Pulled Up	Crystal or 32 kHz CMOS Clock at XTAL_IN	Internal HSO
Pulled Up	Pulled Down (default)	32 kHz CMOS Clock at XTAL_IN	Internal HSO
Pulled Up	Pulled Up	IO_6	External HSO

8.6.2. SWD Debugger Present Configuration

The state of bootstrap pin IO_19 configures whether the SWD debugger is present.

NOTE: This setting only applies in AP configuration. In the Wearable configuration, bootstrap pin IO_19 must be pulled down to allow operation of the SPI flash boot.

Table 21: SWD Debugger Present Configuration

IO_19	Debugger State
Pulled Down	Debugger is not available until after the M4-F CPU core reset is released.
Pulled Up	Debugger access is allowed, as M4-F CPU core reset is released immediately.

8.6.3. AP/Wearable Mode Configuration

The state of bootstrap pin IO_20 determines if the device is in AP mode (SPI Slave) or Wearable mode (SPI Master).

Table 22: AP/Wearable Mode Configuration

IO_20	Mode	SPI Function	SPI I/0s Used
Pulled Down	Wearable	Master	IO_34 - SCLK IO_36 - MISO IO_38 - MOSI IO_39-SS1
Pulled Up	AP	Slave	IO_16 - SCLK IO_17 - MISO IO_19 - MOSI IO_20 - CS

If in wearable mode, the M4-F CPU core reset is de-asserted automatically once the boot code is downloaded by Configuration DMA.

If in AP mode, the application processor controls the release of the M4-F CPU core reset through register settings.

9. Other Peripherals

9.1. Packet FIFO

The packet FIFO bank provides data buffering for data transfers between FFE and/or on-chip programmable logic and/or M4-F to AP and/or M4-F. It is composed of four packet FIFOs of differing sizes. A typical use case may have the FFE push sensor data as packets into the Packet FIFO. When a specific threshold is reached, the programmable interrupt signals to the M4-F or AP to pop off the data for additional processing. The M4-F can also write data into the Packet FIFO and pop off the data. The on-chip programmable logic can push data packets into the Packet FIFOs, depending on the on-chip programmable logic configuration.

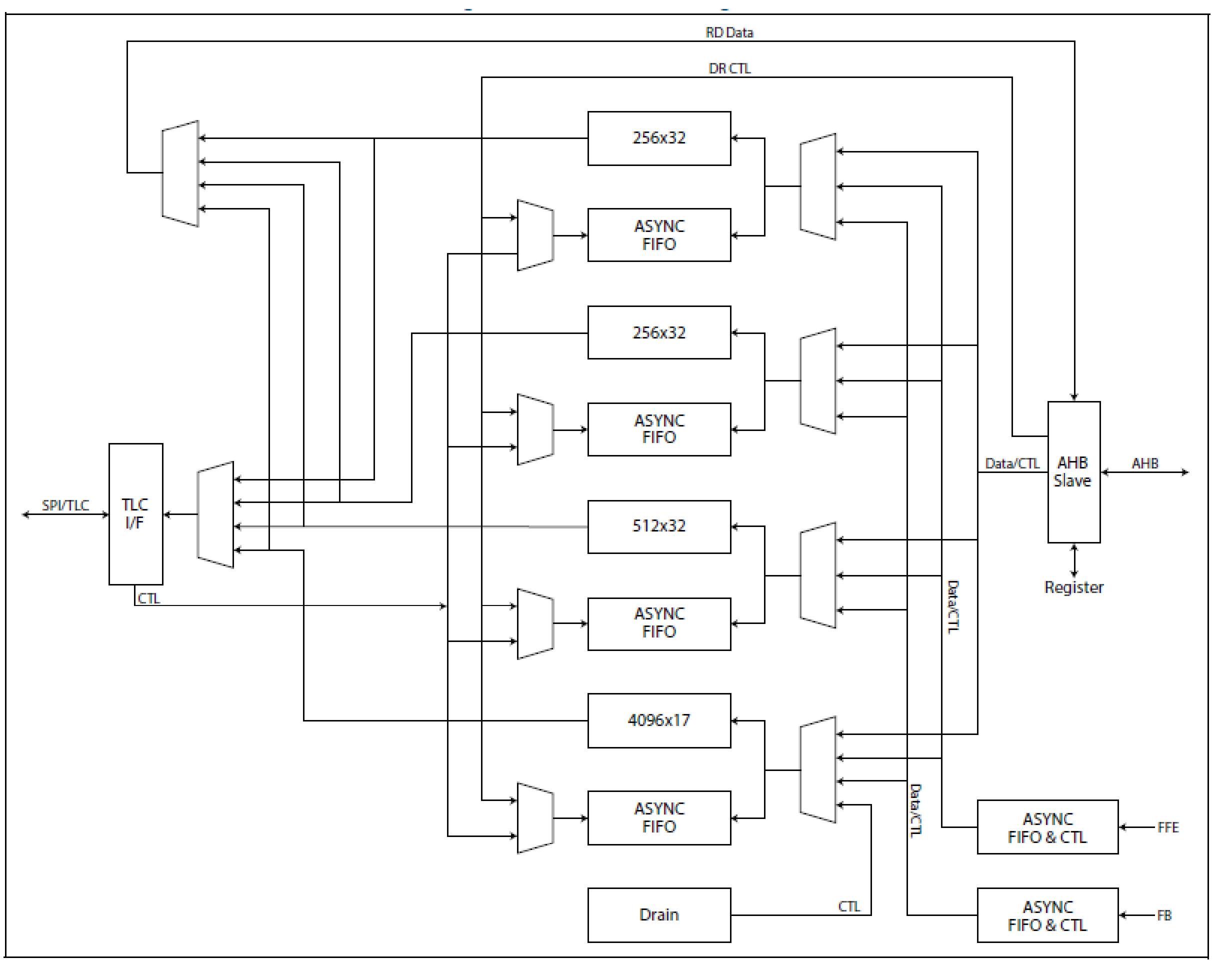
NOTE: The FFE and on-chip programmable logic cannot pop data from Packet FIFOs to perform additional processing.

Packet FIFOs are designed to support two operating modes:

- Packet mode – This FIFO mode differs from a normal FIFO in its generation of the pop side flags (empty and pop word count). It only considers data pushed up to end-of-packet (EOP) boundaries.
- Normal FIFO mode – This mode behaves like a normal FIFO with pop side flags updated for every instance of pushed data.

Figure 48 is a block diagram for Packet FIFO Bank (PKFB).

Figure 48: PKFB Block Diagram



The FIFO instances are listed in the **Table 23**.

Table 23: Packet FIFO Instances

Instance	Depth	Width	Description
FIFO_8K	4096	17	Supports normal or packet FIFO modes. Supports ring buffer mode support (16-bit data, 1-bit SOP). Drainer logic with programmable threshold to implement ring buffer function. PUSH source: FFEs or M4-F. POP destination: AP or M4-F.
FIFO_0	256	32	Supports normal FIFO modes. PUSH source: FFEs or M4-F. POP destination: AP or M4-F.
FIFO_1	128	32	
FIFO_2	128	32	

9.1.1. FIFO_8K

The FIFO_8K is a 4096x17 packet FIFO that has the following configurable options:

- Packet FIFO mode – When enabled, the FIFO behaves as a packet FIFO, otherwise, it behaves like a generic FIFO.
- Ring buffer mode – When enabled, a small drainer block on the pop side of the FIFO will be enabled. This drainer logic is triggered once the pop word count reaches a programmable threshold, which causes it to pop a packet off the top of the FIFO. Once the final pop agent (AP or M4-F) is triggered by an external event to start popping the FIFO, it will disable the drainer logic to start reading the FIFO. Care must be taken when disabling the drainer logic to avoid any type of race condition that can cause coherency issues. One possible way of doing this is for the AP/M4-F to disable the drainer logic before it polls the drainer logic BUSY status. The drainer is designed only to check the enable bit at the start of the pop transactions. Controls for the muxes should consider this to prevent going off-sync (such as changing mux control before the logic drainer is done).

  NOTE: The ring buffer mode can only be used with packet FIFO mode. The software must ensure that it does not enable the ring buffer mode for non-packet FIFO operation.

- Threshold – This register determines the FIFO threshold that triggers either the drainer logic (when used in ring mode) or an interrupt (when used as a normal FIFO). Both threshold triggers are designed to avoid a FIFO overrun condition.

The packet FIFO provides FIFO word count on the pop side. When used in packet FIFO mode, this indicates the exact number of words in the FIFO that represent full packets. When used as a normal FIFO, the FIFO word count specifies the exact number of words in the FIFO regardless of packet boundaries. The AP or M4-F is expected to read the FIFO word count and pop no more data than allowed by the count. In this way, the empty flag signal is never used to throttle the pop of data.

In ring buffer mode, the start of packet (SOP) signal is pushed into the FIFO as data bit[16] to alert the drainer logic and identify the SOP on the pop side so it is able to pop packets when the threshold is reached.

9.1.2. FIFO_0, FIFO_1, FIFO_2

All three FIFOs are designed to be generic FIFOs with the following firmware configurable option:

- Threshold – This register determines the FIFO threshold that triggers an interrupt when reached. The purpose of the threshold is to avoid a FIFO overrun condition.
- Sizes – FIFO_0 is 256x32 bits, while FIFO_1 and FIFO_2 are both 128x32 bits.

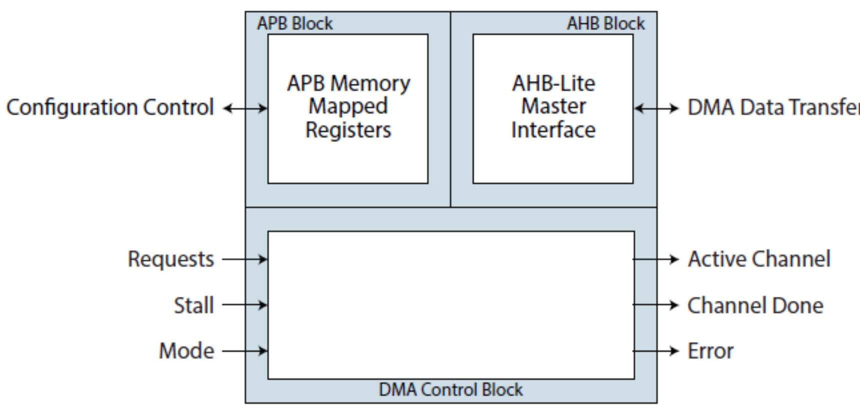
As with FIFO_8K, these FIFOs provide the FIFO word count on the pop side. FIFO word counts specify the exact number of words in the FIFO regardless of packet boundaries. The AP or M4-F is expected to read the FIFO word count, and pop no more than the count allows. In this way, the empty flag signal is never used to throttle the pop of data.

9.2. System DMA

The principal features of the SDMA include the following (and are illustrated in **Table 49**):

- Uses AHB-Lite for the DMA transfers
- Uses APB for programming the registers
- Single AHB-Lite Master for transferring data using a 32-bit address bus and 32-bit data bus
- Supports up to 16 DMA channels
- Dedicated handshake signals on each DMA channel
- Programmable priority level on each DMA channel
- Each priority level arbitrates using a fixed priority that is determined by the DMA channel number
- Supports multiple transfer types:
  - Memory-to-Memory
  - Memory-to-Peripheral
  - Peripheral-to-Memory
- Supports multiple DMA cycle types
- Supports multiple DMA transfer data widths
- Each DMA channel can access a primary and an alternate channel control data structure
- All channel control data is stored in system memory using the little-endian format
- Performs all DMA transfers using the SINGLE AHB-Lite burst type

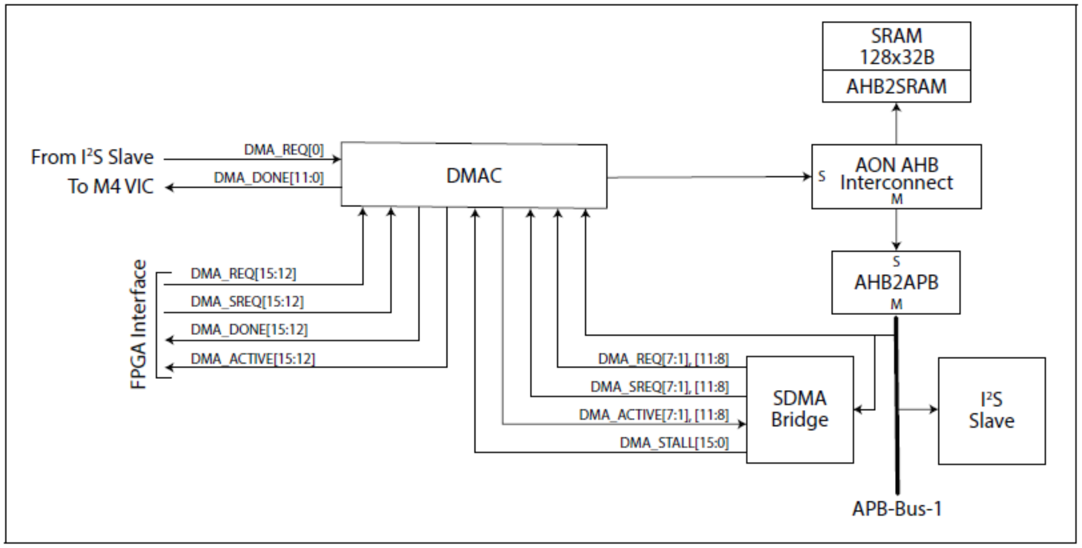
Figure 49: System DMA Block Diagram



9.2.1. Functional Description

The SDMA aids in offloading data move tasks from the M4-F CPU and allows M4-F to sleep as long as possible to save power. The SDMA can support a hardware/software request, and DMA requests can be from a peripheral or initiated by software. The SDMA uses an AHB-Lite Master port for reading DMA descriptors from 128x32 bits SRAM and transferring data from source to destination and the SDMA AHB Master port is connected to the Always-On AHB bus matrix so that it can transfer data even when M4-F power domain is in deep sleep or shut down. The APB bus is the register interface of the SDMA and SDMA Bridge.

Figure 50: System DMA Interface



The blocks that can initiate transfer using SDMA are:

- I²S Slave port
- M4-F processor
- FFE
- On-chip programmable logic

System DMA supports up to 16 DMA channels. Channel assignment is shown in **Table 24**. Each channel has a primary and an alternate descriptor associated with it, and each description consists of four words. The descriptors are hosted in a 128x32 bits SRAM connected to the Always-On AHB bus matrix. The SDMA bridge can generate single/burst DMA requests to the SDMA, which is software controlled.

Table 24: SDMA Channel Assignment

Channel Number	Primary Channel Programming	Alternate Channel Programming	Channel Trigger	Channel Acknowledge	Comment
0	M4-F	FFE	I²S	M4-F	Masking under software control possible.
1-7	M4-F	FFE	SDMA Bridge	SDMA Bridge and M4-F	These channels are under direct control of M4-F.
8-11	FFE	M4-F	SDMA Bridge	SDMA Bridge and M4-F	These channels can be controlled by FFE or M4-F. DMA_DONE signals associated with these channels are connected to M4-F VIC.
12-15	M4-F	FFE	FPGA	FPGA	

The System DMA can perform the following tasks:

- Transfer voice data from M4-F SRAM to I²S Slave.
- Transfer data from M4-F SRAM to FFE CM (swapping).
- Transfer data from M4-F SRAM to FFE DM.
- Transfer data from FFE DM to M4-F SRAM.
- Transfer data from M4-F SRAM to on-chip programmable logic.

- Transfer data from on-chip programmable logic to M4-F SRAM.

9.2.2. SDMA Configurations

The configuration for SDMA transfers includes the following three elements:

- DMA descriptors – Each DMA channel has two associated channel descriptor structures that are normally located in SDMA SRAM. These include the source and destination address for the channel as well as information on number of elements to transfer, data size, transfer type, etc.

  NOTE: When a channel is triggered, the DMA reads the associated descriptor from SRAM, which includes the instructions on what actions the DMA should take. When the channel has finished completing the defined actions, the updated descriptors are written back to the SRAM.

- DMA registers – Common configurations for the DMA, as well as DMA-generated interrupts and trigger sources for the various channels are configured in the DMA registers. The location of the DMA descriptors in SRAM is also configured here.
- Registers in trigger peripheral – The DMA request signals from the peripherals get generated by various events in the peripherals; hence, it is important to configure the peripherals correctly to generate the desired DMA requests.

9.3. Analog-to-Digital Converter

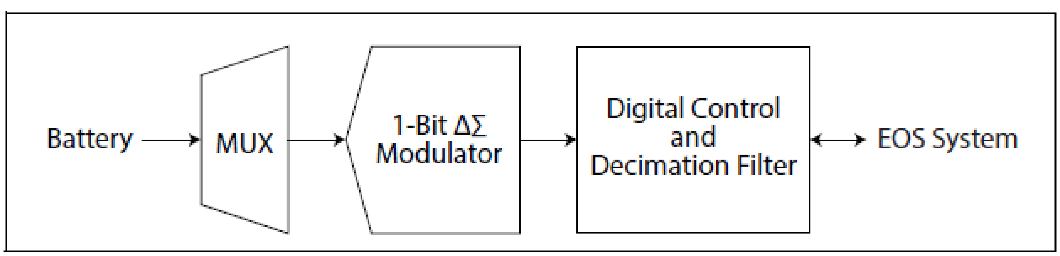
9.3.1. Overview

The ADC is an accurate, high resolution analog-to-digital converter used for voltage monitoring. To achieve excellent repeatability and high Power Supply Rejection Ratio (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 51 shows a general block diagram of the ADC.

Figure 51: ADC Block Diagram



9.3.2. Functional Description

In brief, 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.

9.3.3. Electrical Characteristics

Table 25 shows key electrical characteristics for the ADC module.

A_{GND} = 0V DC, F_{CLK} = 500 kHz, T_J = -25 C to +125 C, unless otherwise specified.

Table 25: ADC Electrical Characteristics

Symbol	Description	Min.	Typ.	Max.	Units
V_{AVDD}	Analog Supply Voltage	1.62	1.80	3.63	V
V_{DAC}	Input Voltage Conversion Range	0	-	1.4	V
F_{CLK}	Delta-Sigma Clock Frequency	200	1000	2000	kHz
Temp	Temperature Range	-20	-	85	°C
I_{AVDD}	Analog Current	0.1	0.2	0.3	mA
IQ_{PD}	Total Power Down Current Consumption	-	-	1	µA
ADC	ADC Resolution Voltage Per Step	-	0.34	-	µV
T_{CONV}	ADC Conversion Time[a]	2.5	12.5	25	ms

a. ADC conversion time is ~5,000 F_{CLK} cycles.

9.3.4. PCB Layout Recommendations

When using ADC, the following PCB layout recommendations include:

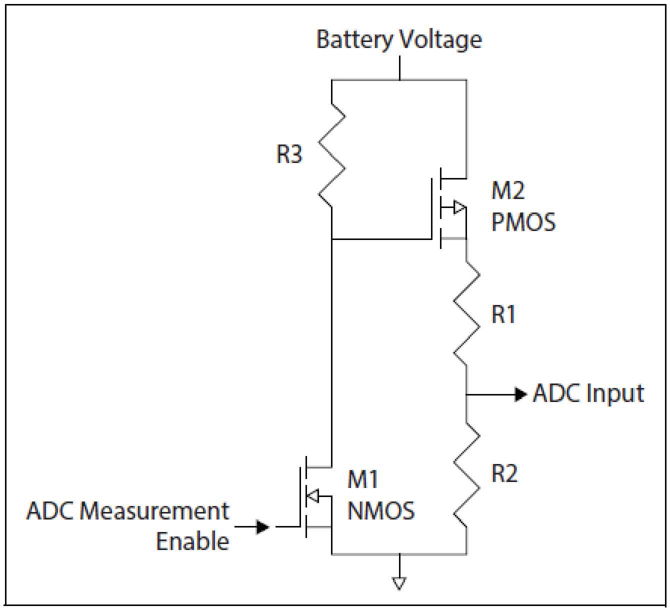
- Minimize the distances between the ADC input and the location where the voltage is being measured. Avoidrouting where it is close to noise sources such as clock generators, DC/DC converters and, data/address buses.
- Minimize inductance and reduce series resistance by using wide tracks. Use grounded guard traces when possible. A trace with a 10-mil minimum width and spacing is recommended.

9.3.5. Example Application

The ADC input voltage conversion range is 0 V to 1.4 V. Most rechargeable battery outputs are higher than the ADC input voltage. To support higher input voltage range, a maximum of up to 4 VDC, use an external analog level shifter to scale the battery voltage to ensure it is compatible with the ADC voltage range.

Figure 52 shows an example.

Figure 52: 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 to best determine the proper 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 enables control components M1, M2, and R3, which allow the ADC to disable the voltage divider between ADC measurements. Not doing so results in a constant current draw on the battery. Any constant current draw causes reduced battery life. Besides extending battery life, an additional benefit is that these components disable the voltage divider when the EOS S3 device is in a low power state.

9.4. Universal Asynchronous Receiver Transmitter (UART)

The UART provides a serial data connection that can be used for communications and trace. The main features of UART include:

- Programmable use as UART or IrDA SIR input/output
- Separate 32 8 transmit and 32 12 receive FIFO memory buffers to reduce CPU interrupts
- Programmable FIFO disabling for 1-byte depth
- Programmable Baud rate generator
- Standard asynchronous communication bits (start, stop and parity)
- Independent masking of transmit FIFO, receive FIFO, receive timeout, modem status, and error condition interrupts
- Support for Direct Memory Access (DMA)
- False start bit detection
- Line break generation and detection
- Support of the modem control functions CTS and RTS
- Programmable hardware flow control
- Fully-programmable serial interface characteristics:
  - Data can be 5 bits, 6 bits, 7 bits, or 8 bits
  - Even, Odd, Stick, or No Parity bit generation and detection
  - 1 or 2 stop bit generation
  - Baud rate generation, direction control up to UARTCLK/16

- IrDA SIR ENDEC block which supports:
  - Programmable use of IrDA SIR or UART input/output
  - IrDA SIR ENDEC functions for data rates up to 115,200 bps half-duplex
  - Normal 3/16 and low-power (1.41-2.23 øs) bit durations
  - Programmable division of the reference clock to generate the appropriate bit duration for low-power IrDA modes

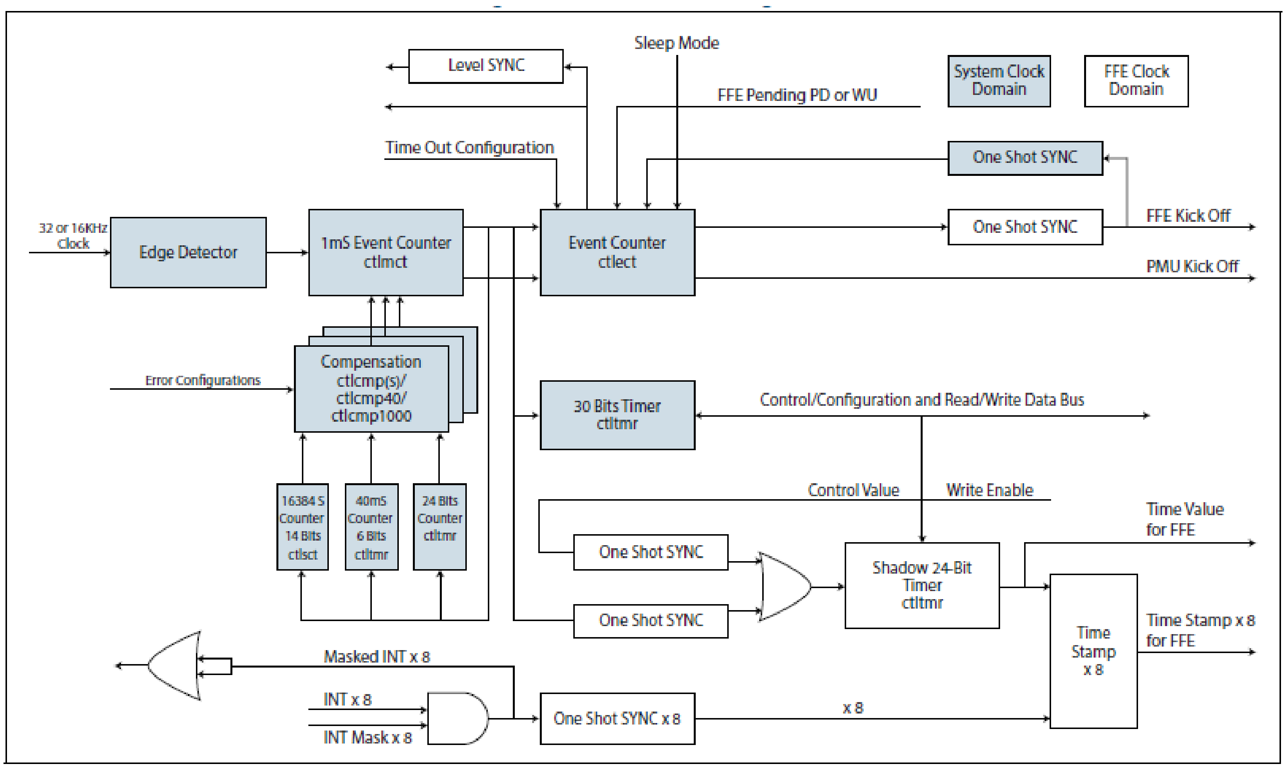
9.5. Timer and Counters

The EOS S3 platform supports several counters that count the clock event to generate the 1 ms event as well as time out events for waking up the FFE and FFE power domain. Counters also provides 24-bit timers and eight time stamps for FFE, and 30-bit timer for software use.

Clock events are generated on both edges of the reference clock. The 1 ms time out event period can be adjusted by configuring the trim bits.

- Clock Event Generator: Generate the Clock Event base on the edge of the reference clock. The reference clock can be either 16 kHz or 32 kHz with certain PPM error.
- 1 ms event counter: The resolution is 1 Clock Event.
- 30-bit/24-bit timer: The resolution is 1 ms (for details, see **Figure 53**; the 30-bit timer associated with the 1 ms event counter, the shadow 24-bit timer associated with the FFE).
- Time stamp: Eight total, timers LSB (lower 16 bits of timers) is latched once the corresponding Interrupt triggers.
- 5 ms time-out event with less than 1% error in a two-hour period.

Figure 53: Timer Block Diagram



9.5.1. 1 ms Event Counter

The 32 kHz Reference Clock 1 ms event alternates between 65 and 66 counts of clock event in the following sequence: 65 66 65 66 65 66 65 …

The 16 kHz Reference Clock 1 ms event alternates between 32 and 33 counts of clock event in the following sequence: 32 33 33 33 32 33 33 33 32 33 33 …

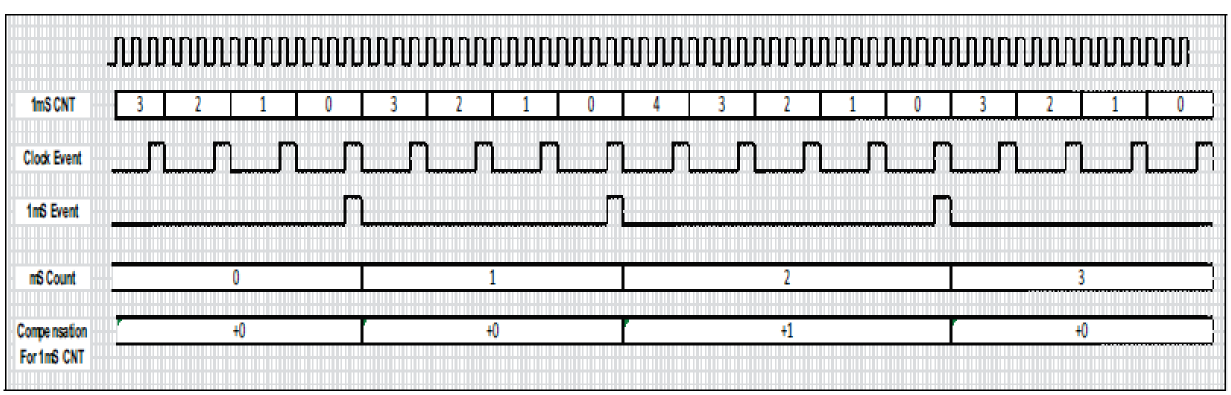
9.5.2. Error Correction for 1 mS Event Counter

The 1 ms event counter implementation accumulates approximately -550 µs error for every 1 second period without error compensation occurring. As a result, an Error Correction circuit is implemented to compensate for potential errors in the corresponding reference clock.

The following list represents multiple layer error correction compensation schemes in use:

- 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 54: 1 ms Count and 1 ms Counter Relationship



9.5.3. Timeout Event Counter

The timeout event counter counts the 1ms events and the time out period (from 1 ms to 255 ms) based on the configured value.

9.5.4. 30-Bit Counter

The 30-bit counter timer counts the 1ms event (in 1 ms resolution) and allows the software to read/write the timer value through Registers space.

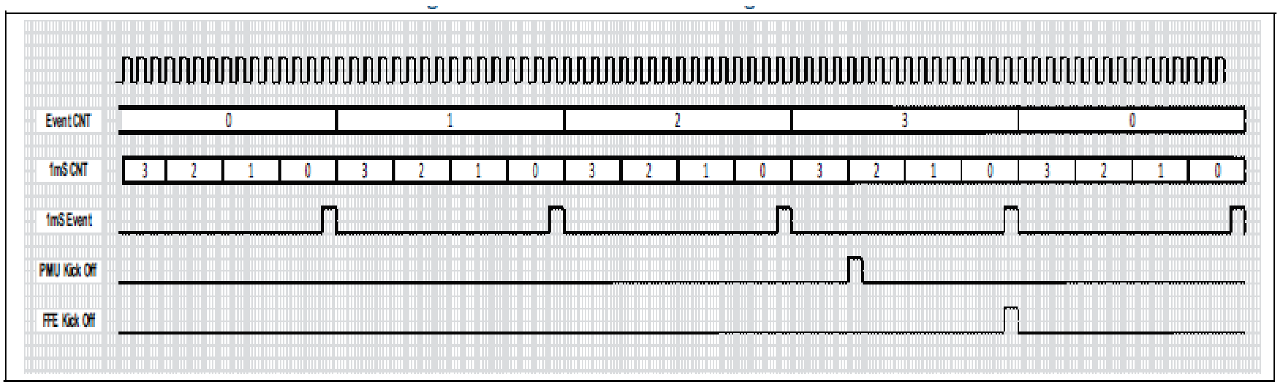
9.5.5. Time Stamp Counters

There are eight time stamps. Each time stamp has 16-bits and the corresponding interrupt source that could be individual mask out. If the interrupt triggers, the lower 16 bits of the 24 bits of the timer loads with the corresponding time stamp.

9.5.6. PMU and FFE Wakeup

This timer allows the PMU to power up the sensor processing subsystem FFE. Once the FFE power domain is ON, a kick-off event is sent to the FFE. There are register controls that control the timing relationship between the time that FFE is powered up and the kick-off event is sent to the FFE, which provides for some timing flexibility for the software.

Figure 55: PMU and FFE Timing Waveform



10. Device Characteristics

10.1. Pinout and Pin Description

Table 26 lists the input and output (I/O) locations and functions for each of the IO pins in EOS S3. It is important to note two things:

- There is at least one default function assigned to each IO (and its default = 0)
- The EOS S3 software does not configure the IO if it is being used as a default

NOTE: Table 26 lists the default functions in **bold** for all EOS S3 IO pins in the Alternate Functions column. This convention visually indicates the default function for each IO. There are also cases indicated where end-users may choose to bootstrap IO pins to suit specific product-based requirements and these are noted (for example, see IO_14).

Table 26: EOS S3 Voice and Sensor Processing Platform Pinout

Signal Name	Signal Type	I/O Bank	WCLSP Ball	BGA Ball	Description	Alternate Function
ADC0	ANALOG	ANALOG	B2	A7	ADC 0 Input	
ADC1	ANALOG	ANALOG	--	C7	ADC 1 Input	
AGND	ANA		C3	A8	Analog Ground	
AVDD	ANA		C2	D8	Analog Power	
FSOURCE	VPP		E5	G3	Connect to ground	
GND	GND		C4, C5	D3, D4, D5	Ground	
LDO_VIN	POWER		B3	A6	Internal LDO Power Input	
XTAL_IN	ANALOG	ANALOG	A1	B8	32 kHz Crystal	32 kHz Clock Input
XTAL_OUT	ANALOG	ANALOG	B1	C8	32 kHz Crystal	
IO_0	IO	VCCIOA	A7	B1		FBIO_0[a], **SCL_0**
IO_1	IO	VCCIOA	B7	C1		FBIO_1, **SDA_0**
IO_2	IO	VCCIOA	--	A1		**FBIO_2**, SPI_SENSOR_SSn2, DEBUG_MON_0, BATT_MON, SENSOR_INT_1
IO_3	IO	VCCIOA	C7	A2		FBIO_3, SENSOR_INT_0
IO_4	IO	VCCIOA	--	B2		**FBIO_4**, SPI_SENSOR_SSn3, DEBUG_MON_1, SDA_1_DPU, SENSOR_INT_2
IO_5	IO	VCCIOA	--	C3		**FBIO_5**, SPI_SENSOR_SSn4, DEBUG_MON_2, **SDA_0_DPU**, SENSOR_INT_3
IO_6	IO	VCCIOA	A6	B3		**FBIO_6**, SPI_SENSOR_MOSI, DEBUG_MON_3, **FCLK**, GPIO(0)[b], IrDA_SIRIN, SENSOR_INT_1

IO_7	IO	VCCIOA	--	A3		**FBIO_7**, SPI_SENSOR_SSn5, DEBUG_MON_4, SWV, SENSOR_INT_4
IO_8	IO	VCCIOA	B6	C4		**FBIO_8**, PDM_CLK_O, I2S_CLK_O, IrDA_SIROUT, SENSOR_INT_2
IO_9	IO	VCCIOA	A5	B4		**FBIO_9**, SPI_SENSOR_SSn1, I2S_WD_CLK_O, GPIO(1),PDM_STAT_I, SENSOR_INT_3
IO_10	IO	VCCIOA	B5	A4		**FBIO_10**, SPI_SENSOR_CLK, SWV, SENSOR_INT_4, I2S_DIN, PDM_DIN
IO_11	IO	VCCIOA	--	C5		**FBIO_11**, SPI_SENSOR_SSn6, DEBUG_MON_5, GPIO(2), SENSOR_INT_5
IO_12	IO	VCCIOA	--	B5		**FBIO_12**, SPI_SENSOR_SSn7, DEBUG_MON_6, IrDA_SIROUT, SENSOR_INT_6
IO_13	IO	VCCIOA	--	D6		**FBIO_13**, SPI_SENSOR_SSn8, DEBUG_MON_7, SWV, SENSOR_INT_7
IO_14	IO	VCCIOA	A4	A5		**FBIO_14**, **SW_DP_CLK**, IrDA_SIROUT, SCL_1, GPIO(3), UART_RXD, SENSOR_INT_5
IO_15	IO	VCCIOA	B4	C6		FBIO_15, **SW_DP_IO**, IrDA_SIRIN, SDA_1, UART_TXD, SENSOR_INT_6
IO_16	IO	VCCIOB	E1	E7		FBIO_16, SPI_SLAVE_CLK, UART_RXD
IO_17	IO	VCCIOB	D1	D7		FBIO_17, **SPI_SLAVE_MISO**, UART_CTS
IO_18	IO	VCCIOB	--	E8		**FBIO_18**, SWV, DEBUG_MON_0, GPIO(4), SENSOR_INT_1
IO_19	IO	VCCIOB	C1	H8		FBIO_19, SPI_SLAVE_MOSI, UART_RTS **Note:** IO_19 can serve as bootstrap for debugger mode as an M4-F reset release mechanism.
IO_20	IO	VCCIOB	F2	G8		FBIO_20, SPI_SLAVE_SSn, UART_TXD
IO_21	IO	VCCIOB	--	H7		**FBIO_21**, DEBUG_MON_1, IrDA_SIRIN, GPIO(5), UART_RTS, SENSOR_INT_2

IO_22	IO	VCCIOB	--	G7		**FBIO_22**, DEBUG_MON_2, IrDA_SIROUT, GPIO(6), UART_CTS, SENSOR_INT_3
IO_23	IO	VCCIOB	E2	H6		**FBIO_23**, SPI_MASTER_SSn2, SWV, GPIO(7), AP_I2S_WD_CLK_IN, SENSOR_INT_7
IO_24	IO	VCCIOB	D2	G6		**FBIO_24**, AP_I2S_DOUT, IrDA_SIRIN, GPIO(0)[b], UART_TXD, SENSOR_INT_1
IO_25	IO	VCCIOB	D3	F7		**FBIO_25**, SPI_MASTER_SSn3, SWV, IrDA_SIROUT, UART_RXD, SENSOR_INT_2
IO_26	IO	VCCIOB	--	F6		**FBIO_26**, SPI_SENSOR_SSn3, DEBUG_MON_3, GPIO(1), SENSOR_INT_4
IO_27	IO	VCCIOB	--	H5		**FBIO_27**, SPI_MASTER_SSn2, SPI_SENSOR_SSn4, DEBUG_MON_4, SENSOR_INT_5
IO_28	IO	VCCIOB	F3	G5		**FBIO_28**, SPI_SENSOR_MOSI, DEBUG_MON_5, GPIO(2), I2S_DIN, PDM_DIN, IrDA_SIRIN, SENSOR_INT_3
IO_29	IO	VCCIOB	E3	F5		**FBIO_29**, SPI_SENSOR_MISO, I2S_CLK_O, PDM_CLK_O, IrDA_SIROUT, SENSOR_INT_4
IO_30	IO	VCCIOB	F4	F4		**FBIO_30**, SPI_SENSOR_SSn1, GPIO(3), I2S_WD_CLK_O, PDM_STAT_I, SENSOR_INT_5
IO_31	IO	VCCIOB	E4	G4		**FBIO_31**, SPI_SENSOR_CLK, GPIO(4), AP_I2S_CLK_IN, SENSOR_INT_6
IO_32	IO	VCCIOB	--	H4		**FBIO_32**, SPI_SENSOR_SSn5, DEBUG_MON_6, SDA_1, SENSOR_INT_6
IO_33	IO	VCCIOB	--	E3		**FBIO_33**, SPI_SENSOR_SSn6, DEBUG_MON_7, SCL_1, SENSOR_INT_7
IO_34	IO	VCCIOB	D5	F3		**SPI_MASTER_CLK**, FBIO_34, DEBUG_MON_0, AP_PDM_STAT_O, SENSOR_INT_7

IO_35	IO	VCCIOB	--	F2		**FBIO_35**, SPI_MASTER_SSn3, SPI_SENSOR_SSn7, DEBUG_MON_1, SENSOR_INT_1
IO_36	IO	VCCIOB	F5	H3		**SPI_MASTER_MISO**, **FBIO_36**, SWV, SPI_SENSOR_SSn2, GPIO(5), SENSOR_INT_1
IO_37	IO	VCCIOB	--	G2		**FBIO_37**, SPI_SENSOR_SSn8, DEBUG_MON_2, SDA_2_DPU, SENSOR_INT_2
IO_38	IO	VCCIOB	E6	E2		**SPI_MASTER_MOSI**, FBIO_38, DEBUG_MON_3, GPIO(6), AP_PDM_CLK_IN, SENSOR_INT_2
IO_39	IO	VCCIOB	F6	H2		FBIO_39, DEBUG_MON_4, AP_PDM_IO, SENSOR_INT_3
IO_40	IO	VCCIOB	--	D2		**SPI_MASTER_SSn1**, **FBIO_40**, SCL_2, DEBUG_MON_5, IrDA_SIRIN, SENSOR_INT_3
IO_41	IO	VCCIOB	--	F1		**FBIO_41**, SDA_2, DEBUG_MON_6, IrDA_SIROUT, SENSOR_INT_6
IO_42	IO	VCCIOB	--	H1		**FBIO_42**, SDA_1_DPU, DEBUG_MON_7, SWV, SENSOR_INT_7
IO_43	IO	VCCIOB	D7	D1		FBIO_43, AP_INTERRUPT
IO_44	IO	VCCIOB	E7	E1		FBIO_44, **SW_DP_IO**, SDA_1, UART_TXD, IrDA_SIRIN, SENSOR_INT_4
IO_45	IO	VCCIOB	F7	G1		**FBIO_45**, **SW_DP_CLK**, SCL_1, UART_RXD, IrDA_SIROUT, GPIO(7), SENSOR_INT_5
STM	INPUT	VCCIOB	D4	E5	Connect to ground	
SYS_RSTn	INPUT	VCCIOB	F1	F8	System Reset Input	
VCCIOA	POWER		C6	C2	Bank A VCC In	
VCCIOB	POWER		D6	E4,E6	Bank B VCC In	
VDD1	POWER		A2	B7	LDO 1 Output	
VDD2	POWER		A3	B6	LDO 2 Output	

a. Each of the 46 multi-function IOs can be used as on-chip programmable logic (FB) IOs. Each IO is assigned a corresponding FBIO function. Refer to subsection **Fabric Inputs/Outputs (FBIOs)** on page 63 for more details.

b. M4-F can only control total of 8 IO pads as GPIOs out of the 46 multi-function IOs. For example, M4-F can control GPIO bit 0 on either IO_6 or IO_24. M4-F can control GPIO bit 1 on either IO_9 or IO_26, etc. Only 1 IO can be selected for each GPIO. Both IOs cannot be selected at the same time. Look at the Alternate Function column to see which

IOs can be used as GPIOs. A complete
explanation and listing can be found in the subsection **General Purpose Inputs/Outputs (GPIOs)** on page 62.

10.2. I/O State

Table 27 shows the default state of the I/Os before and after SYS_RST_N release when all supply rails have reached 90% of level. The I/O states are driven as output if the VCCIO supply is powered up before the VDD core supply.

Table 27: I/O State

GPIO	VCCIO Bank	WLCSP Ball	BGA Ball	SYS_RST_N = 0	SYS_RST_N = 1
GPIO<0>	VCCIO<A>	A7	B1	PU	PU
GPIO<1>	VCCIO<A>	B7	C1	PU	PU
GPIO<2>	VCCIO<A>	NC	A1	Z	Z
GPIO<3>	VCCIO<A>	C7	A2	Z	Z
GPIO<4>	VCCIO<A>	NC	B2	Z	Z
GPIO<5>	VCCIO<A>	NC	C3	Z	Z
GPIO<6>	VCCIO<A>	A6	B3	Z	Z
GPIO<7>	VCCIO<A>	NC	A3	Z	Z
GPIO<8>	VCCIO<A>	B6	C4	PD	PD
GPIO<9>	VCCIO<A>	A5	B4	PD	PD
GPIO<10>	VCCIO<A>	B5	A4	Z	Z
GPIO<11>	VCCIO<A>	NC	C5	Z	Z
GPIO<12>	VCCIO<A>	NC	B5	Z	Z
GPIO<13>	VCCIO<A>	NC	D6	Z	Z
GPIO<14>	VCCIO<A>	A4	A5	PU	GPIO<19>=1 & GPIO<8>=0; PU GPIO<19>=1 & GPIO<8>=1; Z
GPIO<15>	VCCIO<A>	B4	C6	PU	GPIO<19>=1 & GPIO<8>=0; PU GPIO<19>=1 & GPIO<8>=1; Z
GPIO<16>	VCCIO<B>	E1	E7	Z	Z
GPIO<17>	VCCIO<B>	D1	D7	GPIO<20> = 0; 0 GPIO<20> = 1; Z	Z
GPIO<18>	VCCIO<B>	NC	E8	Z	Z
GPIO<19>	VCCIO<B>	C1	H8	PD	PD
GPIO<20>	VCCIO<B>	F2	G8	PD	PD
GPIO<21>	VCCIO<B>	NC	H7	Z	Z
GPIO<22>	VCCIO<B>	NC	G7	Z	Z
GPIO<23>	VCCIO<B>	E2	H6	Z	Z
GPIO<24>	VCCIO<B>	D2	G6	Z	Z
GPIO<25>	VCCIO<B>	D3	F7	Z	Z

GPIO<26>	VCCIO<B>	NC	F6	Z	Z
GPIO<27>	VCCIO<B>	NC	H5	Z	Z
GPIO<28>	VCCIO<B>	F3	G5	Z	Z
GPIO<29>	VCCIO<B>	E3	F5	Z	Z
GPIO<30>	VCCIO<B>	F4	F4	Z	Z
GPIO<31>	VCCIO<B>	E4	G4	Z	Z
GPIO<32>	VCCIO<B>	NC	H4	Z	Z
GPIO<33>	VCCIO<B>	NC	E3	Z	Z
GPIO<34>	VCCIO<B>	D5	F3	Z	Z
GPIO<35>	VCCIO<B>	NC	F2	Z	Z
GPIO<36>	VCCIO<B>	F5	H3	Z	Z
GPIO<37>	VCCIO<B>	NC	G2	Z	Z
GPIO<38>	VCCIO<B>	E6	E2	Z	Z
GPIO<39>	VCCIO<B>	F6	H2	Z	Z
GPIO<40>	VCCIO<B>	NC	D2	Z	Z
GPIO<41>	VCCIO<B>	NC	F1	Z	Z
GPIO<42>	VCCIO<B>	NC	H1	Z	Z
GPIO<43>	VCCIO<B>	D7	D1	Z	Z
GPIO<44>	VCCIO<B>	E7	E1	Z	GPIO<8>=1; GPIO<8>=0
GPIO<45>	VCCIO<B>	F7	G1	Z	GPIO<8>=1; GPIO<8>=0

11. Electrical Specifications

11.1. DC Characteristics

The DC specifications are provided in **Table 28** through **Table 31**.

Table 28: Absolute Maximum Ratings

Parameter	Value	Parameter	Value
LDO Input Voltage	-0.5 V to 3.6 V	ESD Pad Protection	2 kV
VDD Voltage	-0.5 V to 1.26 V	Laminate Package (BGA) Storage Temperature	-55°C to + 125°C
AVDD/VDDIO Voltage	-0.5 V to 3.6 V		
Input Voltage	-0.5 V to 3.6 V	Latch-up Immunity	±100 mA

WARNING: The absolute maximum ratings may cause permanent damage to the EOS S3 platform. Functional operation of the device should follow the recommended operating range in **Table 29**.

Table 29: Recommended Operating Range

Symbol	Parameter[a,b,c,d,e]	Min.	Typ.	Max.	Unit
LDO1_VIN	LDO1 input voltage	1.62		3.6	V
	LDO1 analog current consumption –				
	maximum output current set to:				
LDO1_I	50 mA		30		µA
	8 mA		10		
	1 mA		6		
LDO2_VIN	LDO2 input voltage	1.62		3.6	V
	LDO2 analog current consumption –				
	maximum output current set to:				
LDO2_I	30 mA		20		µA
	8 mA		10		
	1 mA		6		
VDD1 Memory (LDO1_OUT)	Supply voltage during active mode	0.95	1.1	1.21	V
	Supply voltage during initialization	1.05	1.1	1.21	V
VDD2 Logic (LDO2_OUT)	Supply voltage during active mode	0.95	1.1	1.21	V
	Supply voltage during initialization	1.05	1.1	1.21	V
VCCIO	Input tolerance voltage	1.71	-	3.6	V
T_J	Ambient temperature	-20	25	85	°C
AVDD	Analog voltage	1.71	-	3.6	V
XTAL_IN	Crystal input	-	32.768	-	kHz
XTAL_IN Low Level	CMOS input low level	-0.3	-	0.35	V

XTAL_IN High Level	CMOS input high level[f]	0.80	-	3.4[g]	V
HSOSC	High speed oscillator frequency	2	20	80	MHz
CMOS Clock Duty Cycle	CMOS clock duty cycle	40	50	60	Percent
CMOS Clock Input Jitter	CMOS clock input jitter	-	-	280	ns

a. Refer to **Low Dropout Regulators** on page 50 for an explanation of the different LDO configurations.
b. Except where indicated, Min and Max values are tested on 100% of the device at 25°C.
c. Typical values are based on 25°C and nominal voltage (VDD1=VDD2=1.1V, VCCIO=1.8V).
d. Device bootup and initialization should be at 1.1V, and minimum of 1.05V to come out of Power-On reset in LDO Bypass mode.
e. LDO1_VIN and LDO2_VIN must be the same voltage.
f. Special analog pad with CMOS tolerant input.
g. The OSCin/out pads are connected to an AVDD supply. Additional current consumption is drawn through the pin when OSCin high level is higher than AVDD.

Table 30: Weak Pull-Up/Pull-Down Characteristics

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Weak Pull-Up Current	I_{PU}	VDDIO = 3.3V VDDIO = 2.5V VDDIO = 1.8V	37 19 16	64 35 32	1 59 58	µA
Weak Pull-Down Current	I_{PD}	VDDIO = 3.3V VDDIO = 2.5V VDDIO = 1.8V	29 14 15	59 31 31	105 59 56	µA
Input Leakage	I_{IH}/I_{IL}		-	-	<1	µA
Short Circuit Current	I_{OSH}	VDDIO = 3.3V VDDIO = 2.5V VDDIO = 1.8V	-	116 72 69	-	mA
Short Circuit Current	I_{OSL}	VDDIO = 3.3V VDDIO = 2.5V VDDIO = 1.8V	-	109 74 68	-	mA

Table 31: DC Input and Output Levels[a]

Symbol	V_{IL}		V_{IH}		V_{OL}	V_{OH}	I_{OL}	I_{OH}
	V_{MIN}	V_{MAX}	V_{MIN}	V_{MAX}	V_{MAX}	V_{MIN}	mA	mA
LVTTL	-0.3	0.8	2.2	VDDIO + 0.3	0.4	2.4	2.0	-2.0
LVCMOS25	-0.3	0.7	1.7	VDDIO + 0.3	0.4	1.8	2.0	-2.0
LVCMOS18	-0.3	0.63	1.17	VDDIO + 0.3	0.45	VCCIO - 0.45	2.0	-2.0

a. The data in this table represents JEDEC specifications. QuickLogic devices either meet or exceed these requirements. Based on weak pull-down I/O termination disabled.

11.2. Output Drive Current

NOTE: The multi-functional IOs have four programmable drive strength states D[1-0]: D00=2 mA, D01=4 mA, D10=8 mA, D11= 12 mA. The drive strength can be set by programming A0 registers.

Table 32: Output Drive Current (DVDD = 1.8V) in mA

Parameter	Condition	D[1]	D[0]	Min.	Typ.	Max.
I_{OH}	V_{OH} = DVDD – 0.4	0	0	3.42	5.83	9.16
		0	1	6.84	11.7	18.3
		1	0	9.12	15.5	24.4
		1	1	12.5	21.4	33.6
I_{OL}	V_{OL} = 0.4	0	0	4.56	7.86	12.4
		0	1	9.14	15.7	24.8
		1	0	10.1	17.3	27.3
		1	1	14.6	25.2	39.7

Table 33: Output Drive Current (DVDD = 2.5V) in mA

Parameter	Condition	D[1]	D[0]	Min.	Typ.	Max.
I_{OH}	V_{OH} = DVDD – 0.4	0	0	3.60	5.68	8.28
		0	1	5.40	8.53	12.4
		1	0	9.01	14.2	20.7
		1	1	10.8	17.1	24.9
I_{OL}	V_{OL} = 0.4	0	0	4.07	6.65	9.78
		0	1	6.79	11.1	16.3
		1	0	10.9	17.8	26.2
		1	1	13.6	23.3	32.7

Table 34: Output Drive Current (DVDD = 3.3V) in mA

Parameter	Condition	D[1]	D[0]	Min.	Typ.	Max.
I_{OH}	V_{OH} = DVDD – 0.4	0	0	4.94	7.25	9.90
		0	1	7.42	10.9	14.9
		1	0	12.4	18.2	24.8
		1	1	14.8	21.8	29.7
I_{OL}	V_{OL} = 0.4	0	0	5.08	7.91	11.0
		0	1	8.48	13.2	18.3
		1	0	13.6	21.1	29.3
		1	1	17.0	26.4	36.6

11.3. Clock and Oscillator Characteristics

Table 35: Clock and Oscillator Characteristics[a]

Symbol	Min.	Typ.	Max.	Unit
XTAL_IN	16	32.768	-	kHz
HOSC	2	20	80	MHz
SPI Master CLK	2	10	20	MHz
SPI Slave CLK	2	10	20	MHz
SWD CLK	2	5	10	MHz
Voice SS (APB Clock)	-	-	10	MHz
PDM Left Clock	-	-	5	MHz
PDM Right Clock	-	-	5	MHz
I^2S Clock	-	-	5	MHz
LPSD Clock	-	-	1	MHz
FPGA Clock	-	-	10	MHz

a. Maximum frequency is with VDD at 1.1V, ±10%.

11.4. Output Rise/Fall Time

NOTE: The multi-functional IOs also have programmable slew rates (SRs). The two states are SR = 0 (slow) or SR = 1 (fast) and can be programmed from A0 registers.

11.4.1. Output Rise/Fall Time (DVDD = 1.8V)

Table 36: Output Rise/Fall Time (SR = 1, DVDD = 1.8V)

Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.71	1.27	2.23	ns
	0	1	5pF	0.65	1.18	2.16	ns
	1	0	10pF	0.67	1.24	2.30	ns
	1	1	20pF	0.84	1.51	2.81	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.60	1.02	1.80	ns
	0	1	5pF	0.57	0.96	1.80	ns
	1	0	10pF	0.70	1.18	2.14	ns
	1	1	20pF	0.84	1.39	2.51	ns

Table 37: Output Rise/Fall Time (SR = 0, DVDD = 1.8V)

Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.74	1.30	2.28	ns
	0	1	5pF	0.70	1.24	2.26	ns
	1	0	10pF	0.77	1.36	2.50	ns

	1	1	20pF	0.92	1.63	2.97	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.65	1.13	1.97	ns
	0	1	5pF	0.71	1.20	2.13	ns
	1	0	10pF	0.84	1.41	2.47	ns
	1	1	20pF	1.00	1.66	2.90	ns

11.4.2. Output Rise/Fall Time (DVDD = 2.5V)

Table 38: Output Rise/Fall Time (SR = 1, DVDD =2.5V)

Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.73	1.19	1.96	ns
	0	1	5pF	0.78	1.27	2.15	ns
	1	0	10pF	0.85	1.40	2.42	ns
	1	1	20pF	1.11	1.83	3.14	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.65	1.03	1.71	ns
	0	1	5pF	0.68	1.02	1.79	ns
	1	0	10pF	0.77	1.40	2.03	ns
	0	0	20pF	0.73	1.19	1.96	ns

Table 39: Output Rise/Fall Time (SR = 0, DVDD =2.5V)

Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.80	1.31	2.14	ns
	0	1	5pF	0.90	1.45	2.47	ns
	1	0	10pF	1.13	1.82	3.12	ns
	1	1	20pF	1.43	2.29	3.84	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.72	1.16	1.88	ns
	0	1	5pF	0.82	1.29	2.11	ns
	1	0	10pF	1.08	1.69	2.78	ns
	1	1	20pF	1.36	2.08	3.40	ns

11.4.3. Output Rise/Fall Time (DVDD = 3.3V)

Table 40: Output Rise/Fall Time (SR = 1, DVDD =3.3V)

Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.63	0.86	1.42	ns
	0	1	5pF	0.57	0.93	1.57	ns
	1	0	10pF	0.64	1.04	1.76	ns
	1	1	20pF	0.88	1.36	2.25	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.59	0.78	1.32	ns
	0	1	5pF	0.52	0.81	1.37	ns
	1	0	10pF	0.61	0.92	1.53	ns
	1	1	20pF	0.82	1.19	1.94	ns

Table 41: Output Rise/Fall Time (SR = 0, DVDD =3.3V)

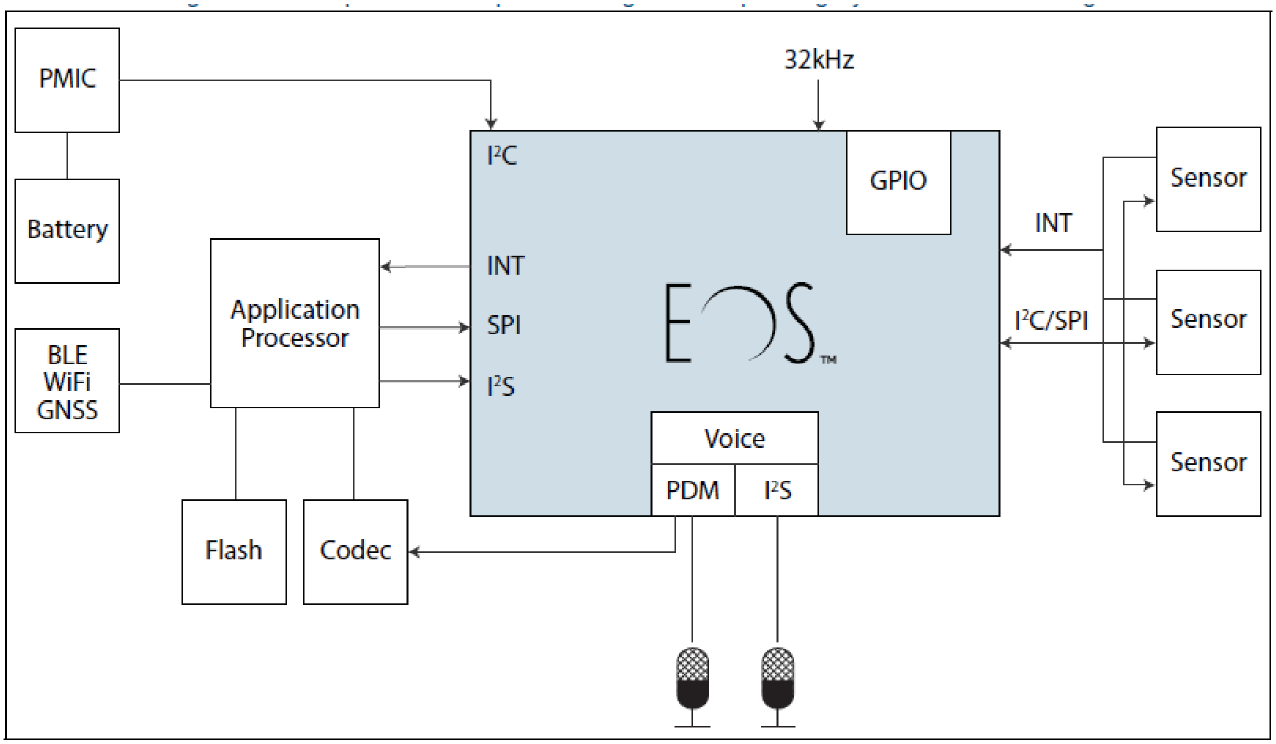
Transition	D[1]	D[0]	C_{LOAD}	Min.	Typ.	Max.	Units
Rise Time PAD↑ (10% to 90%)	0	0	2pF	0.66	0.97	1.59	ns
	0	1	5pF	0.72	1.08	1.81	ns
	1	0	10pF	0.90	1.37	2.23	ns
	1	1	20pF	1.14	1.74	2.83	ns
Fall Time, PAD↓ (90% to 10%)	0	0	2pF	0.62	0.91	1.48	ns
	0	1	5pF	0.68	0.96	1.60	ns
	1	0	10pF	0.88	1.52	2.05	ns
	1	1	20pF	1.10	1.61	2.53	ns

12. Application Examples

12.1. Smartphone or High-Level O/S Wearable Design

Figure 57 illustrates the EOS S3 platform as a discrete sensor hub, offloading the always-on, real-time processing from the Application Processor. The voice subsystem is enabled, it handles the always-on voice recognition. The hardware bypass path on the PDM interface enables voice recognition to be offloaded to the EOS S3 platform, and then normal voice communication to be handed off seamlessly to the dedicated voice CODEC.

Figure 56: Example of a Smartphone or High-Level Operating System Wearable Design

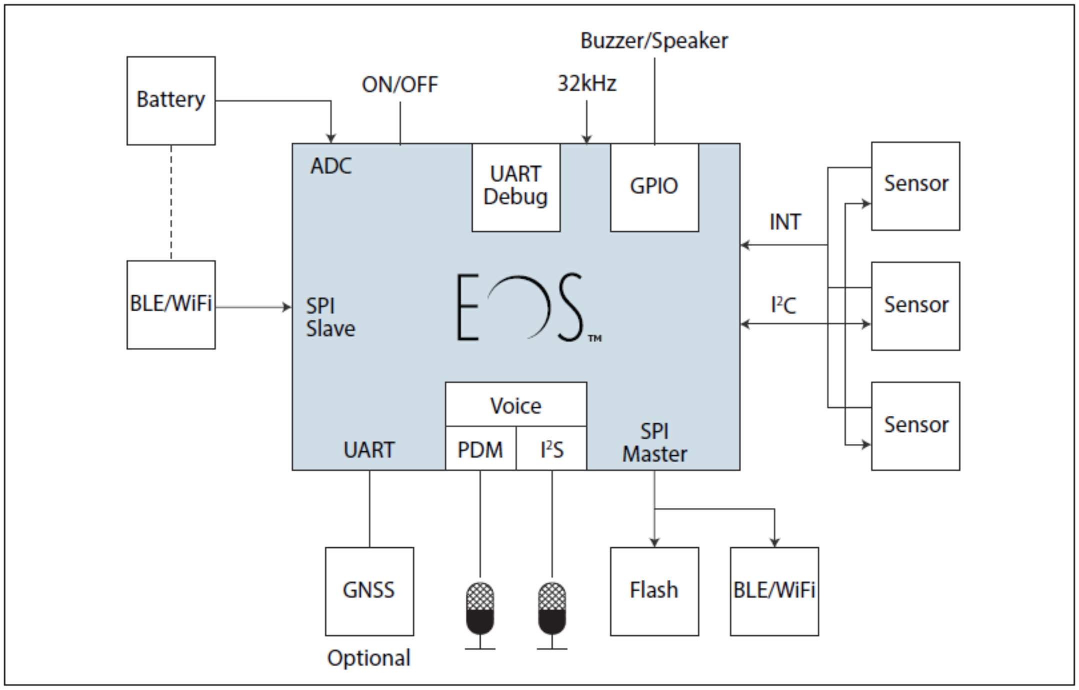


12.2. Real-Time Operating System Wearable Design

Figure 57 illustrates the EOS S3 platform as a true SoC in a RTOS-based Wearable or IoT device. In this use case, the EOS S3 platform acts as the host processor running the operating system, the always-on, real-time sensor processing, and the interface to the connectivity device(s) in the system.

When the voice subsystem is enabled, it handles the always-on voice recognition. The on-chip programmable logic can also be used to handle potential glue-logic or system-level power management within the wearable device.

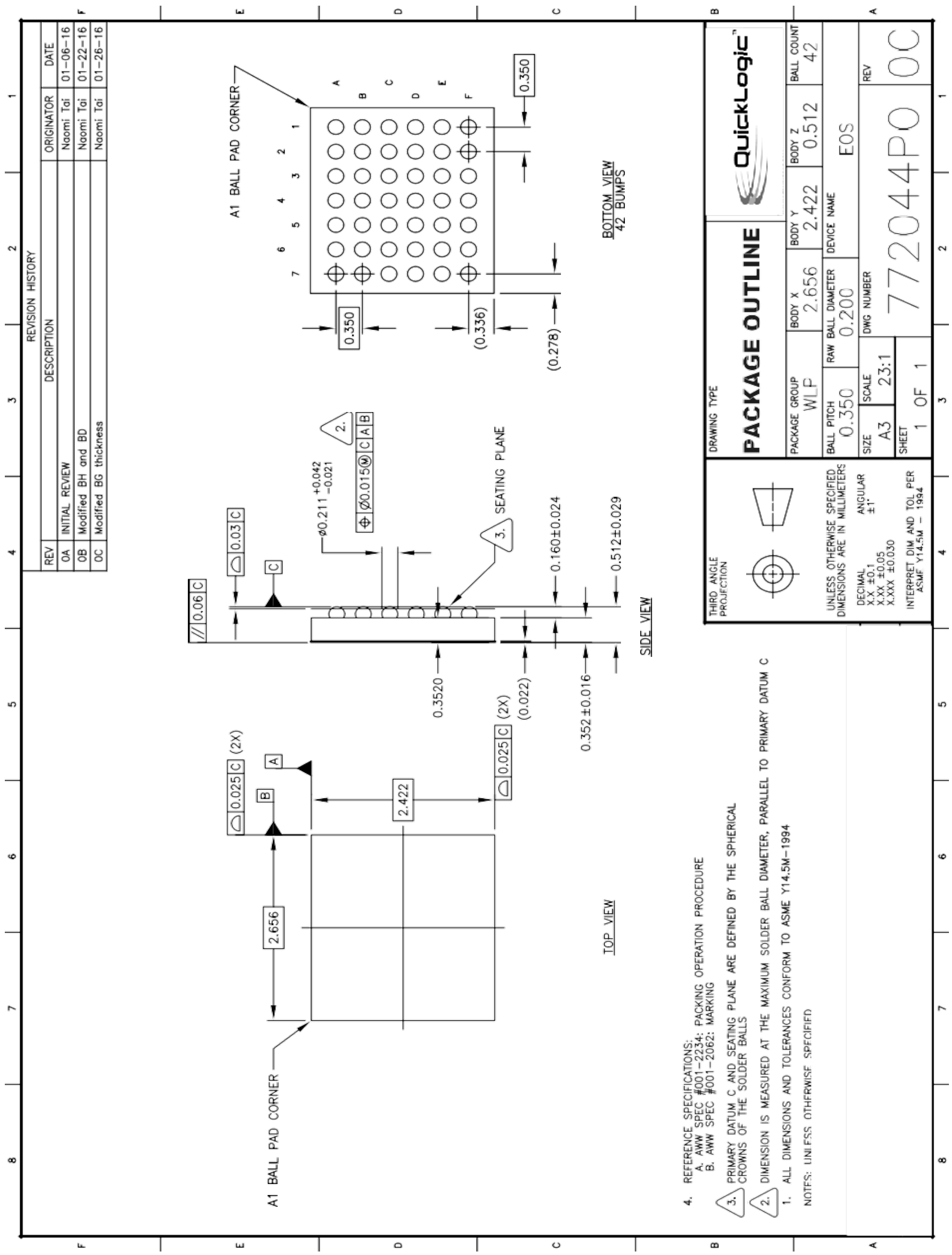
Figure 57: Example of a Real-Time Operating System Wearable Design



13. Package Information

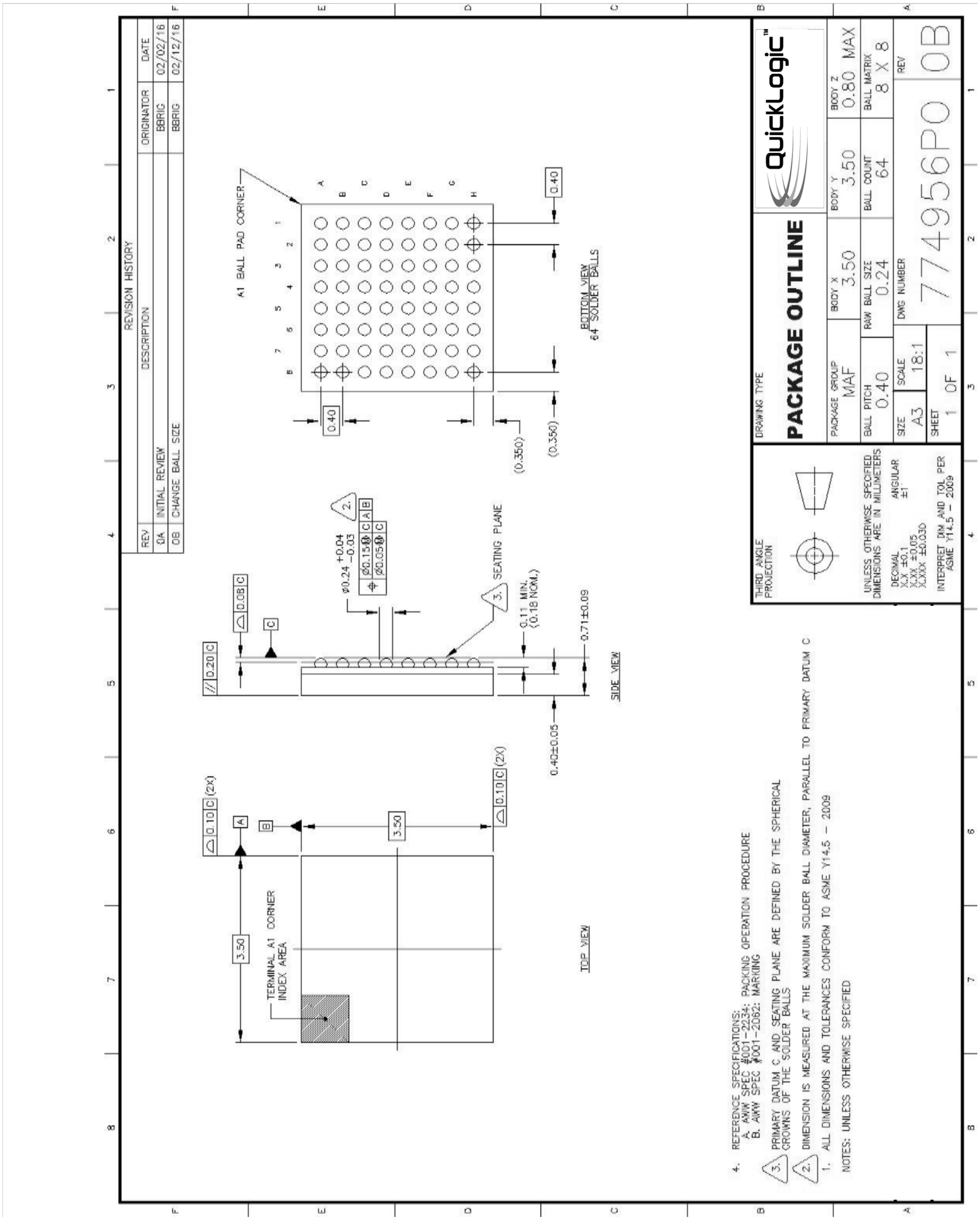
13.1. 42-Ball WLCSP Package Drawing

Figure 58: 42-Ball WLCSP Package Drawing



13.2. 64-Ball BGA Package Drawing

Figure 59: 64-Ball BGA Package Drawing



14. Soldering Information

14.1. Reflow Profile

QuickLogic follows IPC/JEDEC J-STD-020 specification for lead-free devices. **Figure 60** shows the Pb-free component preconditioning reflow profile.

Figure 60: Pb-Free Component Preconditioning Reflow Profile

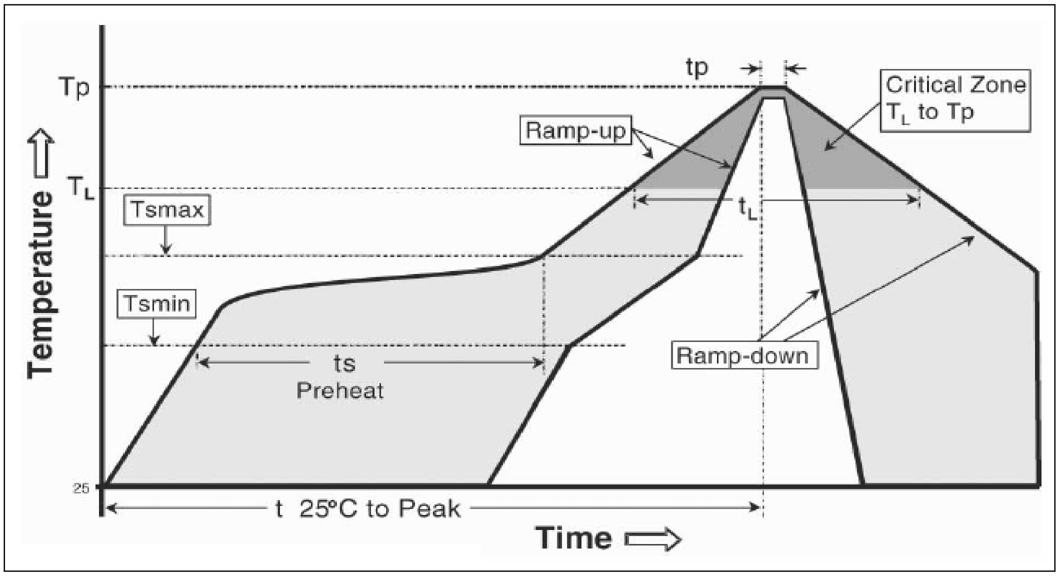


Table 42 shows the Pb-free component preconditioning reflow profile.

Table 42: Pb-Free Component Preconditioning Reflow Profile[a,b]

Profile Feature	Profile Conditions
Average ramp-up rate (Ts_{max}) to Tp)	3°C per sec. max.
Preheat: Temperature Min (Ts_{min}) Temperature Max (Ts_{max}) Time (Ts_{min} to Ts_{max}) (ts)	 150°C 200°C 60 sec. to 120 sec.
Time maintained above: Temperature (T_L) Time (t_L)	 217°C 60 sec. to 150 sec.
Peak Temperature (Tp)	260°C
Time within 5°C of actual peak temperature (260°C)	20 sec. to 40 sec.
Ramp-down rate	6°C per sec. max.
Time 25°C to peak temperature	8 min. max.

a. The above conditions are used for component qualifications. This should not be interpreted as the recommended profile for board mounting. Customers should optimize their board mounting reflow profile based on their specific conditions such as board design, solder paste, etc.

b. All temperatures are measured on the package body surface.

14.2. Package Thermal Characteristics

The EOS S3 voice and sensor processing platform is available for Commercial (-20 C to 85 C) junction temperature ranges.

Thermal Resistance Equations:

$\theta_{JC} = (T_J - T_C)/P$

$\theta_{JA} = (TJ - TA)/P$

$P_{MAX} = (T_{JMAX} - T_{AMAX})/\theta_{JA}$

Parameter Description:

θ_{JC}: Junction-to-case thermal resistance

θ_{JA}: Junction-to-ambient thermal resistance T_J: Junction temperature

T_A: Ambient temperature

P: Power dissipated by the device while operating P_{MAX}: The maximum power dissipation for the device T_{JMAX}: Maximum junction temperature

T_{AMAX}: Maximum ambient temperature

NOTE: The maximum junction temperature (T_{JMAX}) is 125°C. To calculate the maximum power dissipation for a device package, look up θ_{JA} from **Table 43**, pick an appropriate T_{AMAX} and use: P_{MAX} = (125°C - T_{AMAX})/ θ_{JA}.

Table 43: Package Thermal Characteristics

Package Description		θ_{JC} (° C/W)	Air Flow (m/sec)	θ_{JA} (° C/W)
Package Type	Pin Count			
WLCSP	42	5.3	0.0	71.5
			0.5	66.4
			1.0	65.0
			1.5	64.1
BGA	64	36.6	0.0	69.9
			0.5	67.6
			1.0	66.7
			1.5	66.1

15. Revision History

Version	Date	Revision
1.0	April 2019	First release.