
Application Note **AN543**

Arm® Corstone™-700 for MPS3

Non-Confidential

The Arm logo, consisting of the word "arm" in a lowercase, bold, sans-serif font.

Arm® Corstone™-700 for MPS3

Copyright © 2020 Arm Limited or its affiliates. All rights reserved.

Release Information

The following changes have been made to this Application Note.

Change History			
Date	Issue	Confidentiality	Change
13 February 2020	A	Non-Confidential	First version
26 February 2020	B_00	Non-Confidential	Issue B First draft 00
31 March 2020	B_01	Non-Confidential	Block Diagram updated
16 November 2020	C	Non-Confidential	Block Diagram updated ArmDS 2019.1 > 2020.1 Reference documents version update

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2020 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

LICENCE GRANTS

THE END USER LICENCE AGREEMENT FOR THE ARM SYSTEM OR SUBSYSTEM FOR AN ARM FPGA PROTOTYPING BOARD (“THE LICENCE”), LES-PRE-21902, DEFINES THE LICENCE GRANTS.

DELIVERABLES

Part A

Hardware Binaries:

Encrypted FPGA bitstream file containing the Corstone SSE-700 Subsystem and other Arm technology.

Software Binaries:

Motherboard Configuration Controller binary, including Keil USB and SD card drivers, and Analog Devices FMC EEPROM reader.
Self-test binary.

Documentation:

Documentation, provided as PDF

Part B

Example Code:

Platform initialisation source code
Platform specific libraries and source code
Selftest example source code
Arm source code portions of the Self-test

Part C

None

Part D

None

Contents

Arm® Corstone™-700 for MPS3

1 Conventions and Feedback	1-6
2 Preface	2-8
2.1 Purpose of this Application Note	2-8
2.2 References	2-8
2.3 Terms and Abbreviations	2-8
2.4 IP Version Details	2-10
2.5 Encryption Key	2-10
3 Overview	3-11
3.1 System Block Diagram	3-12
3.2 EXTSYS0 - External System 0	3-13
3.3 Components	3-14
3.4 Memory Map Overview	3-18
4 Programmers Model	4-22
4.1 Xilinx Internal BRAM	4-22
4.2 XNVM QSPI Flash	4-22
4.3 OCVMM DDR4	4-22
4.4 UART Support	4-22
4.5 SSE-700 Implementation	4-23
4.6 SSE-700 Secure Enclave Cryptographic Accelerator Model	4-24
4.7 EXTSYS0	4-28
4.8 SBCon I ² C	4-31
4.9 Audio I ² S	4-31
4.10 SMM Registers	4-33
4.11 LEDs	4-34
4.12 FPGA Serial Communication Controller (SCC)	4-34
4.13 FPGA System Control and I/O Registers	4-36
5 Clock Architecture	5-37
5.1 Clocks	5-37
6 ZIP Bundle Description	6-39
6.1 Overall Structure	6-39
6.2 Documentation	6-39
6.3 MPS3 Board Revision and Support	6-39
6.4 Bundle Directory Tree/Structure	6-40
7 Software and Debug	7-42
7.1 Overview	7-42
7.2 Example Software Project	7-42
7.3 Debug Support	7-43
7.4 Building the Software	7-51
7.5 Establishing a Debug Session	7-54
7.6 Modifying the SSE-700 Boot Register	7-57
8 Using AN543 on the MPS3 Board	8-59
8.1 Pre-Requisites	8-59

8.2 Loading the Boardfiles onto the MPS3 SD Card.....	8-59
8.3 UART Serial Ports	8-60
8.4 UART Serial Port Terminal Emulator Settings	8-60
8.5 MPS3 USB Serial Port Drivers for Windows	8-60
8.6 Running the FPGA image file on the MPS3 Board	8-60
8.7 MCC Debug UART – Serial Port 0.....	8-63
8.8 Preload of the SE ROM, EXTSYS0 Code SRAM and the QSPI Flash.....	8-66
8.9 Supported Preload SE ROM, EXTSYS0 Code SRAM and Flash File Types	8-67
8.10 Supported Preload SE ROM, EXTSYS0 Code SRAM and Flash File Naming Format	8-68

1 Conventions and Feedback

The following describes the typographical conventions and how to give feedback:

Typographical conventions

The following typographical conventions are used:

- | | |
|--------------------------------------|---|
| <code>monospace</code> | Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code. |
| <u><code>monospace</code></u> | Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name. |
| <code>monospace</code> <i>italic</i> | Denotes arguments to monospace text where the argument is to be replaced by a specific value. |
| <code>monospace</code> bold | Denotes language keywords when used outside example code. |
| <i>italic</i> | 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. |

Feedback on this product

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

- Your name and company.
- The serial number of the product.
- Details of the release you are using.
- Details of the platform you are using, such as the hardware platform, operating system type and version.
- A small standalone sample of code that reproduces the problem.
- A clear explanation of what you expected to happen, and what actually happened.
- The commands you used, including any command-line options.
- Sample output illustrating the problem.
- The version string of the tools, including the version number and build numbers.

Feedback on documentation

If you have comments on the documentation, e-mail errata@Arm.com. Give:

- The title *Application Note AN543 Arm® Corstone™-700 for MPS3*.
- The number, DAI 0543C.
- If viewing online, the topic names to which your comments apply.
- If viewing a PDF version of a document, the page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Arm periodically provides updates and corrections to its documentation on the Arm Information Center, together with knowledge articles and *Frequently Asked Questions* (FAQs).

Other information

- Arm Documentation, <https://developer.arm.com/documentation/>
- Arm Technical Support, <https://www.arm.com/support/technical-support>
- Arm Support, <https://www.arm.com/support>
- Arm Glossary, <https://developer.arm.com/documentation/aeg0014/g/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.

2 Preface

2.1 Purpose of this Application Note

This document describes the features and functionality of AN543. AN543 is an FPGA implementation based on the Corstone SSE-700 Subsystem, which is extended to support a single external system and hardware peripherals.

2.2 References

- *Arm® MPS3 FPGA Prototyping Board Technical Reference Manual*
- *Arm® Cortex®-M System Design Kit Technical Reference Manual (DDI 0479)*
- *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual (101870_0000_03_en)*
- *Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual (101418_0100_07_en)*
- *Arm® CoreLink™ SSE-050 Subsystem Technical Reference Manual (100918_0001_00_en)*

2.3 Terms and Abbreviations

ADB	AMBA Domain Bridge
AHB	Advanced High-performance Bus.
APB	Advanced Peripheral Bus.
AXI	Advanced Extensible Interface.
BRAM	FPGA Block RAM
CA	Cryptographic Accelerator
CMSDK	Cortex-M System Design Kit
CPU	Central Processing Unit
CVM	On-Chip Volatile Memory
DPRAM	Dual Port RAM
EIS	Engineering Implementation Specification
EXPMST	Expansion Master
EXTSYS0	External System 0
FPGA	Field Programmable Gate Array
FW	Firewall
GIC	Generic Interrupt Controller
JTAG	Joint Test Action Group
KB	Kilo Byte.

MB	Mega Byte.
MCC	Motherboard Configuration Controller.
MHU	Message Handling Unit
MIG	Memory Interface Generator
MPS3	Microcontroller Prototyping System 3
NIC	Network Interconnect
OCVM	Off-Chip Volatile Memory
RAM	Random Access Memory.
RAZ	Read as Zero
ROM	Read Only Memory
RTC	Real time Clock
RTL	Register Transfer Level
SCB	Security Control Bits
SCC	Serial Configuration Controller
SE	Secure Enclave
SMM	Soft Macrocell Model
SoC	System on Chip
SWD	Serial Wire Debug
TBD	To Be Defined
TRM	Technical Reference Manual
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WI	Write Ignored
XNVM	eXecute-in-place Non-volatile Memory (XNVM)

2.4 IP Version Details

Version	Descriptions
r1p1	Arm® CoreLink™ NIC-400
r1p1	CMSDK
r1p0	Corstone™ SSE-700 Subsystem
r0p0	Corstone™ SSE-700 Secure Enclave
r0p1	CoreLink™ SSE-050 Subsystem

Table 2-1 : IP Versions

2.5 Encryption Key

Arm supplies the MPS3 Prototyping Board with a decryption key programmed into the FPGA. This key enables loading of prebuilt encrypted images.

Note

The FPGA programming file that is supplied as part of the bundle is encrypted.

Caution

A battery supplies power to the key storage area of the FPGA. Any keys stored in the FPGA might be lost when battery power is lost. If this happens you must return the board to Arm for reprogramming of the key.

3 Overview

This SMM is based on the Corstone™ SSE-700 Subsystem. The Subsystem is then extended to add a single external system and peripherals to support software development.

3.1 System Block Diagram

The diagram below shows a high-level view of the full MPS3 Prototyping Board Corstone™ SSE-700 Subsystem implementation.

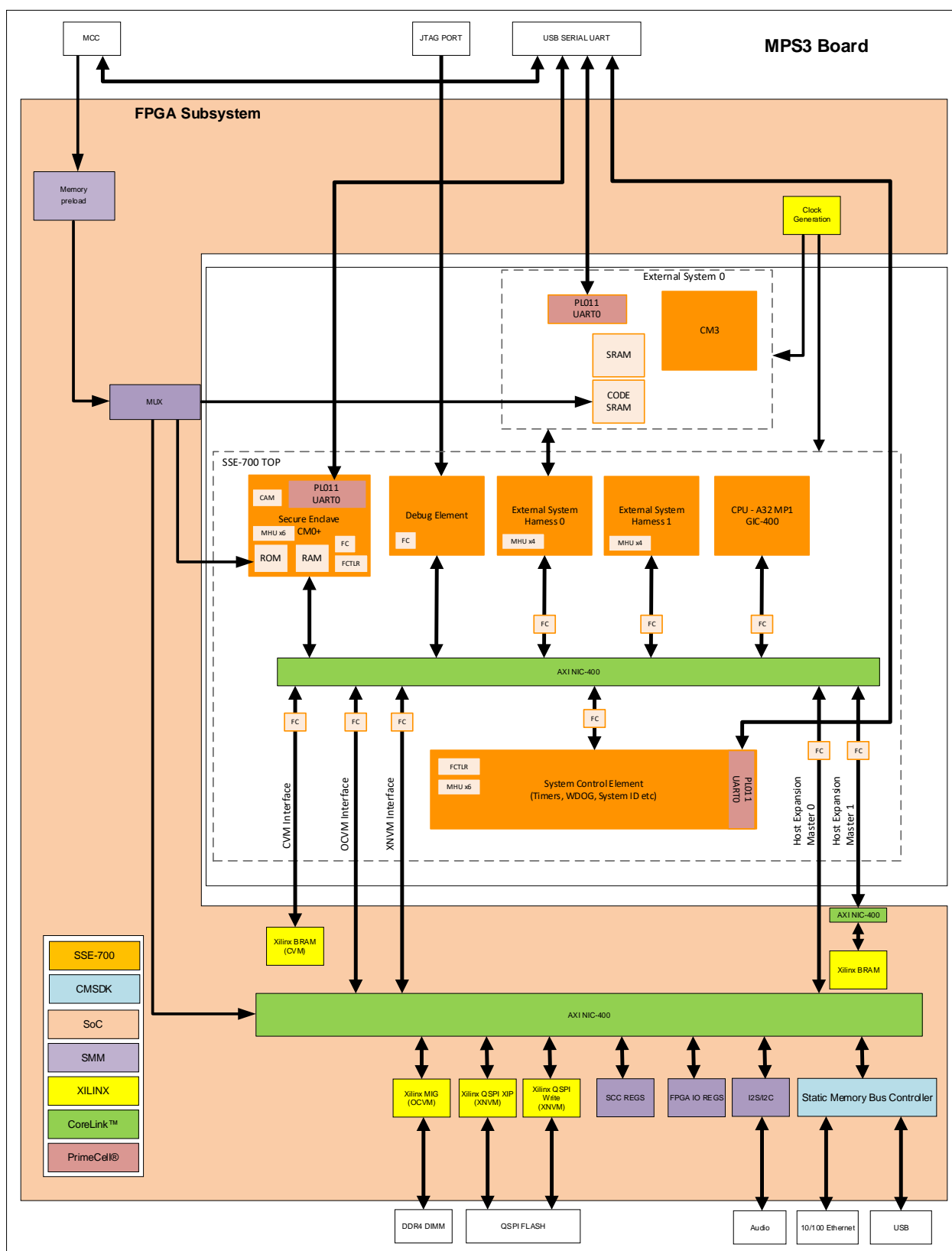


Figure 3-1 : System Overview

3.2 EXTSYS0 - External System 0

The implementation supports a single external system populating one of the EXTSYS0 sockets in SSE-700. The External System 0 is based on a modified SSE-050, a Cortex-M3 based subsystem.

3.2.1 (EXTSYS0) External System 0 Block Diagram

The following diagram shows the External System 0, which is based on SSE-050 with minor modifications.

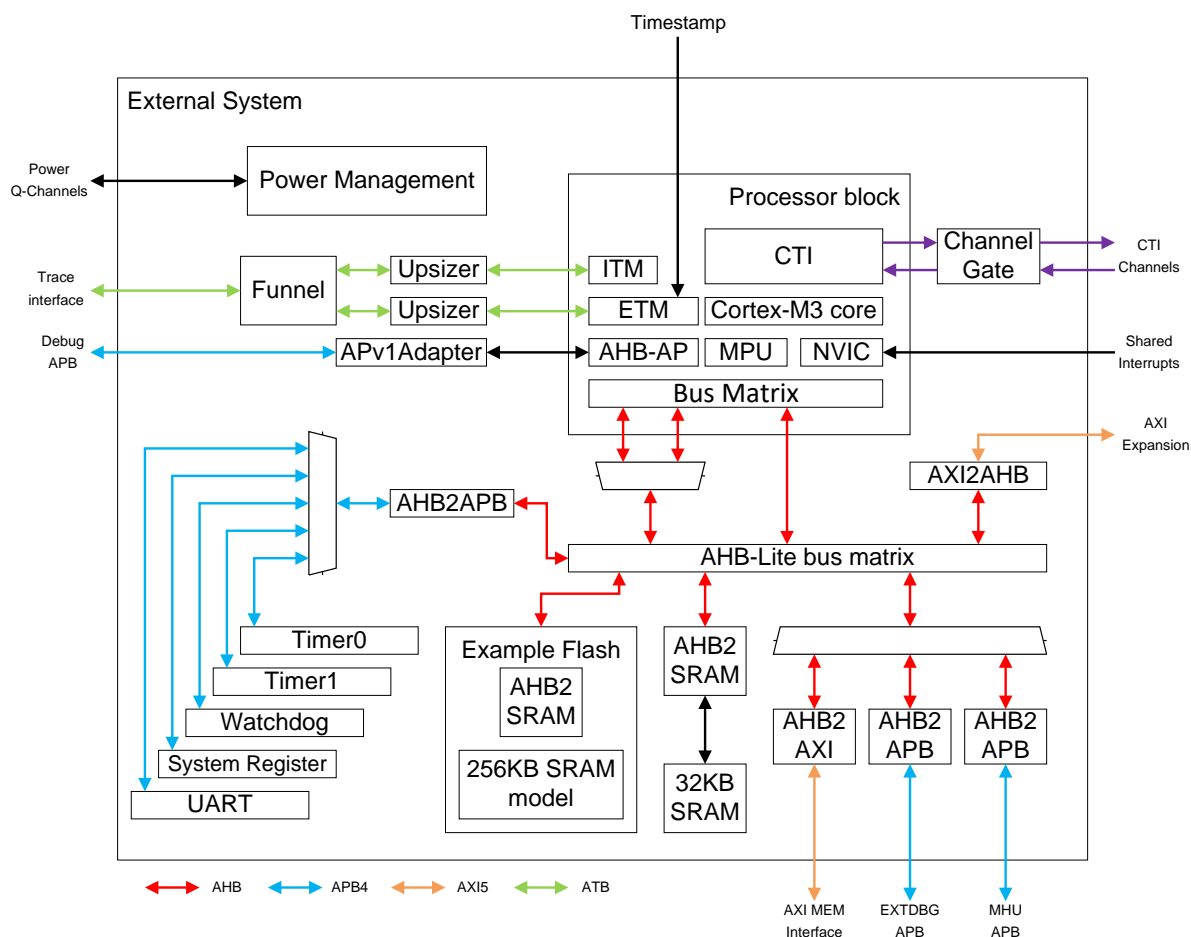


Figure 3-2 : EXTSYS0 System Overview

3.3 Components

The following sections detail the components used in the FPGA Subsystem.

3.3.1 SSE-700 Subsystem

The following sections detail the components used in the SSE-700 Subsystem.

3.3.1.1 Secure Enclave

The following sections detail components and models used in the Secure Enclave.

3.3.1.1.1 Secure Enclave ROM Model and Preload Support

The Secure Enclave ROM is implemented as a *Dual Port RAM* (DPRAM) within the FPGA. The Secure Enclave boots from this ROM. The implementation within this application note supports:

- Default boot ROM embedded as a preload on the SE ROM in the FPGA programming (.bit) file.
- Debug access to the SE ROM/RAM. In cases where the MCC does not preload the memory, the FPGA image implements a pre-load of this SE ROM to enable the processor to reach a state where debug is accessible.
- Preload of ROM from a file on the MPS3 SDCARD. The MCC holds the target SE Cortex-M0+ in reset and CPUWAIT asserted during preload. The MCC then releases the reset, then CPUWAIT on the SE Cortex-M0+ to enable the SE Cortex-M0+ to boot from the SE ROM.

3.3.1.1.2 Secure Enclave Cryptographic Accelerator Model

The SE Cryptographic Accelerator is implemented as a model. See section 4.6 for further details of the model.

3.3.1.2 Single External System (EXTSYS0)

External System 0 includes the following:

- Modified SSE-050 Cortex-M3 based System with CoreSight™ SoC-600 support and Cortex-M3 ETM.
- 256KB Code SRAM
- Code SRAM replaced with DPRAM to provide support for preload from MPS3 MCC.
- 32KB SRAM
- PL011 UART
- Timers, Watchdog, and System Register.
- AXI, AHB, and APB interconnect to support connection to different SSE-700 interfaces.
- ATB network modules to enable connection all trace sources to the SSE-700.
- Channel Gate to be able to limit cross-trigger functionality based on the SE Life Cycle State.
- Addition of PL011 UART to APBTARGEXP2.

3.3.1.2.1 EXTSYS0 – Cortex-M3 Code SRAM Preload Support

The EXTSYS0 code RAM is implemented as a DPRAM within the FPGA. The Cortex-M3 boots from this RAM. The implementation within this application note supports:

- Default boot RAM embedded as a preload on the Cortex-M3 Code SRAM in the FPGA programming (.bit) file.
- Debug access to the Cortex-M3 Code SRAM. In cases where the MCC does not preload the memory as in (a), the FPGA image implements a pre-load of this RAM to enable the processor to reach a state where debug is accessible.
- Preload of Cortex-M3 Code RAM from a file on the MPS3 SDCARD. The MCC holds the target SE Cortex-M0+ CPUWAIT asserted during preload. As the Cortex-M0+ is held in CPUWAIT, the EXTSYS0 CPUWAIT is held asserted as the reset default. The MCC then releases the CPUWAIT on the SE Cortex-M0+ to enable the SE Cortex-M0+ to boot from the SE ROM. The EXTSYS0 CPUWAIT must be de-asserted by software in order to boot the EXTSYS0 from the Cortex-M3 Code SRAM.

3.3.1.3 Host System CPU Configuration

The Host System CPU is a Cortex-A32 (MP1 single core), which is configured with the following parameters:

- NEON_FP: "FALSE" (not present)
- CRYPTO: "FALSE" (not present)
- L1_ICACHE_SIZE: "32KB"
- L1_DCACHE_SIZE: "32KB"
- L2_CACHE_SIZE: "512KB"

3.3.1.4 Host System Configuration

The Host System is configured with the following parameters:

- SSE-700 Expansion SLV0 & SLV1 tied off/un-used.
- Shared Interrupts: 64
- MHU Channels:
 - 32 channels between SE and Host
 - 4 channels between Host and EXTSYS0
 - 4 Channels between SE and EXTSYS0
 - 1 Channel for all MHUs related to External System 1 (External system 1 is not implemented)

3.3.1.5 Host FW Configuration

The following table shows the FW Configuration settings.

FW Configuration	Value
XNVM_RSE_LVL	1
XNVM_NUM_RGN	32
CVM_RSE_LVL	1
CVM_NUM_RGN	32
DBG_NUM_RGN	8
EXTSYS0_NUM_RGN	8
EXTSYS1_NUM_RGN	8
EXPSLV0_NUM_RGN	8
EXPSLV1_NUM_RGN	8
EXPMST0_PE_LVL	2
EXPMST0_RSE_LVL	1
EXPMST0_NUM_RGN	8
EXPMST0_MXRS	29
EXPMST1_PE_LVL	2
EXPMST1_RSE_LVL	1
EXPMST1_NUM_RGN	8
EXPMST1_MXRS	29
OCVM_RSE_LVL	1
OCVM_NUM_RGN	32
HOST_FC_ERR_RESP_DEF_32	0xDEADDEADDEADDEAD
HOST_FC_ERR_RESP_DEF_64	0xDEADDEAD
HOST_FC_ERR_RESP_DEF_128	0xDEADDEADDEADDEADDEADDEADDEADDEADDEAD

Table 3-1 FW Configuration

3.3.2 NIC-400

The NIC-400 connects the SSE-700 Subsystem to the FPGA Subsystem peripherals.

3.3.3 Xilinx QSPI Controller

The QSPI controllers connect the QSPI flash interface to the NIC-400 implemented in the FPGA Subsystem.

3.3.4 Xilinx MIG – DDR4

The Xilinx MIG controller connects the DDR4 Memory interface to the NIC-400 that is implemented in the FPGA Subsystem.

3.3.5 DDR4 (OCVM) SODIMM EEPROM SBCon

The SMM implements a single SBCon I²C module to read the EEPROM on the DDR4 SODIMM Module. See section 4.8 for more details.

3.3.6 Debug UART Peripheral

The following UARTs are accessible over USB on the MPS3 board:

- Secure Enclave UART0 (PrimeCell PL011).
- SSE-700 subsystem UART0 (PrimeCell PL011).
- EXTSYS0 UART0. A single PrimeCell PL011 is implemented (connected to APBTARGEXP2).
- MCC debug UART. The MPS3 board supports a debug serial port connection to the MCC that enables functions such as reset to be initiated by a host terminal. See section 8.7 on MCC Debug UART for more details.

3.3.7 Flash

8MB of QSPI flash is supported on the platform and is connected to the SSE-700 XNVM port. The QSPI flash can be preloaded by the MCC and is accessible from the SE Cortex-M0+, Host Cortex-A32 CPU and EXTSYS0. The QSPI Xilinx controller is utilized to support QSPI flash.

3.3.8 USB

The implementation connects to an ST Microelectronics Hi-Speed USB OTG controller (ISP1763) device through a static memory interface.

3.3.9 10/100 Ethernet

The implementation connects an SMSC LAN9220 device through a static memory interface. The device is accessible from the SSE-700 Expansion MSTR0 interface.

3.3.10 Audio

The SMM implements a single I²S and a single I²C module directly connected to the MPS3 back panel audio sockets. I²S is used for data and I²C is used for control. The device is accessible from the SSE-700 Expansion MSTR0 interfaces.

3.3.10.1 SBCon I²C

A single two-wire I²C SBCon module supports configuration of the Cirrus Logic, Stereo CODEC (CS42L52) on the MPS3 board. See section 4.8 for more details.

3.3.10.2 I²S Interface

A single I²S module connects the internal APB bus to the external I²S Audio CODEC. See section 4.9 for more details.

3.3.11 FPGA Version ID Register

A version register is supported and implemented within the SCC registers. The location of the SCC registers is defined in the memory map.

3.4 Memory Map Overview

3.4.1 SSE-700 Host System Memory Map

The memory map implementation aligns with SSE-700 subsystem Host System memory map. The memory map is expanded to show the supported MPS3 peripherals in the Expansion Master Regions and their mapping.

See *Arm® Corstone™ SSE-700 Technical Reference Manual* for information on the SSE-700 subsystem Host System memory map.

ROW ID	Address From	To	Size	Region Name	Description
1	0x0000_0000	0x0000_0FFF	4KB	Boot Register	
2	0x0000_1000	0x000F_FFFF	1024KB	Reserved	Reserved
3	0x0010_0000	0x00FF_FFFF	15MB	Reserved	Reserved
4	0x0100_0000	0x01FF_FFFF	16MB	Reserved	Reserved
5	0x0200_0000	0x023F_FFFF	4MB	CVM	Volatile Memory implemented as BRAM
6	0x0240_0000	0x03FF_FFFF	28MB	Reserved	CVM space that is not implemented
7	0x0400_0000	0x07FF_FFFF	64MB	Reserved	Reserved
8	0x0800_0000	0x087F_FFFF	8MB	XNVM	eXecute-in-place Non-volatile Memory (Note only 8MB of QPSI Flash is provisioned on the MPS3 board)
9	0x0880_0000	0x09FF_FFFF	24MB	XNVM	Reserved
10	0x0A00_0000	0x0A00_FFFF	64KB	XNVM	QSPI Flash Write
11	0x0A01_0000	0x0FFF_FFFF	95MB	Reserved	XNVM space that is not implemented
12	0x1000_0000	0x19FF_FFFF	160MB	Debug	
13	0x1A00_0000	0x3FFF_FFFF	608MB	Host Peripherals	
14	0x4000_0000	0x4000_FFFF	64KB	EXPMST0	FPGA – SCC Registers (note only 4KB of registers implemented)
15	0x4001_0000	0x4001_FFFF	64KB	EXPMST0	FPGA – I/O Registers (note only 4KB of registers implemented)
16	0x4002_0000	0x4002_FFFF	64KB	EXPMST0	Audio I ² S
17	0x4003_0000	0x4003_FFFF	64KB	EXPMST0	SBCon Audio I ² C (Configuration)
18	0x4004_0000	0x4004_FFFF	64KB	EXPMST0	SBCon DDR4 (OCVM) SODIMM EEPROM
19	0x4005_0000	0x400F_FFFF	704KB	EXPMST0	Reserved
20	0x4010_0000	0x401F_FFFF	1MB	EXPMST0	Ethernet
21	0x4020_0000	0x402F_FFFF	1MB	EXPMST0	USB

ROW ID	Address		Size	Region Name	Description
	From	To			
22	0x4030_0000	0x5FFF_FFFF	1021MB	EXPMST0	Reserved
23	0x6000_0000	0x6000_FFFF	64KB	EXPMST1	Volatile Memory implemented as BRAM
24	0x6001_0000	0x7FFF_FFFF	511MB	EXPMST1	Reserved
25	0x8000_0000	0xFFFF_FFFF	2GB	OCVM	Off Chip Volatile Memory implemented as off chip DDR4

Table 3-2 : SSE-700 Host System Memory Map

3.4.2 SSE-700 AON Expansion Memory Map

The AON expansion interface resides with the SSE-700 Host Peripheral region in the SSE-700 Host System. The implementation supports an RTC within this region. The mapping details can be found below.

ROW ID	Address		Size	Region Name	Description
	From	To			
1	0x1A60_0000	0x1A60_0FFF	4KB	RTC	PrimeCell RTC (PL031)
2	0x1A60_1000	0x1A6F_FFFF	1020KB	RESERVED	Default slave generates an error response

Table 3-3 : SSE-700 AON Expansion Memory Map

3.4.3 SSE-700 Secure Enclave Memory Map

See the *Arm® Corstone™ SSE-700 Secure Enclave TRM* for memory map details.

3.4.4 EXTSYS0 Memory Map

The following table shows the EXTSYS0 memory map.

ROW ID	Address		Size	Region Name	Description
	From	To			
1	0x0000_0000	0x0003_FFFF	256KB	TARGFLASH0	Code Memory
2	0x0004_0000	0x1FFF_FFFF	524032KB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
3	0x2000_0000	0x2000_7FFF	32KB	TARGSRAM0	SRAM0
4	0x2000_8000	0x3FFF_FFFF	524256KB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
5	0x4000_0000	0x4000_0FFF	4KB	TARGAPB0	Timer0
6	0x4000_1000	0x4000_1FFF	4KB	TARGAPB1	Timer1
7	0x4000_2000	0x4000_2FFF	4KB	TARGAPB2	EXTSYS0 UART
8	0x4000_3000	0x4000_3FFF	4KB	TARGAPB3	RESERVED (RAZ/WI)
9	0x4000_4000	0x4000_7FFF	16KB	TARGAPB<4-7>	RESERVED (RAZ/WI)
10	0x4000_8000	0x4000_8FFF	4KB	TARGAPB8	Watchdog
11	0x4000_9000	0x4000_9FFF	4KB	TARGAPB9	RESERVED (RAZ/WI)
12	0x4000_A000	0x4000_AFFF	4KB	TARGAPB10	RESERVED (RAZ/WI)
13	0x4000_B000	0x4000_BFFF	4KB	TARGAPB11	System register
14	0x4000_C000	0x4000_FFFF	16KB	TARGAPB<12-15>	RESERVED (RAZ/WI)
15	0x4001_0000	0x400F_FFFF	960KB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
16	0x4010_0000	0x4010_0FFF	4KB	TARGEXP1	HES MHU0 Receiver – EXTSYS0 MHU interface
17	0x4010_1000	0x4010_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
18	0x4011_0000	0x4011_0FFF	4KB	TARGEXP1	ESH MHU0 Sender EXTSYS0 MHU Interface
19	0x4011_1000	0x4011_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
20	0x4012_0000	0x4012_0FFF	4KB	TARGEXP1	HES MHU1 Receiver – EXTSYS0 MHU interface
21	0x4012_1000	0x4012_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
22	0x4013_0000	0x4013_0FFF	4KB	TARGEXP1	ESH MHU1 Sender – EXTSYS0 MHU interface
23	0x4013_1000	0x4013_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
24	0x4014_0000	0x4014_0FFF	4KB	TARGEXP1	SEES MHU0 Receiver – EXTSYS0 MHU interface
25	0x4014_1000	0x4014_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
26	0x4015_0000	0x4015_0FFF	4KB	TARGEXP1	ESSE MHU0 Sender – EXTSYS0 MHU interface
27	0x4015_1000	0x4015_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
28	0x4016_0000	0x4016_0FFF	4KB	TARGEXP1	SEES MHU1 Receiver – EXTSYS0 MHU interface

ROW ID	Address		Size	Region Name	Description
	From	To			
29	0x4016_1000	0x4016_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
30	0x4017_0000	0x4017_0FFF	4KB	TARGEXP1	ESSE MHU1 Sender – EXTSYS0 MHU interface
31	0x4017_1000	0x4017_FFFF	60KB	TARGEXP1	RESERVED EXTSYS0 MHU interface
32	0x4018_0000	0x43FF_FFFF	64000KB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
33	0x4400_0000	0x447F_FFFF	8MB	TARGEXP1	EXTSYS0 EXT DBG Interface of the Harness
34	0x4480_0000	0x5FFF_FFFF	440MB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
35	0x6000_0000	0x9FFF_FFFF	1GB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
36	0xA000_0000	0xA000_FFFF	64KB	TARGEXP0	Default Slave (Error response)
37	0xA001_0000	0xDFFF_FFFF	1048512KB	TARGEXP0	EXTSYS0 MEM Interface of the Harness
38	0xE000_0000	0xE000_0FFF	4KB	Internal PPB	ITM
39	0xE000_1000	0xE000_1FFF	4KB	Internal PPB	DWT
40	0xE000_2000	0xE000_2FFF	4KB	Internal PPB	FPB
41	0xE000_3000	0xE000_DFF	44KB	Internal PPB	RESERVED
42	0xE000_E000	0xE000_EFFF	4KB	Internal PPB	SCS (System Control space (NVI, SysTick, MPU))
43	0xE000_F000	0xE003_FFFF	196KB	Internal PPB	RESERVED
44	0xE004_0000	0xE004_0FFF	4KB	External PPB	RESERVED (Returns slave error)
45	0xE004_1000	0xE004_1FFF	4KB	External PPB	ETM
46	0xE004_2000	0xE004_2FFF	4KB	External PPB	CTI
47	0xE004_4000	0xE00F_EFFF	762KB	External PPB	RESERVED (Returns slave error)
48	0xE00F_F000	0xE00F_FFFF	4KB	Internal PPB	ROM table

Table 3-4 : EXTSYS0 Memory Map

4 Programmers Model

4.1 Xilinx Internal BRAM

BRAM is used to implement:

- The Secure Enclave ROM (32KB), which is implemented as Dual Port RAM. This is the primary boot memory. See the memory map for mapping details.
- The Secure Enclave RAM (128KB). See the memory map for mapping details.
- The CVM 4MB RAM. See the memory map for mapping details.
- The Secure Enclave Cryptographic Accelerator Model RAM. See section 4.6.2 for mapping details.
- The EXTSYS0 Code SRAM (256 KB). See the memory map for mapping details.
- EXTSYS0 SRAM (32KB)
- Host Cortex-A32 CPU L1 and L2 cache
- Firewall RAM
- 1KB RAM in the EXPMST1 address space. See the memory map for mapping details.
- Other RAMs required within SSE-700.

4.2 XNVM QSPI Flash

The XNVM 8MB memory is implemented using external QSPI flash memory which is accessed over a QSPI interface (implemented using a Xilinx QSPI controller). See the memory map for mapping details.

4.3 OCVM DDR4

The MPS3 board provides a 4GB DDR module. The SMM in this implementation supports 2GB of this external 4GB DDR module.

4.4 UART Support

The PrimeCell PL011 UART is implemented within the SSE-700 Subsystem and EXTSYS0. The UARTs are physically accessible through the USB interface on the MPS3 board. See the memory map for information on the USB interface.

This implementation provides the following UARTs:

- SSE-700 Host UART0
The Host Cortex-A32 MP1 CPU directs its UART output to this port when running the preloaded RAM FPGA image.
- SSE-700 SE UART
The SE Cortex-M0+ directs its UART output to this port.
- SSE-700 EXTSYS0 UART
The Cortex-M3 directs its UART output to this port.

4.5 SSE-700 Implementation

4.5.1 SSE-700 Power Control

Note

Power control is not currently supported in the r0p0 Corstone™ SSE-700 Subsystem.

Once supported the function of the following power states (within this example design, once the features are verified) are defined as:

- **FUNC_RET** – Logic in reset, volatile memory may retain contents (implementation dependent)
- **MEM_RET** – Logic in reset, volatile memory may retain contents (implementation dependent).
- **OFF** – Logic in reset, volatile memory may retain contents (implementation dependant).

4.5.2 SSE-700 Host AON Expansion RTC

The PrimeCell RTC (PL031) is a real time clock module. The RTC module uses the **S32KCLK** for counting. The reset of this slow part is done by the **AONTOPWARMRESETn** signal also, but it is synchronized by the **S32KCLK** clock to de-assert it synchronously. The interrupt signal of the RTC is connected to the **EXPSHDINT** interface of the SSE-700 subsystem.

Please refer to the *Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual* for further details on **S32KCLK**, **AONTOPWARMRESETn** and **EXPSHDINT**.

4.5.3 SSE-700 Expansion Shared Interrupt Map

The following table shows the interrupt connections of the **EXPSHDINT** interface of the SSE-700. These interrupts can be exposed to the Host CPU, the Secure Enclave, or EXTSYS0 by the Interrupt Router inside SSE-700.

IRQ line	Level/Edge	Connection
EXPSHDINT[0]	Level	EXTSYS0 PPU interrupt
EXPSHDINT[1]	Level	RESERVED
EXPSHDINT[2]	Level	EXTSYS0 Watchdog or Lockup reset request
EXPSHDINT[3]	Level	RESERVED
EXPSHDINT[4]	Level	RESERVED
EXPSHDINT[5]	Level	RTC interrupt
EXPSHDINT[28:6]	-	RESERVED
EXPSHDINT[29]	Level	MPS3 USB peripheral interrupt
EXPSHDINT[30]	Level	MPS3 I ² S Audio peripheral interrupt
EXPSHDINT[31]	Level	MPS3 Ethernet peripheral interrupt
EXPSHDINT[63:32]	Level	RESERVED

4.6 SSE-700 Secure Enclave Cryptographic Accelerator Model

The SE Cryptographic Accelerator is implemented in the form of a model, one part in the switched domain, and one part in the AON domain. The two parts are connected in the model using an APB ADB which spans the power and clock domains. The model provides control signals, data paths and protocol conversion in the FPGA.

Note

This model does not support cryptographic acceleration.

See the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual* for further details on the Cryptographic Accelerator.

4.6.1 High Level Block Diagram

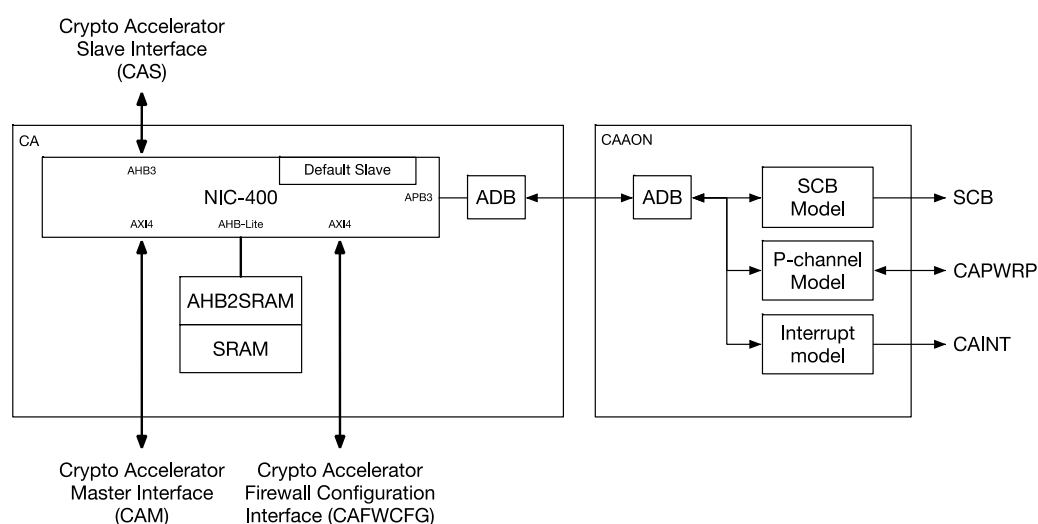


Figure 4-1 : Cryptographic Accelerator Model Block Diagram

4.6.2 CA Module

The module is based on the NIC-400 using the following memory map. The address ranges routed to this module from an AHB-Lite interface are as follows:

Start	End	Target
0x2F00_0000	0x2FFF_FFFF	CA internal access
0x5020_0000	0x503F_FFFF	To be routed to FW control port
0x6000_0000	0xDFFF_FFFF	To be routed to FW data port

Table 4-2 : Cryptographic Accelerator Main Memory Map

Within the CA internal access range, the NIC implementation routes addresses as follows:

Start	End	Target
0x2F80_0000	0x2F80_2000	8KB SRAM
0x2F80_8000	0x2F80_8FFF	AON model

Table 4-3 : Cryptographic Accelerator Internal Memory Map

4.6.3 CA AON Module

Within the AON module there are three functional blocks, SCB, P-Channel, and Interrupts. The following registers are implemented:

Register name	Offset	Field	Description
IRQ0	0x000	[31:1]	RAZ/WI
		[0]	CAINT[0]
IRQ1	0x004	[31:1]	RAZ/WI
		[0]	CAINT[1]
LCS	0x008	[31:4]	RAZ/WI
		[3:0]	LCS[3:0]
SCB0	0x00C	[31:0]	SCB[31:0]
SCB1	0x010	[31:0]	SCB[63:32]
SCB2	0x014	[31:0]	SCB[95:64]
SCB3	0x018	[31:0]	SCB[127:96]
CALC	0x01C	[31:1]	RAZ/WI
		[0]	CALC
PACCEPT	0x020	[31:1]	RAZ/WI
		[0]	paccept_ctrl
PACTIVE	0x024	[31:11]	RAZ/WI
		[10:0]	pactive_ctrl

Table 4-4 Cryptographic Accelerator Registers

4.6.4 Lifecycle State (LCS)

The LCS is used within the Crypto Accelerator to determine the default values applied to the Security Control Bits.

The following table shows the supported values.

LCS Value	State	Short Name	Comment
0b0000	Invalid	INV	Post-reset value while CA initializing
0b0001	Chip Manufacture	CM	-
0b0010	Device Manufacture	DM	-
0b0100	Secure Enable	SE	-
0b1000	Return Merchandise Authorization	RMA	-

Table 4-5 : Lifecycle State Values

All other LCS values are invalid, however there is no check in the model for invalid values.

In the model the LCS resets to 0b0000 – invalid. After 256 cycles, the LCS is then set to a valid value, the default being 0b0001 (CM). A new value can then be written to the LCS using the register at offset 0x8. In the model, this value takes effect immediately, not after a reset, and all transitions are allowed.

4.6.5 Crypto Accelerator Lifecycle Control

This read only register allows software to read the state of the CALC input.

4.6.6 Security Control Bits (SCB)

The Crypto Accelerator provides Security Control Bits (SCBs) to enable and disable features in the SSE-700 subsystem. There is a 1-cycle delay between the LCS becoming valid and the SCBs updating to their default value for that LCS. Four registers in the Crypto Accelerator drive the SCB outputs of the model, as detailed in the table 4-5 above.

The following table shows how the SCBs update after the LCS becomes valid.

SCB Bit position(s)	LCS = CM
[1:0]	0b11
[3:2]	0b00
[10:4]	0x7F
[12:11]	0b00
[41:13]	0xFFFF_FFFF
[47:42]	0x00
[50:48]	0b111
[62:51]	0x000
[63]	0b1
[127:64]	0xFFFF_FFFF_FFFF_FFFF

Table 4-6 SCB Reset defaults

Note

The four SCB[0-3] registers can be written to any value to set/clear the individual SCBs. There is no restriction on these writes based on the LCS of the device.

4.6.7 Interrupts

To drive the interrupt outputs of the Crypto Accelerator model, two separate registers are included.

When IRQ0 is set to 0b1, bit[0] of the CAINT bus is set 4096 cycles later. When it is cleared to 0b0, the interrupt output is cleared immediately.

When IRQ1 is set to 0b1, bit[1] of the CAINT bus is set 4096 cycles later. When it is cleared to 0b0, the interrupt output is cleared immediately.

The delay is present to allow time for the CPU to enter SLEEP mode prior to the interrupt firing.

4.6.8 P-Channel Model

This block controls the P-Channel interface supported by the Crypto Accelerator in the Secure Enclave. In this implementation only the mode ON is supported.

The PACCEPT register controls whether the next state transition request on the P-Channel is accepted (0b1) or denied (0b0). All transitions from OFF are always accepted. When in a non-OFF state, the PACCEPT register controls accept/deny for any further transitions. The model assumes that the P-Channel only makes power mode transitions as defined by the PPU architecture.

The PACTIVE register is used to set the PACTIVE output from the Crypto Accelerator. On reset, the PACTIVE output is set to 0x001- indicating OFF. Writes to the PACTIVE register are delayed before the new value is output on the PACTIVE interface port. The default delay is 4096 cycles.

4.7 EXTSYS0

4.7.1 System Control Registers

EXTSYS0 contains a set of 32-bit system control registers. These registers are mapped into the EXTSYS0 memory map base address. The following table describes how the registers are decoded:

Name	Address Offset	Type	Reset Source	Reset Value	Description
RESETINFO	0x00	RW	EXTSYSPORESETn	0x00	Reset syndrome register. This register logs the reset sources in the system. A write to a bit of this register of: 0b0 clears a logged reset syndrome 0b1 is ignored Decoding: [0]: Power-on reset [1]: nSRST [2]: RESERVED [3]: HOST System reset request [4]: EXTSYS reset request [5]: EXTSYS internal system reset request [31:6]: RESERVED
POWERDOWNEN	0x04	RW	EXTSYSPORESETn	0x0	1-bit register. When the bit is set, if all other requirements are met, the power down of the

EXTSYS domain is enabled.

[31:1]: RESERVED

CLK_CTL	0x08	RW	EXTSYSPORESETn	0x0000_3E1F	Clock control register: [0]: Force the EXTSYSAONCLK to run [8:1]: AON Clock Controller entry delay value [9]: Force the EXTSYSHCLK and EXTSYSDCLK to run [17:10]: CORE HCLK and CORE DCLK Clock Controller entry delay value [31:18]: RESERVED
---------	------	----	----------------	-------------	---

Table 4-7 : EXTSYS0 System Control Registers

Note

Other addresses in this interface behave as RAZ/WI.

Reserved registers bits behave as RAZ/WI

The System Register module never returns a slave error in its APB interface and it never inserts a wait state.

4.7.2 EXTSYS0 Interrupt Connectivity

This following table describes the interrupts that are handled by the Cortex-M3 core in EXTSYS0.

No.	Level/Edge	Priority	Description
16	Level	Programmable	Timer 0 interrupt
17	Level	Programmable	Timer 1 interrupt
18	Level	Programmable	HES MHU0 Combined interrupt
19	Level	Programmable	ESH MHU0 Combined interrupt
20	Level	Programmable	HES MHU1 Combined interrupt
21	Level	Programmable	ESH MHU1 Combined interrupt
22	Level	Programmable	SEES MHU0 Combined interrupt
23	Level	Programmable	ESSE MHU0 Combined interrupt
24	Level	Programmable	SEES MHU1 Combined interrupt
25	Level	Programmable	ESSE MHU1 Combined interrupt
26	Edge	Programmable	HXB-Bridge error interrupt
27	Level	Programmable	EXTSYS0 UART
28	-	-	Reserved
29-60	-	Programmable	EXTSYS0 SHDINT[31:0] from the External System Harness. Note EXTSYS0 SHDINT[0] is routed to interrupt number 29 EXTSYS0 SHDINT[31] is routed to interrupt number 60

See the *Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual* for more details.

61	Level	Programmable	EXTSYS0 SHDINT[32] – EXTSYS0 PPU interrupt
62	-	-	Reserved
63	Level	Programmable	EXTSYS0 SHDINT[34] – EXTSYS 0 Watchdog or Lockup reset request
64	-	-	Reserved
65	-	-	Reserved
66	Level	Programmable	EXTSYS0 SHDINT[37] – RTC interrupt
67	-	-	Reserved

Table 4-8 : EXTSYS0 System Interrupt Connectivity

4.8 SBCon I²C

The following table shows the register map for the two-wire SBCon:

Address	Name	Access	Description
0x40030000	SB_CONTROL	Read	Read serial control bits: Bit [0] is SCL Bit [1] is SDA
0x40030000	SB_CONTROLS	Write	Set serial control bits: Bit [0] is SCL Bit [1] is SDA
0x40030004	SB_CONTROLC	Write	Clear serial control bits: Bit [0] is SCL Bit [1] is SDA

Table 4-9 SBCon Register Map

4.9 Audio I²S

The I²S interface supports transfer of digital audio to and from the Audio CODEC.

The following table shows the register memory map for I²S Audio registers:

Address	Name	Description
0x40020000	CONTROL	<u>Control Register</u> [31:18] Reserved [17] Audio codec reset control (output pin) [16] FIFO reset [15] Reserved [14:12] Rx Buffer IRQ Water Level - Default 2 (IRQ triggers when less than two-word space is available). [11] Reserved [10:8] TX Buffer IRQ Water Level - Default 2 (IRQ triggers when more than two-word space is available). [7:4] Reserved [3] Rx Interrupt Enable [2] Rx Enable

Address	Name	Description
		[1] Tx Interrupt Enable
		[0] Tx Enable
0x40020004	STATUS	<u>Status Register</u>
		[31:6] Reserved
		[5] Rx Buffer Full
		[4] Rx Buffer Empty
		[3] Tx Buffer Full
		[2] Tx Buffer Empty
		[1] Rx Buffer Alert (Depends on Water level)
		[0] Tx Buffer Alert (Depends on Water level)
0x40020008	ERROR	<u>Error Status Register</u>
		[31:2] Reserved
		[1] Rx overrun. Set this bit to clear.
		[0] Tx overrun or underrun. Set this bit to clear.
0x4002000C	DIVIDE	<u>Clock Divide Ratio Register</u> (for left or right clock)
		[31:10] Reserved
		[9:0] LRDIV (Left/Right). The default value is 0x80. $12.288\text{MHz} / 48\text{kHz} / 2^{*(L+R)} = 128.$
0x40020010	TXBUF	<u>Transmit Buffer FIFO Data Register</u> . This is a write-only register.
		[31:16] Left channel
		[15:0] Right channel
0x40020014	RXBUF	<u>Receive Buffer FIFO Data Register</u> . This is a read-only register.
		[31:16] Left channel
		[15:0] Right channel
0x40020014- 0x400202FC	RESERVED	-
0x40020300	ITCR	<u>Integration Test Control Register</u>
		[31:1] Reserved
		[0] ITCR
0x40020304	ITIP1	<u>Integration Test Input Register 1</u>
		[31:1] Reserved
		[0] SDIN
0x40020308	ITOP1	<u>Integration Test Output Register 1</u>

Address	Name	Description
		[31:4] Reserved
		[3] IRQOUT
		[2] LRCK
		[1] SCLK
		[0] SDOUT

Table 4-10 Audio I²S Register Map

4.10 SMM Registers

This design supports SMM FPGA registers, which are two 4KB blocks respectively starting from 0x4000_0000 and 0x4001_0000 in the EXPMST0 region:

- The FPGA SCC register block (see section 4.12 for more details)
 - The block contains registers to control the CPUWAIT and the Write Enable control on the SE ROM (RAM).
 - These registers connect to the NIC-400 using an APB interface. See the memory map for mapping details.
- The FPGA I/O register block (see section 4.13 for more details)
 - This block supports read of the SE SCB bits.
 - This block also provides read status of the MPS3 board USER SW (User DIP switches).
 - These registers connect to the NIC-400 using an APB interface. See the memory map for mapping details.

4.11 LEDs

The following table shows the LEDs that the MPS3 board provides.

LED #	Description
0	SSE-700 Secure Enclave SCB (Security Control Bits) [0]
1	SSE-700 Secure Enclave SCB (Security Control Bits) [1]
2	SSE-700 Secure Enclave SCB (Security Control Bits) [4]
3	SSE-700 Secure Enclave SCB (Security Control Bits) [34]
4	SSE-700 Secure Enclave SCB (Security Control Bits) [35]
5	SSE-700 Secure Enclave SCB (Security Control Bits) [36]
6	SSE-700 Secure Enclave SCB (Security Control Bits) [37]
7	SSE-700 Secure Enclave SCB (Security Control Bits) [48]
8	SSE-700 Secure Enclave SCB (Security Control Bits) [49]
9	SSE-700 Secure Enclave SCB (Security Control Bits) [50]

Table 4-11 : MPS3 Board LED Assignments

4.12 FPGA Serial Communication Controller (SCC)

The SMM implements communication between the microcontroller and the FPGA system through an SCC interface, implemented in the FPGA Fabric

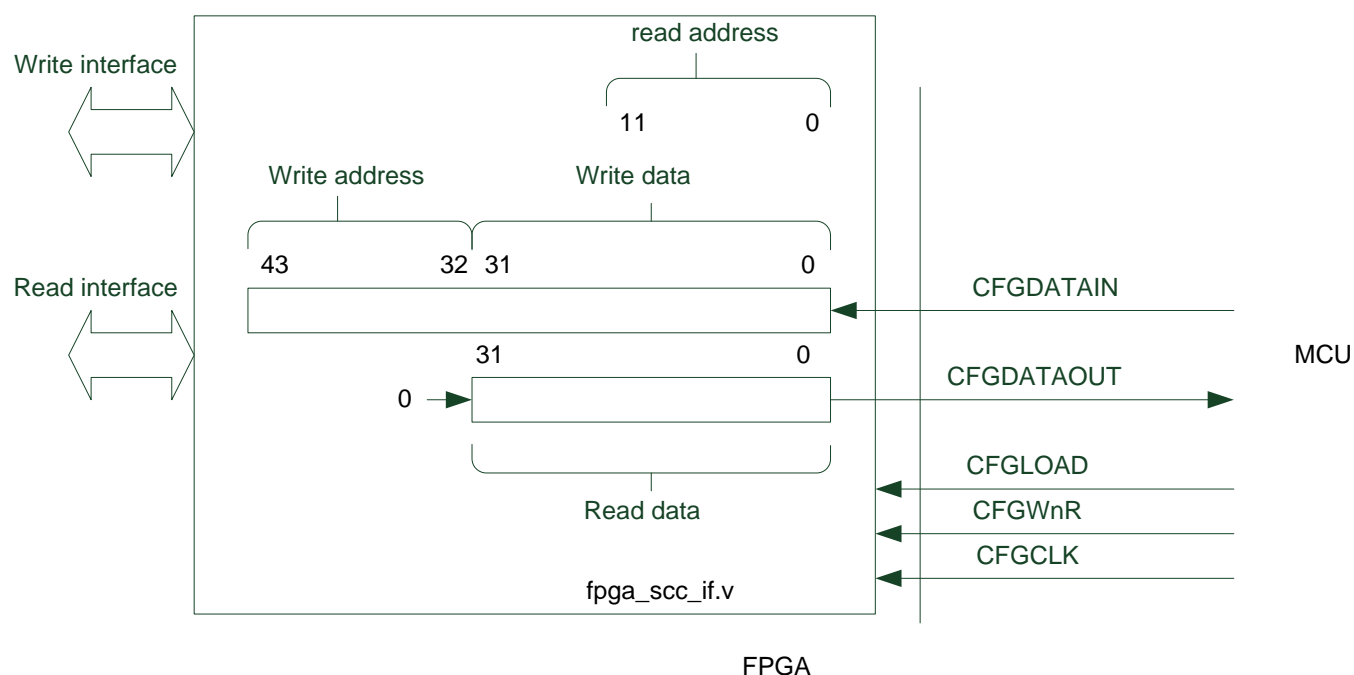


Figure 4-2 : Diagram of the SCC Interface

The read-addresses and write-addresses of the SCC interface do not use bits[1:0]

All address words are word-aligned.

Address	Name	Information
0x000	CFG_REG0	Bits[31:0] Reserved
0x004	CFG_REG1	32-bit DATA [RW]
0x008	CFG_REG2	Bits[31:1] Reserved Bits[0]: QSPI Read/Write Select signal (Read = 0x0, Write = 0x1)
0x00C	CFG_REG3	Bits[31:0] Reserved
0x010	CFG_REG4	Bits[31:4] Reserved Bits[3:0] Board Revision [r]
0x014	CFG_REG5	Bits[31:0] ACLK Frequency in Hz
0x018	CFG_REG6	Bits[31:1] Reserved Bits[0]: CPUWAIT (Active high, resets to high)
0x01C	CFG_REG7	Bits[31:1] Reserved Bits[0] SE Cortex-M0+ ROM (RAM) Enable Write (Enable write = 0x0, disable write = 0x1, resets to low)
0x020 – 0x09C	RESERVED	-
0x0A0	SYS_CFGDATA_RTN	32-bit DATA [RW]
0x0A4	SYS_CFGDATA_OUT	32-bit DATA [RW]
0x0A8 – 0xFF4	RESERVED	-
0xFF8	SCC_AID	SCC AID register is read only Bits[31:24] FPGA build number Bits[23:20] V2M-MPS3 target board revision (A = 0, B = 1, C = 2) Bits[19:8] Reserved Bits[7:0] number of SCC configuration registers
0xFFC	SCC_ID	SCC ID register is read only Bits[31:24] Implementer ID: 0x41 = Arm Bits[23:20] Reserved Bits[19:16] IP Architecture: 0x4 = AHB Bits[15:12] Reserved Bits[11:4] Primary part number: 543 = AN543 Bits[3:0] Reserved

Table 4-12 : SCC Register Memory Map

4.13 FPGA System Control and I/O Registers

The SMM implements an FPGA system control block.

Address	Name	Information
0x4001_0000 - 0x4001_0004	RESERVED	
0x4001_0008	FPGAIO->BUTTON	Buttons Bits[31:2] Reserved Bits[1:0] Buttons
0x4001_000C - 0x4001_0024	RESERVED	
0x4001_0028	FPGAIO->SWITCH	Switches Bits[31:8] Reserved Bits[7:0] Switches
0x4001_002C - 0x4001_004C	RESERVED	
0x4001_0050	FPGAIO->GP_REG1	General Purpose Register 1 Bits[31:0] Secure Enclave SCB [31:0]
0x4001_0054	FPGAIO-> GP_REG2	General Purpose Register 2 Bits[31:0] Secure Enclave SCB [63:32]
0x4001_0058	FPGAIO-> GP_REG3	General Purpose Register 3 Bits[31:0] 0xF00F_F00F
0x4001_005C	FPGAIO-> GP_REG4	General Purpose Register 4 Bits[31:0] 0xABBA1234

Table 4-13 : System Control and I/O Memory Map

5 Clock Architecture

The following tables list the clocks entering the FPGA and internally generated by the SMM.

5.1 Clocks

5.1.1 FPGA External Clocks

The following clocks are inputs to the FPGA.

Clock	Input Pin	Frequency	Note
REFCLK24MHZ	OSCCLK[0]	24MHz	24MHz reference
ACLK	OSCCLK[1]	32MHz	Programmable oscillator
DBGCLK	CS_TCK	Set by debugger	JTAG input
CFGCLK	CFG_CLK	Set by MCC	SCC register clock from MCC
DDR4_REF_CLK	c0_sys_clk_p/n	100MHz	Differential input clock to DDR4 controller
SMBM_CLK	SMBM_CLK	Set by MCC (40MHz)	SMB clock from MCC

Table 5-1 : FPGA External Source Clocks

5.1.2 FPGA Internal Clocks

The following clocks are generated internally from the source clocks.

Clock	Source	Frequency	Note
MAINCLK	OSCCLK[1]	32MHz	
CLK32KHZ	REFCLK24MHZ	32kHz	
CFGCLK	CFG_CLK	Set by MCC	SCC register clock from MCC

Table 5-2 : FPGA Generated Internal Clocks

5.1.3 SSE-700 Clocks

Clock	Source	Frequency	Note
REFCLK	MAINCLK	32MHz	
SECENCREFCLK	MAINCLK	32MHz	
SYSPLL	MAINCLK	32MHz	
CPUPLL	MAINCLK	32MHz	
EXTSYSF0CLK	MAINCLK	32MHz	
UARTCLK	MAINCLK	32MHz	
SWCLKTCK	CS_TCK	Set by debugger	
TRACECLKIN	MAINCLK	32MHz	
S32KCLK	REFCLK24MHZ	32kHz	
CFGCLK	CFG_CLK	Set by MCC	SCC register clock from MCC

Table 5-3 : FPGA Generated SSE-700 Internal Clocks

Note

Clock gating is not supported in the implementation (except within the Cortex-A32) and clock enables are forced to “always enabled”.

Clock dividers are not supported in the implementation and are removed or forced to 1:1.

5.1.4 EXTSYS0 Clocks

Clock	Source	Frequency	Note
EXTSYS0DBGCLKS	MAINCLK	32MHz	
EXTSYS0DBGCLKM	MAINCLK	32MHz	
EXTSYS0MHUCLK	MAINCLK	32MHz	
EXTSYS0ATCLK	MAINCLK	32MHz	
EXTSYS0CTICLK	MAINCLK	32MHz	
EXTSYS0ACLK	MAINCLK	32MHz	

Table 5-4 : FPGA Generated EXTSYS0 Internal Clocks

Note

Clock gating is not supported in the implementation and clock enables are forced to “always enabled”.

Clock dividers are not supported in the implementation and are removed or forced to 1:1.

6 ZIP Bundle Description

6.1 Overall Structure

The accompanying .zip bundle contains:

- This Application Note Document.
- An example Development Studio 2020.1 software project, that can be run on the SE Cortex-M0+ to test supported board peripherals and interfaces.
- Boardfiles/ directory containing the directory structure and files to be loaded onto the MPS3 SD Card. This is required to configure the MPS3 board to load and run this implementation.

6.2 Documentation

This Application Note Document, AN543, is in the Docs/ folder of the bundle.

6.3 MPS3 Board Revision and Support

6.3.1 Identifying the MPS3 Board Revision

The bundle supports MPS3 board revisions A, B and C. The board revision, if not known can be identified from the silk screen text, inside a marked box, on the board as shown in the diagram below:



Board Part Number and Revision

Figure 7-1 : MPS3 Board Revision Identifier

In this example the part number is “HBI0309B”. The last letter at the end of the part number denotes the board revision. The illustration shows a revision **B** board.

6.3.2 Bundle Support for Specific MPS3 Board Revisions

There are three subdirectories in the Boardfiles/MB/ directory that correspond to the three supported revisions:

- HBI0309A
- HBI0309B
- HBI0309C

The contents of each of these directories, within the provided bundle, are identical but the MCC only uses the contents from the directory name that matches the board part number and revision in use (see section 6.3.1 for further details on how to identify the board part number and revision).

Note

Only files modified within the directory name that align with the MPS3 board part number and revision are used by the MCC. Care must be taken to ensure the correct directory contents are modified, if modifications are required.

6.4 Bundle Directory Tree/Structure

The directory structure of the bundle is shown below.

```
-- Boardfiles
|
|-- MB
|   |-- BRD_LOG.TXT
|   |-- HBI0309A
|   |   |-- AN543
|   |   |   |-- AN543_v3.bit
|   |   |   |-- an543_v3.txt
|   |   |   |-- images.txt
|   |   |-- board.txt
|   |   |-- mbb_v141.ebf
|   |-- HBI0309B
|   |   |-- AN543
|   |   |   |-- AN543_v3.bit
|   |   |   |-- an543_v3.txt
|   |   |   |-- images.txt
|   |   |-- board.txt
|   |   |-- mbb_v141.ebf
|   |-- HBI0309C
|   |   |-- AN543
|   |   |   |-- AN543_v3.bit
|   |   |   |-- an543_v3.txt
|   |   |   |-- images.txt
|   |   |-- board.txt
|   |   |-- mbb_v141.ebf
|   |-- SOFTWARE
|   |   |-- Selftest.axf
|   |-- config.txt
|-- Docs
|   |-- DAI0543B_00_arm_corstone_700_for_mps3.pdf
|-- Licence.pdf
|-- Software
|   |-- selftest
|   |   |-- AACI
|   |   |   |-- AAIC_I2C_MPS3.c
|   |   |   |-- AAIC_I2C_MPS3.h
|   |   |   |-- AAIC_I2S_MPS3.c
|   |   |   |-- AAIC_I2S_MPS3.h
|   |   |   |-- apaaci.c
|   |   |   |-- apaaci.h
|   |-- Debug
|   |   |-- Selftest.axf
```

```

|-- Ethernet
|   |-- ETH_MPS3.c
|   |-- ETH_MPS3.h
|   |-- aplan.c
|   `-- aplan.h
|-- USB
|   |-- apusb.c
|   `-- apusb.h
|-- apmain
|   |-- common.c
|   |-- common.h
|   |-- main.c
|   |-- retarget.c
|   |-- uart_private.h
|   |-- uart_stdout.c
|   `-- uart_stdout.h
|-- apmem
|   |-- apmem.c
|   |-- apmem.h
|   `-- apmemsup.s
|-- apqspi
|   |-- apqspi.c
|   `-- apqspi.h
|-- cmsis
|   |-- CMSIS
|   |   |-- Include
|   |   |   |-- arm_common_tables.h
|   |   |   |-- arm_const_structs.h
|   |   |   |-- arm_math.h
|   |   |   |-- cmsis_armcc.h
|   |   |   |-- cmsis_armclang.h
|   |   |   |-- cmsis_compiler.h
|   |   |   |-- cmsis_gcc.h
|   |   |   |-- core_cm0plus.h
|   |   |   `-- mpu_armv7.h
|   |   `-- Device
|   |       |-- Include
|   |       |   |-- CMSDK_IoT
|   |       |   |   |-- CMSDK_IoT.h
|   |       |   |   `-- system_CMSDK_IoT.h
|   |       |   |-- CMSDK_driver.h
|   |       |   |-- host_chassis_control.h
|   |       |   `-- system.h
|   |       `-- Source
|   |           |-- CMSDK_IoT
|   |           |   |-- startup_CMSDK_IoT.s
|   |           |   `-- system_CMSDK_IoT.c
|   |           `-- CMSDK_driver.c
|   `-- selftest_IoT.scad
`-- revision_history.txt

```

7 Software and Debug

7.1 Overview

It is strongly recommended to use Development Studio 2020.1 Silver edition which is the minimum edition needed to support all the processor cores in this implementation and the version supported by this application note.

An example software project that can be built using Arm Development Studio is supplied (targeting the SE Cortex-M0+), as part of the bundle. The test software uses the SE UART to implement a menu-driven software interface, controllable from a host terminal emulator. The test software, which executes on the SE Cortex-M0+, tests the following:

1. Internal FPGA BRAM (CVM)
2. External DDR4 (OCVM)
3. External QSPI flash (XNVM).
4. Ethernet
5. Audio
6. USB

7.2 Example Software Project

The bundle includes the source code and the Development Studio project setup files. The software can be built within Development Studio and the target .axf file can either be directly downloaded to the SE ROM using the Development Studio debugger or by copying the built .axf file to the MPS3 SD Card and changing the `images.txt` file to load the .axf file into the SE ROM with an MPS3 board power on reset.

7.3 Debug Support

See section 7.3.3 for the locations of the debug connectors on the MPS3 board.

7.3.1 DSTREAM Debug Hardware

The Arm DSTREAM High-Performance Debug and Trace units are supported for this application note and all the instructions and guidance in this application note assume the use of the DSTREAM debugger.

7.3.2 Debug Connectivity

The following table shows the supported connectivity between the supported MPS3 Board debug connectors and supported debug in the FPGA implementation:

Debug Connector Type	P-JTAG Debug	SWD	4-bit Trace	16-bit Trace
20 pin Cortex debug and ETM	Cortex-M0+, Cortex-M3 and Cortex-A32	Cortex-M0+, Cortex-M3 and Cortex-A32	Cortex-M3 and Cortex-A32	
20 pin IDC	Cortex-M0+, Cortex-M3 and Cortex-A32	Cortex-M0+, Cortex-M3 and Cortex-A32		
Mictor 38	Cortex-M0+, Cortex-M3 and Cortex-A32	Cortex-M0+, Cortex-M3 and Cortex-A32	Cortex-M3 and Cortex-A32	Cortex-M3 and Cortex-A32

Table 7-1 : Debug Connectivity and Support

Note

The CMSIS-DAP debug, over the Debug USB connector, is not supported.

7.3.3 Creating a Development Studio Debug Configuration

The steps below show the procedure to create a debug configuration:

1. Ensure the DSTREAM debugger is:
 - a. Powered, and connected to the host running the Development Studio software.
 - b. Connected to the MPS3 using the 20-pin Cortex / 20-pin IDC / Mictor 38 connector on the MPS3 as shown below:

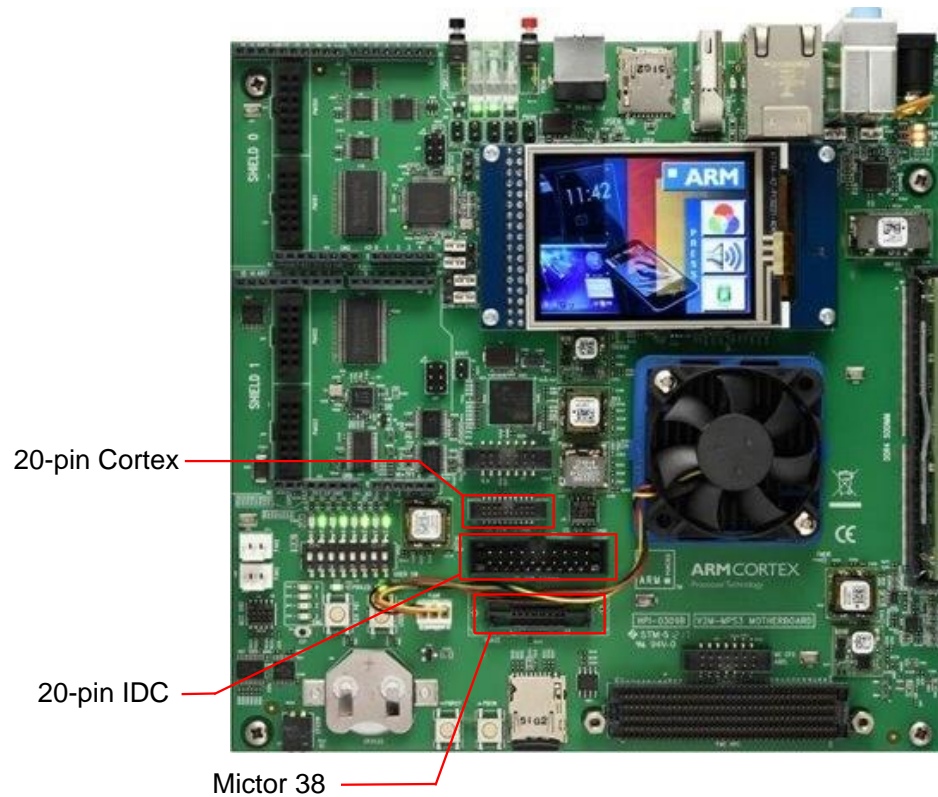
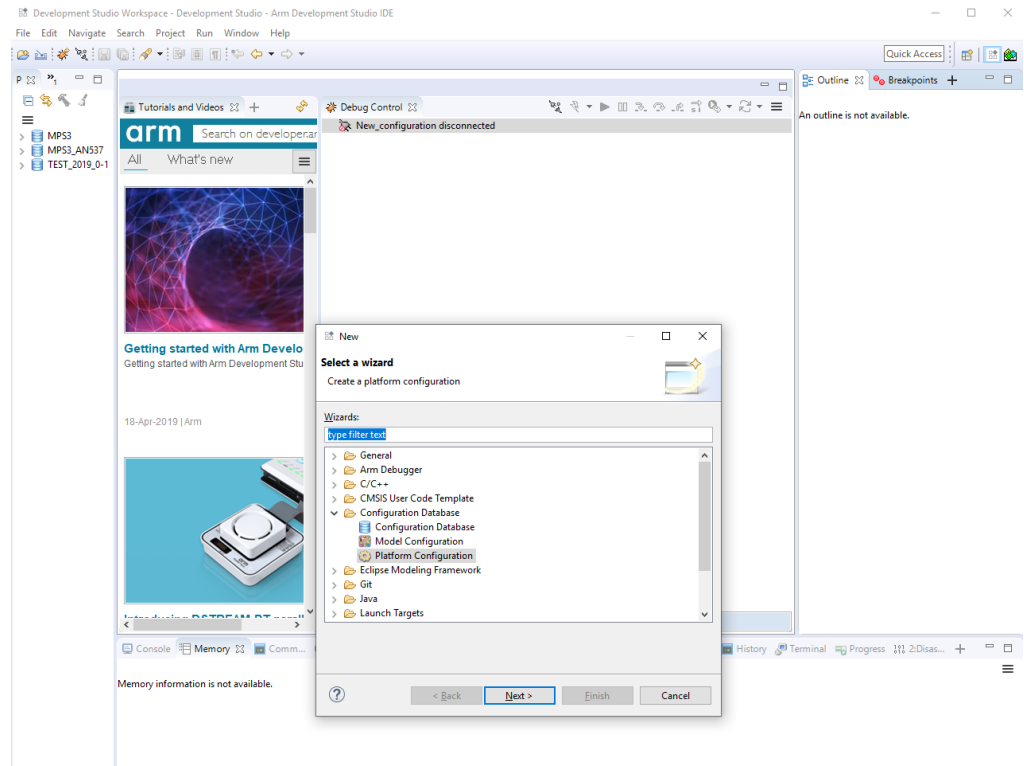


Figure 7-1 : MPS3 Board Debug Connector Locations

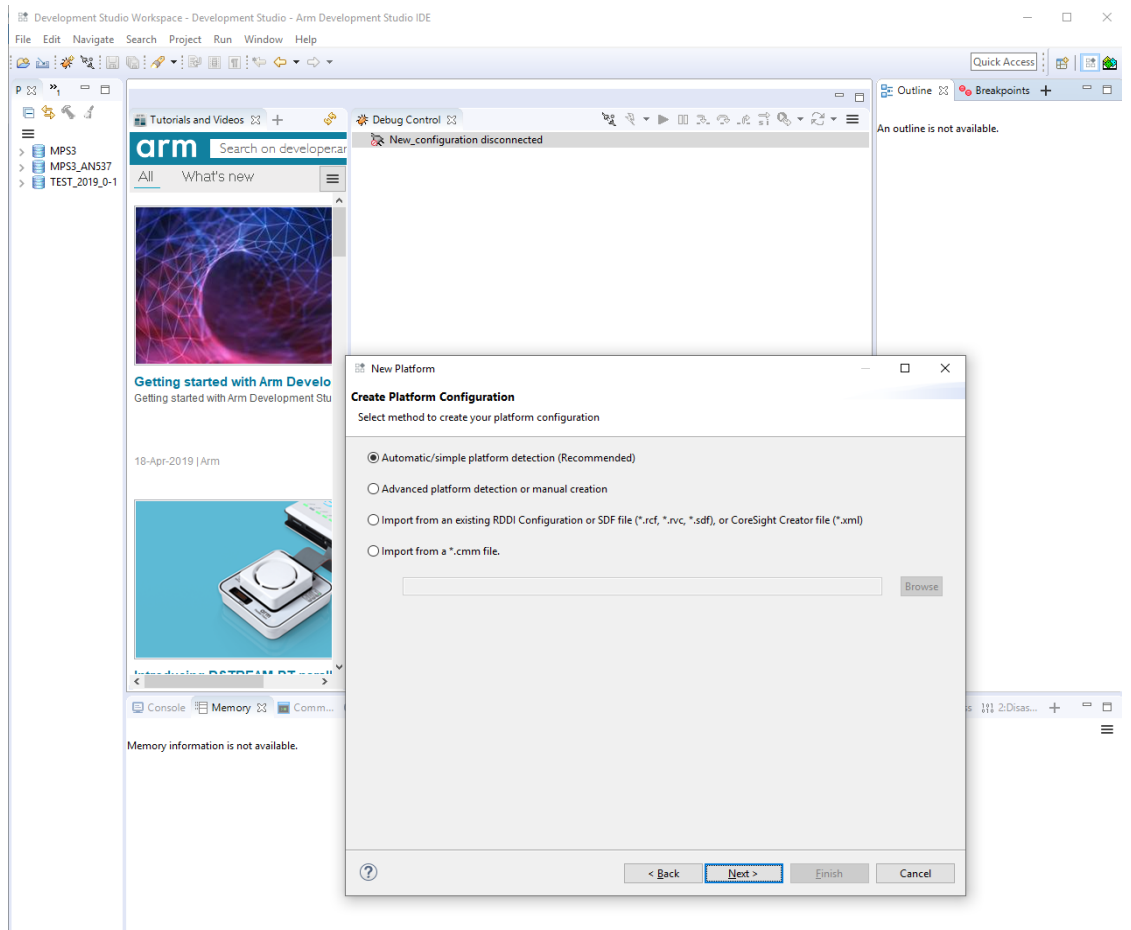
2. Open the Development Studio application on the host machine.

3. Press either Ctrl+N, or from the Menu select **File > New > Other**, to open the wizard to create a **Platform Configuration** as shown below.

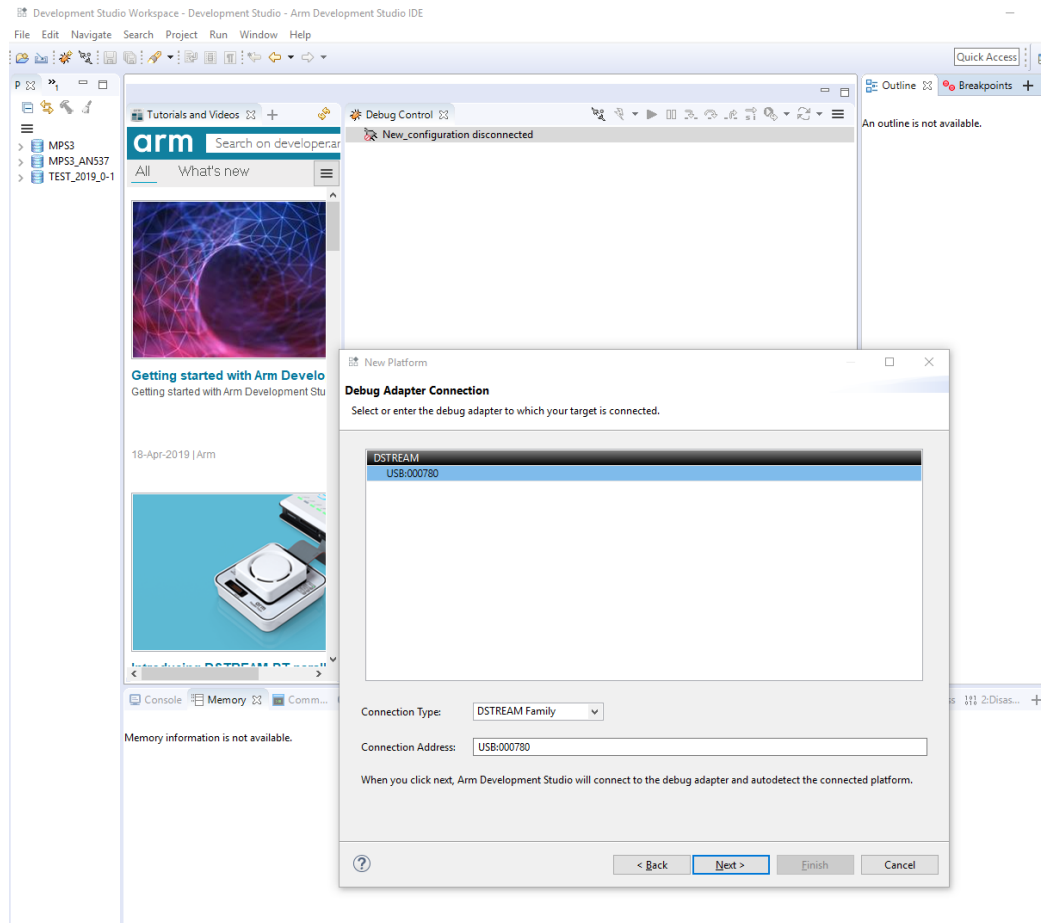


4. Under the **Configuration Database** folder, choose **Platform Configuration** (which highlights when selected) and click **Next**.

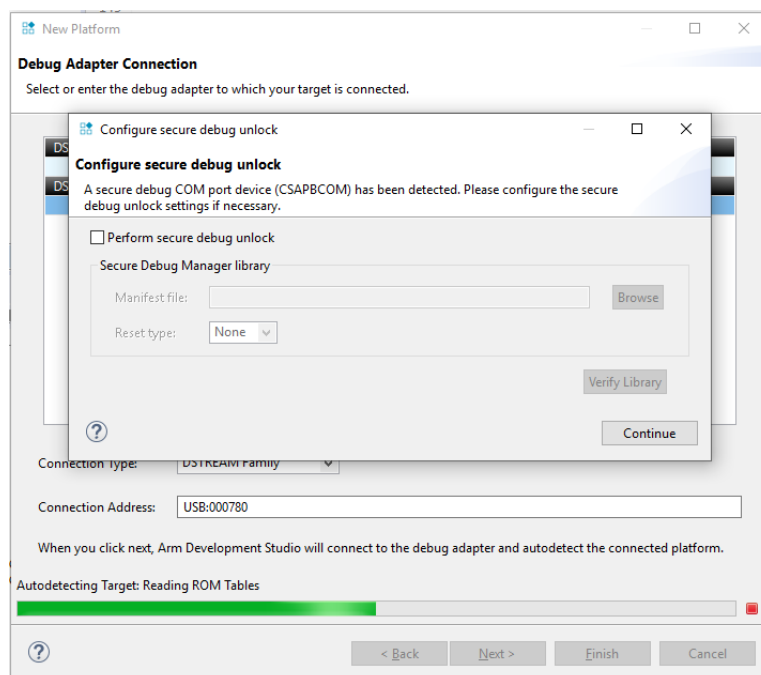
5. Select **Automatic/simple platform detection** and click **Next** as shown below.



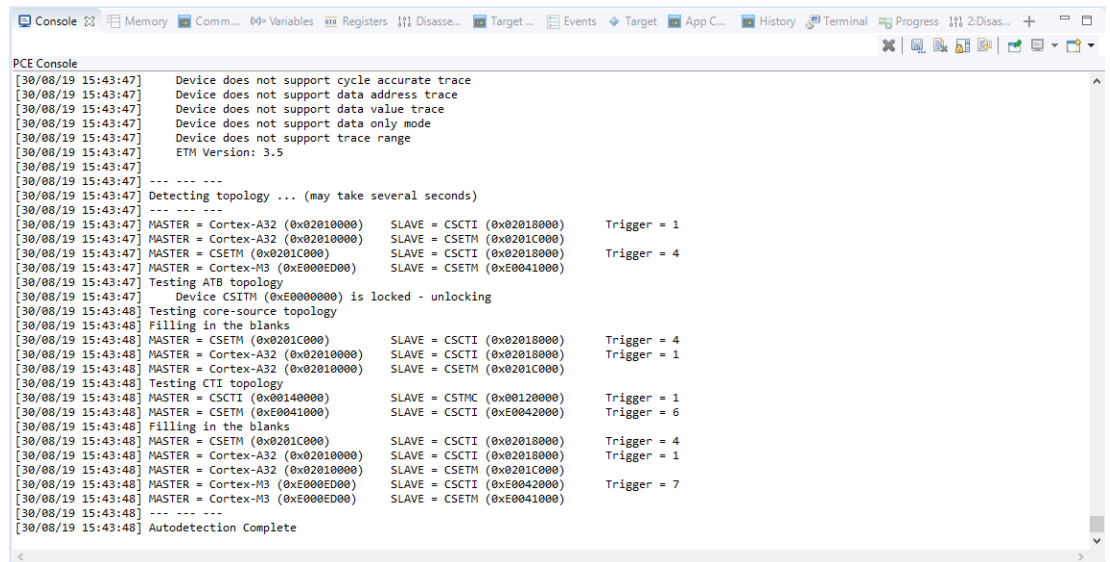
6. Select your connection type and connection address for your target debugger. The figure below shows the case for a DSTREAM debugger with a USB connection.



7. Ensure you have your debugger connected to the target (the MPS3 board in this case) and that the MCC has loaded the FPGA and the implementation has booted. Click **Next**.
8. If the following dialogue appears during the auto discovery process, click **Continue**.

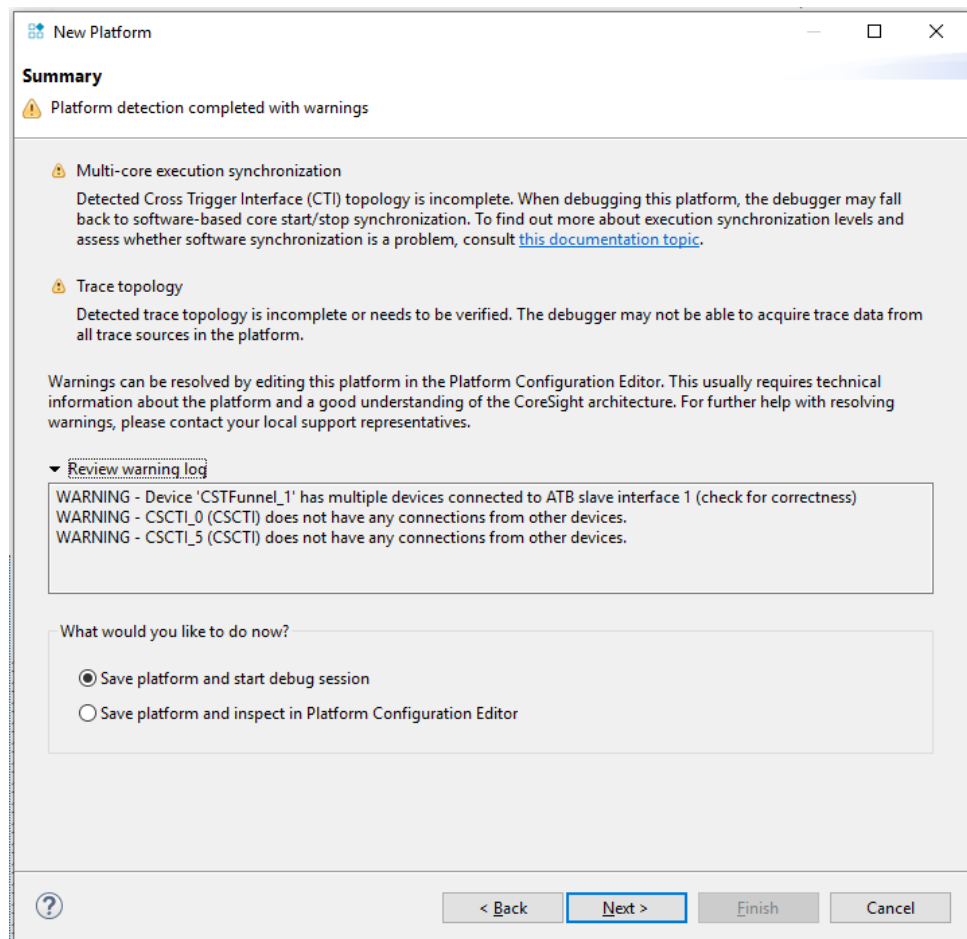


9. Auto Discovery traverses the debug infrastructure to build a configuration for this target. The console shows verbose as shown in the example below or something similar:



```
PCE Console
[30/08/19 15:43:47] Device does not support cycle accurate trace
[30/08/19 15:43:47] Device does not support data address trace
[30/08/19 15:43:47] Device does not support data value trace
[30/08/19 15:43:47] Device does not support data only mode
[30/08/19 15:43:47] Device does not support trace range
[30/08/19 15:43:47] ETM Version: 3.5
[30/08/19 15:43:47]
[30/08/19 15:43:47] -----
[30/08/19 15:43:47] Detecting topology ... (may take several seconds)
[30/08/19 15:43:47] ---
[30/08/19 15:43:47] MASTER = Cortex-A32 (0x02010000) SLAVE = CSCTI (0x02018000) Trigger = 1
[30/08/19 15:43:47] MASTER = Cortex-A32 (0x02010000) SLAVE = CSETM (0x0201C000)
[30/08/19 15:43:47] MASTER = CSETM (0x0201C000) SLAVE = CSCTI (0x02018000) Trigger = 4
[30/08/19 15:43:47] MASTER = Cortex-M3 (0xE000ED00) SLAVE = CSETM (0xE0041000)
[30/08/19 15:43:47] Testing ATB topology
[30/08/19 15:43:47] Device CSITH (0xE0000000) is locked - unlocking
[30/08/19 15:43:48] Testing core-source topology
[30/08/19 15:43:48] Filling in the blanks
[30/08/19 15:43:48] MASTER = CSETM (0x0201C000) SLAVE = CSCTI (0x02018000) Trigger = 4
[30/08/19 15:43:48] MASTER = Cortex-A32 (0x02010000) SLAVE = CSCTI (0x02018000) Trigger = 1
[30/08/19 15:43:48] MASTER = Cortex-A32 (0x02010000) SLAVE = CSETM (0x0201C000)
[30/08/19 15:43:48] Testing CTI topology
[30/08/19 15:43:48] MASTER = CSCTI (0x00140000) SLAVE = CSTMC (0x00120000) Trigger = 1
[30/08/19 15:43:48] MASTER = CSETM (0xE0041000) SLAVE = CSCTI (0xE0042000) Trigger = 6
[30/08/19 15:43:48] Filling in the blanks
[30/08/19 15:43:48] MASTER = CSETM (0x0201C000) SLAVE = CSCTI (0x02018000) Trigger = 4
[30/08/19 15:43:48] MASTER = Cortex-A32 (0x02010000) SLAVE = CSCTI (0x02018000) Trigger = 1
[30/08/19 15:43:48] MASTER = Cortex-A32 (0x02010000) SLAVE = CSETM (0x0201C000)
[30/08/19 15:43:48] MASTER = Cortex-M3 (0xE000ED00) SLAVE = CSCTI (0xE0042000) Trigger = 7
[30/08/19 15:43:48] MASTER = Cortex-M3 (0xE000ED00) SLAVE = CSETM (0xE0041000)
[30/08/19 15:43:48] ---
[30/08/19 15:43:48] Autodetection Complete
```

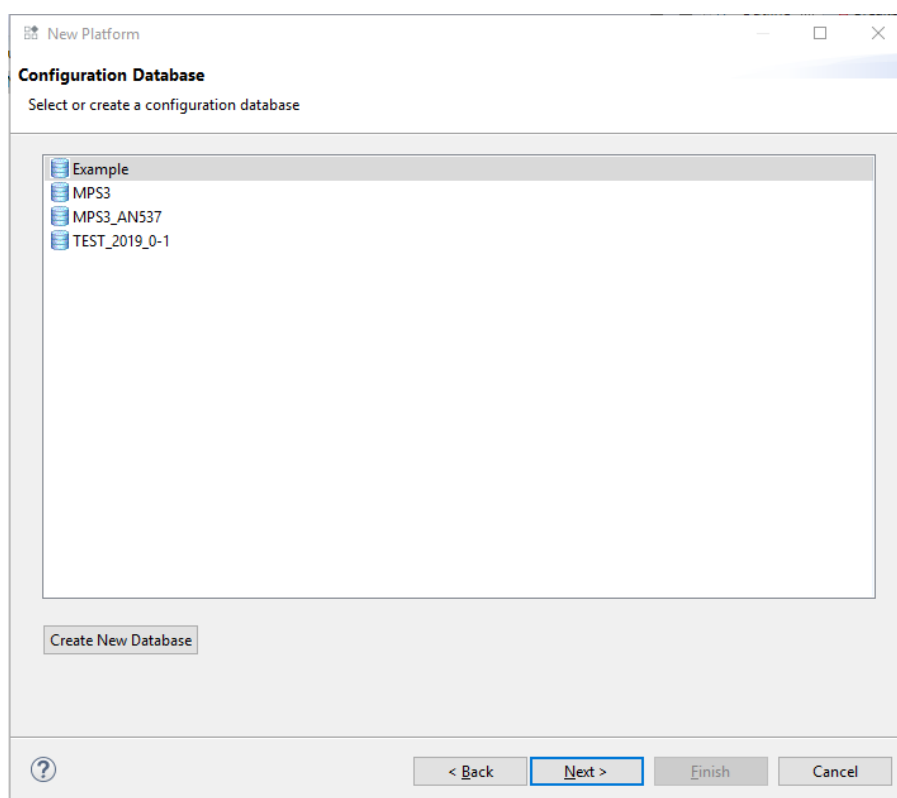
10. After completion, a dialog box shows a summary similar to the example below:



Note

The Warnings observed in the summary are expected and can be ignored.

11. Click **Next** and a dialog box appears, similar to the below:

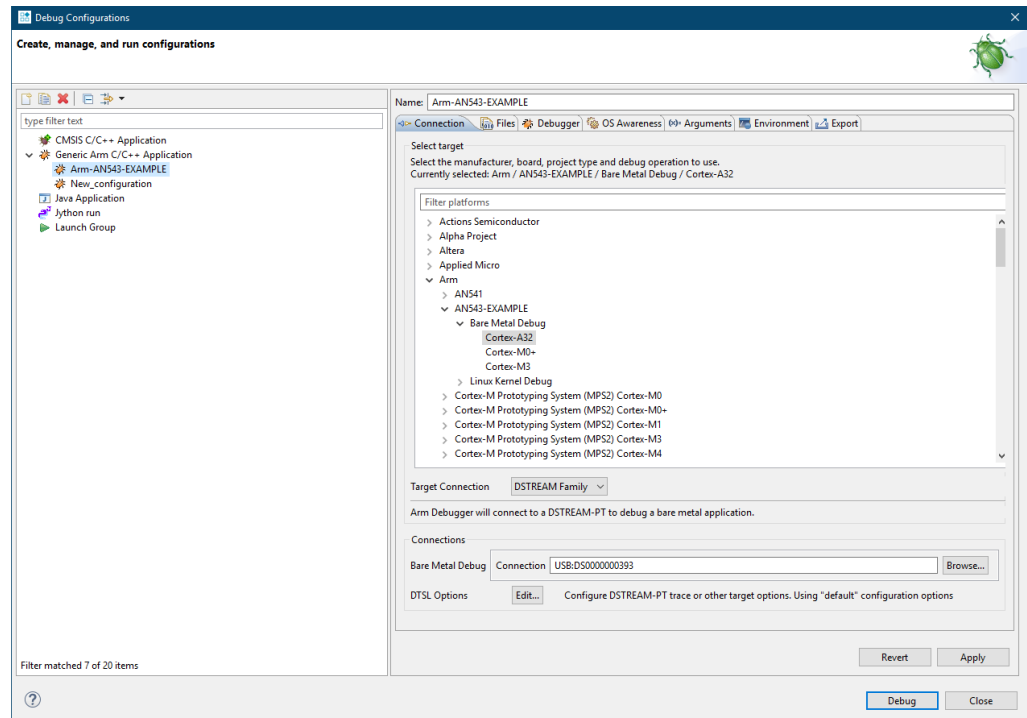


12. Select the target configuration database or create a new one depending on your requirements. In this example a new database **Example** has been created which is used.
13. You now see a dialogue box, as shown below, to define the Platform Information:

A screenshot of a software window titled 'New Platform'. The window has a standard Windows-style title bar. Below the title bar, the text 'Platform Information' is displayed in bold, followed by the instruction 'Use this page to enter identification details for the platform.' There are three input fields: 'Platform Manufacturer:' with the value 'Arm', 'Platform Name:' with the value 'AN543-EXAMPLE', and 'Platform Info URL (Optional):' which is empty. At the bottom of the window, there is a navigation bar with a help icon (question mark), a '< Back' button, a 'Next >' button, a 'Finish' button (which is highlighted with a blue border), and a 'Cancel' button.

14. Click **Finish** to complete the setup process.

15. The newly created debug configuration is now available for selection, similar to the example below:



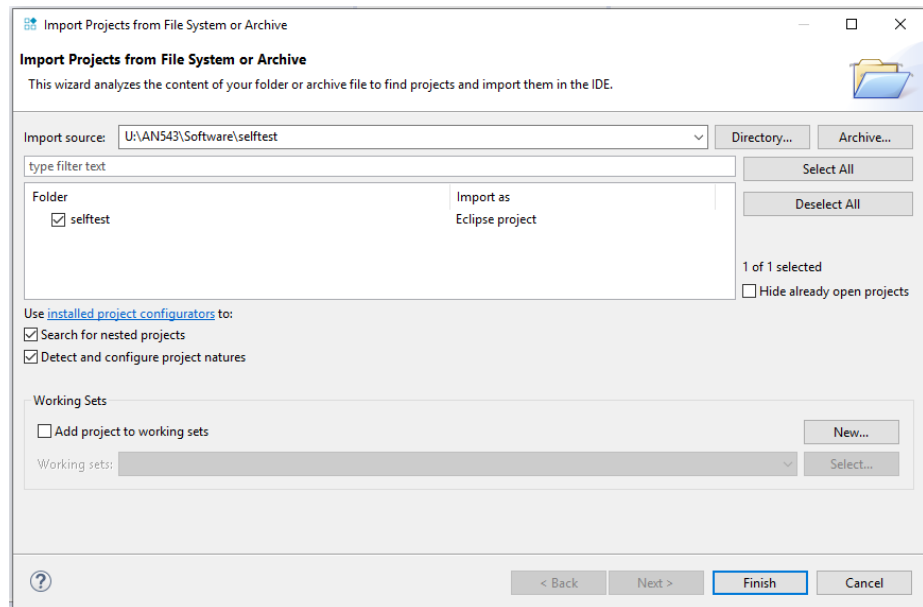
7.4 Building the Software

Requirements:

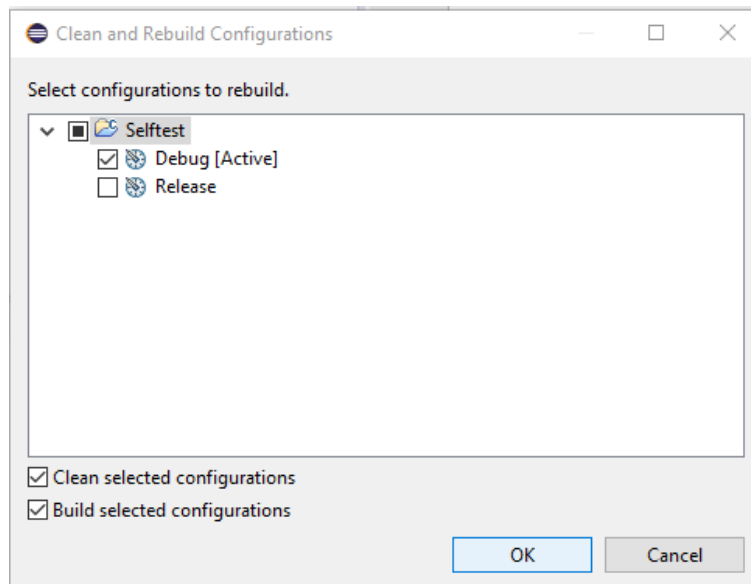
- The software directory from the supplied bundle SOFTWARE/selftest.
- Proficiency with Arm Development Studio, version 2020.1 Silver Edition.

Steps:

1. Open Arm Development Studio.
2. Select **File > Open Projects from File System**, and then select the import source by clicking on the **Directory** button and navigating to the Software/selftest/ directory in the bundle.

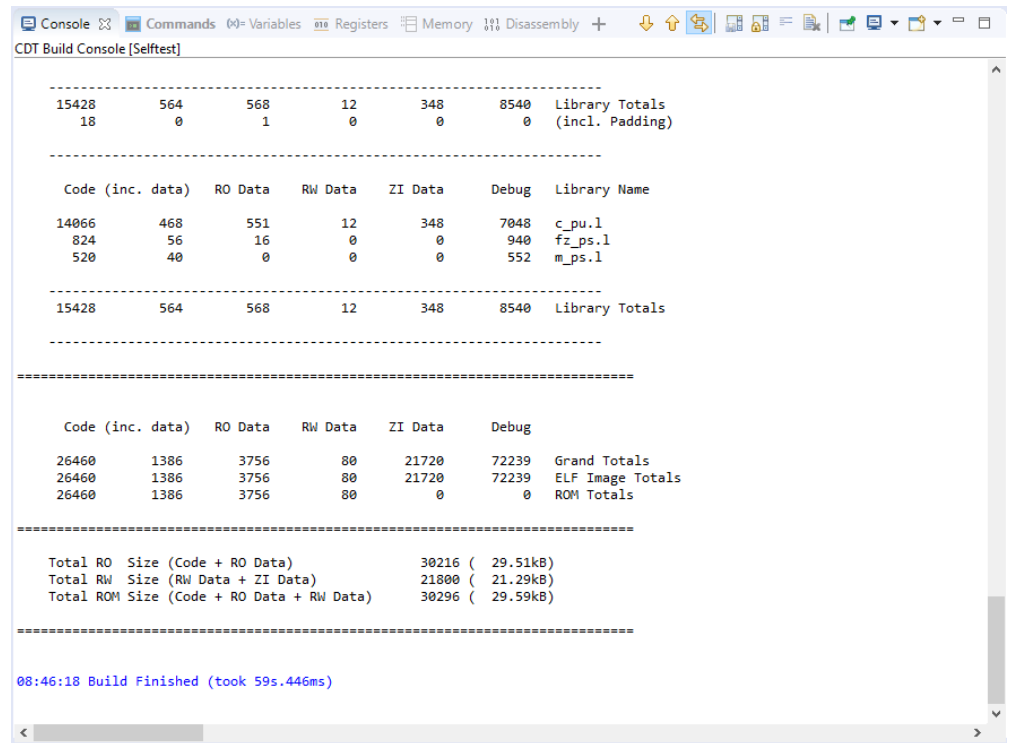


3. Click **Finish**.
4. The project is now imported into the Workspace and is viewable in the **Project Explorer** within Development Studio as **Selftest**.
5. Right-click on the imported project **Selftest** in the **Project Explorer** and select **Build Configurations > Build Selected**.
6. In the **Clean and Rebuild Configurations** window make sure the **Debug [Active]** configuration is selected as shown below:



7. Click **OK**.

8. Ensure you have the console window viewable where you can observe verbose output as the project builds:



The screenshot shows the 'CDT Build Console [Selftest]' window. The output displays memory usage statistics for various components and the final build status.

```
-----  
15428      564      568      12      348      8540  Library Totals  
18          0          1          0          0          0  (incl. Padding)  
-----  
  
Code (inc. data)  RO Data  RW Data  ZI Data  Debug  Library Name  
14066           468           551           12           348           7048  c_pu.l  
824             56             16             0             0           940  fz_ps.l  
520             40              0             0             0           552  m_ps.l  
-----  
15428      564      568      12      348      8540  Library Totals  
-----  
  
=====
```

	Code (inc. data)	RO Data	RW Data	ZI Data	Debug	
26460	1386	3756	80	21720	72239	Grand Totals
26460	1386	3756	80	21720	72239	ELF Image Totals
26460	1386	3756	80	0	0	ROM Totals

```
=====
```

Total	RO	Size (Code + RO Data)	
		30216	(29.51kB)
Total	RW	Size (RW Data + ZI Data)	
		21800	(21.29kB)
Total	ROM	Size (Code + RO Data + RW Data)	
		30296	(29.59kB)

```
=====
```

08:46:18 Build Finished (took 59s.446ms)

9. The project has now built and there is a newly built .axf file in the SOFTWARE/selftest/Debug directory, titled Selftest.axf.

7.5 Establishing a Debug Session

It is possible to connect and establish a debug session to any of the three processors in the implementation:

- The SE Cortex-M0+
- The HOST CPU Cortex-A32
- The EXTSYS0 Cortex-M3

In this example we connect to the SE Cortex-M0+ but the procedure is the same for all three processors in the system.

Steps:

1. Ensure the Development Studio debugger is:
 - a. Powered, and connected to the host running the Development Studio software.
 - b. Connected to the MPS3 using the 20-pin Cortex / 20-pin IDC / Mictor 38 port on the MPS3 as shown below:

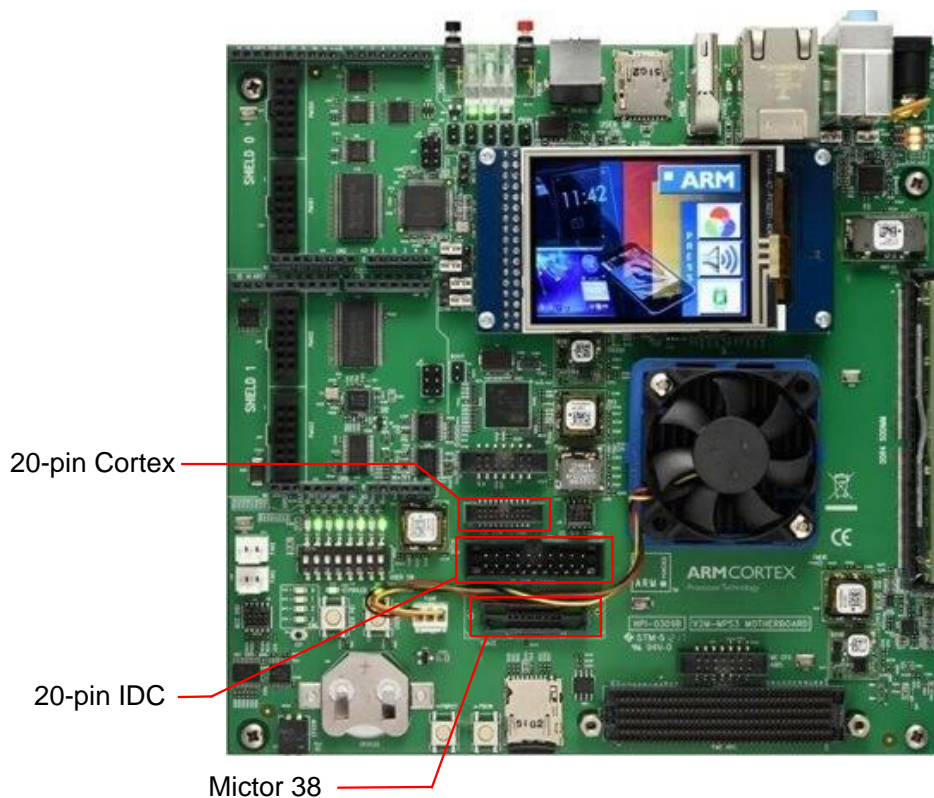
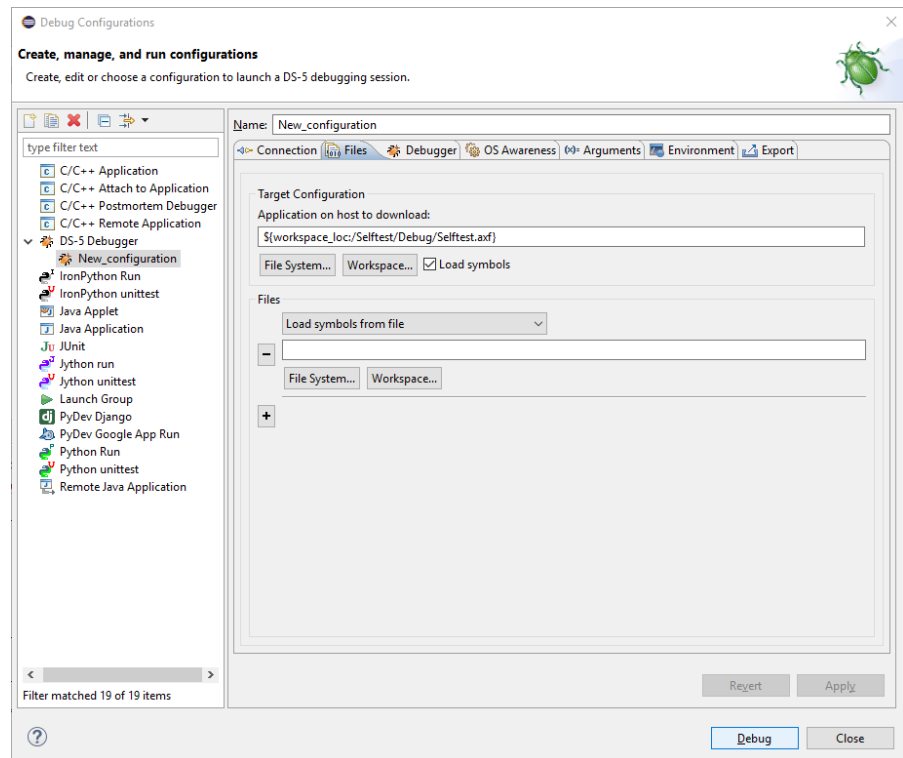


Figure 8-1 : MPS3 Board Debug Connector Locations

2. Open the Debug Configurations dialog box (by right-clicking on the newly created debug session in section 7.3.3 (in the Debug Control Tab)) and ensure that:
 - a. The Cortex-M0+ target is selected under the **Connection** tab.
 - b. Click the **Debugger** tab, select **Debug from entry point** under **Run control**.

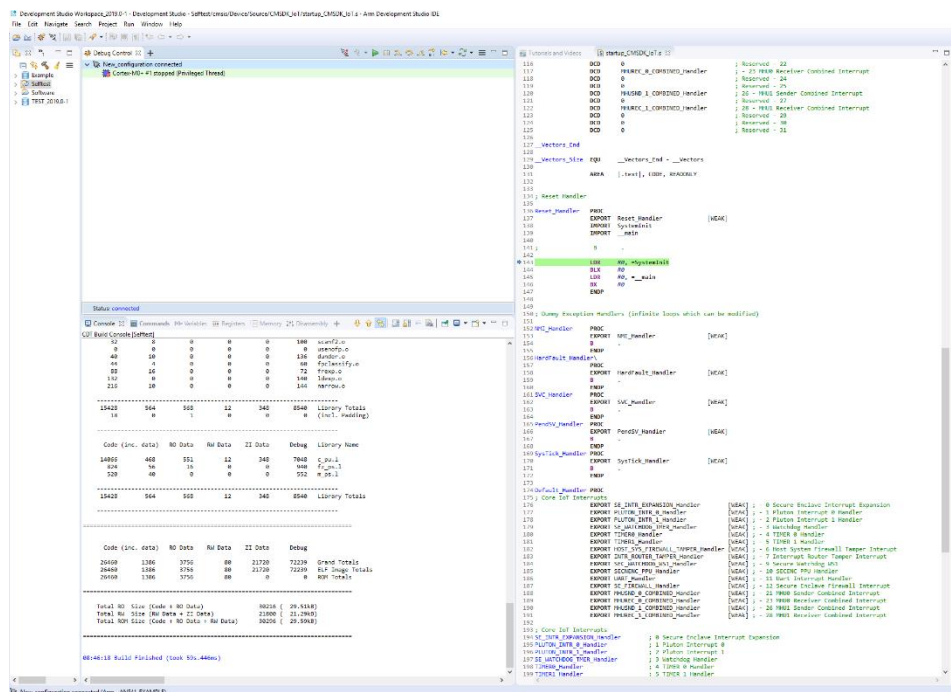
- c. The file `Selftest.axf` that was created in Section 7.4 is entered in the **Application on host to download** field under the **Files** tab and the **Load symbols** checkbox is selected.


An example dialog box is shown below. This example is pointing to the example made in the workspace.



3. Click **Debug**.

4. A debug session is established that has a connection to the SE Cortex-M0+ which has the code waiting at the debug entry point:



-
5. Program execution at this stage can be either single-stepped or set to Run .
 6. On the SE UART, the Cortex-M0+ boots into the test code, as shown below:

```
MPS3 Prototyping Board (V2M-MPS3) Test Suite
Version 2.0 Build date: Aug 12 2020
AN543 V3 Arm Corstone-700 for MPS3
Copyright (C) ARM Ltd 2020. All rights reserved.

CPU: Cortex-M0+ r0p1

Summary of results
=====
 1 Test OCVM (DDR4) & CVM (BRAM) : Not Run
 2 Test XNVM (QSPI FLASH)       : Not Run
 3 Test Ethernet                 : Not Run
 4 Test Audio                    : Not Run
 5 Test USB                      : Not Run

Select the test you wish to run. (X - Exit)

Choice:
```

7.6 Modifying the SSE-700 Boot Register

7.6.1 SSE-700 Boot Register Overview and Volatility

The Host System includes a Boot Register which is a write-once set of registers providing the initial instructions to the Host CPU. The boot register can only be written to from the SE. If the Boot Register is written to more than once, an error response is generated.

Note

1. The embedded test image in the SE ROM writes to the Boot Register as part of its initialisation routine.
2. The Selftest Software application also has identical code that writes to this Boot Register as part of its initialisation routine.
3. The Boot Register is only volatile to an MPS3 board reboot.
4. The debugger cannot reset the domain within which the Boot Register resides.
5. If you attempt to debug code, using the debugger, that has previously written to the Boot Register and this code attempts a subsequent write to the Boot Register then this causes an error response.

See the Boot Register section in the *Arm® Corstone™ SSE-700 Subsystem Technical Reference Manual* for more details.

7.6.2 MPS3 Boot Register Control

The software initialisation code to setup the Boot Register in both the SE ROM embedded test image and the Selftest Software is identical. It uses the following mechanism to avoid incremental writes to the Boot Register post MPS3 reboot or reset. An SE register with the same volatility as the Host system Boot Register is utilised as a flag register. This flag register is used to provide a record of when the Boot Register is written to.

The software initialisation code performs the following steps:

1. After MPS3 reboot (see section 8.7.2 for details on MPS3 reboot and reset) the code reads the SE_GP1 (SE General Purpose 1) register and checks whether bit[0] is set to 0b0.
2. If SE_GP1 bit[0] is read as 0b0, this indicates that the Boot Register has not been written to and the code commences to step 3. If SE_GP1 bit[0] is read as 0b1, this is a flag to indicate that the Boot Register has been written to. The code does not attempt to write to the Boot Register and the code moves to step 5.
3. The Boot Register is written to with the contents as defined in the code and continues to step 4.
4. SE_GP1 register bit[0] is written to as 0b1. This sets the flag bit to indicate that the Boot Register has been written to.
5. The code exits the Boot Register setup phase.

See the Secure Enclave System Control register summary in the *Arm® Corstone™ SSE-700 Secure Enclave Technical Reference Manual* for more details.

7.6.3 Modifying the Boot Register

If changes are made to source code that require a change to the Boot Register, then the following steps must be followed:

1. The template for the code that performs the steps in section 7.6.2 can be found in the example software project, on the SDCARD in the following directory:

Software/selftest/cmsis/Device/Source/CMSDK_IoT/system_CMSDK_IoT.c

The screenshot below shows the provided code from the Selftest software project:

```
78 /*-----*/
79 SE General Purpose Registers in the SE AON region
80 *-----*/
81 #define SE_SYS_CTRL_GP_REG1 0x50080018
82
83 void SystemCoreClockUpdate (unsigned int clock)
84 {
85     SystemCoreClock = clock;
86 }
87
88 void sec_enc_clk_divide(void)
89 {
90     se_sys_ctrl_reg *sys_ctrl_reg = (se_sys_ctrl_reg *) (SE_SYS_CTRL_REG);
91     sys_ctrl_reg->SE_CLK_DIV = 0x0;
92 }
93
94 //Integration layer init function
95 __attribute__((weak)) void il_init(void)
96 {
97     volatile unsigned int *se_sys_ctrl_gp_reg1_word = (volatile unsigned int *) SE_SYS_CTRL_GP_REG1;
98     se_base_sys_ctrl_reg *se_base_sys_ctrl_reg_base = (se_base_sys_ctrl_reg *) SE_BASE_SYS_CTRL_REG;
99
100 // We are using Secure Enclave control, general purpose register 1 (GP1) which is volatile to an AONTOPPORESETn
101 // (which is asserted as part of PORESETn), to preserve as record of the state of whether the host system boot registers have been written
102 // The boot registers are, by design, write once only and any subsequent writes will generate
103 // an error response. The below code implements a basic mechanism to set a flag that has the same
104 // volatility as the "write once" boot registers.
105 // If the flag bit is not set, program the registers and set the flag bit.
106 // If the flag bit is set, do not program the registers.
107 // Please see the application note documentation for more information around this software,
108 // the implementation and considerations if the boot registers need to be modified between
109 // software builds.
110 // This is a simple approach and is not intended to handle any unlikely race conditions
111 // (where the flag bit may get set and the subsequent write to the boot register(s), for
112 // whatever reason, doesn't occur).
113
114 if ((*se_sys_ctrl_gp_reg1_word) != 0x1) {
115     // Boot registers first setup, set flag to say they are being set up
116     *se_sys_ctrl_gp_reg1_word = 0x1;
117
118     //programming boot instruction registers
119     MEM_W(BOOT_REGISTER, 0x0) = 0xEA7FFFFE;
120     MEM_W(BOOT_REGISTER, 0x4) = 0xEA7FFFFE;
121     MEM_W(BOOT_REGISTER, 0x8) = 0xEA7FFFFE;
122     MEM_W(BOOT_REGISTER, 0xC) = 0xEA7FFFFE;
123 }
124 }
125
126
```

2. Compile the code, with the new Boot Register and build an executable file. In this example project an AXF file is built.
3. Copy the built AXF file to the MPS3 SDCARD in the Software/ directory.
4. Ensure the images.txt file is configured such that the MCC preloads the SE ROM with the newly built AXF, from step 3 above, at MPS3 reboot time. See section 8.8.2 for more details on the images.txt file and MCC preloading of the SE ROM.
5. Reboot the MPS3 board.
6. After the MCC preloads the SE ROM with the newly built AXF and releases CPUWAIT, the Cortex-M0+ boots from the newly built AXF file and the Boot Registers assume their new setting(s).

8 Using AN543 on the MPS3 Board

8.1 Pre-Requisites

1. Familiarity with MPS3:

- Familiarisation with the *Arm® MPS3 FPGA Prototyping Board Technical Reference Manual*.
- Understanding of the configuration and boot flow described in the MPS3 TRM.

Ensure you have gained familiarity with the MPS3 board and are able to:

1. Connect a PC to the MPS3 board using a USB connection (which is required to load files onto the MPS3 board SD card in order to run the built .bit file from the FPGA build flow).
2. Power the MPS3 board.
3. The MPS3 board appears as a mapped drive named “V2M_MPS3”.
4. Understand how to power up, reset and establish a serial terminal over the USB connection to a host PC.

8.2 Loading the Boardfiles onto the MPS3 SD Card

Open the bundle ZIP file, navigate into `Boardfiles/` directory, and copy its contents the root directory of the attached MPS3 drive (The MPS3 SD Card). The required content to be copied is:

- `MB/`
- `SOFTWARE/`
- `config.txt`

Note

It is recommended to delete any previous contents on the SD card in favour of these new contents, however, you might want to manually modify and merge the contents for certain configuration files.

The configuration files that can be modified can be found on the SD card here:

- `SDCARD/config.txt`
- `SDCARD/MB/HBI0309C/board.txt`
- `SDCARD/MB/HBI0309C/AN543/images.txt`

8.3 UART Serial Ports

Four serial ports are supported on this implementation and are accessible through the MPS3 board Debug USB port:

- Serial Port 0 is connected to the MCC and outputs verbose debug information about the status of the MCC.
- Serial Port 1 is connected to the SE UART.
- Serial Port 2 is connected to the SSE-700 UART0 (Cortex-A32 verbose can be observed on this UART when booting with the preloaded FPGA test code)
- Serial Port 3 is connected to the EXTSYS0 UART

Note

The logical<>physical mapping of the serial ports on a host PC can be confusing due to the way the driver may allocate the port numbers. The serial port presented with the lowest number aligns to Serial Port 0 above.

8.4 UART Serial Port Terminal Emulator Settings

All serial ports on this implementation use the following terminal/serial port settings:

Baud Rate: 115200 bps

New-Line: CR (Serial port 0) And LF (Serial Port 1,2 and 3 Only)

Data: 8 bits

Parity: none

Stop: 1 bit

Flow control: none

8.5 MPS3 USB Serial Port Drivers for Windows

Information on installing drivers to support USB serial port on MPS3 can be found at:

<https://community.arm.com/dev-platforms/w/docs/381/accessing-mps3-serial-ports-in-windows-10>

8.6 Running the FPGA image file on the MPS3 Board

Power up the MPS3 board using the PBON push button. The LEDs flash rapidly to indicate that new MCC Firmware is being downloaded (this only occurs the first time the MCC Firmware is updated).

Once booted, the embedded test images in the FPGA run on each CPU, and output is observed on the UARTs.

The assumption here is that no external preload CPU images are setup in `images.txt` file on the SD Card.

On the SE UART (Serial Port 1), the Cortex-M0+ boots SE ROM preload test code, as shown below:

```
*****
*****
**** AN543 v3 Arm Corstone-700 for MPS3 ****
**** SECURE ENCLAVE CM0+ ****
**** CPUID is: 410cc601 ****
*****

SE ROM Basic Byte Write/Read 0x00007000 Test
*****

SE ROM ADDR 0x00007000 Byte access          - Writing...DONE
SE ROM ADDR 0x00007000 Byte access          - Reading...DONE

SE Crypto Accelerator RAM Basic Test
*****

Read Crypto Accelerator Model RAM           : Addr: 2f800000 Read: 0
Write Crypto Accelerator Model RAM          : Writing 0xABCD_FAAF to :2f800000
Read Crypto Accelerator Model RAM           : Addr: 2f800000 Read: abcdfaaf

FPGA IO HOST ADDR 0x40010000 & SCC Regs HOST ADDR 0x40000000 Word access
*****

Read FPGA SCC AID Reg                       : Addr: 80000ff8 Read: 3100708
Read FPGA SCC ID Reg                       : Addr: 80000ffc Read: 41045430
Read FPGA IO GP1 Reg (SCB 31 to 0)         : Addr: 80010050 Read: ffffffff
Read FPGA IO GP2 Reg (SCB 63 to 32)        : Addr: 80010054 Read: ffffffff
Read FPGA IO GP3 Reg (SCB 95 to 64)        : Addr: 80010058 Read: f00ff00f
Read FPGA IO GP4 Reg (SCB 127 to 96)       : Addr: 8001005c Read: abba1234

XNVM Basic Test
*****

XNVM FLASH HOSTA ADDR 0x08000000 Word access - Reading...DONE
XNVM FLASH HOSTA ADDR 0x08000000 Byte access - Reading...DONE

OCVM Basic Test
*****

OCVM DDR4 HOST ADDR 0x80000000 Byte access   - Writing...DONE
OCVM DDR4 HOST ADDR 0x80000000 Byte access   - Reading...DONE
OCVM DDR4 HOST ADDR 0xF8000000 Byte access   - Writing...DONE
OCVM DDR4 HOST ADDR 0xF8000000 Byte access   - Reading...DONE

CVM Basic Test
*****

CVN HOST ADDR 0x02300000 Byte access         - Writing...DONE
CVN HOST ADDR 0x02300000 Byte access         - Reading...DONE

EXPMST1 Basic Test
*****

EXPMST1 HOST ADDR 0x60000000 Byte access     - Writing...DONE
EXPMST1 HOST ADDR 0x60000000 Byte access     - Reading...DONE

De-asserting HOST CPUWAIT
*****

Deasserted Host (CA32) CPUWAIT
TEST PASSED OK
```

On the SSE-700 UART0 (Serial Port 2), the Cortex-A32 boots preload test code, as shown below:

```
*****
*****
**** AN543 v3 Arm Corstone-700 for MPS3 ****
****      HOST CPU CA32 MP1      ****
*****
*****

FPGA IO HOST ADDR 0x40010000 & SCC Regs HOST ADDR 0x40000000 Word access
*****

Read FPGA SCC AID Reg           : Addr: 40000ff8 Read: 3100708
Read FPGA SCC ID Reg           : Addr: 40000ffc Read: 41045430
Read FPGA IO GP1 Reg (SCB 31 to 0) : Addr: 40010050 Read: ffffffff
Read FPGA IO GP2 Reg (SCB 63 to 32) : Addr: 40010054 Read: ffffffff
Read FPGA IO GP3 Reg (SCB 95 to 64) : Addr: 40010058 Read: f00ff00f
Read FPGA IO GP4 Reg (SCB 127 to 96) : Addr: 4001005c Read: abba1234

XNVM Basic Test
*****

XNVM FLASH HOSTA ADDR 0x08000000 Word access      - Reading.....DONE
XNVM FLASH HOSTA ADDR 0x08000000 Byte access      - Reading.....DONE

OCVM Basic Test
*****

OCVM DDR4 HOST ADDR 0x80000000 Byte access        - Writing.....DONE
OCVM DDR4 HOST ADDR 0x80000000 Byte access        - Reading.....DONE
OCVM DDR4 HOST ADDR 0xF8000000 Byte access        - Writing.....DONE
OCVM DDR4 HOST ADDR 0xF8000000 Byte access        - Reading.....DONE

CVM Basic Test
*****

CVN HOST ADDR 0x02300000 Byte access              - Writing.....DONE
CVN HOST ADDR 0x02300000 Byte access              - Reading.....DONE

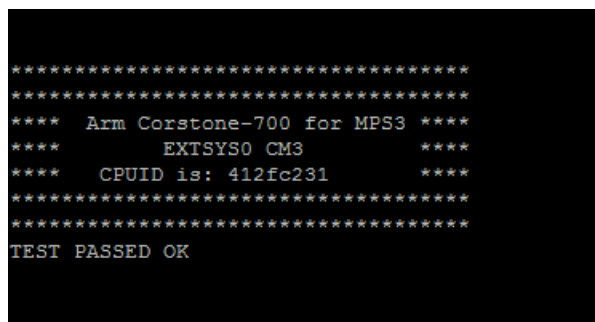
EXPMST1 Basic Test
*****

EXPMST1  HOST ADDR 0x60000000 Byte access        - Writing.....DONE
EXPMST1  HOST ADDR 0x60000000 Byte access        - Reading.....DONE

De-assert EXTSYS0 CPUWAIT
*****

Extsys0 CPUWAIT de-asserted.
TEST PASSED
```

On the EXTSYS0 UART (Serial Port 3), the Cortex-M3 boots preload test code, as shown below:



```
*****
*****
**** Arm Corstone-700 for MPS3 ****
**** EXTSYS0 CM3 ****
**** CPUID is: 412fc231 ****
*****
*****
TEST PASSED OK
```

Note

If the MPS3 board does not boot correctly, refer to the log file LOG.TXT on the SD Card.

8.7 MCC Debug UART – Serial Port 0

The MPS3 board supports an MCC debug UART interface on serial port 0. This interface has the following functionality:

- Bi-directional CLI interface.
- Verbose status output of MPS3 during the power on and boot stages.
- Support to reboot the MCC using command-line syntax.

8.7.1 MCC Debug UART – Verbose Output

An example output from the MCC UART can be seen below:

```
ARM V2M-MPS3 Boot loader v1.0.0
HBI0309 build 1044

ARM V2M-MPS3 Firmware v1.4.1
Build Date: Apr 28 2020
Enabling usb remote...
USB Serial Number = 5005330550290

Cmd>
Powering up system...
Switching on main power...
Configuring motherboard (rev C, var A)...

Reading Board File \MB\HBI0309C\AN543\AN543_v3.txt

Configuring FPGA from file \MB\HBI0309C\AN543\AN543_v3.bit
Address: 0x02880000
FPGA configuration complete.

OSCCLK0 : 25.000000MHz
OSCCLK1 : 32.000000MHz
OSCCLK2 : 50.000000MHz
OSCCLK3 : 50.000000MHz
OSCCLK4 : 24.576000MHz
OSCCLK5 : 23.750000MHz

OSCCLK setup: PASSED
GTXCLK (100): PASSED

Writing SCC 0x010 with board revision C
Writing SCC 0x014 with ACLK 32000000 Hz
Writing SCC 0x000 with 0x00000000
Reading SCC 0xFF8 with 0x03100708
Reading SCC 0xFFC with 0x41045430

Reading images file \MB\HBI0309C\AN543\images.txt

Setting QSPI to XIP (QSPI) mode...
SMSC9220 was identified successfully.
MAC addr test: PASSED

UART0: MCC, UART1: FPGA0
Releasing CB_nRST
Clearing SMC_CPUWAIT (0x06000018:0x00000000)
Enabling debug USB.
USB Serial Number = 5005330550290
```

The verbose output above shows:

- The MCC booting
- The MCC reading the configuration files from the SD CARD
- The MCC configuring the FPGA with the target .bit file
- The MCC releasing the Cortex-M0+ in the SE from reset and de-assertion of CPUWAIT

8.7.2 MCC Debug UART – Reboot Control

The command `reboot` on the terminal emulator console initiates MCC reboot. The following screenshot shows this command.

```
reboot
Rebooting...
Disabling debug USB..
Board rebooting...

ARM V2M-MPS3 Firmware v1.4.1
Build Date: Apr 28 2020

Powering up system...
Switching on main power...
Configuring motherboard (rev C, var A)...

Reading Board File \MB\HBI0309C\AN543\AN543_v3.txt

Configuring FPGA from file \MB\HBI0309C\AN543\AN543_v3.bit
Address: 0x02880000
FPGA configuration complete.

OSCCLK0 : 25.000000MHz
OSCCLK1 : 32.000000MHz
OSCCLK2 : 50.000000MHz
OSCCLK3 : 50.000000MHz
OSCCLK4 : 24.576000MHz
OSCCLK5 : 23.750000MHz

OSCCLK setup: PASSED
GTXCLK (100): PASSED

Writing SCC 0x010 with board revision C
Writing SCC 0x014 with ACLK 32000000 Hz
Writing SCC 0x000 with 0x00000000
Reading SCC 0xFF8 with 0x03100708
Reading SCC 0xFFC with 0x41045430

Reading images file \MB\HBI0309C\AN543\images.txt

Setting QSPI to XIP (QSPI) mode...
SMSC9220 was identified successfully.
MAC addr test: PASSED

UART0: MCC, UART1: FPGA0
Releasing CB_nRST
Clearing SMC CPUWAIT (0x06000018:0x00000000)
Enabling debug USB.
USB Serial Number = 5005330550290
```

8.8 Preload of the SE ROM, EXTSYS0 Code SRAM and the QSPI Flash

The MPS3 MCC supports the pre-loading of the SE ROM (implemented as a RAM), the EXTSYS0 Cortex M3 Code SRAM and the QSPI Flash on the MPS3 board at MCC boot time. The user can control this process by editing `image.txt` file, which specifies the location of each image on the SD card, and whether it is pre-loaded or not.

8.8.1 The Preload Sequence

The preload sequence is:

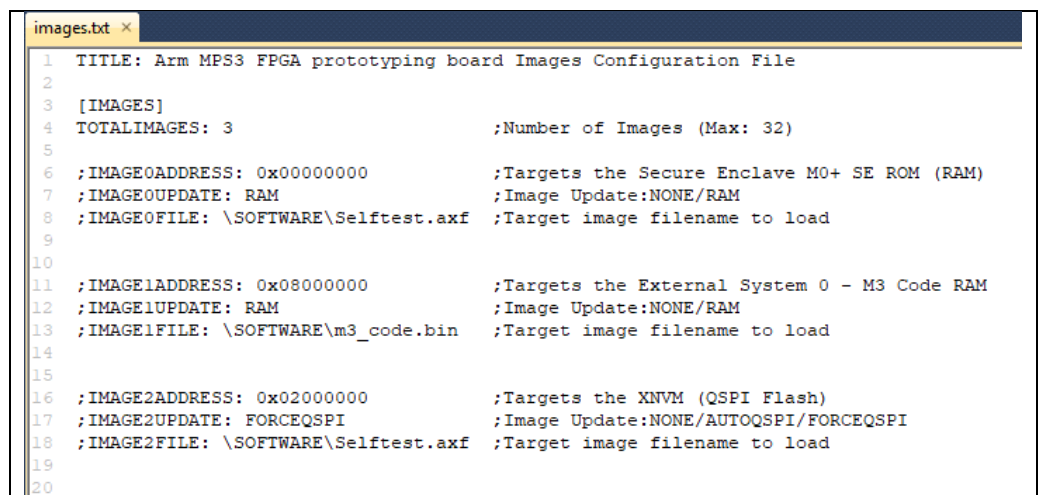
1. MPS3 is powered up.
2. The “PBON” button on the MPS3 board is pressed.
3. The MCC configures the FPGA with the FPGA image, which is specified in the FPGA configuration file in `MB/HBI0309{<boardrev>}/AN543/an543_v3.txt`.
4. The FPGA SoC NIC-400 reset is released.
5. SSE-700 Top and EXTSYS0 are held in reset and the SE Cortex-M0+ CPUWAIT is held asserted.
6. The MCC can then preload the SE ROM, the EXTSYS0 Cortex-M3 Code SRAM and the QSPI Flash using the specified images (if directed to do so in the `images.txt` file settings (see section 8.8.2 for more details)).
7. The MCC de-asserts reset to SSE-700 TOP and EXTSYS0.
8. The MCC then de-asserts the SE Cortex-M0+ CPUWAIT.
9. The SE Cortex-M0+ boots from the SE ROM, booting into the code preloaded into ROM from the designated image file.

8.8.2 Preload Image Configuration (images.txt)

Preload image configuration:

Images to be preloaded are selected/configured by modifying the following file on the SDCARD `Boardfiles/MB/HBI0309C/AN543/images.txt` (Note that in this example the file has been opened in the directory for an MPS3 Rev C board).

For convenience, the file in the bundle has been templated with the syntax (commented out) to load the SE ROM and the QSPI Flash with a prebuilt `Selftest.axf`, also supplied in the bundle. ESO can also be preloaded with a `.bin` file image. This can be observed below:



```
1 TITLE: Arm MPS3 FPGA prototyping board Images Configuration File
2
3 [IMAGES]
4 TOTALIMAGES: 3 ;Number of Images (Max: 32)
5
6 ;IMAGE0ADDRESS: 0x00000000 ;Targets the Secure Enclave M0+ SE ROM (RAM)
7 ;IMAGE0UPDATE: RAM ;Image Update:NONE/RAM
8 ;IMAGE0FILE: \SOFTWARE\Selftest.axf ;Target image filename to load
9
10
11 ;IMAGE1ADDRESS: 0x08000000 ;Targets the External System 0 - M3 Code RAM
12 ;IMAGE1UPDATE: RAM ;Image Update:NONE/RAM
13 ;IMAGE1FILE: \SOFTWARE\m3_code.bin ;Target image filename to load
14
15
16 ;IMAGE2ADDRESS: 0x02000000 ;Targets the XNVM (QSPI Flash)
17 ;IMAGE2UPDATE: FORCEQSPI ;Image Update:NONE/AUTOQSPI/FORCEQSPI
18 ;IMAGE2FILE: \SOFTWARE\Selftest.axf ;Target image filename to load
19
20
```

Note

A “;” at the start of any line in the `images.txt` file denotes a comment and is ignored by the MPS3 MCC. Note that lines 6-8, 11-13 and 16-18 in `images.txt` in the bundle are commented out.

Line 4 of the `images.txt` file denotes how many images are required to be loaded by the MCC on boot. With `TOTALIMAGES` set to 3 the MCC is expecting 3 definitions (within the `images.txt` file) of images to load, defined as “`IMAGE0xxxx`”, “`IMAGE1xxxx`” and “`IMAGE2xxxx`” (where `xxxx` represents parameters such as `ADDRESS`, `UPDATE` and `FILE`). If the `TOTALIMAGES` parameter is set to 1, the MCC expects valid syntax in the file for “`IMAGE0xxxx`” and ignore “`IMAGE1xxxx`”. If the `TOTALIMAGES` parameter is set to 1 and the MCC can’t find any entries in the file for “`IMAGE0xxxx`”, it skips any loading or may report an error.

Lines 6 and 7 specify whether the SE ROM is to be loaded with the “`IMAGE0FILE`” specified on line 8. In this example the SE ROM does not get loaded with the “`IMAGE0FILE`” when the MCC boots, as these lines (6-8) are commented out. If these lines (6-8) have their “;” comment character removed, the MCC would preload the SE ROM with “`IMAGE0FILE`” on boot.

Lines 11 and 12 specify whether the ES0 SRAM is to be loaded with the “`IMAGE1FILE`” specified on line 13. In this example the ES0 SRAM does not get loaded with the “`IMAGE1FILE`” when the MCC boots, as these lines (11-13) are commented out. If these lines (11-13) have their “;” comment character removed, the MCC would preload the ES0 SRAM with “`IMAGE1FILE`” on boot.

Lines 16 and 17 specify whether the QSPI Flash is to be loaded with the “`IMAGE2FILE`” specified on line 18. In this example the QSPI Flash does not get loaded with the “`IMAGE2FILE`” when the MCC boots, as these lines (16-18) are commented out. If these lines (16-18) have their “;” comment character removed, the MCC would preload the QSPI Flash with “`IMAGE2FILE`” on boot.

You can modify the file whilst the MCC/system is booted and running, the file is read by the MCC only on boot.

In order to reboot the MCC after an image file configuration change, you must do one of the following:

- a. press the **PBRST** button on the MPS3 board, followed by a press of the **PBON** button to reboot the MCC.
- b. Issue a **reboot** command on the MCC console, as described in section 8.7.2.

The image files, specified on lines 8 and 13, can be modified to use a user specified file that differs in file name and/or file type. See section 8.10 for details on naming limitations of these files.

8.9 Supported Preload SE ROM, EXTSYS0 Code SRAM and Flash File Types

The MPS3 MCC supports preload of the following file types:

- AXF - Object file, with an `axf` filename extension
- Binary – Binary file with a `bin` filename extension.

8.10 Supported Preload SE ROM, EXTSYS0 Code SRAM and Flash File Naming Format

The files that can be preloaded by the MCC must follow the following file naming format:

- The MPS3 MCC uses the SFN (short filename) 8.3 format.
- The filename must have a maximum length of 8 characters.
- The filename must have a three-character extension.