Arm® Musca-A Test Chip and Board

Technical Reference Manual

arm

101107_0000_02_en

Arm® Musca-A Test Chip and Board

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	31 January 2018	Non-Confidential	First issue
0000-01	01 August 2019	Non-Confidential	Second issue
0000-02	06 January 2020	Non-Confidential	Third issue

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

www.arm.com

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

CE Conformity

The *Waste Electrical and Electronic Equipment* (WEEE) marking, that is, the crossed out wheelie-bin figure, indicates that this product must not be disposed of with general waste within the European Union. To prevent possible harm to the environment from uncontrolled waste disposal, the user is required to recycle the product responsibly to promote reuse of material resources. To comply with EU law, you must dispose of the product in one of the following ways:

- Return it to the distributer where it was purchased. The distributer is required to arrange free collection when requested.
- Recycle it using local WEEE recycling facilities. These facilities are now very common and might provide free collection.
- If purchased directly from Arm, Arm provides free collection. Please e-mail *weee@arm.com* for instructions.

The CE Declaration of Conformity for this product is available on request.

The system should be powered down when not in use.

It is recommended that ESD precautions be taken when handling this product.

The product generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. There is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures:

- Ensure attached cables do not lie across any sensitive equipment.
- Reorient the receiving antenna.
- Increase the distance between the equipment and the receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Note

It is recommended that wherever possible shielded interface cables be used.

Contents
Arm® Musca-A Test Chip and Board Technical Reference Manual

Preface

This preface introduces the *Arm® Musca-A Test Chip and Board Technical Reference Manual*.

It contains the following:

About this book

This book describes the Musca-A test chip and all board variants.

Intended audience

This book is written for experienced hardware and software developers to enable low-power, secure IoT endpoint development using the Musca-A test chip and board.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter introduces the Musca-A test chip and Musca-A board.

Chapter 2 Hardware description
This chapter describes the Musca-A test chip and board hardware.

Chapter 3 Programmers model
This chapter describes the programmers model of the Musca-A board and Musca-A test chip.

Appendix A Signal descriptions
This appendix describes the signals present at the interface connectors.

Appendix B Specifications
This appendix contains electrical specifications of the Musca-A board.

Appendix C Revisions
This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes cross-references, and citations.

bold
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`mono`space
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
Denotes language keywords when used outside example code.

`<and>`

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

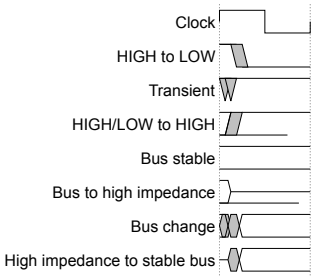

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications

- *Arm® Musca-A Test Chip and Board Technical Overview* (101195).
- *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Overview* (101123).
- *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* (DDI 0574B).
- *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual* (DDI 0571).
- *Arm® Cortex®-M System Design Kit Technical Reference Manual* (DDI 0479).
- *Arm® Cortex®-M33 Processor Technical Reference Manual* (100230).
- *PrimeCell UART (PL011) Technical Reference Manual* (DDI 0183).
- *Arm® PrimeCell Real Time Clock (PL031) Technical Reference Manual* (DDI 0224).
- *CoreSight™ Components Technical Reference Manual* (DDI 0314).
- *Arm® DS-5 Arm DSTREAM User Guide* (DUI 0481).
- *Arm® DS-5 Using the Debug Hardware Configuration Utilities* (DUI 0498).

The following books are only available to licensees or require registration with Arm.

- *Arm® CryptoCell-312 Technical Reference Manual* (100774).
- *Arm® v7-M Architecture Reference Manual* (DDI 0403).
- *Arm® AMBA® 5 AHB Protocol Specification* (IHI 0033).
- *Arm® AMBA® APB Protocol Specification Version 2.0* (IHI 0024).

Other publications

None.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm Musca-A Test Chip and Board Technical Reference Manual*.
- The number 101107_0000_02_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the Musca-A test chip and Musca-A board.

It contains the following sections:

1.1 Precautions

This section describes precautions that ensure safety and prevent damage to your Musca-A board.

This section contains the following subsections:

1.1.1 Ensuring safety

The DAPLink 5V USB connector supplies power to the Musca-A board.

——— Warning ———

Do not use the Musca-A board near equipment that is sensitive to electromagnetic emissions, for example, medical equipment.

———————————

1.1.2 Operating temperature

The Musca-A board has been tested in the temperature range 0°C-30°C.

1.1.3 Preventing damage

The Musca-A board is intended for use within a laboratory or engineering development environment.

——— Caution ———

To avoid damage to the Musca-A board, observe the following precautions:

- Never subject the board to high electrostatic potentials. Observe *ElectroStatic Discharge* (ESD) precautions when handling any board.
- Always wear a grounding strap when handling the board.
- Only hold the board by the edges.
- Avoid touching the component pins or any other metallic element.
- Do not fit an Arduino Shield while the Musca-A board is powered up.

———————————

1.2 About the Musca-A board

The Musca-A board is a development system that demonstrates the foundation of single-chip secure *Internet of Things* (IoT) endpoints.

The Musca-A board provides access to the Musca-A test chip which implements the Arm CoreLink SSE-200 Subsystem for Embedded product. The Musca-A test chip, and the SSE-200, enable design and development of a low-power, secure IoT endpoint.

The Musca-A test chip features two Cortex-M33 processors, a memory system, and sensor interfaces.

The Musca-A board enables development and evaluation of custom software on the Musca-A test chip. The board and test chip provide the following main features:

- CoreLink SSE-200 Subsystem for Embedded that includes, but is not limited to, the following:
  - CPU0 element: One Cortex-M33 processor. No FPU, no DSP, no coprocessor.
  - CPU1 element: One Cortex-M33 processor. FPU, DSP, no coprocessor.
  - Two 2KB caches, one for each processor.
  - 4 × 32KB SRAM: CPU0 96KB + CPU1 32KB *Tightly Coupled Memory* (TCM).
  - 2MB system memory SRAM.
  - Arm CryptoCell-312 (partial functionality).
- On-board DAPLink that provides the following access:
  - *Serial Wire Debug* (SWD).
  - *USB Mass Storage Device* (USBMSD).
  - UART. The UART on the Musca-A test chip does not support hardware flow control.
  - Remote reset.
- On-board:
  - 3-axis orientation and motion sensor (gyro sensor).
  - Temperature sensor/ADC/DAC.
  - *Quad Serial Peripheral Interface* (QSPI) 8MB boot flash.

  ———— **Note** ————

  Normal Musca-A test chip boot operation is from external QSPI 8MB boot flash memory. Only the lowest 256KB of QSPI memory is directly accessible. More memory is accessible through indirect addressing.

  ————————————

- P-JTAG processor debug and SWD header.
- User RGB LED, status LEDs, user reset, and ON/OFF push buttons.
- Board is powered from USB 5V power or li-ion rechargeable battery backup (battery not supplied), selectable by a slider switch.

The Musca-A board provides headers for Arduino Shield expansion to support development of custom designs. The Shield interface provides:

- 16 3V3 GPIO.
- UART, no hardware flow control.
- SPI (master only).
- I^2C.
- I^2S.
- 3-channel PWM.
- 6-channel analog interface.

1.3 Location of components

The following figure shows the physical layout of the Musca-A board.

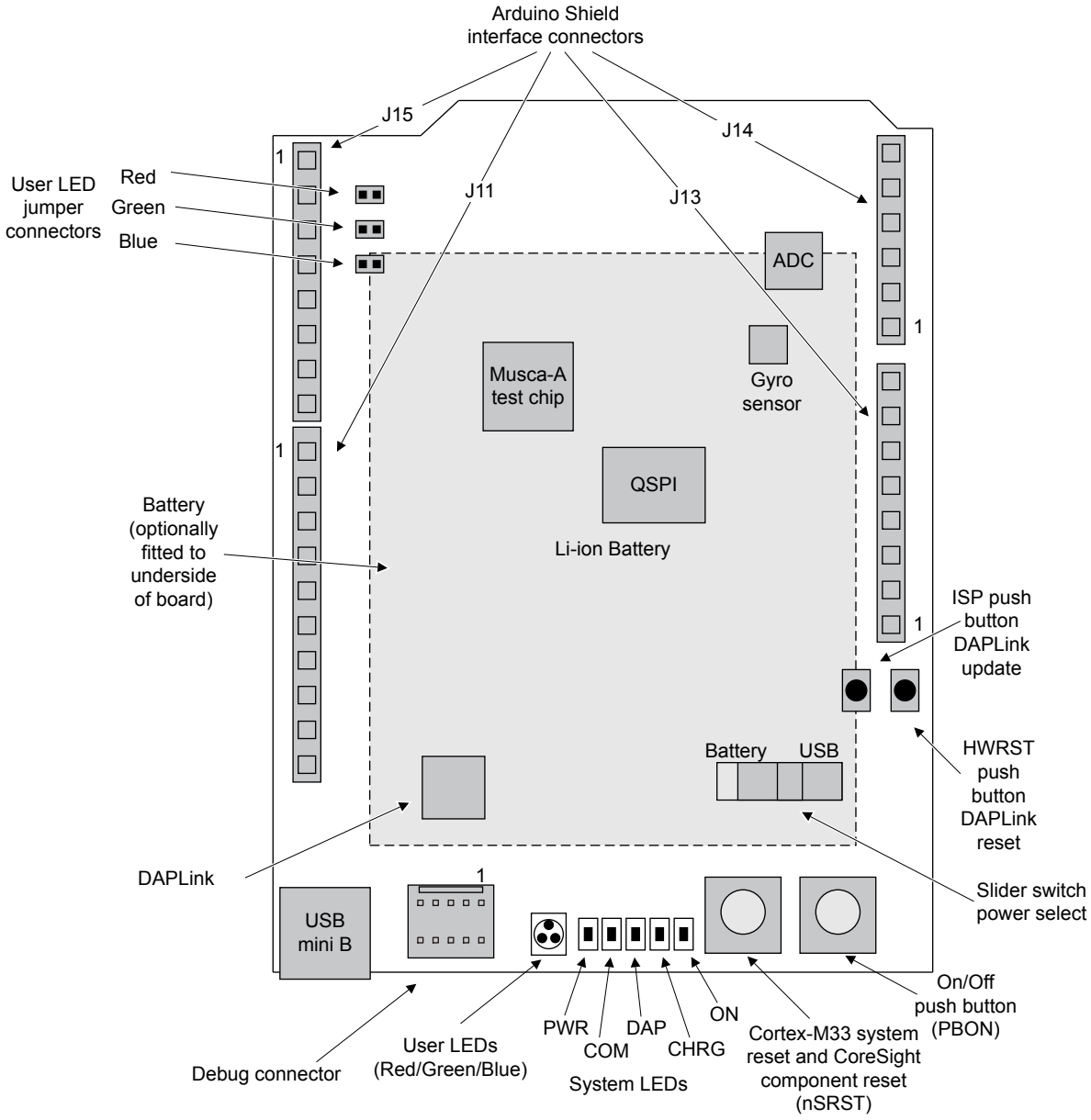

Figure 1-1 Layout of the Musca-A board

1.4 Part numbers and ordering information

The following table contains part numbers for ordering different Musca-A board variants.

Table 1-1 Musca-A board variants

Board variant	Board	Arm part number (MSI number)	Board HBI Number	Comment
varA	MUSCA-A1 TESTCHIP Board	V2M-MUSCA-0345A	HBI-0326A	-
varB	MUSCA-A2 TESTCHIP Board	V2M-MUSCA-0345B	HBI-0326B	Musc-A1 test chip updated to Musca-A2 for hardened security testing.
varC	MUSCA-A2 TESTCHIP Board (HP)	V2M-MUSCA-0345C	HBI-0326C	Musc-A1 test chip updated to Musca-A2 for hardened security testing. More power available to Arduino Shield.

Chapter 2
Hardware description

This chapter describes the Musca-A test chip and board hardware.

It contains the following sections:

2.1 Board hardware

The hardware infrastructure of the Musca-A board provides access to the Musca-A test chip and supports Shield expansion.

Overview of Musca-A board hardware

The user peripheral interfaces connect directly between the Musca-A test chip and the peripheral device, or between the test chip and the Shield header.

The DAPLink controller functions as:

- A DAPLink to provide the following over USB:
  - *Serial Wire Debug* (SWD).
  - UART, The UART on the Musca-A test chip does not support hardware flow control.
  - *USB Mass Storage Device* (USBMSD for user QSPI code or DAPLink firmware update).
  - Remote reset.
- A system controller to control the following:
  - Power supplies.
  - Resets.
  - SCC pre-loading.

The following figure shows the hardware infrastructure of the Musca-A board.

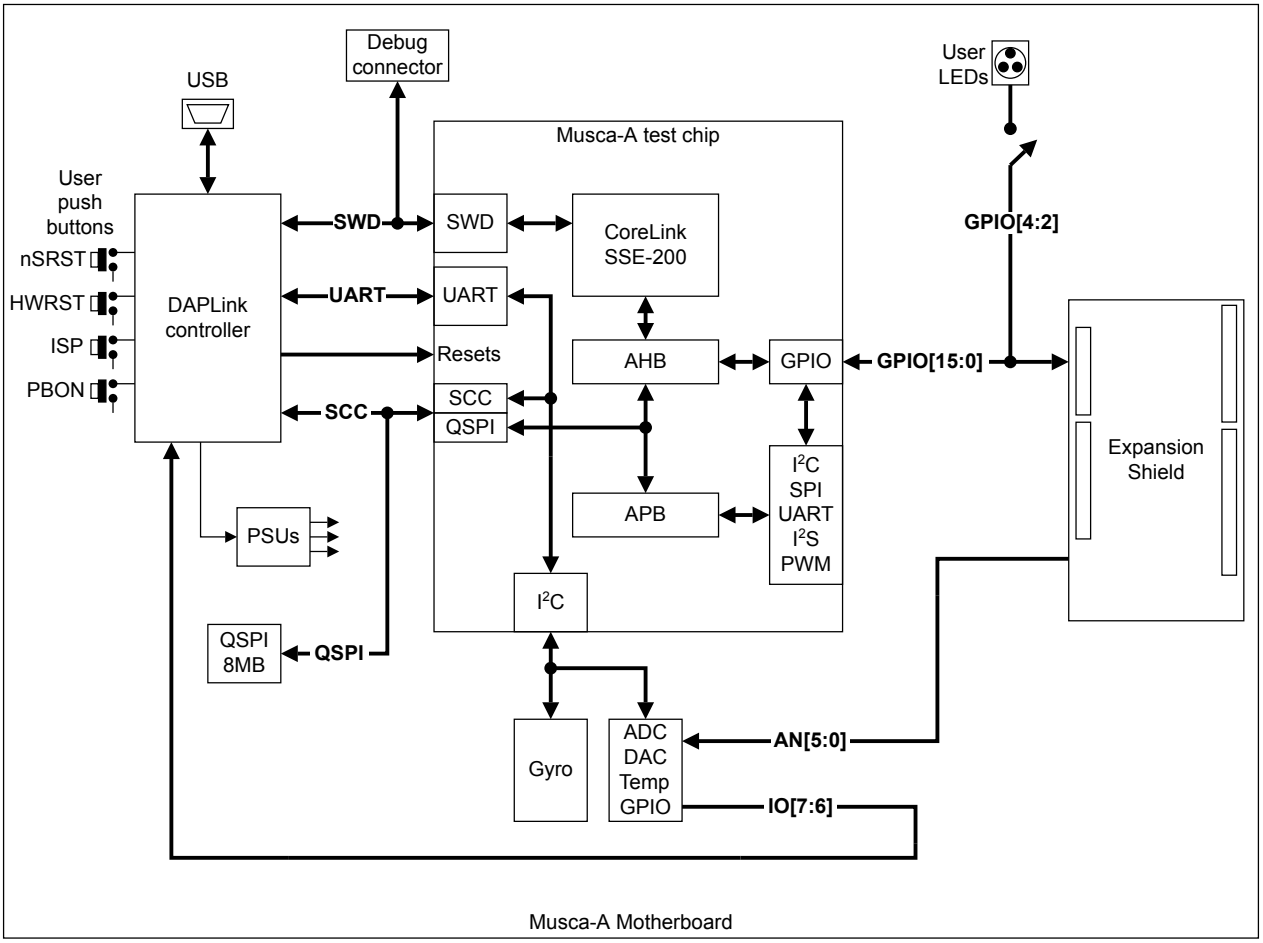

Figure 2-1 Hardware infrastructure of the Musca-A board

Musca-A board components

The Musca-A board contains the following components:

- Musca-A test chip, incorporating CoreLink SSE-200 subsystem.
- Arduino Shield expansion to enable custom designs:
  - UART. The UART on the Musca-A test chip does not support hardware flow control.
  - I^2S.
  - SPI (master only).
  - I^2C.
  - PWM.
  - 6-channel analog input from on-board combined ADC/DAC.
  - 16 3V3 GPIO.
- On-board DAPLink that enables the following functionality over USB:
  - *Serial Wire Debug* (SWD).
  - *USB Mass Storage Device* (USBMSD).
  - UART. The UART on the Musca-A test chip does not support hardware flow control.
  - Remote reset.
- On-board gyro sensor:
  - MMA7660FC 3-axis orientation and motion detection sensor.
  - I^2C interface to Musca-A test chip.
- On-board combined ADC/DAC/temperature sensor:
  - AD5593.
  - 6-channel 3V3 ADC/DAC/GPIO output to Arduino Shield.
  - ADC output to DAPLink controller.
  - Temperature indicator.
- 8MB on-board QSPI boot flash:
  - Both Secure and Non-secure access.
  - Normal Musca-A test chip boot operation is from external QSPI 8MB boot flash memory. Only the lowest 256KB of QSPI memory is directly accessible. More memory is accessible through indirect addressing.
- Debug connector that provides access to:
  - P-JTAG processor debug.
  - *Serial Wire Debug* (SWD).
- User push-buttons:
  - PBON power On/Off.
  - nSRST: Cortex-M33 system reset and CoreSight component reset.
  - ISP: Updates DAPLink firmware.
  - HWRST: Resets DAPLink.
- RGB LED. Jumper connectors provide optional connections between:
  - Red LED and connection between Arduino header and Musca-A test chip GPIO[2] pin, optional PWM0.
  - Green LED and connection between Arduino header and Musca-A test chip GPIO[3] pin, optional PWM1.
  - Blue LED and connection between Arduino header and Musca-A test chip GPIO[4] pin, optional PWM2.

    Note

    The SCC registers select the functions of pins GPIO[4:2], see *2.9 Arduino Shield expansion* on page 2-35 and *3.9 Serial Configuration Control registers* on page 3-89.

- Status LEDs.
- 5V USB or battery power, selectable by slider switch:
  - DAPLink 5V USB connector.
  - CLN 523450, Lithium Ion, 3.7V, 950mAh (not supplied).

Related references

1.3 Location of components on page 1-14

2.2 Musca-A test chip

The Musca-A test chip features Cortex-M33 processors, a memory system, integrated connectivity, and sensor interfaces.

Overview of Musca-A test chip

The Musca-A test chip consists of the SSE-200 subsystem with external memory, interfaces, clock generator, and *Serial Configuration Control* (SCC) registers for setting default powerup values.

See the following documentation for more information on the SSE-200 subsystem:

- *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Overview*.
- *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

The following figure shows a high-level view of the architecture of the Musca-A test chip.

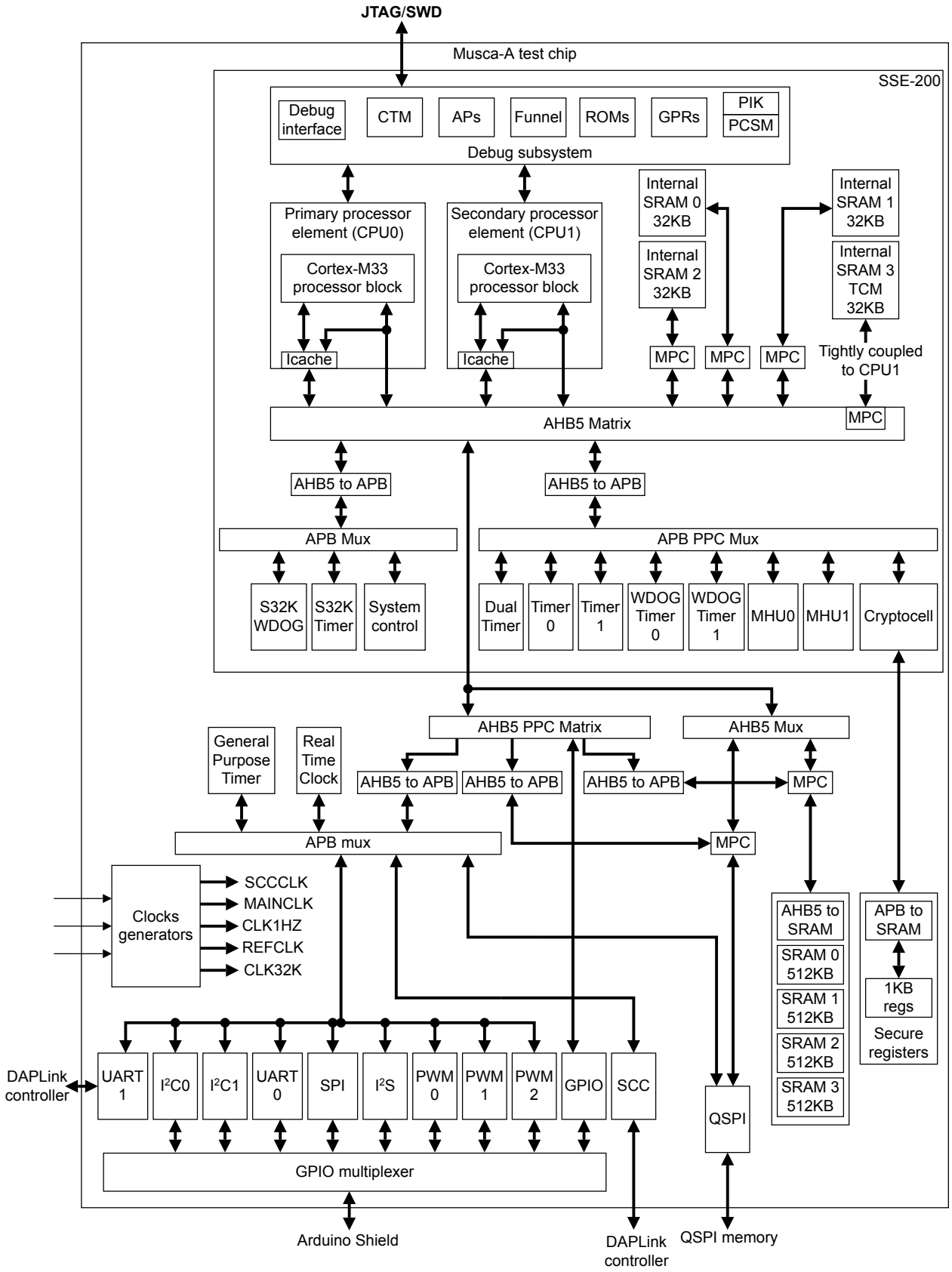

Figure 2-2 Architecture of the Musca-A test chip

Major components and systems of the Musca-A test chip

SSE-200 subsystem

- Two processors:
  - CPU0 element: Cortex-M33 No FPU, no DSP, no coprocessor, 50MHz maximum. Used as main processor.
  - CPU1 element: Cortex-M33 with FPU and DSP, no coprocessor, 170MHz maximum.
- Memory system:
  - One 2KB cache for each Cortex-M33 processor.
  - System 96KB SRAM and 32KB Tightly Coupled Memory (TCM) tight coupled to CPU1. Organized as 4 × 32KB.
- CoreSight component and *Serial Wire Debug* (SWD).
- CryptoCell-312 (partial functionality).
- Secure AMBA interconnect:
  - AHB5 Bus matrix.
  - AHB5 TrustZone® Memory Protection Controller (MPC).
  - AHB5 TrustZone Peripheral Protection Controller (PPC).
  - AHB5 Exclusive Access Monitor (EAM).
  - AHB5 Access Control Gates. (ACG).
  - AHB5 to APB bridges.
  - Expansion AHB5 master and slave buses - two each.
- Security components:
  - Implementation-Defined Attribution Unit (IDAU).
  - Secure and Non-secure configurable peripherals and memory access.
  - Secure boot.
- Secure APB peripherals:
  - One general-purpose timer with configurable security in the **S32KCLK** domain.
  - Two general-purpose timers, Timer0 and Timer1, in the **SYSCLK** domain.
  - One Cortex-M System Design Kit (CMSDK) dual timer with configurable security.
  - One secure watchdog in the **S32KCLK** domain.
  - Two secure watchdogs in the **SYSCLK** domain.

Musca-A test chip outside the SSE-200 subsystem

- 2MB Code SRAM: 4 × 512KB independently power-enabled.
- 1KB Security Registers emulating One-Time Programming (OTP).
- One Real Time Clock (RTC) in the Always ON domain.
- One 32-bit general-purpose timer running on 32.768kHz with programmable interrupts.
- 16 external (GPIO) interrupts.
- 16 GPIO.
- Three-channel I^2S:
  - Two master transmitters.
  - One master receiver.
- Three independent Pulse Width Modulation (PWM) outputs.
- Two UARTs, UART0 user, UART1 debug. The UART on the Musca-A test chip does not support hardware flow control.
- Two I^2C:
  - I^2C0, can be used as Master (default) or slave.
  - I^2C1. Master only to on-board interfaces.
- One SPI (master only).
- One alternate function I/O multiplexer (all function DIOS).
- One QSPI for external flash control with Execute in Place (XIP) capability.
- Off-chip QSPI flash boot.
- External powerup reset.
- Three system clock sources:
  - External **REFCLK**, 32.768kHz.
  - External **FASTCLK**, 32MHz.
  - On-chip PLL. Input 32.768kHz. Output up to 200MHz to Cortex-M33 processors.
  - Debug:
    - One JTAG/SWD port.
- One Serial Configuration Controller (SCC) with dual access port:
  - SCC serial during reset.
  - APB after reset.

2.3 Software, firmware, board, and tools setup

Arm supplies software and firmware for the Musca-A board.

Software and firmware

You can access software and firmware at the Arm Community pages which are accessible from *https://www.arm.com/musca*.

Setting up a project

To power the board, connect the USB port to your computer and press the PBON user push button. The DAPLink interface appears in the Windows device manager as an Mbed composite device part of which is the Mbed serial port (UART). The following figure shows an example configuration which shows the Mbed composite device and the Mbed serial port.

Note

Other components of the Mbed composite device are not visible in the Windows device manager. See *2.1 Board hardware* on page 2-17 for the other components of the Mbed composite device.

The UART on the Musca-A test chip does not support hardware flow control.

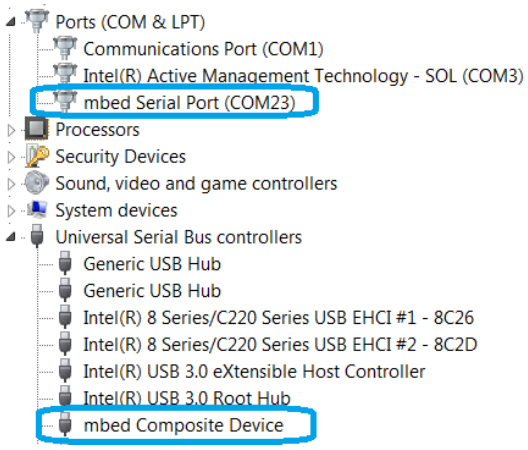

Figure 2-3 DAPLink interface

Updating DAPLink firmware

To update the DAPLink firmware, you can use the DAPLink drag and drop update method:

1. Press and hold the ISP button while powering up the board using the USB lead.
2. Delete file `firmware.bin` that appears in the `CRP DISABLD USB` drive.
3. Copy `DAPLink_QSPI_XTAL_v1.2.bin`, or a later version, to the `CRP DISABLD drive`.
   - From a Windows system, you can simply Drag and Drop the file.
   - On Linux/Mac OS, use the following command:

     ```
     dd if={new_firmware.bin} of=/Volumes/CRP\ DISABLD/firmware.bin conv=notrunc
     ```

4. Power cycle the board using the USB lead. Do not press the ISP button during the power cycle.

Updating QSPI software image

To update the QSPI image, perform the following steps:

1. Power up the board by connecting the USB lead and pressing the PBON push button.
2. Drop a `.bin` format QSPI software image onto the `MBED` drive, for example `blinky.bin`.
3. Power cycle the board or press the nSRST button to reset the system and boot from the new QSPI software image.

Note

The file `blinky.bin` is available at the Arm Community pages which are accessible from *https://www.arm.com/musca*.

DAPLink UART setting

The default DAPLink UART setting is:

- 115,200 baud (8N1).

2.4 User components and status LEDs

The Musca-A board provides three user LEDs, reset and on/off push buttons, and system status LEDs.

User LEDs

One RGB LED, which can be connected to Musca-A test chip GPIO[4:2] outputs by completing the user jumper connections:

- Red: Jumper GP2LED connects this LED to GPIO[2].
- Green: Jumper GP3LED connects this LED to GPIO[3].
- Blue: Jumper GP4LED connects this LED to GPIO[4].

The IOMUX registers, part of the *Serial Configuration Control* (SCC) registers, select the Musca-A test chip default configuration options, ALTF1 or ALTF2. See *3.9.1 IOMUX Registers* on page 3-89.

Power-select slider switch

Selects either USB 5V power or Li-ion 3.7V battery power.

Status LEDs

The Musca-A board provides the following system status LEDs:

- PWR:
  - Orange LED: Indicates that power is connected.
- COM:
  - Green LED: Indicates that USB UART is active.
- DAP:
  - Blue LED: Indicates DAP activity.
- CHRG:
  - Orange LED: Indicates that Li-ion battery charging is in progress.
- ON:
  - Green LED: Indicates board power supplies active.

User push buttons

The Musca-A board provides the following user push buttons:

- PBON power on/off.
- nSRST: Cortex-M33 system reset and CoreSight debug reset.
- ISP: Updates DAPLink firmware.
- HWRST: Resets DAPLink.

See *2.6 Resets and powerup* on page 2-30 for more information on the user push buttons.

Related references

1.3 Location of components on page 1-14

2.5 Clocks

The Musca-A board provides on-board clocks which drive the systems in the Musca-A test chip and board.

The on-board clocks are:

- **REFCLK**:
  - 32.768kHz crystal clock. The clock goes to a PLL in the Musca-A test chip and is multiplied up to drive the Cortex-M33 processors and SSE-200 subsystem.

    ——— **Caution** ———

    The default frequency for both processors is 50MHz. This frequency is the maximum operating frequency for the primary processor, CPU0. This frequency is set by the SCC registers and registers FCLK_DIV and SYSCLK_DIV in the SSE-200 subsystem. You must not attempt to increase the operating frequency of CPU0 above this maximum.

    The maximum operating frequency of the secondary processor, CPU1, is 170MHz. This frequency is set by the SCC registers and register FCLK_DIV in the SSE-200 subsystem. You can increase the operating frequency of CPU1 up to this maximum value.

    ———————————
- **SCCCLK**:
  - Serial Configuration Controller (SCC) interface clock from the DAPLink.

The 24MHz oscillator, **FASTCLK**, and the 32kHz and 32MHz crystals, are reserved for future use.

The following figure shows the Musca-A test chip and board clocking system.

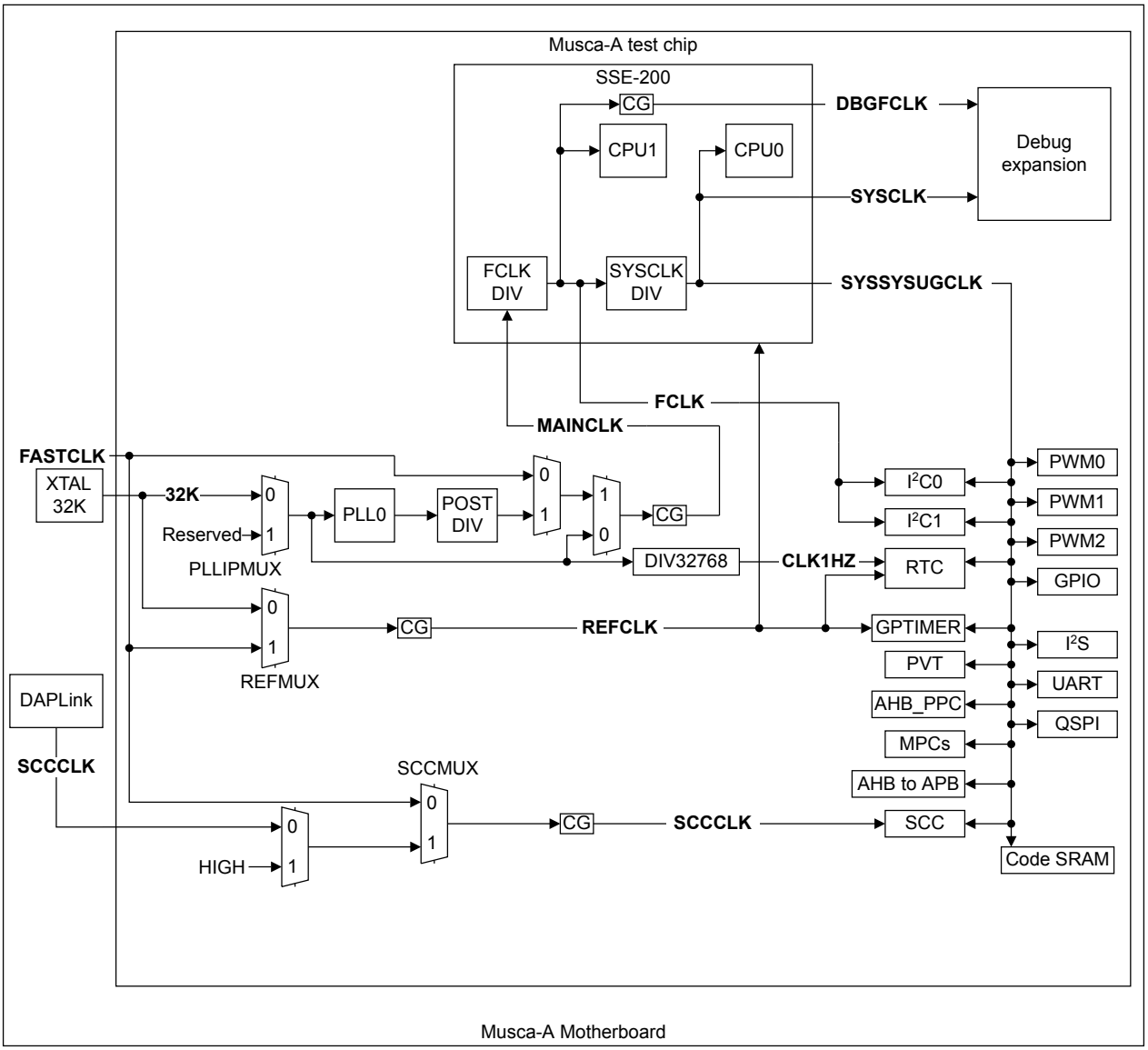

Figure 2-4 Musca-A clock system

Controlling clock frequencies

The SCC registers control the clock system. See *3.9 Serial Configuration Control registers* on page 3-89. The following table shows the SCC clock control registers.

Table 2-1 Clock control SCC registers

Register	Register function	Register description
CLK_CTRL	Controls the following blocks: • PLLIPMUX. • REFMUX. • SCCMUX. • MAINMUX1. • MAINMUX2. • MAINCLK *Clock Gate*, (CG). • REFCLK CG. • SCCCLK CG.	*CLK_CTRL Register* on page 3-94.
PLL_CTRL	Controls the following PLL functions: • Powerdown. • PLL bypass. • PLL feedback value to set PLL frequency multiplication factor. Controls POSTDIV block frequency divide value.	*PLL_CTRL Register* on page 3-95

The system control registers FCLK_DIV and SYSCLK_DIV, control the FCLKDIV and SYSCLKDIV dividers in the SSE-200 subsystem. FCLKDIV derives clock FCLK for secondary processor CPU1 and SYSCLKDIV derives SYSCLK for primary processor CPU0.

The following table shows system control registers FCLK_DIV and SYSCLK_DIV.

Table 2-2 System control registers FCLK_DIV and SYSCLK_DIV

Register	Register function	Register description
FCLK_DIV	Controls divider block FCLKDIV in SSE-200 subsystem to derive clock FCLK for secondary processor CPU1.	*FCLK_DIV Register* on page 3-80.
SYSCLK_DIV	Controls divider block SYSCLKDIV in SSE-200 subsystem to derive clock SYSCLK for primary processor CPU0.	*SYSCLK_DIV Register* on page 3-81

2.6 Resets and powerup

The Musca-A board provides three standard resets which are driven from the DAPLink controller.

Resets

The Musca-A board provides the following resets:

- **CFG_nRST**:
  - The Serial Configuration Controller (SCC) interface reset.
- **CB_nRST**:
  - Logic reset.
- **CS_nSRST**:
  - System reset to the Cortex-M33 processors and the CoreSight component.

User push buttons

The Musca-A board supplies four user push buttons for powerup, powerdown, and reset:

- PBON:
  - On/Off push-button.
  - This button powers up, or powers down, the board.
- nSRST:
  - Cortex-M33 system reset and CoreSight component reset (**CS_nSRST**).
- ISP:
  - Push button for updating the DAPLink firmware:

    1. Insert the USB lead while holding down the ISP push button.
    2. Update the firmware using Drag and Drop.
- HWRST:
  - Resets the DAPLink.

Reset sequence

The following figure shows the reset and power up timing cycle including test chip and board configuration.

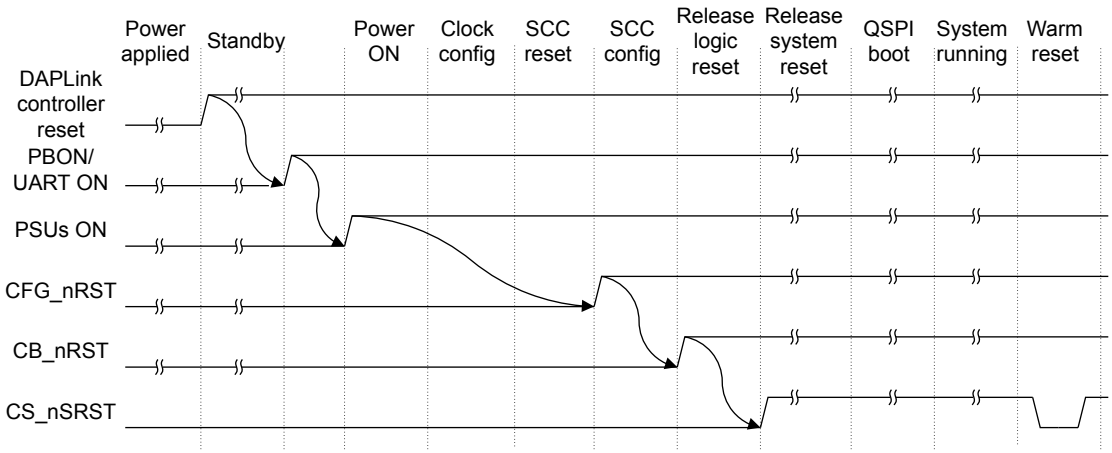

Figure 2-5 Musca-A test chip and board reset and configuration timing

Related references

1.3 Location of components on page 1-14

2.7 Power

The DAPLink 5V USB connector supplies power requirements of the Musca-A board. The motherboard also supports use of an external battery as an alternative to the 5V USB supply.

Overview of board power

The Musca-A board provides on-board regulators to supply power rails in the board and to the test chip.

The varC boards can supply more power to the Arduino Shield than the varA and varB boards.

The following figure shows the power supply scheme for the varA and varB boards.

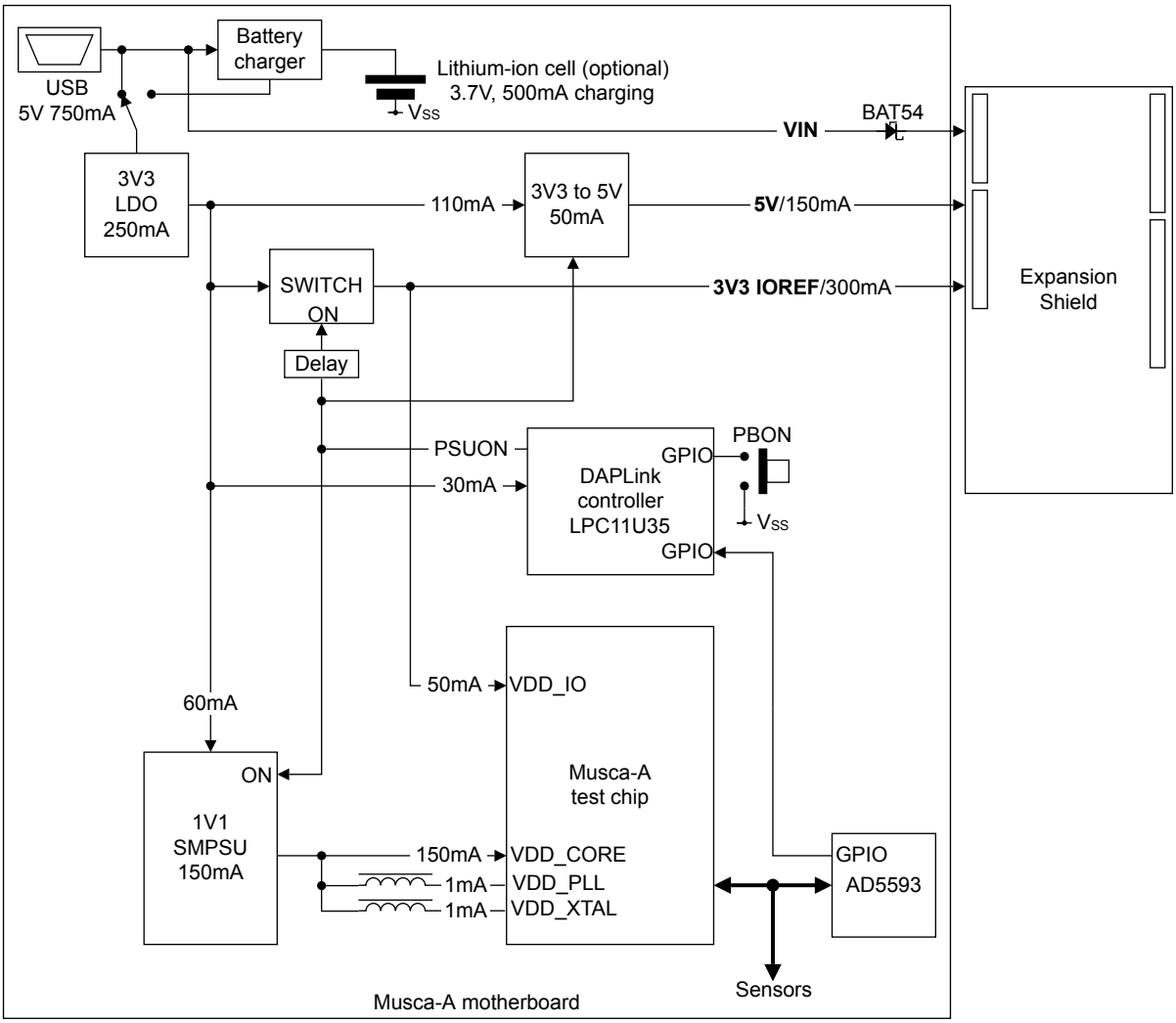

Figure 2-6 Musca-A varA and varB board power supplies

The following figure shows the power supply scheme for the varC board.

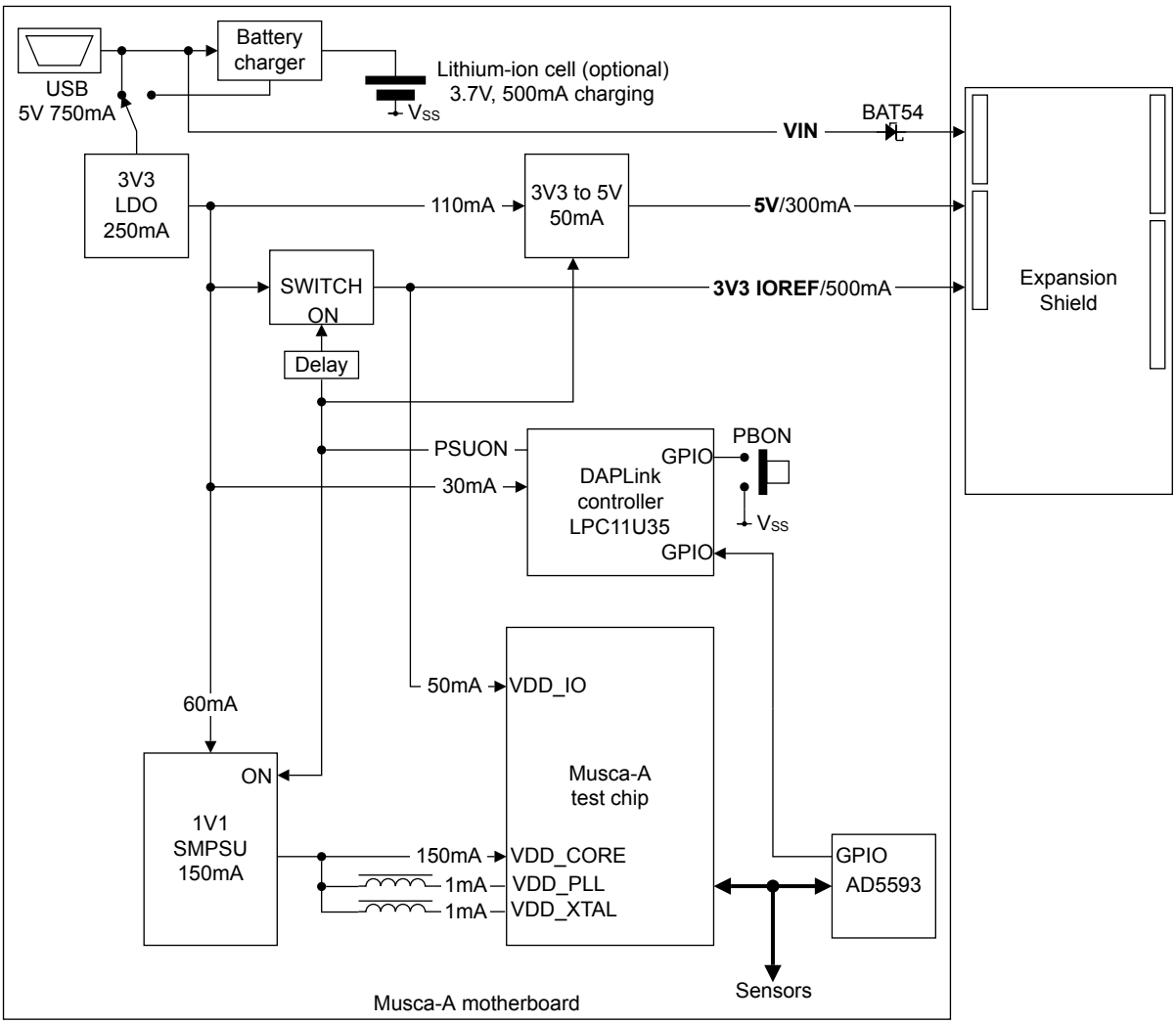

Figure 2-7 Musca-A varC board power supplies

The following table shows the maximum loads that the Musca-A board power rails draw from the power supplies.

Table 2-3 Musca-A varA, varB and varC board power rails

Power rail	Voltage	Max load (mA)	Comment	Board variant
DHUB5	5V	250 (not charging) 750 (charging)	USB power maximum for USB 2.0	All board variants.
VBAT	4.2-3V3	250	Lithium-ion battery	All board variants.
3V3 DAPLink	3V3	30	DAPLink and Musca-A test chip	All board variants.
3V3 IOREF + Arduino 3V3	3V3	300	Arduino 3V3	varA and varB boards.
		500		varC boards.

Table 2-3 Musca-A varA, varB and varC board power rails (continued)

Power rail	Voltage	Max load (mA)	Comment	Board variant
5V	5V	150	Arduino 5V	varA and varB boards.
		300		varC boards.
		50	Startup	varA and varB boards.
		200		varC boards.
VDD_CORE	1.1V	150	Musca-A test chip core supply	All board variants.
VDD_IO	3V3	50	Musca-A test chip I/O supply	All board variants.
VDD_PLL	1V1	1	PLL VDD (PLL VDDV=VDD_IO)	All board variants.
VDD_XTAL	1V1	1	XTAL VDD	All board variants.

Caution

Do not fit an Arduino expansion Shield to the Musca-A board while the Musca-A board is powered up.

Note

The maximum values of decoupling capacitance that can be fitted to the Arduino 3V3 and 5V power rails are:

- varA and varB boards:
  - 22µF to the Arduino 3V3 and 5V power rails.
- varC boards:
  - 100µF to the Arduino 3V3 power rail.
  - 22µF to the Arduino 5V power rail.

External power

All external power to the Musca-A board is supplied through the DAPLink 5V USB connector.

Backup battery

A backup battery can power the Musca-A board, using the connector on the underside of the board.

Arm recommends using the Lithium Ion, CLN 523450, 3.7V, 950mAh battery. The battery is recharged from an external supply during USB 5V operation.

Note

- A slider switch selects the source of board power to be either, external power, or the backup battery. See *1.3 Location of components* on page 1-14 for the location of the slider switch.
- If a battery is fitted while external power is connected, circuitry on the board automatically charges the battery (maximum charging current 500mA).

Related references

1.3 Location of components on page 1-14

2.8 I²C interfaces and sensors

To minimize usage of the test chip pins, the Musca-A test chip provides a single I²C interface for the board sensors.

The board sensors provide basic support for Internet of Things (IoT) software demonstrations and analog support for the Arduino expansion Shield.

The analog converter supports programmable I/O which can be ADC, DAC, or GPIO. Two of the I/O are used as GPIO and connect to the DAPLink controller. This connection can be used to control power by enabling powerdown of the system from the Musca-A test chip application program.

The following figure shows the Musca-A test chip I²C interface and connected peripherals.

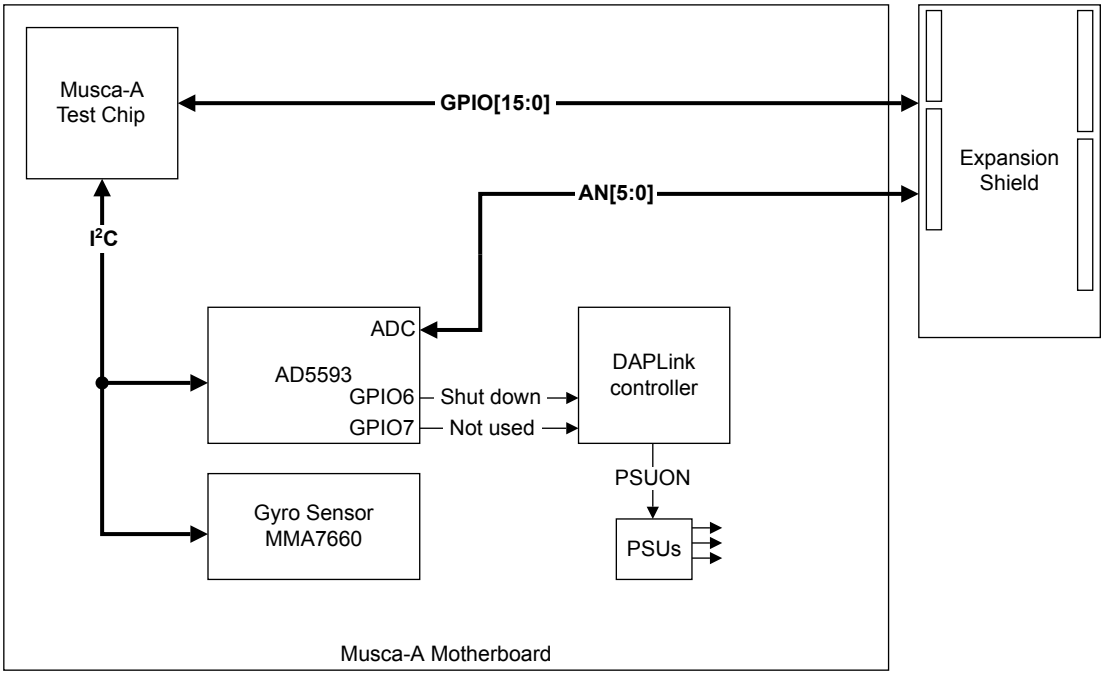

Figure 2-8 I²C interfaces and I²C sensors

Related references

1.3 Location of components on page 1-14

2.9 Arduino Shield expansion

The Musca-A board supports custom system and peripheral design by providing one Arduino Shield interface.

Overview of the Arduino Shield expansion interface

The Arduino Shield interface enables fitting of off-the-shelf boards including:

- Sensors.
- Peripherals.
- PHYs.
- Breakout boards for full custom designs.

Caution

Do not fit an Arduino Shield while the Musca-A board is powered up.

The following figure shows the Shield interface of the Musca-A board.

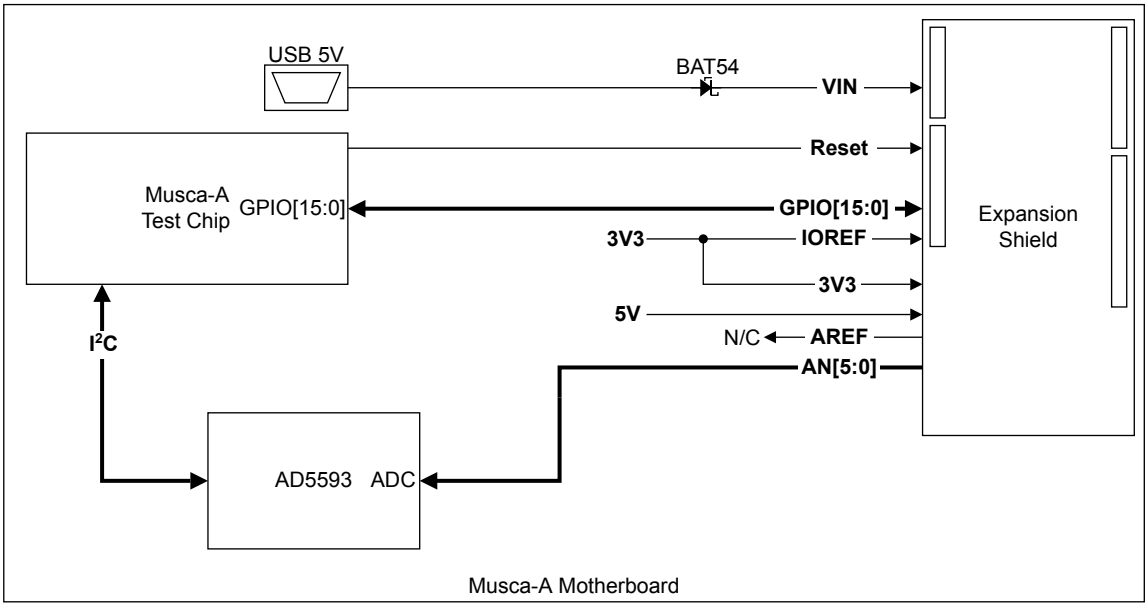

Figure 2-9 Expansion Shield interface

The Arduino Shield expansion interface provides:

- Up to 16 3V3 digital I/O.
- I^2C.
- I^2S.
- SPI (master only).
- UART. The UART on the Musca-A test chip does not support hardware flow control.
- 6-channel 3V3 analog from Shield to Musca-A board.
- **IOREF**, fixed at 3V3.
- Reset.

The IOMUX registers, part of the *Serial Configuration Control* (SCC) registers, select which Shield interface GPIO pin function set connects to the Arduino Shield connectors. See *3.9.1 IOMUX Registers* on page 3-89.

The following table shows the Arduino Shield interface pin functions.

Table 2-4 Arduino Shield interface GPIO functions

Arduino Shield connector	Primary (Reset or powerup)	ALTF1	ALTF2	ALTF3
GPIO[0]	GPIO[0]	UART0 RxD	-	Reserved.
GPIO[1]	GPIO[1]	UART0 TxD	-	
GPIO[2]	GPIO[2]	MR_I^{2}S_SD	PWM0	
GPIO[3]	GPIO[3]	MR_I^{2}S_WS	PWM1	
GPIO[4]	GPIO[4]	MR_I^{2}S_SCK	PWM2	
GPIO[5]	GPIO[5]	MT_I^{2}S_SDO	-	
GPIO[6]	GPIO[6]	MT_I^{2}S_WSO	-	
GPIO[7]	GPIO[7]	MT_I^{2}S1_SD1	-	
GPIO[8]	GPIO[8]	MT_I^{2}S1_WS1	-	
GPIO[9]	GPIO[9]	MT_I^{2}S_SCK	-	
GPIO[10]	GPIO[10]	SPI0 nSS	-	
GPIO[11]	GPIO[11]	SPI0 MOSI	-	
GPIO[12]	GPIO[12]	SPI0 MISO	-	
GPIO[13]	GPIO[13]	SPIO SCK	-	
GPIO[14]	GPIO[14]	I^{2}C0 Data	-	
GPIO[15]	GPIO[15]	I^{2}C0 Clock	-	

Note

MT stands for *Master Transmitter*. MR stands for *Master Receiver*.

Caution

The maximum currents available from the Musca-A board for the Arduino Shield power and reference pins are:

3V3/IOREF Maximum current available is 50mA.

5V Maximum current available is 50mA.

Note

The maximum value of decoupling capacitance that can be fitted to the Arduino 3V3 and 5V power rails is 22µF.

Related references

1.3 Location of components on page 1-14

2.10 QSPI boot memory

Normal Musca-A test chip boot operation is from external QSPI 8MB boot flash memory. Only the lowest 256KB of QSPI memory is directly accessible. More memory is accessible through indirect addressing.

2.11 DAPLink controller

The DAPLink controller is an Arm Mbed™ component that uses a Cortex-M3 processor. The DAPLink controller contains pre-defined firmware that enables access to the CoreSight component in the Musca-A test chip, USB Mass Storage Device (USBMSD), USB UART, and remote reset.

The DAPLink firmware binary image is available from the Arm Community pages which are accessible from *https://www.arm.com/musca*.

Note

The DAPLink controller is only accessible when P-JTAG is not connected to the debug connector.

Related references

1.3 Location of components on page 1-14

2.12 Debug

The debug connector provides access to the CoreSight block and Serial Wire Debug (SWD) on the Musca-A test chip.

The CoreSight component can be accessed over USB using the DAPLink interface.

Chapter 3
Programmers model

This chapter describes the programmers model of the Musca-A board and Musca-A test chip.

It contains the following sections:

3.1 About this programmers model

The following information applies to all register descriptions in this programmers model:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- Unless otherwise stated in the accompanying text:
  - Do not modify undefined register bits.
  - Ignore undefined register bits on reads.
  - All register bits are reset to a logic 0 by a system or powerup reset.
  - All register summary tables in this chapter describe register access type as follows:

    RW Read and write.

    RO Read-only.

    WO Write-only.

3.2 Memory map

The memory map in the Musca-A test chip is based on the SSE-200 memory map. The SSE-200 memory map alternates between Secure and Non-secure regions every 256MB. Only a few address areas are exempted from security mapping because they are related to debug functionality.

SSE-200 system memory map

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

Memory map for the code (AHB expansion) and SRAM regions

The following figure shows the test chip implementation of the code (AHB5 expansion) and SRAM regions of the SSE-200 system memory map.

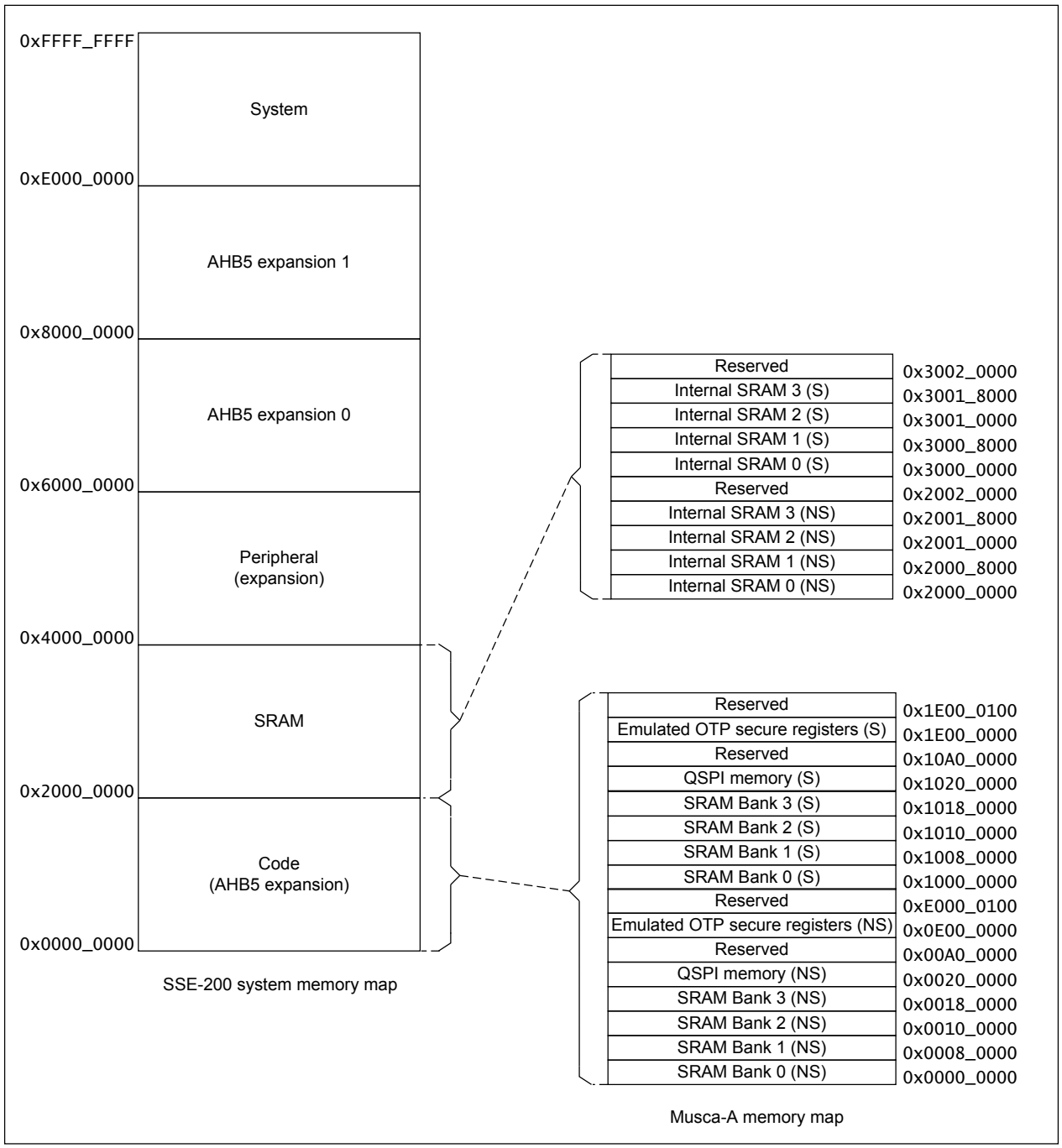

Figure 3-1 Musca-A test chip memory map code and SRAM regions

Caution

Unaligned access to Code SRAM at `0x0000_0000` or `0x1000_0000` is not supported and only unaligned word access can be used in these areas. For code Stack and Heap storage, use the *Tightly Coupled Memory* (TCM) internal SRAM areas starting at `0x2000_0000` or `0x3000_0000`

Memory map for the Peripheral (expansion) region

The following figure shows the test chip implementation of the Peripheral (expansion) region of the SSE-200 memory map.

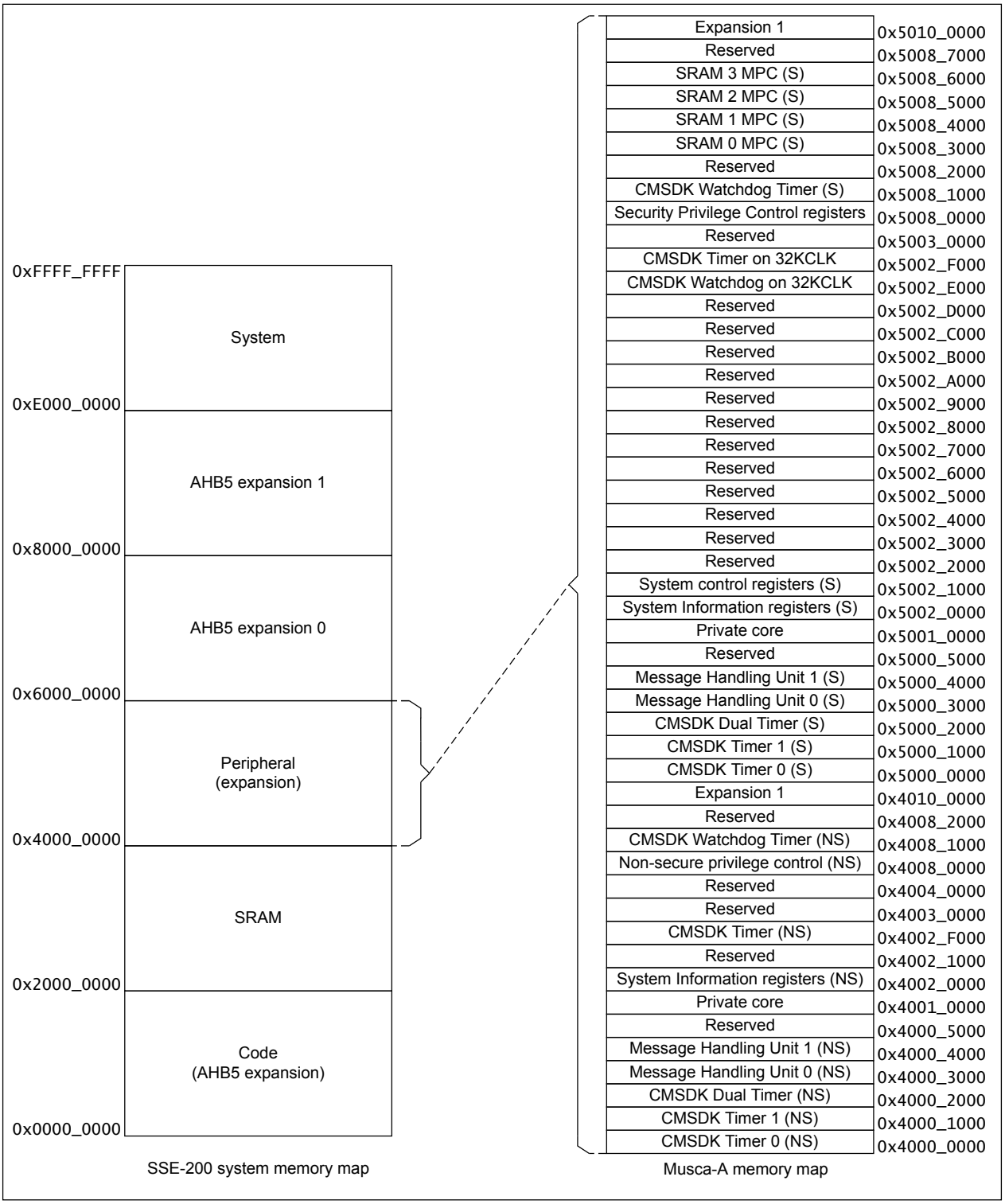

Figure 3-2 Musca-A test chip memory map Peripheral region

Memory map for the Secure and Non-secure expansion 1 regions

The following figure shows the test chip implementation of the Secure and Non-secure Expansion 1 regions of the SSE-200 memory map.

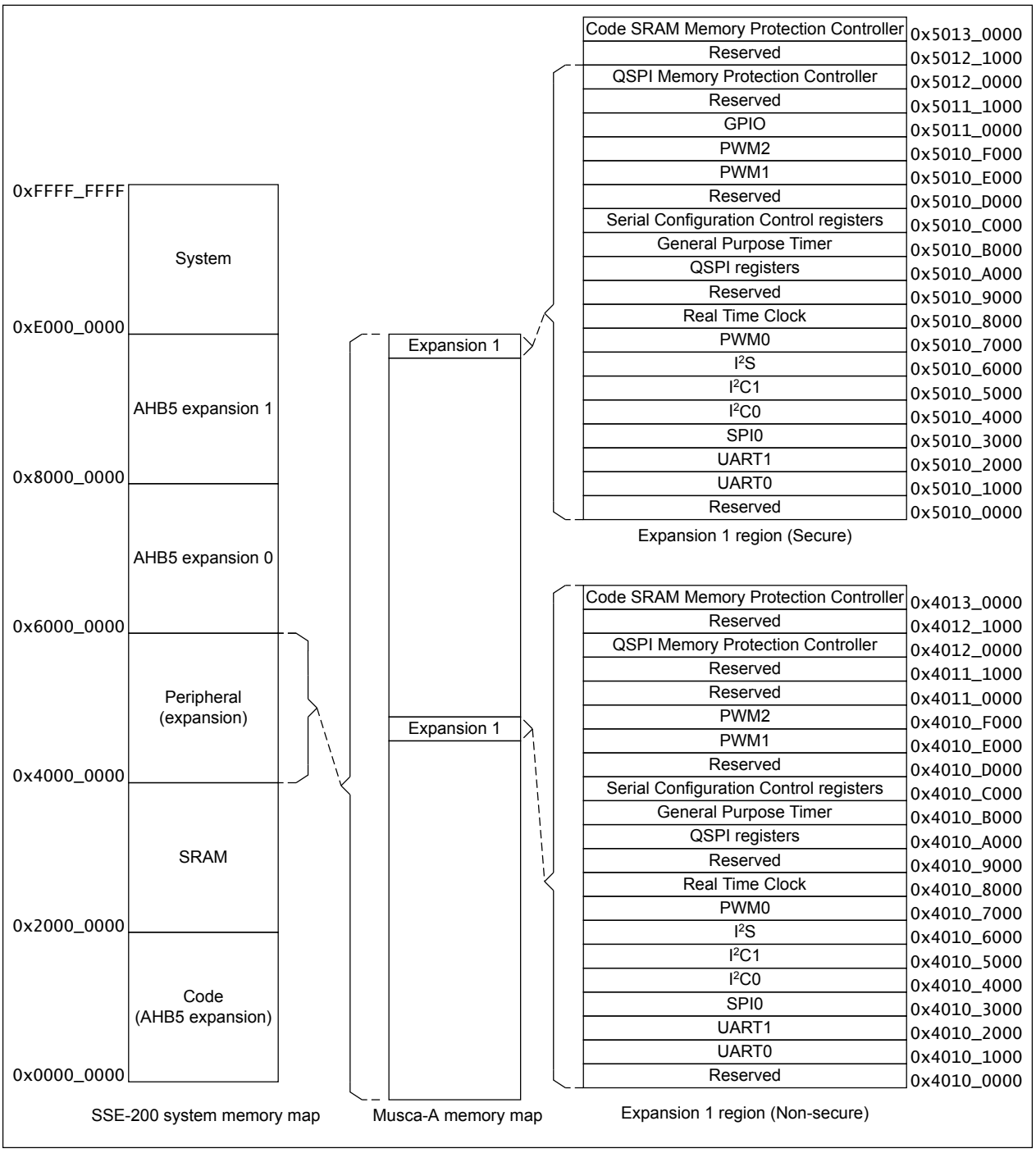

Figure 3-3 Musca-A test chip memory map Secure and Non-secure expansion regions

Memory map for the System region

The following figure shows the test chip implementation of the System region of the SSE-200 memory map.

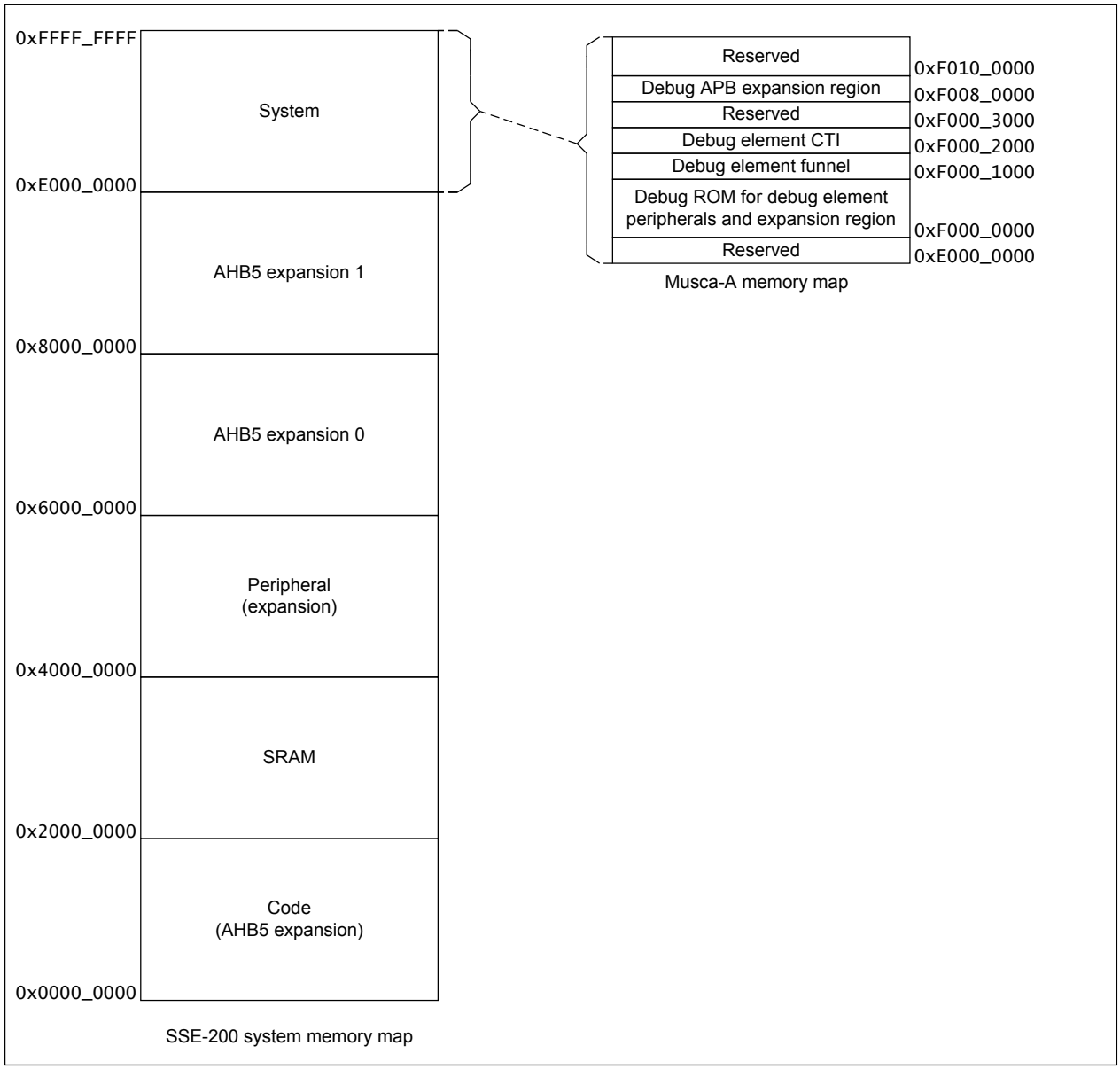

Figure 3-4 Musca-A test chip memory map System region

Complete memory map

The following table shows the complete test chip memory map. Undefined memory locations are reserved and software must not attempt to access these locations.

Table 3-1 Memory map

Non-secure		Secure		Description		
From	To	From	To	Size	Non-secure	Secure
0x0000_0000	0x0007_FFFF	0x1000_0000	0x1007_FFFF	512KB	Code SRAM Bank 0	Code SRAM Bank 0
0x0008_0000	0x000F_FFFF	0x1008_0000	0x100F_FFFF	512KB	Code SRAM Bank 1	Code SRAM Bank 1
0x0010_0000	0x0017_FFFF	0x1010_0000	0x1017_FFFF	512KB	Code SRAM Bank 2	Code SRAM Bank 2

Table 3-1 Memory map (continued)

Non-secure		Secure		Description		
From	To	From	To	Size	Non-secure	Secure
0x0018_0000	0x001F_FFFF	0x1018_0000	0x101F_FFFF	512KB	Code SRAM Bank 3	Code SRAM Bank 3
0x0020_0000	0x009F_FFFF	0x1020_0000	0x109F_FFFF	8MB	QSPI - memory **Note** Only the lowest 256KB of QSPI memory is directly accessible. Additional memory is accessible through indirect addressing. Arm supplies example code that demonstrates how to copy QSPI memory to Code SRAM enabling up to 2MB of Code memory.	QSPI - memory **Note** Only the lowest 256KB of QSPI memory is directly accessible. Additional memory is accessible through indirect addressing. Arm supplies example code that demonstrates how to copy QSPI memory to Code SRAM enabling up to 2MB of Code memory.
0x0E00_0000	0x0E00_00FF	0x1E00_0000	0x1E00_00FF	1KB	OTP secure registers emulating One-Time Programming	OTP secure registers emulating One-Time Programming
0x2000_0000	0x2000_7FFF	0x3000_0000	0x3000_7FFF	32KB	Internal SRAM Bank 0	Internal SRAM Bank 0
0x2000_8000	0x2000_FFFF	0x3000_8000	0x3000_FFFF	32KB	Internal SRAM Bank 1	Internal SRAM Bank 1
0x2001_0000	0x2001_7FFF	0x3001_0000	0x3001_7FFF	32KB	Internal SRAM Bank 2	Internal SRAM Bank 2
0x2001_8000	0x2001_7FFF	0x3001_8000	0x3001_7FFF	32KB	Internal SRAM Bank 3	Internal SRAM Bank 3
0x4000_0000	0x4000_0FFF	0x5000_0000	0x5000_0FFF	4KB	CMSDK Timer 0	CMSDK Timer 0
0x4000_1000	0x4000_1FFF	0x5000_1000	0x5000_1FFF	4KB	CMSDK Timer 1	CMSDK Timer 1
0x4000_2000	0x4000_2FFF	0x5000_2000	0x5000_2FFF	4KB	CMSDK Dual Timer	CMSDK Dual Timer
0x4000_3000	0x4000_3FFF	0x5000_3000	0x5000_3FFF	4KB	Message Handling Unit 0	Message Handling Unit 0
0x4000_4000	0x4000_4FFF	0x5000_4000	0x5000_4FFF	4KB	Message Handling Unit 1	Message Handling Unit 1
0x4001_0000	0x4001_FFFF	0x5001_0000	0x5001_FFFF	64KB	Private core	Private core
0x4002_0000	0x4002_0FFF	0x5002_0000	0x5002_0FFF	4KB	System information registers	System information registers
0x4002_1000	0x4002_0FFF	0x5002_1000	0x5002_0FFF	4KB	Reserved	System control registers
-	-	0x5002_E000	0x5002_EFFF	4KB	-	CMSDK Watchdog
0x4002_F000	0x4002_FFFF	0x5002_F000	0x5002_FFFF	4KB	CMSDK Timer	CMSDK Timer
0x4008_0000	0x4008_0FFF	0x5008_0000	0x5008_0FFF	4KB	Non-secure privilege control	Secure privilege control registers
0x4008_1000	0x4008_1FFF	0x5008_1000	0x5008_1FFF	4KB	CMSDK Watchdog Timer	CMSDK Watchdog Timer

Table 3-1 Memory map (continued)

Non-secure		Secure		Description		
From	To	From	To	Size	Non-secure	Secure
-	-	0x5008_3000	0x5008_3FFF	4KB	-	SRAM 0 Memory Protection Controller
-	-	0x5008_4000	0x5008_4FFF	4KB	-	SRAM 1 Memory Protection Controller
-	-	0x5008_5000	0x5008_5FFF	4KB	-	SRAM 2 Memory Protection Controller
-	-	0x5008_6000	0x5008_6FFF	4KB	-	SRAM 3 Memory Protection Controller
0x4010_1000	0x4010_1FFF	0x5010_1000	0x5010_1FFF	4KB	UART0	UART0
0x4010_2000	0x4010_2FFF	0x5010_2000	0x5010_2FFF	4KB	UART1	UART1
0x4010_3000	0x4010_3FFF	0x5010_3000	0x5010_3FFF	4KB	SPI0	SPI0
0x4010_4000	0x4010_4FFF	0x5010_4000	0x5010_4FFF	4KB	I^2C0	I^2C0
0x4010_5000	0x4010_5FFF	0x5010_5000	0x5010_5FFF	4KB	I^2C1	I^2C1
0x4010_6000	0x4010_6FFF	0x5010_6000	0x5010_6FFF	4KB	I^2S	I^2S
0x4010_7000	0x4010_7FFF	0x5010_7000	0x5010_7FFF	4KB	PWM0	PWM0
0x4010_8000	0x4010_8FFF	0x5010_8000	0x5010_8FFF	4KB	Real Time Clock	Real Time Clock
0x4010_A000	0x4010_AFFF	0x5010_A000	0x5010_AFFF	4KB	QSPI registers	QSPI registers
0x4010_B000	0x4010_BFFF	0x5010_B000	0x5010_BFFF	4KB	General Purpose Timer	General Purpose Timer
0x4010_C000	0x4010_CFFF	0x5010_C000	0x5010_CFFF	4KB	Serial Configuration Control registers.	Serial Configuration Control registers
0x4010_E000	0x4010_EFFF	0x5010_E000	0x5010_EFFF	4KB	PWM1	PWM1
0x4010_F000	0x4010_FFFF	0x5010_F000	0x5010_FFFF	4KB	PWM2	PWM2
-	-	0x5011_0000	0x5011_0FFF	4KB	GPIO	GPIO
0x4012_0000	0x4012_0FFF	0x5012_0000	0x5012_0FFF	4KB	QSPI MPC	QSPI MPC
0x4013_0000	0x4013_0FFF	0x5013_0000	0x5013_0FFF	4KB	Code SRAM MPC	Code SRAM MPC
0x6000_0000	0x6FFF_FFFF	0x7000_0000	0x7FFF_FFFF	2GB	Default slave	Unused AHB Master Exp0

Caution

Unaligned access to Code SRAM at `0x0000_0000` or `0x1000_0000` is not supported and only unaligned word access can be used in these areas. For code Stack and Heap storage, use the *Tightly Coupled Memory* (TCM) internal SRAM areas starting at `0x2000_0000` or `0x3000_0000`

3.3 Base element

This section contains registers that control the functions of certain base element components in the test chip.

This section contains the following subsections:

3.3.1 CMSDK timer

The base element of the test chip has two CMSDK timers and registers that control their function.

TIMER 0 registers are at the following base memory addresses:

- `0x4000_0000` in the Non-secure region.
- `0x5000_0000` in the Secure region.

TIMER 1 registers are at the following base memory addresses:

- `0x4000_1000` in the Non-secure region.
- `0x5000_1000` in the Secure region.

The timer can be halted by CTI triggers from the debug subsystem.

Note

The EXTIN input of the timers is connected to the CTI debug halt logic, and if there is a debug halt access it is used to stop the timer counter logic.

To enable this functionality, the EXTIN must be enabled by writing to the CTRL register.

- CTRL bit[2] = `0b1`.
- CTRL bit[1] = `0b0`.

The timer resides in the PD_SYS power domain and is reset by **nWARMRESETSYS**.

See the *Arm® Cortex®-M System Design Kit Technical Reference Manual*.

The following table shows the CMSDK timer control registers in the base element in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers..

Table 3-2 CMSDK timer control registers summary

Offset	Name	Type	Reset	Width	Function
0x0000	CTRL	RW	0x0000_0000	32	**Bit[3]** Interrupt enable. **Bit[2]** Select external input as clock. **Bit[1]** Select external input as enable. **Bit[0]** Enable.
0x0004	VALUE	RW	0x0000_0000	32	Current value.
0x0008	RELOAD	RW	0x0000_0020	32	Reload value. A write to this register sets the current value.
0x000C	INSTATUS INTCLEAR	RW	0x0000_0020	32	Timer interrupt. Write 0x1 to clear.
0x0FDO	PID4	RO	0x0000_0004	32	Peripheral ID register 4.
0x0FD4	PID5	RO	0x0000_0000	32	Peripheral ID register 5.
0x0FD8	PID6	RO	0x0000_0000	32	Peripheral ID register 6.
0x0FDC	PID7	RO	0x0000_0000	32	Peripheral ID register 7.
0x0FE0	PID0	RO	0x0000_0022	32	Peripheral ID register 0. Bits [7:0] Part number [7:0].
0x0FE4	PID1	RO	0x0000_00B8	32	Peripheral ID register 1. Bits [7:4] jep106_id_3_0. Bits [3:0] Part number [11:8].
0x0FE8	PID2	RO	0x0000_000B	32	Peripheral ID register 2. Bits [7:4] Revision. Bits [3] jedec_used. Bits [2:0] jep106_id_6_4.
0x0FEC	PID3	RO	0x0000_0000	32	Peripheral ID register 3. Bits [7:4] ECO revision number. Bits [3:0] Customer modification number.
0x0FFO	CID0	RO	0x0000_000D	32	Component ID register 0.
0x0FF4	CID1	R0	0x0000_00F0	32	Component ID register 1.
0x0FF8	CID2	R0	0x0000_0005	32	Component ID register 2.
0x0FFC	CID3	R0	0x0000_00B1	32	Component ID register 3.

3.3.2 CMSDK dual timer

The base element in the test chip contains a CMSDK dual timer.

The timers can be halted by CTI triggers from the debug subsystem.

The dual timer resides in the PD_SYS power domain and is reset by **nWARMRESETSYS**.

The base memory addresses of the CMSDK dual timer are:

- `0x4000_2000` in the Non-secure region.
- `0x5000_2000` in the Secure region.

See *Arm® Cortex®-M System Design Kit Technical Reference Manual* for full descriptions of these registers.

The following table shows the dual timer control registers in the test chip in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-3 CMSDK dual timer control registers summary

Offset	Name	Type	Reset	Width	Function
`0x0000`	DTIMER1LOAD	RW	`0x0000_0000`	32	Dual Timer 1 load register.
`0x0004`	DTIMER1VALUE	RO	`0xFFFF_FFFF`	32	Dual Timer 1 current value register.
`0x0008`	DTIMER1CONTROL	RW	`0x0000_0020`	32	Dual Timer 1 control register. Bits [31:8] are reserved. Reserved.
`0x000C`	DTIMER1INTCLR	WO	-	32	Dual Timer 1 interrupt clear register.
`0x0010`	DTIMER1RIS	RO	`0x0000_0000`	32	Dual Timer 1 raw interrupt status register. Bits [31:1] are reserved.
`0x0014`	DTIMER1MIS	RO	`0x0000_0000`	32	Dual Timer 1 interrupt status register. Bits [31:1] are reserved.
`0x0018`	DTIMER1BGLOAD	RW	`0x0000_0000`	32	Dual Timer 1 background load register.
`0x0020`	DTIMER2LOAD	RW	`0x0000_0000`	32	Dual timer 2 load register.
`0x0024`	DTIMER2VALUE	RO	`0xFFFF_FFFF`	32	Dual timer 2 current value register.
`0x0028`	DTIMER2CONTROL	RW	`0x0000_0020`	32	Dual timer 2 control register. Bits [31:8] are reserved.
`0x002C`	DTIMER2INTCLR	WO	-	32	Dual timer 2 interrupt clear register.
`0x0030`	DTIMER2RIS	RO	`0x0000_0000`	32	Dual timer 2 raw interrupt status register. Bits [31:1] are reserved.
`0x0034`	DTIMER2MIS	RO	`0x0000_0000`	32	Dual timer 2 interrupt status register. Bits [31:1] are reserved.
`0x0038`	DTIMER2BGLOAD	RW	`0x0000_0000`	32	Dual timer 2 background load register.
`0x0F00`	DTIMERITCR	RW	`0x0000_0000`	32	Integration test control register.
`0x0F04`	DTIMERITOP	WO	`0x0000_0000`	32	Integration test output set register. Bits [31:2] are reserved.
`0x0FD0`	DTIMERPERIPHID4	RO	`0x0000_0004`	32	Peripheral ID Register 4. Bits [31:8] are reserved.

Table 3-3 CMSDK dual timer control registers summary (continued)

Offset	Name	Type	Reset	Width	Function
0x0FE0	DTIMERPERIPHID0	RO	0x0000_0023	32	Peripheral ID Register 0. Bits [31:8] are reserved.
0x0FE4	DTIMERPERIPHID1	RO	0x0000_00B8	32	Peripheral ID Register 1. Bits [31:8] are reserved.
0x0FE8	DTIMERPERIPHID2	RO	0x0000_000B	32	Peripheral ID Register 2. Bits [31:8] are reserved.
0x0FEC	DTIMERPERIPHID3	RO	0x0000_0000	32	Peripheral ID Register 3. Bits [31:8] are reserved.
0x0FF0	DTIMERPCELLID0	RO	0x0000_000D	32	Component ID Register 0. Bits [31:8] are reserved.
0x0FF4	DTIMERPCELLID1	RO	0x0000_00F0	32	Component ID Register 1. Bits [31:8] are reserved.
0x0FF8	DTIMERPCELLID2	RO	0x0000_0005	32	Component ID Register 2. Bits [31:8] are reserved.
0x0FFC	DTIMERPCELLID3	RO	0x0000_00B1	32	Component ID Register 3. Bits [31:8] are reserved.

3.3.3 CMSDK watchdog timers

The base element implements two CMSDK watchdog timers.

The base memory addresses of the base element CMSDK watchdog timers are:

- 0x4008_1000 in the Non-secure region.
- 0x5008_1000 in the Secure region.

Each watchdog is permanently mapped to either a Secure or a Non-secure region of address space:

- The Non-secure watchdog can raise an interrupt to both CPU cores. On a watchdog reset request event, a separate interrupt is raised instead, but software can also choose to allow it to directly reset the system.
- The Secure watchdog can raise a *Non-Maskable Interrupt* (NMI) to both CPU cores. However, in this case, a watchdog reset event resets the entire system.

The timers can be halted by CTI triggers from the debug subsystem.

The timers reside in the PD_SYS power domain and are reset by **nWARMRESETSYS**.

See *Arm® Cortex®-M System Design Kit Technical Reference Manual* for full descriptions of these registers.

The following table shows the watchdog module control registers in the base element in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-4 CMSDK watchdog timers control registers summary

Offset	Name	Type	Reset	Width	Function
0x0000	WDOGLOAD	RW	0xFFFF_FFFF	32	Watchdog load register. The counter decrements from the value in this register. The count restarts from this value immediately following a write access. Minimum write value: 0b1
0x0004	WDOGVALUE	RO	0xFFFF_FFFF	32	Current value of watchdog counter register.
0x0008	WDOGCONTROL	RW	0x0000_0000	8	Timer 1 control register. Bits [31:2] are reserved. Reserved.
0x000C	WDOGINTCLR	WO	-	32	Watchdog interrupt clear register.
0x0010	WDOGRIS	RO	0x0000_0000	1	Watchdog interrupt status register. Bits [31:1] are reserved. Reserved.
0x0014	WDOGMIS	RO	0x0000_0000	1	Watchdog status register. Bits [31:1] are reserved. Reserved.
0x0C00	WDOGLOCK	RW	0x0000_0000	32	Watchdog lock register.
0x0F00	WDOGITCR	RW	0x0000_0000	32	Watchdog integration test control register. Bits [31:1] are reserved. Reserved.
0x0F04	WDOGITOP	WO	0x0000_0000	32	Watchdog integration test output set register.
0x0FD0	WDOGPERIPHID4	RO	0x0000_0004	8	Peripheral ID register 4. Bits [31:8] are reserved. Reserved.
0x0FE0	WDOGPERIPHID0	RO	0x0000_0024	32	Peripheral ID register 0. Bits [31:8] are reserved. Reserved.
0x0FE4	WDOGPERIPHID1	RO	0x0000_00B8	1	Peripheral ID register 1. Bits [31:8] are reserved. Reserved.
0x0FE8	WDOGPERIPHID2	RO	0x0000_000B	1	Peripheral ID register 2. Bits [31:8] are reserved. Reserved.
0x0FEC	WDOGPERIPHID3	RO	0x0000_0000	32	Peripheral ID register 3. Bits [31:8] are reserved. Reserved.
0x0FF0	WDOGPCELLID0	RO	0x0000_000D	8	Component ID Register 0. Bits [31:8] are reserved. Reserved.
0x0FF4	WDOGCELLID1	RO	0x0000_00F0	8	Component ID Register 1. Bits [31:8] are reserved. Reserved.

Table 3-4 CMSDK watchdog timers control registers summary (continued)

Offset	Name	Type	Reset	Width	Function
0x0FF8	WDOGPCELLID2	RO	0x0000_0005	8	Component ID Register 2. Bits [31:8] are reserved. Reserved.
0x0FFC	WDOGPCELLID3	RO	0x0000_00B1	8	Component ID Register 3. Bits [31:8] are reserved. Reserved.

3.3.4 AHB5 TrustZone® Memory Protection Controller

The base element implements an Arm AHB5 TrustZone Memory Protection Controller (MPC) for each SRAM block.

The base memory addresses of the MPC APB configuration interfaces are in the Secure region. The base memory addresses are:

- 0x5008_3000 for SRAM bank 0.
- 0x5008_4000 for SRAM bank 1.
- 0x5008_5000 for SRAM bank 2.
- 0x5008_6000 for SRAM bank 3.

The AHB5 TrustZone MPC gates transactions towards a memory interface when a security violation occurs. The security checking is done based on block/page level which is configured externally by the security controller through an APB interface.

The configuration registers can only be set by the security controller in the system with secure accesses (PPRTO[1]==0). Any type of access can read the identification registers.

APB accesses are internally aligned to word boundaries, so PADDR[1:0] bits are ignored. The PSTRB[3:0] write strobe signals indicate which byte or bytes of the data bus contain valid data.

The following table shows the AHB5 TrustZone Memory Protection Controller registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-5 AHB5 TrustZone MPC registers

Offset	Name	Type	Reset value	Function
0x0000	CTRL	RW	0x0000_0000	Bit[31] – Security lockdown Bit[30:9] – Reserved Bit[8] – Autoincrement Reserved when BLK_SIZE > ADDR_WIDTH-11 Bit[7] – Data interface gating acknowledge (RO) Reserved when GATE_PRESENT = 0 Bit[6] – Data interface gating request. Reserved when GATE_PRESENT = 0 Bit[5] – Reserved Bit[4] – Security error response configuration (CFG_SEC_RESP) 0:RAZ-WI 1: Bus Error Bit[3:0] – Reserved
0x0010	BLK_MAX	RO	-	Maximum value of block-based index register.

Table 3-5 AHB5 TrustZone MPC registers (continued)

Offset	Name	Type	Reset value	Function
0x0014	BLK_CFG	RO	-	Bit[31] – Init in progress Bit[30:4] - Reserved Bit[3:0] – Block size 0: 32 Bytes 1: 64 Bytes … 15: 1MByte Block size = 1 << (BLK_CFG+5)
0x0018	BLK_IDX	RW	0x0000_0000	Index value for accessing block-based look up table.
0x001C	BLK_LUT[n]	RW	0x0000_0000	block-based gating Look Up Table (LUT): Access to block-based look up configuration space pointed to by BLK_IDX. Bit[31:0] – each bit indicates one block: If BLK_IDX is 0x0, bit[0] is block #0, bit[31] is block#31. If BLK_IDX is 0x1, bit[0] is block #32, bit[31] is block#63. If BLK_IDX is 0x2, bit[0] is block#64, bit[31] is block#95 . … If BLK_IDX is 0xFFF, bit[0] is block#131040, bit[31] is block#131071. The maximum value of BLK_IDX is defined by the BLK_MAX register. For each configuration bit, 0 indicates secure, 1 indicates non-secure. A full word write or read to this register automatically increments the BLK_IDX by one if enabled by CTRL[8]. The upper bits are reserved if BLK_SIZE > ADDR_WIDTH - 11.
0x0020	INT_STAT	RO	0x0000_0000	Bits[31:1] – Reserved. Bit[0] – **mpc_irq** triggered.
0x0024	INT_CLEAR	WO	0x0000_0000	Bits[31:1] – Reserved Bit[0] – **mpc_irq** clear (cleared automatically).
0x0028	INT_EN	RW	0x0000_0000	Bits[31:1] - Reserved. Bit[0] – **mpc_irq** enable. Enables interrupt output generation. The INT_STAT, INT_INFO1, and INT_INFO2 registers are still set for errors.
0x002C	INT_INFO1	RO	0x0000_0000	haddr[31:0] of the first security violating address. Bits are valid when **mpc_irq** is triggered. Subsequent security violation transfers remain blocked, that is, not captured in this register and the register retains its value until **mpc_irq** is cleared.

Table 3-5 AHB5 TrustZone MPC registers (continued)

Offset	Name	Type	Reset value	Function
0x0030	INT_INFO2	RO	0x0000_0000	Additional control bits of the first security violating transfer. Bit [31:18] – Reserved. Bit [17] – **cfg_ns**. Bit [16] – **hnonsec**. Bit [15:0] - **hmaster**. Bits are valid when **mpc_irq** is triggered. Subsequent security violating transfers remain blocked, that is, not captured in this register and the register retains its value until mpc_irq is cleared.
0x0034	INT_SET	WO	0x0000_0000	Bit[31:1] – Reserved. Bit[0] – **mpc_irq** set. Debug purpose only. Sets **mpc_irq** triggered in INT_STAT regardless of the **mpc_irq_enable** input.
0x0FD0	PIDR4	RO	0x0000_0004	Peripheral ID 4. Bits[7:4] block count. Bits [3:0] jep106_c_code.
0x0FD4	PIDR5	RO	0x0000_0000	Peripheral ID 5 (not used).
0x0FD8	PIDR6	RO	0x0000_0000	Peripheral ID 6 (not used).
0x0FDC	PIDR7	RO	0x0000_0000	Peripheral ID 7 (not used).
0x0FE0	PIDR0	RO	0x0000_0060	Peripheral ID 0. Bits [31:8] - Reserved. Bits [7:0]. Part number [7:0].
0x0FE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1. Bits[7:4] jep106_id_3_0. Bits[3:0] Part number[11:8]).
0x0FE8	PIDR2	RO	00000_000B	Peripheral ID 2. Bits[7:4] revision. Bit[3] jedec_used Bits[2:0] jep106_id_6_4.
0x0FEC	PIDR3	RO	0x0000_0000	Peripheral ID 3. Bits[7:4] ECO revision number. Bits[3:0] customer modification number.
0x0FF0	CIDR0	RO	0x0000_000D	Component ID 0.
0x0FF4	CIDR1	RO	0x0000_00F0	Component ID 1 (PrimeCell class).
0x0FF8	CIDR2	RO	0x0000_0005	Component ID 2.
0x0FFC	CIDR3	RO	0x0000_00B1	Component ID 3.

Look Up Table (LUT) examples

The contents of the LUT can be accessed in several ways that might require different configurations of the autoincrement function of the BLK_IDX register.

To dump the full contents of the LUT:

1. Set the autoincrement enable bit, CTRL[8], to `0x1`.
2. Read the BLK_MAX register. This has a value 0xN which represents the last address in the LUT.
3. Write `0x0` to the BLK_IDX register.
4. Read the BLK_LUT register to 0xN times to read the complete LUT.

To rewrite the full contents of the LUT:

1. Set autoincrement enable bit, CTRL[8], to `0x1`.
2. Read the BLK_MAX register. This has a value 0xN which represents the last address in the LUT.
3. Write `0x0` to the BLK_IDX register.
4. Write the new values to the BLK_LUT register 0xN times to fill the complete LUT.

To read-modify-write a single location:

1. Set autoincrement enable bit, CTRL[8], to `0x1`.
2. Write the required address to the BLK_IDX.
3. Read the current contents of the LUT.
4. Write the new contents to the LUT.
5. Write the new contents to the LUT.
   - Even byte accesses can be used to update only the required byte of the register without reading the full contents.

Look Up Table (LUT) examples

The contents of the LUT can be accessed in several ways that might require different configurations of the autoincrement function of the BLK_IDX register.

To read the full contents of the LUT:

1. Set the autoincrement enable bit, CTRL[8], to `0x1`.
2. Read the BLK_MAX register. This register has a value 0xN which represents the last address in the LUT.
3. Write `0x0` to the BLK_IDX register.
4. Read the BLK_LUT register to 0xN times to read the complete LUT.

To write the full contents of the LUT:

1. Set autoincrement enable bit, CTRL[8], to `0x1`.
2. Read the BLK_MAX register. This register has a value 0xN which represents the last address in the LUT.
3. Write `0x0` to the BLK_IDX register.
4. Write the new values to the BLK_LUT register 0xN times to fill the complete LUT.

To read-modify-write a single location:

1. Set autoincrement enable bit, CTRL[8], to `0x1`.
2. Write the required address to the BLK_IDX.
3. Read the current contents of the LUT.
4. Write the new contents to the LUT.
5. Write the new contents to the LUT.
   - Even byte accesses can be used to update only the required byte of the register without reading the full contents.

Configuration lockdown

The AHB5 TrustZone memory protection controller provides a configuration lockdown feature that prevents malicious software from changing the security configuration. Writing `0x1` to the security lockdown bit, CTRL[31], enables the configuration lockdown feature.

When the configuration lockdown feature is enabled:

- The lockdown feature can only be disabled by a component reset which resets CTRL[31] to `0x0`.
- The following registers are read-only:
  - CTRL.
  - BLK_LUT.
  - INT_EN.

Note

Arm recommends that you write `0x1` to the LUT autoincrement bit, CTRL[8] before enabling the configuration lockdown feature. When the feature is enabled, only LUT reading is available which is simpler when BLK_IDX increments automatically during the read sequence.

3.3.5 Message Handling Unit Registers

Two Message Handling Units (MHU) enable software to raise interrupts to the processor cores.

MHU 0 base memory addresses are:

- `0x4000_3000` in the Non-secure region.
- `0x5000_3000` in the Secure region.

MHU 1 base memory addresses are:

- `0x4000_4000` in the Non-secure region.
- `0x5000_4000` in the Secure region.

The TrustZone peripheral protection controller controls which area the MHU resides in.

Only 32-bit writes are supported. Byte and halfword writes are ignored.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

The following table shows the MHU0 and MHU1 registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-6 Message Handling Unit (MHU) registers

Offset	Name	Type	Reset value	Description
`0x0000`	CPU0INTR_STAT	RO	`0x0000_0000`	Core 0 interrupt status register.
`0x0004`	CPU0INTR_SET	WO	`0x0000_0000`	Core 0 interrupt set register.
`0x0008`	CPU0INTR_CLR	WO	`0x0000_0000`	Core 0 interrupt clear register.
`0x0010`	CPU1INTR_STAT	RO	`0x0000_0000`	Core 1 interrupt status register.
`0x0014`	CPU1INTR_SET	WO	`0x0000_0000`	Core 1 interrupt set register.
`0x0018`	CPU1INTR_CLR	WO	`0x0000_0000`	Core 1 interrupt clear register.
`0x0FD0`	PIDR4	RO	`0x0000_0000`	Peripheral ID 4.
`0x0FE0`	PIDR0	RW	`0x0000_0000`	Peripheral ID 0.
`0x0FE4`	PIDR1	RO	`0x0000_0000`	Peripheral ID 1.

Table 3-6 Message Handling Unit (MHU) registers (continued)

Offset	Name	Type	Reset value	Description
0x0FE8	PIDR2	RO	0x0000_0000	Peripheral ID 2.
0x0FEC	PIDR3	WO	0x0000_0000	Peripheral ID 3.
0x0FF0	CIDR0	RO	0x0000_0004	Component ID 0.
0x0FF4	CIDR1	RO	0x0000_0000	Component ID 1.
0x0FF8	CIDR2	RO	0x0000_0000	Component ID 2.
0x0FFC	CIDR3	RO	0x0000_0000	Component ID 3.

3.3.6 Secure Privilege Control registers

The Secure Privilege Control Block implements program-visible states that enable software to control security gating units within the design.

The base memory address of the Secure Privilege Control Block is 0x5008_0000.

Writes to the registers must be 32-bits wide. Attempted byte and halfword writes are ignored.

Reads and writes are supported only from secure privileged access.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

The following table shows the registers in address offset order from the base memory address. Undefined registers are reserved or are not implemented in the Musca-A test chip. Software must not attempt to access these registers.

Table 3-7 Secure Privilege Control registers

Offset	Name	Type	Reset value	Function
0x0000	SPCSECCTRL	RW	0x0000_0000	Secure Privilege Controller Secure Configuration Control register.
0x0004	BUSWAIT	RW	0x0000_0001	Bus Access wait control after reset.
0x0010	SECRESPCFG	RW	0x0000_0000	Security Violation Response Configuration register.
0x0014	NSCCFG	RW	0x0000_0000	Non Secure Callable Configuration for IDAU.
0x001C	SECMPCINTSTATUS	RO	0x0000_0000	Secure MPC Interrupt Status. See *SECMPCINTSTATUS Register* on page 3-61 for information on how this register is implemented in the Musca-A test chip.
0x0020	SECPPCINTSTAT	RO	0x0000_0000	Secure PPC Interrupt Status. See *SECPPCINTSTAT Register* on page 3-62 for information on how this register is implemented in the Musca-A test chip.
0x0024	SECPPCINTCLR	RW	0x0000_0000	Secure PPC Interrupt Clear. See *SECPPCINTCLR Register* on page 3-63 for information on how this register is implemented in the Musca-A test chip.
0x0028	SECPPCINTEN	RW	0x0000_0000	Secure PPC Interrupt Enable. See *SECPPCINTEN Register* on page 3-64 for information on how this register is implemented in the Musca-A test chip.

Table 3-7 Secure Privilege Control registers (continued)

Offset	Name	Type	Reset value	Function
0x0040	BRGINTSTAT	RO	0x0000_0000	Bridge Buffer Error Interrupt Status. See *BRGINTSTAT Register* on page 3-64 for information on how this register is implemented in the Musca-A test chip.
0x0044	BRGINTCLR	RW	0x0000_0000	Bridge Buffer Error Interrupt Clear. See *BRGINTCLR Register* on page 3-65 for information on how this register is implemented in the Musca-A test chip.
0x0048	BRGINTEN	RW	0x0000_0000	Bridge Buffer Error Interrupt Enable. See *BRGINTEN Register* on page 3-65 for information on how this register is implemented in the Musca-A test chip.
0x0060	AHBNSPPCEXP0	RW	0x0000_0000	Defines the non-secure access settings for an associated peripheral: See *AHBNSPPCEXP0 Register* on page 3-66 for information on how this register is implemented in the Musca-A test chip.
0x0070	APBNSPPC0	RW	0x0000_0000	Non-Secure Access APB slave Peripheral Protection Control #0. This register controls the PPC within the Base element.
0x0074	APBNSPPC1	RW	0x0000_0000	Non-Secure Access APB slave Peripheral Protection Control #1. This register controls the PPC within the System Control element.
0x00A0	AHBSPPPCEXP0	RW	0x0000_0000	Defines the Secure unprivileged access setting for associated peripherals: See *AHBSPPPCEXP0 Register* on page 3-66 for information on how this register is implemented in the Musca-A test chip.
0x00B0	APBSPPPC0	RW	0x0000_0000	Secure Unprivileged Access APB slave Peripheral. Protection Control #0. This register controls the PPC within the Base element.
0x00B4	APBSPPPC1	RW	0x0000_0000	Secure Unprivileged Access APB slave Peripheral. Protection Control #1. This register controls the PPC within the System Control element.
0x0FD0	PID4	RO	0x0000_0004	Peripheral ID 4
0x0FE0	PID0	RO	0x0000_0052	Peripheral ID 0
0x0FE4	PID1	RO	0x0000_00B8	Peripheral ID 1
0x0FE8	PID2	RO	0x0000_000B	Peripheral ID 2
0x0FEC	PID3	RO	0x0000_0000	Peripheral ID 3
0x0FF0	CID0	RO	0x0000_000D	Component ID 0
0x0FF4	CID1	RO	0x0000_00F0	Component ID 1
0x0FF8	CID2	RO	0x0000_0005	Component ID 2
0x0FFC	CID3	RO	0x0000_00B1	Component ID 3

SECMPCINTSTATUS Register

The SECMPCINTSTATUS Register characteristics are:

Purpose

Stores the interrupt status of *Memory Protection Controllers* (MPCs).

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

This register is read-only.

The following table shows the bit assignments.

Table 3-8 SECMPCINTSTATUS Register bit assignments

Bits	Name	Function
[31:18]	-	Reserved.
[17]	S_MPCEXP_STATUS[1]	Interrupt Status of SRAM Memory Protection Controller. Reset value `0b0`.
[16]	S_MPCEXP_STATUS[0]	Interrupt Status of QSPI Memory Protection Controller. Reset value `0b0`.
[15:4]	-	Reserved.
[3]	S_MPCSRAM3_STATUS	Interrupt Status of SRAM bank 3 Memory Protection Controller. Reset value `0b0`.
[2]	S_MPCSRAM2_STATUS	Interrupt Status of SRAM bank 2 Memory Protection Controller. Reset value `0b0`.
[1]	S_MPCSRAM1_STATUS	Interrupt Status of SRAM bank 1 Memory Protection Controller. Reset value `0b0`.
[0]	S_MPCSRAM0_STATUS	Interrupt Status of SRAM bank 0 Memory Protection Controller. Reset value `0b0`.

SECPPCINTSTAT Register

The Secure Peripheral Protection Controller interrupt status (SECPPCINTSTAT) Register characteristics are:

Purpose

Stores the interrupt statuses of *Peripheral Protection Controllers* (PPCs).

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

This register is read-only.

The following table shows the bit assignments.

Table 3-9 SECPPCINTSTAT Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved.
[20]	S_AHBPPCEXP_STATUS	Interrupt status of AHB Peripheral Protection Controller. Reset value `0b0`.
[19:2]	-	Reserved.

Table 3-9 SECPPCINTSTAT Register bit assignments (continued)

Bits	Name	Function
[1]	S_APBPPC1PERIP_STATUS	Interrupt status of Peripheral Protection Controller for APB slaves within the System Control Element. Reset value `0b0`.
[0]	S_APBPPC0PERIP_STATUS	Interrupt status of Peripheral Protection Controller for APB slaves within the Base Element. Reset value `0b0`.

SECPPCINTCLR Register

The Secure Peripheral Protection Controller Interrupt Clear (SECPPCINTCLR) Register characteristics are:

Purpose

Enables software to clear the *Peripheral Protection Controllers* (PPC) interrupts.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

This register is W1S access.

The following table shows the bit assignments.

Table 3-10 SECPPCINTCLR Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved.
[20]	S_AHBPPCEXP_CLR	Clear AHB Peripheral Protection Controller interrupt. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b0`.
[19:2]	-	Reserved.
[1]	S_APBPPC1PERIP_CLR	Clear interrupt of Peripheral Protection Controller for APB slaves within the System Control Element. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b0`.
[0]	S_APBPPC0PERIP_CLR	Clear interrupt of Peripheral Protection Controller for APB slaves within the Base Element. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b0`.

SECPPCINTEN Register

The Secure Peripheral Protection Controller Interrupt enable (SECPPCINTEN) Register characteristics are:

Purpose

Enables or disables the *Peripheral Protection Controllers* (PPC) interrupts.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments.

Table 3-11 SECPPCINTEN Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved.
[20]	S_AHBPPCEXP_EN	Enables or disables AHB Peripheral Protection Controller interrupt: `0b0`: Disable interrupt. `0b1`: Enable interrupt. Reset value `0b0`.
[19:2]	-	Reserved.
[1]	S_APBPPC1PERIP_EN	Enables or disables interrupt of Peripheral Protection Controller for APB slaves within the System Control Element. `0b0`: Disable interrupt. `0b1`: Enable interrupt. Reset value `0b0`.
[0]	S_APBPPC0PERIP_EN	Enables or disables interrupt of Peripheral Protection Controller for APB slaves within the Base Element. `0b0`: Disable interrupt. `0b1`: Enable interrupt. Reset value `0b0`.

BRGINTSTAT Register

The Bridge Buffer Error Interrupt Status (BRGINTSTAT) Register characteristics are:

Purpose

Stores the interrupt status of the bridge between CPU1 and the system.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

This register is read-only.

The following table shows the bit assignments.

Table 3-12 BRGINTSTAT Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	BRG_CPU1SYS_STATUS	Interrupt status of the bridge between CPU1 and the system. Reset value `0b0`.

BRGINTCLR Register

The Bridge Buffer Error Interrupt Clear (BRGINTCLR) Register characteristics are:

Purpose

Clears interrupt of the bridge between CPU1 and the system.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

This register is W1S access.

The following table shows the bit assignments.

Table 3-13 BRGINTCLR Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	BRG_CPU1SYS_CLR	Clears interrupt of the bridge between CPU1 and the system. Reset value `0b0`.

BRGINTEN Register

The Bridge Buffer Error Interrupt Enable (BRGINTEN) Register characteristics are:

Purpose

Enables or disables the interrupt of the bridge between CPU1 and the system.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments.

Table 3-14 BRGINTEN Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	BRG_CPU1SYS_EN	Enables or disables the interrupt of the bridge between CPU1 and the system: `0b0`: Disable interrupt. `0b1`: Enable interrupt. Reset value `0b0`.

AHBNSPPCEXP0 Register

The Expansion Non-secure Access AHB Slave Peripheral Protection Controller (AHBNSPPCEXP0) Register 0 characteristics are:

Purpose

Defines the Secure or Non-Secure access setting for the associated AHB slave Peripheral Protection Controller outside the SSE-200 subsystem.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments.

Table 3-15 AHBNSPPCEXP0 Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	AHBNSPPCEXP0	Defines the Secure or Non-Secure access setting for the AHB slave Peripheral Protection Controller outside the SSE-200 subsystem: `0b0`: Enable Secure access only. `0b1`: Enable Non-secure access only. Reset value `0b0`.

AHBSPPPCEXP0 Register

The Expansion Secure Privilege Access AHB Slave Peripheral Protection Controller (AHBSPPPCEXP0) Register 0 characteristics are:

Purpose

Defines the Secure access setting for the associated AHB slave Peripheral Protection Controller outside the SSE-200 subsystem.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments.

Table 3-16 AHBSPPPCEXP0 Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	AHBSPPPCEXP0	Defines the Secure or Non-Secure access setting for the AHB slave Peripheral Protection Controller outside the SSE-200 subsystem: `0b0`: Enable Secure privileged access only. `0b1`: Enable Secure unprivileged and privileged access. Reset value `0b0`.

3.3.7 Non-secure Privilege Control Block

The Non-secure Privilege Control Block implements program visible states that enable software to control various security gating units within the design.

The Non-secure Privilege Control block register base memory address is `0x4008_0000`.

These registers are Non-secure privileged access only and support 32-bit R/W access. Attempted byte and halfword writes are ignored.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

The following table shows the registers in address offset order from the base memory address. Undefined registers are reserved or are not implemented in the Musca-A test chip. Software must not attempt to access these registers.

Table 3-17 Non-secure Privilege Control registers

Offset	Name	Type	Reset value	Function
`0x00A0`	AHBNSPPPCEXP0	RW	`0x0000_0000`	Expansion 0 Non-Secure Unprivileged Access AHB slave Peripheral Protection Control. Bit[0] in this register defines the Non-secure unprivileged access settings for AHB peripherals. See *AHBNSPPPCEXP0 Register* on page 3-67 for information on how this register is implemented in the Musca-A test chip.
`0x00B0`	APBNSPPPC0	RW	`0x0000_0000`	Non-Secure Unprivileged Access APB slave Peripheral Protection Control #0. Each bit in this register defines the Non-secure unprivileged access settings for an associated peripheral. See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for the associated peripherals.
`0x00B4`	APBNSPPPC1	RW	`0x0000_0000`	Non-Secure Unprivileged Access APB slave Peripheral Protection Control #1. Each bit in this register defines the Non-secure unprivileged access settings for an associated peripheral. See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for the associated peripherals.
`0x0FD0`	PIDR4	RO	`0x0000_0004`	Peripheral ID 4
`0x0FE0`	PIDR0	RO	`0x0000_0053`	Peripheral ID 0
`0x0FE4`	PIDR1	RO	`0x0000_00B8`	Peripheral ID 1
`0x0FE8`	PIDR2	RO	`0x0000_000B`	Peripheral ID 2
`0x0FEC`	PIDR3	RO	`0x0000_0000`	Peripheral ID 3
`0x0FF0`	CIDR0	RO	`0x0000_000D`	Component ID 0
`0x0FF4`	CIDR1	RO	`0x0000_00F0`	Component ID 1
`0x0FF8`	CIDR2	RO	`0x0000_0005`	Component ID 2
`0x0FFC`	CIDR3	RO	`0x0000_00B1`	Component ID 3

AHBNSPPPCEXP0 Register

The Expansion 0 Non-secure Unprivileged Access AHB Slave Peripheral Protection Controller (AHBNSPPPCEXP0) Register 0 characteristics are:

Purpose

Defines the Non-secure unprivileged access setting for the AHB slave Peripheral Protection Controller outside the SSE-200 subsystem.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for more information.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments.

Table 3-18 AHBNSPPPCEXP0 Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	AHBNSPPPCEXP0	Defines the Non-Secure Unprivileged Access setting for the AHB slave Peripheral Protection Controller outside the SSE-200 subsystem: `0b0`: Enable Non-secure unprivileged and privileged access. `0b1`: Enable Non-secure privileged access only. Reset value `0b0`.

3.4 CPU elements

The SSE-200 block in the test chip implements two CPU elements. Each CPU element contains a Cortex-M33 core.

CPU0 is the main processor. It is a Cortex-M33 with no FPU, no DSP, and no coprocessor. The maximum clock frequency is 50MHz.

CPU1 is a Cortex-M33 with FPU, DSP, and no coprocessor. The maximum clock frequency is 170MHz.

This section contains the following subsections:

3.4.1 Processor L1 cache registers

The L1 instruction cache for each processor has the following characteristics:

- Cache size 2KB.
- 32-bit AHB5 interface.
- Only read accesses are subject to caching. Security attributes are stored with the cache entry.
- Separate AHB5 interface provides access to the cache control registers.
- 2-way set associative.
- 16-byte cache lines.
- Both secure and Non-secure masters can access the caches. The AHB5 interface does not perform security operations.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

The following table shows the L1 cache registers. Undefined registers are reserved. Software must not attempt to access these registers. All registers are Secure privileged access only.

Table 3-19 Processor L1 cache registers summary

Offset	Name	Type	Reset	Width	Description
0x0000	ICHWPARAMS	RO	0x0000_0000	32	Hardware Parameter Register.
0x0004	ICCTRL	RW	0x0000_0000	32	Instruction Cache Control Register.
0x0100	ICIRQSTAT	RO	0x0000_0000	32	Interrupt Request Status Register.
0x0104	ICHRQSCLR	WO	0x0000_0000	32	Interrupt Status Clear Register.
0x0108	ICIRQEN	RW	0x0000_0000	32	Interrupt Enable Register.
0x010C	ICDBGFILLERR	RO	0x0000_0000	32	Debug Fill Error Register.
0x0300	ICSHR	RO	0x0000_0000	32	Instruction Cache Statistic Hit Register.
0x0304	ICSMR	RO	0x0000_0000	32	Instruction Cache Statistic Miss Count Register.
0x0308	ICSUC	RO	0x0000_0000	32	Instruction Cache Statistic Uncached Count Register.
0x0FD0	PIDR4	RO	0x0000_0004	32	Product ID Register 4.
0x0FE4	PIDR1	RO	0x0000_00B8	32	Product ID Register 1.

Table 3-19 Processor L1 cache registers summary (continued)

Offset	Name	Type	Reset	Width	Description
0x0FE8	PIDR2	RO	0x0000_000B	32	Product ID Register 2.
0x0FEC	PIDR3	RO	0x0000_0000	32	Product ID Register 3.
0x0FF0	CIDR0	RO	0x0000_000D	32	Component ID Register 0.
0x0FF4	CIDR1	RO	0x0000_00F0	32	Component ID Register 1.
0x0FF8	CIDR2	RO	0x0000_0005	32	Component ID Register 2.
0x0FFC	CIDR3	RO	0x0000_00B1	32	Component ID Register 3.

3.4.2 Processor L1 cache programming

Certain practices and techniques are recommended when programming the L1 cache in the test chip.

Initialization

After powerup or reset, the cache powers up in a disabled state and begins the invalidation process. Accesses arriving at the cache are not cached and bypass the cache. The cache can be enabled during the invalidation process by setting the CACHEEN control bit in the Instruction Cache Control Register, ICCTRL, to 0b1. However, all accesses are still treated as uncached and bypass the cache until the cache invalidation process completes.

After the end of the cache invalidation process, the interrupt status signal, IC, in the Interrupt Request Status register, ICIRQSTAT, is asserted. If that interrupt is already enabled or is enabled later, an interrupt is raised. To enable caching of code fetches, you can poll this status register, or wait for this interrupt to be raised before continuing code execution.

Cache disable

The cache can be disabled by clearing the CACHEEN control bit in the Instruction Cache Control Register, ICCTRL. Outstanding accesses are completed before the cache is disabled. Software can read the CDC bit in ICIRQSTAT register, or enable the CDC interrupt and wait for the interrupt to arrive, after clearing the CACHEEN bit.

Cache invalidation

You can invalidate the cache by setting the partial invalidate bit, PINV, or the full invalidate bit, FINV, in the ICCTRL register. Because the cache does not support Locked Lines, setting either of these bits initiates a full cache invalidation. During cache invalidation, all accesses through the cache are treated as uncached and bypass the cache until the invalidation process completes. At the end of the invalidation process the interrupt status, IC, is asserted. If that interrupt is already enabled or is enabled later, an interrupt is raised.

Performance targets

The cache improves the average performance of the connected processor by holding local copies of previously accessed or specified memory locations. The improvement cannot be determined precisely because many design parameters, code behavior, and system considerations affect the performance.

The cache can reduce processor performance in the following events:

- Uncacheable memory:
  - The cache adds a cycle of latency to the transaction.
- Writes:
  - Writes are treated as uncacheable, and create an extra cycle of bus latency.
- Cache misses:
  - A cache miss causes a fetch to occur, and causes an extra cycle of bus latency for the initial data. Subsequent transactions are also stalled while the rest of the fetch process occurs. The time that is taken for the memory subsystem to return the rest of the WRAP4 transaction determines the extra latency.

3.4.3 Ensuring the cache handles memory modifications

The instruction cache does not support coherency between an external code location and a corresponding code line that is already in the cache.

The software must invalidate the cache to modify the external location. Non-coherency between cached lines in the cache and the lines in the external code memory is a security issue.

To invalidate the cache, do the following:

1. Disable the instruction cache.
2. Manually invalidate the full instruction cache.
3. Modify the code space content.
4. Enable the instruction cache.

Cache misses occur when a modification to the Secure Access Unit, or the Memory Protection Controller, changes the security setting of a recently cached memory region. The instruction cache retains the old security attribute and disables hits on the cached line using the new security attribute. This situation can result in Secure and Non-secure versions of the same memory location residing in the cache, reducing its efficiency. If the older cached line, not intended to be available in that system, is accessed with the original access attribute, the presence of the two versions poses a security risk.

——— **Note** ———

Arm recommends that you invalidate the cache to remove this security risk. To invalidate the cache, do the following:

1. Disable the instruction cache.
2. Manually invalidate the instruction cache.
3. Reprogram and reconfigure the code area contents and security behavior.
4. Enable the instruction cache.

———————————

3.4.4 Interrupts

The test chip implements an Arm Nested Vector Interrupt Controller (NVIC) and an Arm Wakeup Interrupt Controller (WIC).

The NVIC supports:

- A programmable priority level of 0-255 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-Maskable Interrupt (NMI).

Interrupt signals from SSE-200 blocks

The following table shows the interrupt signals and exceptions to the two processor cores from blocks in the SSE-200.

Table 3-20 SSE-200 interrupt signals

Interrupt input	CPU0 interrupt source	CPU1 interrupt source
NMI	Combined SECURE WATCHDOG, S32KWATCHDOG, and NMI_Expansion	Combined SECURE WATCHDOG, S32KWATCHDOG, and NMI_Expansion.
IRQ[0]	NON-SECURE WATCHDOG Reset Request	NON-SECURE WATCHDOG Reset Request.
IRQ[1]	NON-SECURE WATCHDOG Interrupt	NON-SECURE WATCHDOG Interrupt.
IRQ[2]	S32K Timer	S32K Timer.
IRQ[3]	TIMER 0	TIMER 0.
IRQ[4]	TIMER 1	TIMER 1.
IRQ[5]	DUAL TIMER	DUAL TIMER.
IRQ[6]	Message Handling Unit 0 CPU0 Interrupt	Message Handling Unit 0 CPU1 Interrupt.
IRQ[7]	Message Handling Unit 1 CPU0 Interrupt	Message Handling Unit 1 CPU1 Interrupt.
IRQ[8]	Reserved	Reserved.
IRQ[9]	MPC Combined (Secure)	MPC Combined (Secure).
IRQ[10]	PPC Combined (Secure)	PPC Combined (Secure).
IRQ[11]	MSC Combined (Secure)	MSC Combined (Secure).
IRQ[12	Bridge Error Combined Interrupt (Secure)	Bridge Error Combined Interrupt (Secure).
IRQ[13]	CPU0 Instruction Cache Invalidation Interrupt	CPU0 Instruction Cache Invalidation Interrupt.
IRQ[14]	Reserved	Reserved.
IRQ[15]	SYS_PPU	SYS_PPU.
IRQ[16]	CPU0_PPU	CPU0_PPU.
IRQ[17]	CPU1_PPU	CPU1_PPU.
IRQ[18]	CPU0DBG_PPU	CPU0DBG_PPU.
IRQ[19]	CPU1DBG_PPU	CPU1DBG_PPU.
IRQ[20]	Reserved	Reserved.
IRQ[21]	Reserved	Reserved.
IRQ[22]	RAM0_PPU	RAM0_PPU.
IRQ[23]	RAM1_PPU	RAM1_PPU.
IRQ[24]	RAM2_PPU	RAM2_PPU.
IRQ[25]	RAM3_PPU	RAM3_PPU.
IRQ[26]	DBG_PPU	DBG_PPU.
IRQ[27]	Reserved	Reserved.
IRQ[28]	CPU0CTIIRQ0	CPU1CTIIRQ0.

Table 3-20 SSE-200 interrupt signals (continued)

Interrupt input	CPU0 interrupt source	CPU1 interrupt source
IRQ[29]	CPU0CTIIRQ1	CPU1CTIIRQ1.
IRQ[31:30]	Reserved	Reserved.

The following table shows the expansion interrupt signals, that is, from test chip blocks outside the SSE-200.

Table 3-21 Expansion interrupt signals from blocks outside the SSE-200

Interrupt input	CPU0 and CPU1 interrupt source	Description
IRQ[32]	Reserved	Reserved
IRQ[33]	GPTIMERINTR	General Purpose Timer Interrupt.
IRQ[34]	I2C0INTR	I^2C0 interrupt.
IRQ[35]	I2C1INTR	I^2C1 interrupt.
IRQ[36]	I2SINTR	I^2S interrupt.
IRQ[37]	SPIINTR	SPI Interrupt.
IRQ[38]	QSPIINTR	QSPI Interrupt.
IRQ[39]	UARTRXINTR0	UART0 receive FIFO interrupt, active HIGH.
IRQ[40]	UARTTXINTR0	UART transmit FIFO interrupt, active HIGH.
IRQ[41]	UARTRTINTR0	UART0 receive timeout interrupt, active HIGH.
IRQ[42]	UARTMSINTR0	UART0 modem status interrupt, active HIGH.
IRQ[43]	UARTEINTR0	UART0 error interrupt, active HIGH.
IRQ[44]	UARTINTR0	UART0 interrupt, active HIGH.
IRQ[45	UARTRXINTR1	UART1 receive FIFO interrupt, active HIGH.
IRQ[46]	UARTTXINTR1	UART1 transmit FIFO interrupt, active HIGH.
IRQ[47]	UARTRTINTR1	UART1 receive timeout interrupt, active HIGH.
IRQ[48]	UARTMSINTR1	UART1 modem status interrupt, active HIGH.
IRQ[49]	UARTEINTR1	UART1 error interrupt, active HIGH.
IRQ[50]	UARTINTR1	UART1 interrupt, active HIGH.
IRQ[66:51]	GPIOINT[15:0]	GPIO interrupts.
IRQ[67]	COMBINT	Combined Interrupt.
IRQ[68]	-	Reserved.
IRQ[69]	-	Reserved.
IRQ[70]	PWMINT0	PWM0 Interrupt.
IRQ[71]	RTCINT	RTC Interrupt.
IRQ[72]	GPTIMERINT1	General Purpose Timer Interrupt[1] (Comparator 1).
IRQ[73]	GPTIMERINT0	General Purpose Timer Interrupt[0] (Comparator 0).
IRQ[74]	PWMINT1	PWM1 Interrupt.

Table 3-21 Expansion interrupt signals from blocks outside the SSE-200 (continued)

Interrupt input	CPU0 and CPU1 interrupt source	Description
IRQ[75]	PWMINT2	PWM2 Interrupt.
IRQ[76]	IOMUX_INTR	IOMUX Interrupt.

Interrupt controller registers

The following table shows the interrupt controller registers.

Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-22 Summary of interrupt controller registers

Address	Name	Type	Reset value	Description
0xE000E004	ICTR	RO	-	Interrupt Controller Type Register.
0xE000_E100-0xE000_E11C	NVIC_ISER0-NVIC_ISER7	RW	0x0000_0000	Interrupt Set Enable Registers.
0xE000_E180-0xE000_E19C	NVIC_ICER0-NVIC_ICER7	RW	0x0000_0000	Interrupt Clear Enable Registers.
0xE000_E200-0xE000_E21C	NVIC_ISPR0-NVIC_ISPR7	RW	0x0000_0000	Interrupt Set Pending Registers.
0xE000_E280-0xE000_E29C	NVIC_ICPR0-NVIC_ICPR7	RW	0x0000_0000	Interrupt Clear Pending Registers.
0xE000_E300-0xE000_E31C	NVIC_IABR0-NVIC_IABR7	RO	0x0000_0000	Interrupt Active Bit Registers.
0xE000_E400-0xE000_E41F	NVIC_IPRO-NVIC_IPR7	RW	0x0000_0000	Interrupt Priority Registers.

See the following documentation for more information on the interrupt controller:
- *Arm® Cortex®-M33 Processor Technical Reference Manual*.
- *Arm® v7-M Architecture Reference Manual*.

3.4.5 Processor core interrupt registers

The SSE-200 block in the test chip implements CPU0 and CPU1 cores interrupt registers. The interrupt registers enable software to raise interrupts, clear interrupts, and check the written value that is used to raise the interrupts to the cores.

Set and Clear registers support setting and clearing of individual which means the individual bits can represent events that can be independently set and cleared.

The CPU0 and CPU1 interrupt registers are:

- CPU0INTR_STAT - Core 0 interrupt status register.
- CPU0INTR_SET - Core 0 interrupt set register.
- CPU0INTR_CLR - Core 0 interrupt clear register.
- CPU1INTR_STAT - Core 1 interrupt status register.
- CPU1INTR_SET - Core 1 interrupt set register.
- CPU1INTR_CLR - Core 1 interrupt clear register.

See the following for more information on the CPU0 and CPU1 interrupt registers.
- *3.3.5 Message Handling Unit Registers* on page 3-59.
- *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

3.4.6 Wakeup Interrupt Controller (WIC)

The Wakeup Interrupt Controller is a peripheral that detects an interrupt signal and can wake the processor from deep sleep mode. The WIC is active only when the system is in deep sleep mode.

The WIC is not programmable and does not have registers or a user interface. It operates entirely under the control of hardware signals.

When the WIC is enabled and the processor is in deep sleep mode, the power management unit in the system can power down most of the processor. When the WIC receives an interrupt, it takes several clock cycles to wake up the processor to a state where it can service the interrupt. Latency is increased in deep sleep mode.

Note

The IoT System uses latches to implement the WIC, unlike in the standard Cortex-M33 processor. FCLK can be gated completely during WIC-based sleep which is not a standard Cortex-M33 processor feature.

3.5 System control element

This section describes the registers that are associated with controlling the SSE-200 block in the test chip.

This section contains the following subsections:

3.5.1 System control regions

The System Control Region contains the peripherals in the System Control element.

The System Control Region occupies the following areas:

- `0x4002_0000` to `0x4003_FFFF`, which is Non-secure.
- `0x5002_0000` to `0x5003_FFFF`, which is Secure.

Table 3-23 System control regions

Row ID (alias)	Address		Size	Region name	Description	Security
	From	To				
1 (5)	`0x4002_0000`	`0x4002_0FFF`	4KB	SYSINFO	System Information Registers Block.	NS
2	`0x4002_1000`	`0x4002_EFFF`		Reserved	Reserved[a]	
3 (18)	`0x4002_F000`	`0x4002_FFFF`	4KB	S32KTIMER	CMSDK Timer running on **S32KCLK**.	NS-PPC
4	`0x4003_0000`	`0x4003_FFFF`		Reserved	Reserved	
5 (1)	`0x5002_0000`	`0x5002_0FFF`	4KB	SYSINFO	System Information Registers Block.	S
6	`0x5002_1000`	`0x5002_1FFF`	4KB	S_SYSCONTROL	System Control Registers Block.	SP
7	`0x5002_2000`	`0x5002_2FFF`	4KB	SYS_PPU	System Power Policy Unit.	SP
8	`0x5002_3000`	`0x5002_3FFF`	4KB	CPU0CORE_PPU	CPU0 Core Power Policy Unit.	SP
9	`0x5002_4000`	`0x5002_4FFF`	4KB	CPU0DBG_PPU[b]	CPU0 Debug Power Policy Unit.	SP
10	`0x5002_5000`	`0x5002_5FFF`	4KB	CPU1CORE_PPU	CPU1 Core Power Policy Unit.	SP
11	`0x5002_6000`	`0x5002_6FFF`	4KB	CPU1DBG_PPU	CPU1 Debug Power Policy Unit.	SP
	`0x5002_7000`	`0x5002_7FFF`	4KB	CRYPTO_PPU	CryptoCell Power Policy Unit.	SP
	`0x5002_8000`	`0x5002_8FFF`	4KB	Reserved	Reserved[c]	
12	`0x5002_9000`	`0x5002_9FFF`	4KB	DEBUG_PPU	System Debug Power Policy Unit.	SP
13	`0x5002_A000`	`0x5002_AFFF`	4KB	RAM0_PPU	SRAM Bank 0 Power Policy Unit.	SP
14	`0x5002_B000`	`0x5002_BFFF`	4KB	RAM1_PPU	SRAM Bank 1 Power Policy Unit.	SP
15	`0x5002_C000`	`0x5002_CFFF`	4KB	RAM2_PPU	SRAM Bank 2 Power Policy Unit.	SP
16	`0x5002_D000`	`0x5002_DFFF`	4KB	RAM3_PPU	SRAM Bank 3 Power Policy Unit.	SP

a This region is RZZ/WI.
b CPU0DBG_PPU and CPU1DBG_PPU regions do not exist if SEPARATE_CPUDBG_PD configuration is False, indicating that separate CPU debug power domains is not supported.
c This region is RAZ/WI.

Table 3-23 System control regions (continued)

Row ID (alias)	Address		Size	Region name	Description	Security
	From	To				
17	0x5002_E000	0x5002_EFFF	4KB	S32KWATCHDOG	CMSDK Watchdog on **S32KCLK**.	SP
18 (3)	0x5002_F000	0x5002_FFFF	4KB	S32KTIMER	CMSDK Timer on **S32KCLK**.	S-PPC
19	0x5003_0000	0x5003_FFFF		Reserved	Reserved.	

Note

- For NS_PPC, any Secure access targeting these regions is blocked. Non-secure access to these regions is controlled by PPCs.
- For S_PPC, any Secure access targeting this region is blocked. Secure access to this region is controlled by PPCs.
- NSP indicates Non-secure private access only.
- SP indicates Secure privilege access only.
- S indicates Secure access only.
- Reserved regions respond with RAZWI when accessed.

3.5.2 System information registers

The System Information Register Block provides information on the system configuration and identity. This register block is read-only and accessible by accesses of any security attributes.

The base memory addresses of the System Information Register block are:

- 0x4002_0000 in the Non-secure region.
- 0x5002_0000 in the Secure region.

Note

The System information registers block is visible to both regions without any security protection.

The following table shows the System information registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-24 System Information Block registers summary

Offset	Name	Access	Reset value	Description	Security
0x000	SYS_VERSION	RO	0x2004_1743	System Version register	All
0x004	SYS_CONFIG	RO	System configuration dependent	System Hardware Configuration register	All
0x010 – 0xFCC	Reserved	-	-	-	-
0xFD0	PIDR4	RO	0x0000_0004	Peripheral ID 4	All
0xFD4	PIDR5	RO	0x0	Reserved	-
0xFD8	PIDR6	RO	0x0	Reserved	-
0xFDC	PIDR7	RO	0x0	Reserved	-
0xFE0	PIDR0	RO	0x0000_0058	Peripheral ID 0	All
0xFE4	PIDR1	RO	0x0000_00B8	Peripheral ID 1	All
0xFE8	PIDR2	RO	0x0000_000B	Peripheral ID 2	All

Table 3-24 System Information Block registers summary (continued)

Offset	Name	Access	Reset value	Description	Security
0xFEC	PIDR3	RO	0x0000_0000	Peripheral ID 3	All
0xFF0	CIDR0	RO	0x0000_000D	Component ID 0	All
0xFF4	CIDR1	RO	0x0000_00F0	Component ID 1	All
0xFF8	CIDR2	RO	0x0000_0005	Component ID 2	All
0xFFC	CIDR3	RO	0x0000_00B1	Component ID 3	All

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual* for information about the SYS_VERSION and SYS_CONFIG registers.

3.5.3 System control register block

The System Control Register Block implements registers for power, clocks, resets, and other general system control.

The base memory address of the System Control Register block is 0x5002_1000 in the Secure region of the base peripheral region.

The System Control registers are secure privilege access only and support only 32-bit writes. Attempted byte and halfword writes are ignored.

See the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

The following table shows the System Control registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-25 System control register block registers summary

Offset	Name	Access	Reset value	Description
0x0000	SECDBGSTAT	RO	0x0000_0000	Secure Debug Configuration Status.
0x0004	SECDBGSET	WO	0x0000_0000	Secure Debug Configuration Set.
0x0008	SECDBGCLR	WO	0x0000_0000	Secure Debug Configuration Clear.
0x000C	SCSECCTRL	RW	0x0000_0000	System Security Control.
0x0010	FCLK_DIV	RW	0x0000_0000	Fast Clock Divider Configuration. See *FCLK_DIV Register* on page 3-80.
0x0014	SYSCLK_DIV	RW	0x0000_0000	System Clock Divider Configuration. See *SYSCLK_DIV Register* on page 3-81.
0x0018	CLOCK_FORCE	RW	0x0000_0000	Clock Force.
0x0100	RESET_SYNDROME	RW	0x0000_0001	Reset syndrome. Register only cleared at Powerup Reset.
0x0104	RESET_MASK	RW	0x0000_0000	Reset mask.
0x0108	SWRESET	WO	0x0000_0000	Software Reset.
0x010C	GRETREG	RW	0x0000_0000	General-Purpose retention.
0x0110	INITSVRTOR0	RW	0x0020_0000	Initial Secure Reset Vector Register for CPU0.

Table 3-25 System control register block registers summary (continued)

Offset	Name	Access	Reset value	Description
0x0114	INITSVRTOR1	RW	0x0020_0000	Initial Secure Reset Vector Register for CPU1.
0x0118	CPUWAIT	RW	0x0000_0010	CPU Boot wait control after reset.
0x011C	NMI_ENABLE	RW	0x0000_0000	NAMI Enable Register.
0x0120	WICCTRL	RW	0x0000_0000	WIC request and acknowledge handshake.
0x0124	EWCTRL	RW	0x0000_0000	External Wakeup Control.
0x0200	PDCM_PD_SYS_SENSE	RW	0x0000_007F	Power Control Dependency Matrix. PD_SYS Power Domain Sensitivity.
0x0204	PDCM_PD_CPU0CORE_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_CPU0CORE Power Domain Sensitivity.
0x0208	PDCM_PD_CPU1CORE_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_CPU1CORE Power Domain Sensitivity.
0x020C	PDCM_PD_SRAM0_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_SRAM0 Power Domain Sensitivity.
0x0210	PDCM_PD_SRAM1_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_SRAM1 Power Domain Sensitivity.
0x0214	PDCM_PD_SRAM2_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_SRAM2 Power Domain Sensitivity.
0x0218	PDCM_PD_SRAM3_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_SRAM3 Power Domain Sensitivity.
0x0230	PDCM_PD_CC_SENSE	RO	0x0000_0000	Power Control Dependency Matrix. PD_CC Power Domain Sensitivity.
0x0240	PDCM_PD_EXP0_OUT_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_EXP0_OUT Sensitivity. Drives the PD_EXP_OUT[0] signal.
0x0244	PDCM_PD_EXP1_OUT_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_EXP1_OUT Sensitivity. Drives the PD_EXP_OUT[1] signal.
0x0248	PDCM_PD_EXP2_OUT_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_EXP2_OUT Sensitivity. Drives the PD_EXP_OUT[2] signal.

Table 3-25 System control register block registers summary (continued)

Offset	Name	Access	Reset value	Description
0x024C	PDCM_PD_EXP3_OUT_SENSE	RW	0x0000_0000	Power Control Dependency Matrix. PD_EXP3_OUT Sensitivity. Drives the PD_EXP_OUT[3] signal.
0x0FD0	PIDR4	RO	0x0000_0004	Peripheral ID4
0x0FE0	PIDR0	RO	0x0000_0054	Peripheral ID0
0x0FE4	PIDR1	RO	0x0000_00B8	Peripheral ID1
0x0FE8	PIDR2	RO	0x0000_000B	Peripheral ID2
0x0FEC	PIDR3	RO	0x0000_0000	Peripheral ID3
0x0FF0	CIDR0	RO	0x0000_000D	Component ID0
0x0FF4	CIDR1	RO	0x0000_00F0	Component ID1
0x0FF8	CIDR2	RO	0x0000_0005	Component ID2
0x0FFC	CIDR3	RO	0x0000_00B1	Component ID3

FCLK_DIV Register

The FCLK_DIV Register characteristics are:

Purpose

Controls the divider value of clock divider FCLKDIV that derives **FCLK**, in the SSE-200 subsystem, from **MAINCLK** in the Musca-A test chip. **FCLK** drives the secondary processor element, CPU1.

Usage constraints

Bits[20:16] are read-only. Bits[4:0] are read-write. The other bits are Reserved.

Memory offset and full register reset value

See *3.5.3 System control register block* on page 3-78.

The following table shows the bit assignments.

Table 3-26 FCLK_DIV Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved.
[20:16]	FCLKDIV_CUR	Current value of FCLKDIV. The division value of FCLKDIV divider is FCLKDIV_CUR+1. These bits are read-only. Reset value 0b00000.

Table 3-26 FCLK_DIV Register bit assignments (continued)

Bits	Name	Function
[15:5]	-	Reserved.
[4:0]	FCLKDIV	Controls FCLKDIV divide value in SSE-200 subsystem. Division value = FCLKDIV+1. These bits are read-write. Reset value `0b00000`, no division.

SYSCLK_DIV Register

The SYSCLK_DIV Register characteristics are:

Purpose

Controls the divider value of clock divider SYSCLKDIV that derives **SYSCLK** from **FCLK** in the Musca-A test chip. **SYSCLK** drives the primary processor element, CPU0.

Usage constraints

Bits[20:16] are read-only. Bits[4:0] are read-write. The other bits are Reserved.

Memory offset and full register reset value

See *3.5.3 System control register block* on page 3-78.

The following table shows the bit assignments.

Table 3-27 SYSCLK_DIV Register bit assignments

Bits	Name	Function
[31:21]	-	Reserved.
[20:16]	SYSCLKDIV_CUR	Current value of SYSCLKDIV. The division value of SYSCLKDIV divider is SYSCLKDIV_CUR+1. These bits are read-only. Reset value `0b00000`.
[15:5]	-	Reserved.
[4:0]	SYSCLKDIV	Controls SYSCLKDIV divide value in SSE-200 subsystem. Division value = SYSCLKDIV+1. These bits are read-write. Reset value `0b00000`, no division.

3.5.4 CMSDK timer

The System Control Element implements a CMSDK watchdog timer running on the **S32KCLK** clock.

The base memory addresses of the CMSDK timer control registers are:

- `0x4002_F000` in the Non-secure region.
- `0x5002_F000` in the Secure region.

See *3.3.1 CMSDK timer* on page 3-50 for a summary of the CMSDK timer control registers.

See *Arm® Cortex®-M System Design Kit Technical Reference Manual* for full descriptions of the CMSDK timer control registers.

3.5.5 CMSDK watchdog timer

The System Control Element implements a CMSDK watchdog timer running on the **S32KCLK** clock.

See *3.3.3 CMSDK watchdog timers* on page 3-53 for a summary of the CMSDK timer control registers.

See *Arm® Cortex®-M System Design Kit Technical Reference Manual* for full descriptions of the CMSDK watchdog timer control registers.

3.6 Real Time Clock (RTC) registers

The test chip implements registers that control the Arm PrimeCell Real Time Clock (RTC).

The base memory addresses of the Real Time Clock control registers are:

- `0x4010_8000` in the Non-secure region.
- `0x5010_8000` in the Secure region.

See the *Arm® PrimeCell Real Time Clock (PL031) Technical Reference Manual*.

Caution

Warm reset of the Musca-A test chip resets the Real Time Clock.

The following table shows the RTC registers in the test chip in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-28 Real time clock control registers summary

Offset	Name	Type	Reset	Width	Function
`0x0000`	RTCDR	RO	`0x0000_0000`	32	Data register.
`0x0004`	RTCMR	RW	`0x0000_0000`	32	Match register.
`0x0008`	RTCLR	RW	`0x0000_0000`	8	Load register.
`0x000C`	RTCCR	RW	`0x0000_0000`	32	Control register.
`0x0010`	RTCIMSC	RW	`0x0000_0000`	1	Interrupt mask set and clear register.
`0x0014`	RTCRIS	RO	`0x0000_0000`	1	Raw interrupt status register.
`0x0018`	RTCMIS	RO	`0x0000_0000`	32	Masked interrupt status register.
`0x001C`	RTCICR	WO	`0x0000_0000`	32	Interrupt clear register.
`0x0FE0`	RTCPeriphID0	RO	`0x0000_0031`	8	Peripheral ID register bits [7:0]
`0x0FE4`	RTCPeriphID1	RO	`0x0000_0010`	8	Peripheral ID register bits [15:8]
`0x0FE8`	RTCPeriphID2	RO	`0x0000_0004`	8	Peripheral ID register bits [23:16]
`0x0FEC`	RTCPeriphID3	RO	`0x0000_0000`	8	Peripheral ID register bits [31:24]
`0x0FF0`	RTCPCellID0	RO	`0x0000_000D`	8	PrimeCell ID register bits [7:0]
`0x0FF4`	RTCPCellID1	RO	`0x0000_00F0`	8	PrimeCell ID register bits [15:8]
`0x0FF8`	RTCPCellID2	RO	`0x0000_0005`	8	PrimeCell ID register bits [23:16]
`0x0FFC`	RTCPCellID3	RO	`0x0000_00B1`	8	PrimeCell ID register bits [31:24]

3.7 General Purpose Timer

The test chip contains registers that control the General Purpose Timer in the **32K** domain.

The base memory addresses of the General Purpose Timer control registers are:

- `0x4010_B000` in the Non-secure region.
- `0x5010_B000` in the Secure region.

The following table shows the General Purpose Timer registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-29 General Purpose Timer control registers summary

Offset	Name	Type	Reset	Width	Function
`0x0000`	GPTRESET	RO	`0x0000_0000`	32	Control Reset Register. See *3.7.1 GPTRESET Register* on page 3-84.
`0x0004`	GPTINTM	RW	`0x0000_0000`	32	Masked interrupt status register. See *3.7.2 GPTINTM Register* on page 3-85.
`0x0008`	GPTINTC	RW	`0x0000_0000`	8	Interrupt clear register. See *3.7.3 GPTINTC Register* on page 3-85.
`0x0010`	GPTALARM0	RW	`0x0000_0000`	32	ALARM0 data value register. See *3.7.4 GPTALARM0 Register* on page 3-86.
`0x0014`	GPTALARM1	RW	`0x0000_0000`	1	ALARM1 data value register. See *3.7.5 GPTALARM1 Register* on page 3-86.
`0x0018`	GPTINTR	RO	`0x0000_0000`	1	Raw interrupt status register. See *3.7.6 GPTINTR Register* on page 3-87.
`0x001C`	GPTCOUNTER	RO	`0x0000_0000`	32	Counter data value register. See *3.7.7 GPTCOUNTER Register* on page 3-87.

This section contains the following subsections:

3.7.1 GPTRESET Register

The GPTRESET Register characteristics are:

Purpose

A write resets the General Purpose Timer counter to 1.

A read returns the current value of the General Purpose Timer counter.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-30 GPTRESET Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved.
[1:0]	GPTRESET	CPU0 interrupt status. Writing `0b1` performs a software reset of the timer counter: **`0b0`** No effect. **`0b1`** Software reset. Bit[1] = ALARM1. Bit[0]=ALARM0. Reset value `0b00`.

3.7.2 GPTINTM Register

The GPTINTM Register characteristics are:

Purpose

Writing 1 to the relevant bit enables the ALARM0 or ALARM1 Interrupt.

Reading the relevant bit gives the current masked status value of the corresponding interrupt.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-31 GPTINTM Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved.
[1:0]	GPTINTM	Current masked status of the interrupt. Writing `0b1` enables the ALARM[n] interrupt: **`0b0`** No effect. **`0b1`** Enable ALARM[n] interrupt. Bit[1] = ALARM1. Bit[0]=ALARM0. Reset value `0b00`.

3.7.3 GPTINTC Register

The GPTINTC Register characteristics are:

Purpose

Writing 1 to the relevant bit disables the ALARM0 or ALARM1 Interrupt.

Reading a bit returns the current value of the bit.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-32 GPTINTC Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved.
[1:0]	GPTINTC	Writing `0b1` disables the ALARM[n] interrupt: **`0b0`** No effect. **`0b1`** Disable ALARM[n] interrupt. Bit[1] = ALARM1. Bit[0]=ALARM0. Reset value `0b00`.

3.7.4 GPTALARM0 Register

The GPTALARM0 Register characteristics are:

Purpose

The ALARM0 data value register, GPTALARM0 stores the 32-bit value that triggers the interrupt when the counter reaches that value.

Reading the register returns the trigger value.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-33 GPTALARM0 Register bit assignments

Bits	Name	Function
[31:0]	GPTALARM0_DATA	Value that triggers the ALARM0 interrupt when the counter reaches that value. Reset value `0x0000_0000`.

3.7.5 GPTALARM1 Register

The GPTALARM1 Register characteristics are:

Purpose

The ALARM1 data value register, GPTALARM1 stores the 32-bit value that triggers the interrupt when the counter reaches that value.

Reading the register returns the trigger value.

Usage constraints

There are no usage constraints.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-34 GPTALARM1 Register bit assignments

Bits	Name	Function
[31:0]	GPTALARM1_DATA	Value that triggers the ALARM1 interrupt when the counter reaches that value. Reset value `0x0000_0000`.

3.7.6 GPTINTR Register

The GPTINTR Register characteristics are:

Purpose

The raw interrupt status register, GPTINTR, stores the current raw status value of the corresponding interrupt before masking.

Reading the register returns the trigger value.

Usage constraints

This register is read-only.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-35 GPTINTR Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[2:0]	GPTINTR	Raw interrupt state, before masking, of the GPTINTR interrupt. • Bit[0]: ALARM0 interrupt status. • Bit[1]: ALARM1 interrupt status. • Bit[2]: Or-ed ALARM0 and ALARM1 interrupt status. Reset value `0b000`.

3.7.7 GPTCOUNTER Register

The GPTCOUNTER Register characteristics are:

Purpose

The counter data value register, GPTCOUNTER, stores the current 32-bit value of the Timer Counter.

Reading the register returns the trigger value.

Usage constraints

This register is read-only.

The following table shows the bit assignments of the CPU0 interrupt registers.

Table 3-36 GPTCOUNTER Register bit assignments

Bits	Name	Function
[31:0]	GPTCOUNTER	Current value of 32-bit Timer Counter. Reset value `0x0000_0000`.

3.8 One-Time Programmable (OTP) secure registers

The test chip supplies 256 memory mapped registers in the non-volatile output interface of the CryptoCell outputs of the SSE-200 block.

Purpose

Software assigns meanings to the OTP registers. Possible uses are life-cycle management, key storage, and non-volatile firmware counters.

Usage constraints

The OTP registers emulate One-Time Programming.

The base memory address of the OTP secure registers are:

- `0x0E00_0000` in the non-secure region of memory.
- `0x1E00_0000` in the secure region of memory.

The following table shows the bit assignments the OTP registers.

Table 3-37 OTP_REG_n Register bit assignments

Bits	Name	Function
[31:0]	OTP_REG_n[31:0]	Software assigns the bit meanings. Reset value `0x0000_0000`.

3.9 Serial Configuration Control registers

This section describes the Musca-A Serial Configuration Control (SCC) registers.

This section contains the following subsections:

3.9.1 IOMUX Registers

The IOMUX registers, which are part of the SCC register bank, control the GPIO multiplexer logic that is connected to the Musca-A test chip I/O pins GPIO[15:0].

The following figure shows the GPIO multiplexer logic.

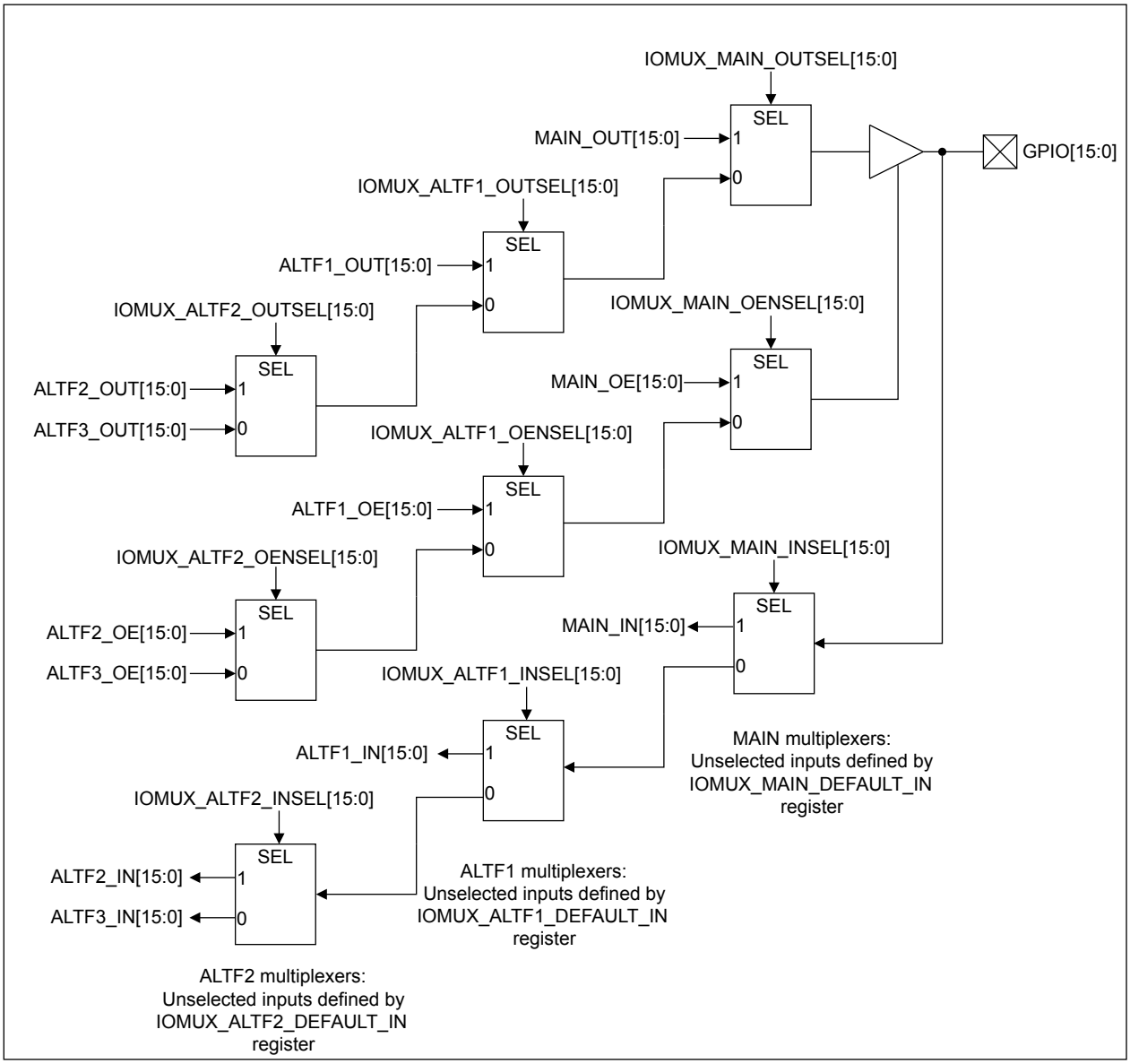

Figure 3-5 GPIO multiplex logic

The IOMUX Registers are:

- *IOMUX_MAIN_INSEL Register* on page 3-100:
  — Selects MAIN_IN or ALTF1 as connection from GPIO input data.
- *IOMUX_MAIN_OUTSEL Register* on page 3-100:
  — Selects MAIN_OUT or ALTF1 as GPIO output data.
- *IOMUX_MAIN_OENSEL Register* on page 3-101:
  — Selects MAIN_OE or ALTF1 as GPIO output enable signal.
- *IOMUX_MAIN_DEFAULT_IN Register* on page 3-101:
  — Drives unselected outputs of MAIN input multiplexers to defined logic levels to prevent floating nodes.
- *IOMUX_ALTF1_INSEL Register* on page 3-102:
  — Selects ALTF1_IN or ALTF2 as connection from MAIN input multiplexer.
- *IOMUX_ALTF1_OUTSEL Register* on page 3-102:
  — Selects ALTF1_OUT or ALTF2 as connection to MAIN_OUT output multiplexer.
- *IOMUX_ALTF1_OENSEL Register* on page 3-103:
  — Selects ALTF1_OE or ALTF2 as connection to MAIN_OESEL multiplexer.
- *IOMUX_ALTF1_DEFAULT_IN Register* on page 3-103:
  — Drives unselected outputs of ALTF1 input multiplexers to defined logic levels to prevent floating nodes.
- *IOMUX_ALTF2_INSEL Register* on page 3-104:
  — Selects ALTF2_IN or ALTF3_IN as connection from ALTF1 input multiplexer.
- *IOMUX_ALTF2_OUTSEL Register* on page 3-104:
  — Selects ALTF1_OUT or ALTF3_OUT as connection to ALTF1 output data multiplexer.
- *IOMUX_ALTF2_OENSEL Register* on page 3-105:
  — Selects ALTF2_OE or ALTF3_OE as connection to ALTF1_OESEL multiplexer.
- *IOMUX_ALTF2_DEFAULT_IN Register* on page 3-105:
  — Drives unselected outputs of ALTF2 input multiplexers to defined logic levels to prevent floating nodes.

See *2.9 Arduino Shield expansion* on page 2-35 or *A.1 Arduino Shield connectors* on page Appx-A-121 for the ALTF1 and ALTF2 GPIO pin functions.

Note

GPIO ALTF3 functions are reserved.

3.9.2 SCC registers summary

The test chip implements Serial Configuration Control (SCC) registers.

The base memory addresses of the SCC registers are:
- `0x4010_C000` in the Non-secure region.
- `0x5010_C000` in the Secure region.

The following table shows the registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-38 Serial Configuration Controller registers summary

Offset	Name	Type	Reset	Width	Description
`0x0000`	RESET_CTRL	RW	`0xFFFF_FFFF`	32	See *RESET_CTRL Register* on page 3-92.
`0x0004`	CLK_CTRL	RW	`0x0001_0627`	32	See *CLK_CTRL Register* on page 3-94.
`0x000C`	PLL_CTRL	RW	`0x0BEB_0118`	32	See *PLL_CTRL Register* on page 3-95.
`0x0010`	DBG_CTRL	RW	`0x0000_003E`	32	See *DBG_CTRL Register* on page 3-96.

Table 3-38 Serial Configuration Controller registers summary (continued)

Offset	Name	Type	Reset	Width	Description
0x0018	INTR_CTRL	RW	0x0000_0002	32	See *INTR_CTRL Register* on page 3-97.
0x0020	CPU0_VTOR_SRAM	RW	0x0020_0000	32	See *CPU0_VTOR_SRAM Register* on page 3-99.
0x0028	CPU1_VTOR_SRAM	RW	0x0020_0000	32	See *CPU1_VTOR_SRAM Register* on page 3-99.
0x0030	IOMUX_MAIN_INSEL	RW	0xFFFF_FFFF	32	See *IOMUX_MAIN_INSEL Register* on page 3-100 and *3.9.1 IOMUX Registers* on page 3-89.
0x0034	IOMUX_MAIN_OUTSEL	RW	0xFFFF_FFFF	32	See *IOMUX_MAIN_OUTSEL Register* on page 3-100 and *3.9.1 IOMUX Registers* on page 3-89.
0x0038	IOMUX_MAIN_OENSEL	RW	0xFFFF_FFFF	32	See *IOMUX_MAIN_OENSEL Register* on page 3-101 and *3.9.1 IOMUX Registers* on page 3-89.
0x003C	IOMUX_MAIN_DEFAULT_IN	RW	0x0000_0000	32	See *IOMUX_MAIN_DEFAULT_IN Register* on page 3-101 and *3.9.1 IOMUX Registers* on page 3-89.
0x0040	IOMUX_ALTF1_INSEL	RW	0x0000_0000	32	See *IOMUX_ALTF1_INSEL Register* on page 3-102 and *3.9.1 IOMUX Registers* on page 3-89.
0x0044	IOMUX_ALTF1_OUTSEL	RW	0xFFFF_FFFF	32	See *IOMUX_ALTF1_OUTSEL Register* on page 3-102 and *3.9.1 IOMUX Registers* on page 3-89.
0x0048	IOMUX_ALTF1_OENSEL	RW	0xFFFF_FFFF	32	See *IOMUX_ALTF1_OENSEL Register* on page 3-103 and *3.9.1 IOMUX Registers* on page 3-89.
0x004C	IOMUX_ALTF1_DEFAULT_IN	RW	0x0000_0000	32	See *IOMUX_ALTF1_DEFAULT_IN Register* on page 3-103 and *3.9.1 IOMUX Registers* on page 3-89.
0x0050	IOMUX_ALTF2_INSEL	RW	0x0000_0000	32	See *IOMUX_ALTF2_INSEL Register* on page 3-104 and *3.9.1 IOMUX Registers* on page 3-89.
0x0054	IOMUX_ALTF2_OUTSEL	RW	0xFFFF_FFFF	32	See *IOMUX_ALTF2_OUTSEL Register* on page 3-104 and *3.9.1 IOMUX Registers* on page 3-89.
0x0058	IOMUX_ALTF2_OENSEL	RW	0xFFFF_FFFF	32	See *IOMUX_ALTF2_OENSEL Register* on page 3-105 and *3.9.1 IOMUX Registers* on page 3-89.
0x005C	IOMUX_ALTF2_DEFAULT_IN	RW	0x0000_0000	32	See *IOMUX_ALTF2_DEFAULT_IN Register* on page 3-105 and *3.9.1 IOMUX Registers* on page 3-89.
0x0064	SPARE0	RW	0x0000_0000	32	See *SPARE0 Register* on page 3-106.
0x0068	IOPAD_DSO	RW	0xFFFF_FFFF	32	See *IOPAD_DS0 and IOPAD_DS1 Registers* on page 3-106.
0x006C	IOPAD_DS1	RW	0xFFFF_FFFF	32	See *IOPAD_DS0 and IOPAD_DS1 Registers* on page 3-106.
0x0070	IOPAD_PE	RW	0xFFFF_FFFF	32	See *IOPAD_PE Register* on page 3-107.
0x0074	IOPAD_PS	RW	0xFFFF_FFFF	32	See *IOPAD_PS Register* on page 3-107.
0x0078	IOPAD_SR	RW	0x0000_0000	32	See *IOPAD_SR Register* on page 3-108.
0x007C	IOPAD_IS	RW	0xFFFF_FFFF	32	See *IOPAD_IS Register* on page 3-108.

Table 3-38 Serial Configuration Controller registers summary (continued)

Offset	Name	Type	Reset	Width	Description
0x0084	STATIC_CONF_SIG0	RW	0x0000_0000	32	See *STATIC_CONF_SIG0 Register* on page 3-109.
0x0088	STATIC_CONF_SIG1	RW	0x0000_0000	32	See *STATIC_CONF_SIG1 Register* on page 3-110.
0x0100	CHIP_ID	RO	0x0797_0477	32	See *CHIP_ID Register* on page 3-111.
0x0104	CLK_STATUS	RO	0x0000_0005	32	See *CLK_STATUS Register* on page 3-112.

RESET_CTRL Register

The RESET_CTRL Register characteristics are:

Purpose

Controls reset signals to test chip peripherals.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-39 RESET_CTRL Register bit assignments

Bits	Name	Function
[31:18]	-	Reserved.
[17]	OTP_PSEL_ENABLE	PSEL enable. Reset value 0b1:
[16]	OTP_RESET	0b0: Reset OTP. 0b1: No effect. Reset value 0b1.
[15]	-	Reserved.
[14]	RTC_RESET	0b0: Reset real time clock. 0b1: No effect. Reset value 0b1.
[13]	PWM2_RESET	0b0: Reset PWM2. 0b1: No effect. Reset value 0b1.
[12]	PWM1_RESET	0b0: Reset PWM1. 0b1: No effect. Reset value 0b1.

Table 3-39 RESET_CTRL Register bit assignments (continued)

Bits	Name	Function
[11]	PWM0_RESET	`0b0`: Reset PWM0. `0b1`: No effect. Reset value `0b1`.
[10]	-	Reserved.
[9]	GPIO_RESET	`0b0`: Reset GPIO. `0b1`: No effect. Reset value `0b1`.
[8]	UART1_RESET	`0b0`: Reset UART1. `0b1`: No effect. Reset value `0b1`.
[7]	UART0_RESET	`0b0`: Reset UART0. `0b1`: No effect. Reset value `0b1`.
[6]	QSPI_RESET	`0b0`: Reset QSPI. `0b1`: No effect. Reset value `0b1`.
[5]	SPI_RESET	`0b0`: Reset SPI. `0b1`: No effect. Reset value `0b1`.
[4]	I2S_RESET	`0b0`: Reset I^2S. `0b1`: No effect. Reset value `0b1`.
[3]	I2C1_RESET	`0b0`: Reset I^2C1. `0b1`: No effect. Reset value `0b1`.
[2]	I2C0_RESET	`0b0`: Reset I^2C0. `0b1`: No effect. Reset value `0b1`.

Table 3-39 RESET_CTRL Register bit assignments (continued)

Bits	Name	Function
[1]	GPTIMER_RESET	`0b0`: Reset general-purpose timer. `0b1`: No effect. Reset value `0b1`.
[0]	SCC_RESET	`0b0`: Reset SCC registers. `0b1`: No effect. Reset value `0b1`.

CLK_CTRL Register

The CLK_CTRL Register characteristics are:

Purpose

Controls the test chip clocks and clock PLLs.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-40 CLK_CTRL Register bit assignments

Bits	Name	Function
[31:17]	-	Reserved.
[16]	CLK_DAP_EN	Enable DAP clock. Reset value `0b1`.
[15]	CLK_TEST_EN	Enable test clock. Reset value `0b0`.
[14:12]	CLK_TEST_SEL	Select test clock: `0b000`: PLLVCLOCK. `0b001`: **FCLK**. `0b010`: **SYSCLK**. `0b011`: Reserved. `0b100`: Reserved. `0b101`: **SCCCLK**. Reset value `0b000`.
[11:8]	-	Reserved.

Table 3-40 CLK_CTRL Register bit assignments (continued)

Bits	Name	Function
[7]		Select PLL input clock: `0b0`: **SCCCLK.** (from pad). `0b1`: Reserved. Reset value `0b0`.
[6]	-	Select SCC clock: `0b0`: **SCCCLK.** (from pad). `0b1`: **FASTCLK.** (from pad). Reset value `0b0`.
[5]	CLK_REF_EN	Enable reference clock. Reset value `0b1`.
[4]	CLK_REF_SEL	Select reference clock: `0b0`: Pad 32k REFCLK. `0b1`: FASTCLK. Reset value `0b0`.
[3]	CLK_SCC_EN	Enable SCC clock. Reset value `0b0`.
[2]	CLK_MAIN_EN	Enable main clock. Reset value `0b1`.
[1:0]	CLK_MAIN_SEL	Select main clock: `0b00` and CLK_PLL_IN_SEL = `0b0`. **REFCLK (32kHz clock).** `0b00` and CLK_PLL_IN_SEL = `0b1`: Reserved `0b10`: **FASTCLK.** `0b11`: **PLL out**. Reset value `0b11`.

PLL_CTRL Register

The PLL_CTRL Register characteristics are:

Purpose

Controls the power and control signals to the PLLs in the test chip.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-41 PLL_CTRL Register bit assignments

Bits	Name	Function
[31:30]	-	Reserved.
[29:16]	FBDIV	PLL feedback divide value. FBDIV=VCO_OP/FREF. Reset value `0x0BEB`.
[15:12]	POSTDIV2	Second post divide value for FOUT2. Not used.
[11:8]	POSTDIV1	First post divide value for FOUT1. DIV value = postdiv1 + 1 (1 ≥ Div2. Reset value `0b0001`.
[7:3]	-	Reserved.
[2]	FOUTPOSTDIV1PD	First post-divide power down: `0b0`: Not powered down. `0b1`: Powered down. Reset value `0b0`.
[1]	BYPASS	FREF is bypassed to FOUT1 and FOUT2. Reset value `0b0`.
[0]	PD	Power down PLL. `0b0`: Not powered down. `0b1`: Powered down. Reset value `0b0`.

DBG_CTRL Register

The DBG_CTRL Register characteristics are:

Purpose

Controls the debug signals in the test chip.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-42 DBG_CTRL Register bit assignments

Bits	Name	Function
[31]	DBG_DCU_FORCE	`0b0`: Use Crypto DCU. `0b1`: Force use of SCC signals. Reset value `0b0`.
[30:11]	-	Reserved.
[10]	CHSEC_SE_CNTL	Reset value `0b0`.
[9]	CHSEC_DISCHARGE_CNTL	Reset value `0b0`.
[8]	CHSEC_BYPASS	Reset value `0b0`.
[7:6]	CHSEC_FREQ_SEL[1:0]	Reset value `0b00`.
[5]	-	Reserved.
[4]	LLCDBGGENIN	Reset value `0b1`.
[3]	SPNIDENIN	Secure Privilege Non-Invasive Debug Enable Input. Reset value `0b1`.
[2]	SPIDENIN	Secure Privilege Invasive Debug Enable Input. Reset value `0b1`.
[1]	NIDENIN	Non-Invasive Debug Enable Input. Reset value `0b1`.
[0]	DEBUGENIN	Debug Enable Input. Reset value `0b0`.

INTR_CTRL Register

The INTR_CTRL Register characteristics are:

Purpose

Controls interrupt signals in the test chip.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-43 INTR_CTRL Register bit assignments

Bits	Name	Function
[31:18]	-	Reserved.
[17]	EXT_INTR_INVERT_EN	Invert polarity for combined **ext_irq**. `0b0`: No effect. `0b1`: Invert polarity. Reset value `0b0`.
[16]	EXT_INTR_EN	PA26 External interrupt enable. `0b0`: No effect. `0b1`: Enable interrupt. Reset value `0b0`.
[15:6]	-	Reserved.
[5]	SRAM_MPC_IRQ_CLEAR	Reset value `0b0`. Clear SRAM Memory Protection Controller Interrupt. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b1`.
[4]	SRAM_MPC_IRQ_ENABLE	Enable SRAM Memory Protection Controller Interrupt. `0b0`: No effect. `0b1`: Enable interrupt. Reset value `0b0`.
[3]	QSPI_MPC_IRQ_CLEAR	Clear QSPI Memory Protection Controller Interrupt. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b0`.
[2]	QSPI_MPC_IRQ_ENABLE	Enable QSPI Memory Protection Controller Interrupt. `0b0`: No effect. `0b1`: Enable interrupt. Reset value `0b0`.

Table 3-43 INTR_CTRL Register bit assignments (continued)

Bits	Name	Function
[1]	AHB_PPC_IRQ_CLEAR	Clear AHB Peripheral Protection Controller Interrupt. `0b0`: No effect. `0b1`: Clear interrupt. Reset value `0b1`.
[0]	AHB_PPC_IRQ_ENABLE	Enable AHB Peripheral Protection Controller Interrupt. `0b0`: No effect. `0b1`: Enable interrupt. Reset value `0b0`.

CPU0_VTOR_SRAM Register

The CPU0_VTOR_SRAM Register characteristics are:

Purpose

Controls Core 0 reset vector signals in the test chip.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-44 CPU0_VTOR_SRAM Register bit assignments

Bits	Name	Function
[31:7]	CPU0_VTOR_SECURE	Reset vector for Core 0 secure mode. Reset value `0x000_4000`.
[6:0]	-	Reserved.

CPU1_VTOR_SRAM Register

The CPU1_VTOR_SRAM Register characteristics are:

Purpose

Controls Core 1 reset vector signals in the test chip.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-45 CPU1_VTOR_SRAM Register bit assignments

Bits	Name	Function
[31:7]	CPU1_VTOR_SECURE	Reset vector for Core 1 secure mode. Reset value `0x000_4000`.
[6:0]	-	Reserved.

IOMUX_MAIN_INSEL Register

The IOMUX_MAIN_INSEL Register characteristics are:

Purpose

Selects MAIN_IN or ALTF1 as input data.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-46 IOMUX_MAIN_INSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_MAIN_INSEL[15:0]	IO main function input data select: `0b0`: GPIO input goes to ALTF1. `0b1`: GPIO input goes to MAIN_IN. Reset value `0xFFFF`.

IOMUX_MAIN_OUTSEL Register

The IOMUX_MAIN_OUTSEL Register characteristics are:

Purpose

Selects MAIN_OUT or ALTF1 as GPIO output data.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-47 IOMUX_MAIN_OUTSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_MAIN_OUTSEL[15:0]	IO main function output data select: `0b0`: Select ALTF1. `0b1`: Select MAIN_OUT. Reset value `0xFFFF`.

IOMUX_MAIN_OENSEL Register

The IOMUX_MAIN_OENSEL Register characteristics are:

Purpose

Selects MAIN_OE or ALTF1 as GPIO output enable signal.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-48 IOMUX_MAIN_OENSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_MAIN_OENSEL[15:0]	IO main function output data select: `0b0`: Selects ALTF1. `0b1`: Selects MAIN_OE. Reset value `0xFFFF`.

IOMUX_MAIN_DEFAULT_IN Register

The IOMUX_MAIN_DEFAULT_IN Register characteristics are:

Purpose

Drives unselected outputs of MAIN input multiplexers to defined logic levels to prevent floating nodes.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-49 IOMUX_MAIN_DEFAULT_IN Register bit assignments

Bits	Name	Function
[31:0]	-	Reserved.
[15:0]	IOMUX_MAIN_DEFAULT_IN[15:0]	Defines value of unselected outputs of MAIN input multiplexers: `0b0`: Defaults to `0b0`. `0b1`: Defaults to `0b1`. Reset value `0x0000`.

IOMUX_ALTF1_INSEL Register

The IOMUX_ALTF1_INSEL Register characteristics are:

Purpose

Selects ALTF1 or ALTF2 as connection from MAIN input multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-50 IOMUX_ALTF1_INSEL Register bit assignments

Bits	Name	Function
[31:0]	-	Reserved.
[15:0]	IOMUX_ALTF1_INSEL[15:0]	Select Alternate Function 1 input data: `0b0`: Select ALTF2. `0b1`: Selects ALTF1. Reset value `0x0000`.

IOMUX_ALTF1_OUTSEL Register

The IOMUX_ALTF1_OUTSEL Register characteristics are:

Purpose

Selects ALTF1_OUT or ALTF2 as connection to MAIN output multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-51 IOMUX_ALTF1_OUTSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF1_OUTSEL [15:0]	Select Alternate Function 1 output data: `0b0`: Select ALTF2. `0b1`: Select ALTF1_OUT. Reset value `0xFFFF`.

IOMUX_ALTF1_OENSEL Register

The IOMUX_ALTF1_OENSEL Register characteristics are:

Purpose

Selects ALTF1_OE or ALTF2 as connection to MAIN_OENSEL multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-52 IOMUX_ALTF1_OENSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[150]	IOMUX_ALTF1_OENSEL[15:0]	Selects Alternate Function 1 output data enable: `0b0`: Selects ALTF2. `0b1`: Selects ALTF1_OE. Reset value `0xFFFF`.

IOMUX_ALTF1_DEFAULT_IN Register

The IOMUX_ALTF1_DEFAULT_IN Register characteristics are:

Purpose

Drives unselected outputs of ALTF1 input multiplexers to defined logic levels to prevent floating nodes. See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-53 IOMUX_ALTF1_DEFAULT_IN Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF1_DEFAULT_IN[15:0]	Defines value of unselected outputs of ALTF1 input multiplexers: `0b0`: Defaults to `0b0`. `0b1`: Defaults to `0b1`. Reset value `0x0000`.

IOMUX_ALTF2_INSEL Register

The IOMUX_ALTF2_INSEL Register characteristics are:

Purpose

Selects ALTF1 or ALTF2 as connection from input multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-54 IOMUX_ALTF2_INSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF2_INSEL	Select Alternate Function 2 input data: `0b0`: Select ALTF3_IN. `0b1`: Select ALTF2_IN. Reset value `0x0000`.

IOMUX_ALTF2_OUTSEL Register

The IOMUX_ALTF2_OUTSEL Register characteristics are:

Purpose

Selects ALTF2_OUT or ALTF3_OUT as connection to ALTF1 output multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-55 IOMUX_ALTF2_OUTSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF2_OUTSEL	Select Alternate Function 2 output data: `0b0`: Select ALTF3_OUT. `0b1`: Select ALTF2_OUT. Reset value `0xFFFF`.

IOMUX_ALTF2_OENSEL Register

The IOMUX_ALTF2_OENSEL Register characteristics are:

Purpose

Selects ALTF2_OE or ALTF3_OE as connection to ALTF1_OENSEL multiplexer.

See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-56 IOMUX_ALTF2_OENSEL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF2_OENSEL[15:0]	Selects Alternate Function 2 output data enable: `0b0`: Select ALTF3_OE. `0b1`: Select ALTF2_OE. Reset value `0xFFFF`.

IOMUX_ALTF2_DEFAULT_IN Register

The IOMUX_ALTF2_DEFAULT_IN Register characteristics are:

Purpose

Drives unselected outputs of ALTF2 input multiplexers to defined logic levels to prevent floating nodes. See *3.9.1 IOMUX Registers* on page 3-89.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-57 IOMUX_ALTF2_DEFAULT_IN Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	IOMUX_ALTF2_DEFAULT_IN[15:0]	Defines value of unselected outputs of ALTF2 input multiplexers: `0b0`: Defaults to `0b0`. `0b1`: Defaults to `0b1`. Reset value `0x0000`.

SPARE0 Register

The SPARE0 Register characteristics are:

Purpose

Spare read-write register for use by software.

Usage constraints

There are no usage constraints.

Memory offset and full register reset value

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-58 SPARE0 Register bit assignments

Bits	Name	Function
[31:0]	SPARE0	Spare read-write register for software. Software assigns bit meanings. Reset value `0x0000_0000`.

IOPAD_DS0 and IOPAD_DS1 Registers

The IOPAD_DS0 and IOPAD_DS1 Register characteristics are:

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

Purpose

The corresponding bits of the two registers combine to form two-bit values that define the corresponding GPIO drive strengths. The following table shows how the bits of the IOPAD_DS1 and IOPAD_DS0 Registers define the drive strengths.

Table 3-59 GPIO drive strengths

IOPAD_DS1/DS0	Drive strength (mA)
`0b00`	2
`0b01`	8
`0b10`	4
`0b11`	12 (default)

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit IOPAD_DS0 bit assignments.

Table 3-60 IOPAD_DS0 Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	DRIVE_STRENGTH0	Least significant bits of two-bit values that define corresponding GPIO drive strengths. Reset value `0xFFFF`.

The following table shows the bit IOPAD_DS1 bit assignments.

Table 3-61 IOPAD_DS1 Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	DRIVE_STRENGTH1	Most significant bits of two-bit values that define corresponding GPIO drive strengths. Reset value `0xFFFF`.

IOPAD_PE Register

The IOPAD_PE Register characteristics are:

Purpose

Enables pull resistors on GPIO pins.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-62 IOPAD_PE Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	PULL_ENABLE	Enable pull resistor on GPIOs: `0b0`: Not enable. `0b1`: Enable. Reset value `0xFFFF`.

IOPAD_PS Register

The IOPAD_PS Register characteristics are:

Purpose

Selects pull mode of GPIO pins.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-63 IOPAD_PS Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	PULL_SELECT	Select pull mode: `0b0`: Pull down. `0b1`: Pull up. Reset value `0xFFFF`.

IOPAD_SR Register

The IOPAD_SR Register characteristics are:

Purpose

Selects slew rate of GPIO pins.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-64 IOPAD_SR Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:0]	SLEW_RATE	Select slew rate: `0b0`: Fast. `0b1`: Slow. Reset value `0x0000`.

IOPAD_IS Register

The IOPAD_IS Register characteristics are:

Purpose

Selects input mode of GPIO pins.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-65 IOPAD_IS Register bit assignments

Bits	Name	Function
[31:16]	-	
[15:0]	INPUT_SELECT	Select input mode: `0b0`: CMOS. `0b1`: Schmitt. Reset value `0xFFFF`.

STATIC_CONF_SIG0 Register

The STATIC_CONF_SIG0 Register characteristics are:

Purpose

Static configuration control register.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-66 STATIC_CONF_SIG0 Register bit assignments

Bits	Name	Function
[31:9]	-	Reserved.
[8]	SPNIDENSELDIS	SPNIDEN selector disable. Disables the SPNIDEN selector logic and forces SPNIDEN to use SPNIDENIN: `0b0`: Not disabled. `0b1`: Disabled. Reset value `0b0`.
[7]	SPIDENSELDIS	SPIDEN selector disable. Disables the SPIDEN selector logic and forces SPIDEN to use SPIDENIN: `0b0`: Not disabled. `0b1`: Disabled. Reset value `0b0`.

Table 3-66 STATIC_CONF_SIG0 Register bit assignments (continued)

Bits	Name	Function
[6]	NIDENSELDIS	NIDEN selector disable. Disables the NIDEN selector logic and forces NIDEN to use NIDENIN: `0b0`: Not disabled. `0b1`: Disabled. Reset value `0b0`.
[5]	DBGENSELDIS	DBGEN selector disable. Disables the DBGEB selector logic and forces DBGEN to use DBGENIN: `0b0`: Not disabled. `0b1`: Disabled. Reset value `0b0`.
[4:1]	CTMCHCIHSBYPASS	Defines whether each pin of CTMCHOUT of the Cross Trigger Channel Interface is synchronous or asynchronous with DBGSYSCLK: `0b0`: Asynchronous. Enables the synchronization and handshake logic on CTMCHOUT[n]. `0b1`: Synchronous. Disables the synchronization and handshake logic on CTMCHOUT[n]. Reset value `0b0000`.
[0]	CTMCHCISBYPASS	Defines whether CTMCHIN and CTMCHOUTACK[3:0] of the Cross Trigger Channel Interface is synchronous or asynchronous with DBGSYSCLK: `0b0`: Asynchronous. Signals are resynchronized internally. `0b1`: Synchronous. Signals are not resynchronized internally. Reset value `0b0`.

STATIC_CONF_SIG1 Register

The STATIC_CONF_SIG1 Register characteristics are:

Purpose

Static configuration control register.

Usage constraints

There are no usage constraints.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-67 STATIC_CONF_SIG1 Register bit assignments

Bits	Name	Function
[31:28]	-	Reserved.
[27:24]	TODBGENSEL	DBGEN mask on CTITRIGOUT: `0b0`: Mask trigger output of associated Cross Trigger Interface output when DBGEN is low. `0b1`: Not mask trigger output of associated Cross Trigger Interface output. Reset value `0b0000`.
[23:16]	TINIDENSEL	NIDEN mask on CTITRIGINT: `0b0`: Mask trigger input of associated Cross Trigger Interface output when NIDEN is low. `0b1`: Not mask trigger output of associated Cross Trigger Interface output. Reset value `0x00`.
[15:12]	TIHSBYPASS	Cross Trigger Interface handshake bypass on CTITRIGOUT. Disables the SPIDEN selector logic and forces SPIDEN to use SPIDENIN: `0b0`: Not disable. `0b1`: Disable. Reset value `0b0000`.
[11:8]	TISBYPASSACK	Cross Trigger Interface synchronous bypass on CTITRIGOUTACK. Set HIGH to bypass the synchronization logic if the CTITRIGOUTACK input is synchronous with DBGSYSCLK and is driven from the same clock domain: `0b0`: Not bypass. `0b1`: Bypass. Reset value `0b0000`.
[7:0]	TISBYPASSIN	Cross Trigger Interface synchronous bypass on CTITRIGIN. Set HIGH to bypass the synchronization logic if the CTITRIGIN input is synchronous with DBGSYSCLK and is driven from the same clock domain: `0b0`: Not bypass. `0b1`: Bypass. Reset value `0x00`.

CHIP_ID Register

The CHIP_ID Register characteristics are:

Purpose

Stores component identification information.

Usage constraints

This register is read-only.

The following table shows the bit assignments.

Table 3-68 CHIP_ID Register bit assignments

Bits	Name	Function
[31:0]	CHIP_ID	Component ID information. The value in the Musca-A test chip is `0x0797_0477`.

CLK_STATUS Register

The CLK_STATUS Register characteristics are:

Purpose

Stores clock status information.

Usage constraints

This register is read-only.

Memory offset and full register reset values

See *3.9.2 SCC registers summary* on page 3-90.

The following table shows the bit assignments.

Table 3-69 CLK_STATUS Register bit assignments

Bits	Name	Function
[31:3]	-	Reserved.
[2]	PLL_LOCK	PLL Lock status: 0: PLL not locked. 1: PLL locked. Reset value `0b1`.
[1]	-	Reserved.
[0]	CLK_MAIN_RDY	CLK_MAIN ready (active): 0: CLK_MAIN not ready. 1: CLK_MAIN ready. Reset value `0b1`.

3.10 GPIO registers

The test chip implements GPIO pin control registers.

The base memory address of the GPIO control registers is `0x5011_0000` in the Secure region.

See *Arm® Cortex®-M System Design Kit Technical Reference Manual*.

The following table shows the GPIO control registers in the test chip in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-70 GPIO control registers summary

Offset	Name	Type	Reset	Width	Function
`0x0000`	GPIODATA	RW	`0x0000_0000`	32	Data value. Bits [31:16] are reserved.
`0x0004`	GPIODATAOUT	RW	`0x0000_0000`	32	Data output value. Bits [31:16] are reserved.
`0x0010`	GPIOOUTENSET	RW	`0x0000_0000`	32	Output enable set. Bits [31:16] are reserved.
`0x0014`	GPIOOUTENCLR	RW	`0x0000_0000`	32	Output enable clear. Bits [31:16] are reserved.
`0x0018`	GPIOALTFUNCSET	RW	`0x0000_0000`	32	Alternative function set. Bits [31:16] are reserved.
`0x001C`	GPIOALTFUNCCLR	RW	`0x0000_0000`	32	Alternative function clear. Bits [31:16] are reserved. Reserved.
`0x0020`	GPIOINTENSET	RW	`0x0000_0000`	32	Interrupt enable set. Bits [31:16] are reserved.
`0x0024`	GPIOINTENCLR	RW	`0x0000_0000`	32	Interrupt enable clear. Bits [31:16] are reserved.
`0x0028`	GPIOINTTYPESET	RW	`0x0000_0000`	32	Interrupt type set. Bits [31:16] are reserved.
`0x002C`	GPIOINTTYPECLR	RW	`0x0000_0000`	32	Interrupt type clear. Bits [31:16] are reserved.
`0x0030`	GPIOINTPOLSET	RW	`0x0000_0000`	32	Polarity-level, edge IRQ configuration. Set interrupt polarity bit. Bits [31:16] are reserved.
`0x0034`	GPIOINTPOLCLR	RW	`0x0000_0000`	32	Polarity-level, edge IRQ configuration. Clear interrupt polarity bit. Bits [31:16] are reserved.
`0x0038`	GPIOINTSTATUS INTCLEAR	RW	`0x0000_0000`	32	Clear interrupt request. Bits [31:16] are reserved.

Table 3-70 GPIO control registers summary (continued)

Offset	Name	Type	Reset	Width	Function
0x0FD0	GPIOPID4	RW	0x0000_0000	32	Peripheral ID Register 4. Bits [31:8] are reserved.
0x0FE0	GPIOPID0	RW	0x0000_0000	32	Peripheral ID Register 0. Bits [31:8] are reserved.
0x0FE4	GPIOPID1	RW	0x0000_0000	32	Peripheral ID Register 1. Bits [31:8] are reserved.
0x0FE8	GPIOPID2	RW	0x0000_0000	32	Peripheral ID Register 2. Bits [31:8] are reserved.
0x0FEC	GPIOPID3	RW	0x0000_0000	32	Peripheral ID Register 3. Bits [31:8] are reserved.
0x0FF0	GPIOCID0	RW	0x0000_0000	32	Component ID Register 0. Bits [31:8] are reserved.
0x0FF4	GPIOCID1	RW	0x0000_0000	32	Component ID Register 1. Bits [31:8] are reserved.
0x0FF8	GPIOCID2	RW	0x0000_0000	32	Component ID Register 2. Bits [31:8] are reserved.
0x0FFC	GPIOCID3	RW	0x0000_0000	32	Component ID Register 3. Bits [31:8] are reserved.

3.11 UART registers

The test chip contains registers that control the functions of the two UARTs, UART0 and UART1.

The base memory addresses of UART0 are:

- `0x4010_1000` in the Non-secure region.
- `0x5010_1000` in the Secure region.

The base memory addresses of UART1 are:

- `0x4010_2000` in the Non-secure region.
- `0x5010_2000` in the Secure region.

See *PrimeCell UART (PL011) Technical Reference Manual*.

Note

The UART on the Musca-A test chip does not support hardware flow control.

The following table shows the UART0 and UART1 control registers in address offset order from the base memory address. Undefined registers are reserved. Software must not attempt to access these registers.

Table 3-71 UART control registers summary

Offset	Name	Type	Reset value	Width	Function
`0x0000`	UART0DR	RW	-	32	Data Register.
`0x0004`	UART0RSR/UART0ECR	RW	`0x0000_0000`	32	Receive Status Register/Error Clear Register.
`0x0018`	UART0FR	RO	`0x0000_0012`	32	Flag Register.
`0x0020`	UART0ILPR	RW	`0x0000_0000`	32	IrDA Low-Power Counter Register.
`0x0024`	UART0IBRD	RW	`0x0000_0000`	32	Integer Baud Rate Register.
`0x0028`	UART0FBRD	RW	`0x0000_0000`	32	Fractional Baud Rate Register.
`0x002C`	UART0LCR_H	RW	`0x0000_0000`	32	Line Control Register.
`0x0030`	UART0CR	RW	`0x0000_0300`	32	Control Register.
`0x0034`	UART0IFLS	RW	`0x0000_0012`	32	Interrupt FIFO Level Select Register.
`0x0038`	UART0IMSC	RW	`0x0000_0000`	32	Interrupt Mask Set/Clear Register.
`0x003C`	UART0RIS	RO	`0x0000_0000`	32	Raw Interrupt Status Register.
`0x0040`	UART0MIS	RO	`0x0000_0000`	32	Masked Interrupt Status Register.
`0x0044`	UART0ICR	WO	-	32	Interrupt Clear Register.
`0x0048`	UART0DMACR	RW	`0x0000_0000`	32	DMA Control Register.
`0x0FE0`	UART0PeriphID0	RO	`0x0000_0011`	32	UART0 peripheral ID Register 0.
`0x0FE4`	UART0PeriphID1	RO	`0x0000_0010`	32	UART0 peripheral ID Register 1.
`0x0FE8`	UART0PeriphID2	RO	`0x0000_0004`	32	UART0 peripheral ID Register 2.
`0x0FEC`	UART0PeriphID3	RO	`0x0000_0000`	32	UART0 peripheral ID Register 3.

Table 3-71 UART control registers summary (continued)

Offset	Name	Type	Reset value	Width	Function
0x0FF0	UART0PCellID0	RO	0x0000_000D	32	UART0 component ID Register 0.
0x0FF4	UART0PCellID1	RO	0x0000_00F0	32	UART0 component ID Register 1.
0x0FF8	UART0PCellID2	RO	0x0000_0005	32	UART0 component ID Register 2.
0x0FFC	UART0PCellID3	RO	0x0000_00B1	32	UART0 component ID Register 3.
0x1000	UART1DR	RW	-	32	Data Register.
0x1004	UART1RSR/UART1ECR	RW	0x0000_0000	32	Receive Status Register/Error Clear Register.
0x1018	UART1FR	RO	0x0000_0012	32	Flag Register.
0x1020	UART1ILPR	RW	0x0000_0000	32	IrDA Low Power Counter Register.
0x1024	UART1IBRD	RW	0x0000_0000	32	Integer Baud Rate Register.
0x1028	UART1FBRD	RW	0x0000_0000	32	Fractional Baud Rate Register.
0x102C	UART1LCR_H	RW	0x0000_0000	32	Line Control Register.
0x1030	UART1CR	RW	0x0000_0300	32	Control Register.
0x1034	UART1IFLS	RW	0x0000_0012	32	Interrupt FIFO Level Select Register.
0x1038	UART1IMSC	RW	0x0000_0000	32	Interrupt Mask Set/Clear Register.
0x103C	UART1RIS	RO	0x0000_0000	32	Raw Interrupt Status Register.
0x1040	UART1MIS	RO	0x0000_0000	32	Masked Interrupt Status Register.
0x1044	UART1ICR	WO	-	32	Interrupt Clear Register.
0x1048	UART1DMACR	RW	0x0000_0000	32	DMA Control Register.
0x1FE0	UART1PeriphID0	RO	0x0000_0011	32	UART1 peripheral ID Register 0.
0x1FE4	UART1PeriphID1	RO	0x0000_0010	32	UART1 peripheral ID Register 1.
0x1FE8	UART1PeriphID2	RO	0x0000_0004	32	UART1 peripheral ID Register 2.
0x1FEC	UART1PeriphID3	RO	0x0000_0000	32	UART1 peripheral ID Register 3.
0x1FF0	UART1PCellID0	RO	0x0000_000D	32	UART1 component ID Register 0.
0x1FF4	UART1PCellID1	RO	0x0000_00F0	32	UART1 component ID Register 1.
0x1FF8	UART1PCellID2	RO	0x0000_0005	32	UART1 component ID Register 2.
0x1FFC	UART1PCellID3	RO	0x0000_00B1	32	UART1 component ID Register 3.

3.12 Third-party IP

The test chip implements third-party IP including control registers.

The test chip implements the following Cadence IP:

- QSPI controller (IP6514E):
  - Base memory address `0x4010_A000` in the Non-secure region.
  - Base memory address `0x5010_A000` in the Secure region.
- I^2C interface (IP6510):
  - I2C0: Base memory address `0x4010_4000` in the Non-secure region.
  - I2C0: Base memory address `0x5010_4000` in the Secure region.
  - I2C1: Base memory address `0x4010_5000` in the Non-secure region.
  - I2C1: Base memory address `0x5010_5000` in the Secure region.
- I^2S-MT/MR controller (IP6718E):
  - Base memory address `0x4010_6000` in the Non-secure region.
  - Base memory address `0x5010_6000` in the Secure region.
- Pulse Width Modulator IP (IP6512):
  - PWM0: Base memory address `0x4010_7000` in the Non-secure region.
  - PWM0: Base memory address `0x5010_7000` in the Secure region.
  - PWM1: Base memory address `0x4010_E000` in the Non-secure region.
  - PWM1: Base memory address `0x5010_E000` in the Secure region.
  - PWM2: Base memory address `0x4010_F000` in the Non-secure region.
  - PWM2: Base memory address `0x5010_F000` in the Secure region.
- SPI master interface (IP6524):
  - Base memory address `0x4010_3000` in the Non-secure region.
  - Base memory address `0x5010_3000` in the Secure region.

Contact your local Cadence representative for information about the QSPI, I^2C, I^2S, PWM, and SPI blocks.

3.13 SSE-200 subsystem debug system

The debug access interface of the SSE-200 provides access to three debug Access Ports within the debug subsystem. The ports provide access to the System Debug region of the memory map, the processor Debug Access Ports, and associated debug logic.

The base memory address of the Debug access interface is `0xF000_0000`.

The following table shows the address map for the debug access interfaces of the SSE-200 for the three Access Ports (APs) in the debug subsystem.

Table 3-72 Debug access region interface

Row ID	Address		Size	Region name	Description
	From	To			
1	`0x0000`	`0x00FF`	256B	SYSTEM APB-AP	Debug System Access APB-AP.
2	`0x0100`	`0x01FF`	256B	CPU0 AHB-AP	CPU0 Access AHB-AP.
3	`0x0200`	`0x02FF`	256B	CPU1 AHB-AP	CPU1 Access AHB-AP.
4	`0x0300`	`0xFFFF`	-	-	Reserved.

The Debug System APB-AP is used to access debug components that are in the debug subsystem, which includes components in the Debug element and components connected to the Debug APB Expansion Interface.

Table 3-73 System APB-AP address map

Row ID	Address		Size	Region name	Description
	From	To			
1	`0x0000_0000`	`0xEFFF_FFFF`	-	-	Reserved.
2	`0xF000_0000`	`0xF000_0FFF`	4KB	SYSCROM	Debug System CoreSight ROM.
3	`0xF000_1000`	`0xF000_1FFF`	256B	SYSFUNNEL	Debug System Trace Funnel.
4	`0xF000_2000`	`0xF000_2FFF`	-	SYSCTI	Debug System Cross Trigger Interface.
5	`0xF000_3000`	`0xF007_FFFF`	500KB	-	Reserved.
6	`0xF008_0000`	`0xF00F_FFFF`	512KB	Debug APB Expansion Interface	Debug APB Expansion Interface Region.
4	`0xF010_0000`	`0xFFFF_FFFF`	-	-	Reserved.

CPU0 AHB-AP is for CPU0 (Primary core) debug access and also for certification access. It also maps a CoreSight ROM and a *Granular Power Requester* (GPR).

The address map for CPU0 depends on the value of CERTDISABLED.

Note

Bit[16] of the System Security Control register, SCSECCTRL, indicates the value of CERTDISABLED. See *3.5.3 System control register block* on page 3-78 and the *Arm® CoreLink™ SSE-200 Subsystem for Embedded Technical Reference Manual*.

The following table shows the map for CPU0 when CERTDISABLED is LOW.

Table 3-74 CPU0 AHB-AP address map when CERTDISABLED is LOW

Row ID	Address		Size	Region name	Description
	From	To			
1	0x0000_0000	0x2FFF_FFFF	-	-	System memory access by the CPU0 debug access port.
2	0x3000_0000	0x3000_1FFF	8KB	CERTMEM	Certificate Access Memory region, residing in SRAM0. Write access is allowed and read data is masked to zero if CERTREADENABLED is LOW. Access bypasses the CPU core.
3	0x3000_2000	0xF000_7FFF	-	-	System memory access by the CPU0 debug access port.
2	0xF000_8000	0xF000_8FFF	4KB	CPU0CSROM	CPU0 Access CoreSight ROM.
3	0xF000_9000	0xF000_9FFF	4KB	CPU0GPR	CPU0 GPR
4	0xF000_A000	0xFFFF_FFFF	-	-	System memory access by the CPU0 debug access port.

The following table shows the map for CPU0 when CERTDISABLED is HIGH.

Table 3-75 CPU0 AHB-AP address map when CERTDISABLED is HIGH

Row ID	Address		Size	Region name	Description
	From	To			
1	0x0000_0000	0xF000_7FFF	-	-	System memory access by the CPU0 debug access port.
2	0xF000_8000	0xF000_8FFF	4KB	CPU0CSCROM	CPU0 Access CoreSight ROM.
3	0xF000_9000	0xF000_9FFF	4KB	CPU0GPR	CPU0 GPR.
4	0xF000_A000	0xFFFF_FFFF	-	-	System memory access by the CPU0 debug access port.

CPU1 AHB-AP is for CPU1 (Secondary core) debug access. It also maps a CoreSight ROM and a Granular Power Requester (GPR).

The following table shows the memory map for CPU1 AHB-AP.

Table 3-76 CPU1 AHB-AP address map

Row ID	Address		Size	Region name	Description
	From	To			
1	0x0000_0000	0xF000_7FFF	-	-	System memory access by the CPU1 debug access port.
2	0xF000_8000	0xF000_8FFF	4KB	CPU1SCROM	CPU1 Access CoreSight ROM.
3	0xF000_9000	0xF000_9FFF	4KB	CPU1GPR	CPU1 GPR.
4	0xF000_A000	0xFFFF_FFFF	-	-	System memory access by the CPU1 debug access port.

Appendix A
Signal descriptions

This appendix describes the signals present at the interface connectors.

It contains the following sections:

A.1 Arduino Shield connectors

Connectors on the Musca-A board provide one Shield expansion interface. The interface provides 16 digital I/O and six analog I/O. The digital and analog I/O operating voltage is 3V3 only.

Arduino Shield interface

The following figure shows the Arduino Shield interface connectors.

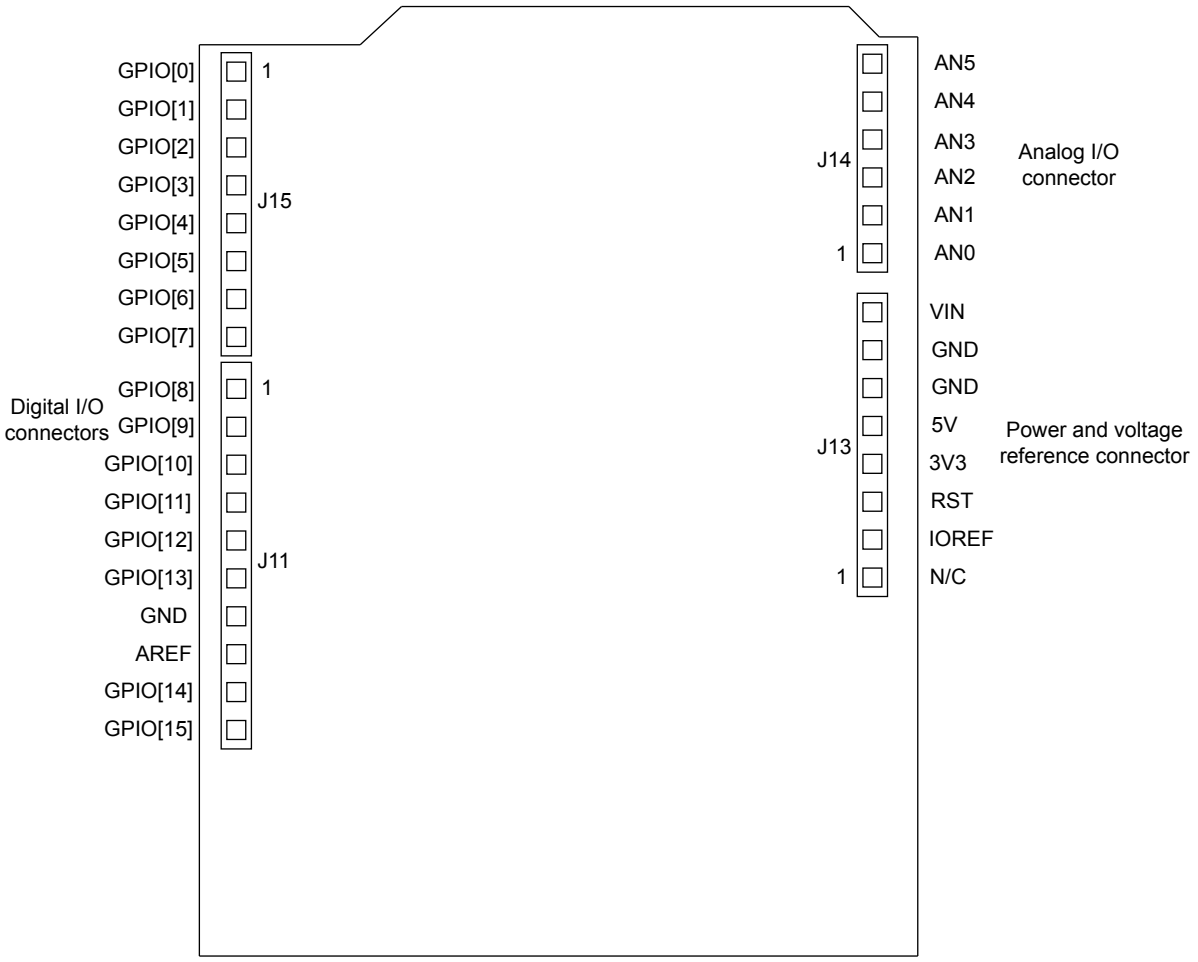

Figure A-1 Arduino Shield interface connectors

Digital I/O connectors, J11, J15

Connector J11 provides Shield digital I/O GPIO[15:8], and connector J15 provides Shield digital I/O GPIO[7:0]. Connector J11 also provides the analog I/O reference voltage.

The Serial Configuration Control (SCC) registers select one of the Shield interface GPIO pin functions sets, ALTF1 or ALTF2. See *3.9.1 IOMUX Registers* on page 3-89.

The following table shows the pin mappings for connector J15.

Table A-1 Shield digital I/O connector J15 signal list

Pin	Primary (Reset or powerup)	ALTF1	ALTF2	ALTF3
1	**GPIO[0]**	UART0 RxD	-	Reserved.
2	**GPIO[1]**	UART0 TxD	-	
3	**GPIO[2]**	MR_I2S_SD2	PWM0	
4	**GPIO[3]**	MR_I2S_WS	PWM1	
5	**GPIO[4]**	MR_I2S_SCK	PWM2	
6	**GPIO[5]**	MT_I2S_SD0	-	
7	**GPIO[6]**	MT_I2S_WS0	-	
8	**GPIO[7]**	MT_I2S1_SD1	-	

The following table shows the pin mappings for connector J11.

Table A-2 Shield digital I/O connector J11 signal list

Pin	Primary (Reset or powerup)	ALTF1	ALTF2	ALTF3
1	**GPIO[8]**	MT_I2S_WS1	-	Reserved.
2	**GPIO[9]**	MT_I2S_SCK	-	
3	**GPIO[10]**	SPI0 nSS	-	
4	**GPIO[11]**	SPI0 MOSI	-	
5	**GPIO[12]**	SPI0 MISO	-	
6	**GPIO[13]**	SPI0 SCK	-	
7	**GND**	**GND**	-	
8	N/C	N/C	-	
9	**GPIO[14]**	I2C0 Data (SDA)	TESTCLK	
10	**GPIO[15]**	I2C0 Clock (SCL)	-	

Shield analog I/O connector J14

Connector J14 provides six analog I/O for the expansion Shield.

The following table shows the pin mapping for connector J14.

Table A-3 Analog I/O connector J14 signal list

Pin	Signal
1	**AN[0]**
2	**AN[1]**
3	**AN[2]**
4	**AN[3]**

Table A-3 Analog I/O connector J14 signal list (continued)

Pin	Signal
5	**AN[4]**
6	**AN[5]**

Shield power and voltage reference connector J13.

Connector J13 provides power and voltage references for the expansion Shield.

The following table shows the pin mapping for connector J13.

Table A-4 Shield power and voltage reference connector J13 signal list

Pin	Signal
1	NC
2	**IOREF**
3	**CB_nRST**
4	**3V3**
5	**5V**
6	**GND**
7	**GND**
8	**VIN**

Related references

1.3 Location of components on page 1-14

A.2 Debug connector

The Musca-A board provides one debug connector that provides access to the CoreSight block and Serial Wire Debug (SWD) on the Musca-A test chip.

The following figure shows the debug connector.

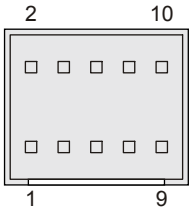

Figure A-2 Debug connector

The following table shows the pin mapping for the debug connector J4.

Table A-5 CoreSight debug connector, J4, pin mapping

Pin	Signal	Pin	Signal
1	**3V3**	2	**SWDIO/CS_TMS**
3	**GND**	4	**SWDCLK/CS_TCK**
5	**GND**	6	**SWO/CS_TDO**
7	NC	8	NC/**CS_TDI**
9	**CS_nDET**	10	**CS_nSRST**

Note

- Pins 2, 6, 8, 9, and 10 have pullup resistors to **3V3**.
- Pin 4 has a pulldown resistor to **GND**.

Related references

1.3 Location of components on page 1-14

A.3 USB connector

The Musca-A board provides one mini-B USB connector that connects to the DAPLink controller which enables access to the CoreSight block in the Musca-A test chip. The connector also enables external 5V power to the board.

The following figure shows the access USB connector.

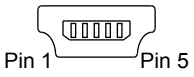

Figure A-3 Mini-B USB connector

The following table shows the pin mapping for the USB connector J9.

Table A-6 USB connector, J9, pin mapping

Pin	Signal	Pin	Signal
1	**5V**	2	**DATA-**
3	**DATA+**	4	**ID**
5	**GND**	6	**GND_EARTH**

Note

The **GND_EARTH** connection is the casing of the mini-B connector.

Appendix B
Specifications

This appendix contains electrical specifications of the Musca-A board.

It contains the following section:

B.1 Electrical specifications

The electrical specifications of the Musca-A board are as follows:

See *2.7 Power* on page 2-31 for information on the Musca-A board power supply rails and maximum current loads.

Appendix C
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

C.1 Revisions

The following table lists the technical changes between released issues of this book.

Table C-1 Issue 101107_0000_00

Change	Location	Affects
No changes, first release.	-	-

Table C-2 Differences between issue 101107_0000_00 and issue 101107_0000_01

Change	Location	Affects
SPI available as master interface only.	*1.2 About the Musca-A board* on page 1-13 *2.1 Board hardware* on page 2-17 *2.2 Musca-A test chip* on page 2-20 *2.9 Arduino Shield expansion* on page 2-35 *3.12 Third-party IP* on page 3-117	All board versions
UARTs do not support hardware flow control.	*1.2 About the Musca-A board* on page 1-13 *2.1 Board hardware* on page 2-17 *2.2 Musca-A test chip* on page 2-20 *2.3 Software, firmware, board, and tools setup* on page 2-24 *2.9 Arduino Shield expansion* on page 2-35 *3.11 UART registers* on page 3-115	All board versions
Expanded clock section.	*2.5 Clocks* on page 2-27	All board versions
Added descriptions for FCLK_DIV and SYSCLK_DIV system control registers.	*FCLK_DIV Register* on page 3-80 *SYSCLK_DIV Register* on page 3-81	All board versions
Correction to interrupt table rows IRQ[72] and IRQ[73].	*3.4.4 Interrupts* on page 3-71	All board versions
GPIO registers only accessible in the Secure region.	*3.10 GPIO registers* on page 3-113	All board versions
Unaligned access to Code SRAM is not supported.	*3.2 Memory map* on page 3-42	All board versions
OTP registers emulate One-Time Programming.	*3.8 One-Time Programmable (OTP) secure registers* on page 3-88	All board versions
Added information about higher power available to Arduino Shields.	*2.7 Power* on page 2-31	varC boards
Added part numbers and board variant ordering information.	*1.4 Part numbers and ordering information* on page 1-15	All board versions
Added Caution about Warm reset resetting Real Time Clock.	*3.6 Real Time Clock (RTC) registers* on page 3-83	All board versions
Added information on how to update DAPLink firmware from a Linux/Mac OS.	*2.3 Software, firmware, board, and tools setup* on page 2-24	All board versions

Table C-3 Differences between issue 101107_0000_01 and issue 101107_0000_02

Change	Location	Affects
Updated CE Conformance Notice.	*Conformance Notices* on page 3	All board versions