## Chapter 1. ARM Cortex-M7 CPU

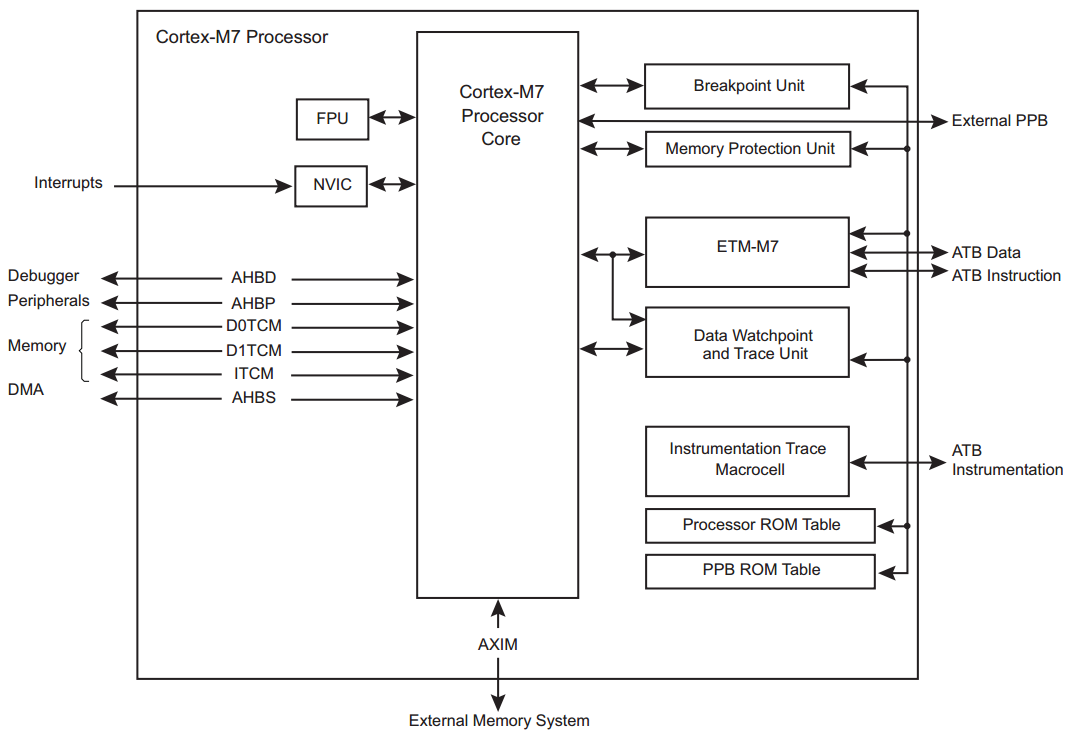
### 1.1 Description

The ARM Cortex-M7 processor implements the ARMv7-M architecture and runs 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions.

The double-precision Floating-Point Unit (FPU) supports the ARMv7 VFPv5 architecture. It is tightly integrated to the ARM Cortex-M7 processor pipeline. It provides trapless execution and is optimized for scalar operation. It can generate an Undefined instruction exception on vector instructions that enables the programmer to emulate vector capability in software.

Note: Refer to ARM reference documents Cortex-M7 Processor User Guide (ARM DUI 0644) and Cortex-M7 Technical Reference Manual (ARM DDI 0489), available on [www.arm.com](http://www.arm.com).

### 1.2 Block Diagram



### 1.3 Embedded Characteristics

* ARM Cortex-M7 with 16 KB of instruction cache and 16 KB of data cache
* 128KB ITCM & 64KB DTCM
* ARMv7-M Thumb instruction set combines high-code density with 32-bit performance
* Tightly Coupled Memory (TCM) interfaces:
* 64-bit ITCM interface
* 2 x 32-bit DTCM interfaces
* Memory Protection Unit (MPU): up to 16 protected memory regions for safety/critical applications
* Dedicated low-latency AHB-Lite peripheral (AHBP) interface
* Dedicated AHB slave (AHBS) interface for system access to TCMs
* Low-latency interrupt processing achieved by a Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor
* DSP extensions for efficient signal processing and complex algorithm execution
* IEEE Standard 754-2008 Floating Point Unit (FPU)
* A low-cost debug solution with the ability to:
* Implement breadpoints.
* Implement watchpoints, tracing, and system profiling
* TPIU
* DAP
* Hardware integer divide instructions
* Extensive debug and trace capabilities:
* Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing, and code profiling

## Chapter 2. Low-Speed Peripherals

### 2.1 UART Controller

#### 2.1.1 Overview

The UART is modeled after the widely used 16550 UART. The register definition is quite the same except that the register offset address in aligned to 32-bit word boundaries due to the 32bit APB bus interface of this UART module.

#### 2.1.2 Block Diagram



Fig. 2.1 UART functional block diagram

#### 2.1.3 Features

* 9 on-chip general UART controller inside Sirius
  + 5 simple UART without DMA feature
  + 1 full UART with DMA feature
  + 3 simple UART with DMA feature
* DMA-based or interrupt-based operation
* For all UART, 16-byte depth transmit and receive FIFO
* 16550 compatible function
  + 5-8 data bits per charater
  + Optional parity bit
  + 1/1.5/2-bit stop bit
* Programmable serial data baud rate as calculated by the following:

Buadrate = sclk/(16xdivisor)

### 2.2 I2C Controller

#### 2.2.1 Overview

The I2C controller provides support for a communication link between integrated circuits on a board. It is a sample two-wire bus which consists of a serial data line (SDA) and a serial clock (SCL). The Sirius provides five I2C Controller to enable system software to communicate serially with I2C buses. Each I2C controller operates as a master.

#### 2.2.2 Block Diagram



Fig. 2.1 I2C Controller functional block diagram

The I2C controller consists of an APB interface, an I2C interface, FIFO logic to buffer data and shift logic for parallel-to-serial and serial-to-parallel conversion. Control logic is responsible for implementing the I2C protocol. The I2C controller is instantiated in Sirius through APB interface and APB clock input (PCLK) is for both APB bridge and I2C internal control logic.

#### 2.2.3 Features

* 5 on-chip I2C controller in Sirius
* All I2C support DMA feature
* Support multi-slave operation
* 16 depth transmit and receive buffers
* Three speeds:
  + Standard mode (0 to 100Kb/s)
  + Fast mode (≤400Kb/s)
  + High-speed mode (≤3.4Mb/s)
* Clock synchronization
* 7-or-10-bit addressing
* 7-or-10-bit combined format transfers
* Bulk transmit mode
* Handles bit and byte waiting at all bus speeds

### 2.3 SPI Controller

#### 2.3.1 Overview

There are 4 SPI master controllers and 2 SPI slaves instantiated in Sirius through APB interface. The APB input clock (PCLK) is for both APB interface and SPI controller internal control logic. The PCLK is 300MHz for AHB/APB bridge.

#### 2.3.2 Block Diagram



Fig. 2.3 SPI Controller functional block diagram

#### 2.3.3 Features

* 6 on-chip SPI masters inside Sirius
  + 4 SPI masters, 2 support one chip-select and 2 support 5 chip-selects output
  + 2 SPI slaves
  + All SPI support DMA feature
* DMA-based or interrupt-based operation
* All SPI with 16 depth transmit and receive buffers
* Independent masking of interrupts
* Programmable features:
  + Clock bit-rate – dynamic control of the serial bit rate of the data transfer
  + Data Item size (4 to 16 bits)

### 2.4 CAN controller

#### 2.4.1 Overview

There are four controller area network (CAN) controllers instantiated in Sirius through APB interface. The CAN controller bus consists of two wire, CAN-H and CAN-L, and the bus level is determined by their potential difference.

#### 2.4.2 Block diagram



Fig. 2.4 CAN controller functional block diagram

#### 2.4.3 Features

The CAN controller in Sirius offers the following features:

* Support CAN2.0B
* Data rate up to 1Mbit/s
* Programmable baud rate prescaler (1 to 1/256)
* 11-bit standard and 29-bit extended identifiers
* Two transmit buffers
  + One Primary Transmit Buffer (PTB)
  + Optional configurable Secondary Transmit Buffer (STB)
* Independent and programmable internal 29 bit acceptance filters
* Configurable interrupt sources

### 2.5 Watchdog Timer

#### 2.5.1 Block Diagram



Fig. 2.5 Watchdog timer functional block diagram

#### 2.5.2 Features

* 32-bit APB bus interface, compliance with AMBA 2.0 specification
* Configurable APB data bus widths of 8, 16 and 32 bits
* Configurable watchdog counter width of 16 to 32 bits
* Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
* If a timeout occurs watchdog timer can perform one of the following operations:
  + Generate a system reset
  + Generate an interrupt, restarts the timer, and if the timer is not cleared before a second timeout occurs, generate a system reset
* Test mode signal to decrease the time required during functional test

### 2.6 Timer With PWM

#### 2.6.1 Overview

There are ten 32-bit timers connected to AHB/APB bus bridge in Sirius. The timer is used for both general-purpose time counting and PWM generating. The timer optionally generates an interrupt when the 32-bit binary count-down timer reaches zero.

#### 2.6.2 Block Diagram



Fig. 2.6 Timer functional block diagram

#### 2.6.3 Features

* 32-bit APB bus interface, compliance with AMBA 2.0 specification
* Up to 8 programmable timers
* Supports for two operation modes: free-running and user-defined count
* Supports interrupt generation
* Supports PWM generation

### 2.7 GPIO

#### 2.7.1 Overview

There are 5 GPIO modules instantiated in Sirius. Each GPIO includes 32 I/O pins. So Sirius supports up to 160 I/Os.

#### 2.7.2 Block Diagram



Fig. 2.7 GPIO functional block diagram

#### 2.7.3 Features

* 32-bit APB bus interface, compliance with AMBA 2.0 specification
* Up to four ports, A to D, which are separately configurable
* Separate data registers and data direction registers for each signal
* Configurable interrupt mode for Port A
* Supports digital debounce
* Supports up to 160 I/O pins
* All GPIOs are always in input direction in default after power-on-reset
* The driver strength for all of GPIOs is software-programmable

## Chapter 3. AHB DMA Controller

### 3.1 Overview

The DMA with two master ahb interfaces and one slave ahb interface. CPU can pass slave ahb interface to configure the DMA, and the dma can be used with two master ahb interface to implement move data between different data sources and destinations.

### 3.2 Block Diagram



### 3.3 Features

* AHB slave interface – used to program the ahb\_dmac
* Channels
* Up to eight channels, one per source and destination pair
* Unidirectional channels – data transfers in one direction only
* Programmable channel priority
* AHB master interface(s)
* Up to four independent AHB master interfaces that allows:
* Up to four simultaneous DMA transfers
* Masters that can be on different AHB layers (multi-layer support)
* Source and destination that can be on different AHB layers (pseudo fly-by

performance)

* Configurable data bus width (up to 256 bits) for each AHB master interface
* Configurable endianness for master interfaces
* Transfers
* Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers
* DW\_ahb\_dmac to or from APB peripherals through the APB bridge
* Configurable identification register
* Component ID parameters for configurable software driver support

## Chapter 4. SD Card Controller

### 4.1 Overview

The SD card module in Sirius has one slave AHB interface, one AHB master interface and an internal DMA engine. The SD card controller is configured by the AHB slave interface. There are two ways to transfer data from the host memory to the card device and vice versa. One is that the data is transferred through the AHB slave interface by the AHB master such as CPU and son on in the system, and the other one is that the internal DMA engine of the SD card controller transfers data between the host and the card through the AHB master interface. The latter data transfer approach is much more efficient in most situations.

### 4.2 Block Diagram



Fig. 4.1 SD card module block diagram

The SD card module consists of two main functional blocks, which are illustrated in Fig. 4.1.

* Bus Interface Unit (BIU) – Provides AMBA AHB and DMA interfaces for register and data read/writes.
* Card Interface Unit (CIU) – Takes care of the SD card protocols and provides the clock management.

### 4.3 Features

* The following are features of the SD card module in Sirius:  
  ■ Supports Secure Digital memory protocol commands  
  ■ Supports Secure Digital I/O protocol commands   
  ■ 32-bit addressing supported for Master Interface and Slave Interface  
  ■ Support for 1.8/3.3V of operation control
* Internal DMA Block Features   
  ■ Single-channel; single engine used for Transmit and Receive, which are mutually exclusive   
  ■ Dual-buffer and chained descriptor linked list  
  ■Descriptor architecture allows large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 4KB of data in chained mode and 8KB of data in dual-buffer mode  
  ■ Comprehensive status reporting for normal operation and transfers with errors   
  ■ Programmable interrupt options for different operational conditions
* AHB Master Interface Features  
  ■ Supports 32-bit data and 32-bit addressing  
  ■ Supports split, retry, and error AHB responses; does not support wrap   
  ■ Allows selection of AHB burst type through software
* Bus Interface Unit (BIU) Features   
  ■ Supports data widths of 32 bits   
  ■ Does not generate split, retry, or error responses on the AMBA Slave AHB bus  
  ■ Supports FIFO over-run and under-run prevention by stopping card clock
* Card Interface Unit (CIU) Features  
  ■ Supports Command Completion Signal and interrupts to host  
  ■ Supports Command Completion Signal disable  
  ■ Supports CRC generation and error detection  
  ■ Supports programmable baud rate. Supports up to 4 clock dividers to support simultaneous  
  operation of multiple cards with different clock speed requirements   
  ■ Supports card detection and initialization  
  ■ Supports write protection  
  ■ Supports SDIO interrupts in 1-bit and 4-bit modes  
  ■ Supports SDIO suspend and resume operation  
  ■ Supports SDIO read wait  
  ■ Supports block size of 1 to 65,535 bytes  
  ■ Supports Busy Clear Interrupt for the write data transfers to the card

UHS-1 and Voltage-Switching Features  
■ Support for UHS 50 and UHS 104 cards with the following speeds, frequencies, and voltages, as appropriate for each card:  
❑ Default Speed Mode – 25 MHz, 3.3V  
❑ High Speed mode – 50 MHz, 3.3V  
❑ SDR12 – 25 MHz, 1.8V  
❑ SDR25 – 50 MHz, 1.8V  
❑ SDR50 – 100 MHz, 1.8V  
❑ SDR104 – 208 MHz, 1.8V  
❑ DDR50 – 50 MHz, 1.8V  
■ Voltage switching

## Chapter 5. Quad-SPI Flash Controller

### 5.1 Overview

The SPI flash controller in Sirius is one instruction based controller. The command instruction code is 64 bit pattern. The controller decodes the command code and issue SPI instruction to flash chip. The controller can support fast read/dual read/quad read mode. The SPI clock frequency can be set to 25Mhz/50Mhz/100Mhz.

### 5.2 Block Diagram



### 5.3 Features

The SPI Flash Controller has the following features:

* Support single/quad/qpi operation
* SPI clock can be programmed to 25Mhz/50Mhz/100Mhz
* Support 3-byte/4-byte address mode
* Support DMA read operation
* Total 32 instruction space
* Instruction can be programmed
* Default instruction is used for winbond flash