



# **DesignWare ARC AXC001 CPU Card User Guide**

---

Version 6300-007 October 2014

## Copyright Notice and Proprietary Information Notice

Copyright © 2014 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at

<http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

Synopsys, Inc.  
700 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)

# Contents

List of Figures .....	5
List of Tables.....	7
1 Package Contents.....	9
1.1 DesignWare ARC AXS101 Software Development Platform .....	9
1.2 DesignWare ARC AXC001 CPU Card (stand-alone) .....	10
2 Getting Started.....	11
3 Default Board Settings .....	13
3.1 Default Jumper Settings on the AXC001 CPU Card .....	13
3.2 Default Boot Mode Settings on the ARC SDP Mainboard .....	13
4 Self-Test.....	15
4.1 Self-Test Overview .....	15
4.2 Executing a Self-Test of the ARC 770D Core .....	17
4.3 Executing a Self-Test of the AS221 Core #1 .....	19
4.4 Executing a Self-Test of the AS221 Core #2 .....	21
4.5 Executing a Self-Test of the ARC EM6 Core.....	23
4.6 Restoring the Self-Tests in the SPI Flash.....	25
5 Hardware Functional Description .....	26
5.1 Board Overview.....	26
5.2 Board Interface Overview .....	28
5.2.1 Power Supply Connector .....	28
5.2.2 HapsTrak-II Connectors (bottom).....	28
5.2.3 HapsTrak-II Connectors (top).....	29
5.2.4 GPIO Outputs.....	29
5.3 Jumpers .....	30
5.4 AXC001 Processor IC Overview .....	31
5.4.1 Main Features of the ARC Cores .....	31
5.4.2 Interrupt.....	32
5.4.3 Clock .....	40
5.4.4 Reset.....	41
5.4.5 Debug .....	42
5.4.6 Control Registers .....	43
5.4.7 GPIO Registers .....	45
5.5 Memories on the AXC001 CPU Card .....	47
5.6 Power Supply .....	47
5.7 Audio Support .....	49
5.7.1 Main Features of the Audio Subsystem .....	49
5.7.2 Stereo Input .....	50
5.7.3 Stereo Output.....	50
5.7.4 8-Channel Audio Output.....	51
5.7.5 S/PDIF Input .....	51
5.7.6 S/PDIF Output.....	51
5.7.7 Audio PLL .....	51
5.8 Usage of ARC SDP Mainboard Resources .....	52
5.8.1 Usage of the Mainboard DIP Switches.....	52

5.8.2	Usage of the Mainboard Push Buttons.....	57
5.8.3	Usage of the Mainboard LEDs .....	59
6	System Memory Map .....	60
6.1	System Memory Map after a Reset.....	60
6.2	System Memory Map after Pre-Bootloader Execution.....	61
6.3	Controlling the Memory Map .....	63
6.3.1	Setting up the AXI Masters on the AXC001 CPU Card .....	63
6.3.2	Setting up the AXI Masters on the ARC SDP Mainboard.....	64
6.3.3	Example Register Settings for the Default Memory Map.....	65
6.4	Memory Map of the Local Peripherals.....	67
7	Programmer's Reference .....	68
7.1	Supported Tools and Operating Systems.....	68
7.2	Boot Modes .....	68
7.2.1	Common Boot Modes .....	68
7.2.2	ARC EM6 Booting from ICCM.....	69
7.3	Pre-Boot.....	70
7.3.1	Pre-Boot Overview .....	70
7.4	U-Boot.....	74
7.4.1	Programming U-Boot in the SPI Flash .....	76
7.4.2	Starting U-Boot on the ARC 770D Core.....	76
7.4.3	Other Supported U-Boot Features .....	77
7.5	Drivers.....	78
7.5.1	Drivers for Baremetal Applications.....	78
7.5.2	Drivers for MQX .....	78
7.5.3	Drivers for Linux.....	78
7.6	Baremetal Package.....	79
7.6.1	Overview .....	79
7.6.2	Building Baremetal Applications Using the MetaWare IDE .....	80
7.6.3	Building Baremetal Applications Using gmake.....	83
7.6.4	Hardware Setup for Debugging.....	85
7.6.5	Running a Baremetal Application in the MetaWare IDE Debugger .....	88
7.6.6	Running a Baremetal Application in the MetaWare Debugger .....	91
7.6.7	Storing an Image in the SPI Flash and Running the Application.....	95
7.6.8	Storing an Image on the SD-Card and Running the Application .....	98
7.7	MQX Package.....	99
7.7.1	Overview .....	99
7.7.2	Building MQX Applications Using the MetaWare IDE .....	99
7.7.3	Hardware Setup for Debugging.....	102
7.7.4	Running an MQX Application in the MetaWare IDE Debugger .....	105
7.8	Linux Package.....	108
7.8.1	Overview .....	108
7.8.2	Loading and Executing the Linux Image .....	108
7.8.3	Building Applications .....	108
7.8.4	Linux Application Examples .....	109
7.8.5	Updating the Linux Image .....	109
8	Software Interfaces.....	110

8.1	ARC EM6 Address Decoder Registers.....	110
8.1.1	CREG_EM6_A_SLV0: ARC EM6 Slave Select Register 0 .....	110
8.1.2	CREG_EM6_A_SLV1: ARC EM6 Slave Select Register 1 .....	111
8.1.3	CREG_EM6_A_OFFSET0: ARC EM6 Address Offset Register 0 .....	112
8.1.4	CREG_EM6_A_OFFSET1: ARC EM6 Address Offset Register 1 .....	113
8.1.5	CREG_EM6_A_BOOT: ARC EM6 Boot Mirror Register .....	114
8.1.6	CREG_EM6_A_UPDATE: ARC EM6 Update Register .....	115
8.2	ARC 770D Address Decoder Registers .....	116
8.2.1	CREG_770_A_SLV0: ARC 770D Slave Select Register 0 .....	116
8.2.2	CREG_770_A_SLV1: ARC 770D Slave Select Register 1 .....	117
8.2.3	CREG_770_A_OFFSET0: ARC 770D Address Offset Register 0 .....	118
8.2.4	CREG_770_A_OFFSET1: ARC 770D Address Offset Register 1 .....	119
8.2.5	CREG_770_A_BOOT: ARC 770D Boot Mirror Register .....	120
8.2.6	CREG_770_A_UPDATE: ARC 770D Update Register .....	121
8.3	ARC AS221 Address Decoder Registers .....	122
8.3.1	CREG_221_A_SLV0: ARC 221 Slave Select Register 0 .....	122
8.3.2	CREG_221_A_SLV1: ARC 221 Slave Select Register 1 .....	123
8.3.3	CREG_221_A_OFFSET0: ARC 221 Address Offset Register 0 .....	124
8.3.4	CREG_221_A_OFFSET1: ARC 221 Address Offset Register 1 .....	125
8.3.5	CREG_221_A_BOOT: ARC 221 Boot Mirror Register .....	126
8.3.6	CREG_221_A_UPDATE: ARC 221 Update Register .....	127
8.4	AXI Tunnel Address Decoder Registers .....	128
8.4.1	CREG_TUN_A_SLV0: AXI Tunnel Slave Select Register 0 .....	128
8.4.2	CREG_TUN_A_SLV1: AXI Tunnel Slave Select Register 1 .....	129
8.4.3	CREG_TUN_A_OFFSET0: AXI Tunnel Address Offset Register 0 .....	130
8.4.4	CREG_TUN_A_OFFSET1: AXI Tunnel Address Offset Register 1 .....	131
8.4.5	CREG_TUN_A_UPDATE: AXI Tunnel Update Register .....	132
8.5	ARC Start Registers .....	133
8.5.1	CREG_EM6_START: ARC EM6 Start Register .....	133
8.5.2	CREG_770_START: ARC 770D Start Register .....	134
8.5.3	CREG_221_START: ARC AS221 Start Register .....	135
8.6	ICTL Registers .....	136
8.6.1	INT_STATUS: Interrupt Status Register .....	136
8.7	GPIO Registers .....	137
8.7.1	SWPORTA_DR: GPIO Port A Output Register .....	137
8.7.2	EXT_PORTA: GPIO Port A Input Register .....	138
9	Glossary and References .....	139
9.1	Glossary .....	139
9.2	References .....	140
	Appendix A .....	141
	A.1 Mounting AXC001 CPU Card on ARC SDP Mainboard .....	141
	Appendix B .....	143
	B.1 Installing and Configuring PuTTY .....	143

## List of Figures

Figure 1	DesignWare ARC AXS101 Software Development Platform .....	9
Figure 2	DesignWare ARC AXC001 CPU Card .....	10
Figure 3	Location of the ARC SDP Mainboard's power supply and power switch .....	11
Figure 4	ARC SDP Mainboard status LEDs after power-on .....	12
Figure 5	Default Jumper Settings on the AXC001 CPU Card .....	13
Figure 6	Default settings of the DIP switches on the ARC SDP Mainboard. ....	14
Figure 7	Location of the ARC SDP Mainboard's power supply and power switch .....	17
Figure 8	Location of the CPU Start button SW2506 for the ARC 770D core .....	17
Figure 9	Screenshot of ARC 770D self-test .....	18
Figure 10	Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard .....	18
Figure 11	Location of the RESET button on the ARC SDP Mainboard .....	18
Figure 12	Location of the ARC SDP Mainboard's power supply and power switch .....	19
Figure 13	Location of the CPU Start button SW2505 for the AS221 core #1 .....	19
Figure 14	Screenshot of the AS221 core #1 self-test .....	20
Figure 15	Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard .....	20
Figure 16	Location of the RESET button on the ARC SDP Mainboard .....	20
Figure 17	Location of the ARC SDP Mainboard's power supply and power switch .....	21
Figure 18	Location of the CPU Start button SW2507 for the AS221 core #2 .....	21
Figure 19	Screenshot of the AS221 core #2 self-test .....	22
Figure 20	Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard .....	22
Figure 21	Location of the RESET button on the ARC SDP Mainboard .....	22
Figure 22	Location of the ARC SDP Mainboard's power supply and power switch .....	23
Figure 23	Location of the CPU Start button SW2504 for the ARC EM6 core .....	23
Figure 24	Screenshot of ARC EM6 self-test .....	24
Figure 25	Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard .....	24
Figure 26	Location of the RESET button on the ARC SDP Mainboard .....	24
Figure 27	Hardware block diagram .....	27
Figure 28	Default Jumper Settings .....	30
Figure 29	Interrupt architecture .....	33
Figure 30	Clock architecture .....	41
Figure 31	Location of the RESET button on the ARC SDP Mainboard .....	42
Figure 32	JTAG daisy-chain .....	42
Figure 33	Pinout diagram of the Power Supply Connector (bottom view) .....	48
Figure 34	Hardware block diagram of the audio subsystem .....	50
Figure 35	Functions and default settings of the DIP switches on the ARC SDP Mainboard .....	56
Figure 36	Location of the CPU Start buttons on the ARC SDP Mainboard. ....	58
Figure 37	Location of the CPU LEDs on the ARC SDP Mainboard .....	59
Figure 38	Default settings of the DIP switches on the ARC SDP Mainboard. ....	71
Figure 39	Pre-Boot mechanism .....	72
Figure 40	DIP switch settings for auto-starting U-Boot .....	75
Figure 41	DIP switch settings for manually starting U-Boot by pushing SW2506 .....	75
Figure 42	Location of the RESET button on the Mainboard .....	77
Figure 43	MetaWare IDE - Select Workspace Directory .....	80
Figure 44	MetaWare IDE – Importing existing projects .....	80
Figure 45	MetaWare IDE - Set Active Build Configurations .....	81
Figure 46	MetaWare IDE – build results in console window .....	82

Figure 47	Build script options .....	83
Figure 48	Settings of the DIP switches on the ARC SDP Mainboard for using the debugger.....	85
Figure 49	Location of the debug interfaces and the corresponding jumpers .....	86
Figure 50	Location of the ARC SDP Mainboard's power supply and power switch .....	86
Figure 51	Location of the CPU Start buttons on the ARC SDP Mainboard. ....	87
Figure 52	MetaWare IDE – Selecting the Debug Configuration (down arrow next to bug icon).....	88
Figure 53	MetaWare IDE – Setting up the debug configuration .....	89
Figure 54	MetaWare IDE – Selecting the debugger target .....	89
Figure 55	MetaWare IDE – Debugger command line options .....	90
Figure 56	Creating a new process.....	91
Figure 57	Debugger options – command-line options .....	92
Figure 58	Debugger options – target selection .....	93
Figure 59	Specifying a path to the .elf file.....	93
Figure 60	Debugger status.....	94
Figure 61	HyperTerminal output.....	94
Figure 62	DIP switch settings for autonomous code execution on the ARC 770D .....	97
Figure 63	MetaWare IDE - Set Active Build Configurations for MQX.....	100
Figure 64	Settings of the DIP switches on the ARC SDP Mainboard for using the debugger.....	102
Figure 65	Location of the debug interfaces and the corresponding jumpers .....	103
Figure 66	Location of the ARC SDP Mainboard's power supply and power switch .....	103
Figure 67	Location of the CPU Start buttons on the ARC SDP Mainboard. ....	104
Figure 68	MetaWare IDE – Selecting the Debug Configuration (down arrow next to bug icon).....	105
Figure 69	MetaWare IDE – Setting up the debug configuration .....	106
Figure 70	MetaWare IDE – Selecting the debugger target .....	106
Figure 71	MetaWare IDE – Debugger command line options .....	107
Figure 72	Alignment of the Power Supply Connector .....	141
Figure 73	Default settings of the DIP switches on the ARC SDP Mainboard .....	142
Figure 74	Identification of COM port.....	143
Figure 75	PuTTY configuration.....	144

## List of Tables

Table 1	Self-test start buttons .....	15
Table 2	Characters on the seven-segment display during the self-test.....	16
Table 3	Jumper functionality .....	30
Table 4	Main features of the ARC cores.....	31
Table 5	Interrupt mapping for the ARC 770D core .....	34
Table 6	Interrupt mapping for ARC AS221 (core #1).....	35
Table 7	Interrupt mapping for ARC AS221 (core #2).....	37
Table 8	Interrupt mapping for the ARC EM6 core.....	38
Table 9	Mainboard ICTL Interrupt mapping.....	39
Table 10	JTAG ID codes.....	42
Table 11	Control register memory map .....	43
Table 12	GPIO register memory map.....	45
Table 13	GPIO port A output register function (SWPORTA_DR) .....	45
Table 14	GPIO port A input register function (EXT_PORTA).....	46
Table 15	Pin description of the Power Supply Connector.....	47
Table 16	Audio Subsystem main features .....	49
Table 17	ARC EM6 boot configuration (Mainboard DIP switch SW2501).....	53
Table 18	ARC 770D boot configuration (Mainboard DIP switch SW2502).....	54
Table 19	AS221 boot configuration (Mainboard DIP switch SW2503).....	55
Table 20	Pre-Bootloader usage of Mainboard DIP switch SW2401 .....	56
Table 21	Usage of the CPU Start buttons of the ARC SDP Mainboard .....	57
Table 22	Control bits of the CPU LEDs on the ARC SDP Mainboard .....	59
Table 23	Memory map after Pre-Bootloader execution .....	61
Table 24	AXC001 CPU Card target slaves.....	63
Table 25	ARC SDP Mainboard target slaves .....	64
Table 26	Default memory map programming for all master on the AXC001 CPU Card .....	65
Table 27	Default memory map programming for all masters on the ARC SDP Mainboard .....	66
Table 28	Peripheral memory map .....	67
Table 29	Meaning of the left character of the seven-segment display .....	72
Table 30	Meaning of the right character of the seven-segment display .....	73
Table 31	Baremetal folder contents.....	79
Table 32	Build options.....	81
Table 33	Command line options for build.bat .....	83
Table 34	CPU Start buttons and seven-segment display values for running applications in the debugger ..	87
Table 35	Selecting the CPU core in the debugger.....	90
Table 36	Selecting the CPU core in the debugger.....	92
Table 37	MQX folder contents.....	99
Table 38	Description of the build configurations .....	100
Table 39	CPU Start buttons and seven-segment display values for running applications in the debugger	104
Table 40	Selecting the CPU core in the debugger.....	107
Table 41	CREG_EM6_A_SLV0 register.....	110
Table 42	CREG_EM6_A_SLV1 register.....	111
Table 43	CREG_EM6_A_OFFSET0 register .....	112
Table 44	CREG_EM6_A_OFFSET1 register .....	113
Table 45	CREG_EM6_A_BOOT register .....	114
Table 46	CREG_EM6_A_UPDATE register .....	115



Table 47	CREG_770_A_SLV0 register .....	116
Table 48	CREG_770_A_SLV1 register .....	117
Table 49	CREG_770_A_OFFSET0 register.....	118
Table 50	CREG_770_A_OFFSET1 register.....	119
Table 51	CREG_770_A_BOOT register.....	120
Table 52	CREG_770_A_UPDATE register .....	121
Table 53	CREG_221_A_SLV0 register .....	122
Table 54	CREG_221_A_SLV1 register .....	123
Table 55	CREG_221_A_OFFSET0 register.....	124
Table 56	CREG_221_A_OFFSET1 register.....	125
Table 57	CREG_221_A_BOOT register.....	126
Table 58	CREG_221_A_UPDATE register .....	127
Table 59	CREG_TUN_A_SLV0 register.....	128
Table 60	CREG_TUN_A_SLV1 register.....	129
Table 61	CREG_TUN_A_OFFSET0 register .....	130
Table 62	CREG_TUN_A_OFFSET1 register .....	131
Table 63	CREG_TUN_A_UPDATE register .....	132
Table 64	CREG_EM6_START register .....	133
Table 65	CREG_770_START register .....	134
Table 66	CREG_221_START register .....	135
Table 67	GPIO port A output register (SWPORTA_DR).....	136
Table 68	GPIO port A output register (SWPORTA_DR).....	137
Table 69	GPIO port A input register (EXT_PORTA).....	138

## 1.1 DesignWare ARC AXS101 Software Development Platform

The DesignWare ARC AXS101 Software Development Platform package contains the following items:

- DesignWare ARC AXC001 CPU Card mounted on ARC SDP Mainboard
- 100-240V AC power adapter (including power cables for U.S., UK and EU outlets)
- USB cable
- Pen-sized plastic DIPSTICK for actuating DIP switches

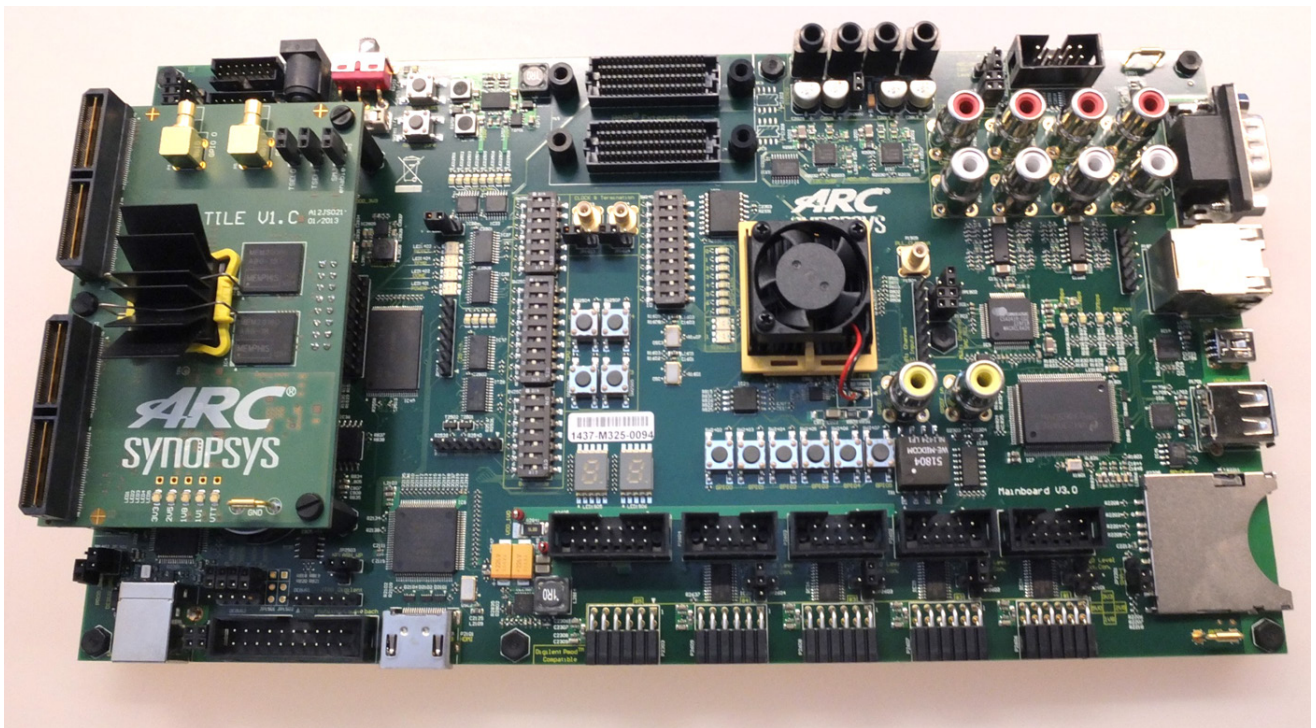


Figure 1 DesignWare ARC AXS101 Software Development Platform



### Warning

The AXC001 CPU Card and the ARC SDP Mainboard contain static sensitive devices.

## 1.2 DesignWare ARC AXC001 CPU Card (stand-alone)

The DesignWare ARC AXC001 CPU Card package contains the following item:

- DesignWare ARC AXC001 CPU Card printed circuit board

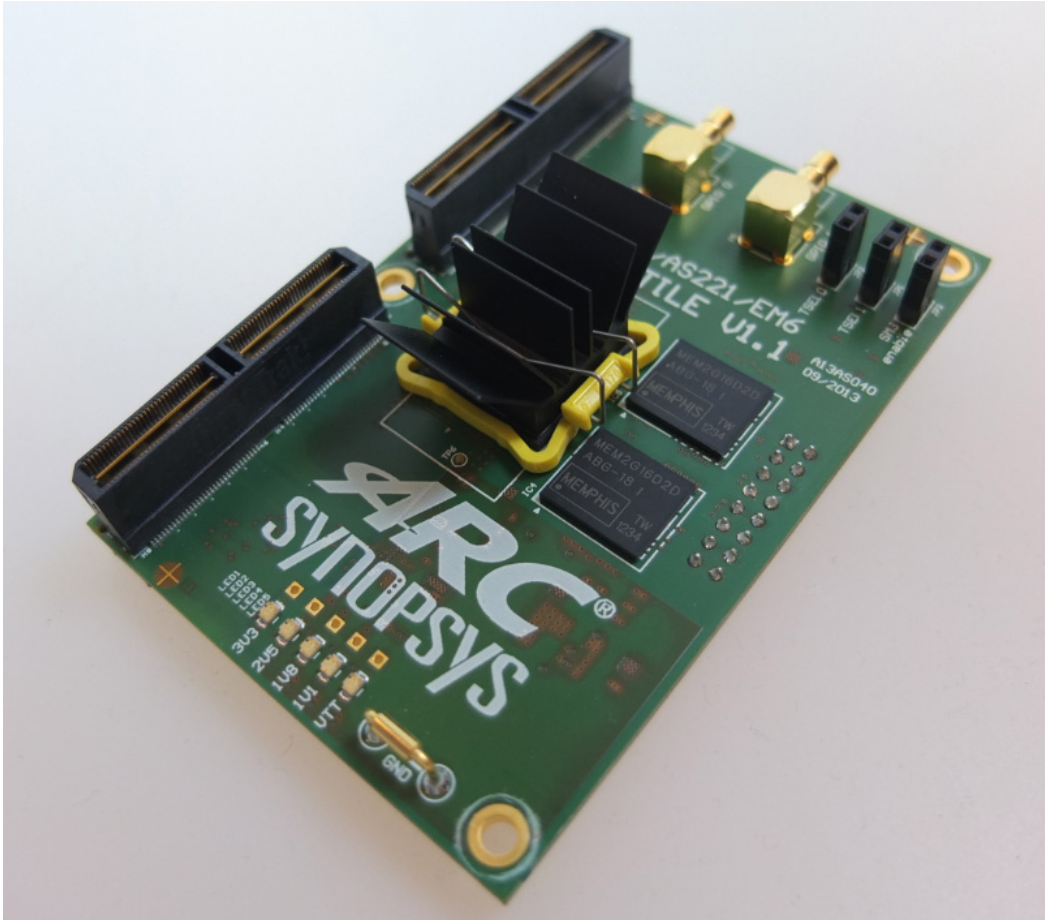


Figure 2 DesignWare ARC AXC001 CPU Card



### Warning

The AXC001 CPU Card contains static sensitive devices.

# 2

## Getting Started

*This chapter contains a step-by-step guide for installing the software package for the AXS101 Software Development Platform, connecting the ARC SDP Mainboard and performing a self-test.*

If you have purchased a stand-alone AXC001 CPU Card, refer to the instructions in “[Appendix A](#)” for mounting the CPU Card on the ARC SDP Mainboard, and thus, obtaining a complete AXS101 Software Development Platform.

Follow the steps below to get the AXS101 Software Development Platform up and running and perform a self-test.

1. Download and unzip the `axs101_software_<version>.zip` file from the ARC SDP download webpage [6].
2. Install the USB-JTAG and USB-UART drivers (Digilent Adept tool) according to the instructions provided in the ARC SDP Mainboard User Guide [7].
3. Connect the ARC SDP Mainboard to your PC using the USB cable, which shall be connected to the USB Dataport of the Mainboard.
4. Connect the power supply to the ARC SDP Mainboard and switch ON the Mainboard.

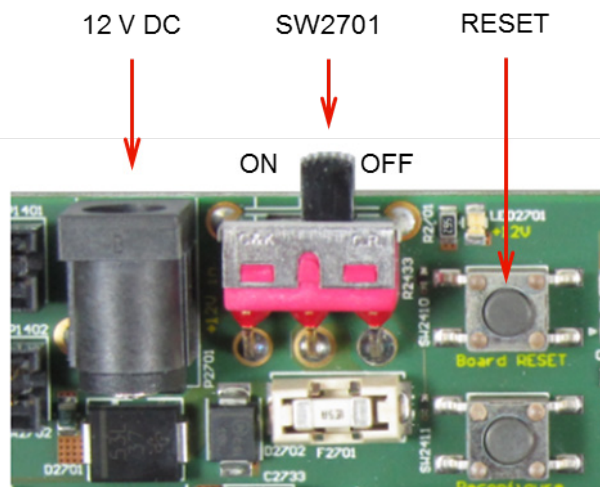


Figure 3 Location of the ARC SDP Mainboard’s power supply and power switch

5. Install PuTTY on your computer as described in “[Appendix B](#)”.
6. The FPGA on the ARC SDP Mainboard is now configured automatically and the Mainboard executes the reset sequence. The status LEDs `DONE`, `RESET`, `TUNNEL0` and `TUNNEL1` on the Mainboard shine red during startup. Wait until all status LEDs except `TUNNEL1` shine green. This may take several seconds. The LED `TUNNEL1` will continue to shine red.

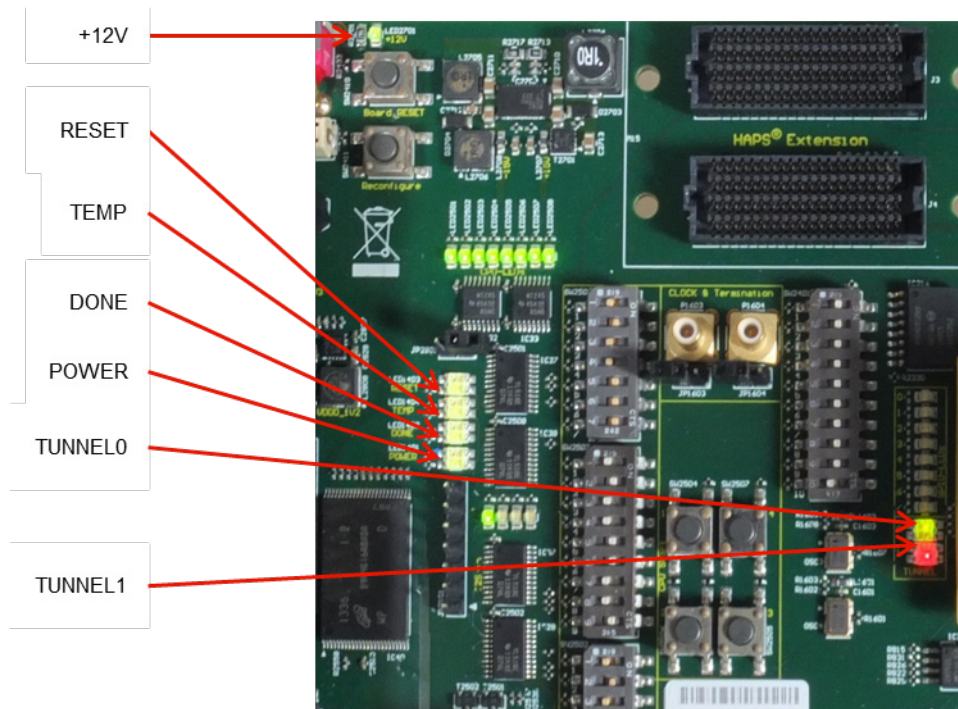


Figure 4 ARC SDP Mainboard status LEDs after power-on

7. Check that the five LEDs on the AXC001 CPU Card all are ON, shining green.
8. Perform the self-test for one or multiple CPU cores of your choice as described in the “[Self-Test](#)” section.

For next steps, please refer to the sections “[Baremetal Package](#)”, “[MQX Package](#)” and “[Linux Package](#)”.

## Default Board Settings

This section describes the factory default settings of the jumpers on the AXC001 CPU Card and the default boot mode settings for the cores on the AXC001 CPU Card, which can be selected by DIP switches on the ARC SDP Mainboard.

### 3.1 Default Jumper Settings on the AXC001 CPU Card

The jumpers on the AXC001 CPU Card must be set according to Figure 5.

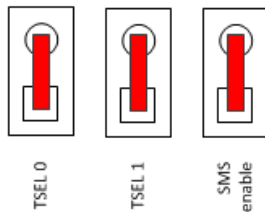


Figure 5 Default Jumper Settings on the AXC001 CPU Card

### 3.2 Default Boot Mode Settings on the ARC SDP Mainboard

The DIP switches on the ARC SDP Mainboard are set according to [Figure 6](#).

All cores are configured to boot via the AXI tunnel and to delay the start of the code execution until the corresponding `CPU Start` button on the ARC SDP Mainboard has been pushed.

The ARC EM6 core starts at address offset 0, the ARC 770D starts with an offset of 2 KByte and the AS221 with an offset of 4 KByte.

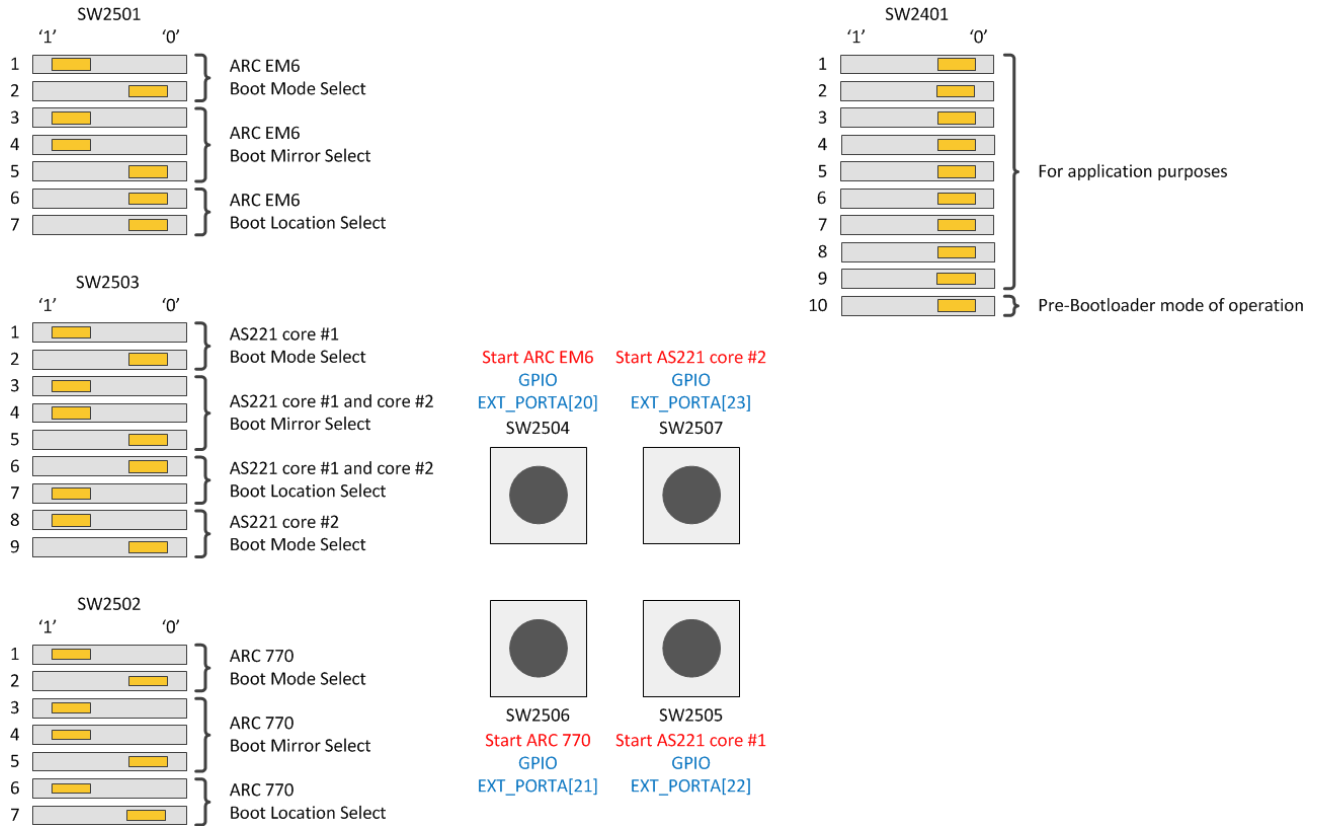


Figure 6 Default settings of the DIP switches on the ARC SDP Mainboard.

# 4 Self-Test

This chapter provides an overview of the self-tests and includes detailed instructions for executing the self-test on each individual CPU core. The expected behavior during the self-test is described as well.

## 4.1 Self-Test Overview

At the time of shipment, the SPI Flash on the ARC SDP Mainboard contains a self-test for each CPU core.

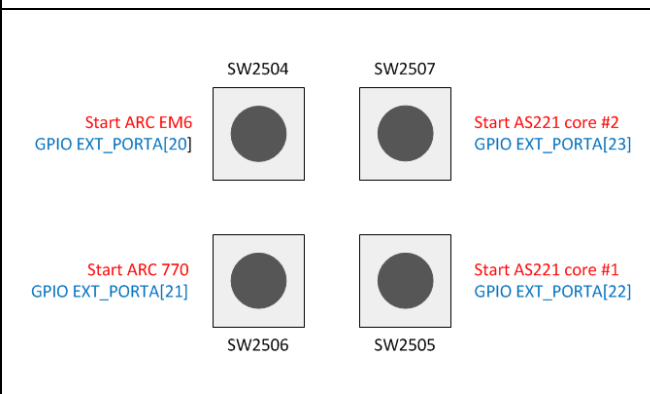
Descriptions in this section are based on the following assumptions:

- The SPI Flash contains self-tests
- The DIP switches on the ARC SDP Mainboard and on the AXC001 CPU Card are set as described in the “[Default Board Settings](#)” section above.
- The steps described in the “[Getting Started](#)” section have been performed.

 **Note** If you have programmed other applications in the SPI Flash, you can restore the self-test as described in the “[Restoring the Self-Tests in the SPI Flash](#)” section below.

The self-test for a particular CPU core is started by pushing the corresponding CPU Start button listed in [Table 1](#).

Table 1 Self-test start buttons

CPU Core	CPU Start Button	Location
ARC 770D	SW2506	
AS221 core #1	SW2505	
AS221 core #2	SW2507	
ARC EM6	SW2504	



The self-test accesses the peripherals of the peripheral subsystem that is implemented in the FPGA on the ARC SDP Mainboard. It displays information on the bit-file version and the available peripherals at a debug console on the PC. Additionally, the current CPU core speed is measured and displayed at the debug console. For a quick start, you may use a hyperterminal, such as PuTTY, as a debug console (see “[Getting Started](#)” above).

Next, the self-test enters an infinite loop, which creates a walking pattern at the CPU-LEDs and the GPIO LEDs on the ARC SDP Mainboard as follows:

The CPU-LEDs (LED2501, LED2502, LED2503, LED2504, LED2505, LED2506, LED2507 and LED2508) display a walking pattern with one of the LEDs switched OFF).

The eight GPIO LEDs show a walking pattern with one of the LEDs switched ON.

The two seven-segment displays on the ARC SDP Mainboard show the characters listed in [Table 2](#) depending on the ARC core that is currently running.

*Table 2 Characters on the seven-segment display during the self-test*

<b>ARC Core</b>	<b>Characters</b>
ARC 770D	40
AS221 core #1	10
AS221 core #2	20
ARC EM6	30

The expected results for each test are provided in the subsequent sections.

Push the RESET button to stop the self-test execution.

## 4.2 Executing a Self-Test of the ARC 770D Core

Follow the steps described below to perform a self-test of the ARC 770D core:

1. Connect the ARC SDP Mainboard to your PC using the USB cable, which shall be connected to the USB Dataport of the Mainboard.
2. Connect the power supply to the Mainboard and switch ON the Mainboard or press the RESET button.

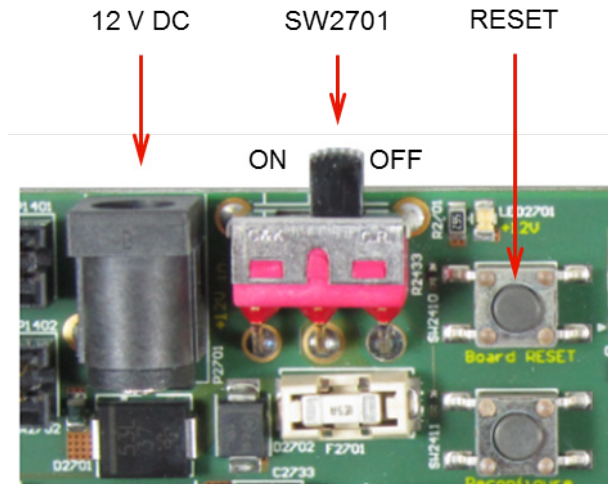


Figure 7 Location of the ARC SDP Mainboard's power supply and power switch

3. Launch PuTTY on your computer
4. Push the CPU Start button SW2506 on the ARC SDP Mainboard.

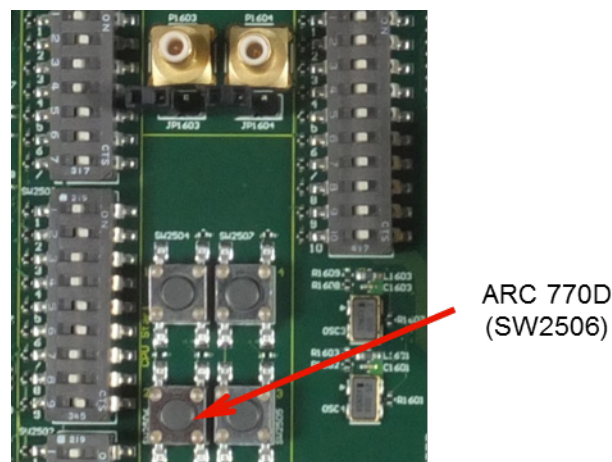


Figure 8 Location of the CPU Start button SW2506 for the ARC 770D core

5. Check that the seven-segment displays shows "40"

6. Observe the output on the console, which should be similar to the screenshot shown in [Figure 9](#).

```

COM12 - PuTTY
** selftest - Application **
Feb 12 2014
23:08:50
CORE:
  ARC770
FPGA-Version:
  Date: 7-2-2014 Time: 00:11
CPLD-Version:
  Date: 4-2-2014 Time: 19:08
PLL freq:
  Fsys: 750
  Fddr: 533
Core freq:
  Fcore: 750
Tunnel freq:
  Ftunnel-tx: 75
  Ftunnel-rx: 75
application finished
Tick 0x1
Tick 0x2
Tick 0x3

```

Figure 9 Screenshot of ARC 770D self-test

7. Observe the walking patterns shown by the CPU LEDs and the GPIO LEDs.

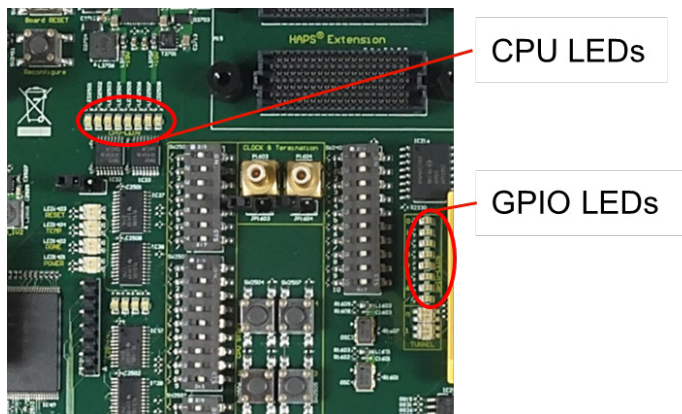


Figure 10 Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard

8. Push the RESET button on the ARC SDP Mainboard to stop the test.



Figure 11 Location of the RESET button on the ARC SDP Mainboard

## 4.3 Executing a Self-Test of the AS221 Core #1

Follow the steps described below to perform a self-test of the AS221 core #1:

1. Connect the ARC SDP Mainboard to your PC using the USB cable, which shall be connected to the USB Dataport of the Mainboard.
2. Connect the power supply to the ARC SDP Mainboard and switch ON the Mainboard or press the RESET button.

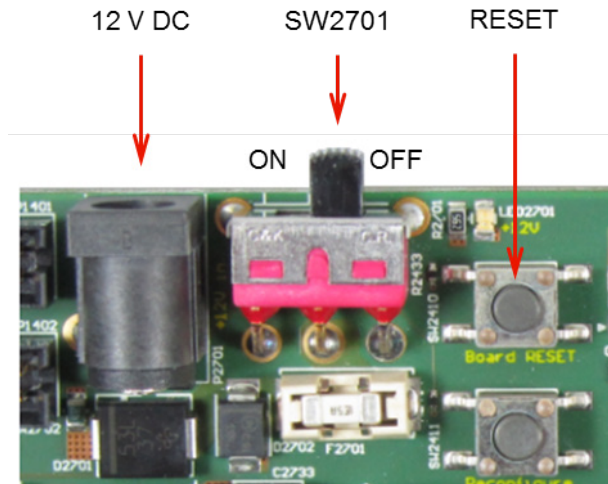


Figure 12 Location of the ARC SDP Mainboard's power supply and power switch

3. Launch PuTTY on your computer
4. Push the CPU start button SW2505 on the ARC SDP Mainboard.

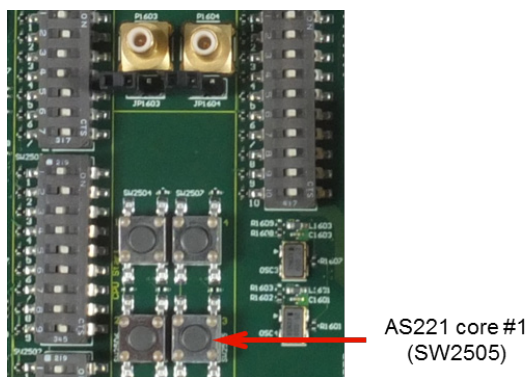
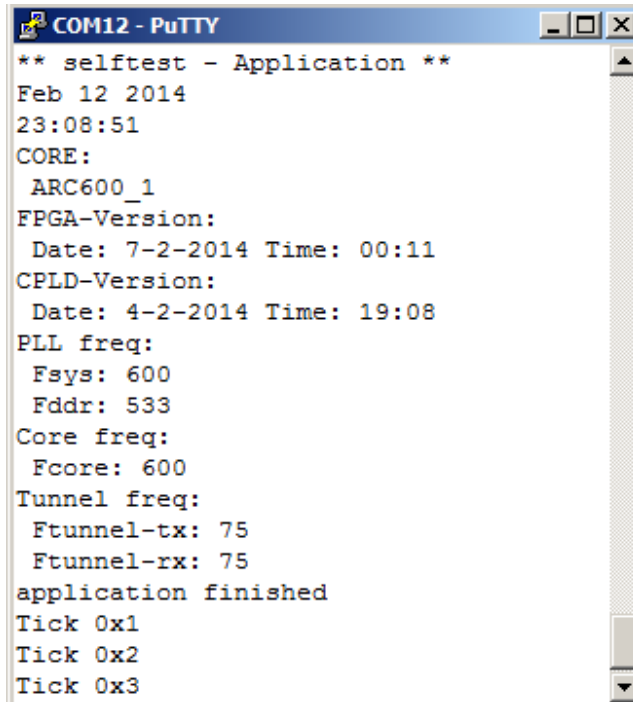


Figure 13 Location of the CPU Start button SW2505 for the AS221 core #1

5. Check that the seven-segment displays shows "10"

- Observe the output on the console, which should be similar to the screenshot shown in [Figure 14](#).



```

COM12 - PuTTY
** selftest - Application **
Feb 12 2014
23:08:51
CORE:
  ARC600_1
FPGA-Version:
  Date: 7-2-2014 Time: 00:11
CPLD-Version:
  Date: 4-2-2014 Time: 19:08
PLL freq:
  Fsys: 600
  Fddr: 533
Core freq:
  Fcore: 600
Tunnel freq:
  Ftunnel-tx: 75
  Ftunnel-rx: 75
application finished
Tick 0x1
Tick 0x2
Tick 0x3

```

Figure 14 Screenshot of the AS221 core #1 self-test

- Observe the walking patterns shown by the CPU LEDs and the GPIO LEDs.

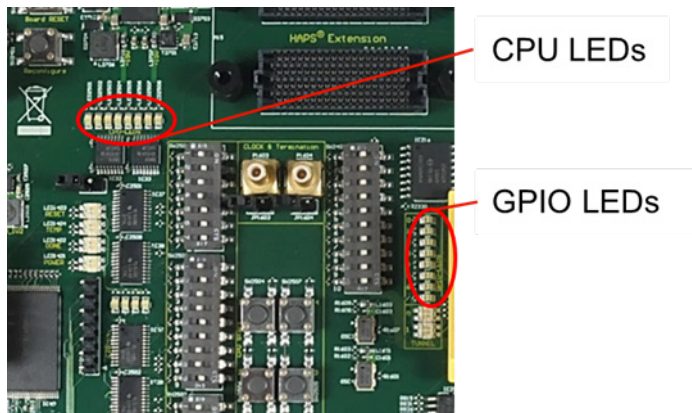


Figure 15 Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard

- Push the RESET button on the ARC SDP Mainboard to stop the test.



Figure 16 Location of the RESET button on the ARC SDP Mainboard

## 4.4 Executing a Self-Test of the AS221 Core #2

Follow the steps described below to perform a self-test of the AS221 core #2:

1. Connect the ARC SDP Mainboard to your PC using the USB cable, which shall be connected to the USB Dataport of the Mainboard.
2. Connect the power supply to the ARC SDP Mainboard and switch ON the Mainboard or press the RESET button.

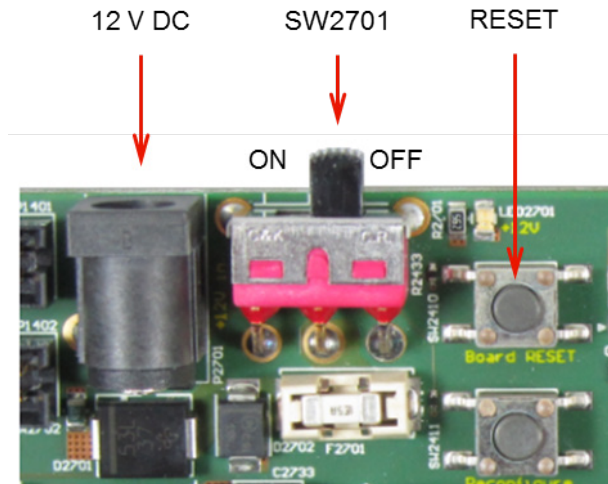


Figure 17 Location of the ARC SDP Mainboard's power supply and power switch

3. Launch PuTTY on your computer
4. Push the CPU start button SW2507 on the ARC SDP Mainboard.

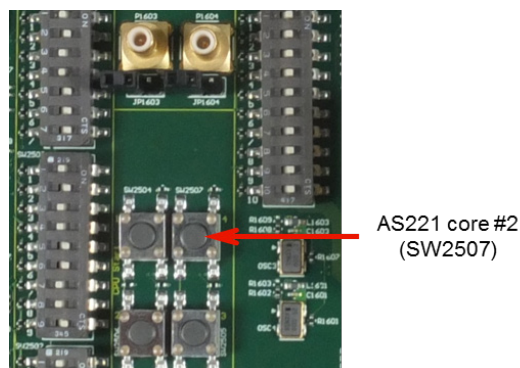
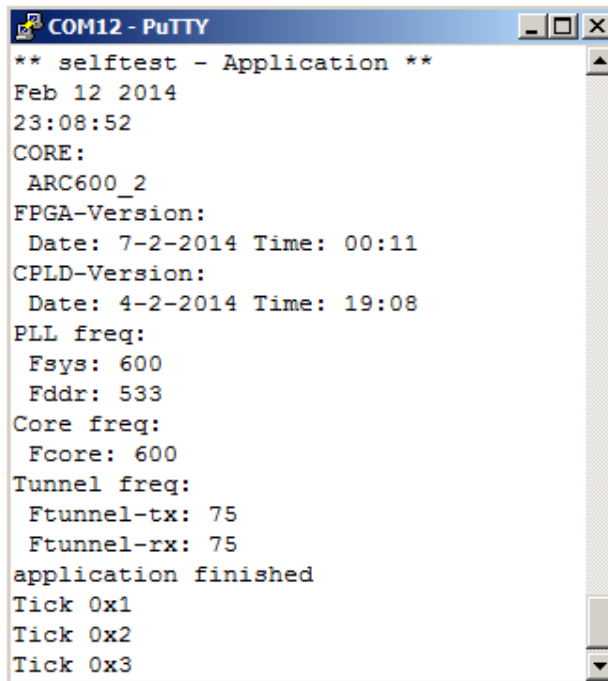


Figure 18 Location of the CPU Start button SW2507 for the AS221 core #2

5. Check that the seven-segment displays shows "20"

- Observe the output on the console, which should be similar to the screenshot shown in [Figure 19](#).



```

COM12 - PuTTY
** selftest - Application **
Feb 12 2014
23:08:52
CORE:
  ARC600_2
FPGA-Version:
  Date: 7-2-2014 Time: 00:11
CPLD-Version:
  Date: 4-2-2014 Time: 19:08
PLL freq:
  Fsys: 600
  Fddr: 533
Core freq:
  Fcore: 600
Tunnel freq:
  Ftunnel-tx: 75
  Ftunnel-rx: 75
application finished
Tick 0x1
Tick 0x2
Tick 0x3

```

Figure 19 Screenshot of the AS221 core #2 self-test

- Observe the walking patterns shown by the CPU LEDs and the GPIO LEDs.

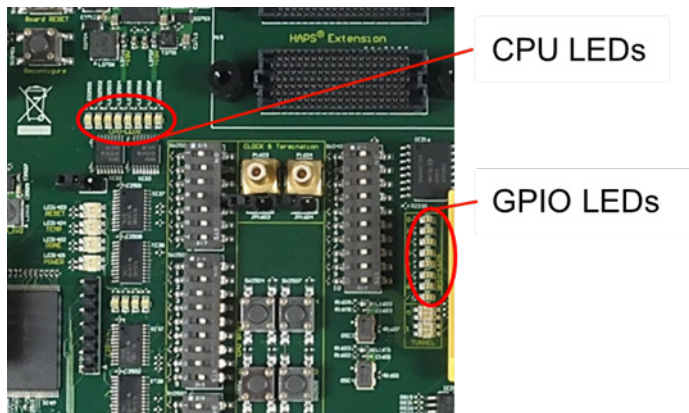


Figure 20 Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard

- Push the RESET button on the ARC SDP Mainboard to stop the test.

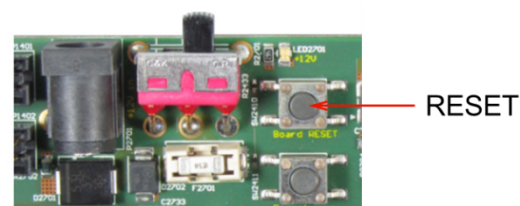


Figure 21 Location of the RESET button on the ARC SDP Mainboard

## 4.5 Executing a Self-Test of the ARC EM6 Core

Follow the steps described below to perform a self-test of the ARC 770D core:

1. Connect the ARC SDP Mainboard to your PC using the USB cable, which shall be connected to the USB Dataport of the Mainboard.
2. Connect the power supply to the Mainboard and switch ON the Mainboard or press the RESET button.

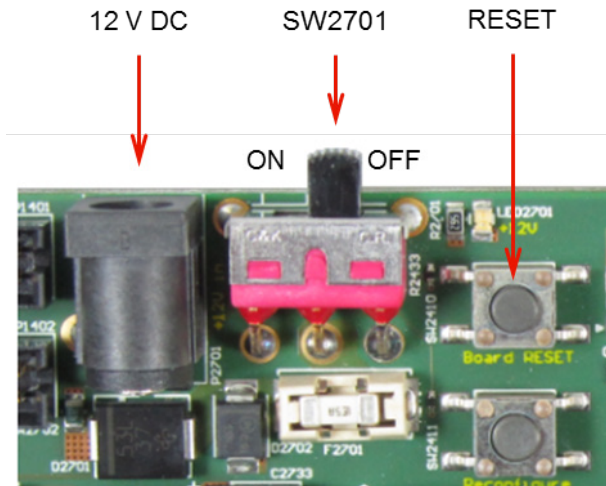


Figure 22 Location of the ARC SDP Mainboard's power supply and power switch

3. Launch PuTTY on your computer
4. Push the CPU Start button SW2504 on the ARC SDP Mainboard.

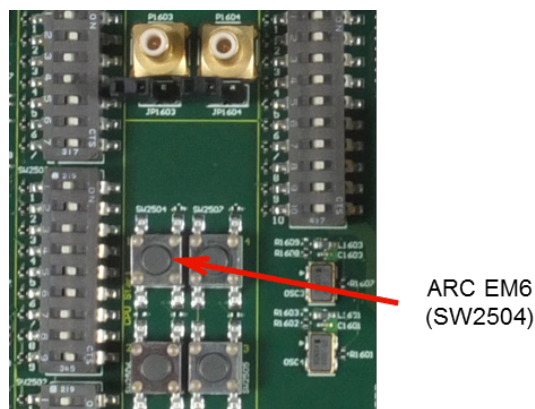
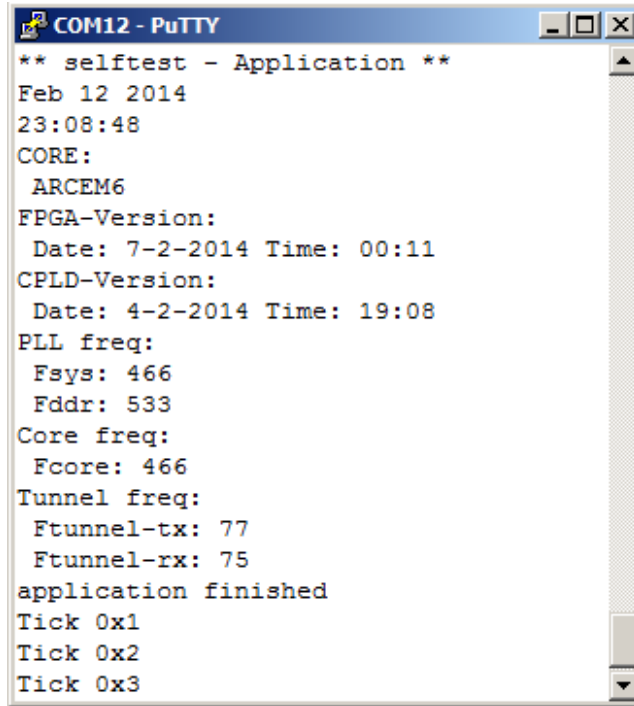


Figure 23 Location of the CPU Start button SW2504 for the ARC EM6 core

5. Check that the seven-segment displays shows "30"



- Observe the output on the console, which should be similar to the screenshot shown in [Figure 24](#).



```

COM12 - PuTTY
** selftest - Application **
Feb 12 2014
23:08:48
CORE:
  ARCEM6
FPGA-Version:
  Date: 7-2-2014 Time: 00:11
CPLD-Version:
  Date: 4-2-2014 Time: 19:08
PLL freq:
  Fsys: 466
  Fddr: 533
Core freq:
  Fcore: 466
Tunnel freq:
  Ftunnel-tx: 77
  Ftunnel-rx: 75
application finished
Tick 0x1
Tick 0x2
Tick 0x3

```

Figure 24 Screenshot of ARC EM6 self-test

- Observe the walking patterns shown by the CPU LEDs and the GPIO LEDs.

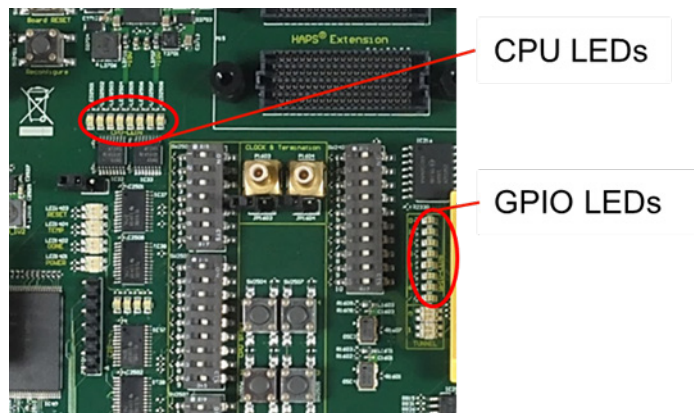


Figure 25 Location of the CPU LEDs and the GPIO LEDs on the ARC SDP Mainboard

- Push the RESET button on the ARC SDP Mainboard to stop the test.



Figure 26 Location of the RESET button on the ARC SDP Mainboard

---

## 4.6 Restoring the Self-Tests in the SPI Flash

If you store your own applications in the SPI Flash on the ARC SDP Mainboard, then the factory-programmed self-tests are typically erased.

To restore the factory-programmed self-tests, follow the steps below:

1) [if you have not done this earlier]

Download and unzip the `axs101_software_<version>.zip` file from the [ARC SDP download webpage](#) [6] and install the `axs_comm` tool as described in the *ARC SDP Mainboard User Guide* [7].

2) Download and unzip the `axs101_selftest_firmware_<version>.zip` file from the [ARC SDP download webpage](#) [6]

3) Navigate to the `/selftest_firmware` folder and double-click on the `axs_comm_program_selftest.bat` batch file.

## Hardware Functional Description

---

*This chapter provides information about the hardware used in the AXC001 CPU Card.*

---

### 5.1 Board Overview

The AXC001 CPU Card hardware is a daughter card for the ARC SDP Mainboard. It includes an ASIC with a total of four of ARC CPUs, allowing processor operation at a target speed:

- One ARC 770D
- One dual-core AS221BD
- One ARC EM6

One of the cores of AS221BD can boot individually as ARC625D. The ARC EM6 can be used as ARC EM4 by disabling the caches. Therefore, the AXC001 CPU Card broadly covers the ARC 700, ARC600 and ARC EM processor families.

Next to the ARC processors, the AXC001 CPU Card includes 512 MByte DDR2 SDRAM and 128 KByte on-chip SRAM.

The AXS101 Software Development Platform consisting of the AXC001 CPU Card and the ARC SDP Mainboard is a powerful, high-speed development platform enabling real-time software development and validation, driver development, code porting, software debugging, and system analysis.

Linux, MQX and baremetal applications are natively supported. OS and driver porting for other operating systems is facilitated as well.

The AXS101 Software Development Platform is HAPS compliant and enables the use of high frequency ARC CPU cores as daughter card for HAPS allowing full SoC prototyping including ARC CPUs.

[Figure 27](#) shows the hardware block diagram of the AXC001 CPU Card.

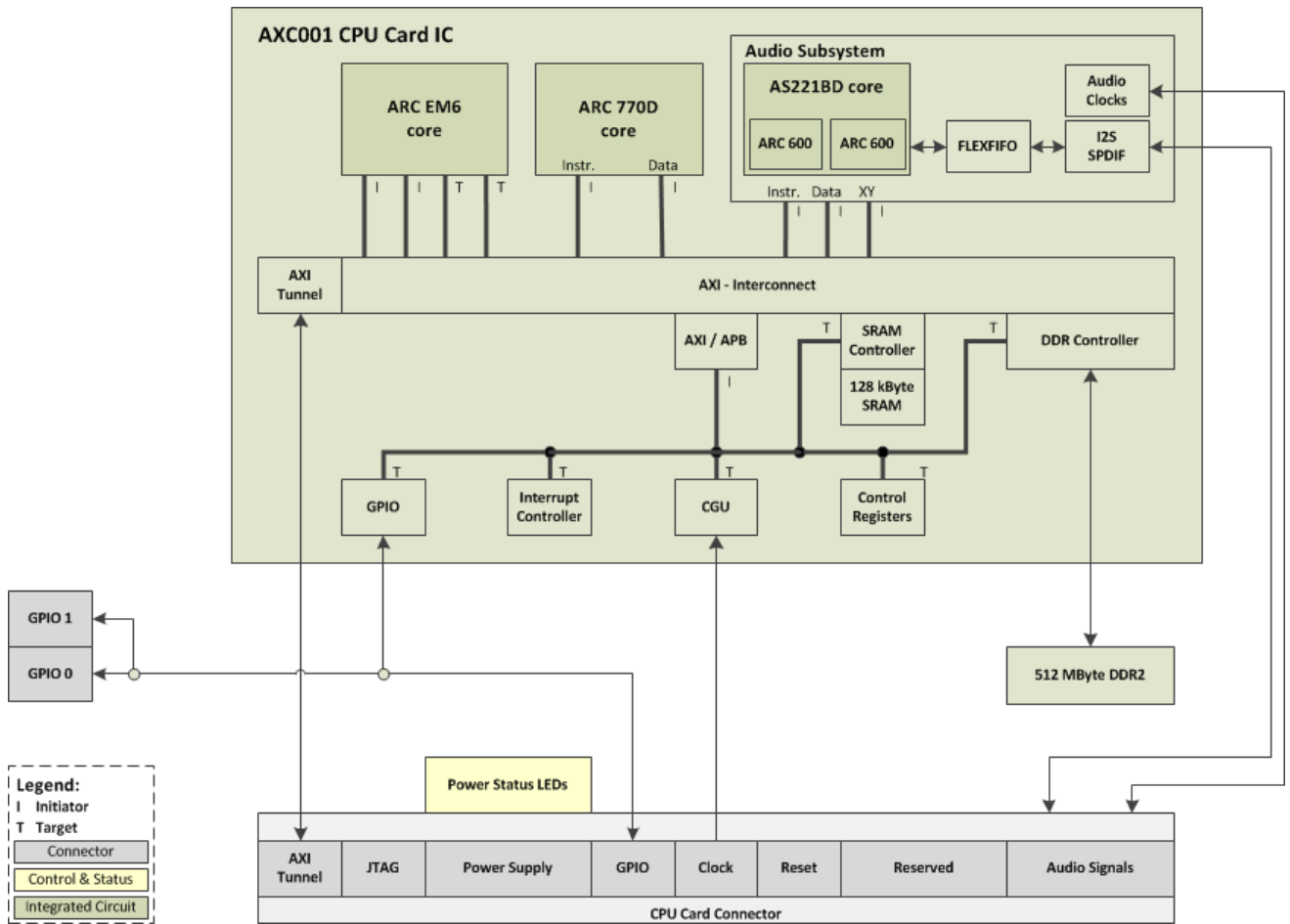


Figure 27 Hardware block diagram

---

## 5.2 Board Interface Overview

The AXC001 CPU Card has two female HapsTrak-II connectors and a female 18-pin power supply connector at the bottom side of the board. These connectors are used for mounting the AXC001 CPU Card on the ARC SDP Mainboard.

At the top side, the AXC001 CPU Card has two male HapsTrak-II connectors, which can be used to connect a HAPS Logic analyzer card for debugging.

Additionally, the AXC001 CPU Card features two male SMB clock outputs.

---

### 5.2.1 Power Supply Connector

Power is supplied to the AXC001 CPU Card by the ARC SDP Mainboard via the power supply connector on the bottom side of the AXC001 CPU Card board. The following voltage levels are provided: 1.1V, 1.8V, 2.5V and 3.3V.

See the [“Power Supply”](#) section for more details.

---

### 5.2.2 HapsTrak-II Connectors (bottom)

The AXC001 CPU Card communicates with the ARC SDP Mainboard via two HapsTrak-II connectors, which carry signal groups such as:

- AXI tunnel
- GPIO
- Clock
- Reset
- JTAG (ARC cores)
- Audio signals
  - Stereo I<sup>2</sup>S slave input
  - Stereo I<sup>2</sup>S slave output
  - 8-channel I<sup>2</sup>S slave output
  - SPDIF input
  - SPDIF output
  - PLL reference clock, PLL clock and lock signal

These connectors include 24 GPIO pins, which are connected to DIP switches on the ARC SDP Mainboard during reset. These switches are used to configure the boot mode of the ARC cores on the AXC001 CPU Card. At the end of the reset the switch settings are latched inside the AXC001 Processor IC. After reset these signals are connected to port A (bits [23:0]) of the GPIO peripheral of the AXC001 Processor IC.

Refer to the [“GPIO Registers”](#) section for details on the functionality.

---

### 5.2.3 HapsTrak-II Connectors (top)

A HAPS logic analyzer card can be connected to the HapsTrak-II connectors at the top side of the AXC001 Processor IC. For example, this can be used to observe the signals of the AXI tunnel between the AXC001 CPU Card and the ARC SDP Mainboard.

---

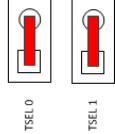
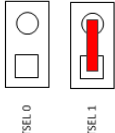
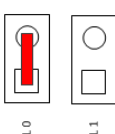
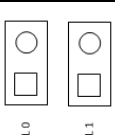
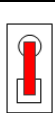
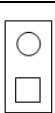
### 5.2.4 GPIO Outputs

Two male SMB connectors `GPIO 0` and `GPIO 1` are located at the top side of the AXC001 CPU Card. They have a voltage level of 1.8V and are connected to port A, bits 0 and 1 of the GPIO peripheral, located inside the AXC001 Processor IC. These GPIO bits are additionally connected to the LEDs `LED2501` and `LED2502` on ARC SDP Mainboard via the HapsTrak-II connector.

## 5.3 Jumpers

Table 3 describes the functionality of the three jumpers on the AXC001 CPU Card. Default jumper settings are shown in Figure 28.

Table 3 Jumper functionality

Jumper Name	Setting	Description
TSEL0 TSEL1		JTAG daisy chain including all ARC cores
		JTAG only includes ARC 770D
		JTAG only includes ARC EM6
		JTAG only includes AS221
SMS enable		Normal Operation
		Reserved

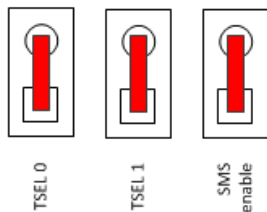


Figure 28 Default Jumper Settings

## 5.4 AXC001 Processor IC Overview

### 5.4.1 Main Features of the ARC Cores

DesignWare ARC processor cores are highly configurable and support a variety of extensions. [Table 4](#) lists the main features of the core configurations available in the AXC001 Processor IC.

Table 4 Main features of the ARC cores

Feature	ARC 770D	AS221BD	ARC EM6
I-Cache (bytes) Associativity Cache-line size	32K 2-way 32 bytes	Core #1: 16K Core #2: 16K 4-way 16 words=64 bytes	32K 4-way 32 bytes
D-Cache (bytes) Associativity Cache-line size	32K 4-way 32 bytes	Core #1: 16K Core #2: 16K 4-way 8 words = 32 bytes	32K 4-way 32 bytes
Internal memory (bytes)	No	Core #1: 8K+8K XY memory Core #2: 8K+8K XY memory	64K ICCM 64K DCCM
DMP port	No	No	Yes
Core interface	AXI (64-bit)	AXI (32-bit)	AHB
Core extensions	XMAC  MMU Performance monitors TimeStampCounter	32x16 XMAC 24x24 XMAC  Performance monitors Timer0 Timer1 Normalize Swap CRC ExtendedArith Multi-core debug	2-cycle multiplier  Performance monitors Code-Density ISA Bit-Scan ISA Divide/remainder Shift ISA SWAP ISA
Floating-point	Single and double precision	Core #1: Single and double precision	Single and double precision
Max. CPU frequency	800 MHz	600 MHz	475 MHz



## 5.4.2 Interrupt

The interrupt architecture of the AXC001 CPU Card distinguishes between two interrupt sources:

- Internal interrupts from sources within the AXC001 Processor IC

The ICTL module combines the interrupt requests from the on-chip interrupt sources into a single interrupt request for each ARC processor. Both cores within the AS221BD share the same interrupt.

The interrupt source responsible for generating the interrupt request can be determined by reading back the interrupt status register in the GPIO sub-module of the ICTL.

The interrupt should be cleared within the GPIO sub-module of the ICTL module.

Use the GPIO driver for accessing the registers inside the ICTL.

- External interrupts

The ICTL module on the Mainboard combines all Interrupt requests from the Mainboard peripherals into a single signal, which is received by the (top level) GPIO module on the AXC001 CPU Card. The interrupt output of the GPIO module is shared between all ARC cores.

In a first step the interrupt handler should read the status register of the ICTL module on the Mainboard to identify the source peripheral of the interrupt. In a second step the interrupt status register of the source peripheral provides the primary root cause. At both intermediate levels (Mainboard ICTL and CPU Card GPIO) the interrupts are level sensitive,

The interrupt should be cleared within the source peripheral.

All interrupts inputs of the ICTL module are edge-sensitive. In general, all interrupt signals are active-high. However, the interrupt inputs of the ARC 770D and the AS221BD are active-low. Inverters are included to adapt the interrupt polarities accordingly.

[Figure 29](#) shows the top-level interrupt architecture of the AXC001 Processor IC.

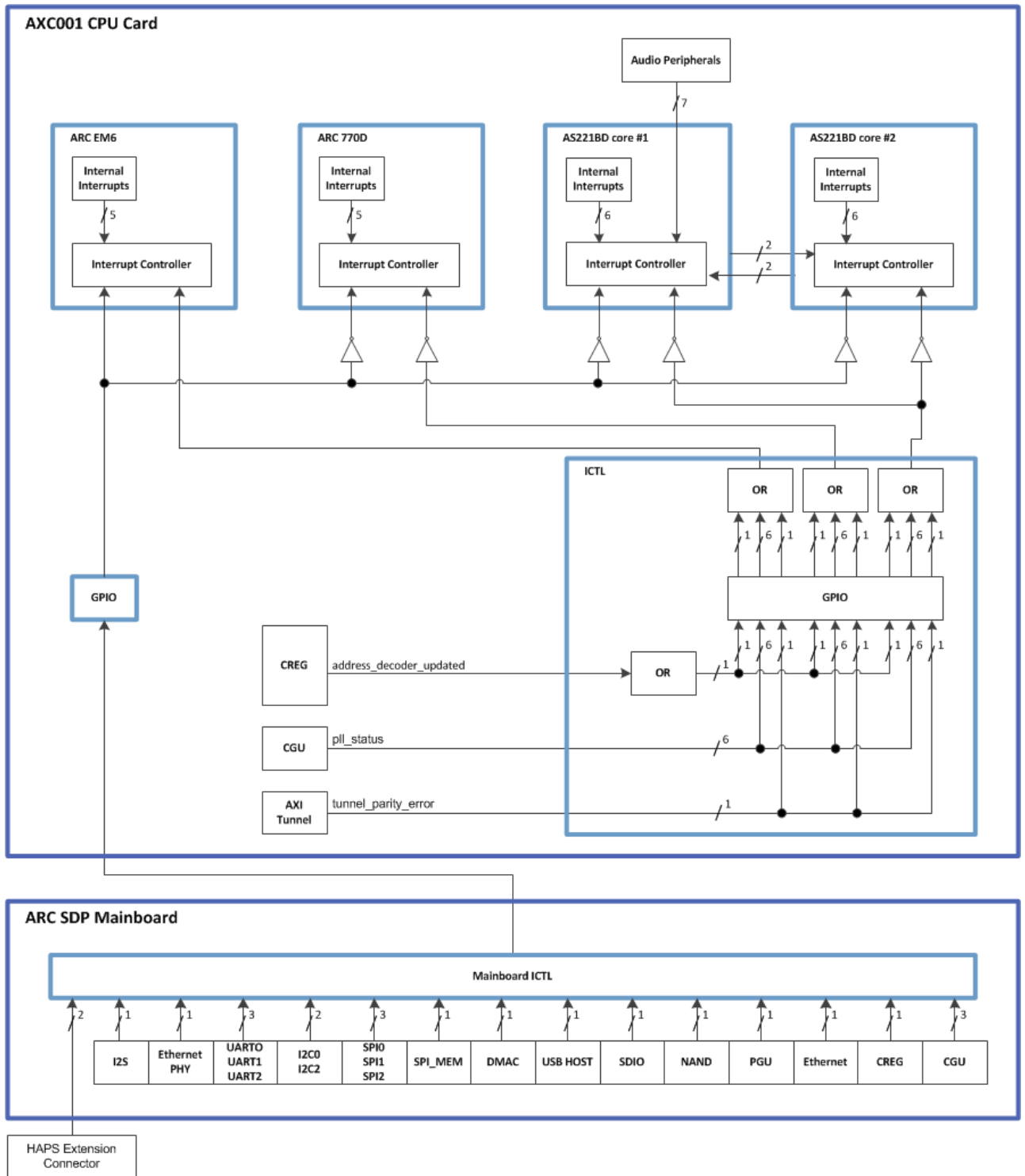


Figure 29 Interrupt architecture

Each ARC core is configured with 11 external interrupt inputs. The interrupt mapping for the ARC cores is listed in the tables below. Additionally, [Table 9](#) shows how the interrupt status register of the Mainboard ICTL can be used to identify the source peripheral for the combined Mainboard interrupt request.

*Table 5 Interrupt mapping for the ARC 770D core*

Interrupt #	Interrupt Source	Remarks														
0	ARC internal interrupt	Reset														
1		Memory error														
2		Instruction error														
3		Timer0														
4		Timer1														
5	Reserved															
6																
7	Combined interrupt from Mainboard peripherals	The interrupt request is received via the ICTL module on the Mainboard and the GPIO register EXT_PORTA[12] on the AXC001 CPU Card. See <a href="#">Table 9</a> .														
8	Reserved															
9																
10																
11																
12																
13																
14																
15	Interrupt from interrupt controller on AXC001 CPU Card	<p>The interrupts are edge-sensitive within the ICTL and should also be cleared at the ICTL level.</p> <table border="1"> <thead> <tr> <th>ICTL bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>12</td> <td>DDR2 PLL locked</td> </tr> <tr> <td>13</td> <td>DDR2 PLL unlocked</td> </tr> <tr> <td>14</td> <td>DDR2 PLL lock error</td> </tr> <tr> <td>15</td> <td>SYS PLL locked</td> </tr> <tr> <td>16</td> <td>SYS_PLL unlocked</td> </tr> <tr> <td>17</td> <td>SYS_PLL lock error</td> </tr> </tbody> </table>	ICTL bit	Description	12	DDR2 PLL locked	13	DDR2 PLL unlocked	14	DDR2 PLL lock error	15	SYS PLL locked	16	SYS_PLL unlocked	17	SYS_PLL lock error
ICTL bit	Description															
12	DDR2 PLL locked															
13	DDR2 PLL unlocked															
14	DDR2 PLL lock error															
15	SYS PLL locked															
16	SYS_PLL unlocked															
17	SYS_PLL lock error															

Table 6 Interrupt mapping for ARC AS221 (core #1)

Interrupt #	Interrupt Source	Remarks	
0	ARC internal interrupt	Reset	
1		Memory error	
2		Instruction error	
3		Timer0	
4		XY	
5	Reserved		
6			
7	ARC internal interrupt	Timer1	
8		ARC AS221 core #2	edge sensitive only
9		ARC AS221 core #2	edge sensitive only
10	Combined interrupt from Mainboard peripherals	The interrupt request is received via the ICTL module on the Mainboard and the GPIO register EXT_PORTA[12] on the AXC001 CPU Card. See <a href="#">Table 9</a> .	
11	Reserved		
12			
13			
14			
15			
16			
17			

Interrupt #	Interrupt Source	Remarks														
18	Combined interrupt from interrupt controller ICTL on AXC001 CPU Card	<p>The interrupts are edge-sensitive within the ICTL and should also be cleared at the ICTL level.</p> <table border="1"> <thead> <tr> <th>ICTL Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>22</td> <td>DDR2 PLL locked</td> </tr> <tr> <td>23</td> <td>DDR2 PLL unlocked</td> </tr> <tr> <td>24</td> <td>DDR2 PLL lock error</td> </tr> <tr> <td>25</td> <td>SYS PLL locked</td> </tr> <tr> <td>26</td> <td>SYS_PLL unlocked</td> </tr> <tr> <td>27</td> <td>SYS_PLL lock error</td> </tr> </tbody> </table>	ICTL Bit	Description	22	DDR2 PLL locked	23	DDR2 PLL unlocked	24	DDR2 PLL lock error	25	SYS PLL locked	26	SYS_PLL unlocked	27	SYS_PLL lock error
ICTL Bit	Description															
22	DDR2 PLL locked															
23	DDR2 PLL unlocked															
24	DDR2 PLL lock error															
25	SYS PLL locked															
26	SYS_PLL unlocked															
27	SYS_PLL lock error															
19	Audio SS internal interrupt	Reserved														
20		Reserved														
21		Reserved														
22		Flexfifo error interrupt														
23		Subsystem controller error interrupt 0														
24		Subsystem controller error interrupt 1														
25		Subsystem controller functional interrupt														
26		SPDIF RX functional interrupt														
27		Reserved														
28		Reserved														
29		Reserved														
30		Flexfifo read channel interrupt														
31		Flexfifo write channel interrupt														

Table 7 Interrupt mapping for ARC AS221 (core #2)

Interrupt #	Interrupt Source	Remarks	
0	ARC internal interrupt	Reset	
1		Memory error	
2		Instruction error	
3		Timer0	
4		XY	
5	Reserved		
6			
7	ARC internal interrupt	Timer1	
8		ARC AS221 core #1	edge sensitive only
9		ARC AS221 core #1	edge sensitive only
10	Combined interrupt from Mainboard peripherals	The interrupt request is received via the ICTL module on the Mainboard and the GPIO register EXT_PORTA[12] on the AXC001 CPU Card. See <a href="#">Table 9</a> .	
11	Reserved		
12			
13			
14			
15			
16			
17			
18	Interrupt from interrupt controller on AXC001 CPU Card	The interrupts are edge-sensitive within the ICTL and should also be cleared at the ICTL level.	
		<b>ICTL Bit</b>	<b>Description</b>
		22	DDR2 PLL locked
		23	DDR2 PLL unlocked
		24	DDR2 PLL lock error
		25	SYS PLL locked
		26	SYS_PLL unlocked
		27	SYS_PLL lock error

Table 8 Interrupt mapping for the ARC EM6 core

Interrupt #	Interrupt Source	Remarks														
0	ARC internal interrupt	Reset														
1		Memory error														
2		Instruction error														
16		Timer0														
17		Timer1														
18		Reserved														
19																
20	Combined interrupt from Mainboard peripherals	The interrupt request is received via the ICTL module on the Mainboard and the GPIO register EXT_PORTA[12] on the AXC001 CPU Card. See <a href="#">Table 9</a> .														
21	Reserved															
22																
23																
24																
25																
26																
27																
28		Interrupt from interrupt controller on AXC001 CPU Card	<p>The interrupts are edge-sensitive within the ICTL and should also be cleared at the ICTL level.</p> <table border="1"> <thead> <tr> <th>ICTL Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>DDR2 PLL locked</td> </tr> <tr> <td>3</td> <td>DDR2 PLL unlocked</td> </tr> <tr> <td>4</td> <td>DDR2 PLL lock error</td> </tr> <tr> <td>5</td> <td>SYS PLL locked</td> </tr> <tr> <td>6</td> <td>SYS_PLL unlocked</td> </tr> <tr> <td>7</td> <td>SYS_PLL lock error</td> </tr> </tbody> </table>	ICTL Bit	Description	2	DDR2 PLL locked	3	DDR2 PLL unlocked	4	DDR2 PLL lock error	5	SYS PLL locked	6	SYS_PLL unlocked	7
ICTL Bit	Description															
2	DDR2 PLL locked															
3	DDR2 PLL unlocked															
4	DDR2 PLL lock error															
5	SYS PLL locked															
6	SYS_PLL unlocked															
7	SYS_PLL lock error															

Table 9 Mainboard ICTL Interrupt mapping

<b>ICTL_MB INT_STATUS Register Bit</b>	<b>Interrupt Source</b>
0	Mainboard CGU: PLL lock interrupt
1	Mainboard CGU: PLL unlock interrupt
2	Mainboard CGU: PLL lock error interrupt
3	Mainboard CREG interrupt
4	Ethernet interrupt
5	PGU interrupt
6	NAND interrupt
7	SDIO interrupt
8	USB HOST interrupt
9	DMAC interrupt
10	SPI_MEM interrupt
11	SPI0 interrupt
12	SPI1 interrupt
13	SPI2 interrupt
14	I2C0 interrupt
15	I2S interrupt
16	I2C2 interrupt
17	UART0 interrupt
18	UART1 interrupt
19	UART2 interrupt
20	Mainboard GPIO0 interrupt
21	Mainboard GPIO1 interrupt
22	Ethernet PHY interrupt
23	Reserved
24	HAPS Extension 0 interrupt (signal HE_intr[0] at connector)
25	HAPS Extension 1 interrupt (signal HE_intr[1] at connector)



### 5.4.3 Clock

The AXC001 CPU Card has three clock inputs from which all other clocks are derived internally:

- 33 MHz reference clock
- AXI tunnel clock
- Audio PLL clock

These clocks are provided at the HAPS Trak-II connectors on the bottom side of the AXC001 CPU Card.

The main clock domains are highlighted by different colors on [Figure 30](#). The AXC001 Processor IC has the following main clock domains:

- 33 MHz clock  
This is the input clock of the AXC001 Processor IC. All other clocks are derived from this clock. Only two exceptions are:
  - source-synchronous input clock for the AXI tunnel
  - audio PLL clock for the Audio Subsystem
- DDR PHY clock  
This is the clock that is used by the DLLs in the DDR2/3 lite PHY. These DLLs generate the different clock phases ( $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  and  $270^\circ$ ) that are required for PHY timing management; the  $0^\circ$  phase output of the command lane DLL (i.e. the MDLL) is also used to clock the DDR memory controller and the AXI bus. The AXI bus runs at the same clock as the DDR controller to minimize DDR access latency.
- ARC EM6 clock
- ARC 770D clock
- ARC AS221 clock  
Besides clocking the ARC AS221, this clock is also used as the system clock for the Audio Subsystem. Internally, the Audio Subsystem uses this clock to generate the APB clock for the audio peripherals like e.g. I<sup>2</sup>S.
- Audio Subsystem clocks  
Besides the above mentioned system clock, the Audio Subsystem operates on the following main clocks:
  - fixed reference clock
  - adjustable reference clock
  - audio PLL clock
- APB clock
- AXI Tunnel clock

Figure 30 shows the top-level clock architecture. The 33 MHz input clock is supplied to the CGU. The CGU generates all the internal clocks using two PLLs and some fractional dividers. There is a dedicated PLL for the DDR PHY, which typically runs at 533 MHz. The other PLL runs at 1 GHz and is used as a reference clock for all other clock domains. Fractional dividers allow for accurate fine-tuning of the internal clocks to the desired frequency.

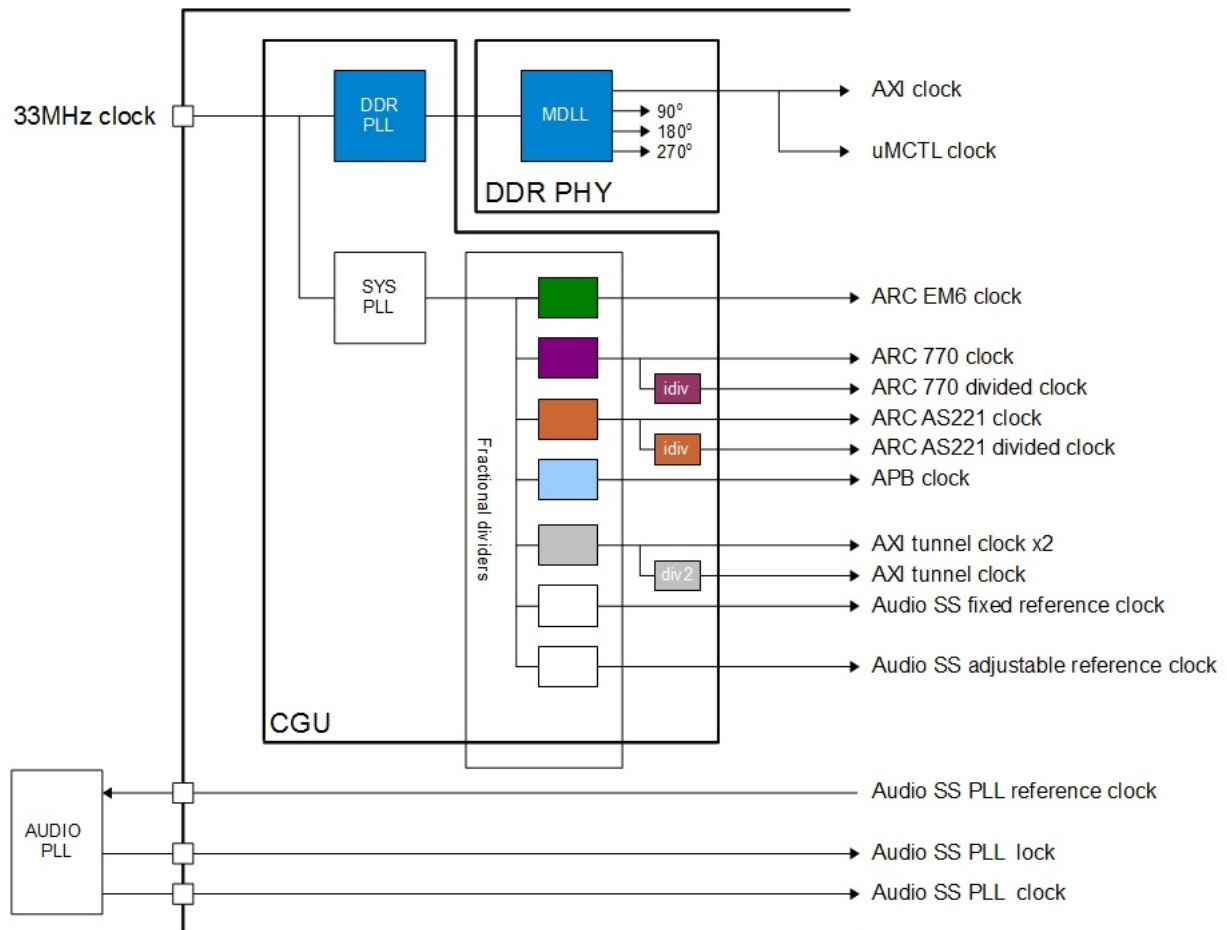


Figure 30 Clock architecture

#### 5.4.4 Reset

The AXC001 Processor IC has one external reset pin that serves as an active low, hardware reset. When the external hardware reset is active, the entire IC is reset. This pin is routed to the HapsTrak-II connector and controlled by reset circuitry on the ARC SDP Mainboard, which also implements the power-on-reset. The CGU on the AXC001 Processor IC performs reset synchronization to the internal clock.

Use the `RESET` button on the ARC SDP Mainboard to trigger a reset of the entire system.

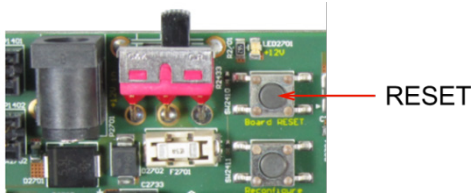


Figure 31 Location of the RESET button on the ARC SDP Mainboard

The AXC001 Processor IC has an external reset output pin, which is routed to the HapsTrak-II connector as well. This reset output is used by the reset controller on the ARC SDP Mainboard.

### 5.4.5 Debug

The ARC cores provide debug access via an IEEE 1149.1 JTAG port. In the AXC001 Processor IC the JTAG ports of the different ARC cores are daisy-chained into a so-called JTAG chain. In a JTAG chain the data output from the first core becomes the data input to the second core and so forth; the control and clock signals are common to all the cores in the chain. The JTAG chain for the AXC001 Processor IC is shown in Figure 32. To distinguish between the individual cores in the JTAG chain each core has a unique JTAG IDCODE as listed in Table 10.

The AXC001 CPU Card provides two jumpers *TSEL 0* and *TSEL 1* (see “Jumpers” section) that can be used to break the JTAG daisy-chain and select one of the cores individually.

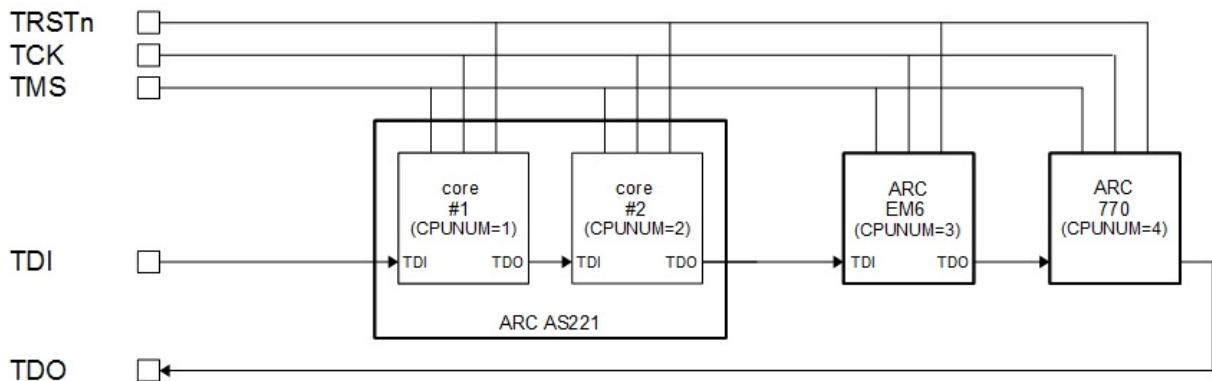


Figure 32 JTAG daisy-chain

Table 10 JTAG ID codes

Core	JTAG ID bit [31:0]	ARC ID	CPUNUM
ARC AS221 #1	0x2004_24B1	0x0126	1
ARC AS221 #2	0x2008_24B1	0x0226	2
ARC EM6	0x200C_44B1	0x0341	3
ARC 770D	0x2010_34B1	0x0434	4

## 5.4.6 Control Registers

The CREG peripheral inside the AXC001 Processor IC provides SW registers to control the following features:

- System memory map (see “[System Memory Map](#)” section)
- Boot mode (see “[Boot Modes](#)” section)

[Table 11](#) lists the control registers including a brief description and provides the address offsets to the base address of the AXI2APB section within the system memory map as well. By default, the base address of the AXI2APB segment is 0xF000\_0000 (see “[System Memory Map after Pre-Bootloader Execution](#)”).

A detailed description of the control registers can be found in the “[Software Interfaces](#)” section.

*Table 11 Control register memory map*

Name	Address Offset	R/W	Description
<b>ARC EM6 Address Decoder Registers</b>			
CREG_EM6_A_SLV0	0x1000	R/W	Address decoder slave select register 0 for ARC EM6
CREG_EM6_A_SLV1	0x1004	R/W	Address decoder slave select register 1 for ARC EM6
CREG_EM6_A_OFFSET0	0x1008	R/W	Address decoder offset register 0 for ARC EM6
CREG_EM6_A_OFFSET1	0x100C	R/W	Address decoder offset register 1 for ARC EM6
CREG_EM6_A_BOOT	0x1010	R/W	Address decoder boot mirror register for ARC EM6
CREG_EM6_A_UPDATE	0x1014	RW1C	Address decoder update register for ARC EM6
<b>ARC 770D Address Decoder Registers</b>			
CREG_770_A_SLV0	0x1020	R/W	Address decoder slave select register 0 for ARC 770D
CREG_770_A_SLV1	0x1024	R/W	Address decoder slave select register 1 for ARC 770D
CREG_770_A_OFFSET0	0x1028	R/W	Address decoder offset register 0 for ARC 770D
CREG_770_A_OFFSET1	0x102C	R/W	Address decoder offset register 1 for ARC 770D
CREG_770_A_BOOT	0x1030	R/W	Address decoder boot mirror register for ARC 770D
CREG_770_A_UPDATE	0x1034	RW1C	Address decoder update register for ARC 770D

Name	Address Offset	R/W	Description
<b>ARC AS221 Address Decoder Registers (shared between core #1 and core #2)</b>			
CREG_221_A_SLV0	0x1040	R/W	Address decoder slave select register 0 for ARC 221
CREG_221_A_SLV1	0x1044	R/W	Address decoder slave select register 1 for ARC 221
CREG_221_A_OFFSET0	0x1048	R/W	Address decoder offset register 0 for ARC 221
CREG_221_A_OFFSET1	0x104C	R/W	Address decoder offset register 1 for ARC 221
CREG_221_A_BOOT	0x1050	R/W	Address decoder boot mirror register for ARC 221
CREG_221_A_UPDATE	0x1054	RW1C	Address decoder update register for ARC 221
<b>Address Decoder Registers for AXI Tunnel</b>			
CREG_TUN_A_SLV0	0x1060	R/W	Address decoder slave select register 0 for AXI Tunnel
CREG_TUN_A_SLV1	0x1064	R/W	Address decoder slave select register 1 for AXI Tunnel
CREG_TUN_A_OFFSET0	0x1068	R/W	Address decoder offset register 0 for AXI Tunnel
CREG_TUN_A_OFFSET1	0x106C	R/W	Address decoder offset register 1 for AXI Tunnel
CREG_TUN_A_UPDATE	0x1074	RW1C	Address decoder update register for AXI Tunnel
<b>CPU Start Registers</b>			
CREG_EM6_START	0x1100	RW1C	CPU start register for ARC EM6
CREG_770_START	0x1104	RW1C	CPU start register for ARC 770D
CREG_221_START	0x1108	RW1C	CPU start register for ARC 221

## 5.4.7 GPIO Registers

The GPIO peripheral inside the AXC001 Processor IC is implemented using the DesignWare `dw_apb_gpio` IP [3] and provides one GPIO port.

[Table 12](#) lists the GPIO registers and provides the address offsets to the base address of the AXI2APB section within the system memory map.

By default, the base address of the AXI2APB segment is `0xF000_0000` (see “[System Memory Map after Pre-Bootloader Execution](#)”).

*Table 12 GPIO register memory map*

Name	Address Offset	R/W	Description
SWPORTA_DR	0x3000	R/W	Port A Data Register Controls LEDs on the ARC SDP Mainboard
EXT_PORTA	0x3050	R	External Port A Register Input from CPU Start buttons on the ARC SDP Mainboard and from the Mainboard’s interrupt request

[Table 13](#) and [Table 14](#) describe the function of the GPIO bits.

*Table 13 GPIO port A output register function (SWPORTA\_DR)*

Bit	Description
0	Connected to LED2501 of the ARC SDP Mainboard and to the GPIO 0 connector. The LED is ON when this bit is cleared to 0.
1	Connected to LED2502 of the ARC SDP Mainboard and to the GPIO 1 connector. The LED is ON when this bit is cleared to 0.
4 : 2	Reserved
5	Connected to LED2503 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.
6	Connected to LED2504 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.
9 : 7	Reserved
10	Connected to LED2505 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.

Bit	Description
11	Connected to LED2506 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.
14:12	Reserved
15	Connected to LED2507 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.
16	Connected to LED2508 of the ARC SDP Mainboard. The LED is ON when this bit is cleared to 0.
31:17	Reserved

Table 14 GPIO port A input register function (EXT\_PORTA)

Bit	Description
11:0	Reserved
12	Connected to the interrupt controller of the ARC SDP Mainboard. Can be used to provide an interrupt from the peripheral subsystem of the ARC SDP Mainboard to a core on the AXC001 CPU Card.
19:13	Reserved
20	Connected to push button SW2504 of the ARC SDP Mainboard
21	Connected to push button SW2506 of the ARC SDP Mainboard
22	Connected to push button SW2505 of the ARC SDP Mainboard
23	Connected to push button SW2507 of the ARC SDP Mainboard
31:24	Reserved

## 5.5 Memories on the AXC001 CPU Card

The following global memories are available on the AXC001 CPU Card:

- 512 MByte DDR2 SDRAM
- 128 KByte SRAM

The memory controller for the DDR2 SDRAM supports two independent ports, which can both access the entire memory.

Local memories (ICCM, DCCM) are available for the ARC EM6 core. Refer to the [“Main Features of the ARC Cores”](#) section.

The ARC SDP Mainboard provides additional global memories.

## 5.6 Power Supply

Power is supplied to the AXC001 CPU Card by the ARC SDP Mainboard via the power supply connector on the bottom side of the AXC001 CPU Card board. The following voltage levels are provided: 1.1V, 1.8V, 2.5V and 3.3V.

[Table 15](#) provides a pin description of the Power Supply Connector and [Figure 33](#) shows the pinout diagram.

*Table 15 Pin description of the Power Supply Connector*

Pin	Name	Description
1, 2	GND	Ground supply pin
3, 4	1V1	1.1 Volt power supply pin
5, 6	GND	Ground supply pin
7, 8	1V8	1.8 Volt power supply pin
9, 10	GND	Ground supply pin
11, 12	2V5	2.5 Volt power supply pin
13, 14	GND	Ground supply pin
15, 16	3V3	3.3 Volt power supply pin



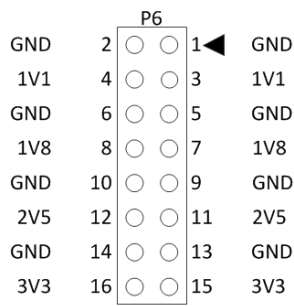



Figure 33 Pinout diagram of the Power Supply Connector (bottom view)

---

 **Note** The power supply connector on the ARC SDP Mainboard has two additional pins, which are not used by the AXC001 CPU Card.

---

## 5.7 Audio Support

The AXC001 CPU Card includes an instance of the DesignWare Soundwave Audio Subsystem and features the following audio interfaces:

- Stereo I<sup>2</sup>S master output
- Stereo I<sup>2</sup>S master input
- 8-channel I<sup>2</sup>S master output
- S/PDIF input
- S/PDIF output
- Audio PLL

These interfaces are all part of the HapsTrak-II connector, which routes them to the ARC SDP Mainboard. The Mainboard includes on-board audio codecs and additionally supports connecting external custom codecs.. The stereo input and output have individual master clocks and may operate at different audio rates.

### 5.7.1 Main Features of the Audio Subsystem

The DesignWare Audio Subsystem IP is highly configurable and supports a variety of audio channels. [Table 16](#) lists the main features of the Audio Subsystem available in the AXC001 Processor IC.

*Table 16 Audio Subsystem main features*

Feature	Value
Size of flexfifo memory (in 24-bit words)	7168
Number of I <sup>2</sup> S analog stereo inputs	1
Number of I <sup>2</sup> S analog stereo outputs	1
Number of I <sup>2</sup> S analog multi-channel 7.1 outputs	1
Number of SPDIF inputs	1
Number of SPDIF outputs	1

[Figure 34](#) shows the hardware block diagram of the audio subsystem in the context of the AXC001 Processor IC. The CGU generates the audio subsystem main clock `audioss_sys_clk` and the audio reference clocks `audioss_audio_adj_ref_clk` and `audioss_audio_fix_ref_clk`. Use the baremetal driver for the CGU to control the clock frequencies.

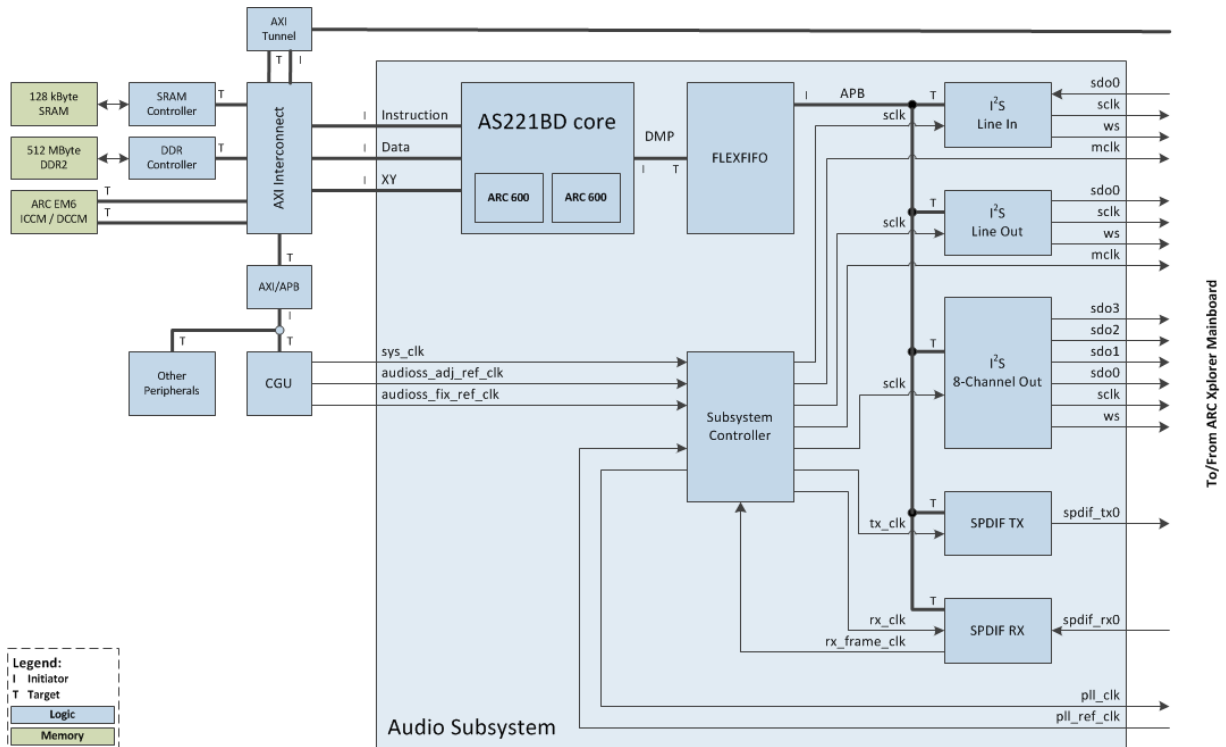


Figure 34 Hardware block diagram of the audio subsystem

## 5.7.2 Stereo Input

The stereo input is provided by the Line In or Microphone input of the ARC SDP Mainboard, or by an external I<sup>2</sup>S slave audio source connected to the Mainboard. Refer to the *ARC SDP Mainboard User Guide [7]* for details.

The signal is connected to the Line In stereo I<sup>2</sup>S master peripheral of the audio subsystem.

## 5.7.3 Stereo Output

The stereo I<sup>2</sup>S output of the AXC001 CPU Card is driven by the Line Out stereo I<sup>2</sup>S master peripheral of the audio subsystem.

On the ARC SDP Mainboard these signals are wired to the I<sup>2</sup>S ports of the audio DAC driving the Line Out and Headphones outputs of the Mainboard or to an external I<sup>2</sup>S slave audio sink connected to the Mainboard. Refer to the *ARC SDP Mainboard User Guide [7]* for details.

---

## 5.7.4 8-Channel Audio Output

The 8-Channel Audio Output of the AXC001 CPU Card is driven by the 8-Channel Out I<sup>2</sup>S master peripheral of the audio subsystem.

On the ARC SDP Mainboard these signals are wired to the I<sup>2</sup>S ports of the audio DAC driving the 8-channel audio jacks of the Mainboard, or to an external I<sup>2</sup>S slave audio sink connected to the Mainboard. Refer to the *ARC SDP Mainboard User Guide [7]* for details

---

## 5.7.5 S/PDIF Input

The S/PDIF input is provided by the S/PDIF In jack of the ARC SDP Mainboard. Refer to the *ARC SDP Mainboard User Guide [7]* for details.

This signal is connected to the SPDIF RX peripheral of the audio subsystem.

---

## 5.7.6 S/PDIF Output

The S/PDIF Output is driven by the SPDIF TX peripheral of the audio subsystem.

On the ARC SDP Mainboard the S/PDIF output is wired to the S/PDIF Out jack. Refer to the *ARC SDP Mainboard User Guide [7]* for details.

---

## 5.7.7 Audio PLL

The AXC001 CPU Card does not include an audio PLL, but feeds a PLL reference clock to an audio PLL on the ARC SDP Mainboard. This reference clock is driven by the port `audio_ss_audio_pll_ref_clk` of the audio subsystem.

The PLL returns the PLL clock and a corresponding PLL lock status signal to the ports `audio_ss_audio_pll_clk` and `audio_ss_audio_pll_locked` of the audio subsystem.

---

## 5.8 Usage of ARC SDP Mainboard Resources

### 5.8.1 Usage of the Mainboard DIP Switches


During a reset, the DIP switches SW2501, SW2502 and SW2503 of the ARC SDP Mainboard are connected to the AXC001 Processor IC via the GPIO signal group of the HapsTrak-II connector. At the end of a reset, the switch settings are latched inside the AXC001 Processor IC and determine the boot behavior of the ARC CPUs as described in [Table 17](#), [Table 18](#) and [Table 19](#).

SW2501 controls the boot configuration of the ARC EM6, SW2502 controls the boot configuration of the ARC 770D and SW2503 controls the boot configuration of the AS221.

The DIP switch settings determine the initial values of some control registers.

**Note**

When a DIP switch is in the right position (  ), the corresponding bit in the control register is '0'.

When a DIP switch is in the left position (  ), the corresponding bit in the control register is '1'.

---

Table 17 ARC EM6 boot configuration (Mainboard DIP switch SW2501)


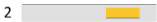





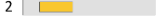
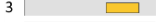



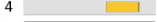
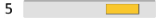





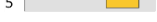

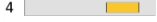

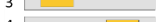
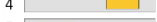
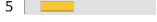






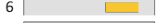
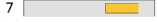






Bit	Description		
1	Boot mode select	<b>Switch Position</b>	<b>CPU Start</b>
		1  2 	By debugger
		1  2 	By CPU Start button; default
		1  2 	By CREG write
2		1  2 	Autonomously
3	Boot mirror select	<b>Switch Position</b>	<b>Boot Mirror</b>
		3  4  5 	Disabled
		3  4  5 	DDR port 0
4		3  4  5 	Local SRAM
		3  4  5 	Pre-Bootloader RAM on Mainboard (via the AXI tunnel); default
5		3  4  5 	ARC EM6 ICCM
		3  4  5 	Reserved
		3  4  5 	Reserved
		3  4  5 	Reserved
6	Boot location select	<b>Switch Position</b>	<b>Boot Location</b>
7		6  7 	0 ... 2 Kbyte; default
		6  7 	2 ... 4 KByte
		6  7 	4 ... 6 KByte
		6  7 	6 ... 8 KByte

Table 18 ARC 770D boot configuration (Mainboard DIP switch SW2502)









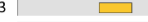



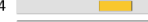
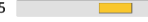


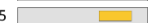







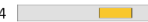

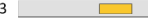





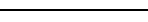





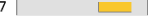

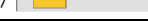
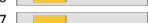





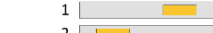


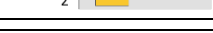



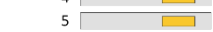






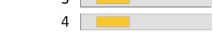



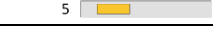

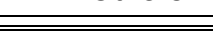
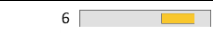



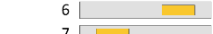


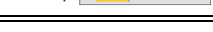
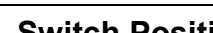
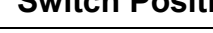


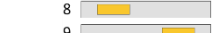




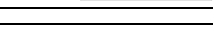


Bit	Description			
1	Boot mode select	<b>Switch Position</b>	<b>CPU Start</b>	
		1  2 	By debugger	
		1  2 	By CPU Start button; default	
		1  2 	By CREG write	
2		1  2 	Autonomously	
		<b>Switch Position</b>		<b>Boot Mirror</b>
		3  4  5 	Disabled	
		3  4  5 	DDR port 0	
4		3  4  5 	Local SRAM	
		3  4  5 	Pre-Bootloader RAM on Mainboard (via the AXI tunnel); default	
		3  4  5 	ARC EM6 ICCM	
		3  4  5 	Reserved	
5		3  4  5 	Reserved	
		3  4  5 	Reserved	
		3  4  5 	Reserved	
6	Boot location select	<b>Switch Position</b>	<b>Boot Location</b>	
		6  7 	0 ... 2 KByte	
		6  7 	2 ... 4 Kbyte; default	
		6  7 	4 ... 6 KByte	
7		6  7 	6 ... 8 KByte	

Table 19 AS221 boot configuration (Mainboard DIP switch SW2503)

Bit	Description		
1	Boot mode select for core #1	<b>Switch Position</b>	<b>CPU Start</b>
		1  2 	By debugger
		1  2 	By CPU Start button; default
		1  2 	By CREG write
2		1  2 	Autonomously
		1  2 	
3	Boot mirror select for cores #1 and #2	<b>Switch Position</b>	<b>Boot Mirror</b>
		3  4  5 	Disabled
		3  4  5 	DDR port 0
		3  4  5 	Local SRAM
		3  4  5 	Pre-Bootloader RAM on Mainboard (via the AXI tunnel); default
		3  4  5 	ARC EM6 ICCM
4		Others	Reserved
6	Boot location select for cores #1 and #2	<b>Switch Position</b>	<b>Boot Location</b>
		6  7 	0 ... 2 KByte
		6  7 	2 ... 4 KByte
		6  7 	4 ... 6 Kbyte; default
7		6  7 	6 ... 8 KByte
		6  7 	
8	Boot mode select for core #2	<b>Switch Position</b>	<b>CPU start</b>
		8  9 	By debugger
		8  9 	By CPU Start button; default
		8  9 	By CREG write
9		8  9 	Autonomously
		8  9 	



If a core is booting from the RAM on the ARC SDP Mainboard via the AXI Tunnel, the Pre-Bootloader is executed and uses bit 10 of the DIP switch SW2401 to control its mode of operation as explained in Table 20. Otherwise, this switch can be used for application purposes.

Table 20 Pre-Bootloader usage of Mainboard DIP switch SW2401

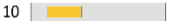

Bit	Description
1 ... 9	Not used by Pre-Bootloader
10	 <p><b>Bypass Mode</b></p> <p>Pre-Bootloader sets up clocks, memory maps and initializes the DDR2 SDRAM. Core is halted afterwards.</p> <p>The left character of the seven-segment display shows a number and a dot.</p>
	 <p><b>Normal Mode</b></p> <p>(Default) Pre-Bootloader sets up clocks, memory maps and initializes the DDR2 SDRAM. It loads an image from the SPI Flash on the Mainboard and starts execution.</p> <p>The left character of the seven-segment display shows a number only (no dot).</p>

Figure 35 shows functions and default settings of the DIP Switches on the ARC SDP Mainboard, which are used by the AXC001 CPU Card.

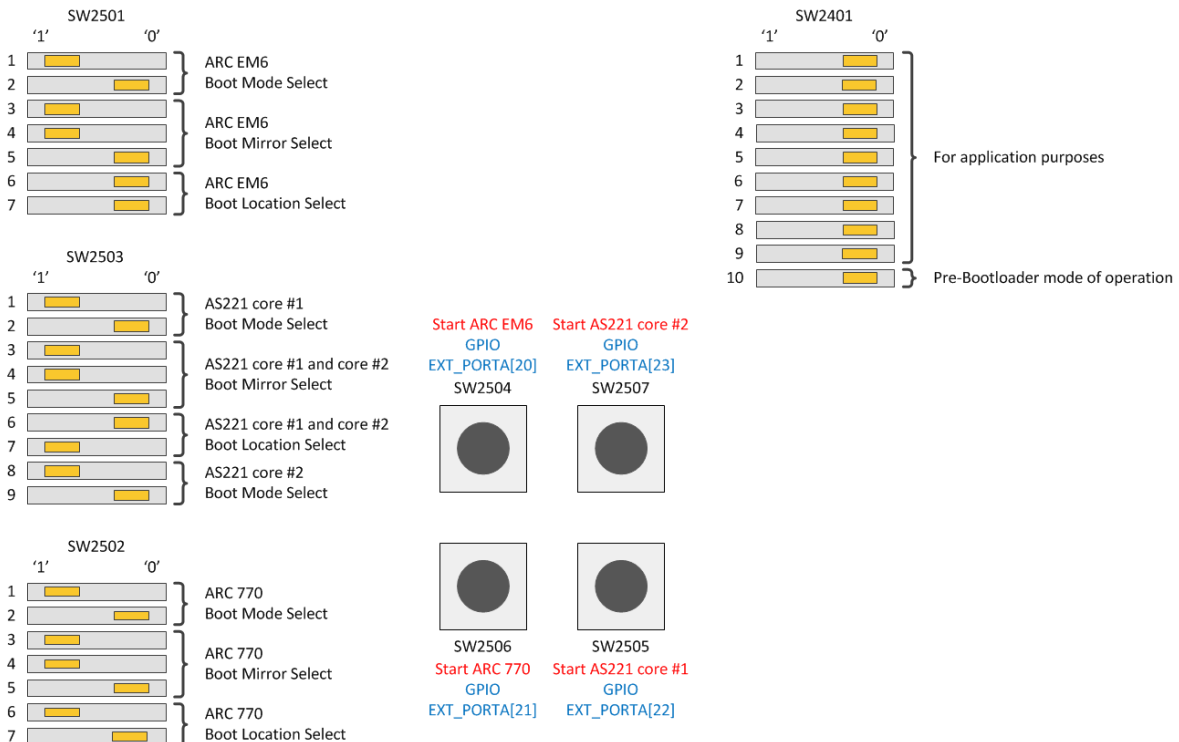


Figure 35 Functions and default settings of the DIP switches on the ARC SDP Mainboard.

The DIP switch settings shown in Figure 35 correspond to the factory default settings. All cores are configured to boot via the AXI tunnel and to delay the start of the code execution

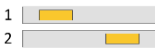

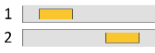
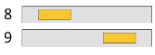
until the corresponding `CPU Start` button is pressed. By default, the boot locations are selected such that the ARC EM6 core starts at address offset 0, the ARC 770D starts with an offset of 2 KByte and the AS221 with an offset of 4 KByte. The Pre-Bootloader loads an image from the RAM on the Mainboard and starts execution.

## 5.8.2 Usage of the Mainboard Push Buttons

The start behavior of the ARC cores on the AXC001 CPU Card is configurable, one of the options being that the core starts automatically after reset. For the remaining options, the core halts after a reset. However, multiple ways to start code execution exist. One of these ways is to start code execution when a button is pressed. The corresponding `CPU Start` buttons are located on the ARC SDP Mainboard. The behavior of the ARC cores is controlled individually using control registers, which are initialized using DIP switches on the ARC SDP Mainboard. Refer to the “[Usage of the Mainboard DIP Switches](#)” section for details.

[Table 21](#) describes how the `CPU Start` buttons of the ARC SDP Mainboard are used to start code execution on the individual cores. [Figure 36](#) shows the location of the `CPU Start` buttons on the ARC SDP Mainboard.

*Table 21 Usage of the CPU Start buttons of the ARC SDP Mainboard*

Push Button	Description
SW2504	CPU start for ARC EM6 when <code>CREG_EM6_START[5:4] = 0x1</code> (boot mode select switch SW2501 set to  during reset), Otherwise, GPIO EXT_PORTA[20]
SW2505	CPU start for AS221 core #1 when <code>CREG_221_START[5:4] = 0x1</code> (boot mode select switch SW2503 set to  during reset), Otherwise, GPIO EXT_PORTA[22]
SW2506	CPU start for ARC 770D when <code>CREG_770_START[5:4] = 0x1</code> (boot mode select switch SW2502 set to  during reset), Otherwise, GPIO EXT_PORTA[21]
SW2507	CPU start for AS221 core #2 when <code>CREG_221_START[21:20] = 0x1</code> (boot mode select switch SW2503 set to  during reset), Otherwise, GPIO EXT_PORTA[23]



**Note** Once an ARC core is running, the corresponding `CPU Start` button can be used for application purposes. However, this is not recommended. A `CPU Start` button should only be used for application purposes when the corresponding core is not configured to start after pressing a button, but to start autonomously via a CREG access or debugger.



Figure 36 Location of the CPU Start buttons on the ARC SDP Mainboard.

### 5.8.3 Usage of the Mainboard LEDs

The ARC SDP Mainboard includes eight CPU LEDs that are controlled by the GPIO peripheral of the AXC001 Processor IC. These green LEDs are ON when the corresponding control bit is cleared to 0 and OFF when the control bit is set to 1. Table 22 lists the control bits of the CPU LEDs and Figure 37 shows the LED positions on the ARC SDP Mainboard.


 **Note** The ARC SDP Mainboard features another 8 GPIO LEDs, which are controlled by the GPIO peripheral in the Mainboard's FPGA.

Table 22 Control bits of the CPU LEDs on the ARC SDP Mainboard

Control Bit	Description
SWPORTA_DR[0]	Controls LED2501
SWPORTA_DR[1]	Controls LED2502
SWPORTA_DR[5]	Controls LED2503
SWPORTA_DR[6]	Controls LED2504
SWPORTA_DR[10]	Controls LED2505
SWPORTA_DR[11]	Controls LED2506
SWPORTA_DR[15]	Controls LED2507
SWPORTA_DR[16]	Controls LED2508

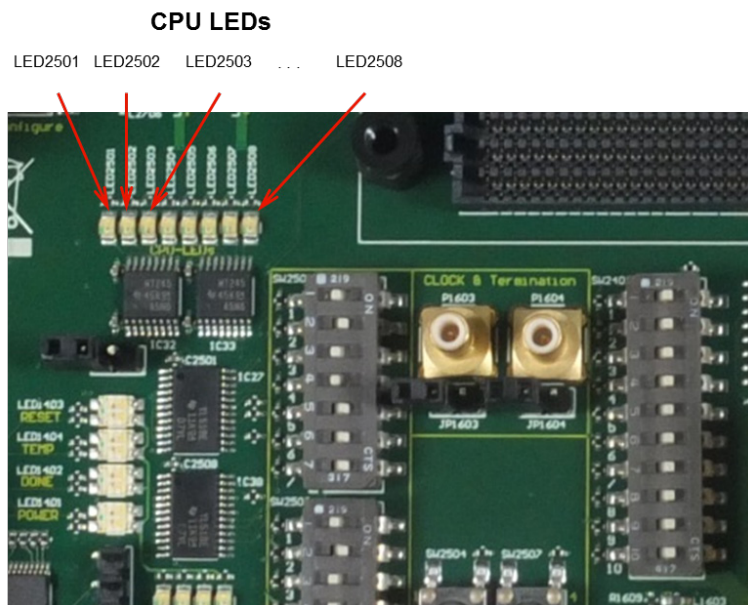


Figure 37 Location of the CPU LEDs on the ARC SDP Mainboard

## 6.1 System Memory Map after a Reset

Following a reset, the memory maps of all AXI masters on the AXC001 CPU Card and on the ARC SDP Mainboard must be set up according to the application requirements. Control registers allow programming the system memory map individually for all AXI masters on the AXC001 CPU Card and on the ARC SDP Mainboard. These control registers are located in the local peripheral area of the AXC001 CPU Card and in the peripheral area of the Mainboard. Following a reset, all CPU cores on the AXC001 CPU Card can access the Mainboard peripherals in the address range `0xE000_0000` to `0xEFFF_FFFF` and the AXC001 CPU Card peripherals in the address range `0xF000_0000` to `0xFFFF_FFFF`.

Additionally, the DDR2 SDRAM needs to be initialized before it can be used.

The Pre-Bootloader programs the corresponding address decoders and, thus, provides the default memory map. It also initializes the DDR2 SDRAM.

The default memory map programmed by the Pre-Bootloader is described in the “[System Memory Map after Pre-Bootloader Execution](#)” section.

If needed, alternative memory maps can be programmed for each AXI master individually by altering the settings of the corresponding control registers. The sections “[Controlling the Memory Map](#)” and “[Software Interfaces](#)” provide additional information.

## 6.2 System Memory Map after Pre-Bootloader Execution

The Pre-Bootloader sets up the memory maps of all AXI masters on the AXC001 CPU Card and on the ARC SDP Mainboard as shown in [Table 23](#). The memory map has been chosen to be identical for all AXI masters. Refer to “[Example Register Settings for the Default Memory Map](#)” for information on the corresponding register settings.

*Table 23 Memory map after Pre-Bootloader execution*

Master Address	Selected Slave	Slave Address
0xFFFF_FFFF 0xF000_0000	AXI2APB on AXC001 CPU Card	0x0FFF_FFFF 0x0000_0000
0xEFFF_FFFF 0xE000_0000	AXI2APB on Mainboard	0xEFFF_FFFF 0xE000_0000
0xDFFF_FFFF 0xD000_0000	AXI Tunnel Slave for HAPS System <sup>1)</sup>	0xDFFF_FFFF 0xD000_0000
0xCFFF_FFFF 0xC000_0000	Unused	
0xBFFF_FFFF  0xA000_0000	DDR2 SDRAM (port 1)	0x1FFF_FFFF  0x0000_0000
0x9FFF_FFFF  0x8000_0000	DDR2 SDRAM (port 0)	0x1FFF_FFFF  0x0000_0000
0x7FFF_FFFF    0x3000_0000	Unused	
0x2FFF_FFFF 0x2000_0000	SRAM on AXC001 CPU Card	0x0FFF_FFFF 0x0000_0000
0x1FFF_FFFF 0x1000_0000	Pre-Bootloader RAM on Mainboard	0x0FFF_FFFF 0x0000_0000
0x0FFF_FFFF 0x0000_0000	Pre-Bootloader RAM on Mainboard	0x0FFF_FFFF 0x0000_0000

- 1) *The slave address is transparently forwarded to the AXI Tunnel master on the HAPS system. Further address decoding depends on your custom design.*

**Note**

The memory map shown in Table 23 is an aggregate of the individual memory map settings on the AXC001 CPU Card and the ARC SDP Mainboard, where the AXI Tunnel between the AXC001 CPU Card and the ARC SDP Mainboard has been abstracted away.

**Note**

A total range of 1 GByte of the aperture is used for DDR2 SDRAM. The AXC001 CPU Card includes 512 MByte of DDR2 SDRAM. This means that apertures 8 and 10 access the same memory addresses. Similarly, apertures 9 and 11 access the same memory addresses.

**Note**

For core #1 of the AS221, the 256 KByte memory range from address `0x0010_0000` to `0x0013_FFFF` is reserved for internal peripherals of the audio subsystem, which are connected to the DMP bus of that core. This is applicable for the LD/ST interface of the core. Therefore, the LD/ST interface of core #1 of the AS221 cannot access this segment of the system memory. However, the instruction fetch interface of the core always accesses the system memory.

**Note**

The ARC EM6 core has internal ICCM and DCCM memories. The locations of these memories depend on register settings in the ARC EM6 core. The pre-boot loader keeps the ICCM at its reset address `0x0000_0000`, but moves the DCCM base address to `0xC000_0000`. Therefore, the ARC EM6 core can only access the RAM on the ARC SDP Mainboard using `0x1000_0000` as the base address. The other cores can access the RAM either using `0x0000_0000` or `0x1000_0000` as the base address. Either start address supports accessing the entire RAM.

**Note**

The SRAM on the AXC001 CPU Card has a size of 128 KByte. The lower 128 KByte within the master 256 MByte aperture access the SRAM, the remaining 262016 KByte are not accessible.

**Note**

The Pre-Bootloader RAM on the ARC SDP Mainboard has a size of 256 KByte. The lower 256 KByte within the master 256 MByte aperture access the RAM, the remaining 261888 KByte are not accessible.

## 6.3 Controlling the Memory Map

### 6.3.1 Setting up the AXI Masters on the AXC001 CPU Card

Control registers are available for each AXI master (i.e. for each core and for the AXI tunnel) to customize its memory map. The full 4 GByte AXI memory map is partitioned into 16 equally sized address apertures of 256 MByte:

- aperture[0]: base address is 0x0000\_0000
- aperture[1]: base address is 0x1000\_0000
- aperture[2]: base address is 0x2000\_0000
- ...
- aperture[15]: base address is 0xF000\_0000

The address map configuration consists of two steps for each 256 MByte aperture within the AXI address space of an AXI master.

First, a target slave is selected from the list shown in [Table 24](#). Then, the desired address offset within the memory map of the target slave is programmed. This offset can be selected in steps of 256 MByte. The specified offset refers to the address offset within the target slave only, i.e. the base address of the aperture is not taken into account.

Table 24 AXC001 CPU Card target slaves

Slave Number	Target Slave
0	No slave selected (default slave provides response)
1	DDR controller port 0
2	SRAM controller
3	AXI tunnel
4	ARC EM6 ICCM
5	ARC EM6 DCCM
6	AXI2APB bridge
7	DDR controller port 1

The AXI tunnel slave transparently forwards the received address to the corresponding AXI tunnel master on the ARC SDP Mainboard. The address map as seen by this master is defined by control registers of the Mainboard. For apertures selecting the AXI Tunnel, it is recommended to set the address offset such that the address issued by the master on the other side and the original address are identical. This can be achieved by setting the `SLV_OFFSET` field of the corresponding register to the aperture number.

The memory map as seen by the AXI2APB bridge is described in the [“Memory Map of the Local Peripherals”](#) section.



### 6.3.2 Setting up the AXI Masters on the ARC SDP Mainboard

The address map of the AXI masters on the Mainboard is defined in a similar way as described in the “[Setting up the AXI Masters on the AXC001 CPU Card](#)” section above.

[Table 25](#) lists the target slaves that are available on the ARC SDP Mainboard. Refer to the *ARC SDP Mainboard User Guide [7]* for more details.

*Table 25 ARC SDP Mainboard target slaves*

Slave Number	Target Slave
0	No slave selected (default slave provides response)
1	TUNNEL0 (AXI tunnel to/from AXC001 CPU Card)
2	TUNNEL1 (AXI tunnel to/from HAPS System)
3	SRAM controller (for Mainboard RAM)
4	AXI2APB (control/status interface of peripherals)

The AXI Tunnel slaves transparently forward the received address to the corresponding AXI Tunnel masters on the AXC001 CPU Card or the HAPS system. For TUNNEL0, this address is then decoded according to the memory map programmed for the AXI Tunnel master on the AXC001 CPU Card. For TUNNEL1, the address issued by the AXI Tunnel master on the HAPS system is decoded according to your custom design.

### 6.3.3 Example Register Settings for the Default Memory Map

The Pre-Bootloader programs the memory map as shown in [Table 26](#) and [Table 27](#).

*Table 26 Default memory map programming for all master on the AXC001 CPU Card*

Aperture #	Master Address	SLV_SEL	SLV_OFFSET	Selected Slave	Slave Address
15	0xFFFF_FFFF 0xF000_0000	6	0x0	AXI2APB	0x0FFF_FFFF 0x0000_0000
14	0xEFFF_FFFF 0xE000_0000	3	0xE	AXI Tunnel	0xEFFF_0000 0xE000_0000
13	0xDFFF_FFFF 0xD000_0000	3	0xD		0xDFFF_FFFF 0xD000_0000
12	0xCFFF_FFFF 0xC000_0000	0	0x0	Unused	
11	0xBFFF_FFFF 0xB000_0000	7	0x1	DDR2 SDRAM (port 1)	0x1FFF_FFFF
10	0xAFFF_FFFF 0xA000_0000	7	0x0		0x0000_0000
9	0x9FFF_FFFF 0x9000_0000	1	0x1	DDR2 SDRAM (port 0)	0x1FFF_FFFF
8	0x8FFF_FFFF 0x8000_0000	1	0x0		0x0000_0000
7	0x7FFF_FFFF 0x7000_0000	0		Unused	
6	0x6FFF_FFFF 0x6000_0000	0			
5	0x5FFF_FFFF 0x5000_0000	0			
4	0x4FFF_FFFF 0x4000_0000	0			
3	0x3FFF_FFFF 0x3000_0000	0			
2	0x2FFF_FFFF 0x2000_0000	2	0x0	Local SRAM	0x0FFF_FFFF 0x0000_0000
1	0x1FFF_FFFF 0x1000_0000	3	0x1	AXI Tunnel	0x1FFF_FFFF
0	0x0FFF_FFFF 0x0000_0000	3	0x0		0x0000_0000

The slave address of the AXI Tunnel on the AXC001 CPU Card is transparently forwarded to the AXI TUNNEL0 master of the ARC SDP Mainboard and becomes the address issued by this master. This master address is then decoded according to the memory map of the AXI TUNNEL0 master on the Mainboard as shown in [Table 27](#). For example, if a core issues the address 0xE000\_0000, the AXI Tunnel is selected. The AXI Master on the other side of the tunnel issues the address 0xE000\_0000 and thus selects the AXI2APB bridge of the Mainboard peripheral area.

Table 27 Default memory map programming for all masters on the ARC SDP Mainboard

Aperture #	Master Address	SLV_SEL	SLV_OFFSET	Selected Slave	Slave Address
15	0xFFFF_FFFF 0xF000_0000	1	0xF	TUNNEL0 (AXC001 CPU Card)	0xFFFF_FFFF 0xF000_0000
14	0xEFFF_FFFF 0xE000_0000	4	0x0	AXI2APB (Peripherals)	0x0FFF_0000 0x0000_0000
13	0xDFFF_FFFF 0xD000_0000	2	0xD	TUNNEL1 (HAPS System)	0xDFFF_0000 0xD000_0000
12	0xCFFF_FFFF 0xC000_0000	0	0x0	Unused	
11	0xBFFF_FFFF 0xB000_0000	1	0xB	TUNNEL0 (AXC001 CPU Card)	0xBFFF_FFFF 0xB000_0000
10	0xAFFF_FFFF 0xA000_0000	1	0xA		0xAFFF_FFFF 0xA000_0000
9	0x9FFF_FFFF 0x9000_0000	1	0x9		0x9FFF_FFFF 0x9000_0000
8	0x8FFF_FFFF 0x8000_0000	1	0x8		0x8FFF_FFFF 0x8000_0000
7	0x7FFF_FFFF 0x7000_0000	0	0x0	Reserved	
6	0x6FFF_FFFF 0x6000_0000	0	0x0		
5	0x5FFF_FFFF 0x5000_0000	0	0x0		
4	0x4FFF_FFFF 0x4000_0000	0	0x0		
3	0x3FFF_FFFF 0x3000_0000	0	0x0		
2	0x2FFF_FFFF 0x2000_0000	0	0x0		
1	0x1FFF_FFFF 0x1000_0000	3	0x0	RAM (Mainboard)	0x0FFF_FFFF 0x0000_0000
0	0x0FFF_FFFF 0x0000_0000	3	0x0	RAM (Mainboard)	0x0FFF_FFFF 0x0000_0000

The slave address of the AXI TUNNEL0 slave on the ARC SDP Mainboard as listed in [Table 27](#) is transparently forwarded via the AXI Tunnel and becomes the address issued by the AXI Tunnel master on the AXC001 CPU Card. It is then decoded according to the memory map programmed for the AXI Tunnel master on the AXC001 CPU Card, which is shown in [Table 26](#).

Likewise, the slave address of the AXI TUNNEL1 slave on the ARC SDP Mainboard is forwarded to the AXI Tunnel master on the HAPS system. It is then decoded according to your custom design.

## 6.4 Memory Map of the Local Peripherals

All local APB peripherals inside the AXC001 Processor IC are mapped into the AXI2APB segment of the system memory map, which has the default base address `0xF000_0000`. [Table 28](#) shows the address offsets of the individual peripherals and the corresponding aperture within the AXI2APB section. This means that the address offset listed in [Table 28](#) has to be added to the base address of the AXI2APB section to obtain the correct base address (to be used by the master) of the peripheral.

**Example:** If the AXI2APB segment is located at its default address `0xF000_0000`, then the CGU base address within the memory map of the master is `0xF000_0000`.

*Table 28 Peripheral memory map*

Peripheral	Address Offset	Aperture [Bytes]	Description
CGU	<code>0x0000_0000</code>	4096	Clock Generation Unit
CREG	<code>0x0000_1000</code>	4096	Control Registers
ICTL	<code>0x0000_2000</code>	128	Interrupt Controller
GPIO	<code>0x0000_3000</code>	128	General Purpose I/O
MCTL	<code>0x0000_4000</code>	4096	Control interface of the DDR memory controller

*This chapter is intended for programmers of the AXC001 CPU Card. It includes an overview of the example applications provided with the AXC001 CPU Card and explains how to use the AXS101 Software Development Platform for software development.*

## 7.1 Supported Tools and Operating Systems

For an overview of the supported tools and operating systems refer to the release notes at the *ARC SDP download webpage [6]*.

## 7.2 Boot Modes

### 7.2.1 Common Boot Modes

All the ARC cores on the AXC001 Processor IC are configured for “HALT-after-reset”. Hence, after reset the ARC cores go into the HALT state and must be started explicitly before they will start executing the boot code. Each of the ARC cores can be started individually in one of the four following ways:

- starting the ARC core with the debugger
- starting the ARC core via a CPU Start button on the Mainboard (HW)
- starting the ARC core via a CREG register bit (SW)
- starting the ARC core autonomously after reset

When an ARC core is started, it starts fetching instructions from the reset vector location. The default reset vector locations for the ARC cores are as follows: <sup>1</sup>

- |                     |             |
|---------------------|-------------|
| • ARC 770D          | 0x0000_0000 |
| • ARC AS221 core #1 | 0x0000_0000 |
| core #2             | 0x0000_0400 |
| • ARC EM6           | 0x1000_0000 |

To ensure maximum flexibility, the ARC cores can boot from different boot sources and different locations within a certain boot source. For this purpose each 256 MByte aperture of

<sup>1</sup> the reset vector address is programmable at run time through the INT\_VECTOR\_BASE register, and may be set to any 1KB aligned address.

the memory map can be designated to serve as a so-called *boot mirror*. A detailed description of the boot mirror and its configurability options is given in the “[Default Boot Mode Settings on the ARC SDP Mainboard](#)” section. When enabled, the boot mirror overrules the memory map that would normally be visible for this aperture and instead a different part of the memory map will be reflected. The remapped part of the memory map is referred to as a ‘*mirror*’.

For AXC001 Processor IC, the following basic boot scenarios are anticipated:

#### boot with debugger

- 1) Select boot mode, source and location via the CPU DIP switches on the Mainboard.
  - Boot mode is “start ARC core with the debugger”
  - Boot mirror is set to DDR2 SDRAM or local SRAM
  - Boot location is set according to the location of the image
- 2) Push the `RESET` button on the Mainboard
- 3) Download code into the selected boot source with the debugger
- 4) Start the ARC core with the debugger

#### boot without debugger

- 2) Select boot mode, source and location via the CPU DIP switches on the Mainboard
  - Boot mode is “start ARC core by `CPU Start` button”
  - Boot mirror is set to “AXI tunnel”
  - Boot location is set according to the location of the image
- 3) Push the `RESET` button on the Mainboard
- 4) Download code into the configured boot source (i.e. into the SPI Flash)
- 5) Start the ARC core by pushing the corresponding `CPU Start` button on the Mainboard

---

## 7.2.2 ARC EM6 Booting from ICCM

The ARC EM6 core supports 64 KByte of Instruction Closely Coupled Memory (ICCM). The ICCM provides the processor with fast and predictably timed access to a fixed-size region of memory, which normally contains program code. After reset, the base address of the ICCM is located at address zero<sup>2</sup>. During program execution the ICCM base address can be re-programmed, and may be mapped to any 256 MByte aligned aperture in the lower half of the memory map. The ARC EM6 core allows other initiators in the system to access its ICCM through an AHB-Lite interface such that another core that boots prior to the ARC EM6 can copy the EM6 boot code to the ICCM and then start the EM6 by writing to the

---

<sup>2</sup> for ARC EM6 v1.0 the default ICCM base address is always at address zero; for ARM EM6 v.1.1 the default ICCM base address is configurable

corresponding control register. That other core can be located on the AXC001 CPU Card or on a HAPS extension board connected to the ARC SDP Mainboard.

Besides the basic boot scenarios “boot with debugger” and “boot without debugger” described in the “[Common Boot Modes](#)” section, the ARC EM6 also supports the following boot option:

#### boot from ICCM with debugger

- 1) Select boot mode, mirror and location via the CPU DIP switches on the Mainboard
  - Boot mode is “start ARC core with the debugger”
  - Boot mirror is arbitrary
  - Boot location is set according to the location of the image
- 2) Push the `RESET` button on the Mainboard
- 3) Download code into the ICCM with the debugger
- 4) Re-program the reset vector to address zero (i.e. reset vector now points to ICCM)
- 5) Start the ARC core with the debugger

---

## 7.3 Pre-Boot

### 7.3.1 Pre-Boot Overview

The AXS101 Software Development Platform includes a set of Pre-Bootloaders, which are functionally identical but compiled for the different cores. The Pre-Bootloader performs two main tasks:

- Board initialization
- Loading an image

The Pre-Bootloaders are included in the bit-file for the FPGA of the ARC SDP Mainboard. A RAM, which is implemented in this FPGA, gets initialized with the Pre-Bootloader and is then used as boot memory.

Booting with the Pre-Bootloader makes use of the boot mirror mechanism and requires that the CPU DIP Switches on the Mainboard are set as shown in [Figure 38](#):

- The “`boot mirror select`” switches need to be set to “Pre-Bootloader RAM on Mainboard (via the AXI tunnel)”.
- The settings for the “`boot location select`” switches are core specific and depend on the location of the Pre-Bootloader image within the RAM and on the reset vector of the core.

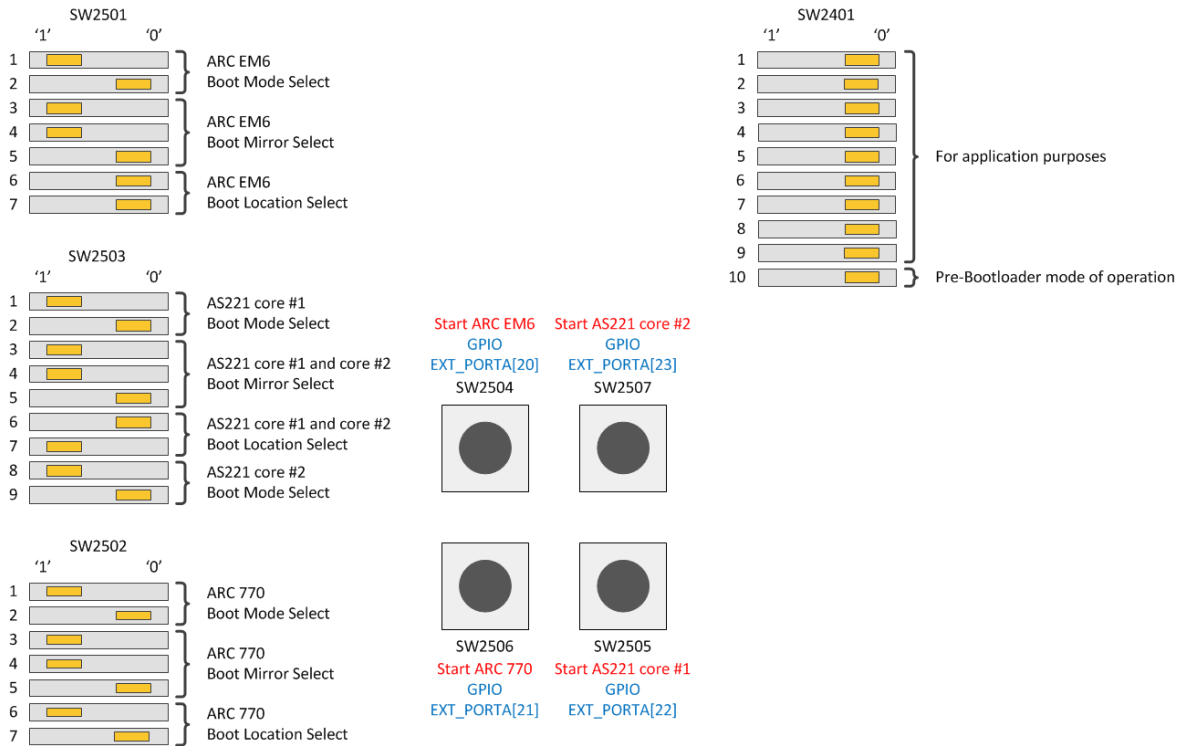


Figure 38 Default settings of the DIP switches on the ARC SDP Mainboard.

During board initialization, Pre-Bootloader programs the clock dividers and the system memory map and initializes the DDR2 SDRAM.

The Pre-Bootloader supports loading an image from the SPI-flash on the ARC SDP Mainboard into the SRAM or the DDR2 SDRAM memory of the AXC001 CPU Card. See the “[Building Baremetal Application](#)” section for instructions on creating the image and for storing the image in the SPI Flash.

Image loading can be bypassed (SW2401: 10 ). In this case, Pre-Bootloader does not load any image, but only performs a board initialization and sets the ARC core into the HALT state.



**Note** This operation mode is useful if you want to load your application using the debugger, but want the board to be initialized automatically. In that case, set the `Boot Mirror Select` switches to “Pre-Bootloader RAM on Mainboard”, set the `Boot Mode Select` bits to “Autonomously” and by-pass the image loading. See “[Usage of the Mainboard DIP Switches](#)” for the exact switch settings.

If image loading is not bypassed (SW2401: 10 ), then the last instruction of the Pre-Bootloader is a `JUMP` to the start address of the loaded image, such that the loaded application starts execution.

As an example, [Figure 39](#) shows how the Pre-Bootloader loads an image from the SPI-Flash on the ARC SDP Mainboard to the DDR2 SDRAM on the AXC001 CPU Card.



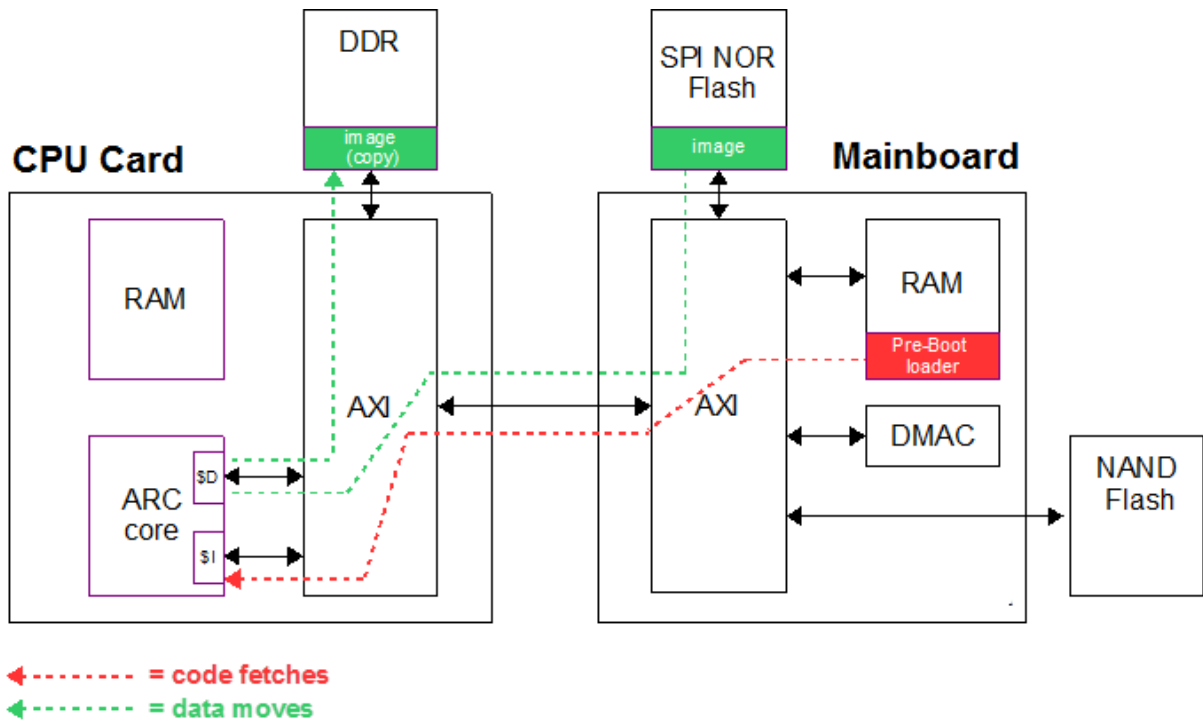


Figure 39 Pre-Boot mechanism

The Pre-Bootloader uses the seven-segment displays on the Mainboard to show status information. The left character shows the CPU number that is also used in the debugger as listed in Table 29. If loading of an image is bypassed (SW2401: 10 ), then a dot is displayed next to the CPU number

The right character shows an error code as explained in Table 30.

Table 29 Meaning of the left character of the seven-segment display

CPU Number	ARC core	Pre-Bootloader Mode
1	AS221 core #1	image loaded and code execution started
1 .		no image loaded
2	AS221 core #2	image loaded and code execution started
2 .		no image loaded
3	ARC EM6	image loaded and code execution started
3 .		no image loaded
4	ARC 770D	image loaded and code execution started
4 .		no image loaded
blank		Pre-Bootloader not yet executed. Check the DIP switch settings and press a CPU start button.

Table 30 Meaning of the right character of the seven-segment display

Error Code	Description
0	No error
1	DDR2 SDRAM initialization error
2	SPI initialization error: SPI IP is not detected.
3	CGU lock error
4	SPI-Flash error: Cannot read the Flash ID
5	SPI-Flash error: Problem reading data
6	SPI-Flash error: Incorrect Flash ID
7	SPI-Flash error: Cannot select next 128 Mbit segment (Problem related to 4 <sup>th</sup> address byte)
8	No valid image in SPI Flash for the ARC core running the Pre-Bootloader
9	CRC error: after copying the image to the target memory location the calculated CRC differs from the one in the header.
blank	When the right character is blank (off) but the left character shows a digit, then the Pre-Bootloader hangs. Please push the RESET button again.

Additional board settings are performed by the `board_init()` function from the file `/software/baremetal/board/axs101/src/board_axs101.c`, which should be executed at the start of every application.

## 7.4 U-Boot

Besides the Pre-Bootloader described in the “Pre-Boot” section, the open source U-Boot bootloader is available for use with the AXC001 CPU Card. U-Boot for ARC is available from <https://github.com/foss-for-synopsys-dwc-arc-processors/u-boot>. A pre-built U-Boot binary is part of the AXS101 Software Development Platform software package `axs101_software_<version>.zip` in two forms:

- `u-boot.elf`: an elf executable that can be loaded and started with the MetaWare debugger
- `u-boot.bin`: a binary version that can be stored in the SPI flash memory on the ARC SDP Mainboard. Refer to “[Programming U-Boot in the SPI Flash](#)” below for instructions.

After unzipping the software package you can find these files in the `/software/axs101_uboot_<u-boot_version>/` folder. The start address of the pre-built binary is `0x8FD_0000`, which is located in the DDR2 SDRAM.

The U-Boot bootloader is built for and should be executed on the ARC 770D core of the AXC001 CPU Card. The U-Boot version for the AXC001 CPU Card supports loading custom images from a FAT32 formatted SD-card, and starting the loaded image.

The main use case for U-Boot in the AXS101 SDP context is loading a Linux image (from the SD card, via Ethernet, or from the on-board NAND flash memory) and boot Linux on the ARC 770D (see “[Starting U-Boot on the ARC 770D Core](#)” below).

Booting – when not using the debugger - is a 2-stage process, which requires that the U-Boot image `u-boot.bin` is stored in the SPI-flash memory on the ARC SDP Mainboard. As a first step the Pre-Bootloader (see “[Pre-Boot](#)” section) loads U-Boot into the DDR2 SDRAM on the AXC001 CPU Card. Next, the CPU core executes the U-Boot as the secondary bootloader.

[Figure 40](#) shows the DIP switch settings on the ARC SDP Mainboard for starting U-Boot on the ARC 770D core automatically after reset. As an alternative, [Figure 41](#) shows the DIP switch settings for starting U-Boot on the ARC 770D after pushing the button SW2506.

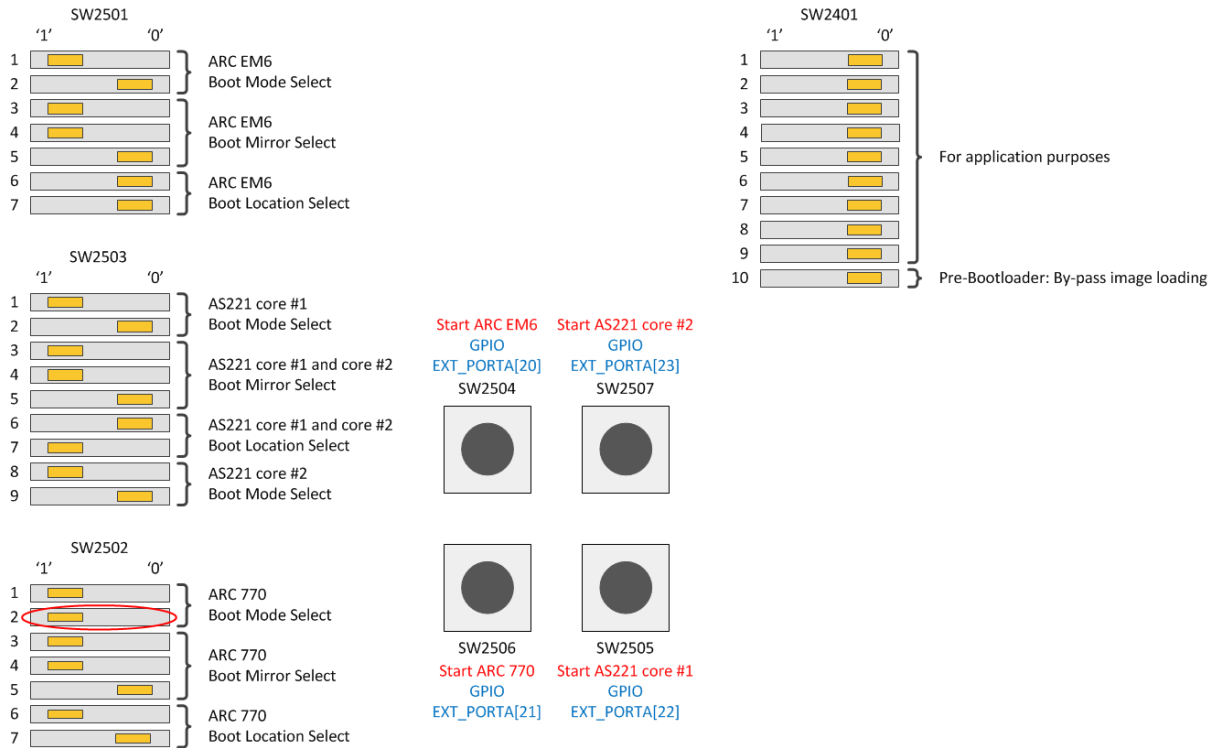


Figure 40 DIP switch settings for auto-starting U-Boot

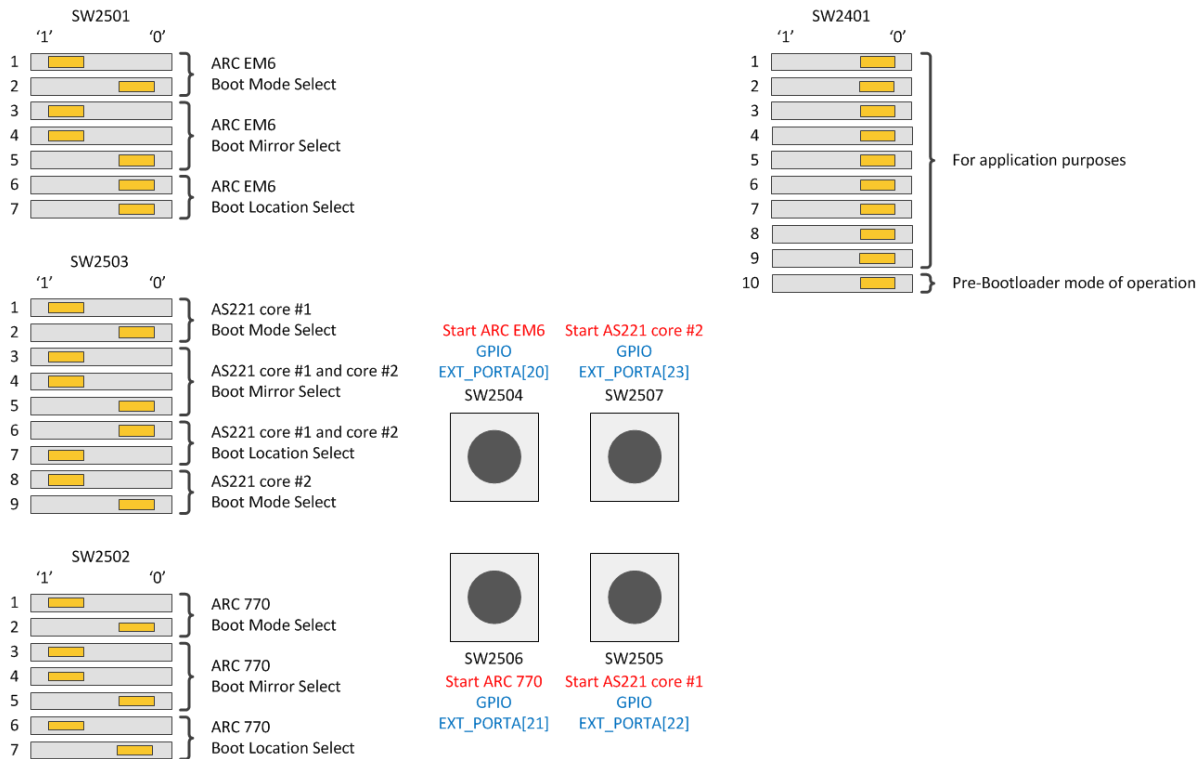


Figure 41 DIP switch settings for manually starting U-Boot by pushing SW2506

## 7.4.1 Programming U-Boot in the SPI Flash

Execute the following steps for programming U-Boot in the SPI Flash on the Mainboard:

1. *[if this have not been done previously]*  
Download and unzip the `axs101_software_<version>.zip` file from the ARC SDP download webpage [6] and install the `axs_comm` tool as described in the *ARC SDP Mainboard User Guide* [7].
2. Navigate to the `/software/axs101_uboot_<u-boot_version>` folder and double-click on `axs_comm_program_uboot.bat`. It stores the file `u-boot.bin` starting at sector 0, page 0 of the SPI Flash.



**Note** Executing `axs_comm_program_uboot.bat` erases the content of sectors 0 to 3 of the SPI Flash device on the Mainboard. By default, these first 4 sectors contain the self-tests. For instructions on how to restore self-tests, refer to the “[Restoring the Self-Tests in the SPI Flash](#)” section.

## 7.4.2 Starting U-Boot on the ARC 770D Core

Follow the steps below to start U-Boot on the ARC 770D core:

1. Install and configure PuTTY on your PC according to the instructions in “[Appendix B](#)”.
2. Connect your PC to the ARC SDP Mainboard via the Mainboard’s USB Dataport.
3. Switch the ARC SDP Mainboard ON.
4. *[if this have not been done previously]* Store the U-Boot image `u-boot.bin` in the SPI Flash of the ARC SDP Mainboard according to the instructions in the “[Programming U-Boot in the SPI Flash](#)” section.
5. Set the DIP switches on the Mainboard according to [Figure 40](#) above.
6. *[Optional]* If you want to boot Linux, copy the uBoot image to a FAT32 formatted SD-card and insert the card in the SD-card slot on the ARC SDP Mainboard.
7. Start a PuTTY terminal and configure it to use the COM port connected to the USB Dataport of the ARC SDP Mainboard. Set the **Connection type** to `Serial` and the **Speed** to `115200`.

8. Push the `RESET` button on the ARC SDP Mainboard.

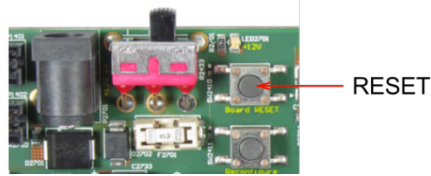


Figure 42 Location of the `RESET` button on the Mainboard

9. [Optional] If you want to boot Linux enter the following commands at the U-Boot prompt `AXS#`:

```
fatload mmc 0
bootm
```

Wait for the Linux login prompt and login as root (empty password).

### 7.4.3 Other Supported U-Boot Features

U-Boot is a versatile bootloader, with many features including defining aliases/scripts, scripting of boot sequences and programming the on-board NAND-Flash memory. Please refer to <http://www.denx.de/wiki/U-Boot/WebHome> for a detailed description of U-Boot.

This release of U-Boot for the AXC001 CPU Card supports the following specific features:

- SD-card support
- NAND-Flash support
- Ethernet support (incl. dhcp/bootp protocols for downloading images from a server using the tftp protocol)
- I<sup>2</sup>C EEPROM for storing environment variables

U-Boot includes a 'help' command that lists all supported commands. Besides the `fatload` and `bootm` commands described earlier, the following commands are regularly used:

- `setenv` – set environment variables
- `saveenv` – save environment variables to persistent storage
- `run` – run commands in an environment variable
- `bootp` – boot image via network using BOOTP/TFTP protocol
- `cp` – memory copy
- `flinfo` – print flash memory information
- `erase` – erase flash memory
- `mtddparts` – define a Linux compatible MTD partition scheme

---

## 7.5 Drivers

The AXS101 Software Development Platform includes drivers for baremetal applications, MQX and Linux.

For a list of available drivers refer to the release notes at the *ARC SDP download webpage [6]*.

---

### 7.5.1 Drivers for Baremetal Applications

Drivers for baremetal applications are included in the file `axs101_software_<version>.zip`, which is available at the *ARC SDP download webpage [6]*. Download and unzip this file. The drivers are located in the `/software/baremetal/io` sub-directory.

Within the `/io` directory, the sub-directories with the `_axs1xx` postfix contain specific drivers for IP located in the peripheral subsystem on the ARC SDP Mainboard. The sub-directories with the `_axc001` postfix contain specific drivers for peripherals on the AXC001 CPU Card. The remaining directories contain generic drivers suited for peripherals located on the ARC SDP Mainboard or a CPU Card (or both).

Each driver directory contains a `makefile` and two sub-directories: `/inc` and `/src`. They contain *include* files and driver source code respectively.

Refer to the “[Building Baremetal Application](#)” section for more details about the remaining content of this zip file.

---

### 7.5.2 Drivers for MQX

Peripheral drivers are included in the pre-built MQX library, which is contained in the file `axs101_software<version>.zip`. This zip file is available on the *ARC SDP download webpage [6]*. After unzipping this file, you can find MQX-related files in the `/software/mqx<mqx_version>` directory. Some examples for using MQX peripheral drivers are included in the zip file as well; see the “[MQX Package](#)” section. A detailed description of the drivers is contained in the documentation provided with the MQX product.

---

### 7.5.3 Drivers for Linux

For most peripherals drivers are available in the Linux kernel upstream [4]. Before new drivers appear upstream development is done on the ARC Linux github site [5].

To configure the Linux kernel for an AXS101 Software Development Platform, please use the `axs101_defconfig` configuration file available at the ARC Linux github site [5].

The pre-built Linux distribution included in the file `axs101_software_<version>.zip` contains drivers for Linux. This zip file is available on the *ARC SDP download webpage [6]*. After unzipping this file you can find the Linux distribution in the `/software/axs101_linux_<linux_version>` folder.

## 7.6 Baremetal Package

Before using the baremetal package, make sure that you have installed the MetaWare toolchain (compiler / linker / debugger). This is a separate product, which is not part of the AXS101 Software Development Platform package.

### 7.6.1 Overview

The baremetal package is part of the zip file `axs101_software_<version>.zip`, which is available at the *ARC SDP download webpage [6]*. The package includes baremetal example applications, peripheral drivers and the corresponding makefiles. Unzip the file and change to the folder `/software/baremetal`. The directory structure of this folder is shown in [Table 31](#).

Table 31 Baremetal folder contents

Folder	Description
Root folder path: <code>/software/baremetal</code>	
<code>/apps</code>	This folder contains individual sub-directories for all application examples. Refer to the release notes on the ARC SDP download webpage [6] for an overview of the available application examples.
<code>/board</code>	This folder contains board-specific <i>include</i> files. Particularly, it contains linker files that include the definition of the memory map, located in the <code>/board/axs101/src/</code> folder. Two linker files are included, <code>map_axs101_ddr.met</code> and <code>map_axs101_ram.met</code> . They are used for building the code for the DDR2 SDRAM or local SRAM respectively.
<code>/inc</code>	This folder contains a general <i>include</i> file for type definitions
<code>/io</code>	This folder contains all basic drivers for the AXS101 Software Development Platform. See the “ <a href="#">Drivers for Baremetal Applications</a> ” section for more details.
<code>/project</code>	This folder contains the files related to the “gmake” build flow. In particular: <ul style="list-style-type: none"> <li><code>rules.mk</code> contains all makefile rules</li> <li><code>options.mk</code> contains more compiler/linker related options</li> </ul>
<code>/project_arcmw</code>	This folder contains files related to the MetaWare IDE flow.
<code>build.bat</code>	This batch script can be used to build the baremetal applications using the “gmake” build flow. See the “ <a href="#">Building Baremetal Applications Using gmake</a> ” section for more details.



## 7.6.2 Building Baremetal Applications Using the MetaWare IDE

The files belonging to the example projects have to be imported in the Eclipse IDE workspace. The instructions below illustrate this process step-by-step:

1. Run MWIDE and select `/software/baremetal/project_arcw` as current workspace.

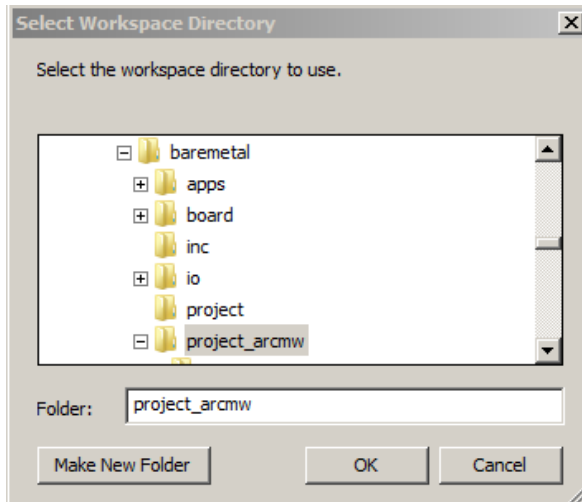


Figure 43 MetaWare IDE - Select Workspace Directory

2. Open the workspace, and select “**File – Import**” from the top menu. Expand the “**General**” folder. Then select “**Existing Projects into Workspace**” and press the “**Next**” button

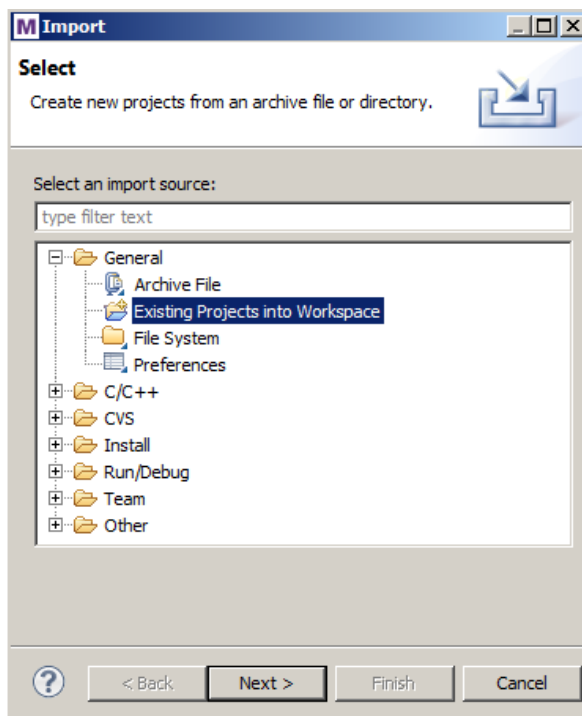


Figure 44 MetaWare IDE – Importing existing projects

3. Select your “/software/baremetal/project\_arcmw” folder as root directory and select all projects available there for import. Then press the “**Finish**” button to import the example projects into your workspace.
4. IDE loads and displays the example projects in your workspace.
5. Select one or multiple projects (excluding the `common` project) and then perform a right-click.
6. Select “**Build Configurations > Set Active**” and select your build target, for example **arcem6**.

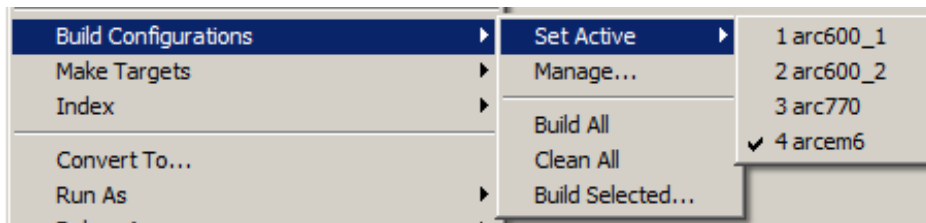


Figure 45 MetaWare IDE - Set Active Build Configurations

This step selects the `arcem6` configuration of the project for building. You can choose another option to match your debug target.

7. By default, the project is built for
  - code execution from DDR (AXS\_MEMORY\_TYPE=ddr)
  - console UART via USB dataport (AXS\_CONSOLE\_TPYE=usb\_uart)

If you wish to set other options, you should edit the file `/software/baremetal/project_arcmw/common/axs101/settings.mk` and set the variables `AXS_CONSOLE_TYPE` and `AXS_MEMORY_TYPE` according to your requirements. [Table 32](#) lists the available options. Changes that are made in this file affect all projects in the workspace.

Table 32 Build options

Variable	Value	Description
AXS_MEMORY_TYPE	ddr	Image to be executed from DDR2 SDRAM; default The code is built using the start address 0x8000_0000
	ram	Image to be executed from local SRAM The code is built using the start address 0x2000_0000

Variable	Value	Description
AXS_CONSOLE_TYPE	uart0	Debug console is connected to the UART0 interface at the DB9 connector <sup>1)</sup> or at the Pmod connector <sup>1) 2)</sup>
	uart1	Debug console is connected to the UART1 interface at the 6-pin header <sup>1)</sup> or at the Pmod connector <sup>1) 2)</sup>
	usb_uart	Debug console is connected via the USB Dataport <sup>1)</sup> (using the UART2 peripheral); default
	nouart	No console output

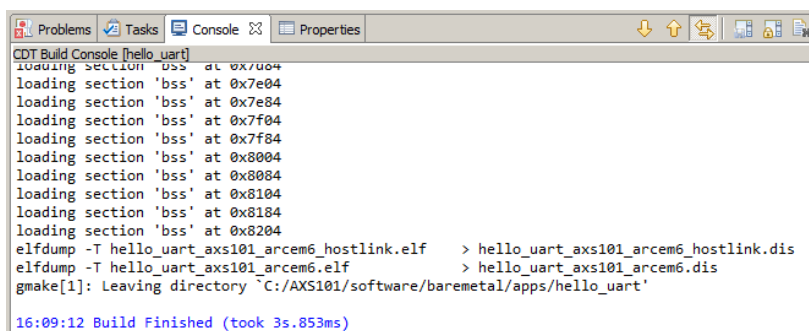
1) This connector is located on the ARC SDP Mainboard.

2) In order to use the Pmod connectors, the `PMOD_MUX_CTRL` register (see the ARC SDP Mainboard User Guide [7]) needs to be modified.

8. Build the project using any of the following methods:

- Right-click the selected project(s) again and select **Build Project**.
- Select **Build Project** from the **Project** menu.
- Enter CTRL-B to build all available projects.

You can see the build results in the console window. An example is shown in Figure 46. Two images get created. Only one of them includes HOSTLINK. The file name of the image with HOSTLINK includes the postfix “\_hostlink”. The images are located in the folder `/software/baremetal/apps/<project_name>`.



```

CDT Build Console [hello_uart]
loading section 'bss' at 0x7004
loading section 'bss' at 0x7e04
loading section 'bss' at 0x7e84
loading section 'bss' at 0x7f04
loading section 'bss' at 0x7f84
loading section 'bss' at 0x8004
loading section 'bss' at 0x8084
loading section 'bss' at 0x8104
loading section 'bss' at 0x8184
loading section 'bss' at 0x8204
elfdump -T hello_uart_axs101_arcem6_hostlink.elf > hello_uart_axs101_arcem6_hostlink.dis
elfdump -T hello_uart_axs101_arcem6.elf > hello_uart_axs101_arcem6.dis
gmake[1]: Leaving directory `C:/AXS101/software/baremetal/apps/hello_uart'
16:09:12 Build Finished (took 3s.853ms)

```

Figure 46 MetaWare IDE – build results in console window

9. For instructions on running the code in the MetaWare IDE Debugger, please refer to the sections “[Hardware Setup for Debugging](#)” and “[Running a Baremetal Application in the MetaWare IDE Debugger](#)”.
10. For instructions on storing the code in the SPI Flash memory of the ARC SDP Mainboard or on an SD-Card, please refer to the sections “[Storing an Image in the SPI Flash and Running the Application](#)” and “[Storing an Image on the SD-Card and Running the Application](#).”

### 7.6.3 Building Baremetal Applications Using gmake

The batch script `build.bat` can be used to build the applications. This script is based on `gmake`. It executes `gmake` passing variables according to the command line options provided. [Figure 47](#) and [Table 33](#) explain the available options of this script.

```
c:\AXS101\software\baremetal>build -h

=====
Usage: build -app <appname> -core    <coretype>
                        -console  <consoletype>
                        -memory   <memorytype>

Arguments:
  <appname>      : application name
  <coretype>     : core type       <arc600_1|arc600_2|arcem6|arc770>
  <consoletype> : console type    <uart0|uart1|usb_uart|none>
  <memorytype>  : memory type     <ram|ddr>
```

Figure 47 Build script options

Table 33 Command line options for `build.bat`

Option	Value	Description
-app		The name of the sub-directory of the <code>/software/baremetal</code> / <code>apps</code> directory, which contains the makefile for the selected application
-core	arc600_1	Select AS221 core #1
	arc600_2	Select AS221 core #2
	arcem6	Select ARC EM6
	arc770	Select ARC 770D; default
-console	uart0	Debug console is connected to the <code>UART0</code> interface at the <code>DB9</code> connector <sup>1)</sup> or at the <code>Pmod</code> connector <sup>1)2)</sup>
	uart1	Debug console is connected to the <code>UART1</code> interface at the 6-pin header <sup>1)</sup> or at the <code>Pmod</code> connector <sup>1)2)</sup>
	usb_uart	Debug console is connected via the <code>USB Dataport</code> <sup>1)</sup> (using the <code>UART2</code> peripheral); default
	nouart	No console output
-memory	ddr	Image to be executed from <code>DDR2 SDRAM</code> , default The code is built using the start address <code>0x8000_0000</code> .
	ram	Image to be executed from local <code>SRAM</code> The code is built using the start address <code>0x2000_0000</code> .

1) This connector is located on the `ARC SDP Mainboard`.

2) In order to use the Pmod connectors, the `PMOD_MUX_CTRL` register (see the ARC SDP Mainboard User Guide [7]) needs to be modified.

To build an application example, open a Windows command prompt, navigate to the `/software/baremetal` folder, and enter the `build` command followed by corresponding options.

The result of the build can be found in the `/software/baremetal/apps/<appname>` folder. The build generates seven types of files:

<code>*.o</code>	Object
<code>*.a</code>	Archive library
<code>*.dis</code>	Disassembly file
<code>*.elf</code>	Elf file for debugger
<code>*.bin</code>	Bin file for SPI-Flash or SD-Card
<code>*.hex</code>	Hex file
<code>*.map</code>	Map file

Two sets of files are generated: One set includes HOSTLINK, the other doesn't. For the set with HOSTLINK the string `"_hostlink"` is appended to the file name.

If the arguments for `-console` or `-memory` differ from the default values, the selected option is included in the filename as demonstrated in the examples below.

### Example 1: Build the selftest application with default arguments

```
build -app selftest
```

This command generates the files:

```
selftest_axs101_arc770.*  
selftest_axs101_arc770_hostlink.*
```

### Example 2: Build the selftest application for a different ARC core

```
build -app selftest -core arcem6
```

This command generates the files:

```
selftest_axs101_arcem6.*  
selftest_axs101_arcem6_hostlink.*
```

### Example 3: Build the selftest application for a different memory and console

```
build -app selftest -memory ram -console uart0
```

This command creates the files:

```
selftest_axs101_ram_uart0_arc770.*
selftest_axs101_ram_uart0_arc770_hostlink.*
```

### 7.6.4 Hardware Setup for Debugging

Follow the steps below to execute an sample application in a debugger:

1. Set the jumpers to their default settings (see the “Jumpers” section). The JTAG interface is in daisy-chained mode.
2. Set the switches on the ARC SDP Mainboard as shown in Figure 48. The change compared to the default settings is marked by a red ellipse.

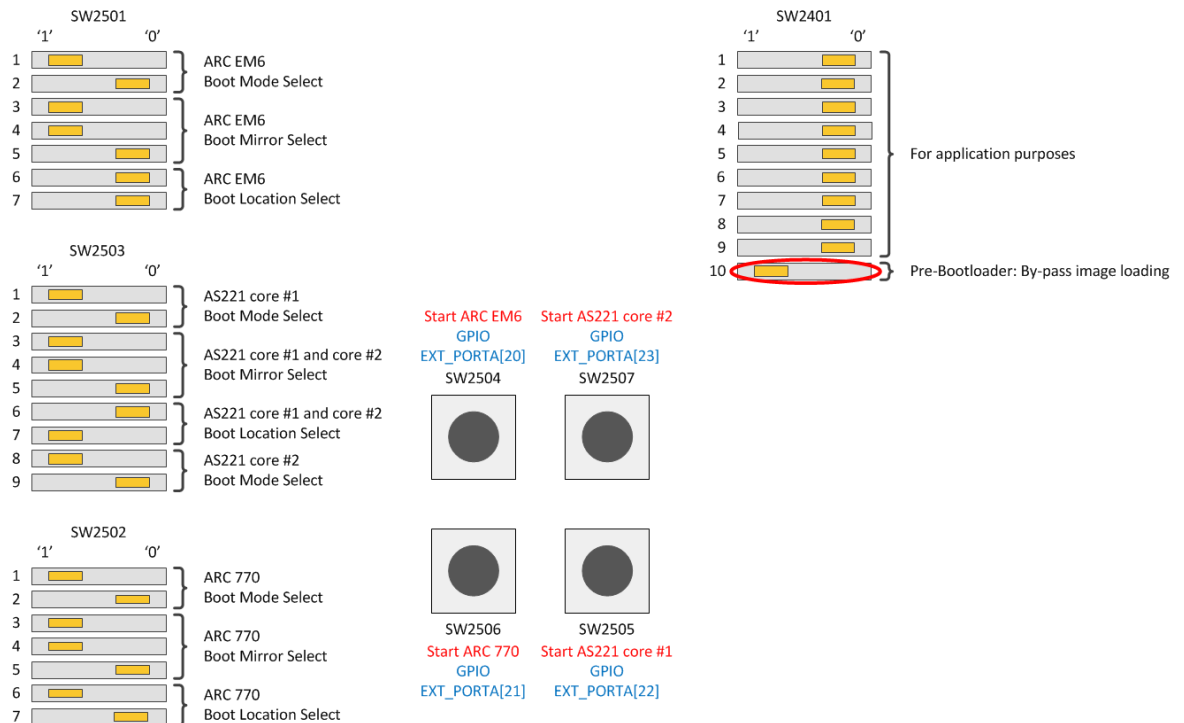


Figure 48 Settings of the DIP switches on the ARC SDP Mainboard for using the debugger

3. Connect the USB Dataport of the ARC SDP Mainboard to your PC.

- If you are using a debug probe rather than the JTAG channel of the USB Dataport, connect the probe to the appropriate connector and remove the jumper JP1402 on the ARC SDP Mainboard. If you are using a Lauterbach probe also remove the jumpers JP2309 and JP2310.

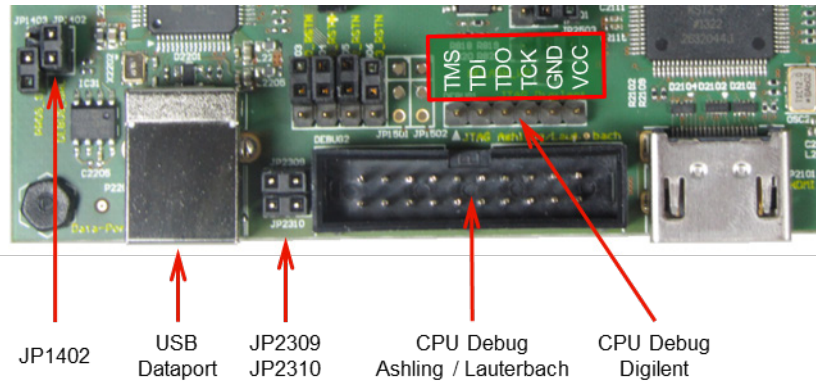


Figure 49 Location of the debug interfaces and the corresponding jumpers

- Switch ON the power supply or push the RESET button

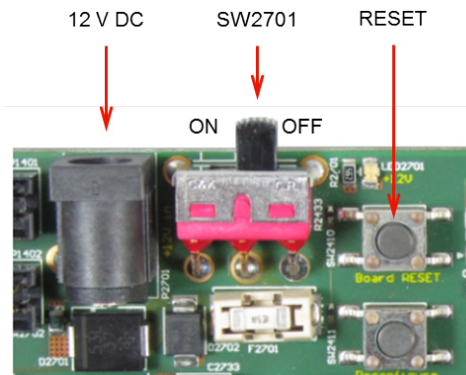


Figure 50 Location of the ARC SDP Mainboard's power supply and power switch

- Push the CPU Start button of your target CPU core according to [Table 34](#). For the location of the CPU Start button see [Figure 51](#). The Pre-Bootloader then automatically initializes the DDR2 SDRAM memory and sets the ARC core frequency correctly, but by-passes loading an image from the SPI Flash.



Figure 51 Location of the CPU Start buttons on the ARC SDP Mainboard.

Table 34 CPU Start buttons and seven-segment display values for running applications in the debugger

ARC Core	Start Button	Seven-Segment Display
ARC 770D	SW2506	4.0
AS221 core #1	SW2505	1.0
AS221 core #2	SW2507	2.0
ARC EM6	SW2504	3.0

When the seven-segment display shows the value listed in [Table 34](#) (note the “dot” between the two digits) the AXS101 Software Development Platform is ready for loading and executing your application.



## 7.6.5 Running a Baremetal Application in the MetaWare IDE Debugger

Once the C Project has been successfully built, you can debug the executable on the AXS101 Software Development Platform. This section provides step-by-step instructions how to configure and run a debug session in the MetaWare IDE.

1. Perform the hardware setup as described in the “[Hardware Setup for Debugging](#)” section.
2. Open a hypertext terminal (such as PuTTY) on your PC and select the COM port that is connected to the `USB Dataport` of the ARC SDP Mainboard. Set the **Connection type** to `Serial` and the **Speed** to `115200`.
3. Launch the MetaWare IDE and open the workspace `/software/baremetal/project_arcmw`
4. Select **Debug Configurations** from the **Run** menu or by clicking on the down arrow next to the bug icon:

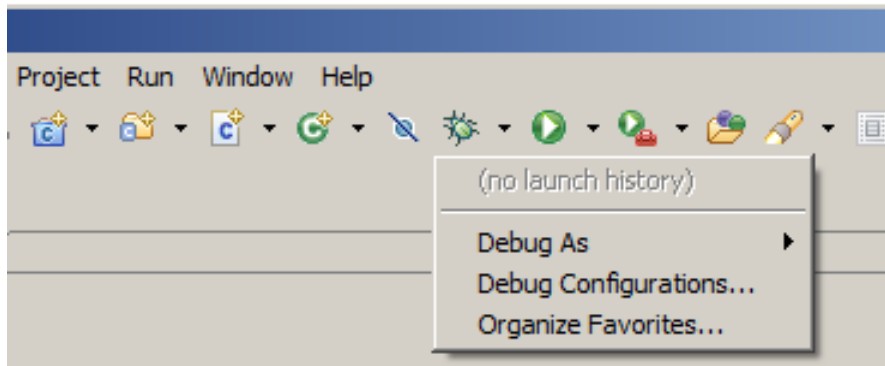


Figure 52 MetaWare IDE – Selecting the Debug Configuration (down arrow next to bug icon)

5. Select one of the pre-existing debug configurations and launch the debugger. If you want to create your own debug configuration, follow the optional steps below.
6. Double click on **C/C++ Application** to create a new debug configuration for the project or select an existing debug configuration. Select the **Main** tab and enter a name of your choice in the **Name** field. It is recommended to compose the name from the project name and the ARC core. Enter the path and the name of the elf-file (with or without HOSTLINK) in the **C/C++ Application** field and enter the application name in the **Project**. See [Figure 53](#) for an example.

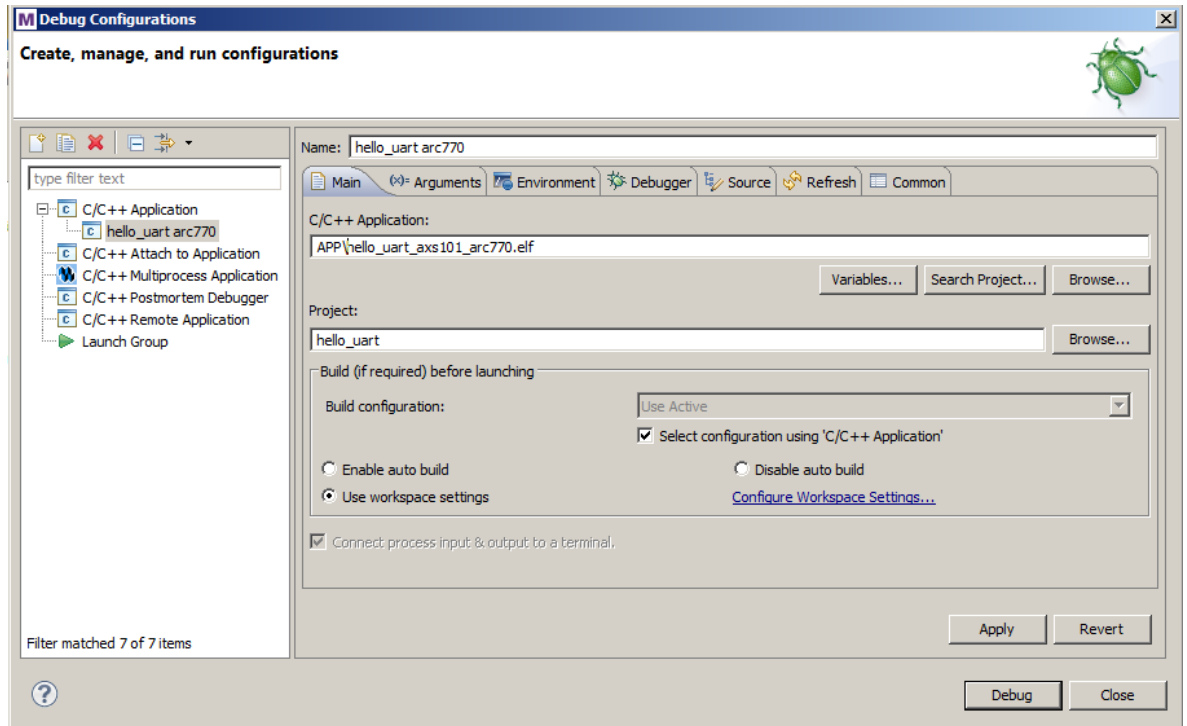


Figure 53 MetaWare IDE – Setting up the debug configuration

7. Click the **Debugger** tab and configure the target to use **Hardware** and connect the hardware via the “Digilent JTAG cable”.

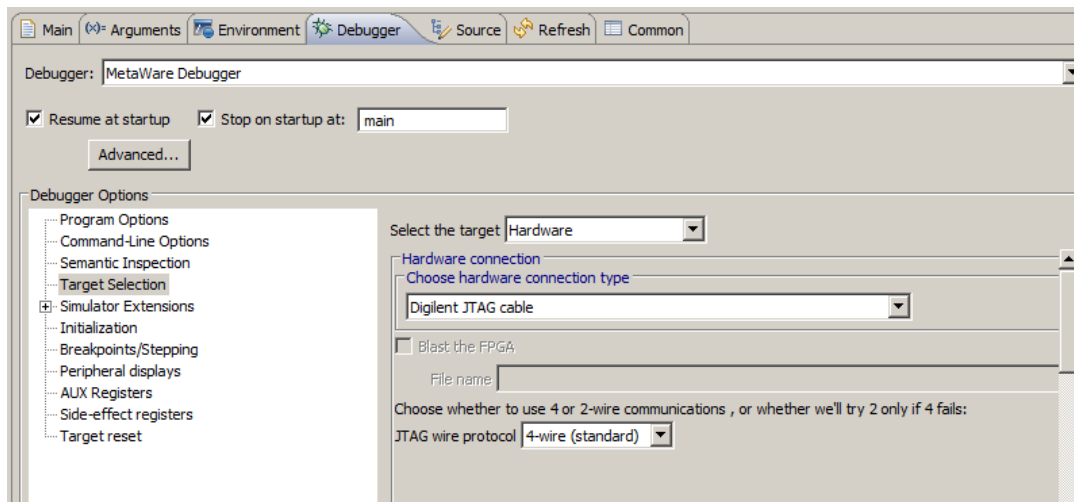


Figure 54 MetaWare IDE – Selecting the debugger target

8. In **Debugger Options** sub-tab **Command-line Options**, select the correct core and device:

-prop=cpunum=4                      implies ARC 770D (see [Table 35](#)).

-prop=dig\_device=AXS                selects the USB Dataport

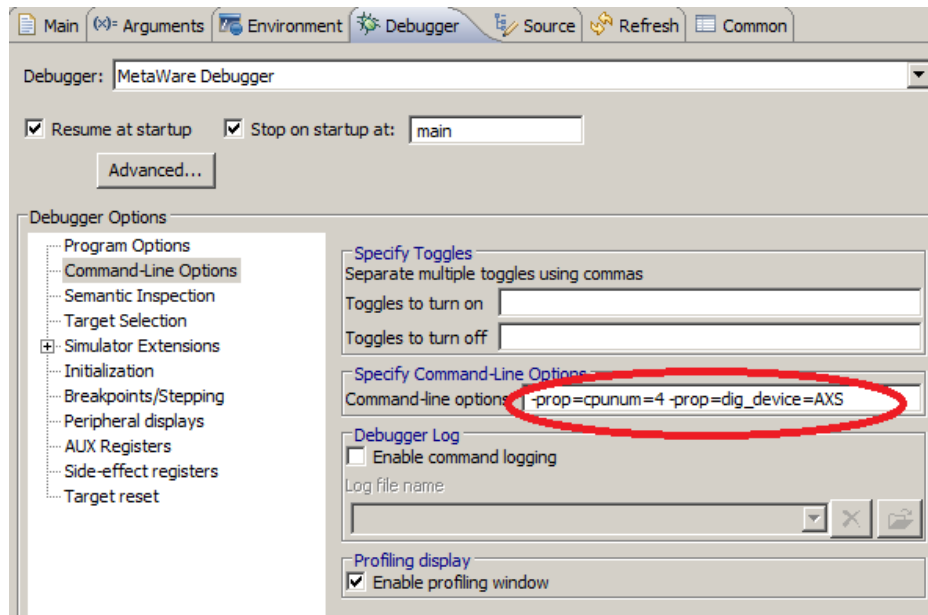


Figure 55 MetaWare IDE – Debugger command line options

Other possible `cpunum` options are:

Table 35 Selecting the CPU core in the debugger

Option	Description
<code>-prop=cpunum=4</code>	Select ARC 770D
<code>-prop=cpunum=3</code>	Select ARC EM6
<code>-prop=cpunum=2</code>	Select AS221 core 2
<code>-prop=cpunum=1</code>	Select AS221 core 1

If you are using a Digilent probe (rather than the USB Dataport) set one of the following options, depending on the type of your probe:

```
-prop=dig_device=JtagHs1
```

or

```
-prop=dig_device=JtagHs2
```

- Click the **Debug** button in the **Debug configurations** dialog to initiate a debug session and switch the IDE to the **Debug** perspective:

## 7.6.6 Running a Baremetal Application in the MetaWare Debugger

This section describes how to execute an image with the MetaWare debugger. In the example below we assume that the application “hello\_uart” is built for execution from the DDR SDRAM. The result of building is a file with the extension “.elf”. Use this file for the debugger.

1. Perform the hardware setup as described in the “[Hardware Setup for Debugging](#)” section.
2. Open a hypertext terminal (such as PuTTY) on your PC and connect to the COM port that is connected to the USB Dataport of the ARC SDP Mainboard. Set the **Connection type** to `Serial` and the **Speed** to 115200.
3. Start the MetaWare debugger.
4. Select **Create new process** (yellow ellipse) and open **Debugger options** (blue ellipse) as shown in [Figure 56](#):

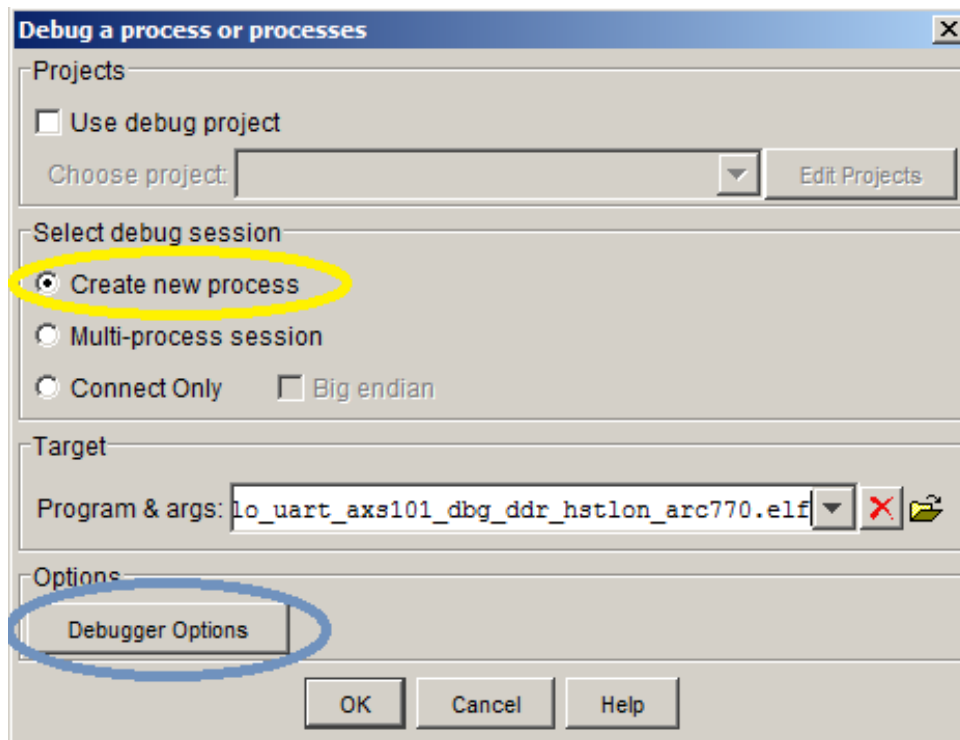


Figure 56 Creating a new process

5. In **Debugger Options** sub-tab **Command-line Options**, select the correct core and device:

-prop=cpunum=4                   implies ARC 770D (see [Table 36](#)).

-prop=dig\_device=AXS           selects the USB Dataport

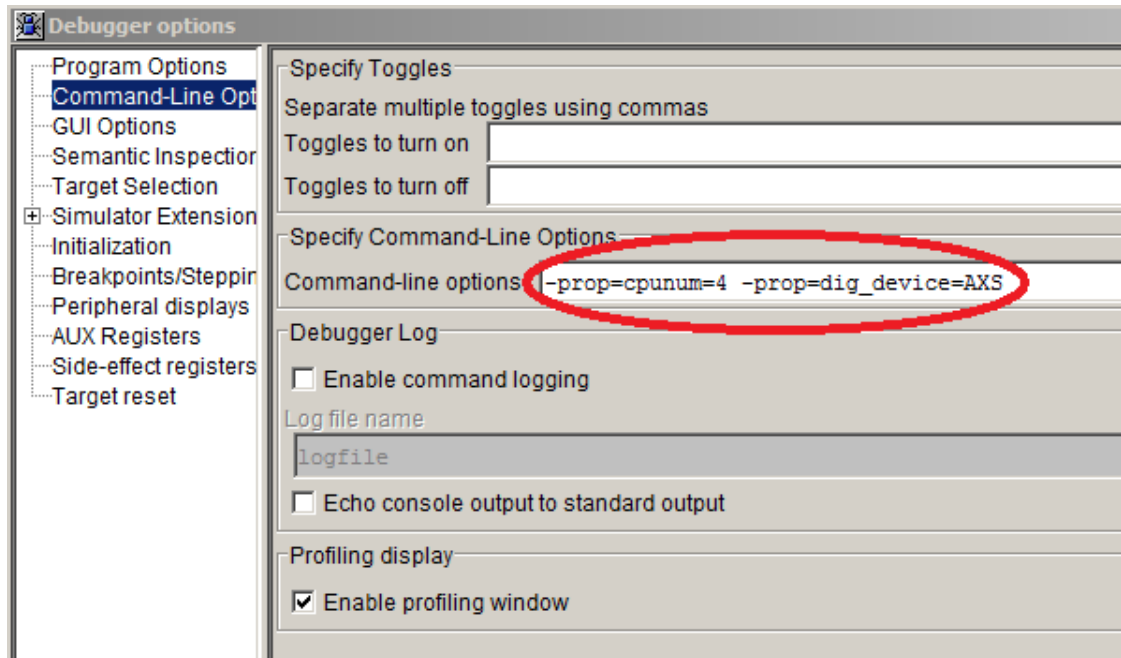


Figure 57 Debugger options – command-line options

Other possible `cpunum` options are:

Table 36 Selecting the CPU core in the debugger

Option	Description
<code>-prop=cpunum=4</code>	Select ARC 770D
<code>-prop=cpunum=3</code>	Select ARC EM6
<code>-prop=cpunum=2</code>	Select AS221 core 2
<code>-prop=cpunum=1</code>	Select AS221 core 1

If you are using a Digilent probe (rather than the USB Dataport) set one of the following options, depending on the type of your probe:

```
-prop=dig_device=JtagHs1
```

or

```
-prop=dig_device=JtagHs2
```

- In **Debugger Options** sub-tab **Target Selection**, select the correct settings for your debug probe. [Figure 58](#) explains this at the example of using the USB Dataport: Select the “Digilent JTAG cable” option and check the setting of the “JTAG wire protocol”:

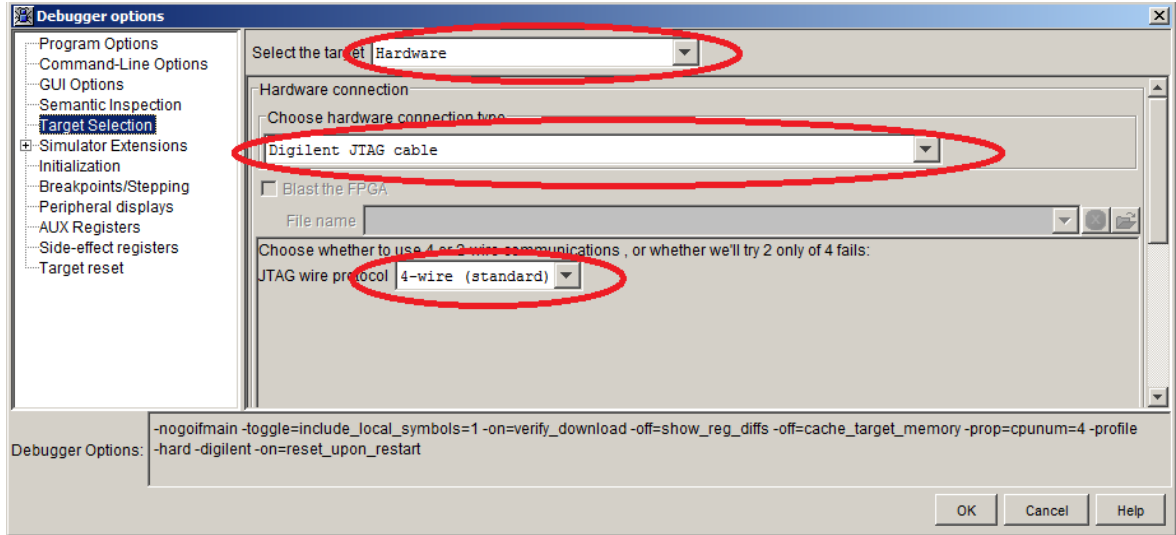


Figure 58 Debugger options – target selection

7. Back in the **Debug a process or processes** window, select the correct “.elf” file and press **OK**:

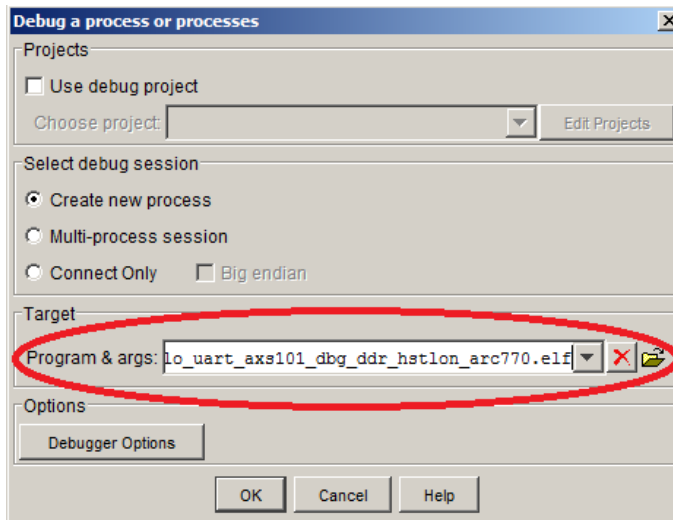


Figure 59 Specifying a path to the .elf file

- The debugger is now ready for executing the image. In the auxiliary register AUX0004, the ARC 770D ID will be visible (red ellipse, 0x434). Furthermore, the program counter points to the valid start instruction (blue ellipse) on address 0x8000\_0200.

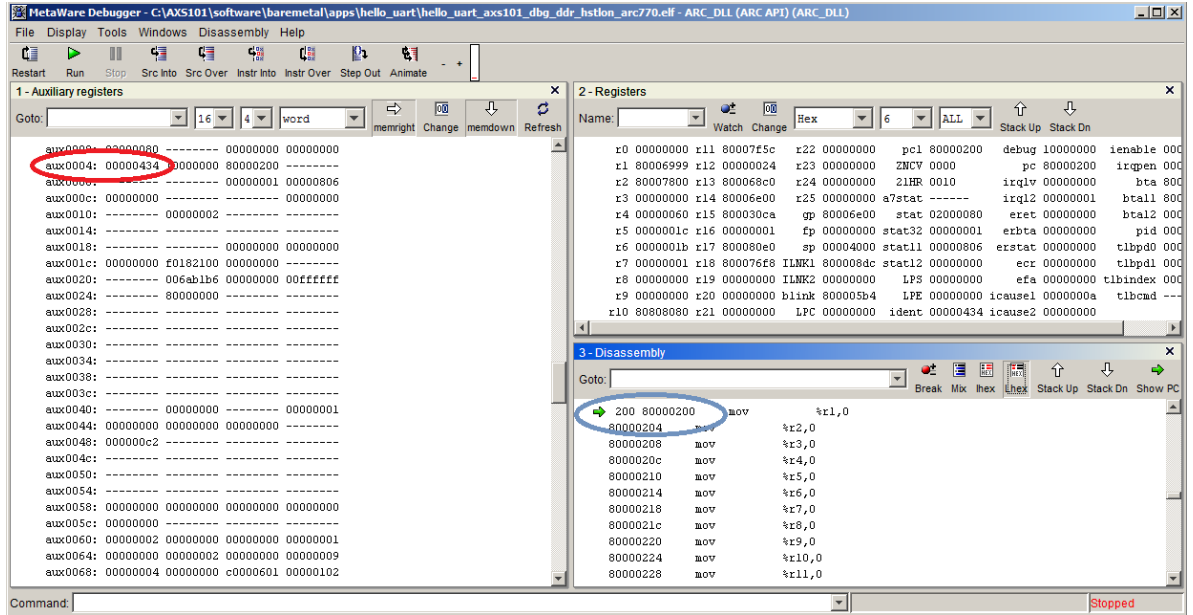


Figure 60 Debugger status

- Execute the program by pressing the “run” button in the debugger. Check the HyperTerminal input on the PC:

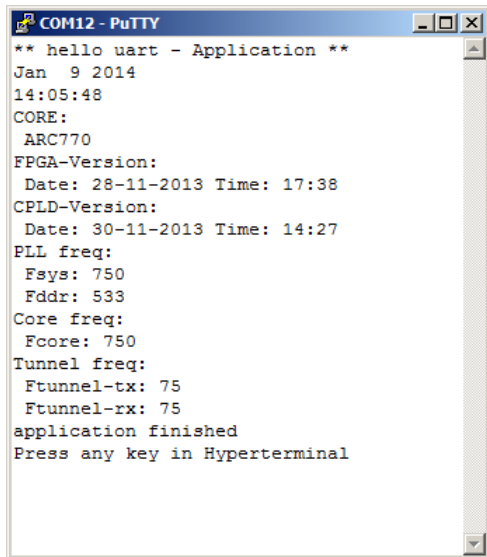


Figure 61 HyperTerminal output

## 7.6.7 Storing an Image in the SPI Flash and Running the Application

The steps below explain the process of storing an image for the SPI Flash and running the application. This is done using the application “hello\_uart” and the ARC 770D core as an example. If you wish to use another core, modify the instructions marked with **yellow** highlight .

Only images without HOSTLINK should be programmed in the SPI Flash.

After building the image use the following command to store the image in SPI-Flash:

```
axs_comm -c 0434 -p 80000000 -f ../../software/baremetal/
apps/hello_uart/hello_uart_axs101_arc770.bin
```

1. To program the hello\_uart example for the ARC 770D core navigate to the /software/tools/axs\_comm folder and use one of the following two commands, depending on the memory for which the image has been built  
Images built for the DDR2 SDRAM shall be programmed with the following command:

```
axs_comm -c 0434 -p 80000000 -f ../../software/baremetal/
apps/hello_uart/hello_uart_axs101_arc770.bin
```

The parameters used here have the following meaning:

**0434**

**ARC ID of ARC 770D**

Use 0126 for AS221 core #1

Use 0226 for AS221 core #2

Use 0341 for ARC EM6

80000000

**Program is built for DDR2 SDRAM @ 0x8000\_0000 so the Pre-Bootloader stores the image at this address**

../../software/baremetal/apps/hello\_uart/hello\_uart\_axs101\_  
**arc770**.bin

**Selected BIN file (result of build flow) including path.**



For images that have been built for the SRAM use this command instead:

```
axs_comm -c 0434 -p 20000000 -f ../../software/baremetal/
apps/hello_uart/hello_uart_axs101_ram_arc770.bin
```

The parameters used here have the following meaning:

0434

ARC ID of ARC 770D

Use 0126 for AS221 core #1

Use 0226 for AS221 core #2

Use 0341 for ARC EM6


20000000

Program is built for SRAM @  
0x2000\_0000 so the Pre-Bootloader  
stores the image at this address

```
../../software/baremetal/apps/hello_uart/hello_uart_axs101_
ram_arc770.bin
```

Selected BIN file (result of build flow  
including path).

Note that the `axs_comm` commands might erase the factory-programmed self-tests. If you wish to restore the self-tests later, refer to the [“Restoring the Self-Tests in the SPI Flash”](#) section.

2. In a typical case the application shall be loaded and executed autonomously after reset. Set the DIP switches on the ARC SDP Mainboard to the default settings, then set bit 2 of the respective DIP switch to the OFF position (2 ). [Figure 62](#) shows the DIP switch settings at the example of the ARC 770D. The change compared to the default settings is marked by a red ellipse. You can as well leave the DIP switches in the default position, if you prefer to start the execution by pushing the CPU Start button of the selected core.

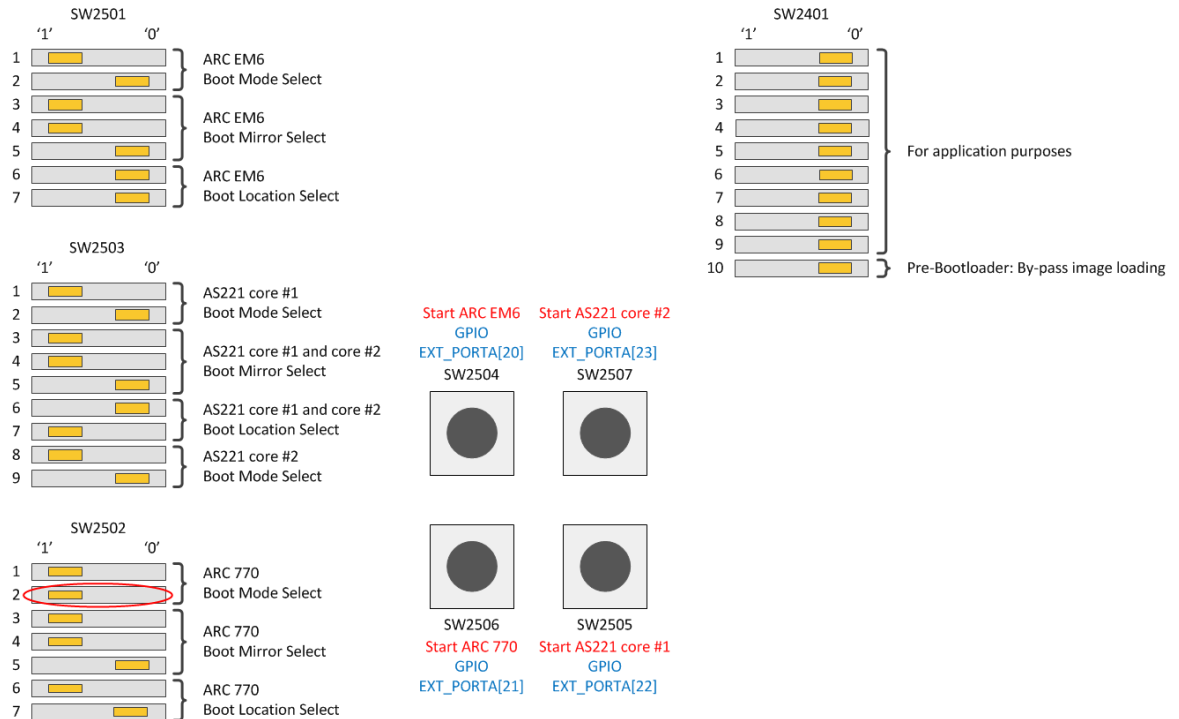


Figure 62 DIP switch settings for autonomous code execution on the ARC 770D

3. Push the `RESET` button on the ARC SDP Mainboard  
The Pre-Bootloader detects a valid image in sector 0 of the SPI flash, loads this image in DDR2 SDRAM or local SRAM and executes it.
4. If in step 2 you have chosen to leave the DIP switches in the default position, now push the `CPU Start` button of the selected core.

## 7.6.8 Storing an Image on the SD-Card and Running the Application

This section explains how to store an image on the SD-card and how to run the application.

Since loading the application requires U-Boot, this description is only applicable for the ARC770 D core.

It is assumed that the code has been built for the DDR memory with HOSTLINK off as described in the sections [“Building Baremetal Applications Using the MetaWare IDE”](#) and [“Building Baremetal Applications Using gmake”](#).

1. Copy the image `<app_name>_axs101_arc770.bin` to a FAT-formatted SD-Card
2. Start U-Boot as described in the [“Starting U-Boot on the ARC 770D Core”](#) section.
3. At the U-Boot prompt “AXS#” enter the following commands to start the application:

```
fatload mmc 0 80000000 <app_name>_axs101_arc770.bin
go 80000000
```

In the commands above the value `80000000` corresponds to the default start address as defined in the file `/software/baremetal/board/axs101/src/map_axs101_ddr.met`.

## 7.7 MQX Package

Before using the MQX package, make sure that you have installed the MetaWare toolchain (compiler / linker / debugger). This is a separate product, which is not part of the AXS101 Software Development Platform package.

### 7.7.1 Overview

The MQX package is part of the zip file `axs101_software_<version>.zip`, which can be obtained from the *ARC SDP download webpage [6]*. The package includes a pre-built binary version of the MQX operating system, peripheral drivers and example code that demonstrates how to build a simple MQX application and work with the peripheral device drivers.

Please unzip the software package into the directory `C:\AXS101`, so that the absolute path is `C:\AXS101\software`.

Table 37 MQX folder contents

Folder	Description
Root folder path: <code>/software/mqx&lt;mqx_version&gt;</code>	
<code>/examples</code>	This folder contains MQX example applications. Refer to the release notes on the <i>ARC SDP download webpage [6]</i> for an overview of the available application examples
<code>/build</code>	This folder contains configuration files for building applications
<code>/docs</code>	MQX documentation
<code>/library</code>	MQX libraries
<code>/mkscripts</code>	This folder contains <i>make</i> scripts for building applications

### 7.7.2 Building MQX Applications Using the MetaWare IDE

The software package includes a shortcut for opening a pre-configured MetaWare IDE workspace that contains example applications. Instructions below explain the process of opening the workspace and building example applications step-by-step:

1. Navigate to the `/software/mqx_<mqx_version>` folder and double-click on the `AXS101 Example Projects` shortcut. This shortcut assumes that the MetaWare Toolkit is installed at `C:\ARC` and the AXS101 SDP software package is installed at `C:\AXS101`.
2. IDE loads and displays the example projects in your workspace.
3. Select one or multiple projects and then perform a right-click.

4. Select “**Build Configurations > Set Active**” and select your build target.

For example, **arcv2em.met**.

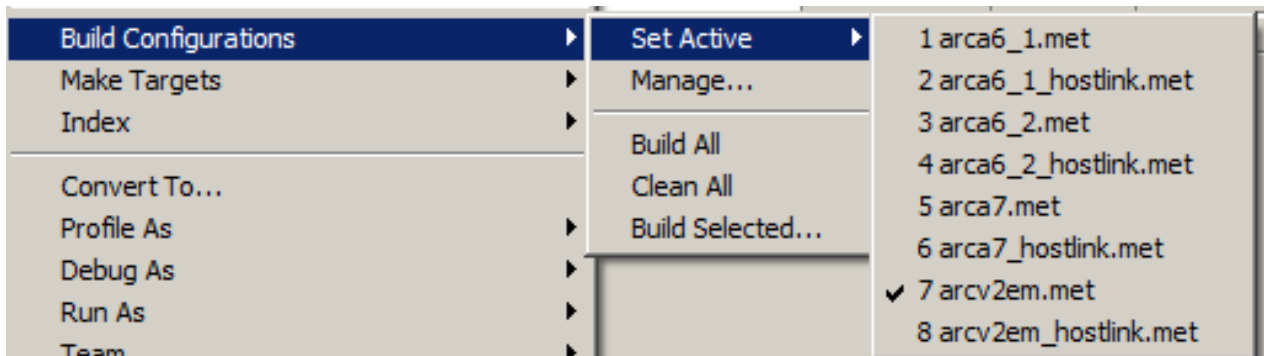


Figure 63 MetaWare IDE - Set Active Build Configurations for MQX

This step selects the *arcv2em* configuration of the project for building. This configuration uses the ARC EM6 core with a fast profile and HOSTLINK excluded. You can choose another option to match your debug target. Available options are described in Table 38.

Table 38 Description of the build configurations

Configuration Name	CPU Core	Description
arca6_1.met	AS221 core #1	MQX_PROFILE=fast HOSTLINK excluded
arca6_1_hostlink.met	AS221 core #1	MQX_PROFILE=debug HOSTLINK included
arca6_2.met	AS221 core #2	MQX_PROFILE=fast HOSTLINK excluded
arca6_2_hostlink.met	AS221 core #2	MQX_PROFILE=debug HOSTLINK included
arca7.met	ARC 770D	MQX_PROFILE=fast HOSTLINK excluded
arca7_hostlink.met	ARC 770D	MQX_PROFILE=debug HOSTLINK included
arcv2em.met	ARC EM6	MQX_PROFILE=fast HOSTLINK excluded
arcv2em_hostlink.met	ARC EM6	MQX_PROFILE=debug HOSTLINK included

The project is always built to execute code from the DDR2 SDRAM, using the start address `0x8000_0000`.

5. Build the project using any of the following methods:
  - Right-click the selected project(s) again and select **Build Project**
  - Select **Build Project** from the **Project** menu
  - Press **CTRL-B** to build all available projects

You can find the build results in the console window. Build process creates a \*.bin file and \*.elf file with the basename axsl01\_<project\_name>.

These images are located in the following folder:

```
/software/mqx_<mqx_version>/examples/axsl01/<project_name>/  
<configuration_name>
```

The base name for an image with HOSTLINK is:

```
axsl01_<project_name>_hostlink
```

For images without the HOSTLINK, the base name is axsl01\_<project\_name>.

6. For instructions on running the code in the MetaWare IDE Debugger, please refer to the sections "[Hardware Setup for Debugging](#)" and "[Running an MQX Application in the MetaWare IDE Debugger](#)".
7. For instructions on storing the code in the SPI Flash memory of the ARC SDP Mainboard or on an SD-Card, refer to the sections "[Storing an Image in the SPI Flash and Running the Application](#)" and "[Storing an Image on the SD-Card and Running the Application.](#)"

### 7.7.3 Hardware Setup for Debugging

Follow the steps below to execute an sample application in a debugger:

1. Set the jumpers to their default settings (see the “Jumpers” section). The JTAG interface is in daisy-chained mode.
2. Set the switches on the ARC SDP Mainboard as shown in Figure 64. The change compared to the default settings is marked by a red ellipse.

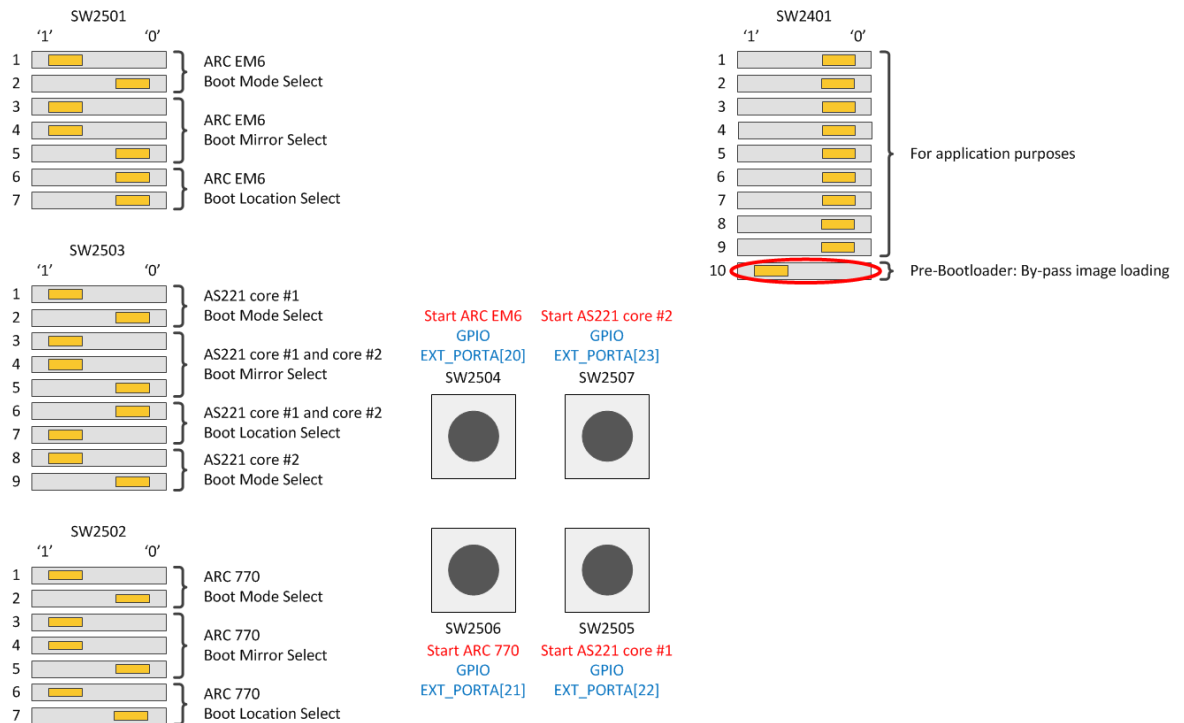


Figure 64 Settings of the DIP switches on the ARC SDP Mainboard for using the debugger

3. Connect the USB Dataport of the ARC SDP Mainboard to your PC.

- If you are using a debug probe rather than the JTAG channel of the USB Dataport, connect the probe to the appropriate connector and remove the jumper JP1402 on the ARC SDP Mainboard. If you are using a Lauterbach probe also remove the jumpers JP2309 and JP2310.

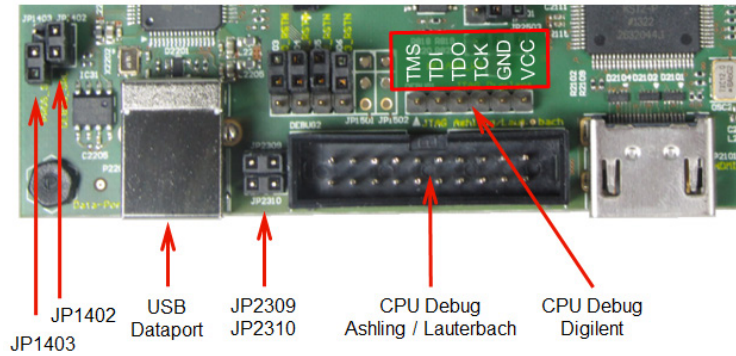


Figure 65 Location of the debug interfaces and the corresponding jumpers

- Switch ON the power supply or push the RESET button

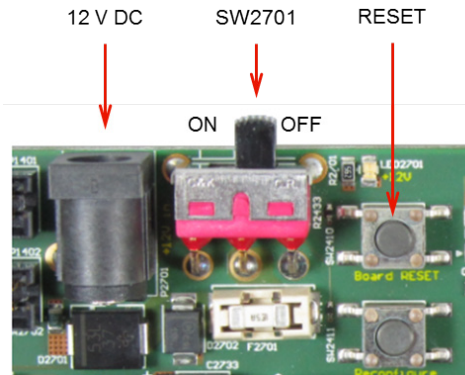


Figure 66 Location of the ARC SDP Mainboard's power supply and power switch

- Push the CPU Start button of your target CPU core according to Table 39. For the location of the CPU Start button see Figure 67. The Pre-Bootloader then automatically initializes the DDR2 SDRAM memory and sets the ARC core frequency correctly, but by-passes loading an image from the SPI Flash.





Figure 67 Location of the CPU Start buttons on the ARC SDP Mainboard.

Table 39 CPU Start buttons and seven-segment display values for running applications in the debugger

ARC Core	Start Button	Seven-Segment Display
ARC 770D	SW2506	4.0
AS221 core #1	SW2505	1.0
AS221 core #2	SW2507	2.0
ARC EM6	SW2504	3.0

When the seven-segment display shows the value listed in [Table 39](#) (please note the dot separator between the two digits) the AXS101 Software Development Platform is ready for loading and executing your application.

## 7.7.4 Running an MQX Application in the MetaWare IDE Debugger

Once the C Project has been successfully built, you can debug the executable on the AXS101 Software Development Platform. This section provides step-by-step instructions how to configure and run a debug session in the MetaWare IDE.

1. Perform the hardware setup as described in the “[Hardware Setup for Debugging](#)” section.
2. Navigate to the `/software/mqx_<mqx_version>` folder and double-click on the AXS101 Example Projects shortcut. This shortcut assumes that the MetaWare Toolkit has been installed at `C:\ARC` and the AXS101 SDP software package has been installed at `C:\AXS101`.
3. Select **Debug Configurations** from the **Run** menu or by clicking on the down arrow next to the bug icon:

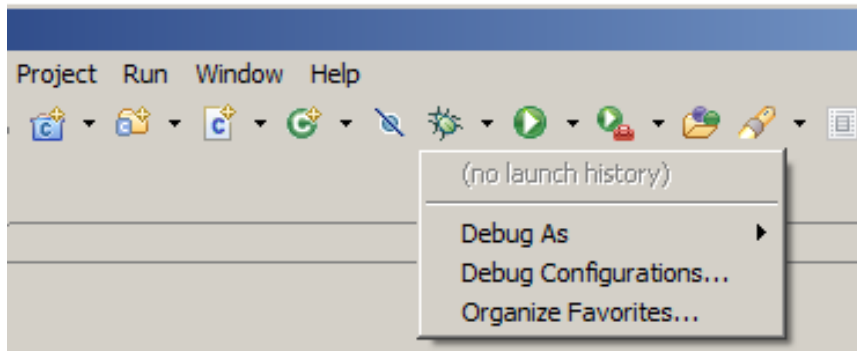


Figure 68 MetaWare IDE – Selecting the Debug Configuration (down arrow next to bug icon)

4. Double click on **C/C++ Application** to create a new debug configuration for the project or select an existing debug configuration. Select the **Main** tab and enter a name of your choice in the **Name** field. It is recommended to compose the name from the project name and the ARC core. Enter the name of the elf-file (with or without HOSTLINK) in the **C/C++ Application** field and enter the application name in the **Project**. See [Figure 69](#) for an example.

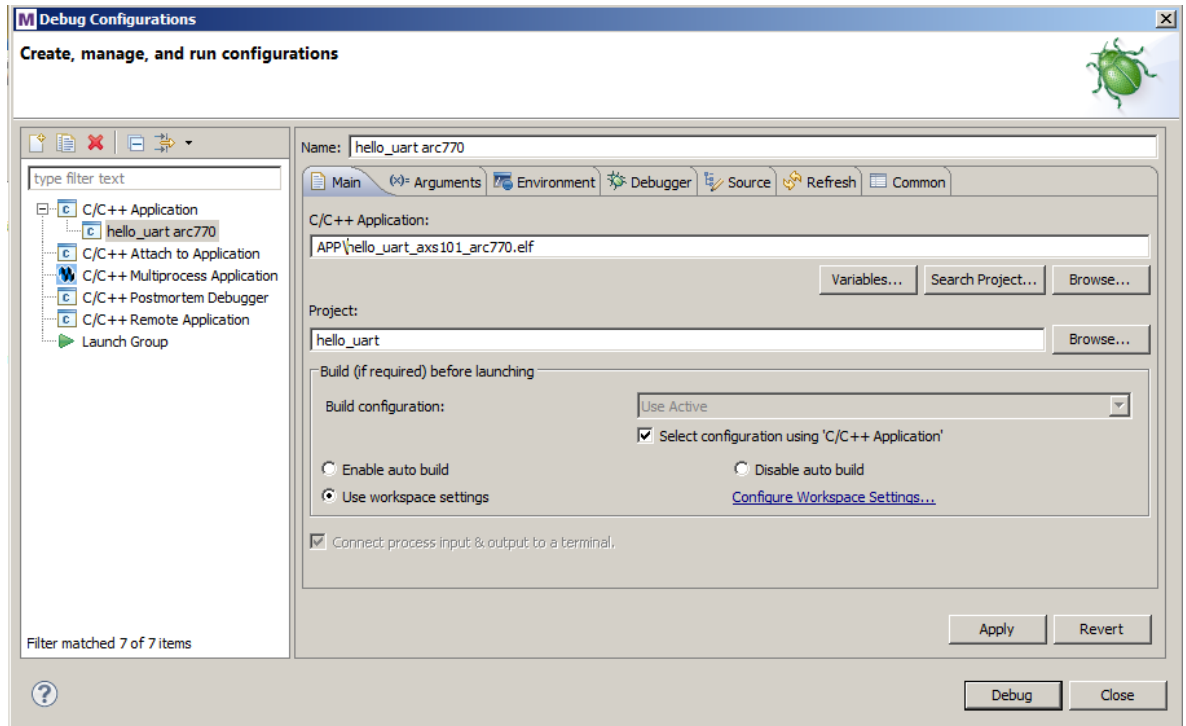


Figure 69 MetaWare IDE – Setting up the debug configuration

5. Click the **Debugger** tab and configure the target to use **Hardware** and connect the hardware via the “Digilent JTAG cable”.

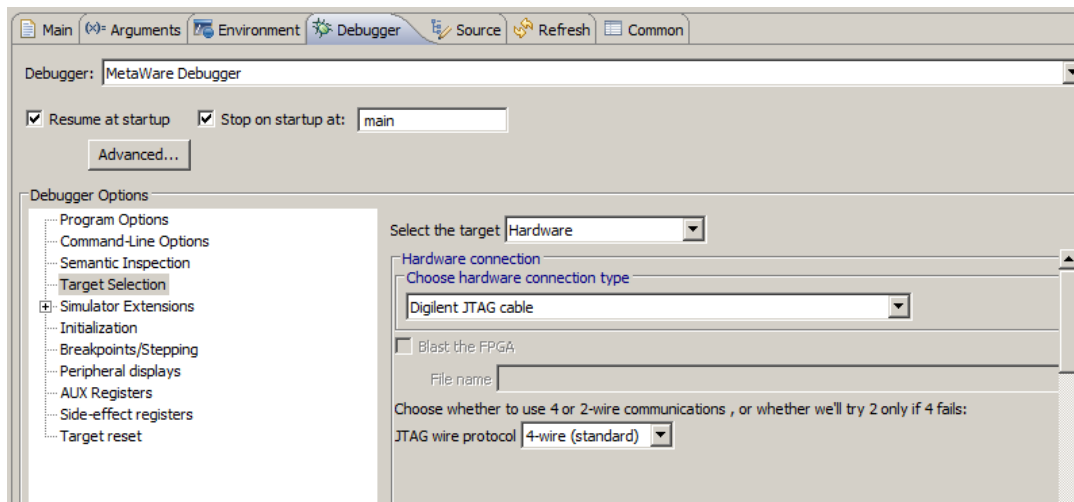


Figure 70 MetaWare IDE – Selecting the debugger target

10. In **Debugger Options** sub-tab **Command-line Options**, select the correct core and device:

-prop=cpunum=4                      implies ARC 770D (see [Table 40](#)).

-prop=dig\_device=AXS                selects the USB Dataport

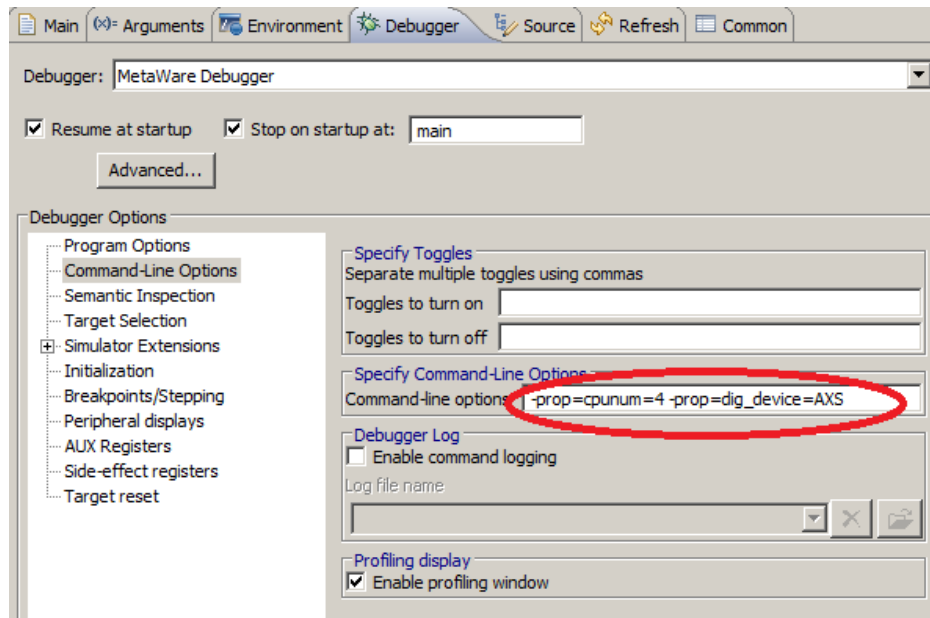


Figure 71 MetaWare IDE – Debugger command line options

Other possible `cpunum` options are:

Table 40 Selecting the CPU core in the debugger

Option	Description
<code>-prop=cpunum=4</code>	Select ARC 770D
<code>-prop=cpunum=3</code>	Select ARC EM6
<code>-prop=cpunum=2</code>	Select AS221 core 2
<code>-prop=cpunum=1</code>	Select AS221 core 1

If you are using a Digilent probe (rather than the USB Dataport) set one of the following options, depending on the type of your probe:

`-prop=dig_device=JtagHs1`

or

`-prop=dig_device=JtagHs2`

6. Click the **Debug** button in the **Debug configurations** dialog to initiate a debug session and switch the IDE to the **Debug** perspective:

---

## 7.8 Linux Package

### 7.8.1 Overview

The ARC 770D processor is well suited to run the GNU/Linux operating system. The Linux kernel and several other packages – including Buildroot for ARC that can be used to compile a Linux kernel and root filesystem from scratch – can be found on <https://github.com/foss-for-synopsys-dwc-arc-processors>.

A pre-built Linux kernel with a basic rootfilesystem is part of the software package `axs101_software_<version>.zip` in the form of a “ulmage”.

---

### 7.8.2 Loading and Executing the Linux Image

After unzipping the the software package `axs101_software_<version>.zip` you can find the `uImage` file in the `/software/axs101_linux_<version>/` folder.

This image can be loaded and started either with the MetaWare debugger, or with the U-Boot bootloader. The start address of the pre-built Linux kernel is `0x8000_0000`, which is the start address of the DDR SDRAM.

To boot the pre-built Linux image with U-Boot, execute the following steps:

1. Copy the file “uImage” to a FAT32 formatted SD-card using the SD-card slot of your PC.
2. Insert the SD-card in the card slot on the ARC SDP Mainboard
3. Boot the system with U-Boot as described in the “[Starting U-Boot on the ARC 770D Core](#)” section.
4. At the U-Boot prompt “AXS#”, enter the following commands:

```
fatload mmc 0
bootm
```

5. The Linux kernel starts. Wait for the login prompt and login as root (empty password)

---

### 7.8.3 Building Applications

For building Linux applications, the ARC GNU cross compiler and other tools that are created during the Buildroot compilation process described above can be used. These can be found in the `output/host/usr/bin` directory (`arc-buildroot-linux-uclibc-gcc`).

---

## 7.8.4 Linux Application Examples

Several standard Linux utilities (Busybox shell, i2c and usb utilities, gdbserver) are shipped as part of the pre-built Linux image that is part of the software package.

Using Buildroot, many other applications and utilities can be cross-compiled for ARC.

---

## 7.8.5 Updating the Linux Image

ARC Open Source solutions are maintained on GitHub and the latest releases for the AXS101 Software Development Platform can be downloaded from <https://github.com/foss-for-synopsys-dwc-arc-processors>. The pre-built Linux image can be updated, recreated and/or modified using the Buildroot package from github, following the standard Buildroot conventions and way-of-working.

The following steps should be performed (on a Linux development host) to create a Linux image for the AXS101 Software Development Platform. Refer to the release notes on the ARC SDP download webpage [6] for the release number and branch name used for the pre-built image that is delivered as part of the AXS101 SDP release.

1. Obtain the Buildroot source tree for ARC from github using the following alternative methods:

- Select the desired release of the AXS101 Software Development Platform at the following URL:

<https://github.com/foss-for-synopsys-dwc-arc-processors/buildroot/releases>

This web page provides download links for the source code and for a binary executable (uImage), which can be loaded with U-Boot.

- Use the git utility. The example commands below use the branch name `arc-axs101-20140128`:

```
git clone https://github.com/foss-for-synopsys-dwc-arc-processors/buildroot
cd buildroot
git checkout -b my-axs101 arc-axs-101-20140217
```

2. In the buildroot source directory, configure and compile the Linux image:

```
make axs101_defconfig
```

```
make
```

3. The result will be a new “uImage” file that can be found in the output/images directory that is created as part of the Buildroot build process. It can be put onto an SD-card and loaded using U-Boot as described in the “[Loading and Executing the Linux Image](#)” section.

This section describes the control registers within the CREG module. The address offset listed for each register needs to be added to the base address of the AXI2APB bridge (default address: 0xF000\_0000) to obtain the register address.

## 8.1 ARC EM6 Address Decoder Registers

The ARC EM6 Address Decoder Registers described below are re-programmed by the Pre-Bootloader. The reset values mentioned here are the reset values prior to running the Pre-Bootloader. See the “[Example Register Settings for the Default Memory Map](#)” section for the register settings after running the Pre-Bootloader.

### 8.1.1 CREG\_EM6\_A\_SLV0: ARC EM6 Slave Select Register 0

Address offset: 0x1000

Reset value: 0x3333\_3324

Table 41 CREG\_EM6\_A\_SLV0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL0	RW		Slave select for address aperture[0]
			0	no slave selected
			1	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4 *	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL1	RW	2 *	Slave select for address aperture[1] <sup>1)</sup>
11:8	SLV_SEL2	RW	3 *	Slave select for address aperture[2] <sup>1)</sup>
15:12	SLV_SEL3	RW	3 *	Slave select for address aperture[3] <sup>1)</sup>
19:16	SLV_SEL4	RW	3 *	Slave select for address aperture[4] <sup>1)</sup>
23:20	SLV_SEL5	RW	3 *	Slave select for address aperture[5] <sup>1)</sup>
27:24	SLV_SEL6	RW	3 *	Slave select for address aperture[6] <sup>1)</sup>
31:28	SLV_SEL7	RW	3 *	Slave select for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_SEL0

## 8.1.2 CREG\_EM6\_A\_SLV1: ARC EM6 Slave Select Register 1

Address offset: 0x1004

Reset value: 0x6333\_1115

Table 42 CREG\_EM6\_A\_SLV1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL8	RW		Slave select for address aperture[8]
			0	no slave selected
			1	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4	slave 4 selected (=> EM6 ICCM)
			5 *	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL9	RW	1 *	Slave select for address aperture[9] <sup>1)</sup>
11:8	SLV_SEL10	RW	1 *	Slave select for address aperture[10] <sup>1)</sup>
15:12	SLV_SEL11	RW	1 *	Slave select for address aperture[11] <sup>1)</sup>
19:16	SLV_SEL12	RW	3 *	Slave select for address aperture[12] <sup>1)</sup>
23:20	SLV_SEL13	RW	3 *	Slave select for address aperture[13] <sup>1)</sup>
27:24	SLV_SEL14	RW	3 *	Slave select for address aperture[14] <sup>1)</sup>
31:28	SLV_SEL15	RW	6 *	Slave select for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_SEL8



### 8.1.3 CREG\_EM6\_A\_OFFSET0: ARC EM6 Address Offset Register 0

Address offset: 0x1008

Reset value: 0x0000\_0000

Table 43 CREG\_EM6\_A\_OFFSET0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET0	RW		Address offset select for address aperture[0]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET1	RW	0*	Address offset for address aperture[1] <sup>1)</sup>
11:8	SLV_OFFSET2	RW	0*	Address offset for address aperture[2] <sup>1)</sup>
15:12	SLV_OFFSET3	RW	0*	Address offset for address aperture[3] <sup>1)</sup>
19:16	SLV_OFFSET4	RW	0*	Address offset for address aperture[4] <sup>1)</sup>
23:20	SLV_OFFSET5	RW	0*	Address offset for address aperture[5] <sup>1)</sup>
27:24	SLV_OFFSET6	RW	0*	Address offset for address aperture[6] <sup>1)</sup>
31:28	SLV_OFFSET7	RW	0*	Address offset for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET0

## 8.1.4 CREG\_EM6\_A\_OFFSET1: ARC EM6 Address Offset Register 1

Address offset: 0x100C

Reset value: 0x0000\_0000

Table 44 CREG\_EM6\_A\_OFFSET1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET8	RW		Address offset for address aperture[8]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET9	RW	0*	Address offset for address aperture[9] <sup>1)</sup>
11:8	SLV_OFFSET10	RW	0*	Address offset for address aperture[10] <sup>1)</sup>
15:12	SLV_OFFSET11	RW	0*	Address offset for address aperture[11] <sup>1)</sup>
19:16	SLV_OFFSET12	RW	0*	Address offset for address aperture[12] <sup>1)</sup>
23:20	SLV_OFFSET13	RW	0*	Address offset for address aperture[13] <sup>1)</sup>
27:24	SLV_OFFSET14	RW	0*	Address offset for address aperture[14] <sup>1)</sup>
31:28	SLV_OFFSET15	RW	0*	Address offset for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET8

## 8.1.5 CREG\_EM6\_A\_BOOT: ARC EM6 Boot Mirror Register

Address offset: 0x1010

Reset value: Depends on DIP switch settings on the ARC SDP Mainboard

Table 45 CREG\_EM6\_A\_BOOT register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	MIRROR_SEL	RW		Boot mirror select for ARC EM6
			0	address aperture[0] is boot mirror
			1 *	address aperture[1] is boot mirror
			2	address aperture[2] is boot mirror
			...	
			15	address aperture[15] is boot mirror
7:4	Reserved	R	0x0	
10:8	BS_SEL	RW		Boot source select for ARC EM6 <sup>1)</sup>
			0	boot mirror disabled
			1	boot source is slave 1 (=> DDR controller port 0)
			2	boot source is slave 2 (=> SRAM controller)
			3	boot source is slave 3 (=> AXI tunnel)
			4	boot source is slave 4 (=>EM6 ICCM)
			5-7	not a valid boot source
12:11	BL_SEL	RW		Boot location select for ARC EM6 <sup>1)</sup>
			0	0 – 2KB
			1	2 – 4KB
			2	4 – 6KB
			3	6 – 8KB
31:13	Reserved	R	0x0 *	

1) Reset values for BS\_SEL and BL\_SEL depend on DIP switch settings on the ARC SDP Mainboard

## 8.1.6 CREG\_EM6\_A\_UPDATE: ARC EM6 Update Register

Address offset: 0x1014

Reset value: 0x0000\_0000

Table 46 CREG\_EM6\_A\_UPDATE register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	UPDATE	RW1C	0	All the address aperture configuration registers (i.e. *_A_SLV and *_A_BOOT) are double-buffered. The newly programmed values will be only be forwarded to the address decoder after writing a '1' to this bit.
31:1	Reserved	R	0x0 *	

## 8.2 ARC 770D Address Decoder Registers

The ARC 770D Address Decoder Registers described below are re-programmed by the Pre-Bootloader. The reset values mentioned here are the reset values prior to running the Pre-Bootloader. See the “[Example Register Settings for the Default Memory Map](#)” section for the register settings after running the Pre-bootloader.

### 8.2.1 CREG\_770\_A\_SLV0: ARC 770D Slave Select Register 0

Address offset: 0x1020

Reset value: 0x3333\_3324

Table 47 CREG\_770\_A\_SLV0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL0	RW		Slave select for address aperture[0]
			0	no slave selected
			1	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4 *	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL1	RW	2 *	Slave select for address aperture[1] <sup>1)</sup>
11:8	SLV_SEL2	RW	3 *	Slave select for address aperture[2] <sup>1)</sup>
15:12	SLV_SEL3	RW	3 *	Slave select for address aperture[3] <sup>1)</sup>
19:16	SLV_SEL4	RW	3 *	Slave select for address aperture[4] <sup>1)</sup>
23:20	SLV_SEL5	RW	3 *	Slave select for address aperture[5] <sup>1)</sup>
27:24	SLV_SEL6	RW	3 *	Slave select for address aperture[6] <sup>1)</sup>
31:28	SLV_SEL7	RW	3 *	Slave select for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_SEL0

## 8.2.2 CREG\_770\_A\_SLV1: ARC 770D Slave Select Register 1

Address offset: 0x1024

Reset value: 0x6333\_1111

Table 48 CREG\_770\_A\_SLV1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL8	RW		Slave select for address aperture[8]
			0	no slave selected
			1 *	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
			7	slave 7 selected (=> DDR controller port 1)
7:4	SLV_SEL9	RW	1 *	Slave select for address aperture[9] <sup>1)</sup>
11:8	SLV_SEL10	RW	1 *	Slave select for address aperture[10] <sup>1)</sup>
15:12	SLV_SEL11	RW	1 *	Slave select for address aperture[11] <sup>1)</sup>
19:16	SLV_SEL12	RW	3 *	Slave select for address aperture[12] <sup>1)</sup>
23:20	SLV_SEL13	RW	3 *	Slave select for address aperture[13] <sup>1)</sup>
27:24	SLV_SEL14	RW	3 *	Slave select for address aperture[14] <sup>1)</sup>
31:28	SLV_SEL15	RW	6 *	Slave select for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_SEL8

### 8.2.3 CREG\_770\_A\_OFFSET0: ARC 770D Address Offset Register 0

Address offset: 0x1028

Reset value: 0x0000\_0000

Table 49 CREG\_770\_A\_OFFSET0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET0	RW		Address offset select for address aperture[0]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET1	RW	0*	Address offset for address aperture[1] <sup>1)</sup>
11:8	SLV_OFFSET2	RW	0*	Address offset for address aperture[2] <sup>1)</sup>
15:12	SLV_OFFSET3	RW	0*	Address offset for address aperture[3] <sup>1)</sup>
19:16	SLV_OFFSET4	RW	0*	Address offset for address aperture[4] <sup>1)</sup>
23:20	SLV_OFFSET5	RW	0*	Address offset for address aperture[5] <sup>1)</sup>
27:24	SLV_OFFSET6	RW	0*	Address offset for address aperture[6] <sup>1)</sup>
31:28	SLV_OFFSET7	RW	0*	Address offset for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET0

## 8.2.4 CREG\_770\_A\_OFFSET1: ARC 770D Address Offset Register 1

Address offset: 0x102C

Reset value: 0x0000\_0000

Table 50 CREG\_770\_A\_OFFSET1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET8	RW		Address offset for address aperture[8]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET9	RW	0*	Address offset for address aperture[9] <sup>1)</sup>
11:8	SLV_OFFSET10	RW	0*	Address offset for address aperture[10] <sup>1)</sup>
15:12	SLV_OFFSET11	RW	0*	Address offset for address aperture[11] <sup>1)</sup>
19:16	SLV_OFFSET12	RW	0*	Address offset for address aperture[12] <sup>1)</sup>
23:20	SLV_OFFSET13	RW	0*	Address offset for address aperture[13] <sup>1)</sup>
27:24	SLV_OFFSET14	RW	0*	Address offset for address aperture[14] <sup>1)</sup>
31:28	SLV_OFFSET15	RW	0*	Address offset for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET8



## 8.2.5 CREG\_770\_A\_BOOT: ARC 770D Boot Mirror Register

Address offset: 0x1030

Reset value: Depends on DIP switch settings on the ARC SDP Mainboard

Table 51 CREG\_770\_A\_BOOT register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	MIRROR_SEL	RW		Boot mirror select for ARC 770D
			0 *	address aperture[0] is boot mirror
			1	address aperture[1] is boot mirror
			2	address aperture[2] is boot mirror
			...	
			15	address aperture[15] is boot mirror
7:4	Reserved	R	0x0 *	
10:8	BS_SEL	RW		Boot source select for ARC 770D <sup>1)</sup>
			0 <sup>1</sup>	boot mirror disabled
			1	boot source is slave 1 (=> DDR controller port 0)
			2	boot source is slave 2 (=> SRAM controller)
			3	boot source is slave 3 (=> AXI tunnel)
			4	boot source is slave 4 (=>EM6 ICCM)
			5-7	not a valid boot source
12:11	BL_SEL	RW		Boot location select for ARC 770D <sup>1)</sup>
			0	0 - 2KB
			1	2 - 4KB
			2	4 - 6KB
			3	6 - 8KB
31:13	Reserved	R	0x0 *	

1) Reset values for BS\_SEL and BL\_SEL depend on DIP switch settings on the ARC SDP Mainboard

## 8.2.6 CREG\_770\_A\_UPDATE: ARC 770D Update Register

Address offset: 0x1034

Reset value: 0x0000\_0000

Table 52 CREG\_770\_A\_UPDATE register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	UPDATE	RW1C	0	All the address aperture configuration registers (i.e. *_A_SLV and *_A_BOOT) are double-buffered. The newly programmed values will be only be forwarded to the address decoder after writing a '1' to this bit.
31:1	Reserved	R	0x0 *	

## 8.3 ARC AS221 Address Decoder Registers

The ARC AS221 Address Decoder Registers described below are re-programmed by the Pre-Bootloader. The reset values mentioned here are the reset values prior to running the Pre-bootloader. See the “[Example Register Settings for the Default Memory Map](#)” section for the register settings after running the Pre-bootloader.

### 8.3.1 CREG\_221\_A\_SLV0: ARC 221 Slave Select Register 0

Address offset: 0x1040

Reset value: 0x3333\_3324

Table 53 CREG\_221\_A\_SLV0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL0	RW		Slave select for address aperture[0]
			0	no slave selected
			1	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4 *	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL1	RW	2 *	Slave select for address aperture[1] <sup>1)</sup>
11:8	SLV_SEL2	RW	3 *	Slave select for address aperture[2] <sup>1)</sup>
15:12	SLV_SEL3	RW	3 *	Slave select for address aperture[3] <sup>1)</sup>
19:16	SLV_SEL4	RW	3 *	Slave select for address aperture[4] <sup>1)</sup>
23:20	SLV_SEL5	RW	3 *	Slave select for address aperture[5] <sup>1)</sup>
27:24	SLV_SEL6	RW	3 *	Slave select for address aperture[6] <sup>1)</sup>
31:28	SLV_SEL7	RW	3 *	Slave select for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_SEL0

### 8.3.2 CREG\_221\_A\_SLV1: ARC 221 Slave Select Register 1

Address offset: 0x1044

Reset value: 0x6333\_1111

Table 54 CREG\_221\_A\_SLV1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL8	RW		Slave select for address aperture[8]
			0	no slave selected
			1 *	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL9	RW	1 *	Slave select for address aperture[9] <sup>1)</sup>
11:8	SLV_SEL10	RW	1 *	Slave select for address aperture[10] <sup>1)</sup>
15:12	SLV_SEL11	RW	1 *	Slave select for address aperture[11] <sup>1)</sup>
19:16	SLV_SEL12	RW	3 *	Slave select for address aperture[12] <sup>1)</sup>
23:20	SLV_SEL13	RW	3 *	Slave select for address aperture[13] <sup>1)</sup>
27:24	SLV_SEL14	RW	3 *	Slave select for address aperture[14] <sup>1)</sup>
31:28	SLV_SEL15	RW	6 *	Slave select for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_SEL8

### 8.3.3 CREG\_221\_A\_OFFSET0: ARC 221 Address Offset Register 0

Address offset: 0x1048

Reset value: 0x6333\_1111

Table 55 CREG\_221\_A\_OFFSET0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET0	RW		Address offset select for address aperture[0]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET1	RW	0*	Address offset for address aperture[1] <sup>1)</sup>
11:8	SLV_OFFSET2	RW	0*	Address offset for address aperture[2] <sup>1)</sup>
15:12	SLV_OFFSET3	RW	0*	Address offset for address aperture[3] <sup>1)</sup>
19:16	SLV_OFFSET4	RW	0*	Address offset for address aperture[4] <sup>1)</sup>
23:20	SLV_OFFSET5	RW	0*	Address offset for address aperture[5] <sup>1)</sup>
27:24	SLV_OFFSET6	RW	0*	Address offset for address aperture[6] <sup>1)</sup>
31:28	SLV_OFFSET7	RW	0*	Address offset for address aperture[7] <sup>1)</sup>

1) Same encoding as for *SLV\_OFFSET0*

### 8.3.4 CREG\_221\_A\_OFFSET1: ARC 221 Address Offset Register 1

Address offset: 0x104C

Reset value: 0x0000\_0000

Table 56 CREG\_221\_A\_OFFSET1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET8	RW		Address offset for address aperture[8]
			0*	0*256MB
			1	1*256MB
			...	....
			15	15*256MB
7:4	SLV_OFFSET9	RW	0*	Address offset for address aperture[9] <sup>1)</sup>
11:8	SLV_OFFSET10	RW	0*	Address offset for address aperture[10] <sup>1)</sup>
15:12	SLV_OFFSET11	RW	0*	Address offset for address aperture[11] <sup>1)</sup>
19:16	SLV_OFFSET12	RW	0*	Address offset for address aperture[12] <sup>1)</sup>
23:20	SLV_OFFSET13	RW	0*	Address offset for address aperture[13] <sup>1)</sup>
27:24	SLV_OFFSET14	RW	0*	Address offset for address aperture[14] <sup>1)</sup>
31:28	SLV_OFFSET15	RW	0*	Address offset for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET8

### 8.3.5 CREG\_221\_A\_BOOT: ARC 221 Boot Mirror Register

Address offset: 0x1050

Reset value: Depends on DIP switch settings on the ARC SDP Mainboard

Table 57 CREG\_221\_A\_BOOT register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	MIRROR_SEL	RW		Boot mirror select for ARC AS221
			0 *	address aperture[0] is boot mirror
			1	address aperture[1] is boot mirror
			2	address aperture[2] is boot mirror
			...	
			15	address aperture[15] is boot mirror
7:4	Reserved	R	0x0 *	
10:8	BS_SEL	RW		Boot source select for ARC AS221 <sup>1)</sup>
			0	boot mirror disabled
			1	boot source is slave 1 (=> DDR controller port 0)
			2	boot source is slave 2 (=> SRAM controller)
			3	boot source is slave 3 (=> AXI tunnel)
			4	boot source is slave 4 (=>EM6 ICCM)
			5-7	not a valid boot source
12:11	BL_SEL	RW		Boot location select for ARC AS221 <sup>1)</sup>
			0	0 - 2KB
			1	2 - 4KB
			2	4 - 6KB
			3	6 - 8KB
31:13	Reserved	R	0x0 *	

1) Reset values of *BS\_SEL* and *BL\_SEL* depend on DIP switch settings on the ARC SDP Mainboard

### 8.3.6 CREG\_221\_A\_UPDATE: ARC 221 Update Register

Address offset: 0x1054

Reset value: 0x0000\_0000

Table 58 CREG\_221\_A\_UPDATE register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	UPDATE	RW1C	0	All the address aperture configuration registers (i.e. *_A_SLV and *_A_BOOT) are double-buffered. The newly programmed values will only be forwarded to the address decoder after writing a '1' to this bit.
31:1	Reserved	R	0x0 *	



## 8.4 AXI Tunnel Address Decoder Registers

The AXI Tunnel Address Decoder Registers described below are re-programmed by the Pre-Bootloader. The reset values mentioned here are the reset values prior to running the Pre-bootloader. See the “[Example Register Settings for the Default Memory Map](#)” section for register settings, after running the Pre-bootloader.

### 8.4.1 CREG\_TUN\_A\_SLV0: AXI Tunnel Slave Select Register 0

Address offset: 0x1060

Reset value: 0x0000\_0024

Table 59 CREG\_TUN\_A\_SLV0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL0	RW		Slave select for address aperture[0]
			0	no slave selected
			1	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4 *	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
7	slave 7 selected (=> DDR controller port 1)			
7:4	SLV_SEL1	RW	2 *	Slave select for address aperture[1] <sup>1)</sup>
11:8	SLV_SEL2	RW	0 *	Slave select for address aperture[2] <sup>1)</sup>
15:12	SLV_SEL3	RW	0 *	Slave select for address aperture[3] <sup>1)</sup>
19:16	SLV_SEL4	RW	0 *	Slave select for address aperture[4] <sup>1)</sup>
23:20	SLV_SEL5	RW	0 *	Slave select for address aperture[5] <sup>1)</sup>
27:24	SLV_SEL6	RW	0 *	Slave select for address aperture[6] <sup>1)</sup>
31:28	SLV_SEL7	RW	0 *	Slave select for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_SEL0

## 8.4.2 CREG\_TUN\_A\_SLV1: AXI Tunnel Slave Select Register 1

Address offset: 0x1064

Reset value: 0x6000\_1111

Table 60 CREG\_TUN\_A\_SLV1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_SEL8	RW		Slave select for address aperture[8]
			0	no slave selected
			1 *	slave 1 selected (=> DDR controller port 0)
			2	slave 2 selected (=> SRAM controller)
			3	slave 3 selected (=> AXI tunnel)
			4	slave 4 selected (=> EM6 ICCM)
			5	slave 5 selected (=> EM6 DCCM)
			6	slave 6 selected (=> AXI2APB bridge)
			7	slave 7 selected (=> DDR controller port 1)
7:4	SLV_SEL9	RW	1 *	Slave select for address aperture[9] <sup>1)</sup>
11:8	SLV_SEL10	RW	1 *	Slave select for address aperture[10] <sup>1)</sup>
15:12	SLV_SEL11	RW	1 *	Slave select for address aperture[11] <sup>1)</sup>
19:16	SLV_SEL12	RW	0 *	Slave select for address aperture[12] <sup>1)</sup>
23:20	SLV_SEL13	RW	0 *	Slave select for address aperture[13] <sup>1)</sup>
27:24	SLV_SEL14	RW	0 *	Slave select for address aperture[14] <sup>1)</sup>
31:28	SLV_SEL15	RW	6 *	Slave select for address aperture[15] <sup>1)</sup>

1) Same encoding as for SLV\_SEL8

### 8.4.3 CREG\_TUN\_A\_OFFSET0: AXI Tunnel Address Offset Register 0

Address offset: 0x1068

Reset value: 0x0000\_0000

Table 61 CREG\_TUN\_A\_OFFSET0 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET0	RW		Address offset select for address aperture[0]
			0*	0*256MB
			1	1*256MB
			...	...
			15	15*256MB
7:4	SLV_OFFSET1	RW	0*	Address offset for address aperture[1] <sup>1)</sup>
11:8	SLV_OFFSET2	RW	0*	Address offset for address aperture[2] <sup>1)</sup>
15:12	SLV_OFFSET3	RW	0*	Address offset for address aperture[3] <sup>1)</sup>
19:16	SLV_OFFSET4	RW	0*	Address offset for address aperture[4] <sup>1)</sup>
23:20	SLV_OFFSET5	RW	0*	Address offset for address aperture[5] <sup>1)</sup>
27:24	SLV_OFFSET6	RW	0*	Address offset for address aperture[6] <sup>1)</sup>
31:28	SLV_OFFSET7	RW	0*	Address offset for address aperture[7] <sup>1)</sup>

1) Same encoding as for SLV\_OFFSET0

## 8.4.4 CREG\_TUN\_A\_OFFSET1: AXI Tunnel Address Offset Register 1

Address offset: 0x106C

Reset value: 0x0000\_0000

Table 62 CREG\_TUN\_A\_OFFSET1 register

Legend: * reset value				
Bit	Name	Access	Value	Description
3:0	SLV_OFFSET8	RW		Address offset for address aperture[8]
			0*	0*256MB
			1	1*256MB
			...	...
			15	15*256MB
7:4	SLV_OFFSET9	RW	0*	Address offset for address aperture[9] <sup>1)</sup>
11:8	SLV_OFFSET10	RW	0*	Address offset for address aperture[10] <sup>1)</sup>
15:12	SLV_OFFSET11	RW	0*	Address offset for address aperture[11] <sup>1)</sup>
19:16	SLV_OFFSET12	RW	0*	Address offset for address aperture[12] <sup>1)</sup>
23:20	SLV_OFFSET13	RW	0*	Address offset for address aperture[13] <sup>1)</sup>
27:24	SLV_OFFSET14	RW	0*	Address offset for address aperture[14] <sup>1)</sup>
31:28	SLV_OFFSET15	RW	0*	Address offset for address aperture[15] <sup>1)</sup>

1) Same encoding as for OFFSET8

## 8.4.5 CREG\_TUN\_A\_UPDATE: AXI Tunnel Update Register

Address offset: 0x1074

Reset value: 0x0000\_0000

Table 63 CREG\_TUN\_A\_UPDATE register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	UPDATE	RW1C	0	All the address aperture configuration registers (i.e. *_A_SLV) are double-buffered. The newly programmed values will be only be forwarded to the address decoder after writing a '1' to this bit.
31:1	Reserved	R	0x0*	

## 8.5 ARC Start Registers

### 8.5.1 CREG\_EM6\_START: ARC EM6 Start Register

Address offset: 0x1100

Reset value: depends on DIP switch settings on the ARC SDP Mainboard

Table 64 CREG\_EM6\_START register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	START	RW1C	0x0 *	Writing a '1' to this bit will generate a cpu_start pulse for ARC EM6
5:4	BM_SEL	RW		Boot mode select
			0x0 <sup>1)</sup>	Start ARC EM6 with debugger
			0x1	Start ARC EM6 via CPU Start button)
			0x2	Start ARC EM6 via CREG (SW)
			0x3	Start ARC EM6 autonomously after reset
8	POL	RW		Polarity of cpu_start pulse
			0x0	active low
			0x1 *	active high
31:9		R	0x0 *	Reserved

1) Reset value of *BM\_SEL* depends on DIP switch settings on the ARC SDP Mainboard

## 8.5.2 CREG\_770\_START: ARC 770D Start Register

Address offset: 0x1104

Reset value: depends on Mainboard DIP switch settings on the ARC SDP Mainboard

Table 65 CREG\_770\_START register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	START	RW1C	0x0 *	Writing a '1' to this bit will generate a cpu_start pulse for ARC 770D
5:4	BM_SEL	RW		Boot mode select
			0x0 <sup>1)</sup>	Start ARC 770D with debugger
			0x1	Start ARC 770D via push-button
			0x2	Start ARC 770D via CREG (SW)
8	POL	RW		Polarity of cpu_start pulse
			0x0	active low
			0x1 *	active high
31:9		R	0x0 *	Reserved

1) Reset value of *BM\_SEL* depends on DIP switch settings on the ARC SDP Mainboard

### 8.5.3 CREG\_221\_START: ARC AS221 Start Register

Address offset: 0x1108

Reset value: Depends on DIP switch settings on the ARC SDP Mainboard

Table 66 CREG\_221\_START register

Legend: * reset value				
Bit	Name	Access	Value	Description
0	START_1	RW1C	0x0*	Writing a '1' to this bit will generate a cpu_start pulse for ARC AS221 core #1
5:4	BM_SEL_1	RW		Boot mode select
			0x0 <sup>1)</sup>	Start ARC AS221 core #1 with debugger
			0x1	Start ARC AS221 core #1 via push-button
			0x2	Start ARC AS221 core #1 via CREG (SW)
8	POL_1	RW		Polarity of cpu_start pulse for ARC AS221 core #1
			0x0	active low
			0x1 *	active high
16	START_2	W	0x0*	Writing a '1' to this bit will generate a cpu_start pulse for ARC AS221 core #2
21:20	BM_SEL_2	RW		Boot mode select
			0x0 <sup>1)</sup>	Start ARC AS221 core #2 with debugger
			0x1	Start ARC AS221 core #2 via push-button
			0x2	Start ARC AS221 core #2 via CREG (SW)
24	POL_2	RW		Polarity of cpu_start pulse for ARC AS221 core #2
			0x0	active low
			0x1 *	active high
31:25		R	0x0 *	Reserved

1) Reset values of BM\_SEL\_0 and BM\_SEL\_1 depend on DIP switch settings on the ARC SDP Mainboard



## 8.6 ICTL Registers

This section describes the interrupt status register of the ICTL as an example for the location of a particular interrupt within the register. The other interrupt-related registers of the ICTL (e.g. for masking and clearing) use the same partitioning. Do not alter the interrupt level and interrupt type (active high, edge-sensitive).

Please note that the ICTL is based on a `dw_apb_gpio` module (see the “[Interrupt](#)” section). Refer to the GPIO drivers included in the software package for accessing the ICTL registers.

### 8.6.1 INT\_STATUS: Interrupt Status Register

Address offset: `0x2040`

Reset value: `0x0000_0000`

Table 67 GPIO port A output register (SWPORTA\_DR)

Legend: * reset value				
Bit	Name	Access	Value	Description
11:0		R	0x0 *	Reserved
12	DDR2_PLL_locked	R	0x0 *	Interrupt request not active
			0x1	DDR2 PLL is locked: Rising edge detected at DDR2 PLL locked signal.
13	DDR2_PLL_unlocked	R	0x0 *	Interrupt request not active
			0x1	DDR2 PLL loss of lock: Rising edge detected at inverted DDR2 PLL lock signal. This corresponds to a falling edge of the lock signal.
14	DDR2_PLL_lock_error	R	0x0 *	Interrupt request not active
			0x1	Rising edge detected at DDR2 PLL lock error signal.
15	SYS_PLL_locked	R	0x0 *	Interrupt request not active
			0x1	SYS PLL is locked: Rising edge detected at SYS PLL locked signal.
16	SYS_PLL_unlocked	R	0x0 *	Interrupt request not active
			0x1	SYS PLL loss of lock: Rising edge detected at inverted SYS PLL lock signal. This corresponds to a falling edge of the lock signal.
17	SYS_PLL_lock_error	R	0x0 *	Interrupt request not active
			0x1	Rising edge detected at SYS PLL lock error signal.
31:18		R	0x0 *	Reserved

## 8.7 GPIO Registers

### 8.7.1 SWPORTA\_DR: GPIO Port A Output Register

Address offset: 0x3000

Reset value: 0x0000\_0000

Table 68 GPIO port A output register (SWPORTA\_DR)

Legend: * reset value				
Bit	Name	Access	Value	Description
0	MB_LED2501	RW	0x0 *	LED2501 on the ARC SDP Mainboard is ON
			0x1	LED2501 on the ARC SDP Mainboard is OFF
1	MB_LED2502	RW	0x0 *	LED2502 on the ARC SDP Mainboard is ON
			0x1	LED2502 on the ARC SDP Mainboard is OFF
4:2		R/W	0x0 *	Reserved
5	MB_LED2503	RW	0x0 *	LED2503 on the ARC SDP Mainboard is ON
			0x1	LED2503 on the ARC SDP Mainboard is OFF
6	MB_LED2504	RW	0x0 *	LED2504 on the ARC SDP Mainboard is ON
			0x1	LED2504 on the ARC SDP Mainboard is OFF
9:7		R/W	0x0 *	Reserved
10	MB_LED2505	RW	0x0 *	LED2505 on the ARC SDP Mainboard is ON
			0x1	LED2505 on the ARC SDP Mainboard is OFF
11	MB_LED2506	RW	0x0 *	LED2506 on the ARC SDP Mainboard is ON
			0x1	LED2506 on the ARC SDP Mainboard is OFF
14:12		R/W	0x0 *	Reserved
15	MB_LED2507	RW	0x0 *	LED2507 on the ARC SDP Mainboard is ON
			0x1	LED2507 on the ARC SDP Mainboard is OFF
16	MB_LED2508	RW	0x0 *	LED2508 on the ARC SDP Mainboard is ON
			0x1	LED2508 on the ARC SDP Mainboard is OFF
23:17		R/W	0x0 *	Reserved
31:24		R	0x0 *	Reserved

## 8.7.2 EXT\_PORTA: GPIO Port A Input Register

Address offset: 0x3050

Reset value: 0xFC00\_0000; after executing `board_init()`

Table 69 GPIO port A input register (EXT\_PORTA)

Legend: * reset value				
Bit	Name	Access	Value	Description
11:0		R	0x0 *	Reserved
12	MB_IntrReq	R	0x0 *	Connected to the interrupt controller of the ARC SDP Mainboard. Can be used to provide an interrupt from the peripheral subsystem of the ARC SDP Mainboard to a core on the AXC001 CPU Card. This bit is configured as a level sensitive, active low interrupt.
15:13		R	0x0 *	Reserved
19:16		R	0xC *	Reserved
20	MB_SW2504	R/W	0x0	CPU Start button SW2504 on the ARC SDP Mainboard pressed
			0x1 *	CPU Start button SW2504 on the ARC SDP Mainboard not pressed
21	MB_SW2506	R/W	0x0	CPU Start button SW2506 on the ARC SDP Mainboard pressed
			0x1 *	CPU Start button SW2506 on the ARC SDP Mainboard not pressed
22	MB_SW2505	R	0x0	CPU Start button SW2505 on the ARC SDP Mainboard pressed
			0x1 *	CPU Start button SW2505 on the ARC SDP Mainboard not pressed
23	MB_SW2507	R	0x0	CPU Start button SW2507 on the ARC SDP Mainboard pressed
			0x1 *	CPU Start button SW2507 on the ARC SDP Mainboard not pressed
31:24		R	0x0 *	Reserved

*This chapter contains a list of specific terms used in this document and references for further reading.*

---

## 9.1 Glossary

**AHB**

Advanced High Performance Bus

**AXI**

Advanced eXtensible Interface

**CGU**

Clock Generator Unit

**DDR2**

Double Data Rate 2

**GPIO**

General Purpose Input/Output

**HW**

Hardware

**HAPS**

High performance ASIC Prototyping System; FPGA based prototyping system of Synopsys

**HAPSTrak\_II**

Standard (SAMTEC) connector type used on HAPS

**IC**

Integrated Circuit

**I<sup>2</sup>S**

Inter-IC Sound, serial bus interface standard for the transfer of audio data

**JTAG**

Joint Test Action Group

**R**

Read-only register

**RW**

Read-write register

**RW1C**

Read-write register; writing a one clears the corresponding bit

**SDP**

Software Development Platform

**SPDIF**

Sony/Philips Digital Interface

**SDRAM**

Synchronous Dynamic Random Access Memory

**SRAM**

Static Random Access Memory

**SW**

Software

---

## 9.2 References

- [1] *HapsTrak-II standard*, [http://www.samtec.com/Documents/WebFiles/Technical\\_Library/Reference/Articles/HapsTrak\\_II.pdf](http://www.samtec.com/Documents/WebFiles/Technical_Library/Reference/Articles/HapsTrak_II.pdf)
- [2] *C/C++ Programmer's Guide for the MetaWare Compiler*
- [3] *Synopsys DesignWare dw\_apb\_gpio Databook* <http://www.synopsys.com>
- [4] *Linux kernel upstream* <http://www.kernel.org>
- [5] *ARC Linux github site* <http://github.com/foss-for-synopsys-dwc-arc-processors/linux>
- [6] *ARC SDP download webpage*  
*You have received the corresponding URL during the purchasing process*
- [7] *ARC SDP Mainboard User Guide*  
*This document can be downloaded from the ARC SDP download webpage.*


## Appendix A

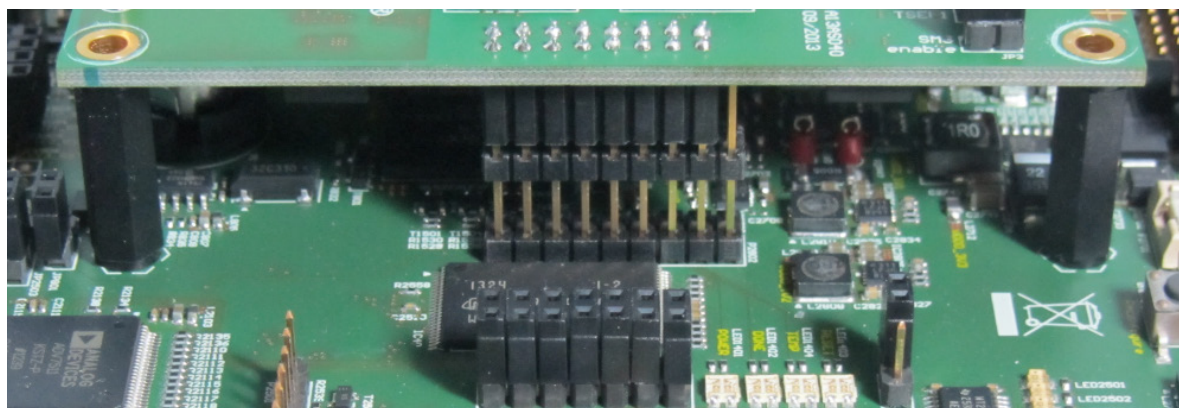
*This appendix describes the process of mounting a stand-alone AXC001 CPU Card on an ARC SDP Mainboard, and thus, assembling a complete AXS101 Software Development Platform.*

### A.1 Mounting AXC001 CPU Card on ARC SDP Mainboard

Take the following steps to mount the AXC001 CPU Card on an ARC SDP Mainboard:

1. Make sure that the ARC SDP Mainboard is switched off
2. Mount the AXC001 CPU Card on the ARC SDP Mainboard and make sure that the power supply connector and the HapsTrak-II connectors for the CPU Card are connected properly.

 **Note** Please note that the power supply connector of the CPU Card has fewer pins than the header on the ARC SDP Mainboard. [Figure 72](#) shows the correct alignment.



*Figure 72 Alignment of the Power Supply Connector*

3. Make sure that the CPU Card specific DIP switches on the ARC SDP Mainboard are set according to [Figure 73](#) below.

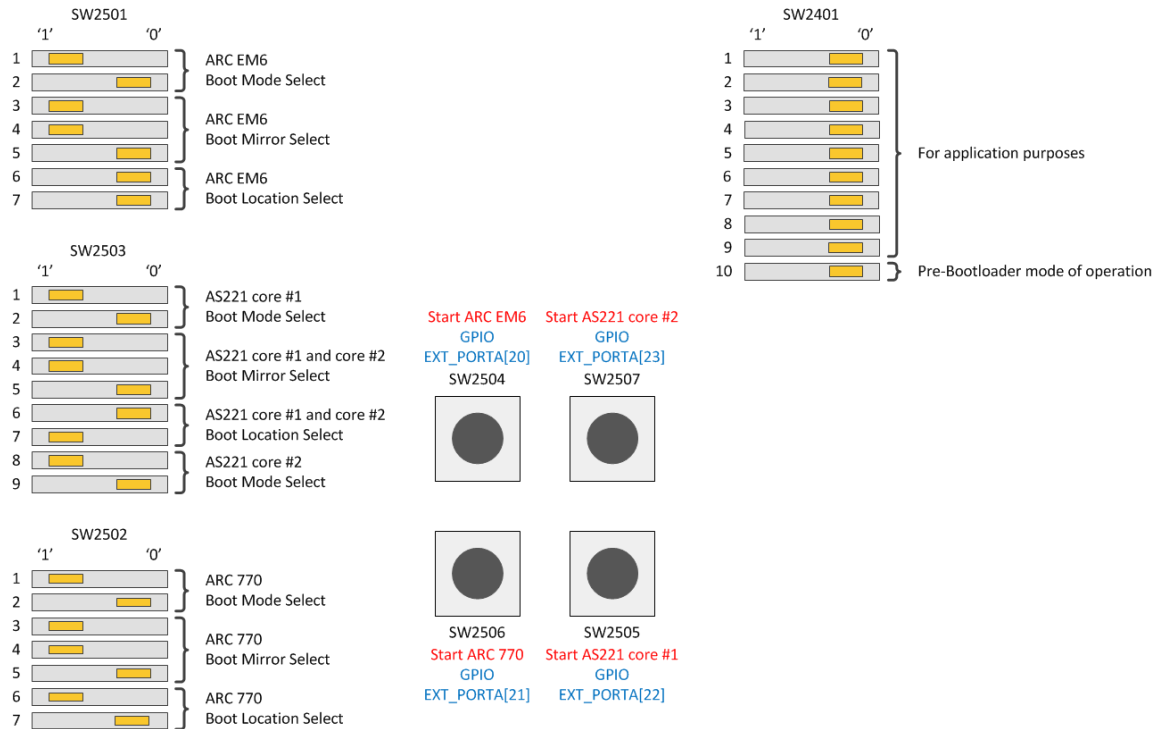


Figure 73 Default settings of the DIP switches on the ARC SDP Mainboard

This appendix describes how to install and configure PuTTY utility, a serial console that can be used for debugging.

### B.1 Installing and Configuring PuTTY

PuTTY is a serial console that can be used intermediately as a simple debug console, when the MetaWare Development Toolkit has not yet been installed. This is an optional step. It is only needed if you are interested in the console output of the built-in self-test.

1. Download `putty.exe` from <http://www.putty.org>
2. Make sure that you have connected the USB cable to your computer and that the USB device drivers have been installed as described in the *ARC SDP Mainboard User Guide [7]*.
3. Open the Windows Control Panel. In the category **Hardware and Sound**, click **View devices and printers**, then **Digilent Adept USB Device**.
4. The **Digilent Adept USB Device Properties** windows opens. Select the **Hardware** tab and take note of the COM port assigned to the USB Serial Port.

The example below uses the COM6 port.

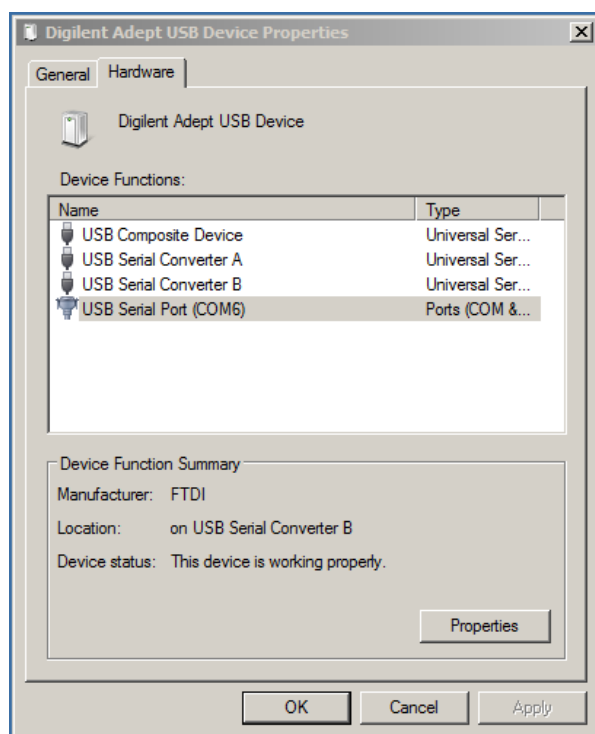


Figure 74 Identification of COM port



5. Execute `putty.exe`
6. The **PuTTY Configuration** window appears. Select the **Connection type** to `Serial`
7. Enter the name of the COM port in the **Serial line** field
8. Set the **Speed** field to `115200` as shown in [Figure 75](#).

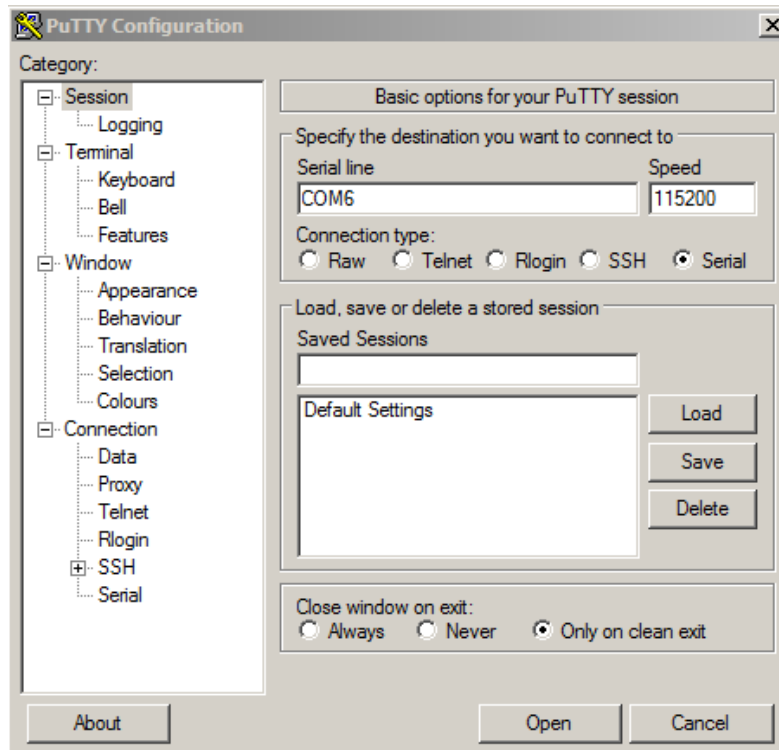


Figure 75 PuTTY configuration

9. Click on **Open** to launch the PuTTY terminal.