Arm® Corstone™-1000 for MPS3 Application Note AN550

Non-Confidential

Issue D

DAI 0550

Arm® Corstone™-1000 for MPS3

Application Note AN550

Release information

Document history

Issue	Date	Confidentiality	Change
A	25 May 2021	Confidential	First release
B	7 Dec 2021	Confidential	V1 bitfile and updated image captures
C	20 Jan 2022	Non-Confidential	Non-Confidential proprietary Notice 1.9 Additional required hardware – section added Figure 2-1: System Overview – updated 2.4.7 Flash - updated
D	04 Jan 2023	Non-Confidential	Version of bitfile changed to V2 and updated image captures, fixed hyperlink captions 8.10 Other applications - updated 7.3.3 Creating a Development Studio Debug Connection - updated 7.5 Establishing a Debug Session – minor fixes. 8.7 Preload of the SE ROM, EXTSYS0 Code SRAM, OTP and the QSPI Flash - updated

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.

Product Status

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

Web Address

developer.arm.com

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email **terms@arm.com**.

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-1000 product and other Arm technology.

Software Binaries:
Motherboard Configuration Controller binary, including Arm® 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 Selftest software project

Part C

None

Part D

None

Contents

1 Introduction

1.1 Purpose of this application note

This document describes the features and functionality of AN550 SMM. AN550 SMM is an FPGA implementation based on the Corstone-1000 product, which is extended to support a single external system and hardware peripherals.

1.2 Intended audience

This application note document is written for experienced hardware, System-on-Chip (SoC) and software engineers who might or might not have experience with Arm products. Such engineers typically have experience in writing Verilog and of performing synthesis but might have limited experience of integrating and implementing Arm products.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

1.3.1 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.

1.3.2 Typographical conventions

Convention	Use
italic	Introduces 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.
`monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace` <u>`underline`</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<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>`

Convention	Use
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.
⚠ Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
⚠ Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
⚠ Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
✎ Note	This represents an important piece of information that needs your attention.
💡 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
📌 Remember	This is a reminder of something important that relates to the information you are reading.

1.4 Additional reading

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

Table 1-1 Arm publications

Document name	Document ID	Licensee only
Arm® MPS3 FPGA Prototyping Board Technical Reference Manual	100765	No
Arm® Cortex®-M System Design Kit Technical Reference Manual	DDI 0479	No
Arm® Corstone™ SSE-710 Subsystem Technical Reference Manual	102342_0000_01_en	No
Arm® Corstone™ SSE-710 Subsystem Configuration and Integration Manual	102343_0000_01_en	Yes
Arm® CoreLink™ SSE-050 Subsystem Technical Reference Manual	100918_0001_00_en	No

Table 1-2 Other publications

Document name	Document ID
Xilinx Vivado Design Suite User Guide	*UG909*

1.5 Feedback

Arm welcomes feedback on this product and its documentation.

1.5.1 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.

1.5.2 Feedback on content

If you have comments on content, send an email to errata@arm.com and give:

- The title *Arm® Corstone™-1000 for MPS3 Application Note AN550*.
- The number DAI 0550, Issue D.
- 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.

1.5.3 Other information

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

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.

1.6 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 Manager
EXTSYS0	External System 0
FPGA	Field Programmable Gate Array
FW	Firewall
GIC	Generic Interrupt Controller
JTAG	Joint Test Action Group
KB	Kilo Byte
LAR	Long Address Range (preloading mechanism)
LUTs	Lookup tables
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 system implemented as an FPGA image and described in this AN
SoC	System on Chip
SWD	Serial Wire Debug
TBD	To Be Defined
TRM	Technical Reference Manual
UART	Universal Asynchronous Receiver/Transmitter

UPC	Universal Part Code
USB	Universal Serial Bus
WI	Write Ignored
XIP	eXecute-In-Place
XNVM	eXecute-in-place Non-volatile Memory (XNVM)

1.7 Subsystem version details

Version	Descriptions
r1p2	CoreLink NIC-400
r1p1	Cortex-M System Design Kit
r0p0	Corstone SSE-710 Subsystem
r0p1	CoreLink SSE-050 Subsystem
r0p1	Cortex-M0+ MCU
r1p0	Cortex-A35 CPU
r2p1	Cortex-M3 MCU

Table 1-3 :IP Versions

1.8 Encryption key

Arm supplies the MPS3 prototyping board with a decryption key programmed into the FPGA. This key is needed to enable 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.

1.9 Additional required hardware

Warning

A 32 MB QSPI flash PMOD module is required to support the XNVM memory (see section 3.2) in this implementation. If this is not fitted, the software application referenced in section 8.10 and section 7.2, will not function as described in this document.

The table below shows the PMOD module required for this implementation:

Description	Manufacturer	UPC
PMOD 32MB SF3 QSPI Flash Board	Digilent	645220775590

Table 1-4: Additional hardware

The PMOD module is not distributed as part of the orderable MPS3 board product.

If the module is not fitted, the following may be observed when running the software applications in section 7.2 and 8.10:

- When running the sefltest program (described in section 7.2), and selecting the #2 Test XNVM (QSPI FLASH), the test will initiate but never complete.
- When not preloading selftest the SE will initiate preload test code in 8.5 but never complete.
- The software application, referenced in section 8.10 will not successfully boot and run as expected.

2 Overview

This SMM is based on the Corstone-1000 product. The product is then extended to add a single external system and peripherals to support software development.

2.1 SSE-710 Subsystem

The SSE-710 Subsystem has configurable options. These options are documented in *Arm® Corstone SSE-710 Configuration and Integration Manual*, Chapter 2 Installation, Configuration, and the OoB test.

The following table shows some of the most important configuration settings and also where AN550 SMM uses non-default values:

Parameter	Implemented Values	Default Values	Description
NEON_FP	"TRUE"	"FALSE"	NEON_FP and FPU present
HOST_CPU_TYPE	Cortex-A35 {2}	Cortex-A32 {1}	Host CPU type
HOST_CPU_NUM_CORES	1	4	Number of Host CPU cores
SEC_ENC_ROM_SIZE	64	32	Size of Secure Enclave ROM in KB
SEC_ENC_RAM_SIZE	512	128	Size of Secure Enclave RAM in KB
EXPMST0_NUM_RGN	8	32	Number of regions for EXPMST0 Firewall Component
EXPMST1_NUM_RGN	8	32	Number of regions for EXPMST1 Firewall Component

Table 2-1 : SSE-710 configuration options

The following table shows some of the most important external to SSE-710 Subsystem settings:

Port	Implemented Values	Meaning	Description
SOCID	0x76B0023B	AN550 (CS1000 FPGA)	SoC Identification
OCVMSIZE	0x1F	2GB	Off-Chip Volatile Memory
XNVMSIZE	0x19	32MB	eXecute in place Non-Volatile Memory
CVMSIZE	0x16	4MB	On-Chip Volatile Memory

Table 2-2: SSE-710 external setting

2.2 System Block Diagram

The diagram below shows a high-level view of the full MPS3 Prototyping Board Corstone-1000 product implementation.

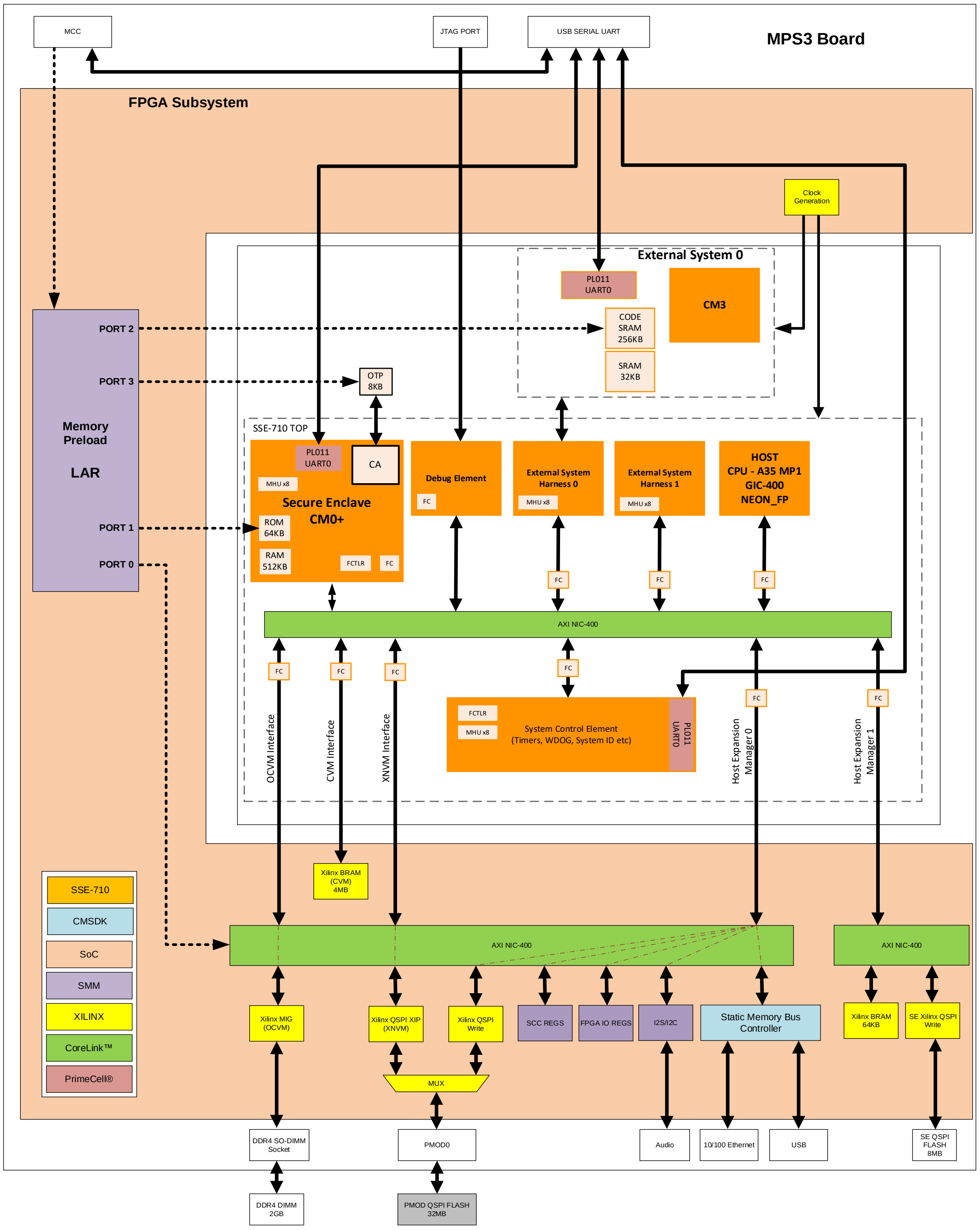


Figure 2-1 : System Overview

2.3 EXTSYS0 - External System 0

The implementation supports a single external system populating one of the EXTSYS sockets in SSE-710. The External System 0 is based on a modified SSE-050, a Cortex-M3 based subsystem.

2.3.1 EXTSYS0 Block Diagram

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

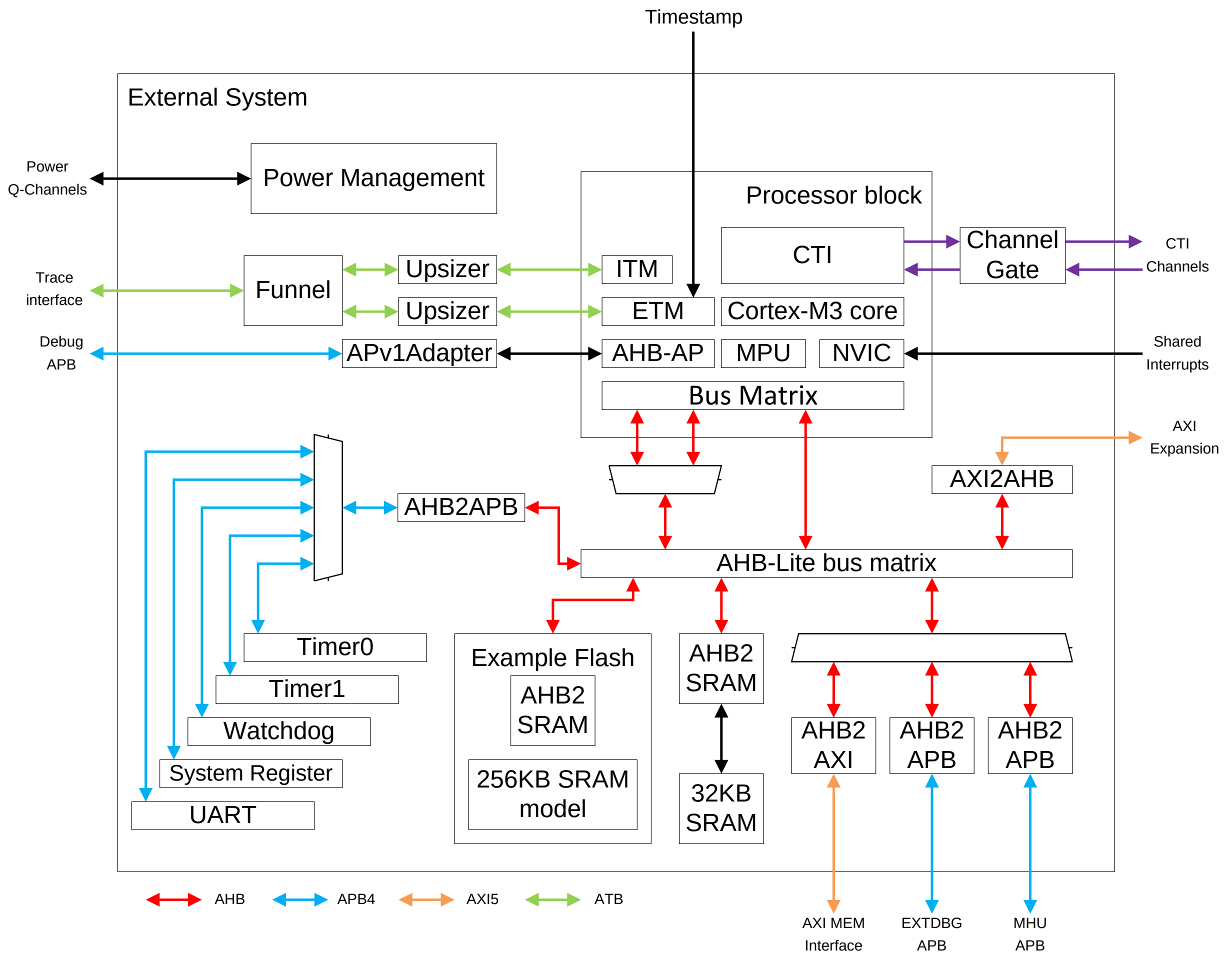


Figure 2-2 : EXTSYS0 System Overview

2.4 Components

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

2.4.1 SSE-710 Subsystem

The following sections detail the components used in the SSE-710 Subsystem.

2.4.1.1 Secure Enclave

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

2.4.1.1.1 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 AN550 SMM 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 config SD card. 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.

2.4.1.2 Single External System (EXTSYS0)

External System 0 includes the following:

- Modified SSE-050 Cortex-M3 based System with Arm® 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-710 interfaces.
- ATB network modules to enable connection all trace sources to the SSE-710.
- Channel Gate to be able to limit cross-trigger functionality based on the SE Life Cycle State.
- Addition of PL011 UART to APBTARGEXP2.

2.4.1.2.1 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 AN550 SMM supports the following:

- 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 the above, 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 SD card. 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.

2.4.1.3 Host System CPU Configuration

The Host System CPU is a Cortex-A35 (MP1 single core), which is configured with the following parameters:

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

2.4.1.4 Host System Configuration

The Host System is configured with the following parameters:

- SSE-710 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)

2.4.1.5 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	0xDEADDEADDEADDEADDEADDEADDEADDEADD EAD

2.4.2 NIC-400

The NIC-400 connects the SSE-710 Subsystem to the FPGA Subsystem peripherals.

2.4.3 Xilinx QSPI Controller

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

2.4.4 Xilinx MIG – DDR4

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

2.4.5 DDR4 (OCVM) SODIMM EEPROM SBCon

The SMM implements a single SBCon I^{2}C module to read the EEPROM on the DDR4 SODIMM Module. See section 3.8 for more details.

2.4.6 Debug UART Peripheral

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

- Secure Enclave UART0 (PrimeCell PL011).
- HOST Subsystem UART0 (PrimeCell PL011).
- HOST Subsystem UART1 Secure (PrimeCell PL011).
- EXTSYS0 UART0 (PrimeCell PL011).
- 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 8.6 on MCC Debug UART for more details.

2.4.7 Flash

- 32MB of QSPI flash is supported on the platform and is accessible for reading and writing through the EXPMST0 interface, and for read only XIP through the XNVM interface. The external QSPI PMOD flash module fitting is shown in the Loading the Boardfiles onto the MPS3 SD Card section. The QSPI flash can be preloaded by the MCC and is accessible from the SE Cortex-M0+, Host Cortex-A35 CPU and EXTSYS0. The QSPI Xilinx controller is utilized to support QSPI flash.
- SE 8MB of QSPI flash is supported on the platform and is connected to the SSE-710 EXPMST1 port and the intention is to be accessible only from the SE Cortex-M0+. The QSPI Xilinx controller is utilized to support QSPI flash.

2.4.8 USB

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

2.4.9 10/100 Ethernet

The implementation connects a Microchip LAN9220 device through a static memory interface. The device is accessible from the SSE-710 Expansion MSTR0 interface.

2.4.10 Audio

The SMM implements a single I^{2}S and a single I^{2}C module directly connected to the MPS3 back panel audio sockets. I^{2}S is used for data and I^{2}C is used for control. The device is accessible from the SSE-710 Expansion MSTR0 interfaces.

2.4.10.1 SBCon I^{2}C

A single two-wire I^{2}C SBCon module supports configuration of the Cirrus Logic, Stereo CODEC (CS42L52) on the MPS3 board. See section 3.8 for more details.

2.4.10.2 I^{2}S Interface

A single I^{2}S module connects the internal APB bus to the external I^{2}S Audio CODEC. See section 3.9 for more details.

2.4.11 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.

2.4.12 FPGA Utilization

AN550 SMM is designed for MPS3 board which uses a Xilinx Kintex Ultrascale XCKU115 FPGA. The FPGA features up to 8MB BRAM (2160 BlockRAM tiles) and up to 663,360 LUTs.

Full part number: XCKU115-FLVB1760-1-C.

The following table shows the total number of LUTs and BRAMs that are used in the SD card image.

Site Type	Used	Util%
LUTs	519262	79
BlockRAM Tile	1486	69

Table 2-3 AN550 utilization summary

Note

These numbers relate to the complete image, not individual IP blocks. The numbers must not be used to infer IP size, or the relative sizes of different IP blocks, because the implementation and system design can significantly differ.

2.5 Memory Map Overview

2.5.1 SSE-710 Host System Memory Map

The memory map implementation aligns with SSE-710 subsystem Host System memory map. The memory map is expanded to show the supported MPS3 peripherals in the Expansion Manager Regions and their mapping.

See *Arm® Corstone SSE-710 Subsystem Technical Reference Manual* for information on the SSE-710 subsystem Host System memory map.

ROW ID	Address		Size	Region Name	Description
	From	To			
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	0x09FF_FFFF	32MB	XNVM	eXecute-in-place Non-volatile Memory (Note: an external QPSI PMOD Flash module should be connected to MPS3 board)
9	0x0A00_0000	0x0FFF_FFFF	95MB	Reserved	XNVM space that is not implemented
10	0x1000_0000	0x19FF_FFFF	160MB	Debug	
11	0x1A00_0000	0x3FFF_FFFF	608MB	Host Peripherals	
12	0x4000_0000	0x4000_FFFF	64KB	EXPMST0	FPGA – SCC Registers (note only 4KB of registers implemented)
13	0x4001_0000	0x4001_FFFF	64KB	EXPMST0	FPGA – I/O Registers (note only 4KB of registers implemented)
14	0x4002_0000	0x4002_FFFF	64KB	EXPMST0	Audio I^{2}S
15	0x4003_0000	0x4003_FFFF	64KB	EXPMST0	SBCon Audio I^{2}C (Configuration)
16	0x4004_0000	0x4004_FFFF	64KB	EXPMST0	SBCon DDR4 (OCVM) SODIMM EEPROM
17	0x4005_0000	0x4005_FFFF	64KB	EXPMST0	R/W Xilinx QSPI controller (PG153 v3.2)
18	0x4006_0000	0x400F_FFFF	704KB	EXPMST0	Reserved
19	0x4010_0000	0x401F_FFFF	1MB	EXPMST0	Ethernet Controller (LAN9220)

ROW ID	Address		Size	Region Name	Description
	From	To			
20	0x4020_0000	0x402F_FFFF	1MB	EXPMST0	USB Controller (ISP1763)
21	0x4030_0000	0x5FFF_FFFF	1021MB	EXPMST0	Reserved
22	0x6000_0000	0x6000_FFFF	64KB	EXPMST1[1]	Volatile Memory implemented as BRAM
23	0x6001_0000	0x6001_FFFF	64KB	EXPMST1[1]	SE R/W Xilinx QSPI controller (PG153 v3.2)
24	0x6002_0000	0x7FFF_FFFF	511MB	EXPMST1	Reserved
25	0x8000_0000	0xFFFF_FFFF	2GB	OCVM	Off Chip Volatile Memory implemented as off chip DDR4

Table 2-4 : SSE-710 Host System Memory Map

[1] – intention is to be accessible only by SE (depends on the software implementation).

2.5.2 SSE-710 AON Expansion Memory Map

The AON expansion interface resides with the SSE-710 Host Peripheral region in the SSE-710 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 subordinate generates an error response

Table 2-5 : SSE-710 AON Expansion Memory Map

2.5.3 SSE-710 Secure Enclave Memory Map

See the *Arm® Corstone SSE-710 Subsystem TRM* for memory map details.

2.5.4 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	512MB	TARGEXP1	EXTSYS0 MEM Interface of the Harness
3	0x2000_0000	0x2000_7FFF	32KB	TARGSRAM0	SRAM0
4	0x2000_8000	0x3FFF_FFFF	512MB	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

ROW ID	Address		Size	Region Name	Description
	From	To			
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 *
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 subordinate (Error response)
37	0xA001_0000	0xDFFF_FFFF	1023MB	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 subordinate 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 subordinate error)
48	0xE00F_F000	0xE00F_FFFF	4KB	Internal PPB	ROM table

Table 2-6 : EXTSYS0 Memory Map

* - The letters before the MHU describe the source and the transmitter followed by the receiver of that MHU, where H = Host, ES = External System, SE = Secure Enclave.

2.5.5 Memory Sizes – Summary

Entity	Memory Type	Location	Memory Size
Host System	Shared RAM (CVM)	FPGA BRAM	4MB
	Host Expansion Manager 1	FPGA BRAM	64KB
	QSPI Flash (XNVM)	External	32MB
	DDR (OCVM)	External	2GB
Secure Enclave	ROM	FPGA BRAM	64KB
	RAM	FPGA BRAM	512KB
	QSPI Flash (Exp Manager 1)	External	8MB
External System 0	Code SRAM	FPGA BRAM	256KB
	RAM	FPGA BRAM	32KB
OTP	RAM	FPGA BRAM	8KB

Table 2-7 : Memory Sizes - Summary

3 Programmers Model

3.1 Xilinx Internal BRAM

BRAM is used to implement:

- The Secure Enclave ROM (64KB), which is implemented as Dual Port RAM. This is the primary boot memory. See section 2.5 for mapping details.
- The Secure Enclave RAM (512KB). See section 2.5 for mapping details.
- The CVM RAM (4MB). See section 2.5 for mapping details.
- The OTP (8KB). See section 2.5 for mapping details.
- The EXTSYS0 Code SRAM (256KB). See section 2.5.4 for mapping details.
- The EXTSYS0 SRAM (32KB).
- Host Cortex-A35 CPU L1 and L2 cache
- Firewall RAM
- 1KB RAM in the EXPMST1 address space. See section 2.5 for mapping details.

3.2 XNVM QSPI Flash

The XNVM 32MB memory is implemented using an external QSPI flash PMOD module, which is accessed over a QSPI interface (implemented using a Xilinx QSPI controller). See section 2.5 for mapping details.

3.3 SE QSPI Flash

The SE 8MB memory is implemented using on-board Microchip memory, which is accessed over a QSPI interface (implemented using a Xilinx QSPI controller). See section 2.5 for mapping details.

3.4 OCVM DDR4

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

3.5 UART Support

The PrimeCell PL011 UART is implemented within the SSE-710 Subsystem and EXTSYS0. The UARTs are physically accessible through the USB interface on the MPS3 board. See the memory map in section 2.5 for information on the USB interface.

UART Clock input to every UART is connected to REFCLK24MHZ (24MHz).

This implementation provides the UARTs described in chapter 2.4.6.

3.6 SSE-710 Implementation

3.6.1 SSE-710 Power Control

Note

Power control is not supported as it is a limitation of the FPGA implementation.

3.6.2 SSE-710 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. S32KCLK must run to allow the RTC module to count. The interrupt signal of the RTC is connected to the EXPSHDINT interface of the SSE-710 subsystem.

Please refer to the *Arm® Corstone SSE-710 Subsystem Technical Reference Manual* for further details on S32KCLK, AONTOPWARMRESETn and EXPSHDINT.

3.6.3 SSE-710 Expansion Shared Interrupt Map

The following table shows the interrupt connections of the EXPSHDINT interface of the SSE-710. These interrupts can be exposed to the Host CPU, the Secure Enclave, or EXTSYS0 by the Interrupt Router inside SSE-710.

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

Table 3-1 : SSE-710 Expansion Shared Interrupt Map

3.7 EXTSYS0

3.7.1 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 3-2 : 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 subordinate error from its APB interface and it never inserts a wait state.

3.7.2 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-710 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 3-3 : EXTSYS0 System Interrupt Connectivity

3.8 SBCon I^{2}C

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 3-4 SBCon Register Map

3.9 Audio I^{2}S

The I^{2}S interface supports transfer of digital audio to and from the Audio CODEC.

The following table shows the register memory map for I^{2}S Audio registers.

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

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

Address	Name	Description	
		Bits[31:1]	Reserved
		Bit[0]	ITCR
0x40020304	ITIP1	Integration Test Input Register 1	
		Bits[31:1]	Reserved
		Bit[0]	SDIN
0x40020308	ITOP1	Integration Test Output Register 1	
		Bits[31:4]	Reserved
		Bit[3]	IRQOUT
		Bit[2]	LRCK
		Bit[1]	SCLK
		Bit[0]	SDOUT

Table 3-5 Audio I2S Register Map

3.10 SMM Registers

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

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 3.12 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 section 2.5 for mapping details.
- The FPGA I/O register block (see section 3.13 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.

3.11 LEDs

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

LED #	Description
0	SSE-710 Secure Enclave SCB (Security Control Bits) [0]
1	SSE-710 Secure Enclave SCB (Security Control Bits) [1]
2	SSE-710 Secure Enclave SCB (Security Control Bits) [4]
3	SSE-710 Secure Enclave SCB (Security Control Bits) [34]
4	SSE-710 Secure Enclave SCB (Security Control Bits) [35]
5	SSE-710 Secure Enclave SCB (Security Control Bits) [36]
6	SSE-710 Secure Enclave SCB (Security Control Bits) [37]
7	SSE-710 Secure Enclave SCB (Security Control Bits) [48]
8	SSE-710 Secure Enclave SCB (Security Control Bits) [49]
9	SSE-710 Secure Enclave SCB (Security Control Bits) [50]

Table 3-6 : MPS3 Board LED Assignments

3.12 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.

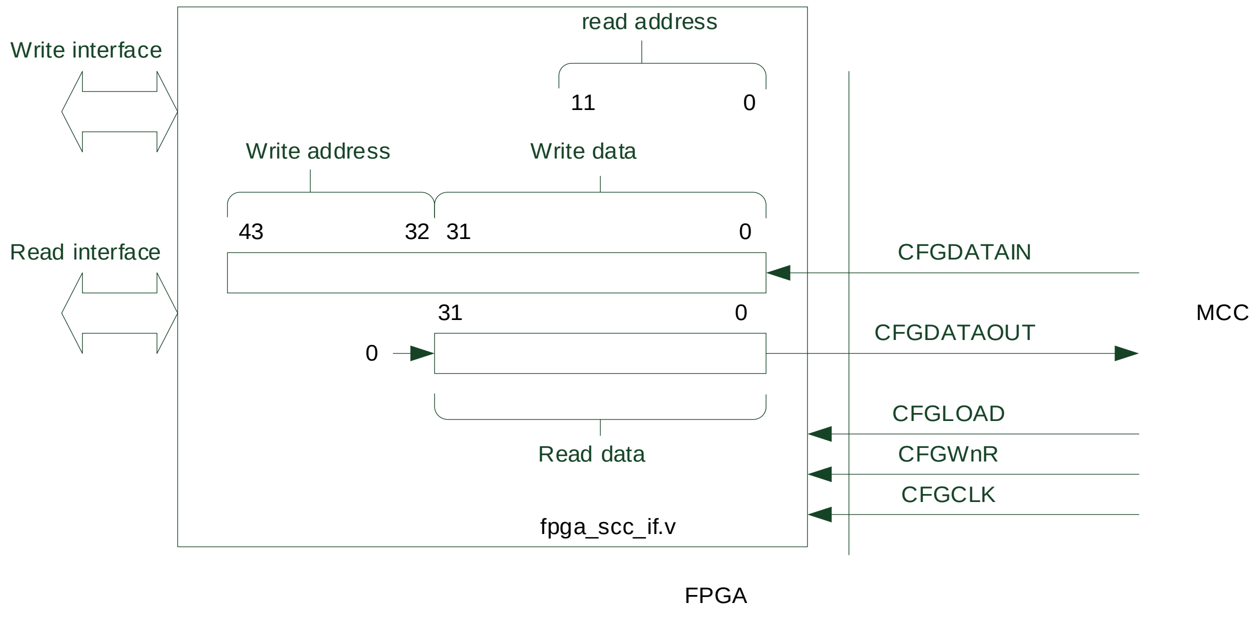


Figure 3-1 : Diagram of the SCC Interface

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

The following table describes the SCC Registers:

Address of all registers are word-aligned.

Address	Name	Information
0x000	CFG_REG0	Bits[31:1] Reserved Bit[0]: HOST CPU AA64nAA32 Select bit (Aarch32 = 0x0, Aarch64 = 0x1)
0x004	CFG_REG1	32-bit DATA [RW]
0x008	CFG_REG2	Bits[31:1] Reserved Bit[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 Bit[0]: CPUWAIT (Active high, resets to high)
0x01C	CFG_REG7	Bits[31:1] Reserved Bit[0] SE Cortex-M0+ ROM (RAM) Enable Write (Enable write = 0x0, disable write = 0x1, resets to low)
0x020	BUILD_LONG	Bits[31:20] Reserved Bits[19:0] Build Date (YYMMDD)
0x024 – 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 (decimal) 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

Address	Name	Information
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: 0x5 =AXI Bits[15:12] Reserved Bits[11:4] Primary part number (in BCD): 0x550 Bits[3:0] Reserved

Table 3-7 : SCC Register Memory Map

3.13 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 3-8 : System Control and I/O Memory Map

Note

All counters are driven from FPGA generated clock MAINCLK.

4 Clock Architecture

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

4.1 Clocks

4.1.1 FPGA External Clocks

The following clocks are inputs to the FPGA.

Clock	Input Pin	Frequency	Note
REFCLK24MHZ	OSCCLK[0]	24MHz	24MHz reference
MAINCLK	OSCCLK[1]	50MHz	Programmable oscillator
SE QSPI PHY	OSCCLK[2]	50MHz	Programmable oscillator
DBGCLK	CS_TCK	Set by debugger	JTAG input
CFGCLK	CFG_CLK	Set by MCC (5MHz)	SCC register clock from MCC
DDR4_REF_CLK	c0_sys_clk_p/n	125MHz	Differential input clock to DDR4 controller
SMBM_CLK	SMBM_CLK	Set by MCC (25MHz)	SMB clock from MCC

Table 4-1 : FPGA External Source Clocks

4.1.2 FPGA Generated Internal Clocks

The following clocks are generated internally from the source clocks.

Clock	Source	Frequency	Note
CLK32KHZ	REFCLK24MHZ	32,768kHz	

Table 4-2 : FPGA Generated Internal Clocks

4.1.3 SSE-710 Clocks

Clock	Source	Frequency	Note
REFCLK	MAINCLK	50MHz	
SECENCREFCLK	MAINCLK	50MHz	
SYSPLL	MAINCLK	50MHz	
CPUPLL	MAINCLK	50MHz	
EXTSYSF0CLK	MAINCLK	50MHz	
UARTCLK	REFCLK24MHZ	24MHz	
SWCLKTCK	CS_TCK	Set by debugger	
TRACECLKIN	MAINCLK	50MHz	
S32KCLK	REFCLK24MHZ	32,768kHz	

Table 4-3 : FPGA Generated SSE-710 Internal Clocks

Note

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

4.1.4 EXTSYS0 Clocks

The following table lists the clocks of EXTSYS0.

Clock	Source	Frequency	Note
EXTSYS0DBGCLKS	MAINCLK	50MHz	
EXTSYS0DBGCLKM	MAINCLK	50MHz	
EXTSYS0MHUCLK	MAINCLK	50MHz	
EXTSYS0ATCLK	MAINCLK	50MHz	
EXTSYS0CTICLK	MAINCLK	50MHz	
EXTSYS0ACLK	MAINCLK	50MHz	

Table 4-4 : FPGA Generated EXTSYS0 Internal Clocks

Note

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

5 Reset Architecture

5.1 Resets

5.1.1 FPGA Resets

The following table shows the top-level resets signals and their direction to the FPGA.

Please refer to MPS3 reference manual for MCC reset control.

Input Source Reset	Note	SSE-710 Example Design Destination Reset	Reset control
CB_nPOR	Power-on Reset Reset signal to all FPGA Subsystem	Not connected	Embedded in MCC firmware, refer to Figure 5-2
CB_nRST	System Reset Reset signal to SSE-710 Subsystem	PORESETn	
IOFPGA_nRST	Auxiliary reset signal Connected to Audio clock generator	Not connected	
CFG_nRST	SCC Register block Reset	Not connected	
CS_nSRST	Warm SSE-710 Subsystem reset	nSRST	Debugger, please refer to debugger user manual
CS_nTRST	JTAG Reset	nTRST	

Output Source Reset	Note
SH_nRST	Shield reset output connected directly to CB_nPOR
SMBF_nRST	Ethernet/USB PHY Reset signal Synchronized with MAINCLK CB_nPOR signal.
AUD_nRST	Audio codec Reset Managed by I2S controller register

Table 5-1 : FPGA Resets List

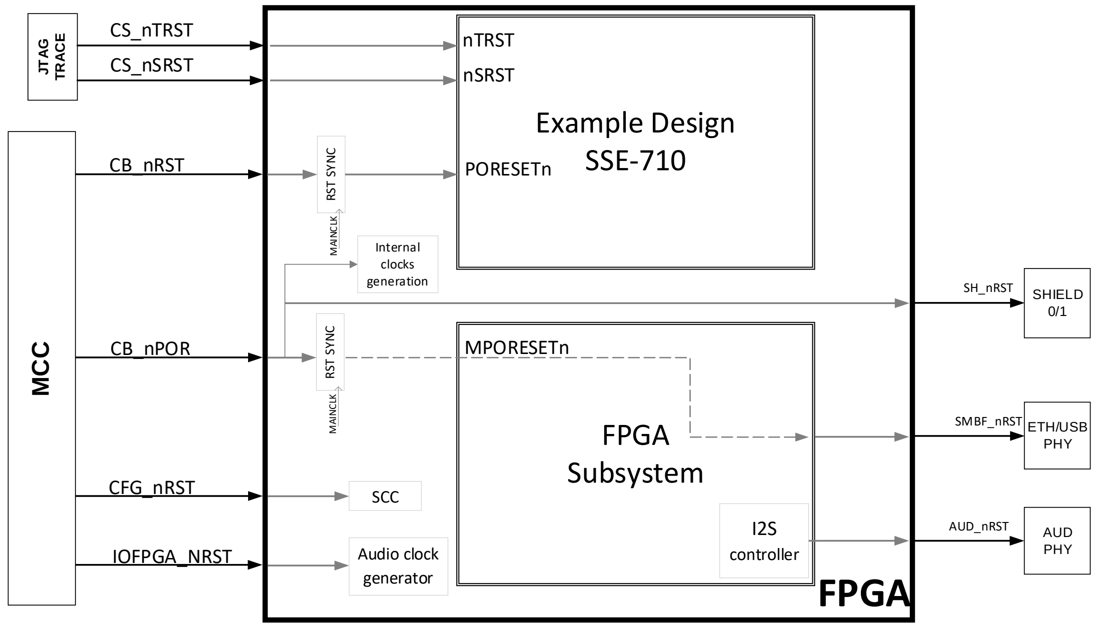


Figure 5-1 : FPGA Resets

The following timing diagram shows a typical reset de-assertion sequence after power on.

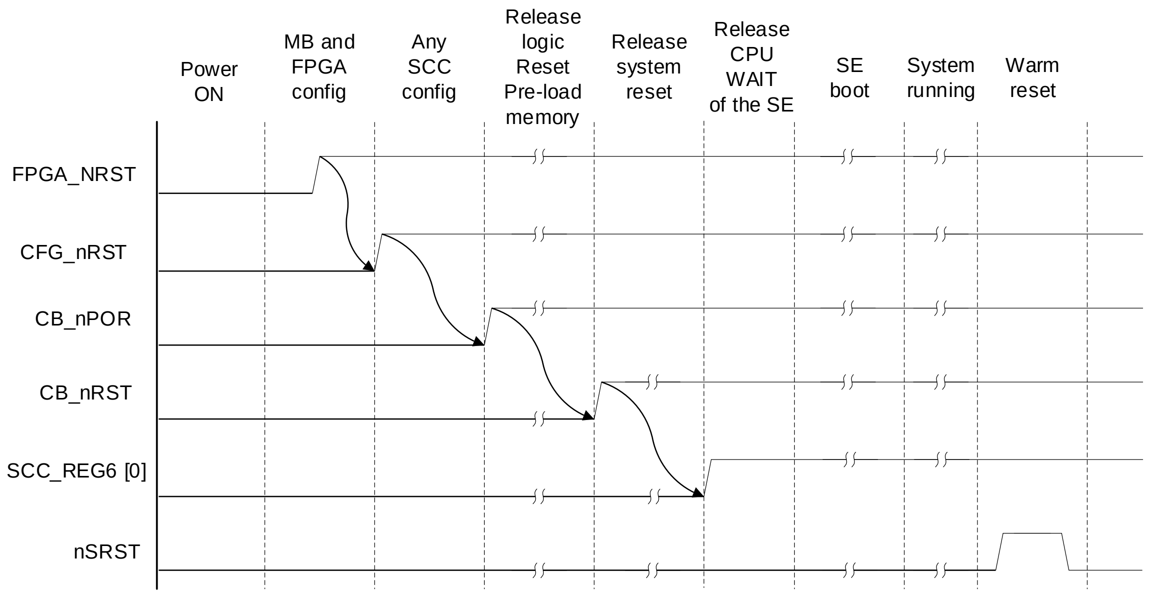


Figure 5-2 : Reset Timing

Note

SECPUWAIT is controlled by SCC_REG[6], please refer to paragraph 3.12

6 ZIP Bundle Description

6.1 Overall Structure

The accompanying .zip bundle contains:

- AN550 SMM Document.
- An example Development Studio 2022.2 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

AN550 SMM Document 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 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:

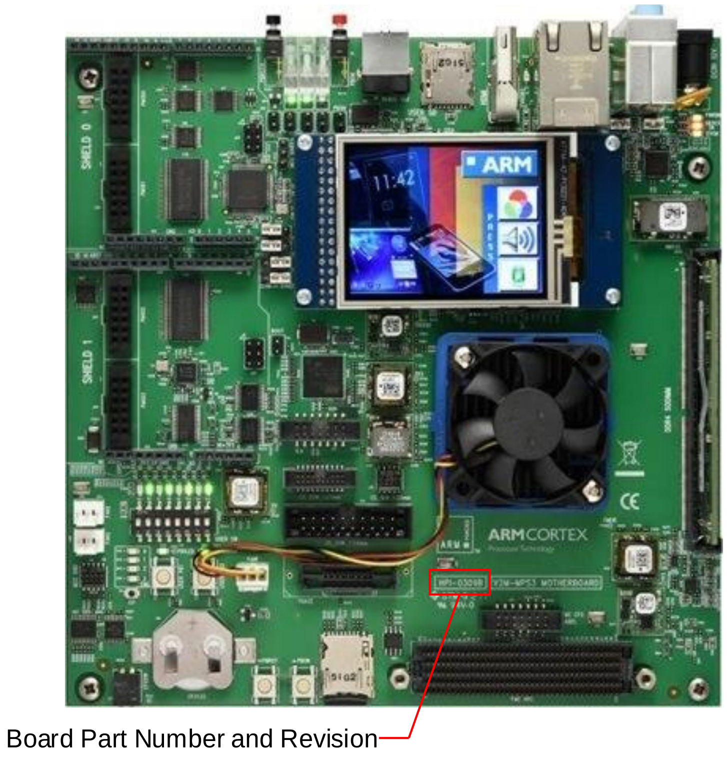


Figure 7-1 : MPS3 Board Revision Identifier

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:

- 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
|   |   |-- HBI0309B
|   |   |   |-- AN550
|   |   |   |   |-- AN550_v2.bit
|   |   |   |   |-- an550_v2.txt
|   |   |   |   `-- images.txt
|   |   |   |-- board.txt
|   |   |   `-- mbb_v210.ebf
|   |   `-- HBI0309C
|   |       |-- AN550
|   |       |   |-- AN550_v2.bit
|   |       |   |-- an550_v2.txt
|   |       |   `-- images.txt
|   |       |-- board.txt
|   |       `-- mbb_v210.ebf
|   |-- SOFTWARE
|   |   |-- ES0.bin
|   |   |-- SE.bin
|   |   `-- an550_st.axf
|   `-- config.txt
|-- Docs
|   `-- DAI0550D_arm_corstone_1000_for_mps3.pdf
|-- Licence.pdf
|-- Software
|   `-- Selftest
|       |-- AACI
|       |   |-- AACI_I2C_MPS3.c
|       |   |-- AACI_I2C_MPS3.h
|       |   |-- AACI_I2S_MPS3.c
|       |   |-- AACI_I2S_MPS3.h
|       |   |-- apaaci.c
|       |   `-- apaaci.h
|       |-- Ethernet
|       |   |-- ETH_MPS3.c
|       |   |-- ETH_MPS3.h
|       |   |-- aplan.c
|       |   `-- aplan.h
|       |-- USB
|       |   |-- apusb.c
|       |   `-- apusb.h
|       |-- an550_st.axf
|       |-- an550_st.bin
|       |-- .cproject
|       |-- .project
|       |-- 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
|       |-- apotp
|       |   |-- apotp.c
|       |   `-- apotp.h
|       |-- 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
|       |       |   `-- system.h
|       |       `-- Source
|       |           `-- CMSDK_IoT
|       |               |-- startup_CMSDK_IoT.s
|       |               `-- system_CMSDK_IoT.c
|       |-- move.bat
|       `-- selftest_IoT.scat
`-- revision_history.txt
```

7 Software and Debug

7.1 Overview

It is strongly recommended to use Development Studio 2022.2 Silver edition which is the minimum edition needed to support all the processor cores in this implementation and the version supported by AN550 SMM.

An example software project that can be built using Arm Development Studio (targeting the SE Cortex-M0+) is provided 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+, offers the following tests:

- Internal FPGA BRAM (CVM)
- External DDR4 (OCVM)
- External QSPI flash (XNVM)
- External SE QSPI flash (EXPMST0)
- Ethernet
- Audio
- USB

7.2 Example Software Project

Zip 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 AN550 SMM 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 Cortex-A35	Cortex-M0+ Cortex-M3 Cortex-A35	Cortex-M3 Cortex-A35	▨
20 pin IDC	Cortex-M0+ Cortex-M3 Cortex-A35	Cortex-M0+ Cortex-M3 Cortex-A35	▨	▨
Mictor 38	Cortex-M0+ Cortex-M3 Cortex-A35	Cortex-M0+ Cortex-M3 Cortex-A35	Cortex-M3 Cortex-A35	Cortex-M3 Cortex-A35

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 Connection

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

1. Ensure the DSTREAM debugger is:
   a. Powered, and connected to the host running the Development Studio software version 2022.2 Silver Edition..
   b. Connected to the MPS3 using the 20-pin Cortex / 20-pin IDC / Mictor 38 connector on the MPS3 as shown below:

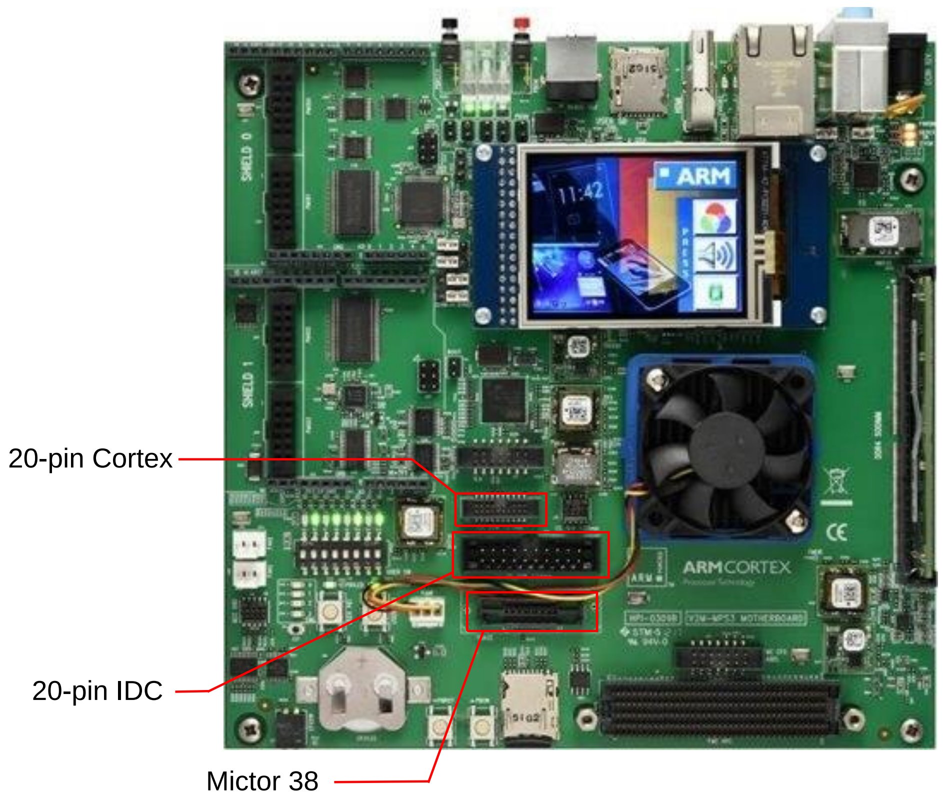
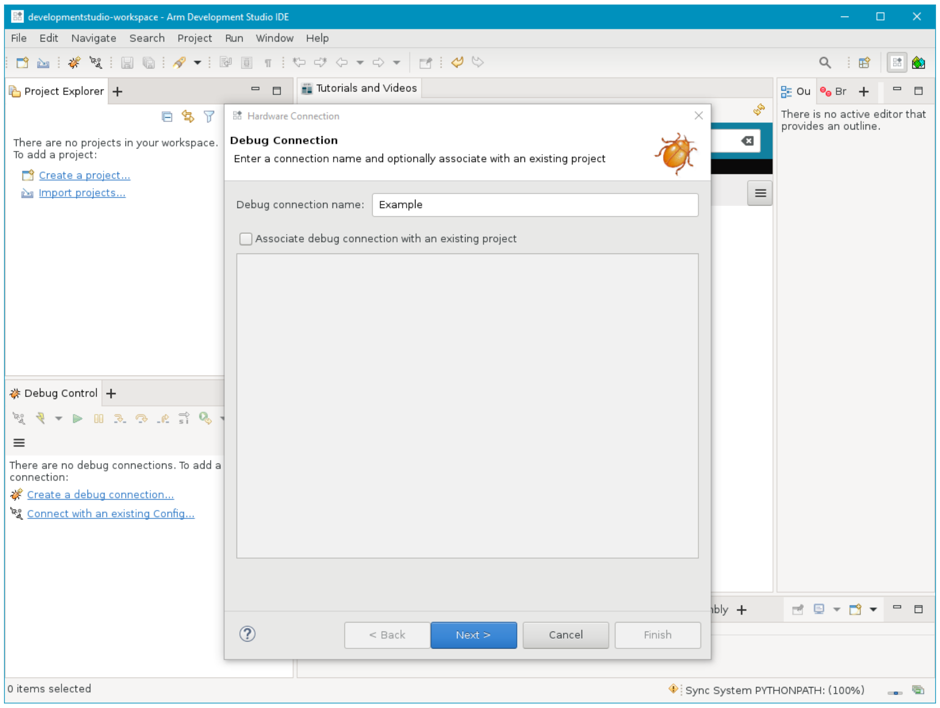


Figure 7-1 : MPS3 Board Debug Connector Locations

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

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


developmentstudio-workspace - Arm Development Studio IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Tutorials and Videos
Ou
Br
Debug Control
< Back
Next >
Cancel
Finish

Hardware Connection

Debug Connection

Enter a connection name and optionally associate with an existing project

Debug connection name: Example

☐ Associate debug connection with an existing project

There are no projects in your workspace. To add a project:
Create a project...
Import projects...

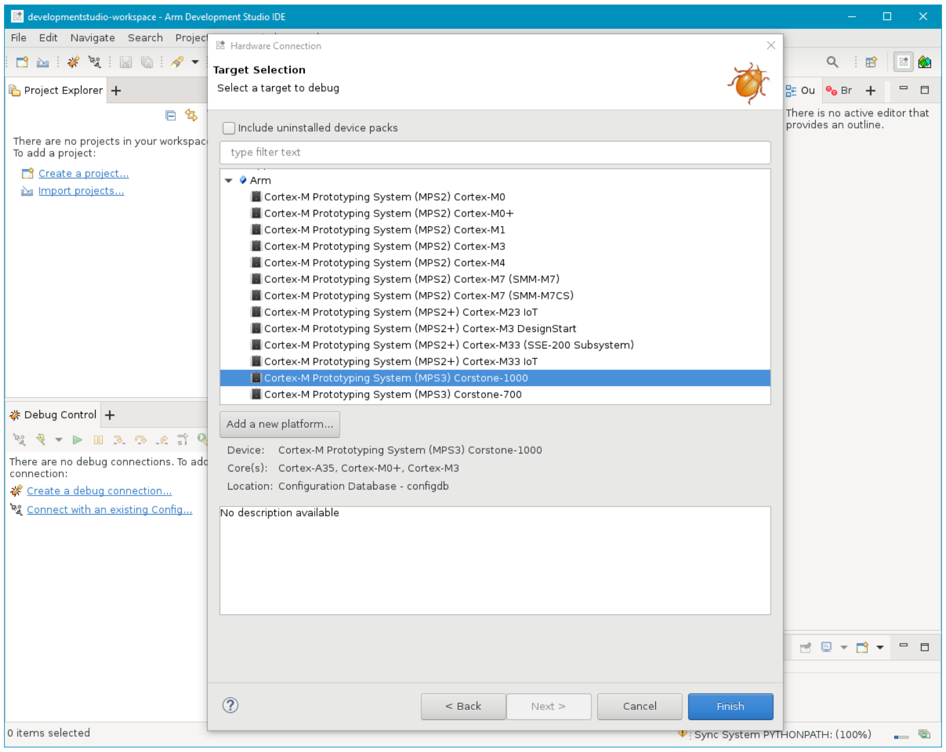
There are no debug connections. To add a connection:
Create a debug connection...
Connect with an existing Config...

There is no active editor that provides an outline.

0 items selected

Sync System PYTHONPATH: (100%)

4. Add Debug connection name and click **Next**.
5. In the **Target Selection** dialog, under **Arm** folder choose **Cortex-M Prototyping System (MPS3) Corestone-1000** (which highlights when selected) and click **Finish**.


developmentstudio-workspace - Arm Development Studio IDE
File Edit Navigate Search Project
Project Explorer
Debug Control
Ou
Br
Add a new platform...
< Back
Next >
Cancel
Finish

Hardware Connection

Target Selection

Select a target to debug

☐ Include uninstalled device packs

type filter text

Arm
- Cortex-M Prototyping System (MPS2) Cortex-M0
- Cortex-M Prototyping System (MPS2) Cortex-M0+
- Cortex-M Prototyping System (MPS2) Cortex-M1
- Cortex-M Prototyping System (MPS2) Cortex-M3
- Cortex-M Prototyping System (MPS2) Cortex-M4
- Cortex-M Prototyping System (MPS2) Cortex-M7 (SMM-M7)
- Cortex-M Prototyping System (MPS2) Cortex-M7 (SMM-M7CS)
- Cortex-M Prototyping System (MPS2+) Cortex-M23 IoT
- Cortex-M Prototyping System (MPS2+) Cortex-M3 DesignStart
- Cortex-M Prototyping System (MPS2+) Cortex-M33 (SSE-200 Subsystem)
- Cortex-M Prototyping System (MPS2+) Cortex-M33 IoT
- Cortex-M Prototyping System (MPS3) Corstone-1000
- Cortex-M Prototyping System (MPS3) Corstone-700

Device:	Cortex-M Prototyping System (MPS3) Corstone-1000
Core(s):	Cortex-A35, Cortex-M0+, Cortex-M3
Location:	Configuration Database - configdb

No description available

There are no projects in your workspace. To add a project:
Create a project...
Import projects...

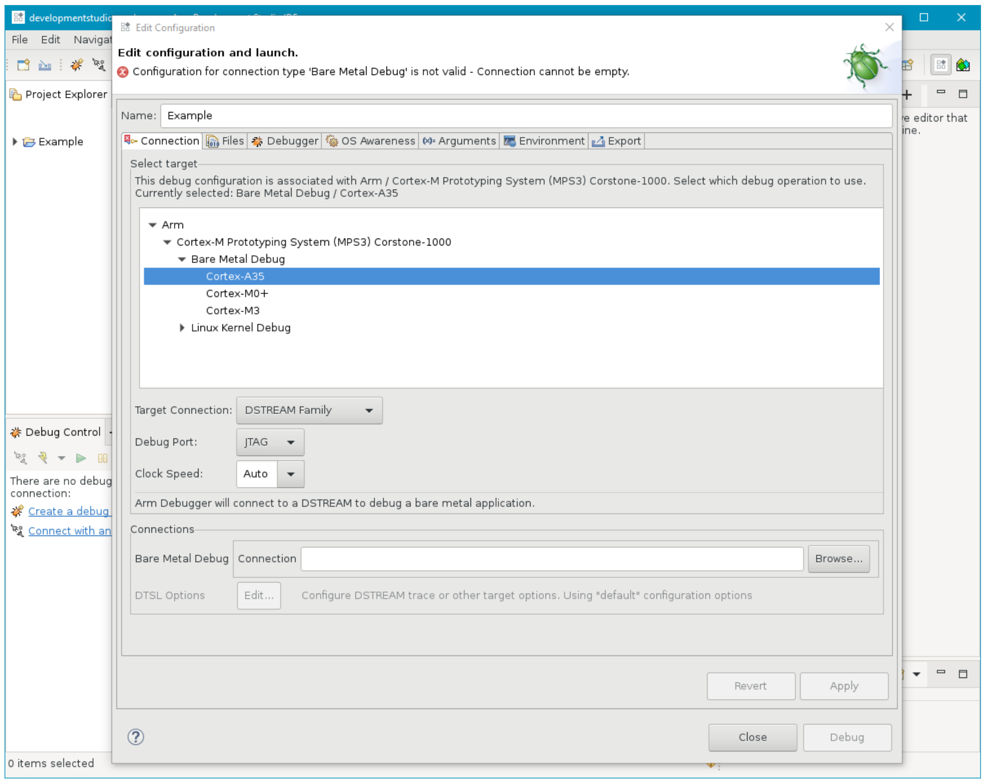
There are no debug connections. To add a connection:
Create a debug connection...
Connect with an existing Config...

There is no active editor that provides an outline.

0 items selected

Sync System PYTHONPATH: (100%)

6. Select your connection type and connection address for your target debugger and click **Debug** as shown below.



Edit configuration and launch.

Configuration for connection type 'Bare Metal Debug' is not valid - Connection cannot be empty.

Name: Example

Select target

This debug configuration is associated with Arm / Cortex-M Prototyping System (MPS3) Corstone-1000. Select which debug operation to use.
Currently selected: Bare Metal Debug / Cortex-A35

- Arm
  - Cortex-M Prototyping System (MPS3) Corstone-1000
    - Bare Metal Debug
      - Cortex-A35
      - Cortex-M0+
      - Cortex-M3
    - Linux Kernel Debug

Target Connection: DSTREAM Family

Debug Port: JTAG

Clock Speed: Auto

Arm Debugger will connect to a DSTREAM to debug a bare metal application.

Connections

Bare Metal Debug Connection

DTSL Options Edit... Configure DSTREAM trace or other target options. Using "default" configuration options

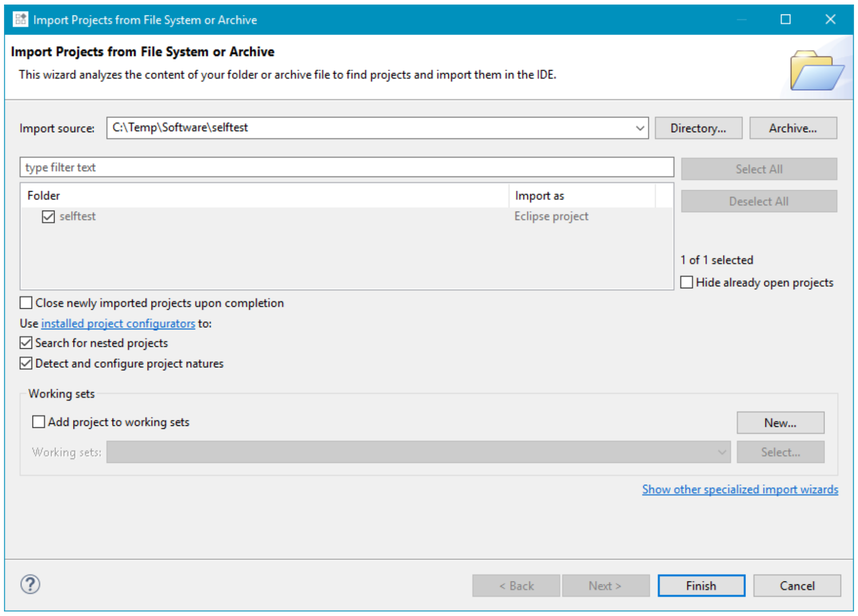
7.4 Building the Software

Requirements:

- The software directory from the supplied bundle `SOFTWARE/selftest`.
- Proficiency with Arm Development Studio, version 2022.2 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.



Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Temp\Software\selftest

Folder	Import as
☑ selftest	Eclipse project

1 of 1 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

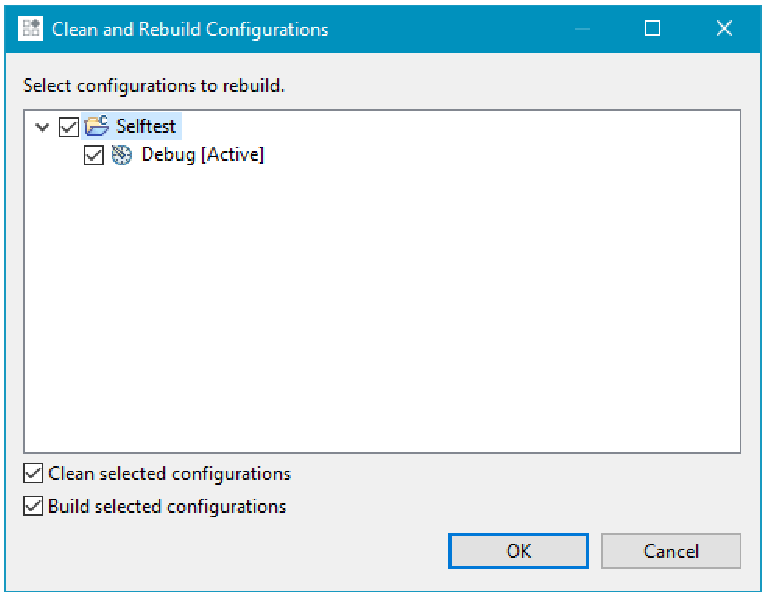
Working sets

☐ Add project to working sets

Working sets:

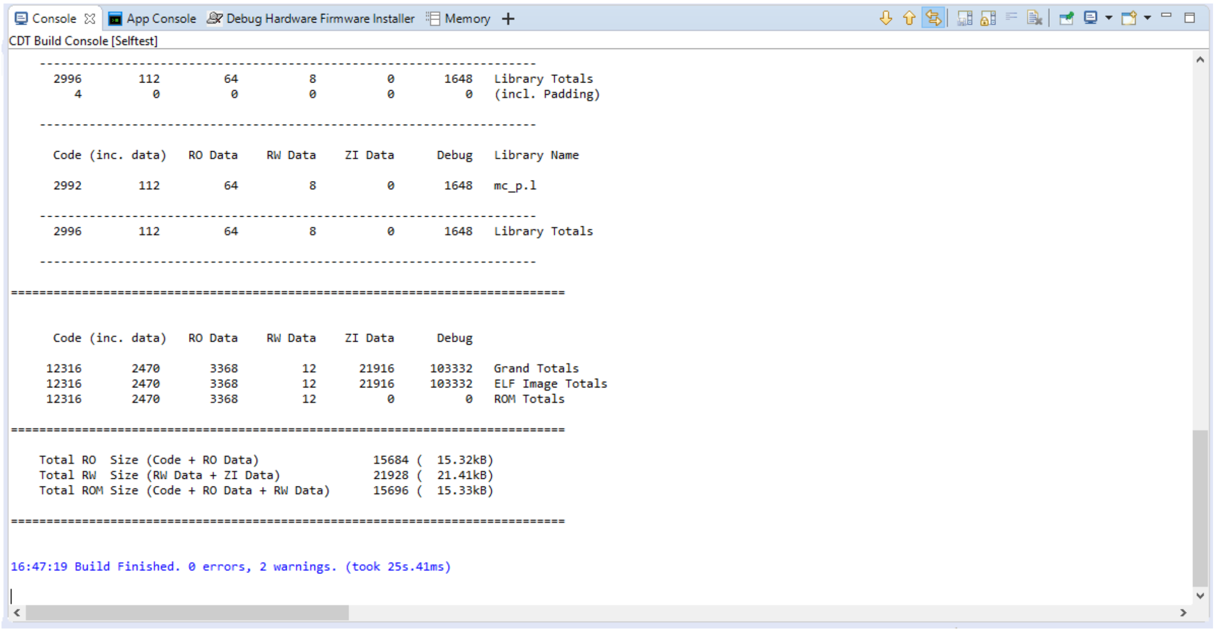
3. Click **Finish**.
4. The project is now imported into the Workspace and is viewable in the Project Explorer within Development Studio as Selftest.
5. Right-click on the imported project Selftest in the Project Explorer and select **`Build Configurations > Build Selected`**.

6. In the Clean and Rebuild Configurations window make sure the Debug [Active] configuration is selected as shown below:


Clean and Rebuild Configurations
Select configurations to rebuild.
Selftest
Debug [Active]
Clean selected configurations
Build selected configurations
OK
Cancel

7. Click **OK**.

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



```
CDT Build Console [Selftest]
    ------------------------------------------------------------------
      2996        112         64          8          0       1648   Library Totals
         4          0          0          0          0          0   (incl. Padding)

    ------------------------------------------------------------------

      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Library Name

      2992        112         64          8          0       1648   mc_p.l

    ------------------------------------------------------------------
      2996        112         64          8          0       1648   Library Totals

    ------------------------------------------------------------------

==============================================================================


      Code (inc. data)   RO Data    RW Data    ZI Data      Debug

     12316       2470       3368         12      21916     103332   Grand Totals
     12316       2470       3368         12      21916     103332   ELF Image Totals
     12316       2470       3368         12          0          0   ROM Totals

==============================================================================

    Total RO  Size (Code + RO Data)                15684 (  15.32kB)
    Total RW  Size (RW Data + ZI Data)             21928 (  21.41kB)
    Total ROM Size (Code + RO Data + RW Data)      15696 (  15.33kB)

==============================================================================


16:47:19 Build Finished. 0 errors, 2 warnings. (took 25s.41ms)
```

9. The project has now built and there is a newly built **.axf** file in the **SOFTWARE/selftest/Debug** directory, titled **an550_st.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:

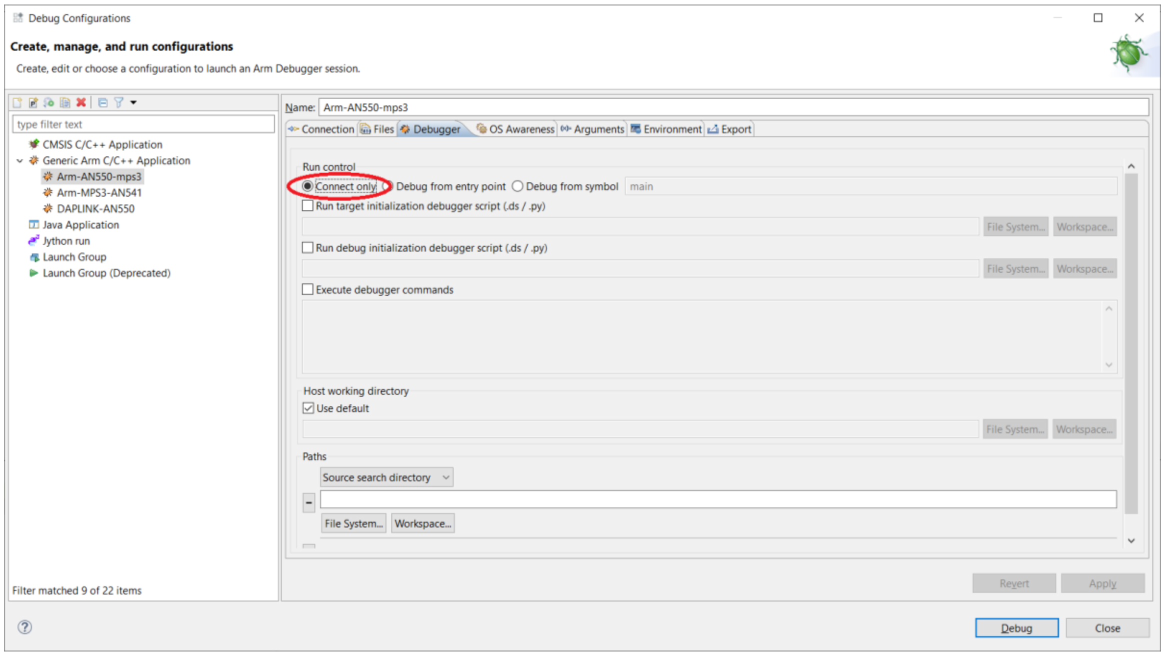
- The SE Cortex-M0+
- The HOST CPU Cortex-A35
- The EXTSYS0 Cortex-M3

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 on **Figure 7-1 : 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 Connect only under Run control.

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



Debug Configurations

Create, manage, and run configurations

Create, edit or choose a configuration to launch an Arm Debugger session.

type filter text

- CMSIS C/C++ Application
- Generic Arm C/C++ Application
  - Arm-AN550-mps3
  - Arm-MPS3-AN541
  - DAPLINK-AN550
- Java Application
- Jython run
- Launch Group
- Launch Group (Deprecated)

Filter matched 9 of 22 items

Name: Arm-AN550-mps3

Connection | Files | Debugger | OS Awareness | Arguments | Environment | Export

Run control

[✓ Connect only] Debug from entry point Debug from symbol main

☐ Run target initialization debugger script (.ds / .py)

File System... Workspace...

☐ Run debug initialization debugger script (.ds / .py)

File System... Workspace...

☐ Execute debugger commands

Host working directory

☑ Use default

File System... Workspace...

Paths

Source search directory

File System... Workspace...

Revert Apply

Debug Close

3. Click **Debug**.

The previous debug configuration can be customized to connect with any of the three cores.

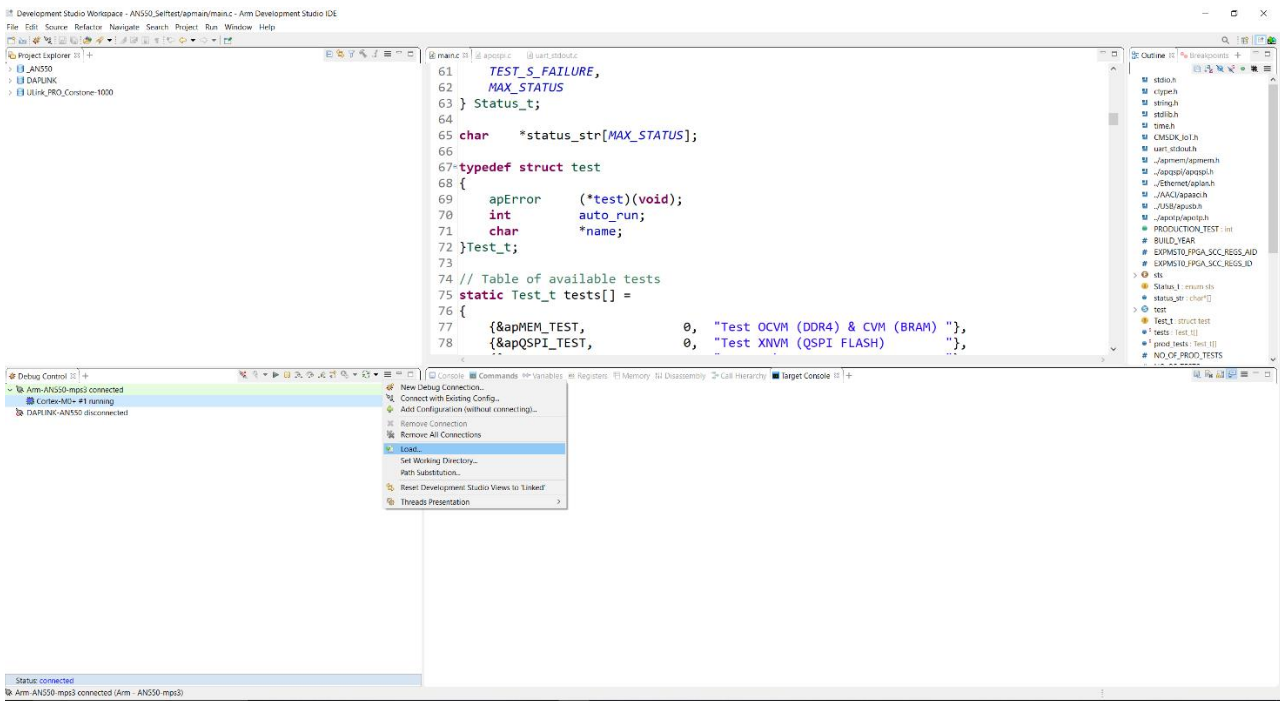
It is also possible to align Arm Development Studio and the debugger to step into the selftest project code having an550_st.axf running on the board. To do so it's necessary to rebuild the selftest project in Arm Development Studio and drag and drop an550_st.axf binary file into the SD card under SOFTWARE/.

Note

debugger loads symbols only and is not able to download flash. The axf file must be loaded using the SD card.

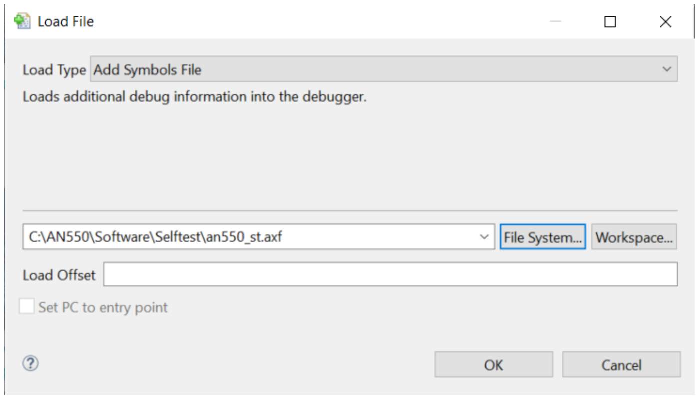
---

1. In the Debug Control tab select "**Load...**" from the hidden three horizontal lines menu



```
    TEST_S_FAILURE,
    MAX_STATUS
} Status_t;

char    *status_str[MAX_STATUS];

typedef struct test
{
    apError     (*test)(void);
    int         auto_run;
    char        *name;
}Test_t;

// Table of available tests
static Test_t tests[] =
{
    {&apMEM_TEST,              0,  "Test OCVM (DDR4) & CVM (BRAM) "},
    {&apQSPI_TEST,             0,  "Test XNVM (QSPI FLASH)        "},
```

2. Now select the axf file from your hard drive **File System**'s path and click OK.



Load Type: Add Symbols File

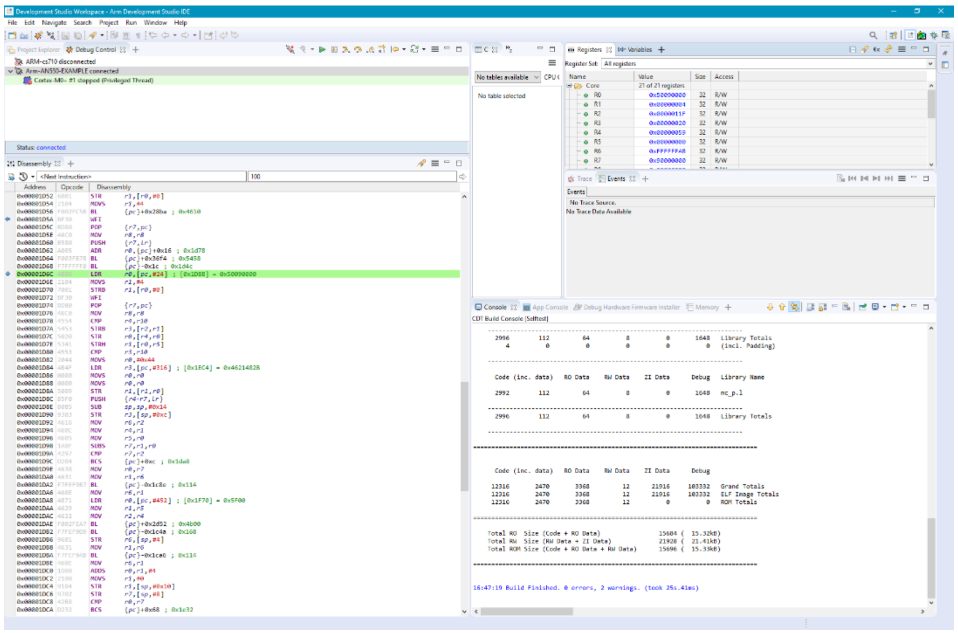
Loads additional debug information into the debugger.

C:\AN550\Software\Selftest\an550_st.axf

Load Offset:

Set PC to entry point

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



ARM-cs710 disconnected
Arm-AN550-EXAMPLE connected
Cortex-M0+ #1 stopped (Privileged Thread)

Status: connected

<Next Instruction>   100

```
Address     Opcode    Disassembly
0x00001D52  6001      STR     r1,[r0,#0]
0x00001D54  2104      MOVS    r1,#4
0x00001D56  F002FC58  BL      {pc}+0x28ba ; 0x4610
0x00001D5A  BF30      WFI
0x00001D5C  BD80      POP     {r7,pc}
0x00001D5E  46C0      MOV     r8,r8
0x00001D60  B580      PUSH    {r7,lr}
0x00001D62  A005      ADR     r0,{pc}+0x16 ; 0x1d78
0x00001D64  F003FB78  BL      {pc}+0x36f4 ; 0x5458
0x00001D68  F7FFFFF0  BL      {pc}-0x1c ; 0x1d4c
0x00001D6C  4806      LDR     r0,[pc,#24] ; [0x1D88] = 0x50090000
0x00001D6E  2104      MOVS    r1,#4
0x00001D70  7001      STRB    r1,[r0,#0]
0x00001D72  BF30      WFI
0x00001D74  BD80      POP     {r7,pc}
0x00001D76  46C0      MOV     r8,r8
0x00001D78  4554      CMP     r4,r10
0x00001D7A  5453      STRB    r3,[r2,r1]
0x00001D7C  5020      STR     r0,[r4,r0]
0x00001D7E  5341      STRH    r1,[r0,r5]
0x00001D80  4553      CMP     r3,r10
0x00001D82  2044      MOVS    r0,#0x44
0x00001D84  484F      LDR     r3,[pc,#316] ; [0x1EC4] = 0x4621482B
0x00001D86  0000      MOVS    r0,r0
0x00001D88  0000      MOVS    r0,r0
0x00001D8A  5009      STR     r1,[r1,r0]
0x00001D8C  B5F0      PUSH    {r4-r7,lr}
0x00001D8E  B085      SUB     sp,sp,#0x14
0x00001D90  9303      STR     r3,[sp,#0xc]
0x00001D92  4616      MOV     r6,r2
0x00001D94  468C      MOV     r4,r1
0x00001D96  4605      MOV     r5,r0
0x00001D98  1A0F      SUBS    r7,r1,r0
0x00001D9A  4297      CMP     r7,r2
0x00001D9C  D204      BCS     {pc}+0xc ; 0x1da8
0x00001D9E  4638      MOV     r0,r7
0x00001DA0  4631      MOV     r1,r6
0x00001DA2  F7FEF9B7  BL      {pc}-0x1c8e ; 0x114
0x00001DA6  460E      MOV     r6,r1
0x00001DA8  4871      LDR     r0,[pc,#452] ; [0x1F70] = 0x5F00
0x00001DAA  4629      MOV     r1,r5
0x00001DAC  4622      MOV     r2,r4
0x00001DAE  F002FEA7  BL      {pc}+0x2d52 ; 0x4b00
0x00001DB2  F7FEF9D9  BL      {pc}-0x1c4a ; 0x168
0x00001DB6  9601      STR     r6,[sp,#4]
0x00001DB8  4631      MOV     r1,r6
0x00001DBA  F7FEF9AB  BL      {pc}-0x1ca6 ; 0x114
0x00001DBE  460E      MOV     r6,r1
0x00001DC0  1D08      ADDS    r0,r1,#4
0x00001DC2  2100      MOVS    r1,#0
0x00001DC4  9104      STR     r1,[sp,#0x10]
0x00001DC6  9702      STR     r7,[sp,#8]
0x00001DC8  4288      CMP     r0,r7
0x00001DCA  D232      BCS     {pc}+0x68 ; 0x1e32
```

No tables available

No table selected

Register Set: All registers

Name	Value	Size	Access
Core	21 of 21 registers		
R0	0x50090000	32	R/W
R1	0x00000004	32	R/W
R2	0x0000011F	32	R/W
R3	0x00000020	32	R/W
R4	0x00000059	32	R/W
R5	0x00000000	32	R/W
R6	0xFFFFFFA8	32	R/W
R7	0x90000000	32	R/W

Events

No Trace Source.

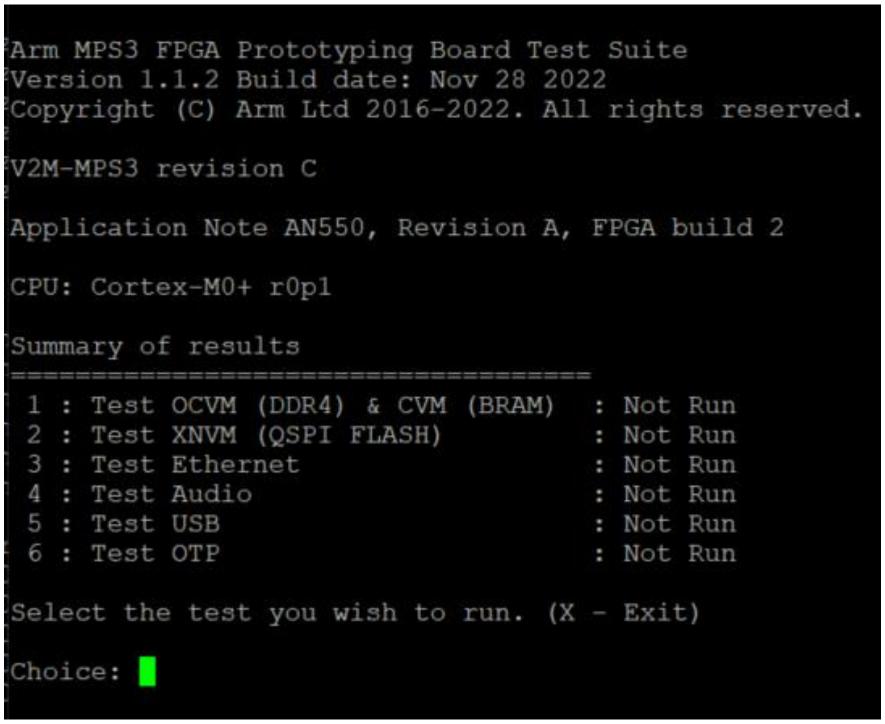
No Trace Data Available

CDT Build Console [Selftest]

```
    ----------------------------------------------------------------------
      2996        112         64          8          0       1648   Library Totals
         4          0          0          0          0          0   (incl. Padding)

    ----------------------------------------------------------------------

      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Library Name

      2992        112         64          8          0       1648   mc_p.l

    ----------------------------------------------------------------------
      2996        112         64          8          0       1648   Library Totals

    ----------------------------------------------------------------------

==============================================================================


      Code (inc. data)   RO Data    RW Data    ZI Data      Debug

     12316       2470       3368         12      21916     103332   Grand Totals
     12316       2470       3368         12      21916     103332   ELF Image Totals
     12316       2470       3368         12          0          0   ROM Totals

==============================================================================

    Total RO  Size (Code + RO Data)                15684 (  15.32kB)
    Total RW  Size (RW Data + ZI Data)             21928 (  21.41kB)
    Total ROM Size (Code + RO Data + RW Data)      15696 (  15.33kB)

==============================================================================


16:47:19 Build Finished. 0 errors, 2 warnings. (took 25s.41ms)
```

4. Program execution at this stage can be either single-stepped or set to Run ▶.

5. At this point, to launch the selftest program press the red PBON button on mps3 board and on SE UART, the Cortex-M0+ will boot into the test code, as shown below:



```
Arm MPS3 FPGA Prototyping Board Test Suite
Version 1.1.2 Build date: Nov 28 2022
Copyright (C) Arm Ltd 2016-2022. All rights reserved.

V2M-MPS3 revision C

Application Note AN550, Revision A, FPGA build 2

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
 6 : Test OTP                        : Not Run

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

Choice:
```

7.6 Modifying the SSE-710 Boot Register

7.6.1 SSE-710 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 Processor. 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 initialization 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-710 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 MSP3 reboot (see section 8.6.2 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-710 Secure Enclave Technical Reference Manual* for more details.

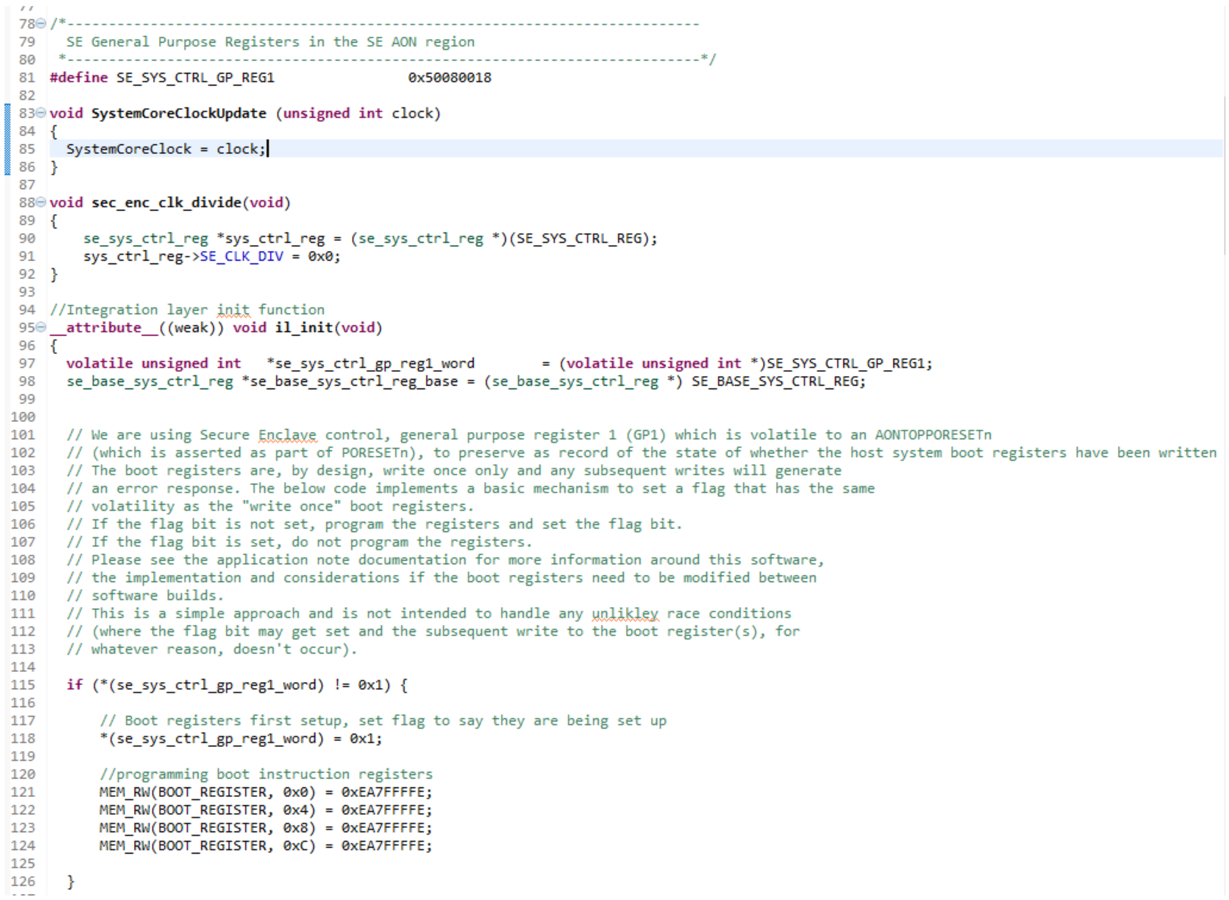
Modifying the Boot Register

If changes are made to source code that require a change to the Boot Register, 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:



```
/*---------------------------------------------------------------------------
  SE General Purpose Registers in the SE AON region
 *----------------------------------------------------------------------------*/
#define SE_SYS_CTRL_GP_REG1                 0x50080018

void SystemCoreClockUpdate (unsigned int clock)
{
  SystemCoreClock = clock;
}

void sec_enc_clk_divide(void)
{
    se_sys_ctrl_reg *sys_ctrl_reg = (se_sys_ctrl_reg *)(SE_SYS_CTRL_REG);
    sys_ctrl_reg->SE_CLK_DIV = 0x0;
}

//Integration layer init function
__attribute__((weak)) void il_init(void)
{
  volatile unsigned int   *se_sys_ctrl_gp_reg1_word        = (volatile unsigned int *)SE_SYS_CTRL_GP_REG1;
  se_base_sys_ctrl_reg *se_base_sys_ctrl_reg_base = (se_base_sys_ctrl_reg *) SE_BASE_SYS_CTRL_REG;


  // We are using Secure Enclave control, general purpose register 1 (GP1) which is volatile to an AONTOPPORESETn
  // (which is asserted as part of PORESETn), to preserve as record of the state of whether the host system boot registers have been written
  // The boot registers are, by design, write once only and any subsequent writes will generate
  // an error response. The below code implements a basic mechanism to set a flag that has the same
  // volatility as the "write once" boot registers.
  // If the flag bit is not set, program the registers and set the flag bit.
  // If the flag bit is set, do not program the registers.
  // Please see the application note documentation for more information around this software,
  // the implementation and considerations if the boot registers need to be modified between
  // software builds.
  // This is a simple approach and is not intended to handle any unlikley race conditions
  // (where the flag bit may get set and the subsequent write to the boot register(s), for
  // whatever reason, doesn't occur).

  if (*(se_sys_ctrl_gp_reg1_word) != 0x1) {

      // Boot registers first setup, set flag to say they are being set up
      *(se_sys_ctrl_gp_reg1_word) = 0x1;

      //programming boot instruction registers
      MEM_RW(BOOT_REGISTER, 0x0) = 0xEA7FFFFE;
      MEM_RW(BOOT_REGISTER, 0x4) = 0xEA7FFFFE;
      MEM_RW(BOOT_REGISTER, 0x8) = 0xEA7FFFFE;
      MEM_RW(BOOT_REGISTER, 0xC) = 0xEA7FFFFE;

  }
```

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 build AXF, from step 3 above, at MPS3 reboot time. See section 8.7.2 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 AN550 on the MPS3 Board

8.1 Pre-Requisites

Before attempting to use the board you must:

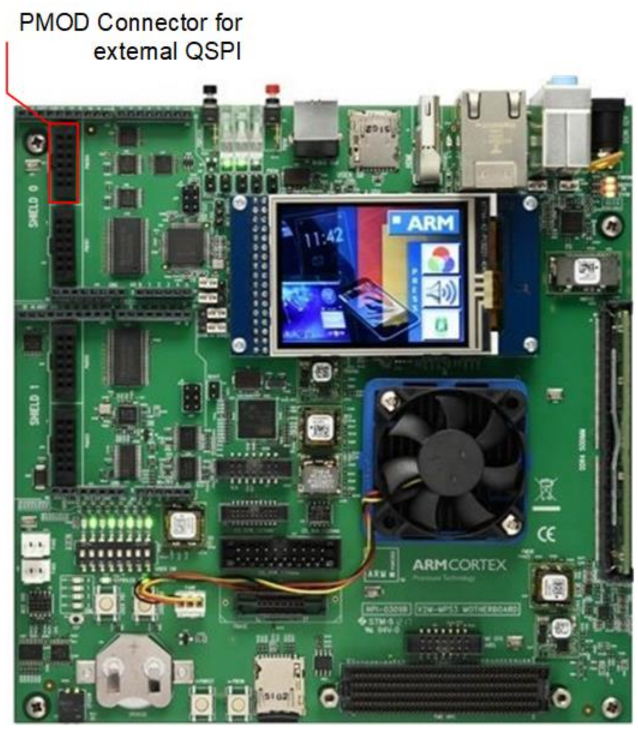
- Read the *Arm® MPS3 FPGA Prototyping Board Technical Reference Manual*. In particular become familiar with the description of the configuration and boot flow.

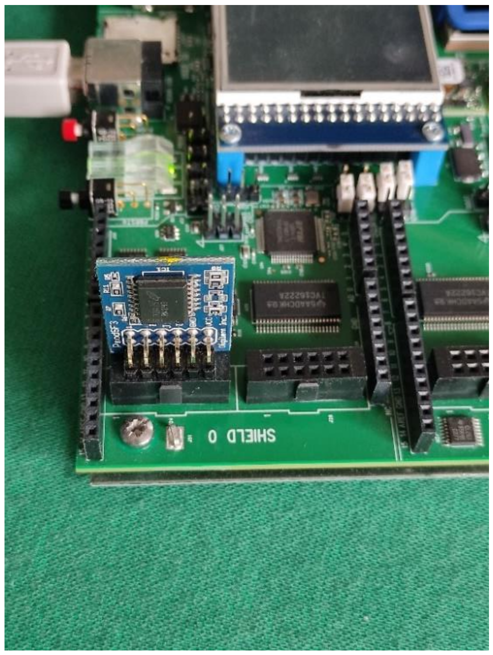
You must be able to:

- 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).
- Power the MPS3 board.
    a. The MPS3 board appears as a mapped drive named V2M-MPS3.
- 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

Before loading the image file on the board ensure the PMOD SF3 QSPI Flash module is plugged into the board correctly into the PMOD0 connector of MPS3 board as it is shown in these pictures.





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/HBI0309{<boardrev>}/board.txt`**
- **`SDCARD/MB/ HBI0309{<boardrev>}/AN550/images.txt`**

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 initially connected to the MCC and outputs verbose configuration and preload information about the status of the MCC. Then this port will be switch to the Host Secure UART1.
- Serial Port 1 is connected to the SE UART.
- Serial Port 2 is connected to the SSE-710 UART0 (Cortex-A35 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.

8.3 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

8.4 MPS3 USB Serial Port Drivers for Windows

For information on installing drivers to support USB serial port on MPS3 see:

https://community.arm.com/oss-platforms/w/docs/589/accessing-mps3-serial-ports-in-windows-10

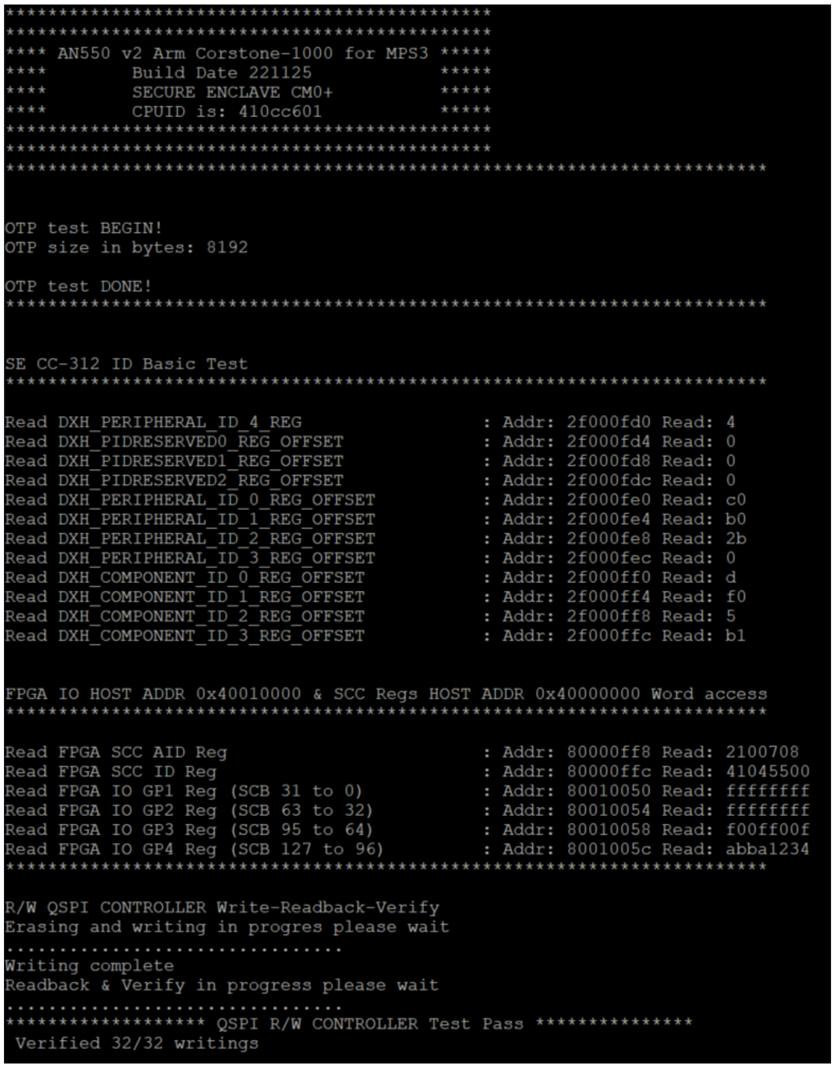
8.5 Running the FPGA image file on the MPS3 Board

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).

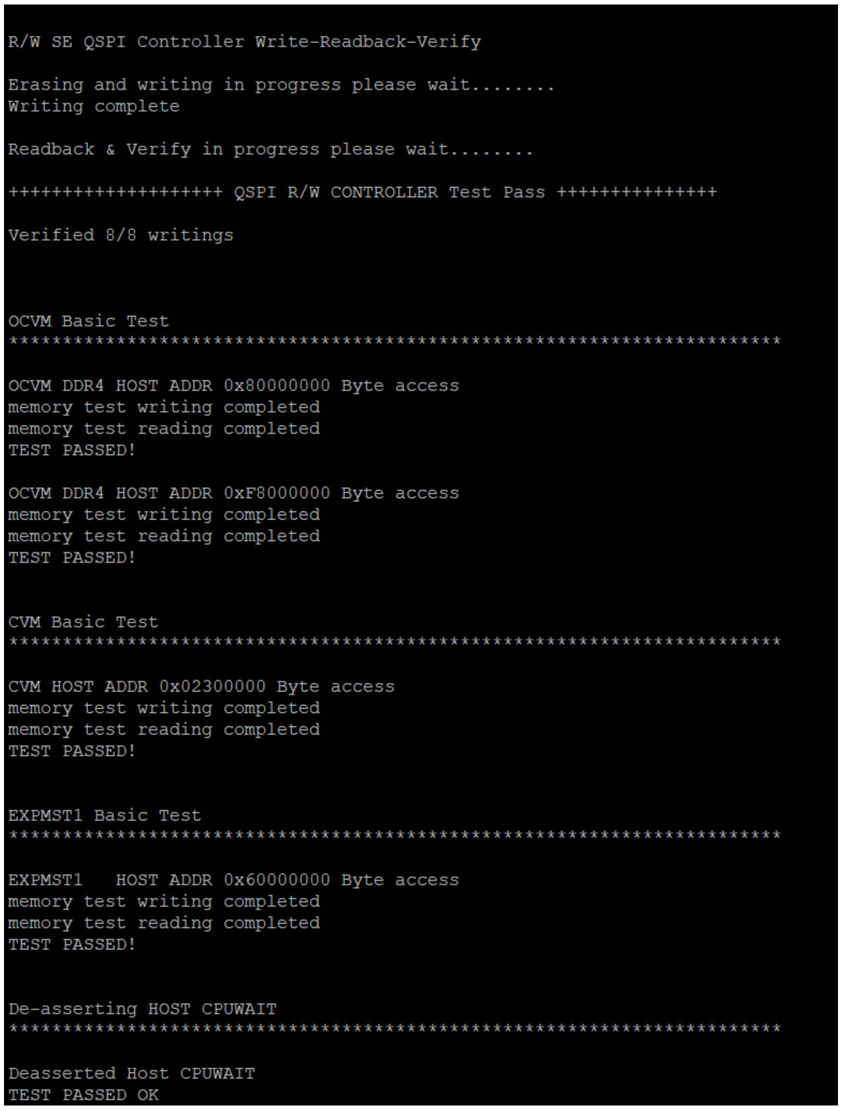
Once booted, the embedded test images in the FPGA run on each CPU, and output is observed on the UARTs.

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.

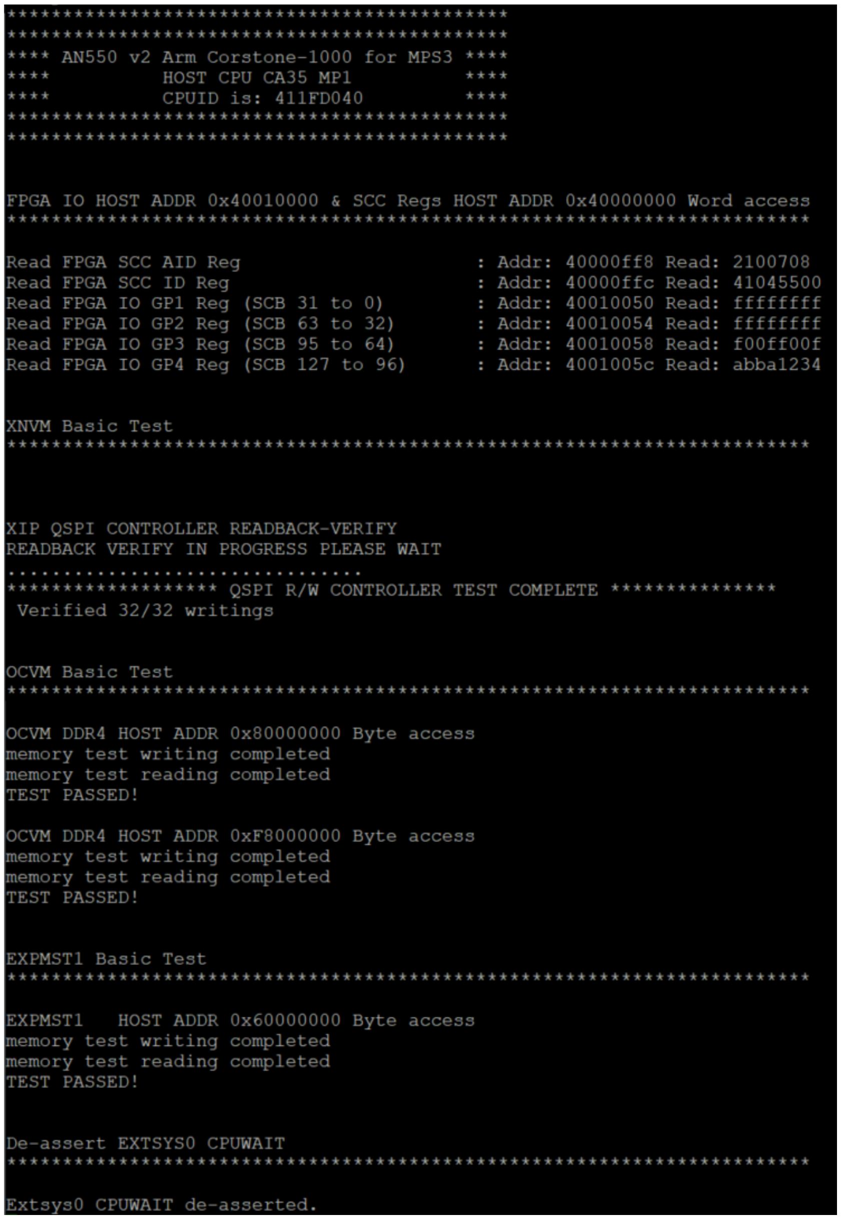


```
***********************************************
***********************************************
**** AN550 v2 Arm Corstone-1000 for MPS3 *****
****         Build Date 221125           *****
****         SECURE ENCLAVE CM0+         *****
****         CPUID is: 410cc601          *****
***********************************************
***********************************************
*************************************************************************


OTP test BEGIN!
OTP size in bytes: 8192

OTP test DONE!
*************************************************************************


SE CC-312 ID Basic Test
*************************************************************************

Read DXH_PERIPHERAL_ID_4_REG                  : Addr: 2f000fd0 Read: 4
Read DXH_PIDRESERVED0_REG_OFFSET              : Addr: 2f000fd4 Read: 0
Read DXH_PIDRESERVED1_REG_OFFSET              : Addr: 2f000fd8 Read: 0
Read DXH_PIDRESERVED2_REG_OFFSET              : Addr: 2f000fdc Read: 0
Read DXH_PERIPHERAL_ID_0_REG_OFFSET           : Addr: 2f000fe0 Read: c0
Read DXH_PERIPHERAL_ID_1_REG_OFFSET           : Addr: 2f000fe4 Read: b0
Read DXH_PERIPHERAL_ID_2_REG_OFFSET           : Addr: 2f000fe8 Read: 2b
Read DXH_PERIPHERAL_ID_3_REG_OFFSET           : Addr: 2f000fec Read: 0
Read DXH_COMPONENT_ID_0_REG_OFFSET            : Addr: 2f000ff0 Read: d
Read DXH_COMPONENT_ID_1_REG_OFFSET            : Addr: 2f000ff4 Read: f0
Read DXH_COMPONENT_ID_2_REG_OFFSET            : Addr: 2f000ff8 Read: 5
Read DXH_COMPONENT_ID_3_REG_OFFSET            : Addr: 2f000ffc Read: b1


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

Read FPGA SCC AID Reg                         : Addr: 80000ff8 Read: 2100708
Read FPGA SCC ID Reg                          : Addr: 80000ffc Read: 41045500
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
*************************************************************************

R/W QSPI CONTROLLER Write-Readback-Verify
Erasing and writing in progres please wait
................................
Writing complete
Readback & Verify in progress please wait
................................
******************* QSPI R/W CONTROLLER Test Pass ***************
 Verified 32/32 writings
```



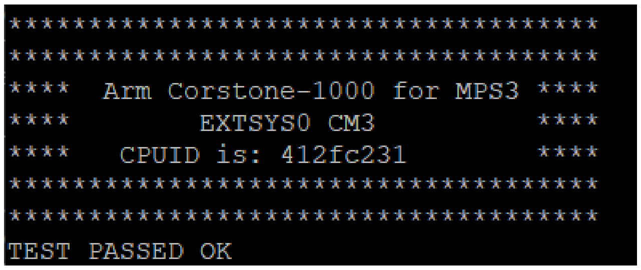
```
R/W SE QSPI Controller Write-Readback-Verify

Erasing and writing in progress please wait........
Writing complete

Readback & Verify in progress please wait........

+++++++++++++++++++ QSPI R/W CONTROLLER Test Pass +++++++++++++++

Verified 8/8 writings



OCVM Basic Test
************************************************************************

OCVM DDR4 HOST ADDR 0x80000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!

OCVM DDR4 HOST ADDR 0xF8000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!


CVM Basic Test
************************************************************************

CVM HOST ADDR 0x02300000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!


EXPMST1 Basic Test
************************************************************************

EXPMST1   HOST ADDR 0x60000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!


De-asserting HOST CPUWAIT
************************************************************************

Deasserted Host CPUWAIT
TEST PASSED OK
```

On the SSE-710 UART0 (Serial Port 2), the Cortex-A35 boots preload test code, as shown below.



```
**********************************************
**********************************************
**** AN550 v2 Arm Corstone-1000 for MPS3 ****
****          HOST CPU CA35 MP1          ****
****          CPUID is: 411FD040         ****
**********************************************
**********************************************


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

Read FPGA SCC AID Reg                     : Addr: 40000ff8 Read: 2100708
Read FPGA SCC ID Reg                      : Addr: 40000ffc Read: 41045500
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
************************************************************************



XIP QSPI CONTROLLER READBACK-VERIFY
READBACK VERIFY IN PROGRESS PLEASE WAIT
................................
******************* QSPI R/W CONTROLLER TEST COMPLETE ***************
 Verified 32/32 writings


OCVM Basic Test
************************************************************************

OCVM DDR4 HOST ADDR 0x80000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!

OCVM DDR4 HOST ADDR 0xF8000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!


EXPMST1 Basic Test
************************************************************************

EXPMST1   HOST ADDR 0x60000000 Byte access
memory test writing completed
memory test reading completed
TEST PASSED!


De-assert EXTSYS0 CPUWAIT
************************************************************************

Extsys0 CPUWAIT de-asserted.
```

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



```
*************************************
*************************************
****   Arm Corstone-1000 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.

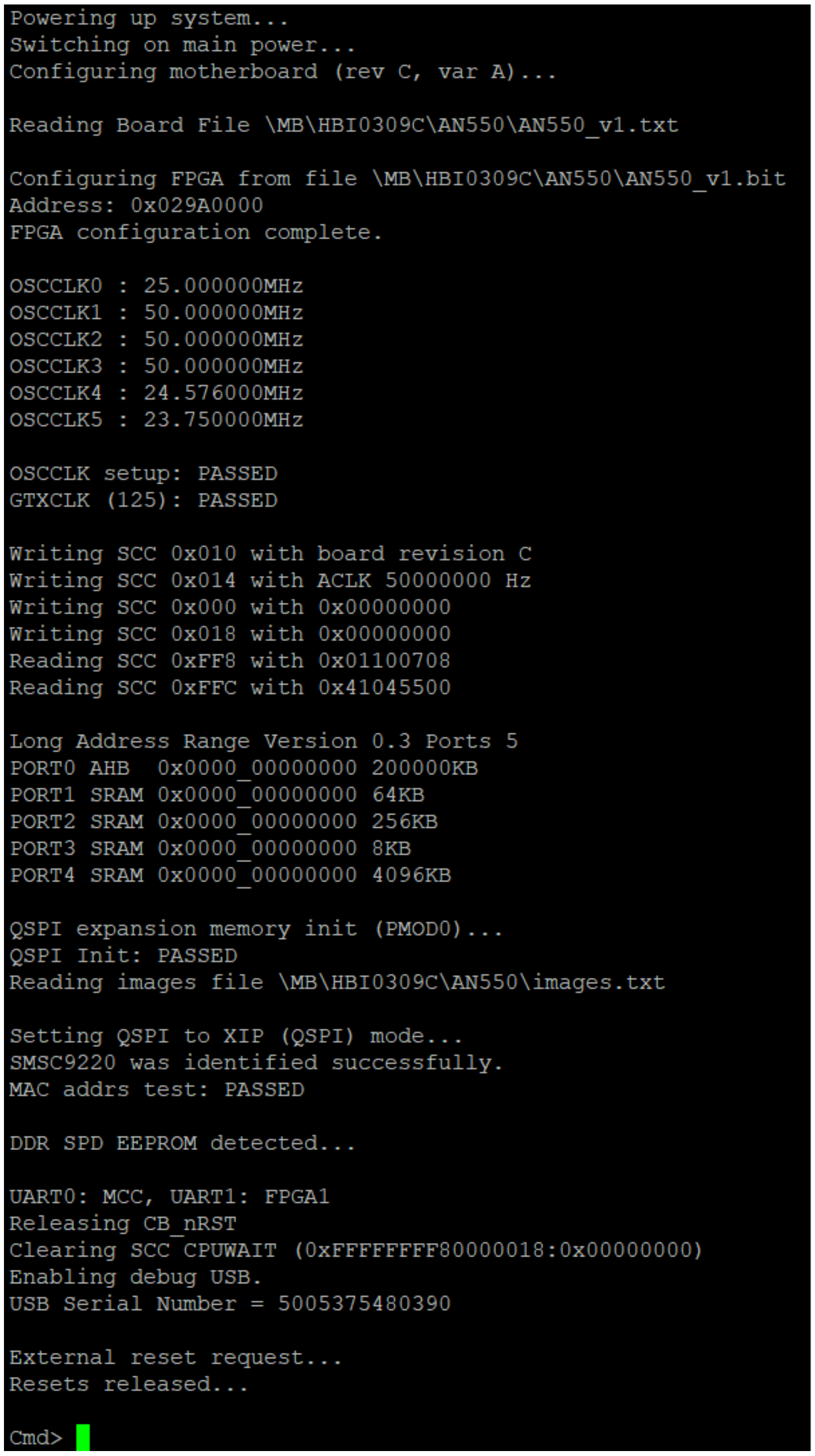
8.6 MCC Debug UART – Serial Port 0

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

8.6.1 MCC Debug UART – Verbose Output

It is possible to enter debug mode on the MCC UART. To do so change images.txt file TOTALIMAGES parameter to 0 and change config.txt UARTMODE from 2 to 1. At this point reboot the board. Once the initialization process is completed and you have received output on all four UARTs press PBON to perform a software reset. An example output from the MCC UART at this point in time can be seen below.



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

Reading Board File \MB\HBI0309C\AN550\AN550_v1.txt

Configuring FPGA from file \MB\HBI0309C\AN550\AN550_v1.bit
Address: 0x029A0000
FPGA configuration complete.

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

OSCCLK setup: PASSED
GTXCLK (125): PASSED

Writing SCC 0x010 with board revision C
Writing SCC 0x014 with ACLK 50000000 Hz
Writing SCC 0x000 with 0x00000000
Writing SCC 0x018 with 0x00000000
Reading SCC 0xFF8 with 0x01100708
Reading SCC 0xFFC with 0x41045500

Long Address Range Version 0.3 Ports 5
PORT0 AHB  0x0000_00000000 200000KB
PORT1 SRAM 0x0000_00000000 64KB
PORT2 SRAM 0x0000_00000000 256KB
PORT3 SRAM 0x0000_00000000 8KB
PORT4 SRAM 0x0000_00000000 4096KB

QSPI expansion memory init (PMOD0)...
QSPI Init: PASSED
Reading images file \MB\HBI0309C\AN550\images.txt

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

DDR SPD EEPROM detected...

UART0: MCC, UART1: FPGA1
Releasing CB_nRST
Clearing SCC CPUWAIT (0xFFFFFFFF80000018:0x00000000)
Enabling debug USB.
USB Serial Number = 5005375480390

External reset request...
Resets released...

Cmd>
```

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

8.6.2 MCC Debug UART – Reboot Control

The command **`reboot`** on the terminal emulator console initiates MCC reboot.

8.7 Preload of the SE ROM, EXTSYS0 Code SRAM, OTP 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, OTP and the QSPI Flash on the MPS3 board at MCC boot time. The user can control this process by editing **`MB/ HBI0309{<boardrev>}/AN550/images.txt`** file, which specifies the location of each image on the SD card, and whether it is pre-loaded or not.

8.7.1 The Preload Sequence

The preload sequence is:

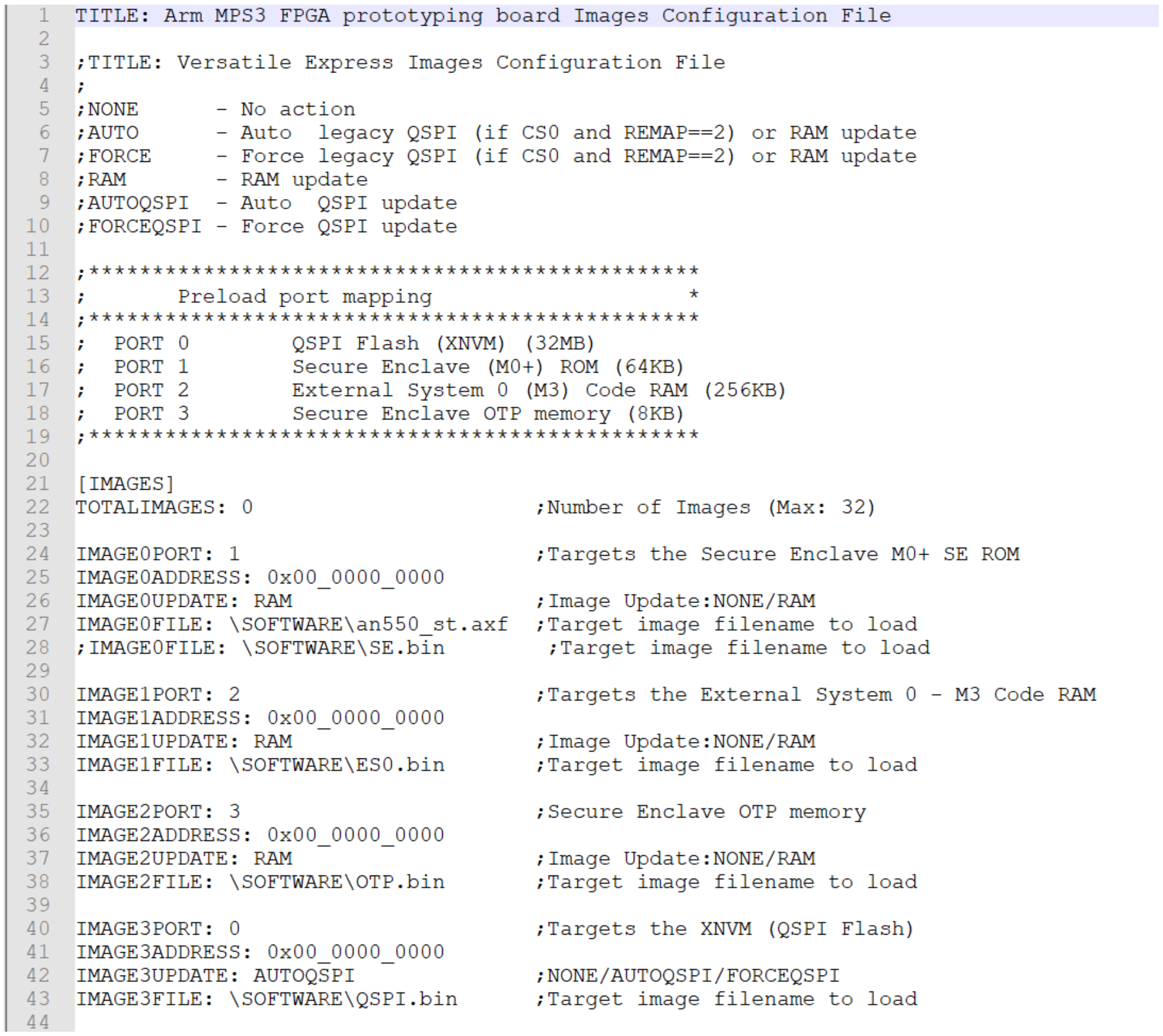
1. MPS3 is powered up.
2. The "PBON" button on the MPS3 board is pressed.
3. The MCC configures the FPGA with the FPGA image, which is specified in the FPGA configuration file in **`MB/HBI0309{<boardrev>}/AN550/an550_v2.txt`**.
4. The FPGA SoC NIC-400 reset is released.
5. SSE-710 Top and EXTSYS0 are 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, OTP and the QSPI Flash using the specified images (if directed to do so in the **`images.txt`** file settings (see section 8.7.2 for more details)).
7. The MCC de-asserts reset to SSE-710 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.

8.7.2 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/ HBI0309{<boardrev>}/AN550/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 to load the SE ROM, EXTSYS0 Cortex-M3 Code SRAM, OTP and the QSPI Flash. Some of prebuilt images are also supplied in the bundle. This can be observed below:



```
TITLE: Arm MPS3 FPGA prototyping board Images Configuration File

;TITLE: Versatile Express Images Configuration File
;
;NONE      - No action
;AUTO      - Auto  legacy QSPI (if CS0 and REMAP==2) or RAM update
;FORCE     - Force legacy QSPI (if CS0 and REMAP==2) or RAM update
;RAM       - RAM update
;AUTOQSPI  - Auto  QSPI update
;FORCEQSPI - Force QSPI update

;*************************************************
;       Preload port mapping                     *
;*************************************************
;  PORT 0        QSPI Flash (XNVM) (32MB)
;  PORT 1        Secure Enclave (M0+) ROM (64KB)
;  PORT 2        External System 0 (M3) Code RAM (256KB)
;  PORT 3        Secure Enclave OTP memory (8KB)
;*************************************************

[IMAGES]
TOTALIMAGES: 0                      ;Number of Images (Max: 32)

IMAGE0PORT: 1                       ;Targets the Secure Enclave M0+ SE ROM
IMAGE0ADDRESS: 0x00_0000_0000
IMAGE0UPDATE: RAM                   ;Image Update:NONE/RAM
IMAGE0FILE: \SOFTWARE\an550_st.axf  ;Target image filename to load
;IMAGE0FILE: \SOFTWARE\SE.bin        ;Target image filename to load

IMAGE1PORT: 2                       ;Targets the External System 0 - M3 Code RAM
IMAGE1ADDRESS: 0x00_0000_0000
IMAGE1UPDATE: RAM                   ;Image Update:NONE/RAM
IMAGE1FILE: \SOFTWARE\ES0.bin       ;Target image filename to load

IMAGE2PORT: 3                       ;Secure Enclave OTP memory
IMAGE2ADDRESS: 0x00_0000_0000
IMAGE2UPDATE: RAM                   ;Image Update:NONE/RAM
IMAGE2FILE: \SOFTWARE\OTP.bin       ;Target image filename to load

IMAGE3PORT: 0                       ;Targets the XNVM (QSPI Flash)
IMAGE3ADDRESS: 0x00_0000_0000
IMAGE3UPDATE: AUTOQSPI              ;NONE/AUTOQSPI/FORCEQSPI
IMAGE3FILE: \SOFTWARE\QSPI.bin      ;Target image filename to load

```

Note

A ";" at the start of any line in the **`images.txt`** file denotes a comment and is ignored by the MPS3 MCC.

Line 22 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 PORT, 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", "IMAGE2xxxx", "IMAGE3xxx"`**. 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.

8.7.3 Preloading QSPI Memories

`IMAGExUPDATE` - Informs the MCC how it should load the data. The options for this are:

o **`NONE | RAM | AUTOQSPI | FORCEQSPI`**

It is advised that any new entries use **`RAM`** except when targeting QSPI memories when **`AUTOQSPI / FORCEQSPI`** should be used.

o **`RAM`** - Informs the MCC to load the specified file to the specified address.

o **`FORCEQSPI`** - Informs the MCC to load QSPI memory using the required driver to the base address specified in `an550_v2.txt` plus the offset specified by **`IMAGExADDRESS`**

o **`AUTOQSPI`** - Informs the MCC to load QSPI memory using the required driver to the base address specified in `an550_v2.txt` plus the offset specified by **`IMAGExADDRESS`** only if stored on QSPI content in specified address is different from content of the file in **`IMAGExFILE`** field

• **`IMAGExFILE`** - This is the location on the SD card of the software image to load and it must use the SFN 8.3 format for the filename.

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:

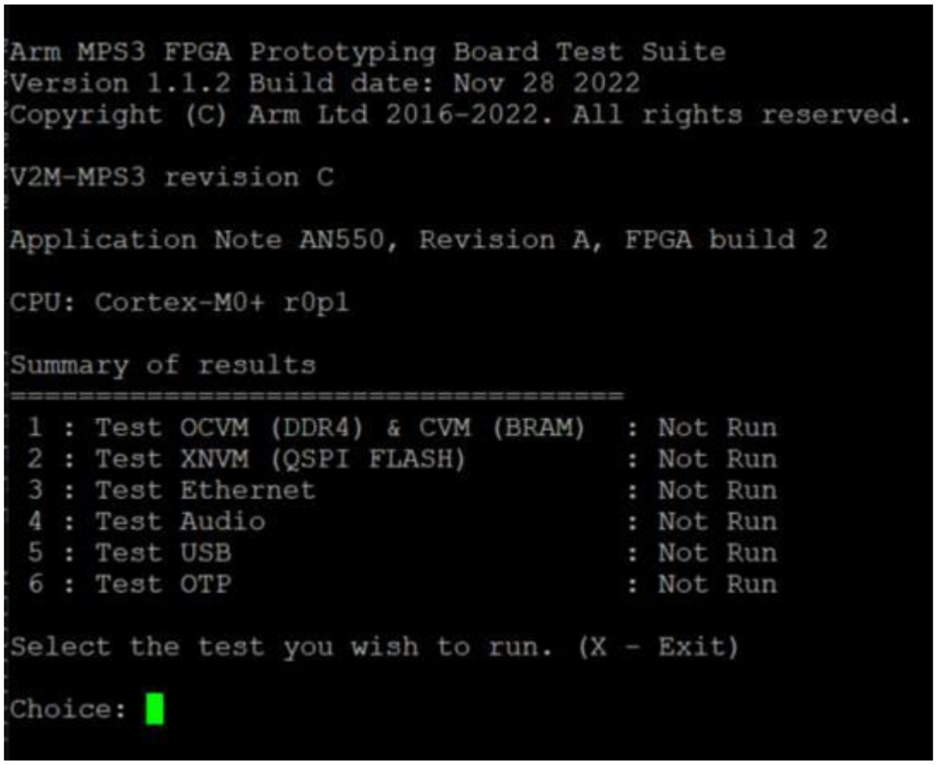
a. press the **`PBRST`** button on the MPS3 board, followed by a press of the **`PBON`** button to reboot the MCC.
b. Issue a **`reboot`** command on the MCC console, as described in section 8.6.2.

8.7.4 Preload selftest software

To preload the selftest software (`an550_st.axf`) shipped with the application note zip bundle, use the following settings in `images.txt` file and reboot the board.

```
;TITLE: Arm MPS3 FPGA prototyping board Images Configuration File

;NONE      - No action
;AUTO      - Auto  legacy QSPI (if CS0 and REMAP==2) or RAM update
;FORCE     - Force legacy QSPI (if CS0 and REMAP==2) or RAM update
;RAM       - RAM update
;AUTOQSPI  - Auto  QSPI update
;FORCEQSPI - Force QSPI update

;*************************************************
;       Preload port mapping                     *
;*************************************************
;  PORT 0        QSPI Flash (XNVM) (32MB)
;  PORT 1        Secure Enclave (M0+) ROM (64KB)
;  PORT 2        External System 0 (M3) Code RAM (256KB)
;  PORT 3        Secure Enclave OTP memory (8KB)
;*************************************************

[IMAGES]
TOTALIMAGES: 1                      ;Number of Images (Max: 32)

IMAGE0PORT: 1                       ;Targets the Secure Enclave M0+ SE ROM
IMAGE0ADDRESS: 0x00_0000_0000
IMAGE0UPDATE: RAM                   ;Image Update:NONE/RAM
IMAGE0FILE: \SOFTWARE\an550_st.axf  ;Target image filename to load
;IMAGE0FILE: \SOFTWARE\SE.bin        ;Target image filename to load
```

After the board boots up, press the `PBON` push button on the MPS3 board. This will load the selftest menu on SE UART



```
Arm MPS3 FPGA Prototyping Board Test Suite
Version 1.1.2 Build date: Nov 28 2022
Copyright (C) Arm Ltd 2016-2022. All rights reserved.

V2M-MPS3 revision C

Application Note AN550, Revision A, FPGA build 2

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
 6 : Test OTP                       : Not Run

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

Choice:
```

8.8 Supported Preload SE ROM, EXTSYS0 Code SRAM, OTP 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

8.9 Supported Preload SE ROM, EXTSYS0 Code SRAM, OTP 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.

8.10 Other applications

See https://corstone1000.docs.arm.com/ for more information on the application software available for this image.