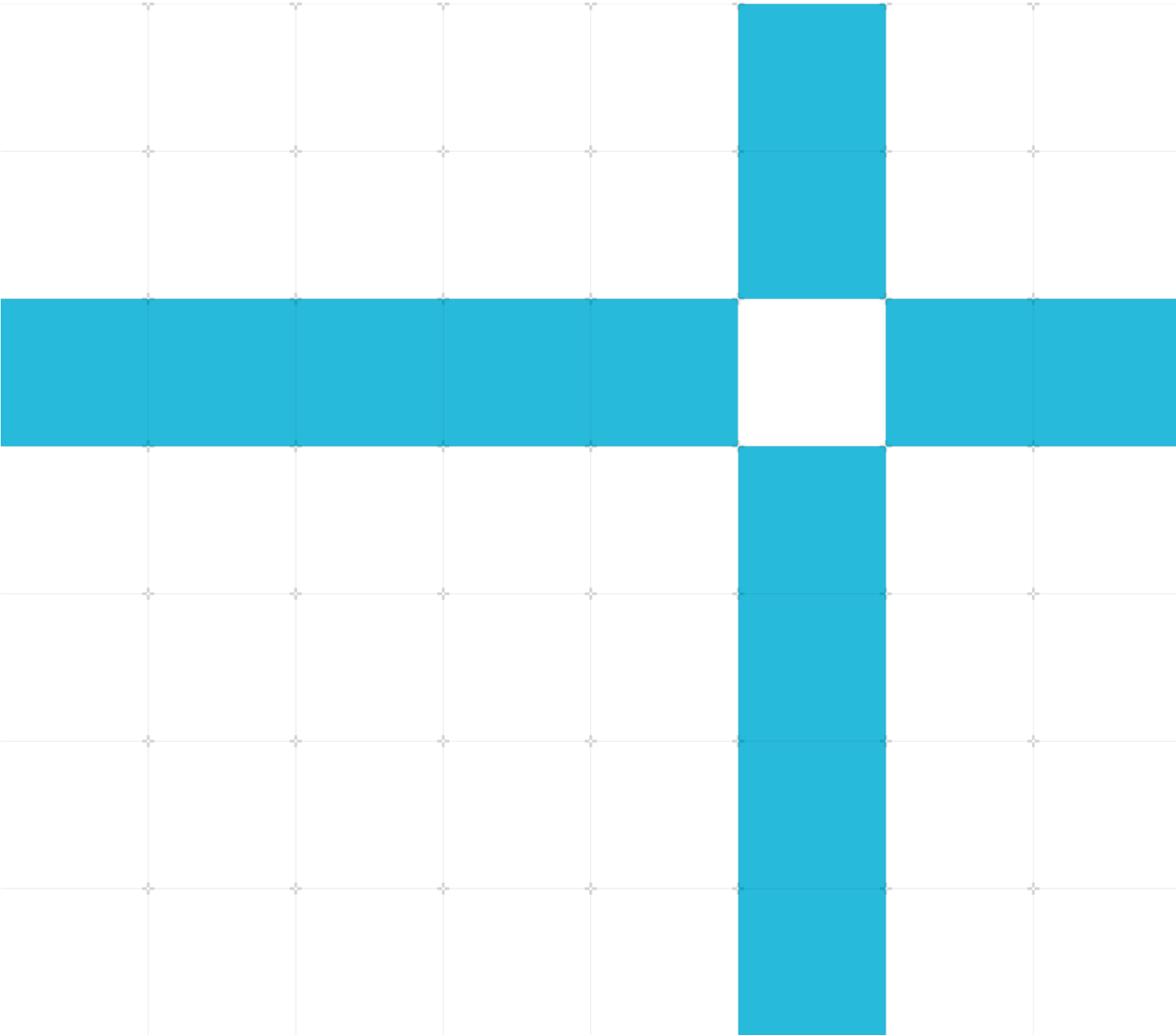
Example Cortex™-R52x2 Subsystem for MPS3

Application Note AN536

Non-Confidential

Issue B

DAI 0536



Example Cortex™-R52x2 Subsystem for MPS3

Application Note AN536

Release information

Document history

Issue	Date	Confidentiality	Change
A	18 February 2019	Non-Confidential	First release.
B	30 September 2021	Non-Confidential	Update to the document template Section 2.2 FPGA utilization has been added Table 2-3 : Memory map overview table has been updated with TCM sizes Added section 12 on debug

Non-Confidential Proprietary Notice

languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

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

Product Status

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

Web Address

http://www.arm.com

Progressive terminology 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.
This document includes terms that can be offensive. We will replace these terms in a future issue of this document. If you find offensive terms in this document, please contact 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 SSE-200 Subsystem and other Arm technology. FPGA bitstream file containing example of User Partition.

Hardware Source Code:

Hardware netlists of Arm® CoreLink™ NIC-400, PrimeCell Infrastructure AMBA™ 2 AHB™ to AMBA 3 AXI™ Bridges (BP136).
RTL of Arm® PrimeCell Synchronous Serial Port (PL022) and Real Time Clock (PL031)
apb_i2s_top, CharLCDI, SBCon.
RTL of components in the Arm® Cortex®-M System Design Kit (CMSDK) including:
cmsdk_sram, cmsdk_ahb_gpio, cmsdk_apb_uart, cmsdk_irq_sync, cmsdk_to_extmem.

Software Binaries:
Motherboard Configuration Controller binary, including Arm®Keil® USB and SD card drivers, and Analog Devices FMC EEPROM reader.
selftest binary.

Documentation:
Documentation, provided as PDF

Part B

Wrapper:
Wrapper file(s) identified in the documentation provided as hardware source files and netlists.

Part C

Example Code:
Platform initialisation source code
Platform specific libraries and source code
selftest example source code
Demo example source code
Arm source code portions of the selftest

Part D

None

Contents

1 Introduction

1.1 Purpose of this application note

This document describes the features and functionality of application note AN536. AN536 is an FPGA implementation of the Cortex R52x2 Subsystem that uses SoC-400 and NIC-400 components together with CMSDK peripherals to provide an example design.

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 special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`Monospace` **`bold`**	Denotes language keywords when used outside example code.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace underline`	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>`
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
	Warning
	Note

1.4 Additional reading

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

Document name	Document ID	Licensee only Y/N
Arm Cortex-R52 Processor Technical Reference	*100026_0101_01_en*	No
Arm CoreSight™ SoC-400 Technical Reference Manual	*ARM DDI 0480G ID042315*	No
Arm CoreLink SSE-200 Subsystem for Embedded Technical Reference Manual	*101104_0200_00_en*	No
Arm MPS3 FPGA Prototyping Board Technical Reference Manual	*100765_0000_04_en*	No
Arm Cortex-M System Design Kit Technical Reference Manual	*DDI 0479*	No
Arm MPS3 FPGA Prototyping Board Getting Started Guide	-	No
MCBQVGA-TS-Display-v12 – Keil MCBSTM32F200 display board schematic	-	No

Table 1-1 : Arm publications

Document name	Document ID
Xilinx Vivado Design Suite User Guide	UG909

Table 1-2 : Other publications

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:

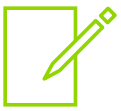
- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.5.2 Feedback on content

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

- The title *Example Cortex-R52x2 Subsystem for MPS3- Application Note AN536*.
- The number DAI 0536, Issue B.
- 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.



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

AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
AXI	Advanced eXtensible Interface
AXIM	Advanced eXtensible Interface Manager Port
BRAM	Block Random Access Memory
CMSDK	Cortex® -M System Design Kit
DCP	Design Checkpoint
DMA	Direct Memory Access
DS	Arm® Development Studio
EAM	Exclusive Access Controller
FPGA	Field Programmable Gate Array
I2S	Inter-IC Sound
IDAU	Implementation Defined Attribution Unit
KB	Kilo Byte
LLPP	Low Latency Peripheral Port
MB	Mega Byte
MCC	Motherboard Configuration Controller
MPC	Memory Protection Controller
MSC	Manager Security Controller
PPC	Peripheral Protection Controller
PR	Partial Reconfiguration
QSPI	Quad serial peripheral interface
RAM	Random Access Memory
RTC	Real Time Clock
RTL	Register Transfer Level
SSE	Subsystem for Embedded
SCC	Serial Configuration Controller
SMB	Static Memory Bus
SMM	Soft Macrocell Model
SPI	Serial Peripheral Interface
TRM	Technical Reference Manual
UART	Universal Asynchronous Receiver/Transmitter
XIP	Execute in place

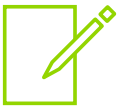
1.7 Subsystem version details

Version	Descriptions
r1p1	Arm Cortex-R52 Processor The Cortex-R52 processor is a mid-performance, in-order, superscalar processor primarily for use in automotive and industrial applications. It is also suited to a wide variety of other embedded applications such as communication and storage devices.
r3p2	Arm CoreSight SoC-400 CoreSight SoC-400 is a solution for debug and trace of complex SoCs
BP210	Cortex-M System Design Kit Full version of the design kit supporting Cortex-M0, Cortex-M0 DesignStart®, Cortex-M0+, Cortex-M3 and Cortex-M4. Also contains the AHB Bus Matrix and advanced AHB components.
r1p3-00rel1	PL022 Arm PrimeCell Synchronous Serial Port
r1p0	NIC-400 The CoreLink NIC-400 Network Interconnect is highly configurable and enables you to create a complete high performance, optimized, and AMBA-compliant network infrastructure

Table 1-3 : Module 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.



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

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.

2 Overview

This SMM is based around the dual Cortex-R52 system, the system is then extended with interconnect and peripherals.

The SMM is implemented using Partial Reconfiguration which allows the user to modify the user partition shown below.

2.1 System block diagram

The diagram below shows the high level of the full MPS3 CR52x2 FPGA System.

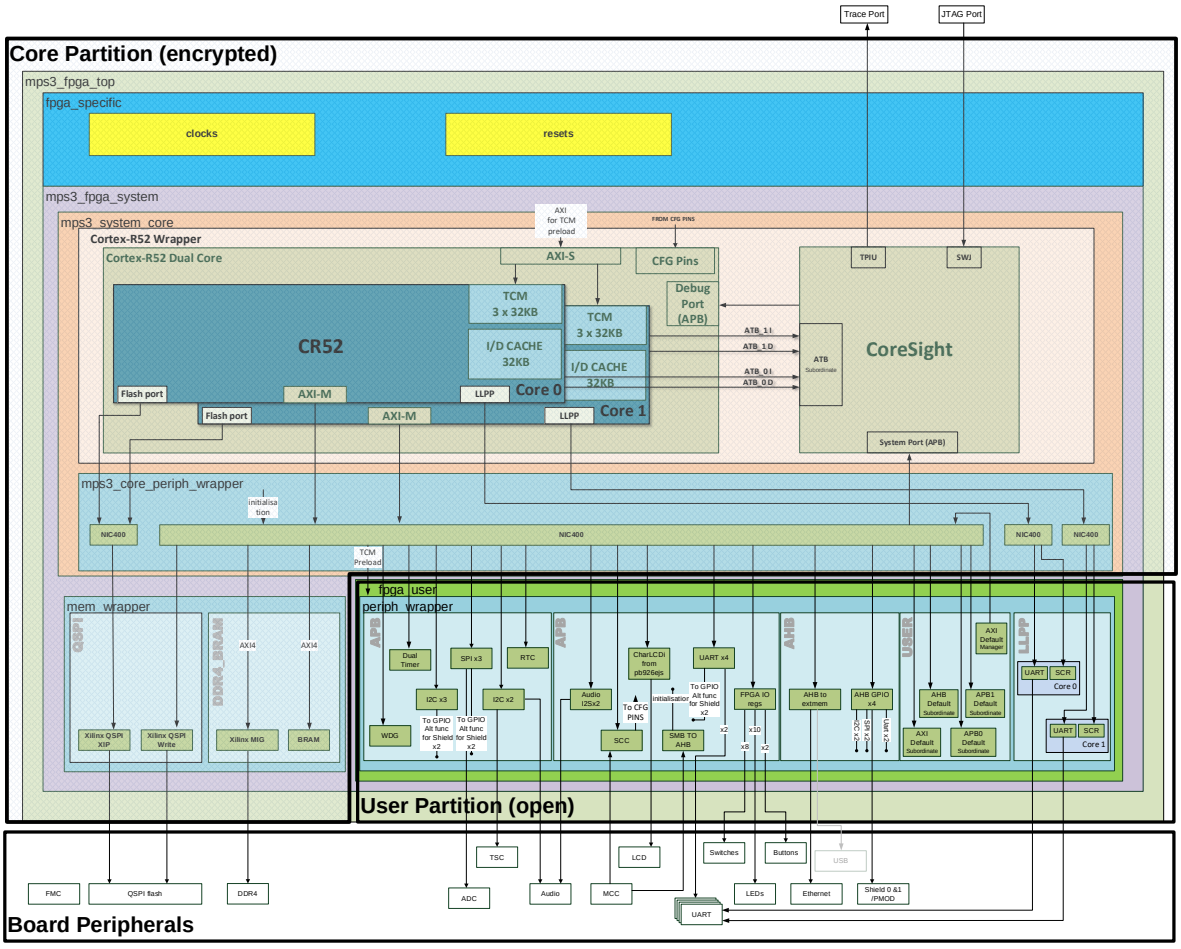


Figure 2-1 : System overview

Note how the FPGA Subsystem extends the SSE-200 Subsystem by adding to its expansion interfaces.

2.2 FPGA Utilization

This application note is designed for MPS3 board. The board will use a a Xilinx Kintex Ultrascale XCKU115 FPGA. The FPGA features up to 8MB BRAM (2160 BlocRAM tiles) and up to 663360 LUTs.

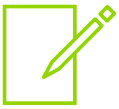
Full part number: XCKU115-FLVB1760-1-C.

2.2.1 Total design utilization

The following table shows the total number of LUTs and BRAMs currently used in the provided image.

Site Type	Used	Utilization %
LUTs	349098	52
BlockRAM Tile	259	12

Table 2-1 : AN536 utilization summary



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.

2.2.2 User partition

User-reserved area is about 20% of total FPGA. The provided example design is using 6% of User-reserved area.

Site Type	Total Available	Used	Utilization %
LUTs	135840	8687	6
BlockRAM Tile	450	0	0

Table 2-2 : Size and utilization of the User Partition

2.3 Memory map overview

The memory map as viewed from the CR52 core:

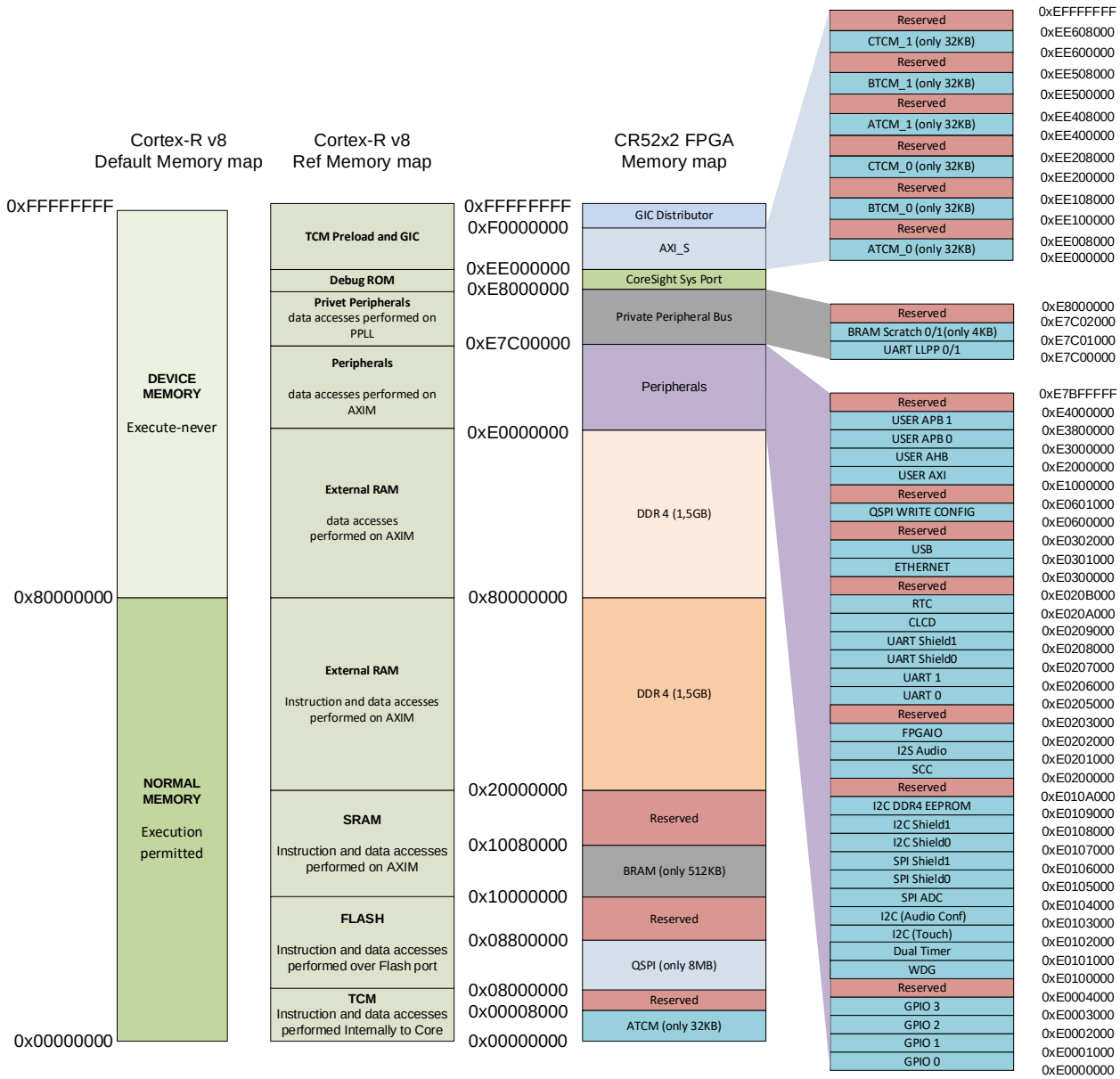


Figure 2-1 : Memory Map

The following table shows the memory map overview:

ROW ID	Address From	Address To	Size	Region Name	Description
1	0x0000_0000	0x0000_7FFF	32KB	ATCM Memory	BRAM
2	0x0000_8000	0x07FF_FFFF	128MB	Reserved	Reserved
3	0x0800_0000	0x087F_FFFF	8MB	Flash Memory	QSPI
4	0x0880_0000	0x0FFF_FFFF	120MB	Reserved	Alias to QSPI
5	0x1000_0000	0x1007_FFFF	512KB	SRAM	BRAM
6	0x1008_0000	0x1FFF_FFFF	255MB	Reserved	Alias to SRAM
7	0x2000_0000	0xDFFF_FFFF	3GB	DRAM	DDR4
8	0xE000_0000	0xE3FF_FFFF	64MB	Peripherals	Shared Peripheral Region.
9	0xE400_0000	0xE7BF_FFFF	60MB	Reserved	Reserved
10	0xE7C0_0000	0xE7C0_1FFF	8KB	Private Peripherals	Private Peripheral Region
11	0xE7C0_2000	0xE7FF_FFFF	4MB	Reserved	Reserved
12	0xE800_0000	0xEDFF_FFFF	96MB	CoreSight	Debug System port.
13	0xEE00_0000	0xEE00_7FFF	32KB	ATCM_0 Memory	BRAM
14	0xEE00_8000	0xEE0F_FFFF	992KB	Reserved	Reserved
15	0xEE10_0000	0xEE10_7FFF	32KB	BTCM_0 Memory	BRAM
16	0xEE10_8000	0xEE1F_FFFF	992KB	Reserved	Reserved
17	0xEE20_0000	0xEE20_7FFF	32KB	CTCM_0 Memory	BRAM
18	0xEE20_8000	0xEE3F_FFFF	2016KB	Reserved	Reserved
19	0xEE40_0000	0xEE40_7FFF	32KB	ATCM_1 Memory	BRAM
20	0xEE40_8000	0xEE4F_FFFF	992KB	Reserved	Reserved
21	0xEE50_0000	0xEE50_7FFF	32KB	BTCM_1 Memory	BRAM
22	0xEE50_8000	0xEE5F_FFFF	992KB	Reserved	Reserved
23	0xEE60_0000	0xEE60_7FFF	32KB	CTCM_1 Memory	BRAM
24	0xEE60_8000	0xEFFF_FFFF	26MB	Reserved	Reserved
23	0xF000_0000	0xF019_FFFF	1664KB	GIC Distributor	Internal GIC
24	0xF01A_0000	0xFFFF_FFFF	254MB	Reserved	Reserved

Table 2-3 : Memory map overview

2.4 MCC Memory map

MCC should have certain visibility into memory for initiating boot memory areas and configuring peripherals if needed. MCC has a limited access to the design memory map (just 4x64MB), so it is unable to cover the whole map, hence only those regions which are necessary for the design functionality are visible.

The memory map as viewed from the MCC :

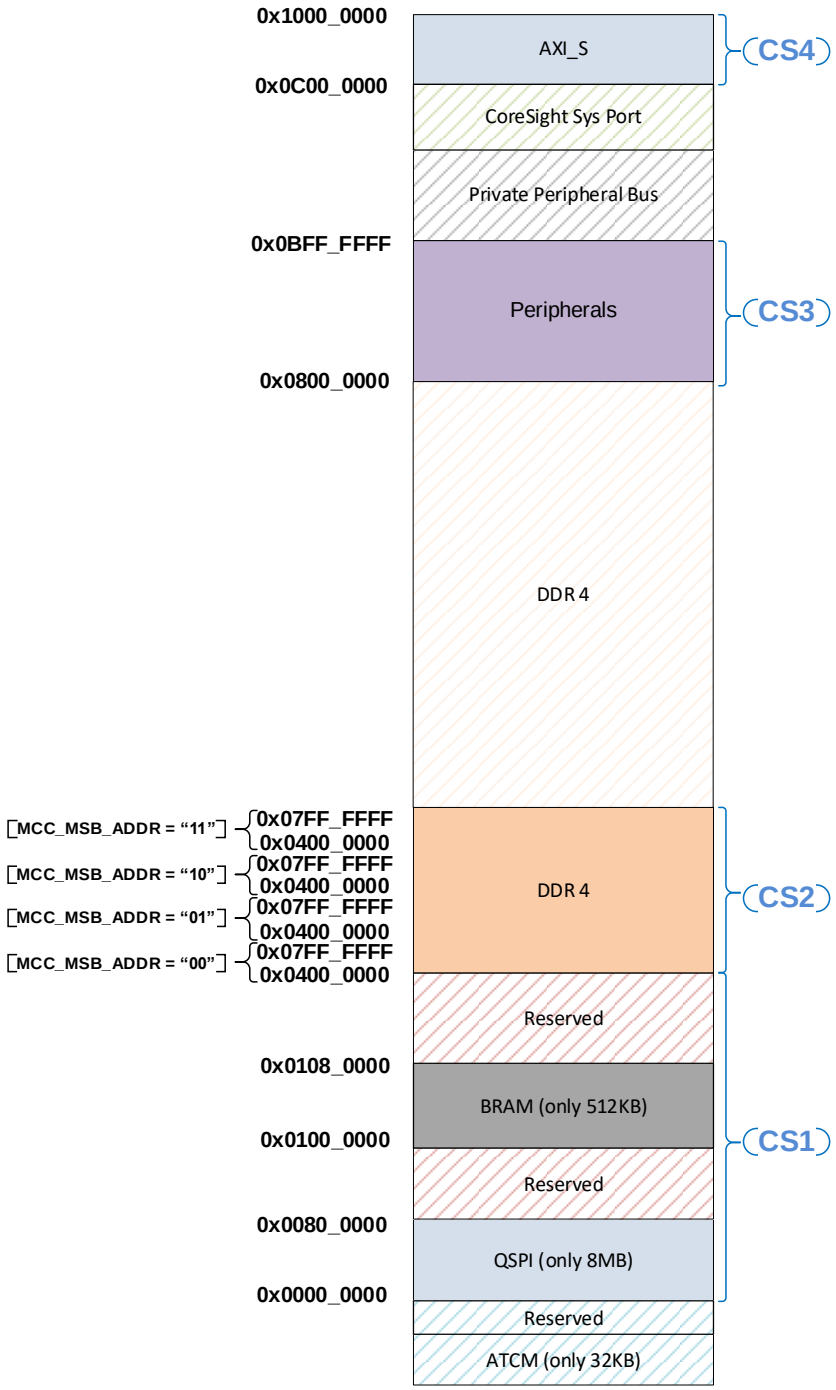


Figure 2-2 : MCC memory map for AN536

2.5 Shared peripherals

Shared peripherals are accessible from cores over AXIM or from other interconnect managers in the memory map region from 0xE000_0000 to 0xE7BF_FFFF.

ROW ID	Address		Size	Description	Port
	From	To			
1	0xE000_0000	0xE000_0FFF	4K	GPIO 0	AHB
2	0xE000_1000	0xE000_1FFF	4K	GPIO 1	
3	0xE000_2000	0xE000_2FFF	4K	GPIO 2	
4	0xE000_3000	0xE000_3FFF	4K	GPIO 3	
	0xE000_4000	0xE00F_FFFF		Reserved	
5	0xE010_0000	0xE010_0FFF	4K	WDG	APB
6	0xE010_1000	0xE010_1FFF	4K	Dual Timer	
7	0xE010_2000	0xE010_2FFF	4K	FPGA - SBCon I2C (Touch)	
8	0xE010_3000	0xE010_3FFF	4K	FPGA - SBCon I2C (Audio Conf)	
9	0xE010_4000	0xE010_4FFF	4K	FPGA - PL022 (SPI ADC)	
10	0xE010_5000	0xE010_5FFF	4K	FPGA - PL022 (SPI Shield0)	
11	0xE010_6000	0xE010_6FFF	4K	FPGA - PL022 (SPI Shield1)	
12	0xE010_7000	0xE010_7FFF	4K	SBCon (I2C - Shield0)	
13	0xE010_8000	0xE010_8FFF	4K	SBCon (I2C - Shield1)	
14	0xE010_9000	0xE010_9FFF	4K	FPGA - SBCon I2C (DDR4 EEPROM)	
	0xE010_A000	0xE01F_FFFF		Reserved	
15	0xE020_0000	0xE020_0FFF	4K	FPGA - SCC registers	APB
16	0xE020_1000	0xE020_1FFF	4K	FPGA - I2S (Audio)	
17	0xE020_2000	0xE020_2FFF	4K	FPGA - IO (System Ctrl + I/O)	
	0xE020_3000	0xE04F_FFFF		Reserved	
18	0xE020_5000	0xE020_5FFF	4K	UART0 - UART_F[2]	
19	0xE020_6000	0xE020_6FFF	4K	UART1 - UART_F[3]	
20	0xE020_7000	0xE020_7FFF	4K	UART3 - UART Shield 0	
21	0xE020_8000	0xE020_8FFF	4K	UART4 - UART Shield 1	
22	0xE020_9000	0xE020_9FFF	4K	CLCD Config Reg	
23	0xE020_A000	0xE020_AFFF	4K	RTC	
	0xE020_B000	0xE02F_FFFF		Reserved	
24	0xE030_0000	0x E030_0FFF	4K	Ethernet	AHB
25	0xE030_1000	0x E030_1FFF	4K	USB	
	0xE030_2000	0x E05F_FFFF		Reserved	
26	0xE060_0000	0xE060_0FFF	4K	QSPI Write Config	APB
	0xE060_1000	0x E0FF_FFFF		Reserved	
27	0xE100_0000	0xE1FF_FFFF	16M	User AXI	AXI
28	0xE200_0000	0xE2FF_FFFF	16M	User AHB	AHB
29	0xE300_0000	0xE37F_FFFF	8M	User APB0	APB
30	0xE380_0000	0xE3FF_FFFF	8M	User APB1	APB
	0xE400_0000	0xE7B_FFFF		Reserved	

Table 2-4 : FPGA Shared Peripheral Map

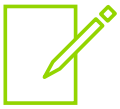
Reserved regions should not be accessed .

2.6 Private peripherals

Private peripherals are accessible only from cores over LLPP in the memory map region from 0xE7C0_0000 to 0xE7FF_FFFF per each core.
The following table shows the FPGA private peripheral map:

ROW ID	Address		Size	Description	Port
	From	To			
1	0xE7C0_0000	0xE7C0_0FFF	4K	UART LLPP	APB
2	0xE7C0_1000	0xE7C0_1FFF	4K	SCR BRAM	AHB
	0xE7C0_2000	0xE7FF_FFFF		Reserved	

Table 2-5 : FPGA private peripheral map



Reserved regions should not be accessed.

3 Programmers Model

3.1 CR52x2 Cluster

The SMM Cortex-R52x2 is based on Cortex-R52 r1p1-00rel0. The Cortex-R52 Dual Core Integration Level is used with additional logic wrapped around it to create the system design

Processor feature	Configuration	Notes
Number CPUs	2	-
LOCK_STEP	No	-
I-cache	32kB	-
D-cache	32kB	-
ATCM	32kB	-
BTCM	32kB	-
CTCM	32kB	-
FPU	Single and Double Precision	-
NEON	NEON SIMD support	-
LLPP	2	-
AXI Managers	2	-
AXI Subordinates	1	-

Table 3-1 : Processor configuration

3.2 Debug

The design contains a CoreSight implementation with a trace module. Connection to the debugger is via the JTAG (CS_20W_2.54mm) J14 connector on the MPS3 motherboard. Connection to the trace module is via the Trace J13 connector on the MPS3 mother board.

3.3 CMSDK components

This programmer's model is supplemental to the CMSDK documentation which covers many of the included components in more detail. Figure 2-1 : System overview shows the connectivity of the system.

3.4 BRAM

The memory is 512KB of Internal FPGA SRAM. This memory is mapped to the address range 0x10000000 - 0x1007FFFF.

3.5 QSPI

The secondary memory is 8MB of external Flash memory which is accessed via a QSPI interface. This memory is mapped to the address-range 0x08000000 - 0x087FFFFF by default but can be removed by R52 configuration pins: CFGFLASHENx in SCC registers.

3.6 DDR4

The AN also includes 3GB of External DDR4 memory which is allocated to the address-range 0x20000000 - 0xDFFFFFFF.

3.7 AHB GPIO

The SMM uses four CMSDK AHB GPIO blocks, each providing 16 bits of IO. These are connected to the two Arduino compatible headers shield 0 and 1 as follows.

Shield	GPIO
SH0_IO [15:0]	GPIO0[15:0]
SH0_IO [17:16]	GPIO2[1:0]
SH1_IO [15:0]	GPIO1[15:0]
SH1_IO [17:16]	GPIO2[3:2]

Table 3-2 : GPIO Mapping

The GPIO alternative function lines select whether or not peripherals or GPIOs are available on each pin. See section 6 - Shield Support for mappings.

3.8 SPI (Serial Peripheral Interface)

The SMM implements three PL022 SPI modules:

- One general purpose SPI module (SPI ADC) is used for communication with an onboard ADC. The analog pins of the Shield headers are connected to the input channels of the ADC.
- Two general purpose SPI modules connect to the Shield headers and provide an SPI interface on each header. These are alt-functions on the GPIO ports. See section 6 - Shield Support for mappings.

3.9 SBCon (I^2C)

The SMM implements five SBCon serial modules:

- One SBCon module for use by the Color LCD touch interface.
- One SBCon module to configure the audio controller.
- Two general purpose SBCon modules that connect to the Shield0 and Shield1 and provide an I2C interface on each header. These are alt-functions on the GPIO ports. See section 6 - Shield Support for mappings.
- One SBCon module is used to read EEPROM from DDR4 SODIMM.

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

Address	Name	Access	Description
0x000	SB_CONTROL	Read	Read serial control bits: Bit [0] is SCL Bit [1] is SDA
0x000	SB_CONTROLS	Write	Set serial control bits: Bit [0] is SCL Bit [1] is SDA
0x004	SB_CONTROLC	Write	Clear serial control bits: Bit [0] is SCL Bit [1] is SDA

Table 3-3 : SBCon Register Map

3.10 UART

The SMM implements six CMSDK UARTs:

- UART 0 – LLPP_UART0
- UART 1 – LLPP_UART1
- UART 2 – FPGA_UART2
- UART 3 - FPGA_UART3
- UART 4 - Shield 0
- UART 5 - Shield 1

UART 4 and 5 are alt-functions on the GPIO ports. See section 6 - Shield Support for mappings.

3.11 Color LCD parallel interface

The color LCD module has two interfaces:

- Parallel bus for sending image data to the LCD
- I^2C to transfer data input from the touch screen

The color LCD Module is a custom peripheral that provides an interface to an STMicroelectronics STMPE811QTR Port Expander with Advanced Touch Screen Controller on the Keil MCBSTM32C display board. (Schematic listed in the reference section). The Keil display board contains an AM240320LG display panel and uses a Himax HX8347-D LCD controller.

The selftest that is provided with the MPS3 includes drivers and example code for both these interfaces.

The following table shows the control and data registers for the CLCD interface :

Address	Name	Type	Information
0x4130_A000	CHAR_COM	Write command, read busy status	A write to this address causes a write to the LCD command register. A read from this address causes a read from the LCD busy register.
0x4130_A004	CHAR_DAT	Write data RAM, Read data RAM	A write to this address causes a write to the LCD data register. A read from this address causes a read from the LCD data register.
0x4130_A008	CHAR_RD	Read captured data from an earlier read command	Bits [7:0] contain the data from last request read, valid only when bit[0] is set in CHAR_RAW. Bits [31:8] are reserved.
0x4130_A00C	CHAR_RAW	Write to reset access complete flag, Read to determine if data in CHAR_RD is valid	Bit [0] indicates Access Complete (write 0b0 to clear). The bit is set if read data is valid. Bits [31:1] are reserved.
0x4130_A010	CHAR_MASK	Write interrupt mask	Set Bit [0] to 0b1 to enable Access Complete to generate an interrupt.
0x4130_A014	CHAR_STAT	Read status	Bit[0] is the state of Access Complete ANDed with the CHAR_MASK.
0x4130_A04C	CHAR_MISC	Miscellaneous Control	Bits [31:7] : Reserved. Bit[6]: CLCD_BL. Bit[5]: CLCD_RD. Bit[4]: CLCD_RS. Bit[3]: CLCD_RESET. Bit[2]: Reserved. Bit[1]: CLCD_WR. Bit[0]: CLCD_CS.

Table 3-4 : CLCD interface register map

3.12 Ethernet

The SMM design connects to an SMSC LAN9220 device through a static memory interface.

The self-test program includes example code for a simple loopback operation.

3.13 USB

The SMM design connects to a Hi-Speed USB OTG controller (ISP1763) device through a static memory interface.

USB will not be supported by software. The self test program will only include example code for simple ID register

3.14 Real Time Clock, RTC

The SMM uses PL031 PrimeCell Real Time Clock Controller (RTC). A counter in the RTC is incremented every second. The RTC can therefore be used as a basic alarm function or long time-base counter.

3.15 Audio I^2S and Configuration

The SMM has a single I2S module directly connected to the MPS3 back panel audio sockets.

The SMM also implements a simple SBCon interface based on I^2C. It is used to configure the Cirrus Logic Low Power Codec with Class D Speaker Driver, CS42L52 part on the MPS3 board.

3.16 FPGA system control and I/O

The SMM implements an FPGA system control block.

Address	Name	Information
0xE0202000	FPGAIO->LED0	LED connections Bits [31:10] : Reserved Bits [9:0] : LED
0xE0202004	RESERVED	
0xE0202008	FPGAIO->BUTTON	Buttons Bits [31:2] : Reserved Bits [1:0] : Buttons
0xE020200C	RESERVED	
0xE0202010	FPGAIO->CLK1HZ	1Hz up counter
0xE0202014	FPGAIO->CLK100HZ	100Hz up counter
0xE0202018	FPGAIO->COUNTER	Cycle Up Counter Increments when 32-bit prescale counter reach zero.
0xE020201C	FPGAIO->PRESCALE	Bits[31:0] – reload value *for* prescale counter.
0xE0202020	FPGAIO->PSCNTR	32-bit Prescale counter – current value of the pre-scaler counter. The Cycle Up Counter increment when the prescale down counter reach 0. The pre-scaler counter is reloaded with PRESCALE after reaching 0.
0xE0202024	RESERVED	-
0xE0202028	FPGAIO->SWITCH	Switches Bits [31:8] : Reserved Bits [7:0] : Switches
0xE020204C	FPGAIO->MISC	Misc control Bits [31:3] : Reserved Bit [2] : SHIELD1_SPI_nCS Bit [1] : SHIELD0_SPI_nCS Bit [0] : ADC_SPI_nCS

Table 3-5: System Control and I/O Memory Map

3.17 Serial Communication Controller (SCC)

The SMM implements communication between the microcontroller and the FPGA system through an SCC interface.

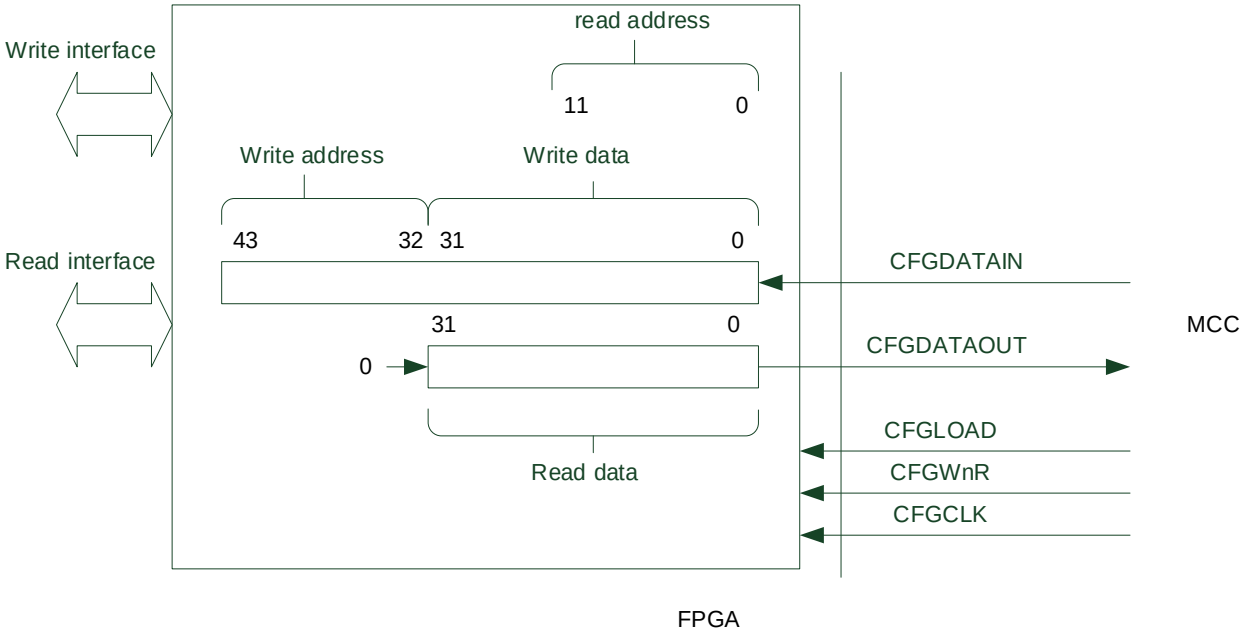


Figure 3-1 : Diagram of the SCC Interface

The read-addresses and write-addresses of the SCC interface do not use bits[1:0]
All address words are word-aligned.

Address	Name	Information
0x000	CFG_REG0	Bits [31:6]: Reserved Bit [5]: CPU 1 Reset (1 for Reset) Bit [4]: CPU 0 Reset (1 for Reset) Bits [3:2]: Reserved Bit [1]: Core 1 Halt mode (0 for Halt, 1 for Run) Bit [0]: Core 0 Halt mode (0 for Halt, 1 for Run)
0x004	CFG_REG1	Bits [31:6]: Reserved Bit [5]: CPU 1 Flash interface enabled or disabled out of reset. Bit [4]: CPU 1 ATCM enabled and at address 0x0 out of reset. Bit [1]: CPU 0 Flash interface enabled or disabled out of reset. Bit [0]: CPU 0 ATCM enabled and at address 0x0 out of reset.
0x008	CFG_REG2	Bits [31:1]: Reserved Bit [0]: QSPI Select signal
0x00C	MCC_MSB_ADDR	Bits [31:2]: Reserved Bits [1:0]: Additional MCC addressing bits
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 [31:5]: Core 0 Vector table base address out of reset. Bits [4:0]: Reserved
0x01C	CFG_REG7	Bits [31:5]: Core 1 Vector table base address out of reset. Bits [4:0]: Reserved
0x020 – 0x09C	RESERVED	-

Address	Name	Information
0x0A0	SYS_CFGDATA_RTN	32bit DATA [r/w]
0x0A4	SYS_CFGDATA_OUT	32bit DATA [r/w]
0x0A8	SYS_CFGCTRL	Bit [31] : Start (generates interrupt on write to this bit) Bit [30] : R/W access Bits[29:26] : Reserved Bits[25:20] : Function value Bits[19:12] : Reserved Bits[11:0] : Device (value of 0/1/2 for supported clocks)
0x0AC	SYS_CFGSTAT	Bit [0] : Complete Bit [1] : Error
0x0B0 – 0x0FC	RESERVED	-
0x100	SCC_DLL	DLL lock register Bits [31:24] DLL LOCK MASK[7:0] - These bits indicate if the DLL locked is masked. Bits [23:16] DLL LOCK MASK[7:0] - These bits indicate if the DLLs are locked or unlocked. Bits [15:1] : Reserved Bit [0] This bit indicates if all enabled DLLs are locked:
0x104 – 0xFF4	RESERVED	-
0xFF8	SCC_AID	SCC AID register is read only Bits[31:24] : FPGA build number Bits[23:20] : V2M-MPS3 target board revision (A = 0, B = 1, C = 2) Bits[19:8] : Reserved Bits[7:0] : number of SCC configuration register
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[15:4] : Primary part number: 536 = AN536 Bits[3:0] : Reserved

Table 3-6: SCC Register memory map

4 Clock architecture

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

4.1 Source clocks

The following clocks are inputs to the FPGA:

Input Pin	Board File Name	Frequency	Note
OSCCLK[0]	-	24MHz	Constant 24MHz reference, used for RTC and timers.
OSCCLK[1]	OSC1	50MHz	ACLK, main clock used to clock SSE-200 subsystem. Frequency can be changed in the board file an536_v1.txt
OSCCLK[2]	OSC2	50MHz	MCLK, Reserved
OSCCLK[3]	OSC3	50MHz	GPUCLK, aux clock used to generate PERIPH_CLK for user space. Frequency can be changed in the board file an536_v1.txt
OSCCLK[4]	OSC4	24.576MHz	AUDCLK, clock used to clock I2S audio module. Frequency can be changed in the board file an536_v1.txt
OSCCLK[5]	OSC5	23.75MHz	HDLCDCLK, clock can be used to clock video module. MCC overrides this value. Frequency can be changed in the board file an536_v1.txt
c0_sys_clk_p/n	OSC6 (GTX Clock)	100MHz	DDR4_REF_CLK, Constant Differential input clock for DDR4 controller

Table 4-1 : Source clocks

4.2 User clocks

The following clocks are generated internally from the source clocks:

Clock	Source	Frequency	Note
CLK	ACLK	50MHz	Main system clock
PERIPH_CLK	GPUCLK	50MHz	AUX clock for user peripherals
MBMSMC_CLK	SMBM_CLK	Set by MCC	SMB clock from MCC
MCLK_u_apb_i2s_top	AUDCLK	12.29MHz	
SCLK_u_apb_i2s_top	AUDCLK	3.07MHz	
clk_100hz_u_fpga_io_regs	REFCLK24MHZ	100Hz	
CLK1HZ_u_rtc	REFCLK24MHZ	1Hz	
CFGCLK	CFG_CLK	Set by MCC	SCC register clock from MCC

Table 4-2 : Generated internal clocks

5 Interrupt Map

5.1 FPGA interrupt map

Internal to CR52 GIC used in this design as Interrupt controller with base address of 0xF000_0000.

The following table shows the interrupts from the FPGA subsystem wired up to SPI (active high) and PPI (active low) port of GIC :

Shared peripheral interrupts	Interrupt Source	External private peripheral interrupts	Interrupt Source
SPI [0]	WDG	EXTPPI0 [0]	UART 0 Receive Interrupt
SPI [1]	DualTimer 1	EXTPPI0 [1]	UART 0 Transmit Interrupt
SPI [2]	DualTimer 2	EXTPPI0 [2]	UART 0 Combined Interrupt
SPI [3]	DualTimer Combined	EXTPPI0 [3]	UART 0 Overflow
SPI [4]	RTC	EXTPPI1 [0]	UART 1 Receive Interrupt
SPI [5]	UART 2 Receive Interrupt	EXTPPI1 [1]	UART 1 Transmit Interrupt
SPI [6]	UART 2 Transmit Interrupt	EXTPPI1 [2]	UART 1 Combined Interrupt
SPI [7]	UART 3 Receive Interrupt	EXTPPI1 [3]	UART 1 Overflow
SPI [8]	UART 3 Transmit Interrupt		
SPI [9]	UART 4 Receive Interrupt		
SPI [10]	UART 4 Transmit Interrupt		
SPI [11]	UART 5 Receive Interrupt		
SPI [12]	UART 5 Transmit Interrupt		
SPI [13]	UART 2 Combined Interrupt		
SPI [14]	UART 3 Combined Interrupt		
SPI [15]	UART 4 Combined Interrupt		
SPI [16]	UART 5 Combined Interrupt		
SPI [17]	UART Overflow (2, 3, 4 & 5)		
SPI [18]	Ethernet		
SPI [19]	USB		
SPI [20]	FPGA Audio I2S		
SPI [21]	Touch Screen		
SPI [22]	SPI ADC		
SPI [23]	SPI (Shield 0)		
SPI [24]	SPI (Shield 1)		
SPI [25]	HDCLCD Interrupt		
SPI [26]	GPIO 0 Combined Interrupt		

SPI [27]	GPIO 1 Combined Interrupt
SPI [28]	GPIO 2 Combined Interrupt
SPI [29]	GPIO 3 Combined Interrupt
SPI [30:45]	GPIO 0 individual interrupts
SPI [46:61]	GPIO 1 individual interrupts
SPI [62:77]	GPIO 2 individual interrupts
SPI [78:93]	GPIO 3 individual interrupts

Table 3 : FPGA Expansion Interrupt Map.

5.2 UARTS Interrupts

There are six CMSDK UARTs in the system, and each has the following interrupt pins:

- TXINT
- RXINT
- TXOVRINT
- RXOVRINT
- UARTINT

The TXINT, RXINT and UARTINT interrupt signal of each UART drive a separate interrupt inputs of the Cortex-R52 CPU. In addition, The TXOVERINT and RXOVRINT interrupt signals of all UARTs, are logically ORed together which added one more interrupt signal per each UART.

6 Shield Support

This SMM support external shield devices. To enable the Shield support, two SPI, two UART and two I2C interfaces are multiplexed with GPIO over the Shield headers.

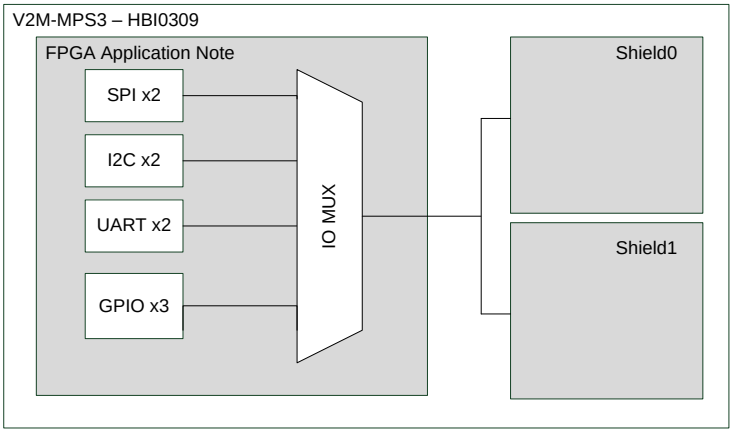


Figure 6-1 : Shield device expansion

Multiplexing is controlled by the alternative function output from the associated GPIO Register. The following table shows the Shield alternative function pinout:

MPS3	Proposed	Alt Function	Alt Peripheral	Alt Description
SH0_IO0	GPIO0_0	SH0_RXD	UART3	SH0 UART
SH0_IO1	GPIO0_1	SH0_TXD		
SH0_IO2	GPIO0_2			
SH0_IO3	GPIO0_3			
SH0_IO4	GPIO0_4			
SH0_IO5	GPIO0_5			
SH0_IO6	GPIO0_6			
SH0_IO7	GPIO0_7			
SH0_IO8	GPIO0_8			
SH0_IO9	GPIO0_9			
SH0_IO10	GPIO0_10	SH0_nCS		
SH0_IO11	GPIO0_11	SH0_DO	SPI3	SH0 SPI
SH0_IO12	GPIO0_12	SH0_DI		
SH0_IO13	GPIO0_13	SH0_CLK		
SH0_IO14	GPIO0_14	SH0_SDA	I2C2	SH0 I2C
SH0_IO15	GPIO0_15	SH0_SCL		
SH0_IO16	GPIO2_0			
SH0_IO17	GPIO2_1			

MPS3	Proposed	Alt Function	Alt Peripheral	Alt Description
SH1_IO0	GPIO1_0	SH1_RXD	UART4	SH1 UART
SH1_IO1	GPIO1_1	SH1_TXD		
SH1_IO2	GPIO1_2			
SH1_IO3	GPIO1_3			
SH1_IO4	GPIO1_4			
SH1_IO5	GPIO1_5			
SH1_IO6	GPIO1_6			
SH1_IO7	GPIO1_7			
SH1_IO8	GPIO1_8			
SH1_IO9	GPIO1_9			
SH1_IO10	GPIO1_10	SH1_nCS	SPI4	SH1 SPI
SH1_IO11	GPIO1_11	SH1_DO		
SH1_IO12	GPIO1_12	SH1_DI		
SH1_IO13	GPIO1_13	SH1_CLK		
SH1_IO14	GPIO1_14	SH1_SDA	I2C3	SH1 I2C
SH1_IO15	GPIO1_15	SH1_SCL		
SH1_IO16	GPIO2_2			
SH1_IO17	GPIO2_3			

Table 6-1 : Shield alternative function pinout

7 ZIP Bundle Description

7.1 Overall Structure

The accompanying `.zip` bundle contains:

- This Application Note Document.
- An example armDS Version 2021.1 software project, that can be run on the MPS3 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.
- END USER LICENCE AGREEMENT
- Release Notes file containing Bundle Directory Tree/Structure and Errata

7.2 Documentation

This Application Note Document, AN536, is in the `Docs/` folder of the bundle.

8 Board Revision And Support

8.1 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 :

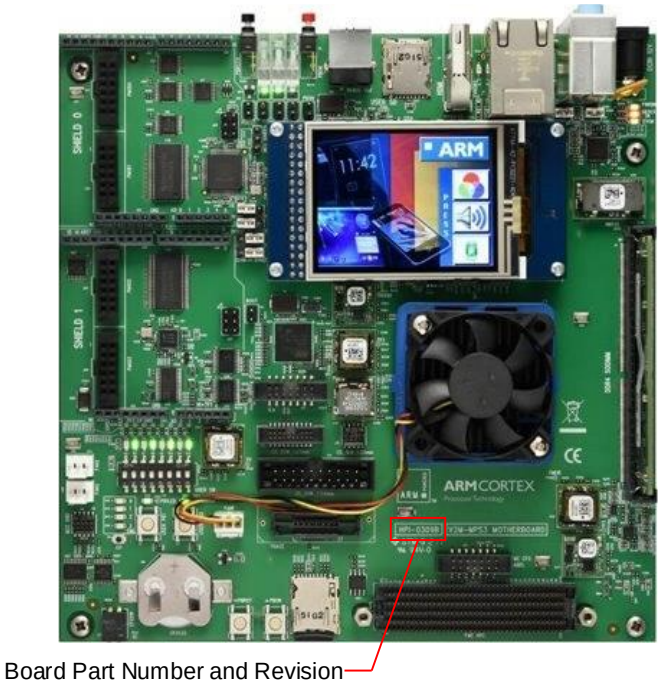


Figure 8-1 : MPS3 board revision identifier

In this example the part number is "HBI0309**B**". The last letter at the end of the part number denotes the board revision. The illustration shows a revision **B** board.

8.2 Bundle support for specific MPS3 board revisions

There are two subdirectories in the Boardfiles/MB/ directory that correspond to the two 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 8.1 for further details on how to identify the board part number and revision).

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 that the correct directory contents are modified if required.

9 Modifying and building AN536

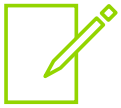
9.1 Partial reconfiguration

AN536 for MPS3 makes use of Xilinx's partial reconfiguration (PR) flow. With partial reconfiguration, specific design blocks can be allocated to a PR partition. These partitions can then be compiled to independent bitstreams. The PR bitstreams can be loaded to the FPGA to change the functionality of the FPGA within the PR design block.

In this flow, the mps3_fpga_user subsystem is designed as a PR partition and the contents of that partition can be modified by the user. The remaining functionality, (CR52x2 subsystem), is delivered as a pre-compiled encrypted bitstream and cannot be modified.

A Xilinx DCP file is provided to allow the users to compile their modified versions of the mps3_fpga_user subsystem. This is a preplaced design file containing all placement and routing for the enclosing top-level functionality which wraps around the mps3_fpga_user subsystem.

For further understanding of partial reconfiguration using the Xilinx PR flow the user is directed to the *Xilinx Vivado Design Suite User Guide 909* – Partial Reconfiguration.



With reference to the Xilinx Partial Reconfiguration terminology; "static image" aligns with the top level encrypted bitstream and Reconfigurable Module(RM) aligns with PR partition.

9.2 Pre-requisites

To build the AN536 FPGA, the user must have a licensed copy of Xilinx Vivado HLx Edition. Version 2019.1 has been used for this application note . The license must also support partial reconfiguration.

The Vivado executable must be included in the user's path.

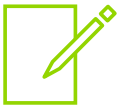
9.3 Flow overview

The files provided to the user consists of:

- Top level static DCP
- Encrypted bitstream containing the top level and CR52 x2 subsystem, (`536_t_X.bit`).
- Source files to build mps3_fpga_user

In overview, the flow consists of:

1. User synthesizes mps3_fpga_user into a DCP file.
2. The top level static DCP is combined with mps3_fpga_user DCP, and a stub DCP for the system core.
3. Place and route are then run.



Since the top level is preplaced and routed only the mps3_fpga_user partition is placed and routed.

4. PR bitfile is produced for the mps3_fpga_user PR partition.

   The following two files are produced for any PR partition:

   - `536_uc_X.bit :` The clearing bitstream to clear the appropriate part of FPGA configuration memory.
   - `536_u_X.bit` : The programming bitstream.

5. Top level static encrypted bitfile is downloaded to MPS3 board.
6. Two user PR partition bitfiles downloaded to MPS3 board.
7. CR52x2 subsystem boots.

9.4 Flow detail

The user partition code is in *<install_dir>/Socrates/CR52x2_1/logical*. The top-level file, *user_wrapper.v* is further located in the *wrapper* directory.

The following procedure describes how to build a new version of AN536:

1. Modify the code in the hierarchy under *user_wrapper.v* to include your new code. Note that the ports of *user_wrapper.v* itself must not be changed as this match the provided top level DCP. It is strongly recommended that the user add their code within one of the existing hierarchical layers rather than directly into *user_wrapper.v*
2. Navigate to *<install_dir>/FPGA/smm_toplevel/xilinx/scripts*
3. If different version numbers are required for the planned bitfiles, then edit *user_pr_impl.tcl* and set the variable *FPGA_BUILD* to the desired single digit number

The version number of the supplied files is 1. The default value of FPGA_BUILD set in the user scripts is 1. Therefore, in order to avoid any new bitfiles overwriting the pre compiled files it is suggested that the value of FPGA_BUILD is modified.

4. For a Linux system, execute:

   ```
   $ ./user_pr_flow.scr
   ```

   For a Windows system, execute:

   ```
   > user_pr_flow.bat
   ```

   from the Vivado HLS Command Prompt.

5. When the flow has completed, it will produce two bitfiles, `536_u_X.bit`, and `536_uc_X.bit`. These will be written to the `<install_dir>/Boardfiles/MB/HBI0309r/AN536` directory, where `r` – revision of the board. "`X`" will equate to the value of `FPGA_BUILD` written into `user_pr_impl.tcl`.

6. Copy the new bitfiles `536_u_X.bit` and `536_uc_X.bit` to the corresponding directory on the MPS3 board.
7. Copy the new bitfiles `536_u_X.bit`, and `536_uc_X.bit` to the directory `<MPS3_dir>MB/HBI0309C/AN536/` on the MPS3 board.
8. Edit the configuration file `an536_v1.txt` in the same directory to use the new files

```
F1FILE: 536_uc_1.bit   ;FPGA1 Filename - clear system PR – change this line
F1MODE: FPGA           ;FPGA1 Programming Mode
F2FILE: 536_u_1.bit    ;FPGA2 Filename - write system PR– change this line
F2MODE: FPGA           ;FPGA2 Programming Mode
```

Here 536_uc_1.bit and 536_u_1.bit are the files provided with the AN536 zip bundle.

9. Power on the MPS3 board. Check using either the debug UART or `log.txt` file that the new files were successfully programmed.

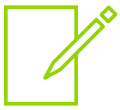
The MPS3 board is now programmed with the user code.

10 Using AN536 on the MPS3 board

10.1 Loading a prebuilt FPGA image onto the MPS3 board

The following procedure describes how to load the pre-built AN536 image:

1. Power up the MPS3 board using the PBON push button and wait for the V2M_MPS3 drive to appear.
2. Format the V2M_MPS3 drive and copy all the contents of `<install_dir>/Boardfiles` and paste them into the root directory of the attached V2M_MPS3 drive



You can manually modify and merge the contents for certain configuration files. Alternatively, you can restore the existing configuration files from the /Boardfiles directory. The affected configuration files are:

`<install_dir>/Boardfiles/config.txt`

`<install_dir>/Boardfiles/MB/HBI0309C/board.txt`

`<install_dir>/Boardfiles/MB/HBI0309C/AN536/images.txt`

3. Eject the V2M_MPS3 volume from your computer to unmount the drive.
4. Power cycle the MPS3 board using the PBRST push button and then launch firmware update and FPGA configuration by pressing PBON push button. The LEDs flash rapidly to indicate that new firmware is being downloaded (this only occurs the first time when the firmware is being updated) and that the prebuilt image is being downloaded onto the board. If you have configured the `images.txt` file, so that the MCC loads the selftest program and configured the UARTMODE to its default value of "0" in the `config.txt` file, the debug UART0 terminal simultaneously shows the selftest menu for Application Note AN536.
5. If the MPS3 board does not boot correctly, refer to the `log.txt` in the root directory of the MPS3 board which provides a log file of the files loaded at bootup.

10.2 UART serial ports

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

- Serial Port 0 is connected to the MCC and outputs verbose debug information about the status of the MCC.
- Serial Port 1 is connected to the UART 0.
- Serial Port 2 is connected to the UART 1.
- Serial Port 3 is connected to the UART 2.

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.

10.3 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
Data:	8 bits
Parity:	none
Stop:	1 bit
Flow control:	none

See the *Arm® MPS3 FPGA Prototyping Board Getting Started Guide* accompanying the MPS3 board and *Arm® MPS3 FPGA Prototyping Board Technical Reference Manual* for more information.

10.4 MPS3 USB serial port drivers for Windows

See the following information on installing drivers to support the USB serial port on MPS3:
https://community.arm.com/developer/tools-software/oss-platforms/w/docs/589/accessing-mps3-serial-ports-in-windows-10

11 Software

FPGA design has three boot option, depending on the type of memory you are using for booting.

It can be BRAM, TCM, or QSPI FLASH. Each case has its own settings for software build.

11.1 Rebuilding Software

Requirements

- The software directory from the download
- Arm Development Studio (armDS) v. 2020.1 or later

The following instructions apply to all software packages provided

- `File -> Import -> Existing Projects into Workspace`
- Navigate to `<install_dir>/Software/selftest_Cortex-R52/`
- Once loaded, choose project boot option (type of boot memory)
    - `Project - > Build Configuration -> Set Active -> <memory_type>`
- The project can be rebuilt by:
    - `Project - > Build All`
- The output can then be found in :

`<install_dir>/Software/selftest_Cortex-R52/<memory_type>/selftest_Cortex-R52_<memory_type>.axf`

11.2 Loading software to the MPS3 board

Requirements

- MPS3 board powered and USB cable connected
- MPS3 USB mass storage open in a file explorer

The following instructions apply to all versions of software

- Copying the software `<install_dir>/Software/ selftest_Cortex-R52/<memory_type>/ selftest_Cortex-R52_<memory_type>.axf` to the board `<MPS3_dir>/Software folder.` Rename the file to `CR52x2.axf`
- Navigate to `<MPS3_dir>MB/HBI0309r/AN536`, where *r* – revision of the board and update files (`image.txt` and `an536_v1.txt`) with the values (highlighted in bold) for fields mentioned below according to chosen boot memory (BRAM / TCM / QSPI flash) :

For BRAM:

- `images.txt` file:

```
IMAGE0ADDRESS:      0x01000000
IMAGE0UPDATE:       FORCE
```

- `an536_v1.txt` file:

```
SYSCON: 0x018       0x10000000
SYSCON: 0x01C       0x10000000
```

For TCM:

- `images.txt` file:

```
IMAGE0ADDRESS:      0x0C000000
IMAGE0UPDATE:       FORCE
```

- `an536_v1.txt` file:

```
SYSCON: 0x018       0x00000000
SYSCON: 0x01C       0x00000000
```

For QSPI FLASH:

- `images.txt` file:

```
IMAGE0ADDRESS:      0x01000000
IMAGE0UPDATE:       FORCEQSPI
```

- `an536_v1.txt` file:

```
SYSCON: 0x018       0x08000000
SYSCON: 0x01C       0x08000000
```

The MPS3 can now be booted up as per the instructions in the Arm Cortex Prototyping System (MPS3) - Getting Started Guide accompanying the MPS3 board.

12 Debug

Debug solution of this SMM is based on Arm CoreSight SoC-400 System and provides user with debug and trace capability. Arm Development Studio (armDS) with Dstream debugger can be used to simplify debug and trace access into system.

12.1 Debug Connectivity

The following table shows the supported connectivity between the supported MPS3 Board debug connectors (See Figure 12-1 : MPS3 Board debug connectors for locating the connectors on board) 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	Yes	Yes	Yes	No
20 pin IDC	Yes	Yes	No	No
Mictor 38	Yes	Yes	Yes	Yes

Table 12-1 : Debug Connectivity and Support

12.2 Debug and Trace support for Arm Development Studio

In Arm Development studio 2020.1 or above debug configurations can be found under Run>Debug Configurations.

12.2.1 Establishing a Debug Session

Following steps needs to be carried out to establish a debug connection :

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

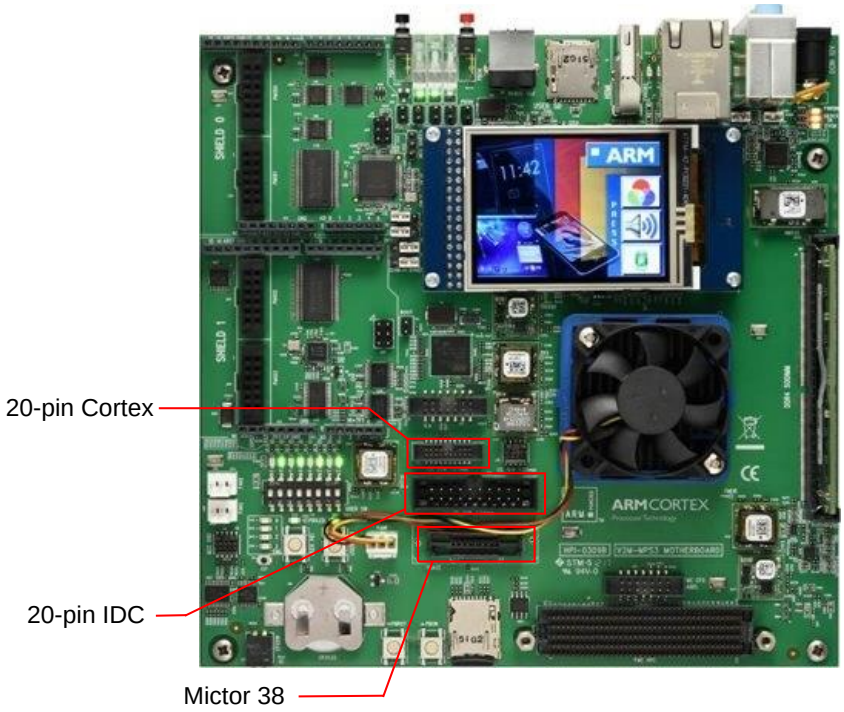
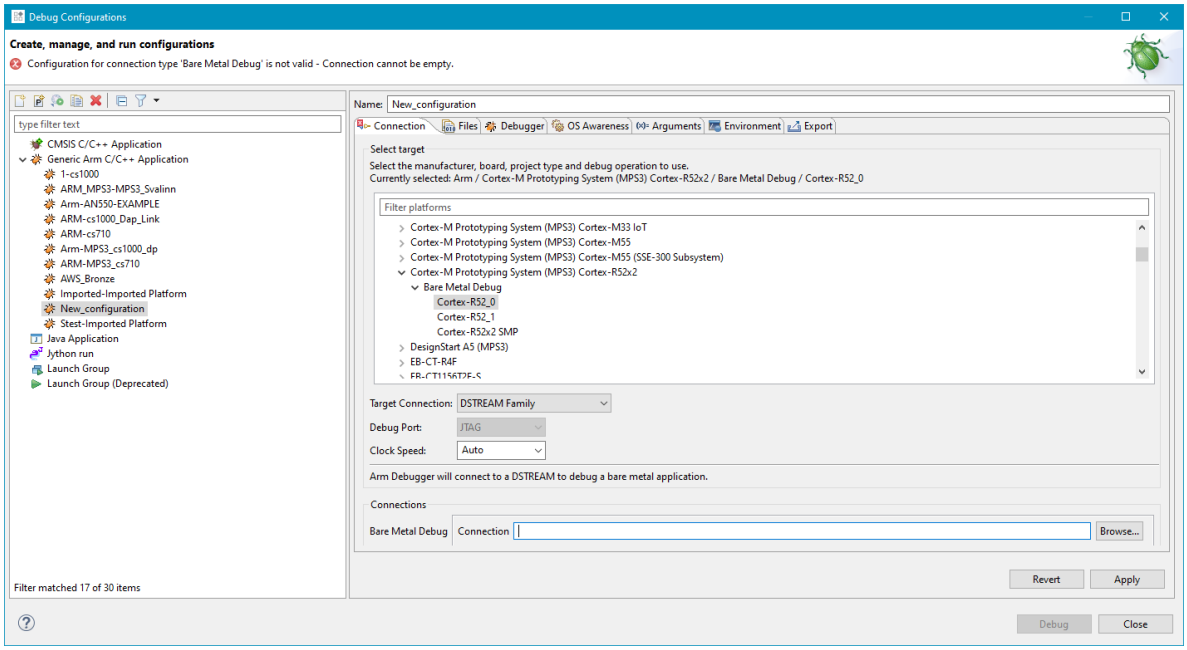
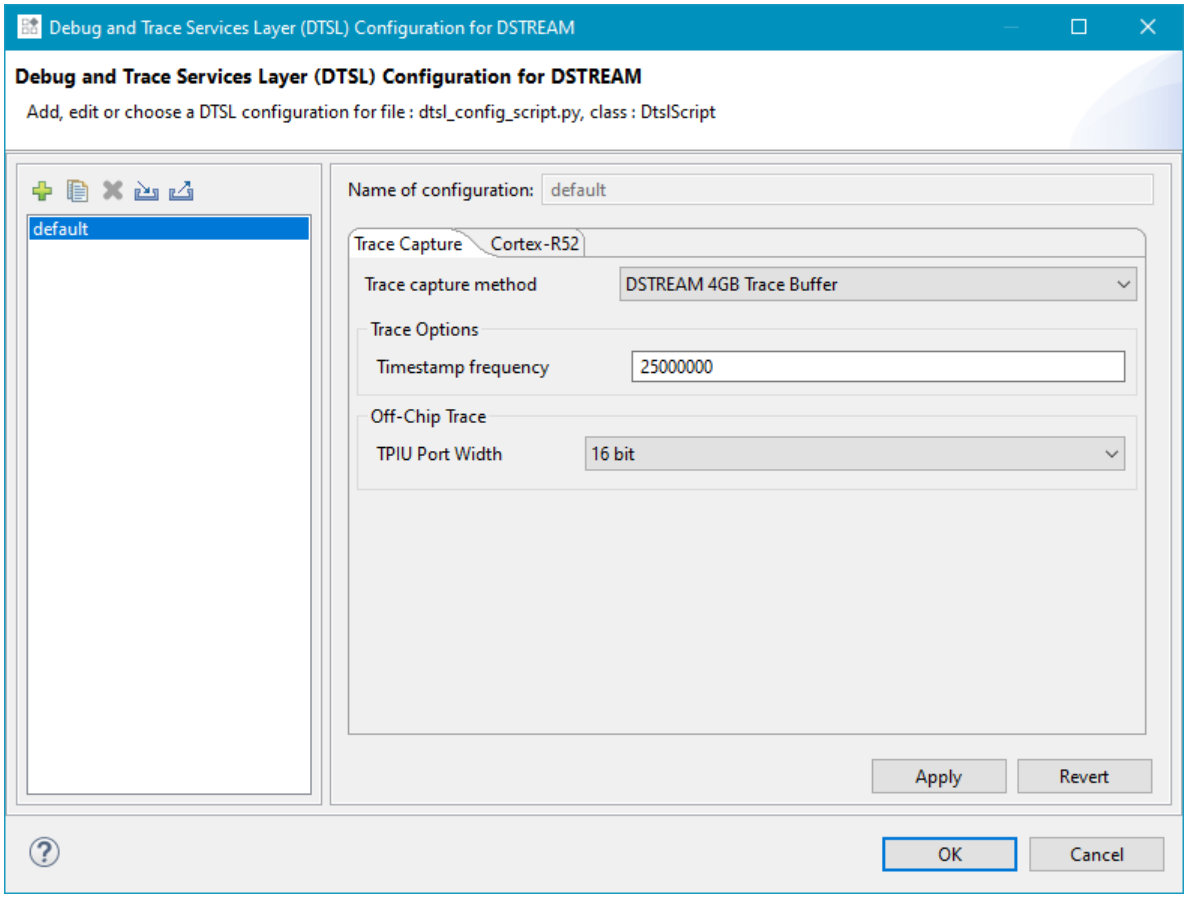


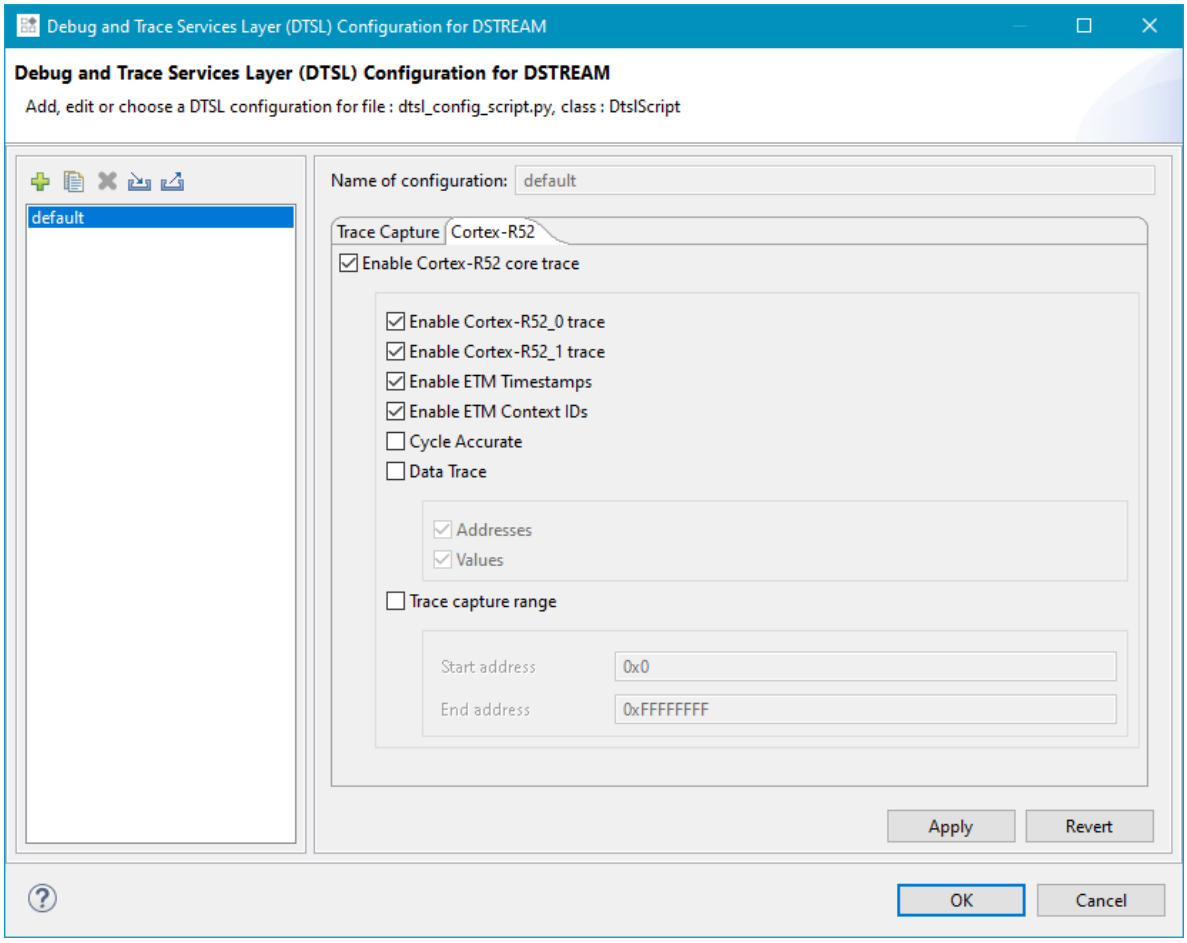
Figure 12-1 : MPS3 Board debug connectors

2. Select Arm Cortex-M Prototyping System (MPS3) Cortex-R52x2 > Bare Metal Debug > Cortex – R52_0.



3. Enable trace capture by editing the DTSL options as below:





4. Click Ok, Apply and then Click on Debug. Trace should now be visible in the Trace window.

13 Known Limitations

Design does not support Exclusive Access (LDREX/STREX) to sharable non-cacheable memory like BRAM or DDR4.