



# **DesignWare ARC SDP Mainboard User Guide**

---

Version 6301-013 April 2017

# Copyright Notice and Proprietary Information Notice

© 2017 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

## 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.

## Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

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

# Contents

---

Contents.....	3
List of Figures .....	7
List of Tables.....	9
1 Getting Started.....	11
2 Hardware Functional Description .....	12
2.1 Board Overview .....	12
2.2 Hardware Interface Overview .....	14
2.2.1 ARC CPU Card Interfaces.....	15
2.2.2 Connectivity Interfaces .....	16
2.2.3 Serial Interfaces .....	16
2.2.4 Audio Interfaces .....	17
2.2.5 HAPS Extension Interface.....	18
2.2.6 Peripheral Extension Interfaces.....	18
2.2.7 Debug Interfaces .....	20
2.2.8 SD-Card Slot .....	20
2.2.9 Miscellaneous Interfaces.....	21
2.3 CPLD Board Supervisor .....	22
2.4 USB Dataport .....	23
2.4.1 JTAG Debug Channel .....	23
2.4.2 Communication Channel.....	24
2.5 JTAG Programming Header .....	25
2.6 FPGA Temperature and Fan Control.....	28
2.7 Clock Generation .....	29
2.7.1 FPGA Reference Clock .....	30
2.7.2 AMBA Clocks .....	30
2.7.3 Tunnel Clocks.....	30
2.7.4 PGU Clock .....	30
2.7.5 Storage Clocks .....	31
2.7.6 Serial Connectivity Clocks.....	32
2.7.7 Audio Reference Clock.....	32
2.7.8 ARC CPU Card Clock .....	32
2.8 Ethernet .....	33
2.9 USB .....	34
2.10 HDMI Output.....	34
2.10.1 HDMI Audio Content .....	35
2.11 Audio Support.....	37
2.11.1 Analog Stereo Inputs/Outputs .....	39
2.11.2 Analog 8-Channel Audio Output.....	40
2.11.3 S/PDIF Inputs and Outputs .....	41
2.11.4 Audio PLL.....	42
2.11.5 Advanced Audio Use Cases.....	43
2.12 SD-Card.....	47
2.13 Real Time Clock .....	48
2.14 Internal I <sup>2</sup> C Bus .....	49

2.15 I <sup>2</sup> C Interfaces .....	50
2.15.1 Routing Options for External I <sup>2</sup> C Interfaces.....	50
2.15.2 Using the Pmod4 Interface in I <sup>2</sup> C Mode .....	50
2.16 UART (RS232) Interfaces .....	52
2.16.1 Routing Options for External UART Interfaces.....	53
2.16.2 Default UART Interfaces .....	53
2.16.3 Using Extension Interfaces in UART Mode .....	57
2.17 SPI Interfaces .....	60
2.17.1 Routing Options for External SPI Interfaces.....	61
2.17.2 Default SPI Connectors .....	61
2.17.3 Using Extension Interfaces in SPI Mode .....	61
2.18 Debug .....	65
2.18.1 Using the USB Dataport for Debugging.....	66
2.18.2 Using an Ashling Opella XD Probe.....	66
2.18.3 Using Lauterbach Probes .....	68
2.18.4 Using a Digilent Probe.....	69
2.19 Push Buttons, Switches and LEDs .....	70
2.19.1 Push Buttons.....	71
2.19.2 Switches .....	73
2.19.3 LEDs .....	74
2.20 Seven-Segment Display .....	79
2.21 On-Board Memories .....	80
2.21.1 I <sup>2</sup> C EEPROM.....	80
2.21.2 SPI Flash Application Memory .....	80
2.21.3 FPGA-Configuration FLASH.....	80
2.21.4 NAND Flash .....	81
2.21.5 RAM .....	81
2.22 Power Supply .....	81
2.23 Peripheral Subsystem FPGA Overview .....	82
2.23.1 AXI Tunnel .....	83
2.23.2 Interrupt Controller (ICTL) .....	83
2.23.3 Clock Generator Unit (CGU) .....	85
2.23.4 DMA Controller (DMAC) .....	85
2.23.5 SPI Controller for SPI Flash .....	86
2.23.6 GPIO Modules.....	86
2.24 HAPS HapsTrak-3 Extension .....	87
2.25 Pmod Extension Options .....	87
2.25.1 Pmod0 Connector .....	91
2.25.2 Pmod1 Connector .....	92
2.25.3 Pmod2 Connector .....	92
2.25.4 Pmod3 Connector .....	93
2.25.5 Pmod4 Connector .....	93
2.25.6 Pmod Interface Type 1 (GPIO).....	94
2.25.7 Pmod Interface Type 2 (SPI).....	94
2.25.8 Pmod Interface Type 2a (Expanded SPI).....	94
2.25.9 Pmod Interface Type 3 (UART).....	94

2.25.10 Pmod Interface Type 4 (UART) .....	94
2.25.11 Pmod Interface Type 4a (Expanded UART) .....	95
2.25.12 Pmod I <sup>2</sup> C Interface .....	95
2.26 Other Extension Options .....	95
2.26.1 Extension0 Connector .....	97
2.26.2 Extension1 Connector .....	99
2.26.3 Extension2 Connector .....	100
2.26.4 Extension3 Connector .....	102
2.27 Power Supply for Extension Boards .....	104
2.28 Mounting an ARC CPU Card .....	105
2.29 Backup Battery .....	105
3 System Memory Map .....	106
3.1 Controlling the Memory Map .....	106
3.2 Memory Map After Reset .....	109
3.3 Memory Map of Peripheral Subsystem .....	110
4 Software Interfaces .....	112
4.1 Clock Generation Registers .....	114
4.1.1 TUNNEL PLL .....	114
4.1.2 PGU PLL .....	119
4.1.3 Audio I2S Clock .....	123
4.2 Control Registers .....	125
4.2.1 AXI_m_SLV_SEL0 Register .....	125
4.2.2 AXI_m_SLV_SEL1 Register .....	126
4.2.3 AXI_m_SLV_OFFSET0 Register .....	127
4.2.4 AXI_m_SLV_OFFSET1 Register .....	128
4.2.5 AXI_UPDATE Register .....	129
4.2.6 AXI_UPDATE_CLR Register .....	129
4.2.7 AXI_UPDATE_STAT Register .....	130
4.2.8 TUN_CTRL Register .....	130
4.2.9 TUN_STAT Register .....	131
4.2.10 PMOD_MUX_CTRL Register .....	131
4.2.11 AUDIO_CLK_MUX_CTRL Register .....	132
4.2.12 SPI_FLASH_MUX_CTRL Register .....	133
4.2.13 SW_RESET Register .....	133
4.3 ICTL Registers .....	134
4.3.1 ICTL_INT_STATUS: Interrupt Status Register .....	134
4.4 GPIO Registers .....	135
4.4.1 GPIO0_SWPORTA_DR: GPIO0 Port A Output Register .....	135
4.4.2 GPIO0_SWPORTB_DR: GPIO0 Port B Output Register .....	136
4.4.3 GPIO0_SWPORTC_DR: GPIO0 Port C Output Register .....	137
4.4.4 GPIO0_EXT_PORTA: GPIO0 Port A Input Register .....	138
4.4.5 GPIO0_EXT_PORTB: GPIO0 Port B Input Register .....	139
4.4.6 GPIO0_EXT_PORTC: GPIO0 Port C Input Register .....	140
4.4.7 GPIO1_SWPORTC_DR: GPIO1 Port C Output Register .....	141
4.4.8 GPIO1_EXT_PORTA: GPIO1 Port A Input Register .....	142
4.4.9 GPIO1_EXT_PORTB: GPIO1 Port B Input Register .....	143

5 Software Installation.....	144
5.1    USB-JTAG and USB-UART Driver Installation .....	144
5.2    AXS Communicator Tool – <code>axs_comm</code> .....	144
5.2.1    Tool Overview and Installation Instructions .....	144
5.2.2    Usage .....	145
Appendix A.....	146
A.1    Jumper Overview .....	146
Appendix B.....	154
B.1    Programming the SPI Flash Memory .....	154
B.1.1    Using <code>axs_comm</code> for SPI FLASH programming .....	154
Appendix C .....	157
C.1    Configuring the Real Time Clock .....	157
Appendix D .....	158
D.1    HAPS Trak-3 Extension Connector Pins .....	158
Appendix E.....	160
E.1    Using GPIO Pins at the ARC CPU Card Connectors.....	160
E.2    ARC CPU Card Power Supply Connector .....	161
Appendix F.....	163
F.1    Adding System Extensions via HAPS Extension Interface .....	163
Appendix G .....	166
G.1    Replacing the Fuse.....	166
G.2    Backup Battery .....	167
Glossary and References .....	168
Glossary.....	168
References .....	169

# List of Figures

---

Figure 1	Principal block diagram.....	13
Figure 2	ARC SDP Mainboard Hardware Interfaces .....	14
Figure 3	Overview of peripheral extension interfaces .....	19
Figure 4	Location of the USB Dataport connector and the corresponding LEDs and jumpers .....	23
Figure 5	Jumper setting for connecting the debugger via the USB Dataport.....	24
Figure 6	Jumper setting for connecting the debugger via debug cable connectors.....	24
Figure 7	Location of the JTAG In and JTAG Out connectors and the corresponding jumpers .....	25
Figure 8	Pinout diagrams of the JTAG In and JTAG Out connectors .....	26
Figure 9	Location of the TEMP LED .....	28
Figure 10	Mainboard clock architecture .....	29
Figure 11	Selecting SDIO card clock frequency via the SDIO controller .....	31
Figure 12	Location of Ethernet connector and Ethernet LEDs .....	33
Figure 13	Location of HDMI-related connectors and jumpers on the Mainboard .....	35
Figure 14	Pinout diagram of the HDMI 8-Channel I2S Slave Input (JP2101).....	36
Figure 15	ARC SDP Mainboard audio interfaces .....	37
Figure 16	Jumper settings for using the on-board stereo codecs (default).....	40
Figure 17	Location of the stereo inputs and outputs and of the corresponding jumpers .....	40
Figure 18	Location of the 8-Channel Output and the corresponding jumpers .....	41
Figure 19	Location of the S/PDIF connectors .....	42
Figure 20	Audio PLL jumper settings .....	43
Figure 21	Location of the PLL_CLK Out connector and of the corresponding jumpers .....	43
Figure 22	Jumper settings for using an external custom stereo codec.....	44
Figure 23	Location of the External Stereo Codec I2S In / Out connector and the corresponding jumpers.....	44
Figure 24	Pinout diagram of the external stereo codec I <sup>2</sup> S master input / output.....	45
Figure 25	Location of the External 8-Channel Codec I2S Out connector .....	46
Figure 26	Pinout diagram of the External 8-Channel Codec I2S Out connector .....	46
Figure 27	Jumper settings for SD-card with and without authentication.....	48
Figure 28	Location of SD-card slot and corresponding jumper.....	48
Figure 29	Location of the Pmod4 Connector Supporting I <sup>2</sup> C Mode .....	51
Figure 30	Pinout diagram of the Pmod4 connector (standard 8-pin interface) .....	51
Figure 31	Pinout diagram of the Pmod4 connector (4-pin subsets).....	52
Figure 32	Location of the UART0 and UART1 connectors .....	54
Figure 33	Pinout diagram of the UART0 DB9 connector .....	54
Figure 34	Pinout diagram of the UART1 connector .....	56
Figure 35	Location of Pmod and Extension connectors supporting UART mode .....	57
Figure 36	Pinout diagrams of the Pmod0 connector in UART mode (UART0 / UART1) .....	58
Figure 37	Pinout diagrams of the Extension0 connector in UART mode (IART0 / UART1) .....	58
Figure 38	Voltage level selection for the Extension0 connector .....	59
Figure 39	Pinout diagram of the Pmod2 connector in UART mode (UART0).....	59
Figure 40	Pinout diagram of the Extension2 connector in UART mode (UART0) .....	60
Figure 41	Voltage level selection for the Extension2 connector .....	60
Figure 42	Location of Pmod and Extension connectors supporting SPI mode .....	62
Figure 43	Pinout diagram of the Pmod2 connector in SPI mode (SPI1).....	62
Figure 44	Pinout diagram of the Extension2 connector in SPI mode (SPI1) .....	63
Figure 45	Voltage level selection for the Extension2 connector .....	63
Figure 46	Pinout diagram of the Pmod3 connector in SPI mode (SPI0 / SPI1) .....	64
Figure 47	Pinout diagram of the Extension3 connector in SPI mode (SPI0 / SPI1) .....	64
Figure 48	Voltage level selection for the Extension3 connector .....	65

Figure 49	Location of the debug interfaces and the corresponding jumpers .....	66
Figure 50	Jumper setting for connecting the debugger via the USB Dataport.....	66
Figure 51	Pinout diagram and jumper settings for connecting an Ashling probe.....	68
Figure 52	Pinout diagram and jumper settings for connecting a Lauterbach probe .....	69
Figure 53	Pinout diagram and jumper settings for connecting a Digilent probe .....	70
Figure 54	Devices on GPIO1 .....	71
Figure 55	Location of the push buttons.....	71
Figure 56	Location of the switches .....	73
Figure 57	Location of the Board Status LEDs.....	76
Figure 58	Location of the GPIO LEDs .....	77
Figure 59	Location of the CPU LEDs .....	78
Figure 60	Location of the USB Dataport Mode LEDs .....	79
Figure 61	Jumper settings for NAND flash write protection .....	81
Figure 62	Location of the NAND Flash write protection header.....	81
Figure 63	Location of power inlet, power switch and power status LEDs .....	82
Figure 64	Principle block diagram of the peripheral subsystem .....	83
Figure 65	P Devices on GPIO0.....	88
Figure 66	Pinout diagram of the Pmod0, Pmod1, Pmod2, Pmod3 and Pmod4 connectors .....	90
Figure 67	Pinout diagram of the Extension0, Extension1, Extension2 and Extension3 connectors .....	97
Figure 68	Pinout diagram of the Extension Power Supply connector.....	97
Figure 69	Location of the Extension0 connector and its associated jumpers .....	98
Figure 70	Voltage level selection for the Extension0 connector .....	99
Figure 71	Location of the Extension1 connector and its associated jumpers .....	99
Figure 72	Voltage level selection for the Extension1 connector .....	100
Figure 73	Location of the Extension2 connector and its associated jumpers .....	101
Figure 74	Voltage level selection for the Extension2 connector .....	102
Figure 75	Location of the Extension3 connector and its associated jumpers .....	102
Figure 76	Voltage level selection for the Extension3 connector .....	103
Figure 77	Pinout diagram of the Extension Power Supply connector.....	104
Figure 78	Jumper location overview .....	146
Figure 79	Power supply of the ARC CPU Card .....	162
Figure 80	Location of the fuse .....	166

# List of Tables

Table 1	ARC CPU Card interfaces .....	15
Table 2	Connectivity Interfaces .....	16
Table 3	Serial interfaces .....	16
Table 4	Peripheral extension interfaces .....	18
Table 5	Debug interfaces .....	20
Table 6	Miscellaneous interfaces .....	21
Table 7	Pin description of the JTAG In connector .....	26
Table 8	Pin description of the JTAG Out connector .....	27
Table 9	AMBA clocks .....	30
Table 10	AXI tunnel clocks (after reset) .....	30
Table 11	AXI tunnel clocks (after pre-boot) .....	30
Table 12	PGU reference clock .....	31
Table 13	Storage reference clocks .....	31
Table 14	Serial Connectivity reference clocks .....	32
Table 15	Audio Reference Clocks .....	32
Table 16	Ethernet LEDs .....	33
Table 17	Pin Description of the HDMI 8-Channel I <sup>2</sup> S Slave Input .....	35
Table 18	Audio interfaces on the Mainboard .....	38
Table 19	Pin description of the external stereo codec I <sup>2</sup> S master input / output .....	44
Table 20	Pin description of the External 8-Channel Codec I2S Out connector .....	46
Table 21	Internal I <sup>2</sup> C bus slave addresses .....	49
Table 22	I <sup>2</sup> C Interface selection .....	50
Table 23	UART Interface selection .....	53
Table 24	Pin description of the UART0 (DB9) connector .....	55
Table 25	Pin description of the 6-pin UART1 connector .....	56
Table 26	SPI Interface selection .....	61
Table 27	Pin description of the CPU Debug Ashling/Lauterbach connector in Ashling mode .....	66
Table 28	Pin description of the CPU Debug Ashling/Lauterbach connector in Lauterbach mode .....	68
Table 29	Pin description of the CPU Debug Digilent connector .....	70
Table 30	List of push buttons .....	72
Table 31	List of switches .....	73
Table 32	Board status LEDs .....	75
Table 33	GPIO LEDs (controlled by Mainboard GPIO1 module) .....	76
Table 34	USB Dataport LEDs .....	79
Table 35	ICTL Interrupt mapping .....	84
Table 36	DMAC flow control interface mapping .....	85
Table 37	GPIO1 I/O register function .....	86
Table 38	Pmod pin multiplexing overview .....	88
Table 39	UART signal description .....	89
Table 40	SPI Signal Description .....	90
Table 41	I <sup>2</sup> C Signal Description .....	90
Table 42	Pin description of the Pmod0 connector .....	91
Table 43	Pin description of the Pmod1 connector .....	92
Table 44	Pin description of the Pmod2 connector .....	92
Table 45	Pin description of the Pmod3 connector .....	93
Table 46	Pin description of the Pmod4 connector .....	93

Table 47	Extension connector pin multiplexing overview .....	96
Table 48	UART signal description .....	96
Table 49	SPI Signal Description .....	97
Table 50	Pin description of the Extension0 connector.....	98
Table 51	Pin description of the Extension1 connector.....	100
Table 52	Pin description of the Extension2 connector.....	101
Table 53	Pin description of the Extension3 connector.....	103
Table 54	Pin description of the extension power supply connector.....	104
Table 55	AXI Masters .....	106
Table 56	Target slaves .....	107
Table 57	Slave Select and Address Offset Registers .....	107
Table 58	Address Offset Settings for Example 3 .....	108
Table 59	Memory map after reset.....	109
Table 60	Peripheral memory map .....	110
Table 61	Peripheral registers overview .....	112
Table 62	I2S sclk Divider Settings .....	123
Table 63	Jumper functional overview and default settings .....	146
Table 64	Pin description of the HAPS Trak-3 extension connectors J3 and J4 .....	158
Table 65	GPIO field of ARC CPU Card Interface .....	160
Table 66	Pin description of the ARC CPU Card power supply connector .....	161

# 1

## Getting Started

---

*The getting started procedure depends on the type of ARC CPU Card used. Refer to the documentation of a corresponding ARC CPU Card for more information.*

---

## Hardware Functional Description

*This chapter provides information about the hardware of the ARC SDP Mainboard.*

### 2.1 Board Overview

The ARC SDP Mainboard is the foundation of the ARC Software Development Platform. It provides connectors and power supply for the ARC CPU Card and provides the main infrastructure and connectivity.

The ARC Software Development Platform is a powerful, high-speed development platform, enabling real-time software development and validation, code porting, software debugging, and system analysis. Readily available DesignWare IP has been used to build the ARC Software Development Platform, which is thus well-suited for SoC prototyping including ARC CPUs.

The ARC CPU Card is connected to the ARC SDP Mainboard through a 18-pin header for power supply and two HapsTrak-3 connectors for a fast AXI tunnel operating at up to 150 MHz and supplying additional clocks, data and control signals. The AXI tunnel is the backbone of the ARC Software Development Platform. It connects the CPUs on the ARC CPU Card with a peripheral subsystem that is implemented on an FPGA on the ARC SDP Mainboard. The mainboard FPGA is located in the center of the mainboard, under the fan.

A rich set of interfaces is available including audio, USB 2.0 host, HDMI, Ethernet, and several serial protocols.

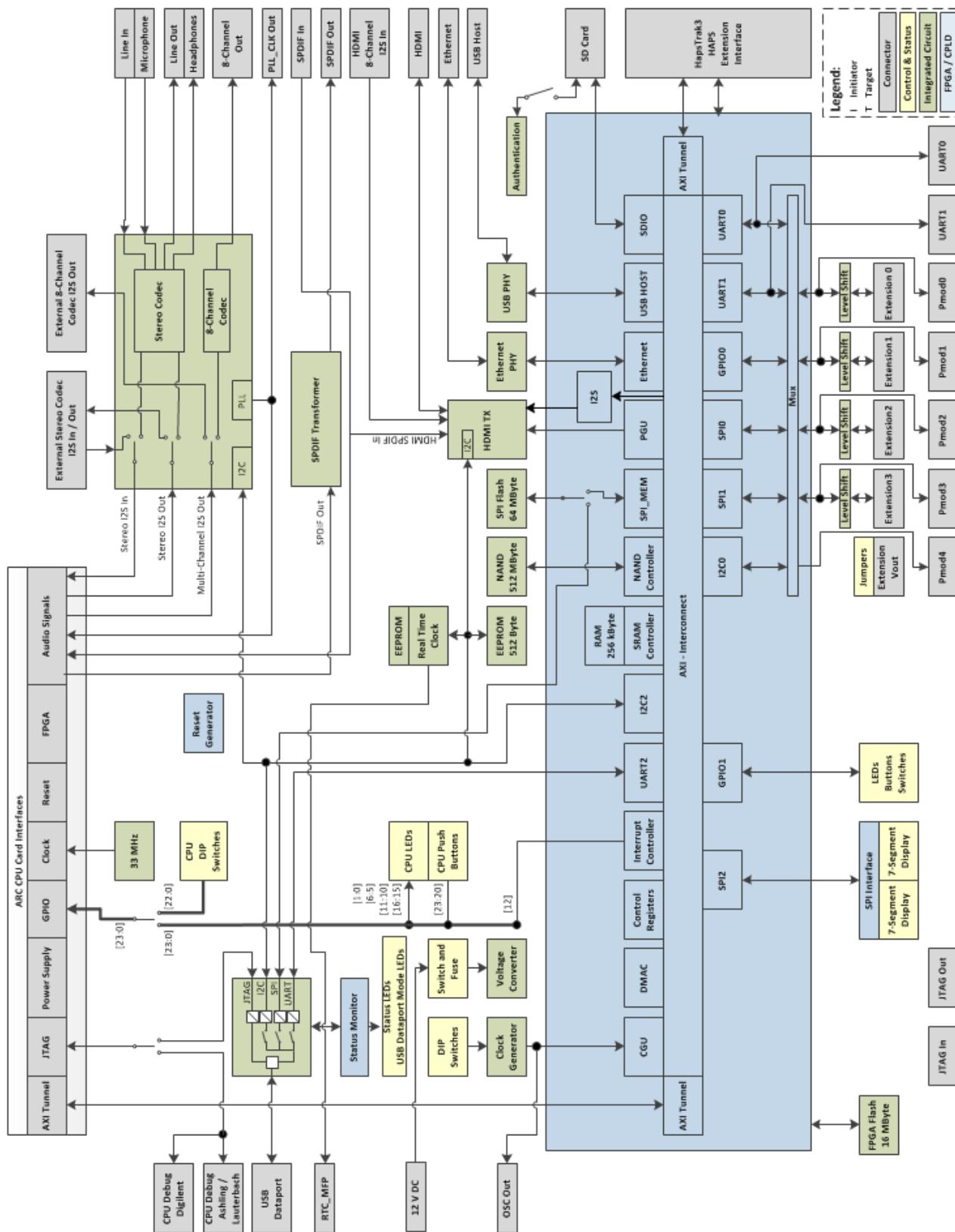
The ARC SDP Mainboard features an SD-card reader and includes multiple memories for storing boot code, application code, operating system, and data.

Two HapsTrak-3 connectors allows connecting a HAPS system to the ARC SDP Mainboard also using a fast AXI tunnel. Custom hardware extensions such as additional application specific peripherals can use this AXI tunnel for communicating seamlessly with the cores, memories, and peripherals available on the ARC SDP Mainboard and ARC CPU Card. Custom hardware extensions can thus be integrated conveniently and operated at high speed.

Multiple other extension interfaces such as five Digilent Pmod™ compatible connectors and four extension headers with selectable voltage levels are available as well. These extension interfaces support GPIO, SPI, I<sup>2</sup>C, and UARTs.

The board operates on a single 12V DC power supply. An AC power adapter is included. All required voltage levels and clocks are generated on the board, so that no lab equipment is required to get started with development.

Figure 1 shows the principal block diagram of the ARC SDP Mainboard.



*Figure 1 Principal block diagram*

## 2.2 Hardware Interface Overview

Figure 2 shows the position of the hardware interfaces on the ARC SDP Mainboard.

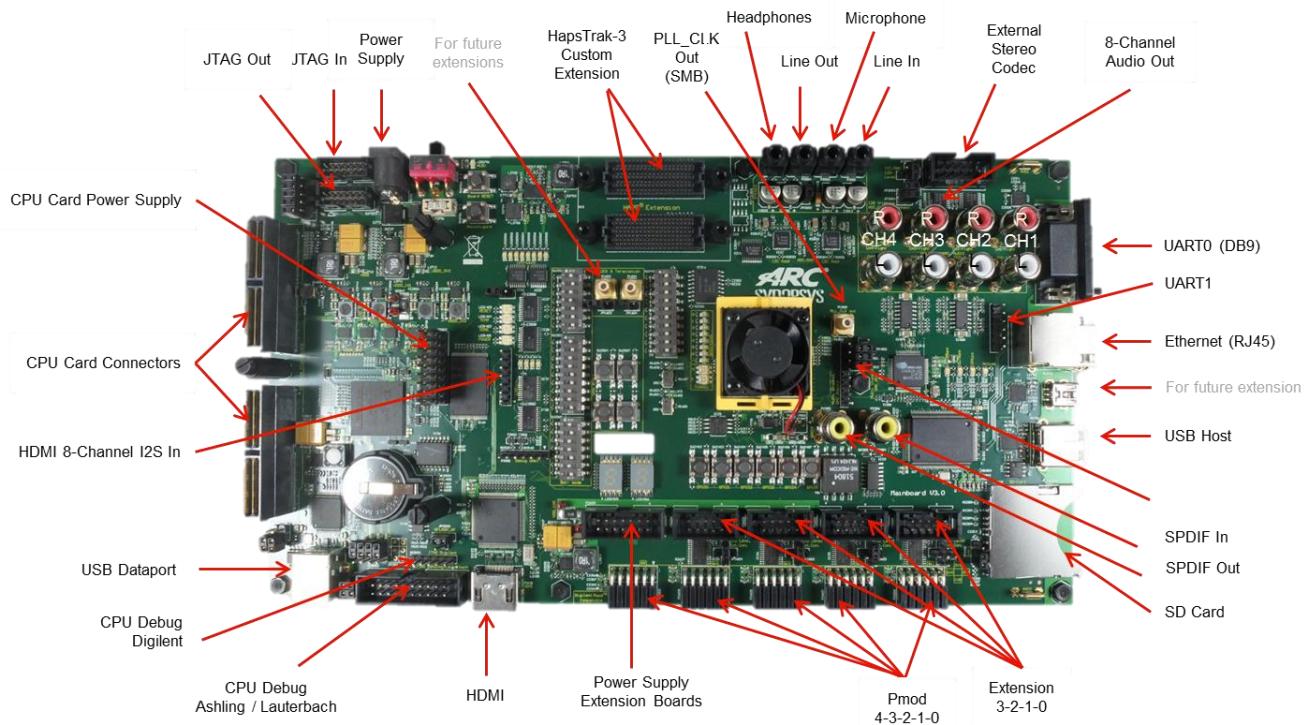


Figure 2 ARC SDP Mainboard Hardware Interfaces

The ARC SDP Mainboard features many hardware interfaces, which can be grouped in the following categories:

- ARC CPU Card interfaces
- Connectivity interfaces
- Serial interfaces
- Audio interfaces
- Extension interfaces
- Debug interfaces
- SD-card slot
- Miscellaneous interfaces

## 2.2.1 ARC CPU Card Interfaces

The ARC SDP Mainboard communicates with the ARC CPU Card using two HapsTrak-3 connectors, which carry signal groups such as

- AXI tunnel
- GPIO
- Clock
- Reset
- Audio signals
  - Stereo I<sup>2</sup>S slave input
  - Stereo I<sup>2</sup>S slave input
  - 8-channel I<sup>2</sup>S slave output
  - SPDIF input
  - SPDIF output

Additionally, the Mainboard provides power to the CPU Card via a separate connector.

Table 1 below provides a brief overview of the ARC CPU Card interfaces.

Table 1 ARC CPU Card Interfaces

Name	Description	Voltage Level
CPU Card Connectors	Two HapsTrak-3 connectors	
CPU Card Power Supply	18-pin header for power supply	12V 3V3 2V5 1V8 1V1

## 2.2.2 Connectivity Interfaces

The ARC SDP Mainboard supports the following communication protocols:

- Ethernet 10/100 Mb/s
- HDMI TX
- USB 2.0 Host

Table 2 provides an overview of the connectivity interfaces. Additional serial protocols such as SPI, I<sup>2</sup>C, and UART are supported as well.

See "[Serial Interfaces](#)" and "[Peripheral Extension Interfaces](#)" sections for an overview about the serial interfaces of the Mainboard.

*Table 2 Connectivity Interfaces*

Name	Description	Voltage Level
Ethernet	RJ45 Ethernet jack	
HDMI	HDMI type A female connector for HDMI TX	
USB Host	USB 2.0 host interface (type A)	

## 2.2.3 Serial Interfaces

Table 3 provides an overview of the dedicated serial interfaces on the ARC SDP Mainboard.

Additional I<sup>2</sup>C, SPI, and UART ports are available at the multi-purpose extension interfaces Pmod0 to Pmod4 and Extension0 to Extension3. See the "[Peripheral Extension Interfaces](#)" section for an overview.

*Table 3 Serial interfaces*

Name	Description	Voltage Level
UART0	DB9 connector for UART0	3V3
UART1	6-pin header for UART1 Pinout compatible with FTDI USB to TTL Serial Cable [6].	3V3

## 2.2.4 Audio Interfaces

The ARC SDP Mainboard has the following external audio interfaces:

- Stereo line out
- Stereo headphones out
- Stereo line in
- Stereo microphone in
- 8-channel out
- SPDIF input
- SPDIF output

The digital sources for these audio outputs and the sinks for the inputs are located on the ARC CPU Card. The following interfaces exist in the ARC CPU Card header:

- Stereo I<sup>2</sup>S slave input
- Stereo I<sup>2</sup>S slave output
- 8-channel I<sup>2</sup>S slave input
- SPDIF input
- SPDIF output

For advanced use cases, external custom codecs can be used instead of the on-board audio codecs. The ARC SDP Mainboard features a stereo I<sup>2</sup>S master input, a stereo I<sup>2</sup>S master output, and an eight-channel I<sup>2</sup>S master output to connect your custom codecs.

In addition, the output of the on-board audio PLL is accessible at an SMB connector.

See the “[Audio Support](#)” section for details.

The audio output of the on-board HDMI connector can be provided by the SPDIF-IN RCA signal, or by the eight-channel I<sup>2</sup>S slave in the FPGA. See the “[HDMI Audio Content](#)” section for details.

## 2.2.5 HAPS Extension Interface

The ARC SDP Mainboard includes two HapsTrak-3 connectors for connecting prototyping systems from the HAPS product family. These FPGA-based prototyping systems allow further system extensions (for example, CPUs, memories, peripherals).

The connector provides a high-speed AXI tunnel and a few control signals. The voltage level swing is 1.8 Volts, supporting all HAPS systems.

The HAPS Extension Interface should not be used to connect a HAPS daughter board.

## 2.2.6 Peripheral Extension Interfaces

The ARC SDP Mainboard features five 12-pin Pmod connectors plus four 10-pin extension headers, which can be used to Pmod or other extension modules with additional peripherals to the system. Table 4 provides an overview of the available peripheral extension interfaces.

A total of 40 I/O pins of the peripheral subsystem are routed to these connectors, where pin-sharing is done between the `Pmod` connectors and the `Extension` connectors as shown in Figure 3.

*Table 4 Peripheral extension interfaces*

Name	Description	I/O Voltage Level
Pmod 0 Pmod 1 Pmod 2 Pmod 3 Pmod 4	Five Pmod connectors 2x6	3V3
Extension 0 Extension 1 Extension 2 Extension 3	Four 2x5 headers for hardware extension boards	Selectable with jumper: 5V0 3V3 2V5 1V8
Power Supply Extension Boards	14-pin header with power supply for extension boards  Two output pins per voltage. Six ground pins.	5V0 3V3 2V5 1V8

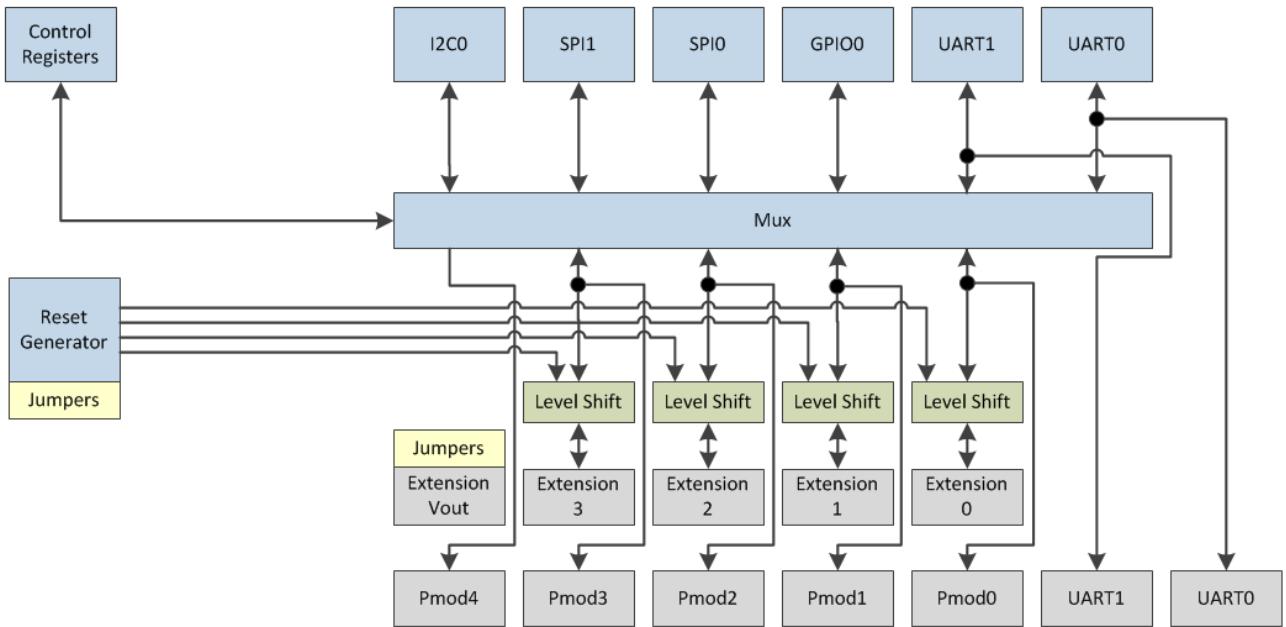


Figure 3 Overview of peripheral extension interfaces

#### 2.2.6.1 Pmod Connectors

The Pmod interfaces natively support the following Pmod interface types:

- Pmod Interface Type 1 (GPIO)
- Pmod Interface Type 2 (SPI)
- Pmod Interface Type 2A (expanded SPI)
- Pmod Interface Type 4 (UART)
- Pmod Interface Type 4A (expanded UART)
- I<sup>2</sup>C

Modules featuring the deprecated Pmod Interface 3 (UART) can be connected to the ARC SDP Mainboard using a crossover cable (available from Digilent Inc.) or flying lead cables (see [3]).

For more information on the on-board Pmod interfaces, refer to the following sections:

- ["Pmod Extension Options"](#) for the Pmod interfaces in general
- “
- ["UART \(RS232\) Interfaces"](#) for using Pmod interfaces in UART mode
- ["SPI Interfaces"](#) for using Pmod interfaces in SPI mode
- ["I<sup>2</sup>C Interfaces"](#) for using Pmod interfaces in I<sup>2</sup>C mode

### 2.2.6.2 Extension Connectors

The headers Extension0, Extension1, Extension2, and Extension3 have jumper-selectable voltage levels and support the following interface types:

- GPIO
- SPI
- UART

These extension connectors provide the same logical signals as the corresponding Pmod connectors Pmod0, Pmod1, Pmod2, and Pmod3. In addition to the Pmod connectors the Extension connectors provide a reset output and have selectable voltage levels.

For more information on the Extension connectors, refer to the "[Other Extension Options](#)" section.

A separate header for the power supply of the extension boards is available.

### 2.2.7 Debug Interfaces

Multiple interfaces are available for software debugging. The ARC SDP Mainboard features a USB Dataport and a USB converter chip providing a JTAG debug channel and a UART console through a single USB cable. Alternatively, debug probes by Digilent, Ashling or Lauterbach can be connected to one of the two CPU debug connectors.

*Table 5 Debug interfaces*

Name	Description	Voltage Level
USB Dataport	USB 2.0 dataport (type B)	
CPU Debug Ashling/Lauterbach	20-pin connector for debug probes by Ashling and Lauterbach	1V8
CPU Debug Digilent	6-pin connector for Digilent debug probe	1V8

See the "[Debug](#)" section for detailed information on connecting the debugger.

### 2.2.8 SD-Card Slot

The ARC SDP Mainboard includes a slot for standard SD-cards. A jumper allows to enable / disable authentication support. See the "[SD-Card](#)" section for detailed information.

## 2.2.9 Miscellaneous Interfaces

Table 6    *Miscellaneous interfaces*

Name	Description	Voltage Level
Power Supply	DC input See the “ <a href="#">Power Supply</a> ” section for details.	12 V
OSC Out	SMB jack for FPGA main clock Frequency selected by DIP switches. See the “ <a href="#">FPGA Reference Clock</a> ” section for details.	3V3
JTAG In	JTAG chain input and output	3V3
JTAG Out	See the “ <a href="#">JTAG Programming Header</a> ” section for details.	

## 2.3 CPLD Board Supervisor

The ARC SDP Mainboard includes a CPLD (Coolrunner-II) that is used to implement the following main functions:

- Reset controller  
Controls the reset sequencing of the FPGA, ARC CPU Card, on-board integrated circuits, and hardware extensions connected to the extension interfaces
- Status monitor  
Monitors power supply, reset, FPGA temperature, and FPGA configuration status and indicates the status using LEDs (power, reset, FPGA done, and temperature alert)
- JTAG multiplexer  
Selects between JTAG sources (JTAG In / Out, USB-to-JTAG converter) and between JTAG targets (such as CPU debugging or FPGA programming)
- SPI interface for seven-segment displays  
Provides an SPI interface for the two on-board seven-segment displays
- Control for SPI Flash programming

The CPLD is located at the top left of the board, near the jumpers.

## 2.4 USB Dataport

The USB Dataport has a USB-B connector and can be connected to your PC using the USB cable included in the product package. A USB converter by FTDI (FT2232HL) converts one USB channel to a serial communication protocol (UART, I<sup>2</sup>C or SPI). The other channel is converted to JTAG.

The location of the USB Dataport connector on the ARC SDP Mainboard is shown in Figure 4. The location of the LEDs and jumpers that are related to the USB Dataport are shown as well.

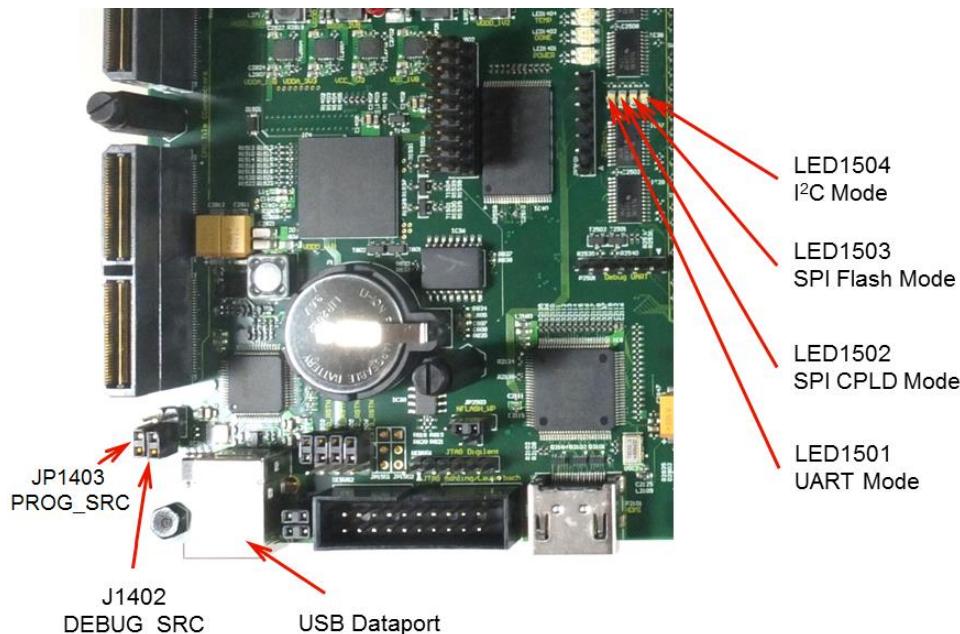


Figure 4 Location of the USB Dataport connector and the corresponding LEDs and jumpers

### 2.4.1 JTAG Debug Channel

For software debugging, the debugger tool can be connected to the ARC cores on the ARC CPU Card via the JTAG debug channel of the USB Dataport or via one of the debug cable connectors (CPU Debug Ashling / Lauterbach and CPU Debug Digilent). The jumpers JP1403/JP1402 select between these two modes. The jumper setting for connecting the debugger via the USB Dataport is shown in Figure 5 below.

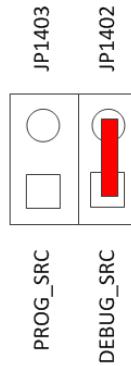


Figure 5 Jumper setting for connecting the debugger via the USB Dataport

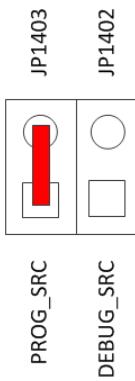


Figure 6 Jumper setting for connecting the debugger via debug cable connectors

## 2.4.2 Communication Channel

The communication channel is used by the `axs_comm` tool (see “[AXS Communicator Tool – `axs\_comm`](#)”) to access multiple on-board resources. It operates in four modes as described below:

- UART Mode

In UART mode the user can invoke a standard hyperterminal on the PC, which can be used as a console for software debugging. This is the default mode after a reset. LED1501 shines green when the USB Dataport is in UART mode.

- I<sup>2</sup>C Mode

The I<sup>2</sup>C mode is used to control and program the on-board real time clock, HDMI transmitter, and the EEPROM. LED1504 shines green when the USB Dataport is in I<sup>2</sup>C mode.

- SPI Flash Mode

The SPI Flash mode is used to initialize or read back the SPI flash application memory using the USB interface. LED1503 shines green when the `USB Dataport` is in SPI Flash mode.

- SPI CPLD Mode

The SPI CPLD mode is used to access registers in the CPLD device. LED1502 shines green when the `USB Dataport` is in SPI CPLD mode.

Switching between these four modes is automatically performed by the `axs_comm` tool.

---

## 2.5 JTAG Programming Header

The `JTAG In` and `JTAG Out` interfaces are connected to the CPLD device, which performs JTAG chaining of these multiple targets. These JTAG interfaces are automatically chained when the `JTAG Out` header is connected to another 10-pin JTAG header at which pin 2 is connected to a 3V3 level.

Optionally, `JTAG In` can be used to install firmware updates and is typically connected to a Xilinx programming cable. `JTAG Out` allows including another board (such as a HAPS system) in the JTAG chain. However, firmware updates are usually installed via the `USB_Dataport`. Only the USB cable included in the product package is required for this method. Additionally, the package also includes batch scripts and installation instructions.

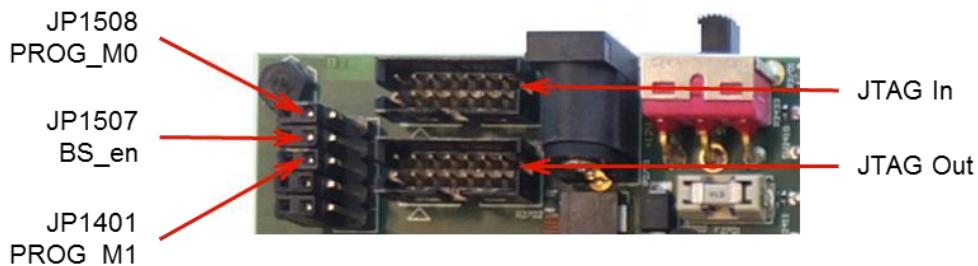


Figure 7 Location of the `JTAG In` and `JTAG Out` connectors and the corresponding jumpers

Figure 7 shows the location of the `JTAG In` and `JTAG Out` connectors and the related jumpers on the ARC SDP Mainboard. Figure 8 shows the pinout diagrams of the

connectors. Table 7 and Table 8 provide pin descriptions of the JTAG In and JTAG Out connectors. The connectors are compatible with Xilinx programming cables.

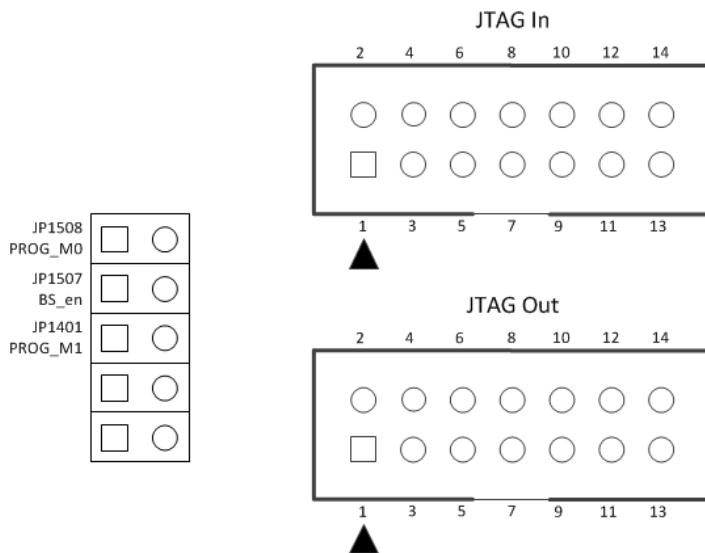


Figure 8 Pinout diagrams of the JTAG In and JTAG Out connectors



**Note** The jumpers JP1401, JP1507 and JP1508 shall be out for normal operation.

Table 7 Pin description of the JTAG In connector

Pin	Name	Description	Direction
1	GND	Ground supply pin	
2	VREF	3V3 reference voltage related to outputs to previous board in JTAG chain	Output
3	GND	Ground supply pin	
4	TMS	Test mode select	Input
5	GND	Ground supply pin	
6	TCK	Test clock	Input
7	GND	Ground supply pin	
8	TDO	Test data to previous board in JTAG chain	Output
9	GND	Ground supply pin	
10	TDI	Test data and commands from previous board in JTAG chain	Input
11	GND	Ground supply pin	
12		Not connected	

<b>Pin</b>	<b>Name</b>	<b>Description</b>	<b>Direction</b>
13	GND	Ground supply pin	
14		Not connected	

*Table 8 Pin description of the JTAG Out connector*

<b>Pin</b>	<b>Name</b>	<b>Description</b>	<b>Direction</b>
1	GND	Ground supply pin	
2	VREF	Reference voltage related to input from next board in JTAG chain	Input
3	GND	Ground supply pin	
4	TMS	Test Mode Select	Output
5	GND	Ground supply pin	
6	TCK	Test Clock	Output
7	GND	Ground supply pin	
8	TDO	Test Data read back from next board in JTAG chain	Input
9	GND	Ground supply pin	
10	TDI	Test data and commands to next board in JTAG chain	Output
11	GND	Ground supply pin	
12		Not connected	
13	GND	Ground supply pin	
14		Not connected	

## 2.6 FPGA Temperature and Fan Control

The temperature of the on-board FPGA is monitored by a MAX6670 device. A fan is mounted on top of the FPGA and automatically turned on when the FPGA temperature reaches a certain trigger level.

The LED TEMP changes from green to red when the FPGA temperature becomes too high.

- 75 °C, 167 °F → TEMP LED constant RED
- 90 °C, 194 °F → TEMP LED flashing RED

When this LED turns RED, the user should power down the board and investigate the root cause of this overheating. Figure 9 shows the location of the LED on the ARC SDP Mainboard.



Figure 9 Location of the TEMP LED

## 2.7 Clock Generation

The ARC SDP Mainboard internally generates all the required clocks. A high level overview of the ARC SDP Mainboard clock architecture is shown in Figure 10. The majority of the required clocks are generated by the Clock Generation Unit inside the FPGA.

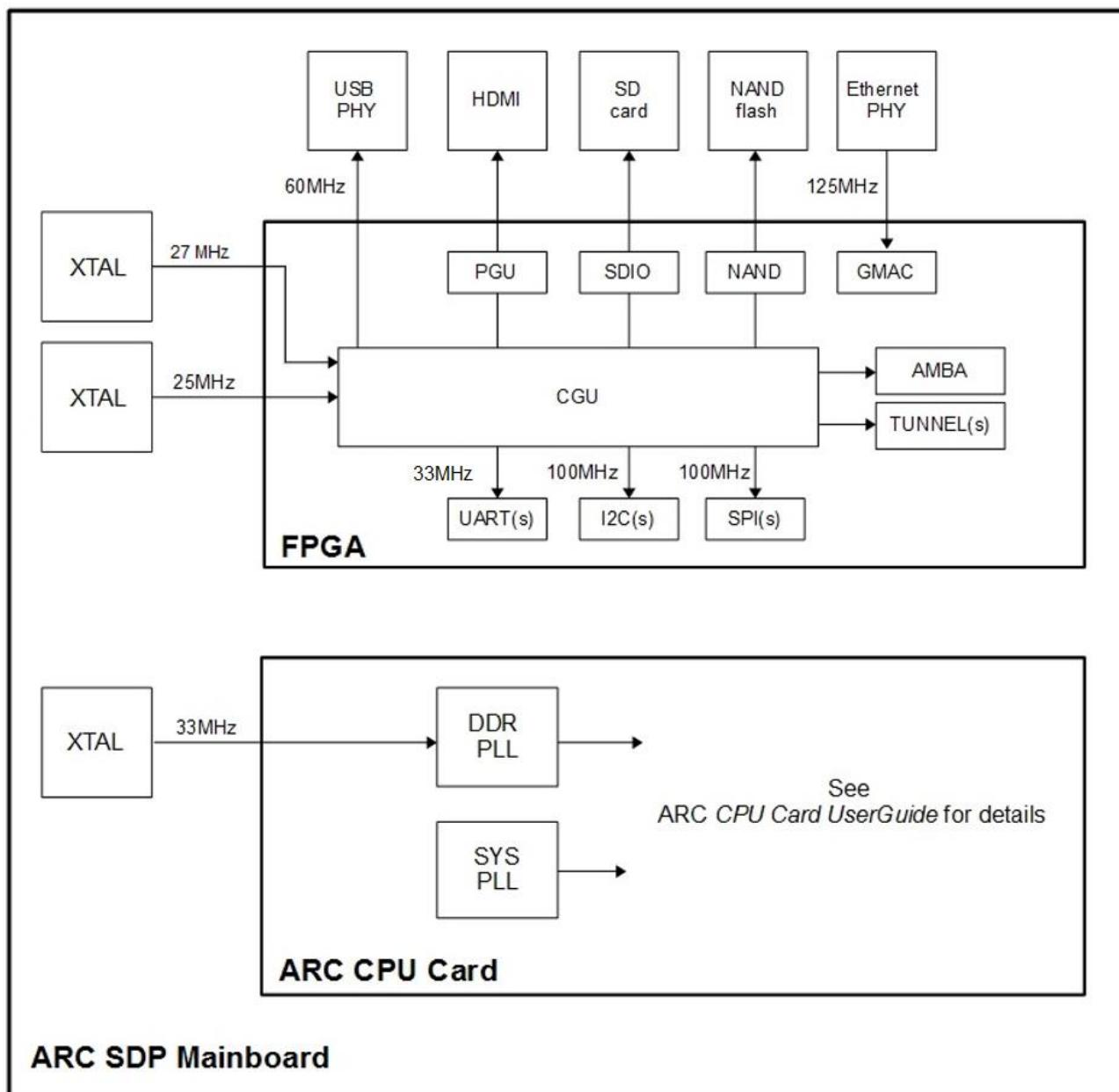


Figure 10 Mainboard clock architecture

## 2.7.1 FPGA Reference Clock

The ARC SDP Mainboard contains a 25 MHz and a 27 MHz FPGA input reference clock.

## 2.7.2 AMBA Clocks

The Clock Generation Unit inside the FPGA generates fixed clocks for the AMBA interconnect network. The frequencies for the AMBA interconnect network are listed in Table 9:

*Table 9 AMBA clocks*

Clock Name	Frequency (MHz)
APB clock	100
AXI clock	100

## 2.7.3 Tunnel Clocks

The Clock Generation Unit inside the FPGA generates programmable clocks for the two AXI tunnels. The default frequencies for the AXI tunnels (before and after loading the bootloader) are listed in Table 10 and Table 11.

See the “[Clock Generation Registers](#)” section for details on programming the AXI tunnel clocks for different frequencies.

*Table 10 AXI tunnel clocks (after reset)*

Clock Name	Frequency (MHz)	Remarks
Tunnel clock 0 (CPU Card)	25	Max. frequency 100MHz
Tunnel clock 1 (HAPS system)	25	Max. frequency 75MHz

*Table 11 AXI tunnel clocks (after pre-boot)*

Clock Name	Frequency (MHz)	Remarks
Tunnel clock 0 (CPU Card)	100	Max. frequency 100MHz
Tunnel clock 1 (HAPS system)	25	Max. frequency 75MHz

## 2.7.4 PGU Clock

The Clock Generation Unit inside the FPGA generates a programmable reference clock for the PGU. The PGU reference clock frequency is dependent on the target HDMI resolution. The default frequencies for the PGU reference clock (before loading the bootloader) is listed in Table 12.

See the “[Clock Generation Registers](#)” section for details on programming the PGU reference clock to different frequencies.

*Table 12 PGU reference clock*

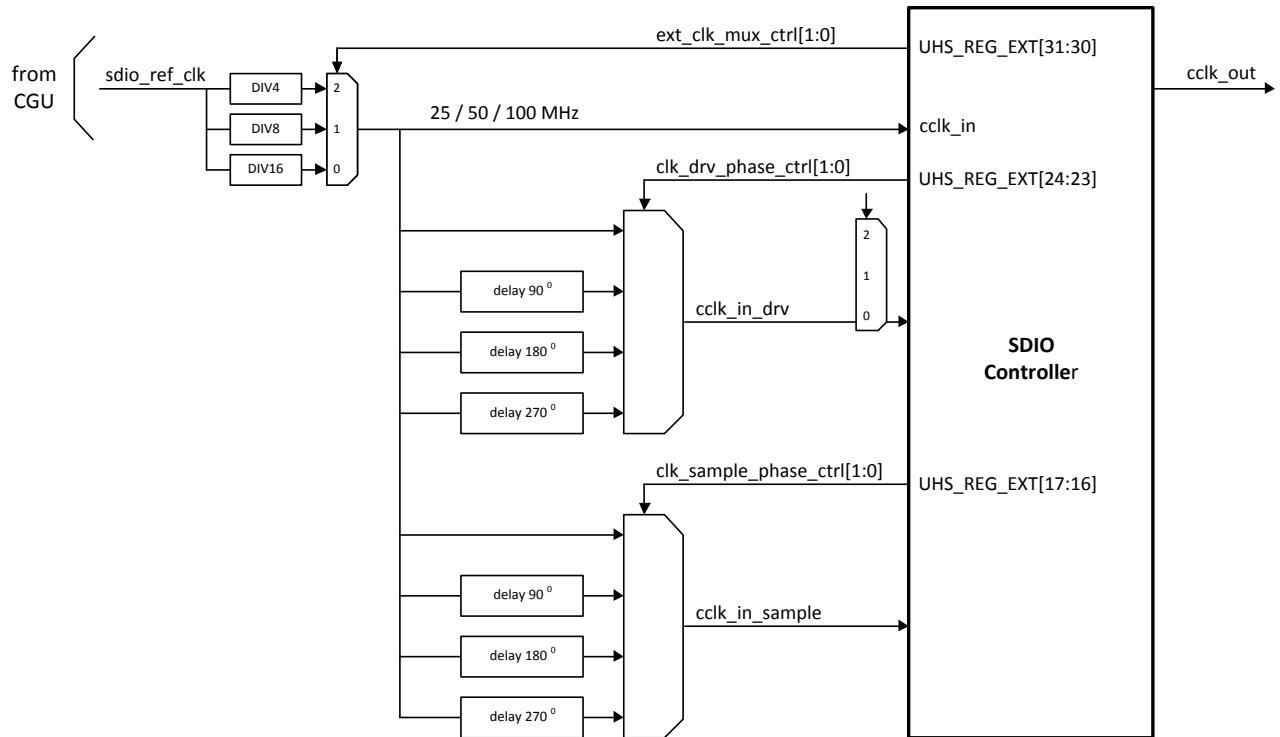
Clock Name	Frequency (MHz)	Remarks
PGU reference clock	74.25	Max. frequency 74.25MHz

## 2.7.5 Storage Clocks

The Clock Generation Unit inside the FPGA generates fixed reference clocks for the storage IP (i.e. SDIO, and NAND). The desired operating storage speed can be programmed in the peripheral. The frequencies for the storage connectivity reference clocks are listed in the table below:

*Table 13 Storage reference clocks*

Clock Name	Frequency (MHz)
NAND reference clock	25
SDIO reference clock	400



*Figure 11 Selecting SDIO card clock frequency via the SDIO controller*

## 2.7.6 Serial Connectivity Clocks

The Clock Generation Unit inside the FPGA generates fixed reference clocks for the serial connectivity peripherals (i.e. I<sup>2</sup>C, UART and SPI). The desired operating baudrate can be programmed in the peripheral. The frequencies for the serial connectivity reference clocks are listed in the table below:

*Table 14 Serial Connectivity reference clocks*

Clock Name	Frequency (MHz)
UART reference clock	33
I <sup>2</sup> C reference clock	100
SPI reference clock	50
SPI_FLASH reference clock	100

## 2.7.7 Audio Reference Clock

The Clock Generation Unit inside the FPGA generates two fixed audio reference clocks for I<sup>2</sup>S audio output to HDMI PHY. One of the two reference clocks can be selected by software using the `AUDIO_CLK_MUX_CTRL` register (see "[Control Registers](#)").

See the "[Clock Generation Registers](#)" section for details on dividing down the fixed audio reference clock to the desired I<sup>2</sup>S sclk and mclk frequency.

*Table 15 Audio Reference Clocks*

Clock Name	Frequency (MHz)
12.288 MHz audio reference clock	12.288
28.224 MHz audio reference clock	28.224

## 2.7.8 ARC CPU Card Clock

The ARC SDP Mainboard generates a fixed 33 MHz clock for the ARC CPU Card.

## 2.8 Ethernet

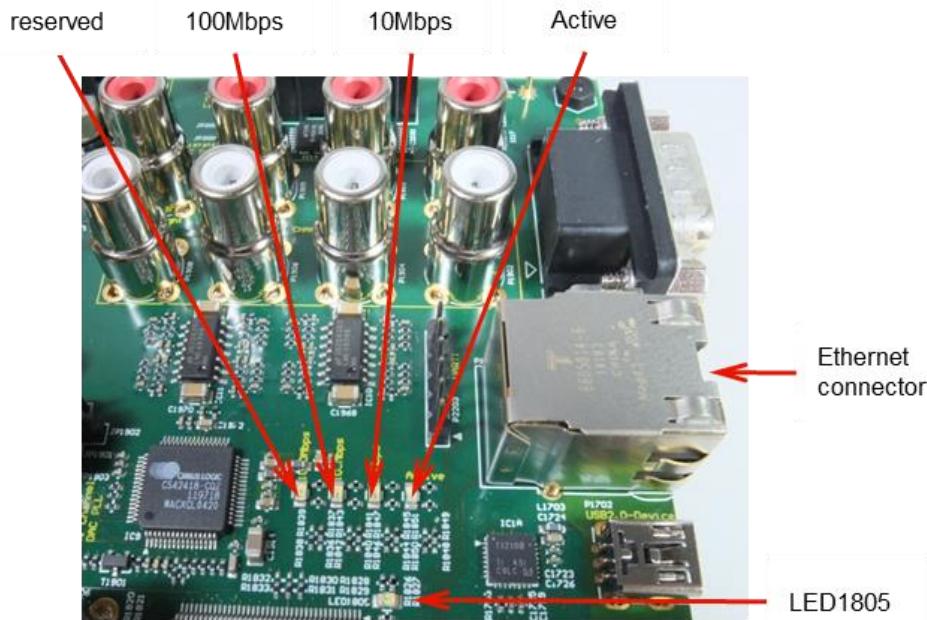
The ARC SDP Mainboard features a DP83865 Gigabit Ethernet Physical Layer from National Semiconductor and supports the 10/100 Mb/s Ethernet protocols. The PHY is connected to the FPGA using the RGMII interface of this chip. A standard RJ45 connector is used as external interface.

The LEDs related to the Ethernet PHY are listed in Table 16. Figure 12 shows the position of the Ethernet LEDs on the board.

See [7] for functional details of the Ethernet PHY. Refer to the release notes at the ARC SDP download webpage [12] for the driver availability information.

*Table 16 Ethernet LEDs*

LED Name (Mainboard)	LED Name (DP83865)	Description
Active	ACTIVITY_LED	Indicates the occurrence of either idle error or packet transfer
10Mbps	LINK10_LED	PHY established a good link at 10Mbps
100Mbps	LINK100_LED	PHY established a good link at 100Mbps
LED1805	DUPLEX_LED	PHY is in full duplex operation after the link is established



*Figure 12 Location of Ethernet connector and Ethernet LEDs*

## 2.9 USB

The ARC SDP Mainboard has a USB host interface, which is implemented using a TUSB1210 standalone USB Transceiver Chip [10] from Texas Instruments supporting the USB 2.0 and On-The-Go protocols. The PHY is connected to the FPGA using the ULPI interface. A standard USB A connector is used for the USB host.

The USB miniB connector on the board is reserved for future extensions.

Refer to the release notes at the ARC SDP download webpage [12] for the driver availability information.

---

## 2.10 HDMI Output

The ARC SDP Mainboard supports HDMI/DVI video output using the HDMI transceiver ADV7511 from Analog Devices [8]. The ADV7511 is a high speed High Definition Multimedia Interface (HDMI) transmitter that is capable of supporting an input data rate up to 165MHz (1080p @ 60Hz, UXGA @ 60Hz) and an output data rate up to 225MHz. The HDMI transmitter is controlled by a PGU (Pixel Graphics Unit) sending 3x8 bits RGB data. The external interface is a standard 19-pin HDMI Plug Type A.

The HDMI transceiver is controlled through its I<sup>2</sup>C interface, which is connected to the internal I<sup>2</sup>C bus. The I2C2 module of the peripheral subsystem and the communication channel of the USB Dataport are masters on this bus. Set the I<sup>2</sup>C address byte to 0xE5 for reading and to 0xE4 for writing.

Refer to the datasheet of the HDMI chip [9] for programming details.

Refer to the release notes at the ARC SDP download webpage [12] for the availability of a driver.

The SPDIF-IN audio interface of the HDMI transceiver is connected to the SPDIF-IN RCA connector. Furthermore, the 8-channel I<sup>2</sup>S master input of the HDMI transceiver is connected to an I2S device in the FPGA on the Mainboard. See the "[HDMI Audio Content](#)" sub-section below for details on the HDMI audio interfacing options.

Figure 13 shows the location of HDMI-related connectors on the Mainboard.



Figure 13 Location of HDMI-related connectors and jumpers on the Mainboard

## 2.10.1 HDMI Audio Content

The HDMI audio content can be provided by the SPDIF-IN RCA plug, or by the I<sup>2</sup>S master mapped in the FPGA on the Mainboard. The HDMI transceiver chip SPDIF-out is left unconnected.

### 2.10.1.1 Using HDMI I2S Audio Source Input

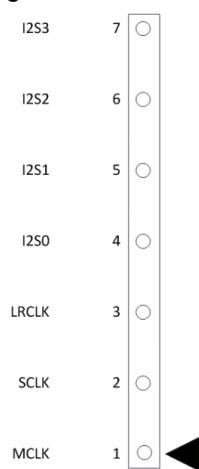
The I2S audio input of the HDMI transmitter is connected to a Multi-Channel I2S device in the PSS of the FPGA on the Mainboard. Table 17 provides the pin description of the HDMI 8-channel I<sup>2</sup>S slave input and Figure 14 shows the pinout diagram of the HDMI 8-channel I<sup>2</sup>S slave input. Figure 13 on page 35 shows the location of the HDMI connector and the HDMI 8-Channel I2S In connector on the Mainboard.

Changing the HDMI audio source also requires re-programming the HDMI transceiver chip.

Table 17 Pin Description of the HDMI 8-Channel PS Slave Input

Pin	Name	Description	Direction
1	MCLK	HDMI multi-channel oversampling clock	Input
2	SCLK	HDMI multi-channel bit clock	Input
3	LRCLK	Multi-channel word select (left/right clock)	Input
7-4	I2S3-0	Multi-channel data 3-0	Input

Figure 14 Pinout diagram of the HDMI 8-Channel I2S Slave Input (JP2101)



### 2.10.1.2 Using HDMI SPDIF Audio Input

The SPDIF-IN RCA plug is connected to the ARC CPU card as well as to the SPDIF-IN of the HDMI transceiver chip. The HDMI audio content is provided by this SPDIF-IN RCA plug. Changing the HDMI audio source also requires re-programming the HDMI transceiver chip.

## 2.11 Audio Support

Figure 15 shows the location of the available audio connectors on the board and Table 18 provides a description of the connectors.

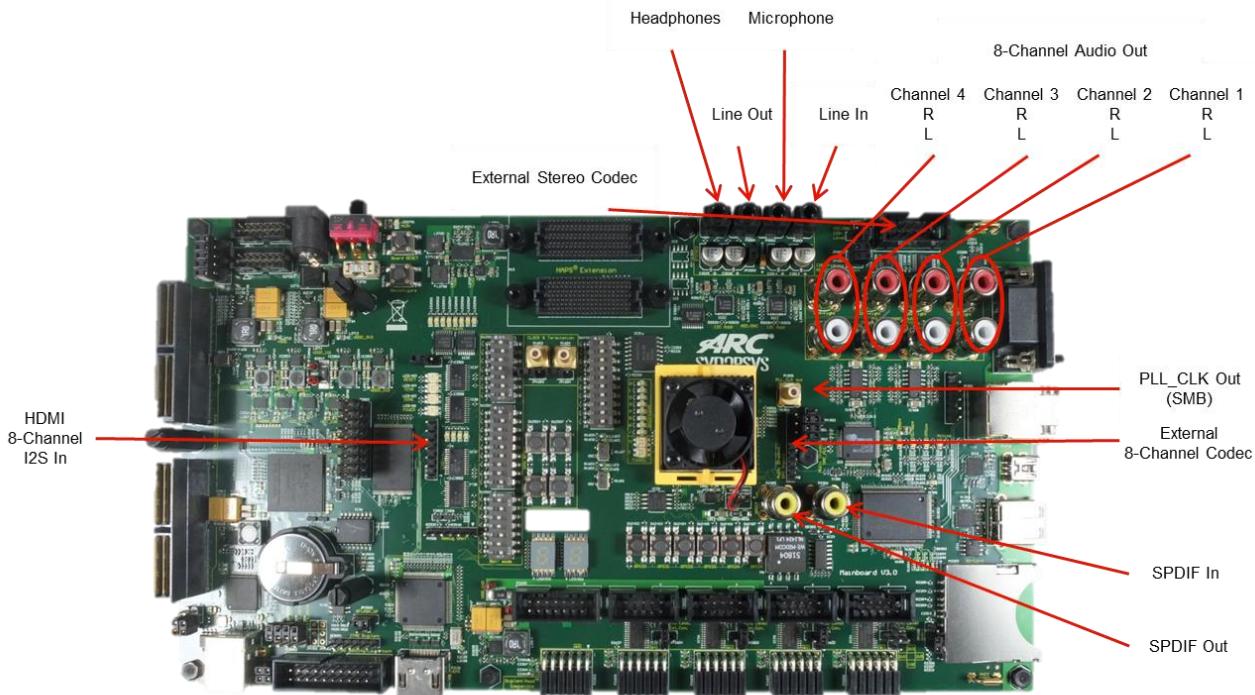


Figure 15 ARC SDP Mainboard audio interfaces

The ARC SDP Mainboard includes stereo and eight-channel audio codecs supporting the following analog audio interfaces:

- 7:1 multi-channel line-out (RCA Audio connectors)
- Stereo line out (3.5 mm stereo Jack)
- Headphone out (3.5 mm stereo Jack)
- Stereo line Input (3.5 mm stereo Jack)
- Microphone Input (3.5 mm stereo Jack)

Additionally, the following digital S/PDIF interfaces are available:

- S/PDIF input (RCA plug)
- S/PDIF output (RCA plug)

The digital sources and sinks for these interfaces are routed to the ARC CPU Card. Refer to the documentation of your ARC CPU Card to find out to what extent the CPU Card supports audio.

*Table 18 Audio interfaces on the Mainboard*

Name	Description	Voltage Level
Stereo Line Out	3.5 mm female TRS stereo jack for line out	
Headphones	3.5 mm female TRS stereo jack for headphones output	
Stereo Line In	3.5 mm female TRS stereo jack for line in	
Microphone	3.5 mm female TRS stereo jack for microphone input	
CH1-left	RCA jack for channel 1 left	
CH1-right	RCA jack for channel 1 right	
CH2-left	RCA jack for channel 2 left	
CH2-right	RCA jack for channel 2 right	
CH3-left	RCA jack for channel 3 left	
CH3-right	RCA jack for channel 3 right	
CH4-left	RCA jack for channel 4 left	
CH4-right	RCA jack for channel 4 right	
SPDIF In	RCA jack for SPDIF input	
SPDIF Out	RCA jack for SPDIF output	
PLL_CLK Out	SMB jack for audio PLL output	3V3
External Stereo Codec	10-pin connector for alternative stereo I <sup>2</sup> S master input and I <sup>2</sup> S master output; These ports allow using a custom audio codec instead of the on-board stereo codecs	Selectable by jumper: 3V3 2V5 1V8
External 8-Channel Codec	8-pin header for 8-channel I <sup>2</sup> S master output. This break-out header allows using a custom audio codec instead of the on-board eight-channel codec	3V3
HDMI 8-Channel I <sup>2</sup> S In	7-pin break-out header for observing 8-channel I <sup>2</sup> S data driven by MultiChannel-I <sup>2</sup> S device in Main Board FPGA design	

## 2.11.1 Analog Stereo Inputs/Outputs

The ARC SDP Mainboard features stereo audio jacks for Line Out, Headphones, Line In and Microphone. The stereo audio input and output signals are converted using two NXP UDA1380 stereo codecs, which by default provide the interface between the stereo I<sup>2</sup>S ports of the CPU Card and the Line Out, Headphones, Line In and Microphone interfaces of the Mainboard.

One of the codecs is used “DAC-only mode”, receives audio data from the stereo I<sup>2</sup>S master output of the CPU Card and drives the Line Out and Headphones outputs of the Mainboard.

The other codec is used in “ADC-only mode”, samples the analog signals on the Line In and Microphone inputs of the Mainboard and provides digital data to the stereo I<sup>2</sup>S master input of the CPU Card. Software selects between the Line In and the Microphone input by programming the audio DAC chip via the internal I<sup>2</sup>C bus.

The ADC part supports audio rates up to 55 kHz, and the DAC part supports up to 100 kHz. The resolution is 24 bits.

The stereo codecs are controlled through their I<sup>2</sup>C interfaces, which are connected to the I2C2 module of the peripheral subsystem via the internal I<sup>2</sup>C bus. To access the control registers of the stereo codecs set the I<sup>2</sup>C address byte to 0x31 (Line In) or to 0x35 (Line Out) for reading and to 0x30 (Line In) or to 0x34 (Line Out) for writing.

The digital I<sup>2</sup>S interfaces of the stereo codecs are routed to the ARC CPU Card connector.

The jumper settings for using the on-board stereo codecs are shown in Figure 16. They correspond to the factory default settings. Figure 17 shows the location of the analog stereo inputs and outputs and of the corresponding jumpers on the Mainboard.

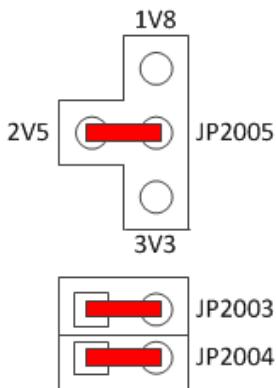


Figure 16 Jumper settings for using the on-board stereo codecs (default)

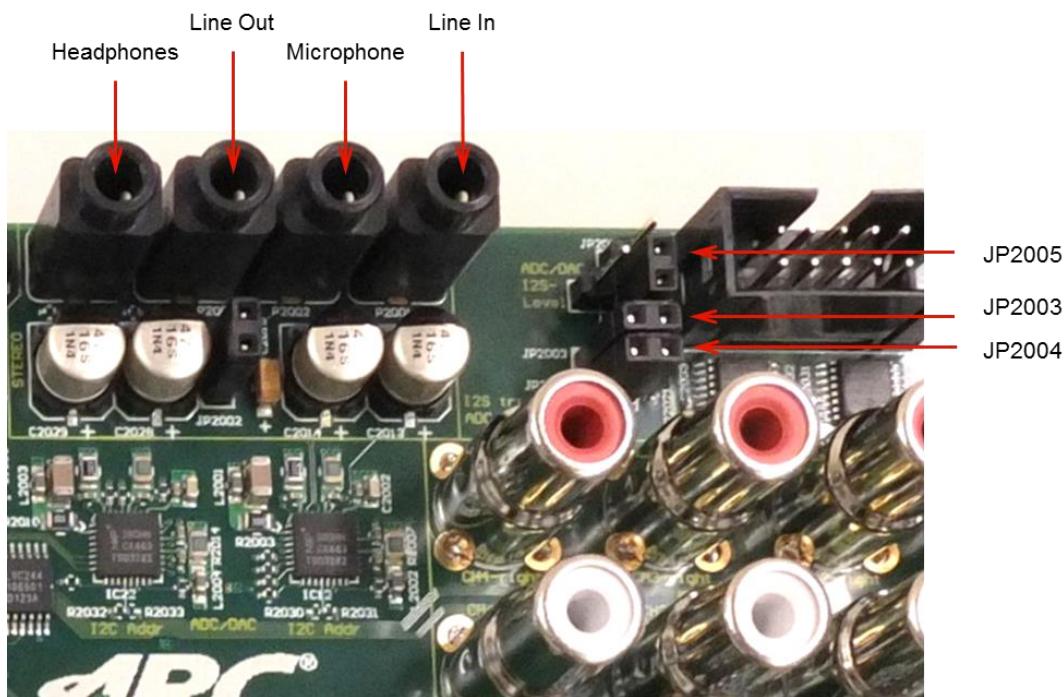


Figure 17 Location of the stereo inputs and outputs and of the corresponding jumpers



**Note** For advanced use cases, the stereo I<sup>2</sup>S ports of the CPU card can alternatively be connected to custom codecs using the External Stereo Codec connector; see “Using a Custom Stereo Codec”.

## 2.11.2 Analog 8-Channel Audio Output

The analog eight-channel audio output is provided through the Cirrus Logic CS42428 or CS42418 audio codec, which supports audio rates up to 192 kHz. The codec has an integrated PLL and supports four multi-bit analog-to-digital and eight multi-bit digital-to-analog delta-sigma converters. The ADCs of this device are not used, however. The outputs of the eight-channel DAC are routed through individual unity-gain low pass filters, which are each

AC-coupled to an RCA audio jack. The low pass filters are implemented using the Texas Instruments LME49740 operational amplifiers.

The eight-channel codec is controlled through its I<sup>2</sup>C interface, which is connected to the I<sup>2</sup>C module of the peripheral subsystem via the internal I<sup>2</sup>C bus. Set the I<sup>2</sup>C address byte to 0x9D for reading the control registers of the eight-channel codec and set the address byte to 0x9C for writing the registers.

The digital I<sup>2</sup>S interface of the eight-channel codec is routed to the 8-channel I<sup>2</sup>S master output of the ARC CPU Card. Figure 18 shows the location of the interface and of the break-out connector JP1904 on the Mainboard.

For advanced use cases, the 8-channel I<sup>2</sup>S master output of the CPU Card can alternatively be connected to an external custom multi-channel audio D/A converter. Refer to the “[Using a Custom Multi-Channel D/A Converter](#)” section for details.

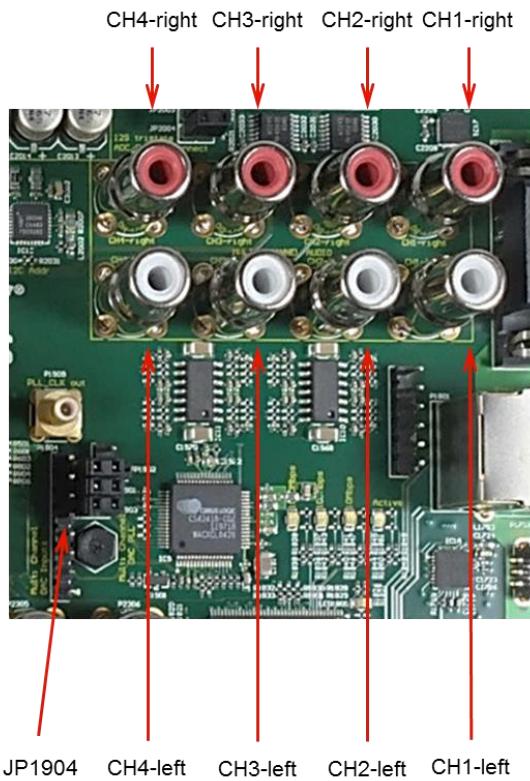


Figure 18 Location of the 8-Channel Output and the corresponding jumpers

### 2.11.3 S/PDIF Inputs and Outputs

The ARC SDP Mainboard features two RCA connectors providing an S/PDIF input and an S/PDIF output. These connectors are routed to the S/PDIF interfaces of the CPU Card. Figure 19 shows the location of the S/PDIF connectors on the Mainboard.



Figure 19 Location of the S/PDIF connectors

#### 2.11.3.1 S/PDIF Input

The ARC SDP Mainboard routes the S/PDIF input signal from the RCA jack SPDIF In to the S/PDIF input of the CPU Card, which is located at the CPU Card Connector. Furthermore, this SPDIF In signal is also routed to the HDMI Transceiver.

#### 2.11.3.2 S/PDIF Out

The S/PDIF output of the CPU Card, which is located at the CPU Card Connector, is connected to the SPDIF Out RCA output of the ARC SDP Mainboard. The S/PDIF output signal is coupled to the RCA jack using an audio transformer for robustness.

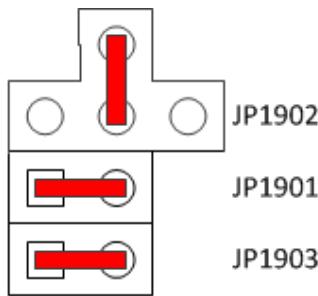
---

### 2.11.4 Audio PLL

The Cirrus CS42428 Multi-Channel Audio Codec that is used for the analog 8-channel audio output includes a programmable audio PLL.

Figure 20 shows the jumper settings for the audio PLL. Other jumper settings are reserved for future extensions.

The low-jitter PLL follows the PLL reference clock provided by the ARC CPU Card. The PLL clock output (RMCK) and its corresponding lock signal are routed back to the CPU Card. The PLL clock output of the CS42428 is also routed to the SMB clock connector `PLL_CLK_Out`.



PLL follows a reference clock provided by the ARC CPU Card

Figure 20 Audio PLL jumper settings

The location of these jumpers and of the `PLL_CLK Out` connector is shown in Figure 21.

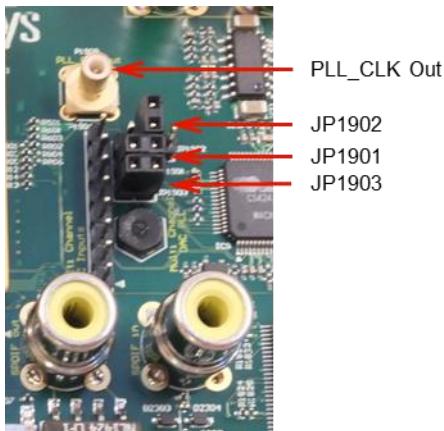


Figure 21 Location of the `PLL_CLK Out` connector and of the corresponding jumpers

## 2.11.5 Advanced Audio Use Cases

### 2.11.5.1 Using a Custom Stereo Codec

The ARC SDP Mainboard supports using custom audio codecs instead of the on-board codecs. The external codecs should have I<sup>2</sup>S slave interfaces, which operate at 1V8, 2V5 or 3V3 and should be connected to the External Stereo Codec I<sup>2</sup>S In / Out connector. Figure 22 shows the jumper settings for using an external custom stereo codec and Figure 23 shows the location of the External Stereo Codec I<sup>2</sup>S In / Out connector and the corresponding jumpers on the Mainboard.

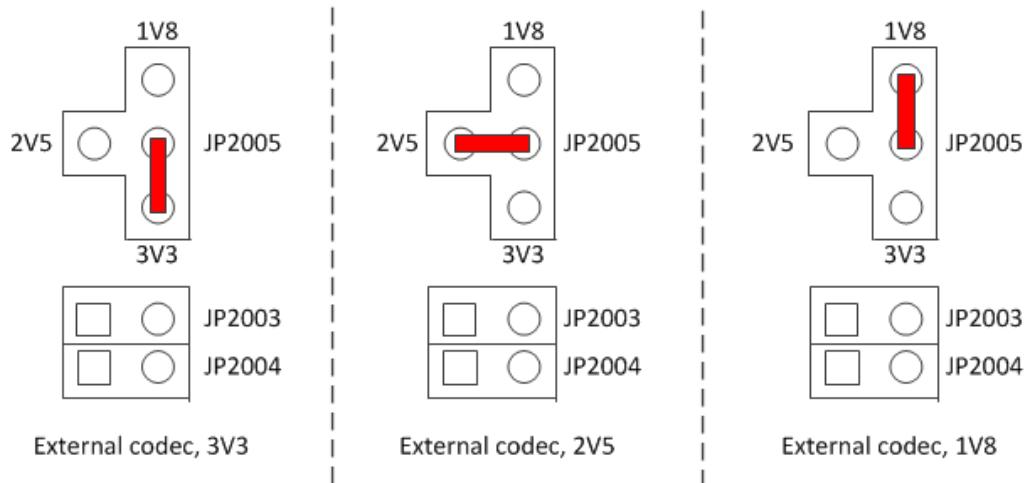


Figure 22 Jumper settings for using an external custom stereo codec

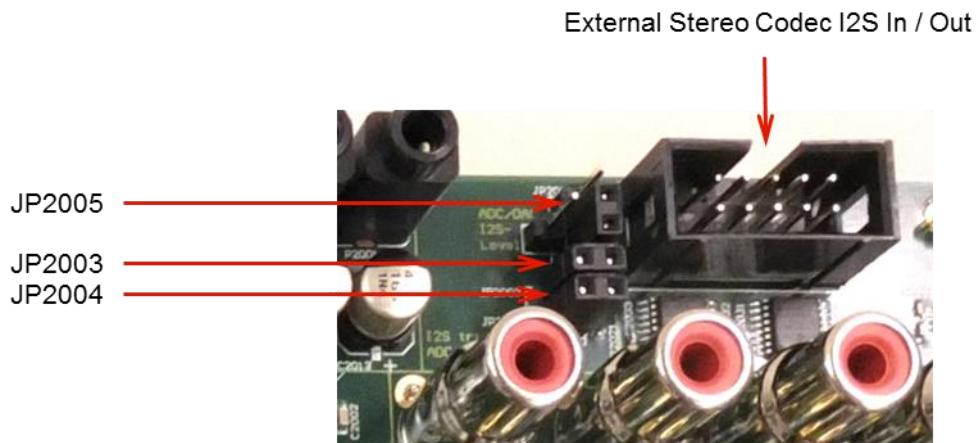


Figure 23 Location of the External Stereo Codec I2S In / Out connector and the corresponding jumpers

A pin description of the External Stereo Codec I2S In / Out connector is provided in Table 19, and the pinout diagram of the connector is shown in Figure 24.

Table 19 Pin description of the external stereo codec I2S master input / output

Pin	Name	Description	Direction
1	SCLK_IN	Stereo input bit clock	Output
2	MCLK_IN	Stereo input oversampling clock	Output
3	WS_IN	Stereo input word select left/right clock)	Output
4	SDI_IN	Stereo input serial data	Input
5	SCLK_OUT	Stereo Output bit clock	Output
6	MLCK_OUT	Stereo output oversampling clock	Output
7	SDO_OUT	Stereo output serial data	Output

Pin	Name	Description	Direction
8	n.c.	Not connected	
9	WS_OUT	Stereo output word select (left/right clock)	Output
10	RST_N	Reset, active low	Output

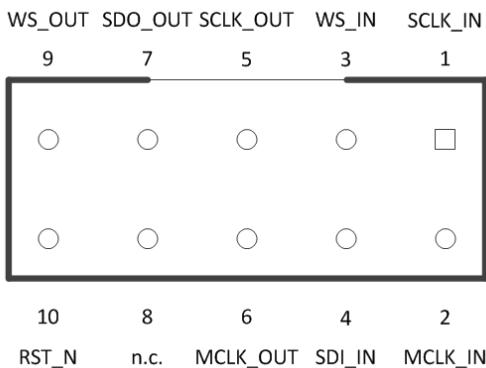


Figure 24 Pinout diagram of the external stereo codec I<sup>2</sup>S master input / output

## 2.11.5.2

### 2.11.5.3 Using a Custom Multi-Channel D/A Converter

The on-board eight-channel codec can be disabled, and a custom codec can be used instead. The ARC SDP Mainboard supports external codecs that have I<sup>2</sup>S slave interfaces for up to eight audio channels, which operate at 3V3.

An external eight-channel codec can be connected to the External 8-Channel I<sup>2</sup>S Out connector on header JP1904. The location of the connector on the Mainboard is shown in Figure 25. The pinout diagram of the connector is shown in Figure 26 and the pin description is provided in Table 20.

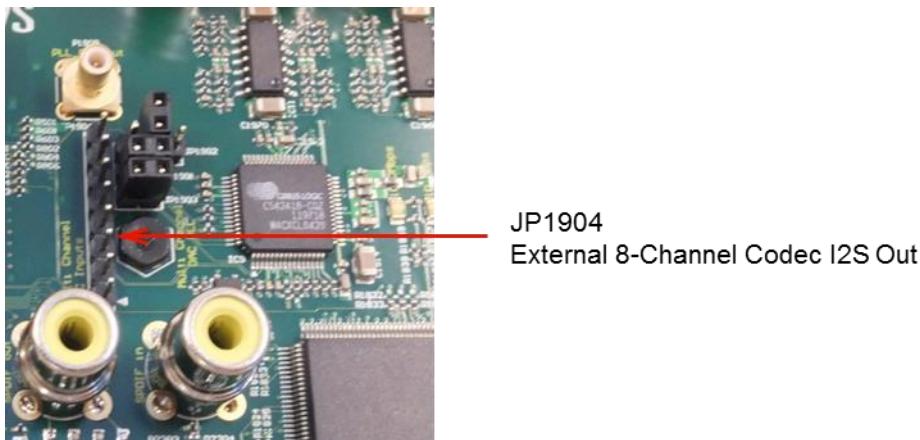


Figure 25 Location of the External 8-Channel Codec I2S Out connector

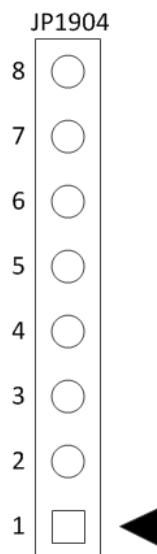


Figure 26 Pinout diagram of the External 8-Channel Codec I2S Out connector

Table 20 Pin description of the External 8-Channel Codec I2S Out connector

Pin	Name	Description	Direction
1	MCLK	Multi-channel oversampling clock	Output
2	SCLK	Multi-channel bit clock	Output
3	LRCLK	Multi-channel word select (left/right clock)	Output
4	SDO_0	Multi-channel data 0	Output
5	SDO_1	Multi-channel data 1	Output
6	SDO_2	Multi-channel data 2	Output

Pin	Name	Description	Direction
7	SDO_3	Multi-channel data 3	Output
8	RST_N	Multi-channel reset, active low	Output

## 2.12 SD-Card

The ARC SDP Mainboard features an SD-card interface supporting the following standard size cards (32 mm × 24 mm × 2.1 mm):

- SD (SDSC)
- SDIO (Secure Digital Input Output)
- SDHC (The Secure Digital High Capacity, capacities up to 32 GB)
- SDXC (The Secure Digital Extended Capacity, supports cards up to 2 TB)

To ensure stable SD card operation, it is best to follow these settings (pre-programmed by the bootloader):

- sdio card clock is 25 MHz
- use hold register
- sample clock delay is 180 degrees
- drive clock delay is 90 degrees

The above mentioned settings operate the SD card in SDR12 speed mode. Higher speed modes (for example SDR25) requires re-tuning of the sample and/or drive clock delays.

The SD card interface supports authentication by an on-board Atmel authentication chip ATSHA204, which can optionally be connected to pin 6 of the SD-card holder, and is normally connected to the ground.

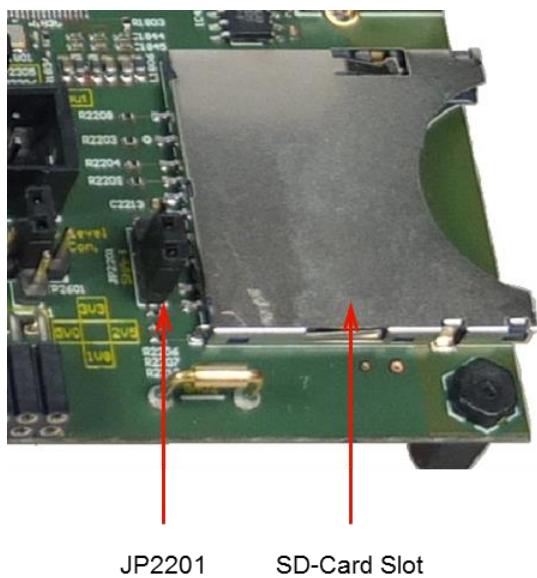
*Figure 27 shows the jumper settings for using the SD-card with or without authentication and*

*Figure 28 shows the location of the SD-card slot and the corresponding jumper on the Mainboard.*

Figure 27 Jumper settings for SD-card with and without authentication



Figure 28 Location of SD-card slot and corresponding jumper



## 2.13 Real Time Clock

The ARC SDP Mainboard includes an extremely accurate real time clock (RTC) chip DS3231SN from Maxim with battery back-up that keeps track of the actual time and day. The RTC is controlled using its I<sup>2</sup>C interface, which is connected to the I<sup>2</sup>C2 module of the peripheral subsystem via the internal I<sup>2</sup>C bus. Additionally, the RTC can be accessed via the communication channel of the USB Dataport, which can be set to operate as another I<sup>2</sup>C master on this bus.

Set the I<sup>2</sup>C address byte to 0xD1 for reading and to 0xD0 for writing the control registers of the RTC chip.

A real time clock is particularly useful for Linux (and Android) applications to ensure correct day and time for file timestamps. The real time clock either uses the 24-hour format or the 12-hour format with an AM/PM indicator.

“[Appendix C](#)” describes batch files for setting and reading the time and day from your PC via the USB Dataport.

Refer to the release notes at the ARC SDP download webpage [12] for the availability of RTC application examples suitable for your ARC CPU Card.

## 2.14 Internal I<sup>2</sup>C Bus

The ARC SDP Mainboard uses an internal I<sup>2</sup>C bus to control the following on-board devices:

- HDMI transmitter
- Eight-Channel audio codec
- Stereo codec for line in and microphone
- Stereo codec for line out and headphones
- Real time clock
- EEPROM

The I2C2 module of the peripheral subsystem is typically used as the master on this bus. Additionally, the communication channel of the USB Dataport can be set by the `axs_comm` tool to operate as another master on the bus. Typically, the I2C2 peripheral is used for settings that are modified by software at run-time. The I<sup>2</sup>C mode of the USB Dataport is only used to initialize the RTC or to initiate the I<sup>2</sup>C EEPROM. Table 21 lists the I<sup>2</sup>C slave addresses.

*Table 21 Internal I<sup>2</sup>C bus slave addresses*

<b>Device Name</b>	<b>Description</b>	<b>7-bit I<sup>2</sup>C Address</b>	<b>I<sup>2</sup>C Address Byte</b>	
			<b>Write</b>	<b>Read</b>
ADV7511	HDMI Transmitter	1110010 = 0x72	0xE4	0xE5
UDA1380	Line In Stereo Codec	0011000 = 0x18	0x30	0x31
UDA1380	Line Out Stereo Codec	0011010 = 0x1A	0x34	0x35
CS42428	8-channel Codec	1001110 = 0x4E	0x9C	0x9D
DS3231SN	RTC	1101000 = 0x68	0xD0	0xD1
PCF8594	I <sup>2</sup> C EEPROM <sup>1)</sup>	1010100 = 0x54	0xA8	0xA9
		1010101 = 0x55	0xAA	0xAB

1) This device has separate I<sup>2</sup>C addresses for the lower and the upper half of the memory.

## 2.15 I<sup>2</sup>C Interfaces

The ARC SDP Mainboard includes an I<sup>2</sup>C peripheral (I2C0) for external serial communication, and can be programmed to operate in master or slave mode supporting standard and fast I<sup>2</sup>C mode up to 400 kHz.

A second peripheral (namely I2C2) is available and meant to be used as a master on the internal I<sup>2</sup>C bus of the Mainboard. Additionally, the [USB Dataport](#) can be programmed to provide access to the internal I<sup>2</sup>C bus. See “[Internal I<sup>2</sup>C Bus](#)” and “[USB Dataport](#)” for details.

### 2.15.1 Routing Options for External I<sup>2</sup>C Interfaces

I2C0 can be routed to the Pmod4 interface. Routing is controlled by software using the `PMOD_MUX_CTRL` Register as described in Table 22.

*Table 22 I<sup>2</sup>C Interface selection*

Peripheral	PMOD_MUX_CTRL Register		Selected Connector
	PM4[0]	PM4[2]	
I2C0	0	0	Not Connected to Pmod4 (default after reset)
	1	1	Pmod4 8-pin Pmod Interface
	1	0	Pmod4 upper row only. 4-pin subset of Pmod specification
	0	1	Pmod4 lower row only. 4-pin subset of Pmod specification
I2C2	Ignored	Ignored	Internal I <sup>2</sup> C bus

### 2.15.2 Using the Pmod4 Interface in I<sup>2</sup>C Mode

The I2C0 interface can be routed to the extension connector Pmod4 by programming the `PMOD_MUX_CTRL` register. Figure 29 shows the location of the Pmod4 connector on the ARC SDP Mainboard.

Figure 29 Location of the Pmod4 Connector Supporting I<sup>2</sup>C Mode

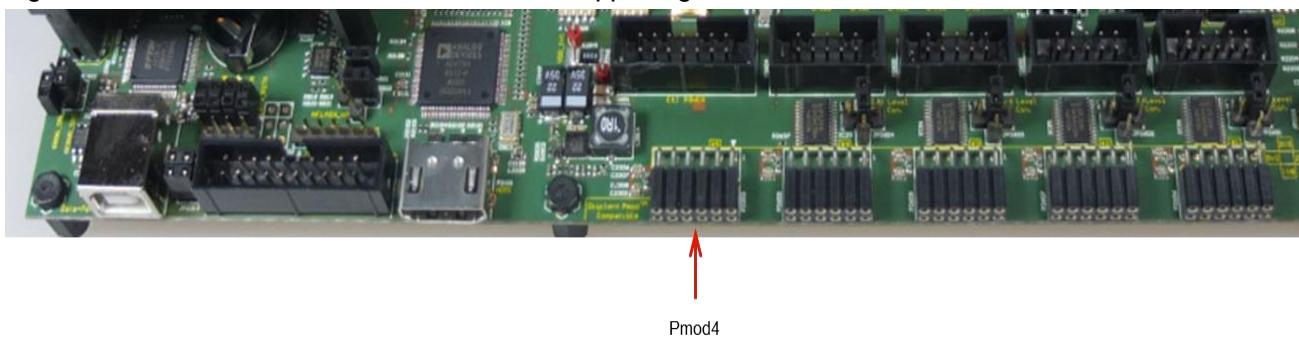
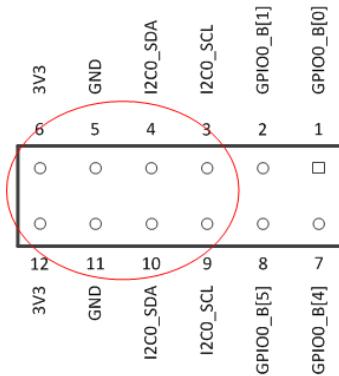


Figure 30 and Figure 31 show the pinout diagrams of the Pmod4 connector and the Extension0 connector for the following three cases:

- I2C0 routed to Pmod4 (PM4[0]=1, PM4[2]=1)
- I2C0 routed to Pmod4 upper row only (PM4[0]=1, PM4[2]=0)
- I2C0 routed to Pmod4 lower row only (PM4[0]=0, PM4[2]=1)



PM4[0] = 1, PM4[2] = 1

Figure 30 Pinout diagram of the Pmod4 connector (standard 8-pin interface)

Figure 30 shows the recommended configuration, which provides the standard 8-pin Pmod I<sup>2</sup>C interface. The section of the Pmod4 connector that is marked by the red ellipse shall be used to directly plug in peripheral modules having an 8-pin Pmod I<sup>2</sup>C interface. The same section of the Pmod4 connector can be used to connect up to two 4-wire cables for daisy-chaining multiple I<sup>2</sup>C peripheral modules.

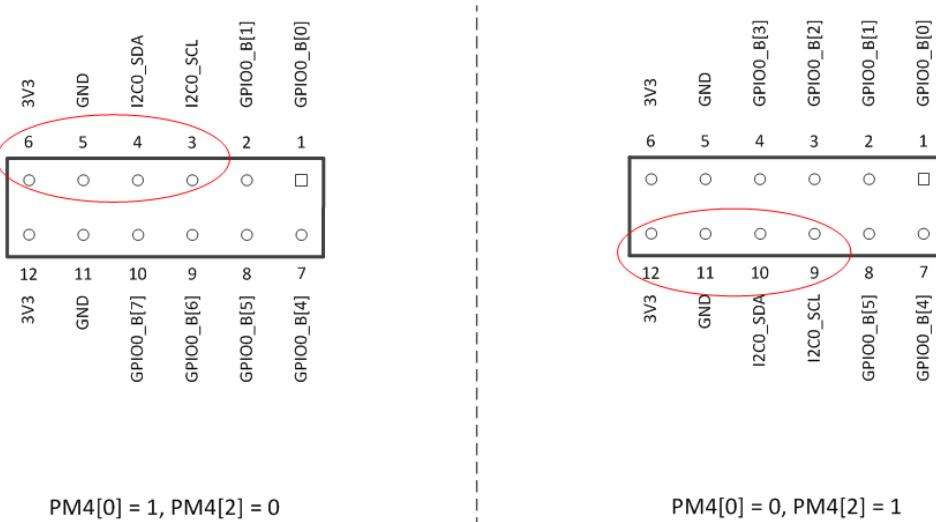


Figure 31 Pinout diagram of the Pmod4 connector (4-pin subsets)

The pinouts shown in Figure 31 are 4-pin subsets of the Pmod I<sup>2</sup>C Interface, which can be used to keep additional GPIO pins available. The section of the Pmod4 connector that is marked by the red ellipse shall be used to connect an I<sup>2</sup>C Pmod module or an I<sup>2</sup>C daisy chain via a 4-wire cable.



**Note** The I<sup>2</sup>C clock and data signals (`I2C0_SCL` and `I2C_SDA`) and all GPIO signals at the Pmod4 connector are pulled-up internally.

## 2.16 UART (RS232) Interfaces

The ARC SDP Mainboard supports multiple UART interfaces.

One RS232 port can be routed to a DB9 connector, which is wired as a host (DCE) device, or alternatively to the Pmod connectors Pmod0 or Pmod2.

A second UART port is can be routed to a dedicated six-pin header or alternatively to the Pmod connector Pmod0. The six-pin header supports a standard FTDI USB to TTL Serial Cable.

A third UART is connected to a USB-to-UART converter chip and is thus accessible through the [USB Dataport](#). See the “[USB Dataport](#)” section for more information.

These three UARTs support an automatic hardware flow control using RTS/CTS signals.

The Pmod connectors support UARTs compliant to Pmod Interface Type 4 and Type 4a as defined in the Pmod specification [3].

## 2.16.1 Routing Options for External UART Interfaces

By default, the UART peripherals UART0 and UART1 are routed to the UART0 (DB9) connector and to the 6-pin UART1 connector. Alternatively, the UART0 peripheral can be routed to the connectors Pmod0 / Extension0 or Pmod2 / Extension2. The UART1 peripheral can be optionally routed to the Pmod0 / Extension0 connectors. Routing is controlled by software using the PMOD\_MUX\_CTRL Register as described in Table 23.

Table 23 *UART Interface selection*

Peripheral	PMOD_MUX_CTRL Register			Selected Connector
	PM0[0]	PM0[2]	PM2[2]	
UART0	0	Ignored	0	UART0 DB9 connector (default after reset)
	1	Ignored	0 or 1	Pmod0 upper row Extension0 (pins 1,2,4,6)
	0	Ignored	1	Pmod2 lower row Extension2 (pins 3,5,7,9)
UART1	Ignored	0	Ignored	6-pin UART1 connector (default after reset)
	Ignored	1	Ignored	Pmod0 lower row Extension0 (pins 3,5,7,9)

## 2.16.2 Default UART Interfaces

Figure 32 shows the location of the UART0 (DB9) and 6-pin UART1 connectors on the ARC SDP Mainboard. Pin descriptions of these connectors are provided in Table 24 and

Table 25, and the pinout diagrams of the connectors are shown in Figure 33 and Figure 34.



**Note** If UART0 or UART1 is re-routed to Pmod0 / Extension0 or Pmod2 / Extension2, then the corresponding default connector UART0 (DB9) or 6-pin UART1 becomes inactive.

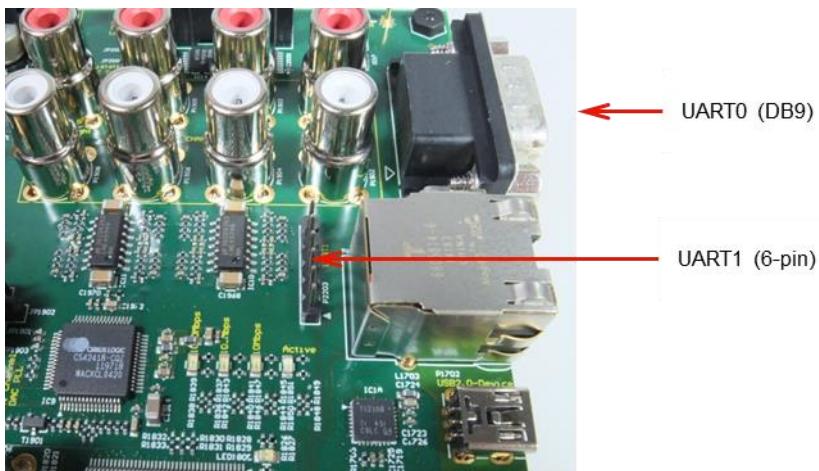


Figure 32 Location of the UART0 and UART1 connectors

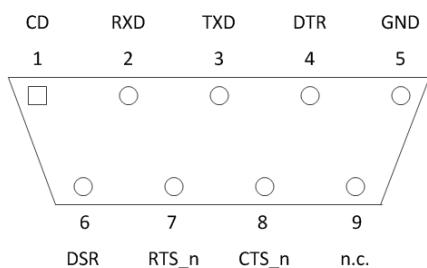
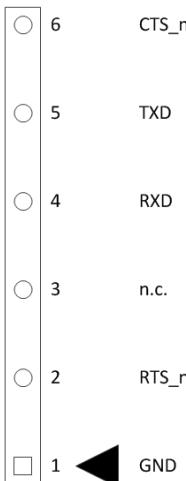


Figure 33 Pinout diagram of the UART0 DB9 connector

*Table 24 Pin description of the UART0 (DB9) connector*

<b>Pin</b>	<b>Name</b>	<b>Description</b>	<b>Direction</b>
1	CD	Carrier detect; DTR, DSR and CD are looped back	Input
2	RXD	Receive data Data from external peripheral to on-board host	Input
3	TXD	Transmit data Data from external peripheral to on-board host	Output
4	DTR	Data terminal ready; DTR, DSR and CD are looped back	Output
5	GND	Ground supply pin	
6	DSR	Data set ready; DTR, DSR and CD are looped back	Input
7	RTS_n	Ready to send (active low) Handshake signal from on-board host to external peripheral	Output
8	CTS_n	Clear to send (active low) Handshake signal from external peripheral to on-board host. Pulled-down internally to support 2-wire UART protocols.	Input
9	n.c.	Not connected	



*Figure 34 Pinout diagram of the UART1 connector*

*Table 25 Pin description of the 6-pin UART1 connector*

Pin	Name	Description	Direction
1	GND	Ground supply pin	
2	RTS_n	Ready to send (active low) Handshake signal from on-board host to external peripheral	Output
3	n.c.	Not connected	
4	RXD	Receive data Data from external peripheral to on-board host	Input
5	TXD	Transmit data Data from external peripheral to on-board host	Output
6	CTS_n	Clear to send (active low) Handshake signal from external peripheral to on-board host. Pulled-down internally to support 2-wire UART protocols.	Input



**Note** The pinout of the 6-pin UART connector shown in Figure 34 is compatible with the FTDI USB to TTL serial cable.

## 2.16.3 Using Extension Interfaces in UART Mode

As listed in Table 23 above the UART interfaces can be re-routed to the extension connectors Pmod0 / Extension0 and Pmod2 / Extension2 by programming the PMOD\_MUX\_CTRL register.

 **Note** If UART0 or UART1 is re-routed to Pmod0 / Extension0 or Pmod2 / Extension2, then the corresponding default connector UART0 (DB9) or 6-pin UART1 becomes inactive.

 **Note** Pmod0 and Extension0 share the FPGA outputs and are thus active simultaneously. Only a single UART should be connected to either Pmod0 or Extension0.

The same holds for Pmod2 and Extension2.

Figure 35 shows the location of the Pmod0, Pmod2, Extension0 and Extension2 connectors supporting UART mode. The jumpers for setting the voltage levels and the reset polarities at Extension0 and Extension2 are shown as well.

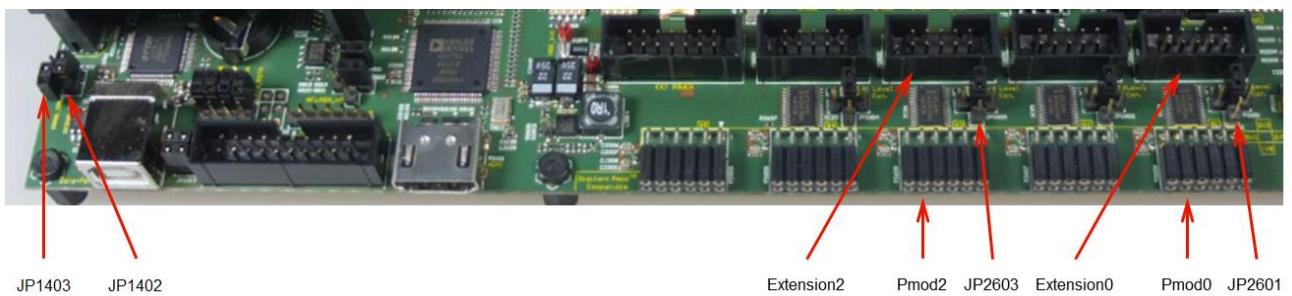


Figure 35 Location of Pmod and Extension connectors supporting UART mode

### 2.16.3.1 UART Interfaces at the Pmod0 and Extension0 Connectors

Figure 36 and Figure 37 show the pinout diagrams of the Pmod0 connector and the Extension0 connector for the following three cases:

- UART0 re-routed to Pmod0 / Extension0 (PM0[0]=1, PM0[2]=0)
- UART1 re-routed to Pmod0 / Extension0 (PM0[0]=0, PM0[2]=1)
- UART0 and UART1 re-routed to Pmod0 / Extension0 (PM0[0]=1, PM0[2]=1)

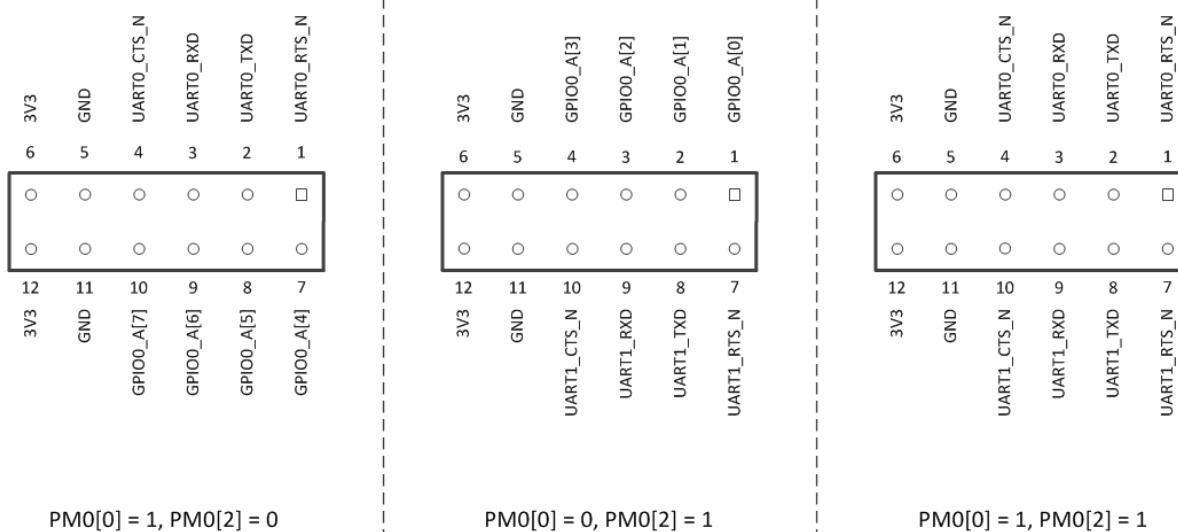


Figure 36 Pinout diagrams of the Pmod0 connector in UART mode (UART0 / UART1)

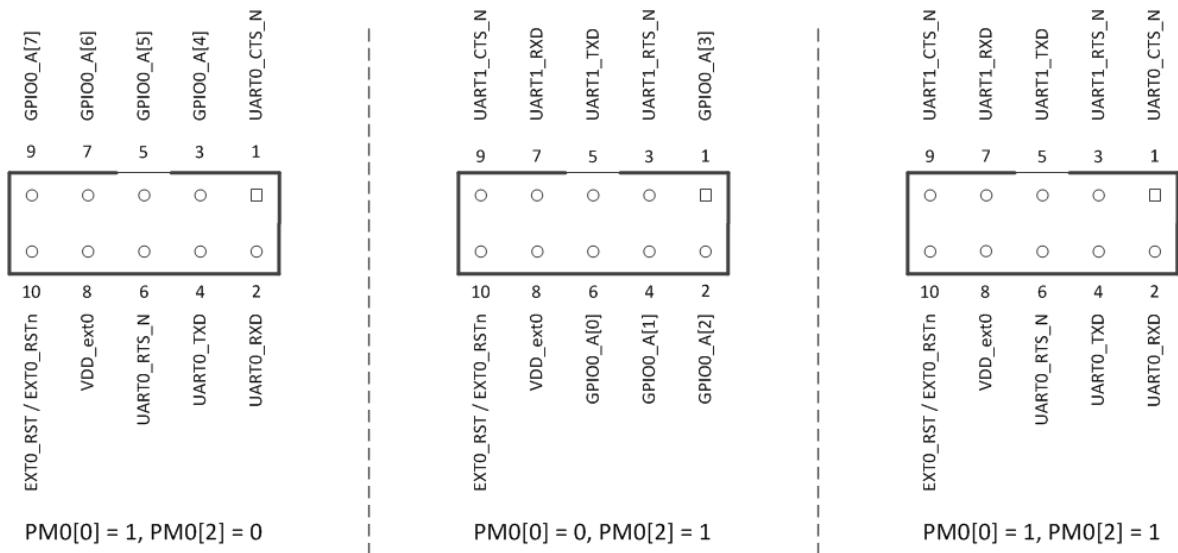


Figure 37 Pinout diagrams of the Extension0 connector in UART mode (IART0 / IART1)

The reset signal on pin 10 of Extension0 is active high when the jumper JP1503 is in place and active low when this jumper is removed.

The voltage level at Extension0 is selected by jumper JP2601 as shown in Figure 38.

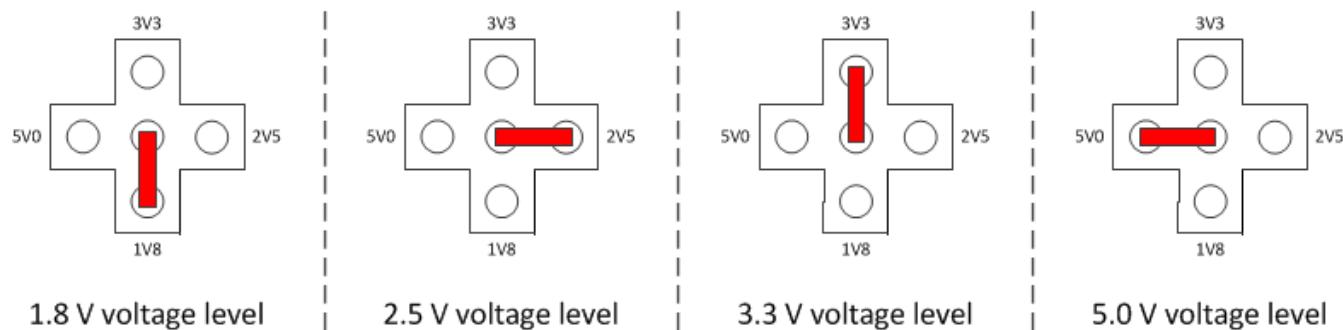
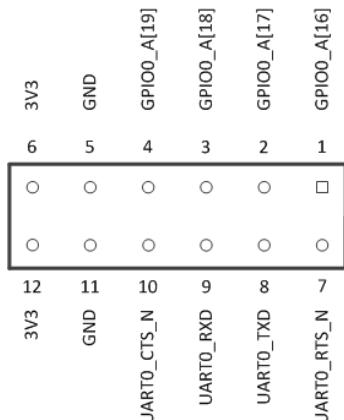


Figure 38 Voltage level selection for the Extension0 connector

### 2.16.3.2 UART Interfaces at the Pmod2 and Extension2 Connectors

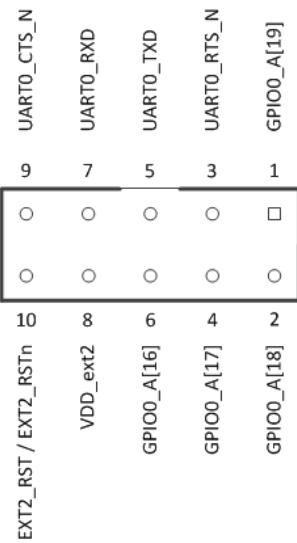
Figure 39 and Figure 40 show the pinout diagrams of the Pmod2 connector and the Extension2 connector for the following case:

- UART0 re-routed to Pmod2 / Extension2 (PM2[0]=0, PM2[2]=1)



PM2[0] = 0, PM2[2] = 1

Figure 39 Pinout diagram of the Pmod2 connector in UART mode (UART0)



PM2[0] = 0, PM2[2] = 1

Figure 40 Pinout diagram of the Extension2 connector in UART mode (UART0)

The reset signal on pin 10 of Extension2 is active high when the jumper JP1505 is in place and active low when this jumper is removed.

The voltage level at Extension2 is selected by jumper JP2603 as shown in Figure 41.

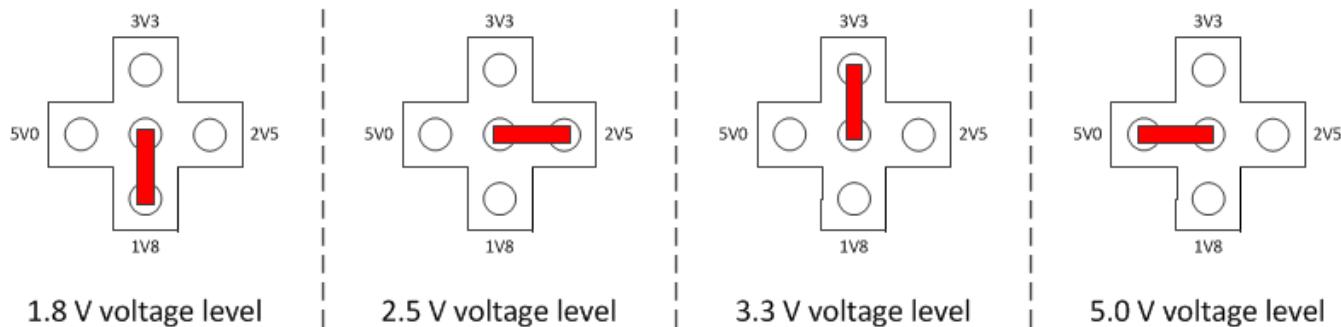


Figure 41 Voltage level selection for the Extension2 connector

## 2.17 SPI Interfaces

The ARC SDP Mainboard supports up to two external SPI master interfaces located at the Pmod or Extension interfaces. Both SPI interfaces are connected to individual SPI peripherals (SPI0 and SPI1) and are therefore fully independent of each other. They support all four SPI modes (polarities of clock and data lines), which can be selected by software.

Two additional SPI peripherals SPI2 and SPI\_MEM are available for internal communication with the CPLD and the SPI-Flash memory.

The remainder of this section describes using SPI0 and SPI1 for external serial communication.

### 2.17.1 Routing Options for External SPI Interfaces

The SPI peripherals SPI0 and SPI1 can optionally be connected to the external interfaces Pmod2 / Extension2 and Pmod3 / Extension3. Routing is controlled by software using the PMOD\_MUX\_CTRL Register as described in Table 26.

*Table 26 SPI Interface selection*

<b>Peripheral</b>	<b>PMOD_MUX_CTRL Register</b>			<b>Selected Connector</b>
	<b>PM2[0]</b>	<b>PM3[0]</b>	<b>PM3[2]</b>	
SPI0	Ignored	0	Ignored	None (default after reset)
	Ignored	1	Ignored	Pmod3 upper row External3 (pins 1,2,4,6)
SPI1	0	Ignored	0	None (default after reset)
	1	Ignored	0 or 1	Pmod2 upper row External2 (pins 1,2,4,6)
	0	Ignored	1	Pmod3 lower row External3 (pins 3,5,7,9)
SPI2	Ignored	Ignored	Ignored	Internal bus to CPLD
SPI_MEM	Ignored	Ignored	Ignored	Internal bus SPI Flash memory

### 2.17.2 Default SPI Connectors

The ARC SDP Mainboard does not have a dedicated SPI interface. This means that after a reset none of the SPI peripherals are routed to any external connector.

### 2.17.3 Using Extension Interfaces in SPI Mode

As listed in Table 26 above the SPI interfaces can be re-routed to the extension connectors Pmod2 / Extension2 and Pmod3 / Extension3 by programming the PMOD\_MUX\_CTRL register.

Figure 42 shows the location of the Pmod2, Pmod3, Extension2 and Extension3 connectors supporting SPI mode. The jumpers for setting the voltage levels and the reset polarities at Extension2 and Extension3 are shown as well.

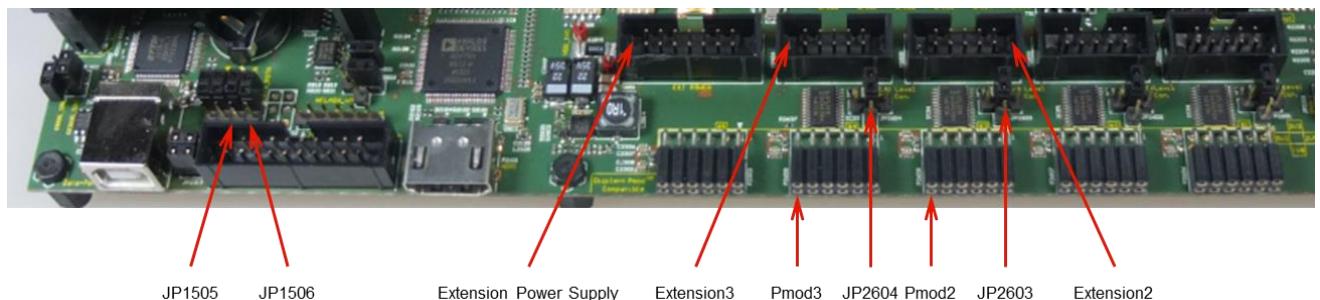
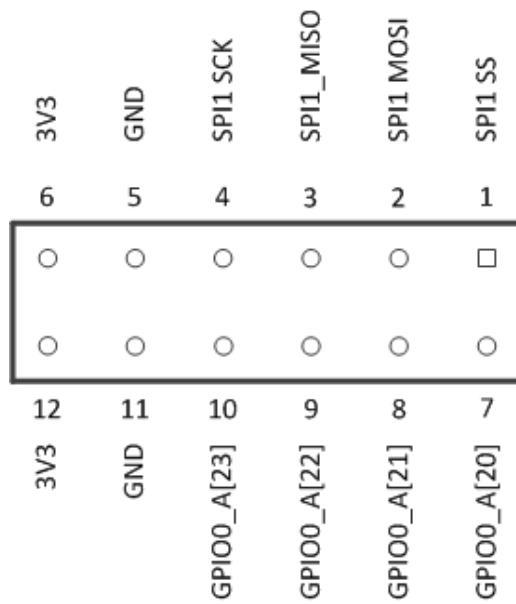


Figure 42 Location of Pmod and Extension connectors supporting SPI mode

### 2.17.3.1 SPI Interfaces at the Pmod2 and Extension2 Connectors

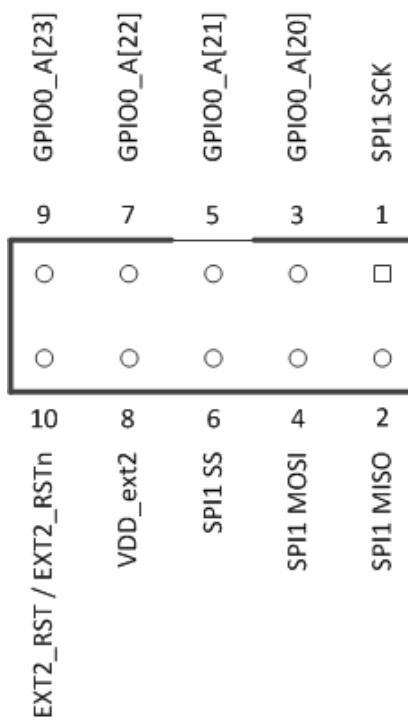
Figure 43 and Figure 44 show the pinout diagrams of the Pmod2 connector and the Extension2 connector for the following case:

- SPI1 routed to Pmod2 / Extension2 (PM2[0]=1, PM2[2]=0)



PM2[0] = 1, PM2[2] = 0

Figure 43 Pinout diagram of the Pmod2 connector in SPI mode (SPI1)



$\text{PM2}[0] = 1, \text{PM2}[2] = 0$

Figure 44 Pinout diagram of the Extension2 connector in SPI mode (SPI1)

The reset signal on pin 10 of Extension2 is active high when the jumper JP1505 is in place and active low when this jumper is removed.

The voltage level at Extension2 is selected by jumper JP2603 as shown in Figure 45.

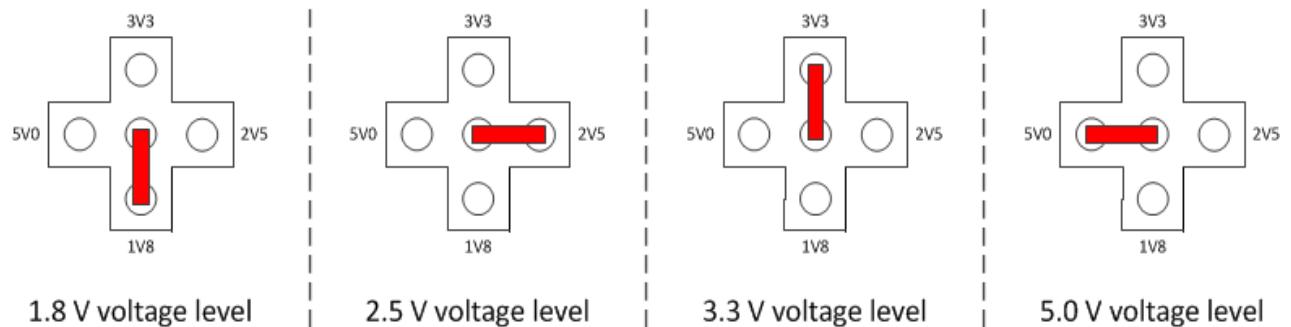


Figure 45 Voltage level selection for the Extension2 connector

### 2.17.3.2 SPI Interfaces at the Pmod3 and Extension3 Connectors

Figure 46 and Figure 47 show the pinout diagrams of the Pmod3 connector and the Extension3 connector for the following three cases:

- SPI0 routed to Pmod3 / Extension3 (PM3[0]=1, PM3[2]=0)
- SPI1 routed to Pmod3 / Extension3 (PM3[0]=0, PM3[2]=1)
- SPI0 and SPI1 routed to Pmod3 / Extension3 (PM3[0]=1, PM3[2]=1)

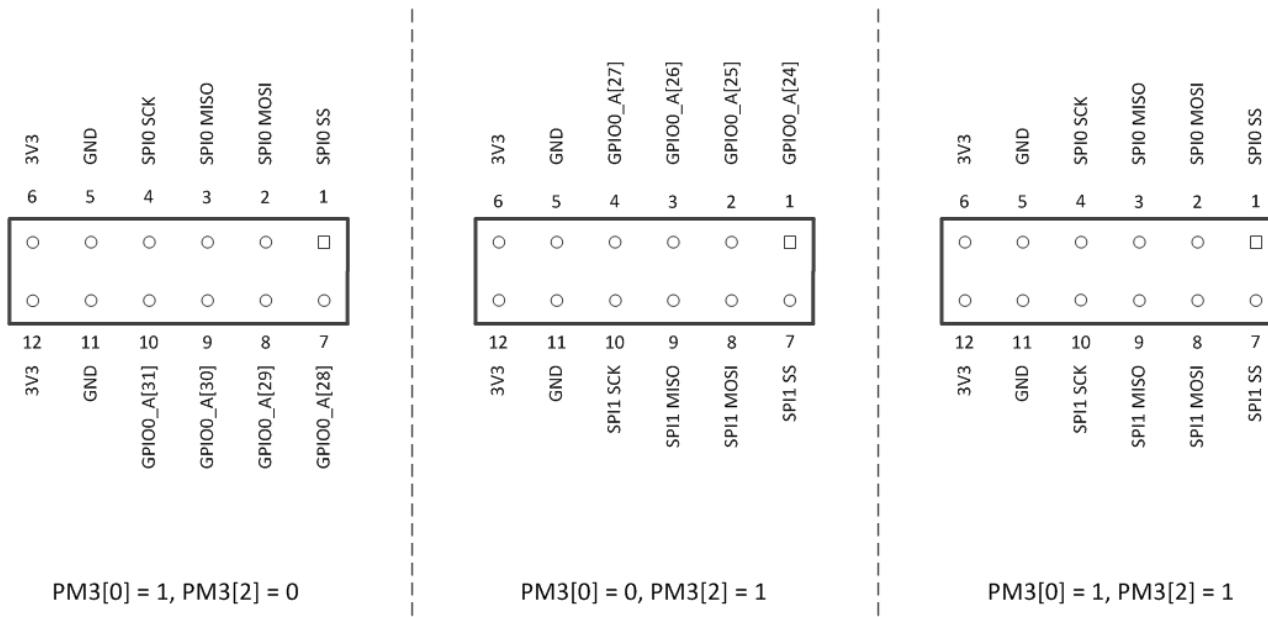


Figure 46 Pinout diagram of the Pmod3 connector in SPI mode (SPI0 / SPI1)

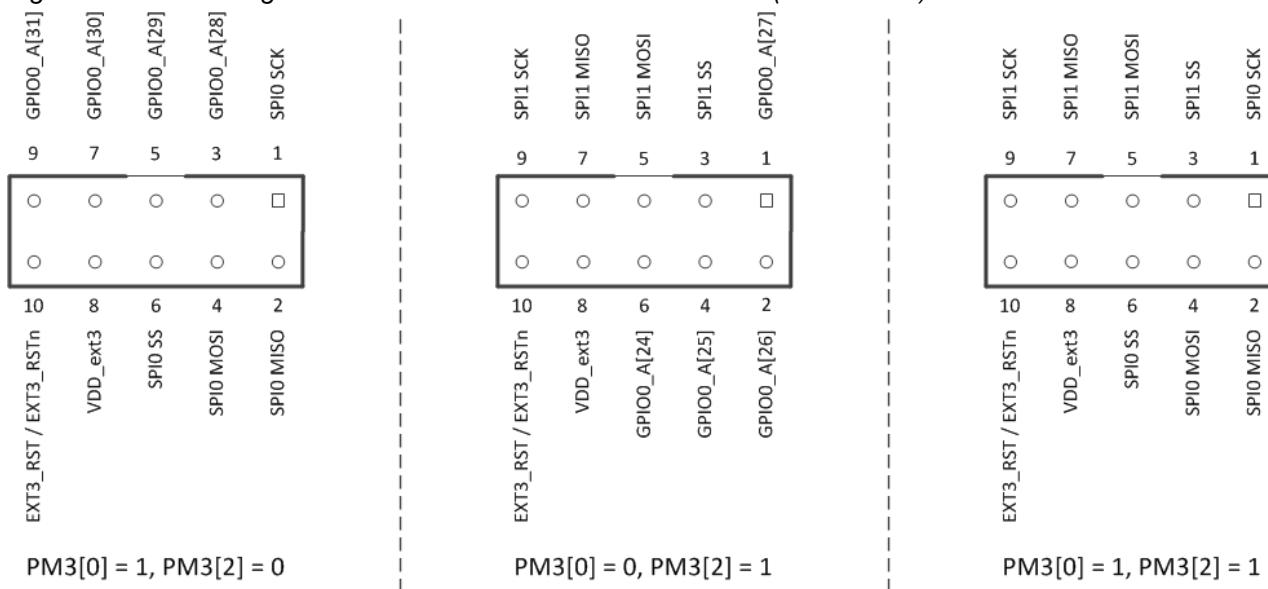


Figure 47 Pinout diagram of the Extension3 connector in SPI mode (SPI0 / SPI1)

The reset signal on pin 10 of Extension3 is active high when the jumper JP1506 is in place and active low when this jumper is removed.

The voltage level at Extension3 is selected by jumper JP2604 as shown in Figure 48.

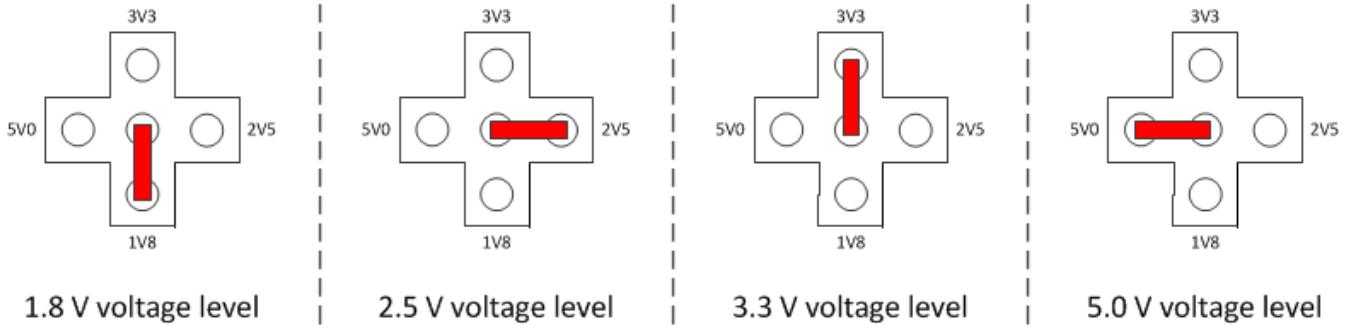


Figure 48 Voltage level selection for the Extension3 connector

## 2.18 Debug

The ARC SDP Mainboard features a converter, providing a JTAG debug interface and a debug console through a single USB cable connected to the USB Dataport. Therefore, a dedicated debug probe is not required. The following options are supported for debugging SW that is running on a CPU at an ARC CPU Card:

- USB cable (no probes required)  
To be connected to the USB Dataport
- Digilent JTAG HS1 Programming Cable  
To be connected to the 6-pin connector CPU Debug Digilent
- Ashling Opella-XD (TPAOP-ARC20)  
To be connected to the 20-pin JTAG connector CPU Debug Ashling / Lauterbach
- Lauterbach probe  
To be connected to the 20-pin JTAG connector CPU Debug Ashling / Lauterbach

The jumper JP1402 selects between using the JTAG channel of the USB Dataport or using one of the probe connectors. See the sub-sections below for information on the individual debug connectors and the required jumper settings for a particular probe. By default, JP1402 selects the USB Dataport.

Figure 49 shows the location of the debug interfaces and the corresponding jumpers on the Mainboard.

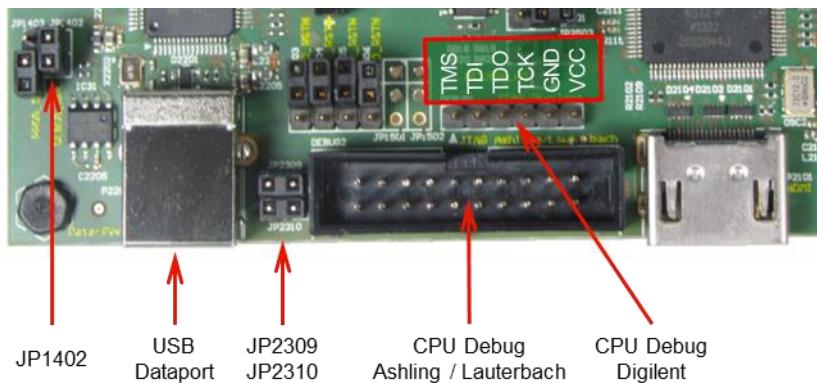


Figure 49 Location of the debug interfaces and the corresponding jumpers

## 2.18.1 Using the USB Dataport for Debugging

The USB Dataport has a USB type B jack and can be connected to your PC using the USB cable included in the product package. The jumper setting for connecting the debugger via the USB Dataport is shown in Figure 50 below.



Figure 50 Jumper setting for connecting the debugger via the USB Dataport

## 2.18.2 Using an Ashling Opella XD Probe

Debug probes by Ashling can be connected to the CPU Debug Ashling / Lauterbach connector. Two jumpers allow making the connector compatible with the Ashling Opella-XD (TPAOP-ARC20) debug probe or with Lauterbach debug probes. A pin description for the Ashling mode is provided in Table 27.

Table 27 Pin description of the CPU Debug Ashling/Lauterbach connector in Ashling mode

Pin	Name	Description
1	VDD	JTAG IO reference voltage, 1V8
2		Not connected
3	DBG_RSTN	Debug probe reset (active low)
4	GND	Ground supply pin
5	DBG_TDI	Debug probe test data in

Pin	Name	Description
6	GND	Ground supply pin
7	DBG_TMS	Debug probe test mode select
8	GND	Ground supply pin
9	DBG_TCLK	Debug probe test clock
10	GND	Ground supply pin
11	TC_RTCK	Target probe return test clock
12	GND	Ground supply pin
13	DBG_TDO	Debug probe test data out
14	GND	Jumper JP2309 selects whether this pin is open or connected to ground. Connect this pin to GND for Ashling probes.
15	RSTn	Reset (active low)
16	GND	Jumper JP2310 selects whether this pin is open or connected to ground. Connect this pin to GND for Ashling probes.
17	TC_EVTI	Target probe event in
18	GND	Ground supply pin
19	TC_EVTO	Target probe event out
20	GND	Ground supply pin

The jumper JP1402 selects whether the USB Dataport is used for debugging or whether a debug probe is connected to one of the CPU Debug connectors (CPU Debug Ashling / Lauterbach or CPU Debug Digilent). The jumpers JP2309 and JP2310 select the type of the debug probe. The jumper settings for connecting an Ashling probe to the CPU Debug Ashling / Lauterbach connector and the location of the pins on the connector are shown in Figure 51.

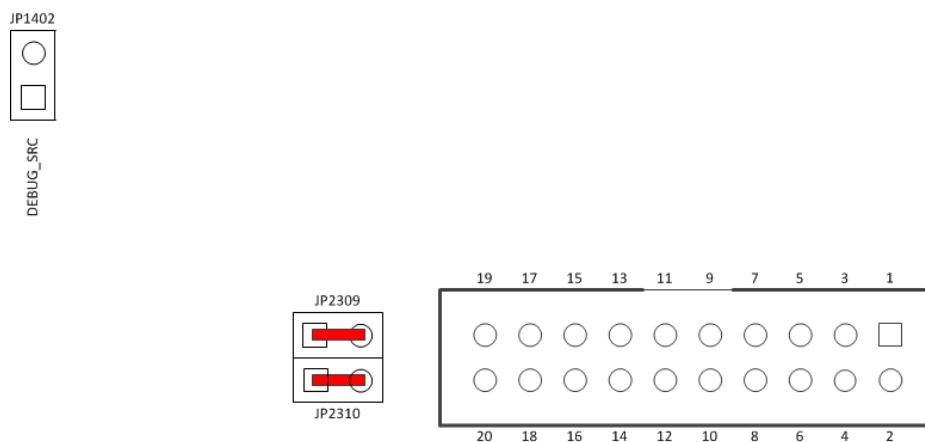


Figure 51 Pinout diagram and jumper settings for connecting an Ashling probe

### 2.18.3 Using Lauterbach Probes

Debug probes by Lauterbach can be connected to the CPU Debug Ashling / Lauterbach connector. Two jumpers allow making the connector compatible with the Ashling Opella-XD (TPAOP-ARC20) debug probe or with Lauterbach debug probes. A pin description of this connector in Lauterbach mode is listed in Table 28.

Table 28 Pin description of the CPU Debug Ashling/Lauterbach connector in Lauterbach mode

Pin	Name	Description
1	VDD	JTAG IO reference voltage, 1V8
2		Not connected
3	DBG_RSTN	Debug probe reset (active low)
4	GND	Ground supply pin
5	DBG_TDI	Debug probe test data in
6	GND	Ground supply pin
7	DBG_TMS	Debug probe test mode select
8	GND	Ground supply pin
9	DBG_TCLK	Debug probe test clock
10	GND	Ground supply pin
11	TC_RTCK	Target probe return test clock
12	GND	Ground supply pin
13	DBG_TDO	Debug probe test data out
14	n.c.	Jumper JP2309 selects whether this pin is open or connected to ground.

Pin	Name	Description
		This pin is not connected in Lauterbach mode.
15	RSTn	Reset (active low)
16	n.c.	Jumper JP2310 selects whether this pin is open or connected to ground. This pin is not connected in Lauterbach mode.
17	TC_EVTI	Target probe event in
18	GND	Ground supply pin
19	TC_EVTO	Target probe event out
20	GND	Ground supply pin

The jumper JP1402 selects whether the USB Dataport is used for debugging or whether a debug probe is connected to one of the CPU Debug connectors (CPU Debug Ashling / Lauterbach or CPU Debug Digilent). The jumpers JP2309 and JP2310 select the type of the debug probe. The jumper settings for connecting a Lauterbach probe to the CPU Debug Ashling / Lauterbach connector and the location of the pins on the connector are shown in Figure 52.

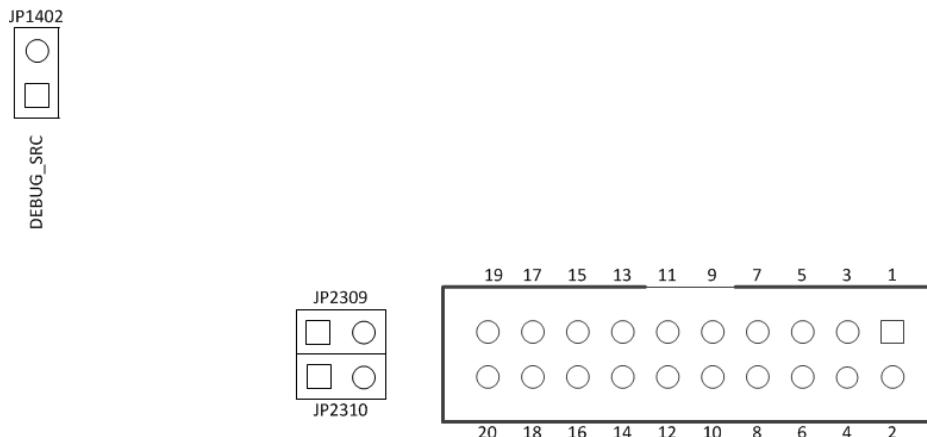


Figure 52 Pinout diagram and jumper settings for connecting a Lauterbach probe

#### 2.18.4 Using a Digilent Probe

Debug probes by Digilent can be connected to the CPU Debug Digilent connector. The jumper JP1402 selects whether the USB Dataport is used for debugging or whether a debug probe is connected to one of the CPU Debug connectors (CPU Debug Ashling / Lauterbach or CPU Debug Digilent).

A pin description of the CPU Debug Digilent connector is provided in Table 29 and the pinout is shown in Figure 53 as well as the corresponding jumper settings.



**Note** The MetaWare Debugger should be started using one of the following command line options depending on the type of your probe:

-prop=dig\_device=JtagHs1 or -prop=dig\_device=JtagHs2

Table 29 Pin description of the CPU Debug Digilent connector

Pin	Name	Description	Direction
1	TMS	Test mode select	Input
2	TDI	Test data in	Output
3	TDO	Test data out	Input
4	TCK	Test clock	Input
5	GND	Ground supply pin	-
6	VDD	JTAG IO reference voltage, 1V8	Output

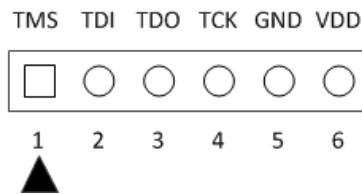
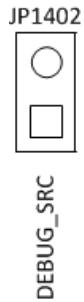


Figure 53 Pinout diagram and jumper settings for connecting a Digilent probe

## 2.19 Push Buttons, Switches and LEDs

The ARC SDP Mainboard supports multiple buttons, switches, and LEDs. A part of these resources is connected to on-board devices such as the peripheral subsystem or the status

monitor. Another part is connected to the CPU Card Connector and is used directly by the ARC CPU Card.

The push buttons, switches, and LEDs are connected to GPIO1.



Figure 54 Devices on GPIO1

### 2.19.1 Push Buttons

The GPIO push buttons are connected to port A of the GPIO1 module within the peripheral subsystem on the ARC SDP Mainboard.

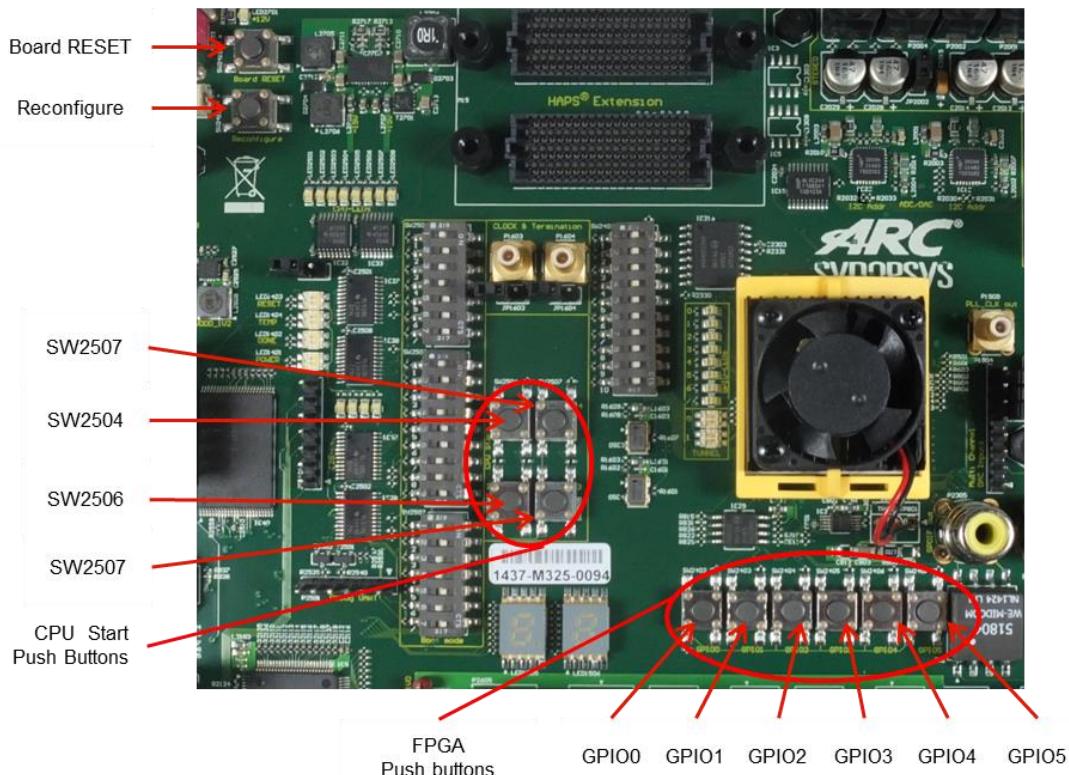


Figure 55 Location of the push buttons

Table 30 on page 72 list the push buttons and their functions. Note that *GPIO0* as a button label refers to a push button on GPIO1.

Table 30 List of push buttons

Device Name	GPIO1 Bit	Description
Board RESET	Not applicable	Pushing this button performs a reset of the ARC SDP Mainboard and the ARC CPU Card. Additionally, the reset outputs at the Extension connectors become active.
Reconfigure	Not applicable	Pushing this button causes the onboard FPGA to re-load its configuration information from the FPGA Flash memory.
GPIO0	EXT_PORTA[24]	The button is active high and can be used for custom applications. Note that <i>GPIO0</i> as a button label refers to a push button on GPIO1.
GPIO1	EXT_PORTA[25]	The button is active high and can be used for custom applications. Note that <i>GPIO1</i> as a button label refers to a push button on GPIO1.
GPIO2	EXT_PORTA[26]	The button is active high and can be used for custom applications.
GPIO3	EXT_PORTA[27]	The button is active high and can be used for custom applications.
GPIO4	EXT_PORTA[28]	The button is active high and can be used for custom applications.
GPIO5	EXT_PORTA[29]	The button is active high and can be used for custom applications.
SW2504 SW2505 SW2506 SW2507		These push buttons are connected to the CPU Card Connector and are directly used by the ARC CPU Card. Refer to the documentation of your ARC CPU Card for information on how these push buttons are used.

The GPIO1 module, to which the six push buttons GPIO0 to GPIO5 are connected, can be programmed to perform deglitching and to raise interrupts when a button is pushed. The clock frequency for the deglitching circuit is fixed at 781.25 kHz, which is sufficiently low to filter out any unwanted transients when a push button is pressed.

## 2.19.2 Switches

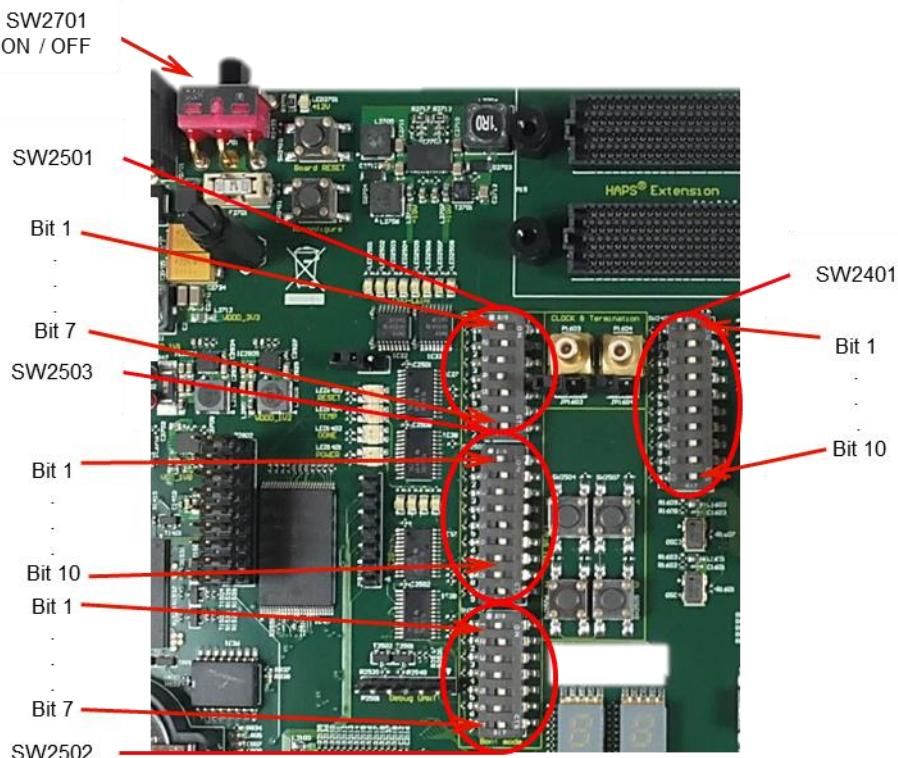


Figure 56 Location of the switches

Table 31 List of switches

Device Name	Description
SW2701	On/off switch for the 12 V main power supply.

Device Name	Description			
	DIP Switch	GPIO1 Bit	Function	
SW2401	Bit 1	EXT_PORTB[0]	For application purposes	
	Bit 2	EXT_PORTB[1]		
	Bit 3	EXT_PORTB[2]		
	Bit 4	EXT_PORTB[3]		
	Bit 5	EXT_PORTB[4]		
	Bit 6	EXT_PORTB[5]		
	Bit 7	EXT_PORTB[6]		
	Bit 8	EXT_PORTB[7]		
	Bit 9	EXT_PORTB[8]		
	Bit 10	EXT_PORTB[9]		
SW2501	This 10-bit DIP switch is connected to port B of the GPIO1 module of the peripheral subsystem on the Mainboard. The bit-mapping is shown below.			
SW2502	This 7-bit DIP switch is connected to the CPU Card Connector while the reset status output of the CPU Card is active. The switch settings are latched on the CPU Card at the de-assertion of the reset.			
SW2503	This 7-bit DIP switch is connected to the CPU Card Connector while the reset status output of the CPU Card is active. The switch settings are latched on the CPU Card at the de-assertion of the reset.			
This 9-bit DIP switch is connected to the CPU Card Connector while the reset status output of the CPU Card is active. The switch settings are latched on the CPU Card at the de-assertion of the reset.				

Refer to the documentation of your ARC CPU Card for information on how the switches SW2501, SW2502 and SW2503 are used.

### 2.19.3 LEDs

The ARC SDP Mainboard includes the following LEDs:

- Seven LEDs indicating the board status
- Eight green GPIO LEDs for custom
- Two dual-color LEDs (green/red) for custom applications
- Eight green CPU LEDs; usage depends on the type of the ARC CPU Card
- Five LEDs indicating the status of the Ethernet PHY
- Four USB Dataport mode LEDs

### 2.19.3.1 Board Status LEDs

Table 32 lists the board status LEDs and describes their function. Figure 57 shows the location of the LEDs on the board.

*Table 32 Board status LEDs*

Device Name	Description	
+12V	Green LED indicating that the 12V DC power supply is switched on.	
RESET (LED1403)	Indicates the reset status of the board:	
	Red	Reset is active
	Green	Normal operation
TEMP (LED1404)	Indicates the status of the temperature monitor	
	Red	FPGA temperature higher than 75 °C, 167 °F Switch OFF the board!.
	Flashing Red	FPGA temperature higher than 90 °C, 194 °F Switch OFF the board!
	Green	FPGA temperature OK
DONE (LED1402)	Indicates the status of the FPGA	
	Red	FPGA configuration ongoing
	Green	FPGA configuration complete
POWER (LED1401)	Indicates the status of the power monitor	
	Red	Failure One or several of the required voltage levels is not available.
	Green	All required voltage levels are OK
TUNNEL0	Indicates the status of the AXI tunnel to the ARC CPU Card	
	Red	Failure
	Green	OK
TUNNEL1	Indicates the status of the AXI tunnel to the HAPS system	
	Red	Failure
	Green	OK

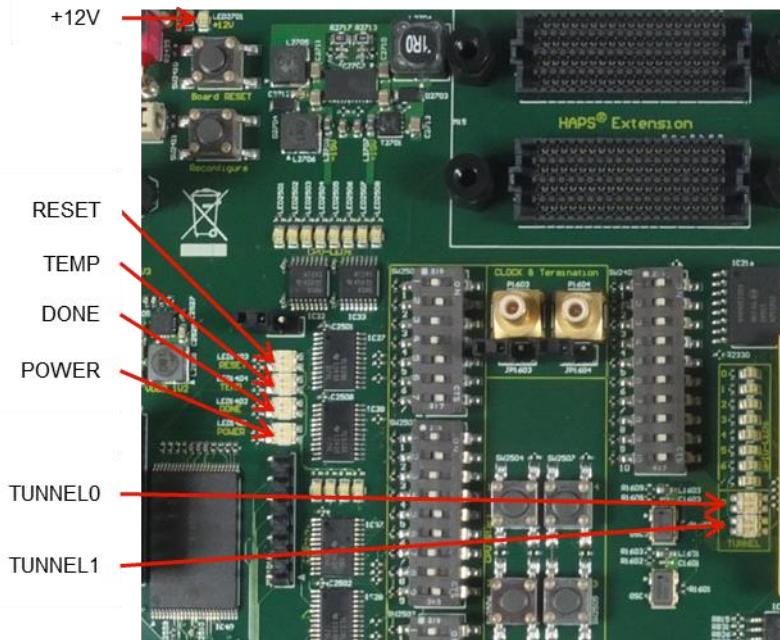


Figure 57 Location of the Board Status LEDs

### 2.19.3.2 GPIO LEDs

The GPIO LEDs are connected to port C of the GPIO1 module of the peripheral subsystem on the ARC SDP Mainboard. They can be used for custom applications.

Table 33 describes the function of these LEDs. Figure 58 shows the location of the GPIO LEDs on the board.

Table 33 GPIO LEDs (controlled by Mainboard GPIO1 module)

LED Name	Control Bit	Description
GPIO LED0	SWPORTC_DR[0]	Green LED ON when control bit is set to '1'
GPIO LED1	SWPORTC_DR[1]	Green LED ON when control bit is set to '1'
GPIO LED2	SWPORTC_DR[2]	Green LED ON when control bit is set to '1'
GPIO LED3	SWPORTC_DR[3]	Green LED ON when control bit is set to '1'
GPIO LED4	SWPORTC_DR[4]	Green LED ON when control bit is set to '1'
GPIO LED5	SWPORTC_DR[5]	Green LED ON when control bit is set to '1'

LED Name	Control Bit	Description
GPIO LED6	SWPORTC_DR[6]	Green LED ON when control bit is set to '1'
GPIO LED7	SWPORTC_DR[7]	Green LED ON when control bit is set to '1'

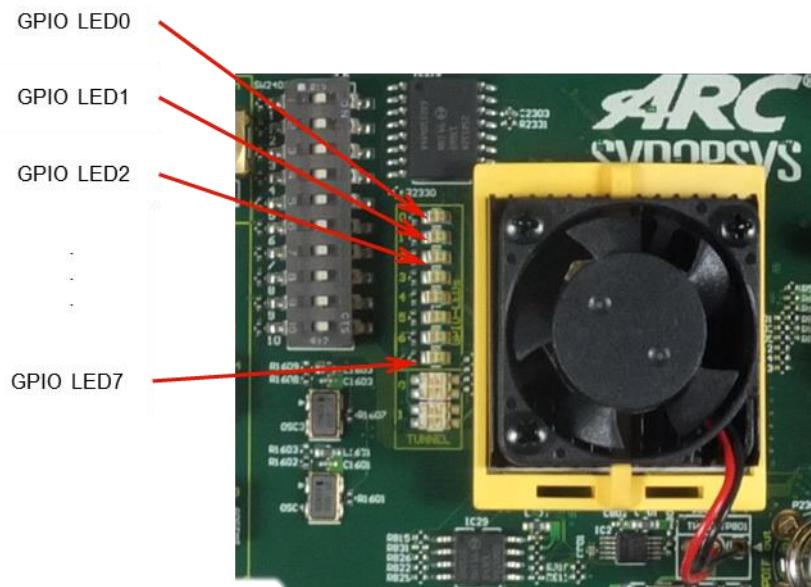


Figure 58 Location of the GPIO LEDs

#### 2.19.3.3 CPU LEDs

Eight green CPU LEDs are connected to the CPU Card Connector. Refer to the documentation of your ARC CPU Card for detailed information on their function. Figure 59 shows the location of the CPU LEDs on the board.

### CPU LEDs

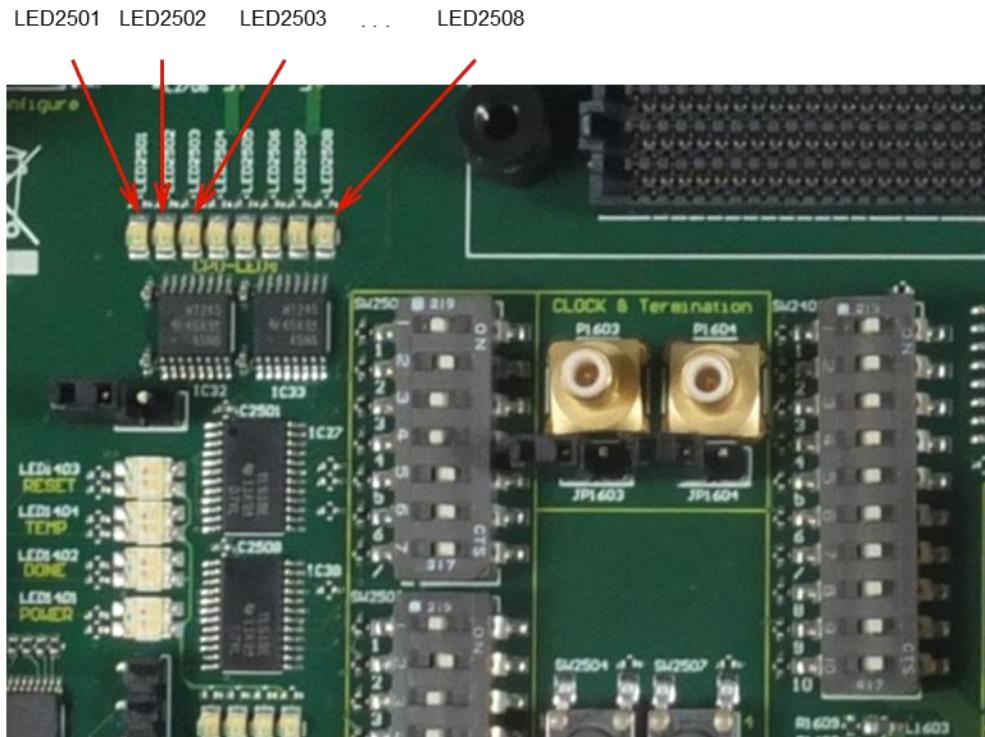


Figure 59 Location of the CPU LEDs

#### 2.19.3.4 Ethernet Status LEDs

See the "[Ethernet](#)" section for detailed information.

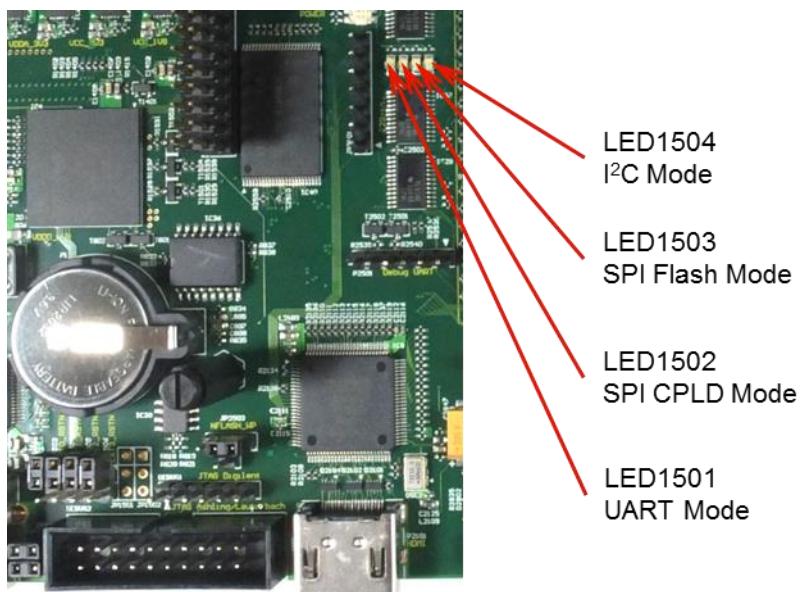
#### 2.19.3.5 USB Dataport Mode LEDs

The four USB Dataport mode LEDs indicate the operating mode of the USB Dataport. The operating mode is set by the [axs\\_comm](#) tool (see "[AXS Communicator Tool – axs\\_comm](#)"). Four modes are available:

- Mode 1: USB Dataport is in UART mode.  
That means that the FPGA UART2 signals are routed to the USB Dataport.
- Mode 2: USB Dataport is in SPI-CPLD mode.  
This mode allows accessing internal registers of the CPLD.
- Mode 3: USB Dataport is in SPI-FLASH mode.  
The SPI FLASH memory is accessible via the USB Dataport.
- Mode 4: USB Dataport is in I<sup>2</sup>C mode.  
The internal I<sup>2</sup>C bus (see "[Internal I<sup>2</sup>C Bus](#)") is accessible via the USB Dataport.

*Table 34 USB Dataport LEDs*

<b>LED Name</b>	<b>Description</b>
LED1501	Green LED, USB Dataport is in UART mode
LED1502	Green LED, USB Dataport is in SPI-CPLD mode
LED1503	Green LED, USB Dataport is in SPI-FLASH mode
LED1504	Green LED, USB Dataport is in I <sup>2</sup> C mode

*Figure 60 Location of the USB Dataport Mode LEDs*

## 2.20 Seven-Segment Display

The ARC SDP Mainboard includes a dual seven-segment display, which is controlled by the SPI master peripheral SPI2 that is located in the peripheral subsystem.

The seven-segment display is used to show status information of the pre-bootloader, refer to the documentation of your CPU Card for details.

## 2.21 On-Board Memories

The following memories are available on the ARC SDP Mainboard:

- 512 Byte I<sup>2</sup>C EEPROM
- 64 Mbyte SPI Flash for general purpose usage
- 16 Mbyte SPI Flash used to store the FPGA image. The flash device supports legacy SPI protocol the new Quad I/O or Dual I/O SPI protocol.
- 512 Mbyte NAND Flash
- 256 Kbyte RAM

Additionally, the ARC SDP Mainboard includes a small I<sup>2</sup>C EEPROM, which is connected to the HAPS extension interface. This memory stores the HAPS type and identity and is not available for user purposes.

---

### 2.21.1 I<sup>2</sup>C EEPROM

This EEPROM has an I<sup>2</sup>C interface and can store 512 bytes. The I<sup>2</sup>C interface is hooked up to the internal I<sup>2</sup>C bus (see “[Internal I<sup>2</sup>C Bus](#)”). The I<sup>2</sup>C2 peripheral of the peripheral subsystem is a master on this bus. Additionally, the EEPROM can be accessed via the USB Dataport, which can be set to operate as another I<sup>2</sup>C master on this bus.

The 7-bit I<sup>2</sup>C address of this EEPROM is “1010100” for the lower 256 bytes and “1010101” for the upper 256 bytes. This means that the I<sup>2</sup>C address byte should be set to 0xA8 or 0xAA for writing and to 0xA9 or 0xAB for reading.

This memory is available for application purposes.

---

### 2.21.2 SPI Flash Application Memory

This SPI flash memory can store 64 Mbyte. It is typically used to store the boot code for the processor(s) on the ARC CPU Card. This memory is controlled by an SPI module of the peripheral subsystem, which includes a DMA handshake interface and thus allows high data throughput.

The last 4 Kbyte subsector (address range from 0x3FF\_F000 to 0x3FF\_FFFF) is reserved for internal purposes of the ARC Software Development Platform. Nevertheless, it is allowed to erase this sector, for example, as part of a full die erase.

This memory can be initialized using the USB Dataport. See “[Appendix B](#)” for more information.

---

### 2.21.3 FPGA-Configuration FLASH

This SPI flash memory can store 16 Mbyte. It is used to store the FPGA configuration including the initial content of the Mainboard RAM, which has been implemented in the FPGA. Additionally, it stores the boot loader code.

## 2.21.4 NAND Flash

The ARC SDP Mainboard includes a 512 Mbyte NAND flash memory for application purposes. It can be used to store program code or data.

The NAND flash memory can be write-protected. The protection mode can be SW-controlled or hard-wired depending on jumper settings at the header JP2503. Figure 61 shows the jumper settings for the different modes. Figure 62 shows the location of the header on the Mainboard.

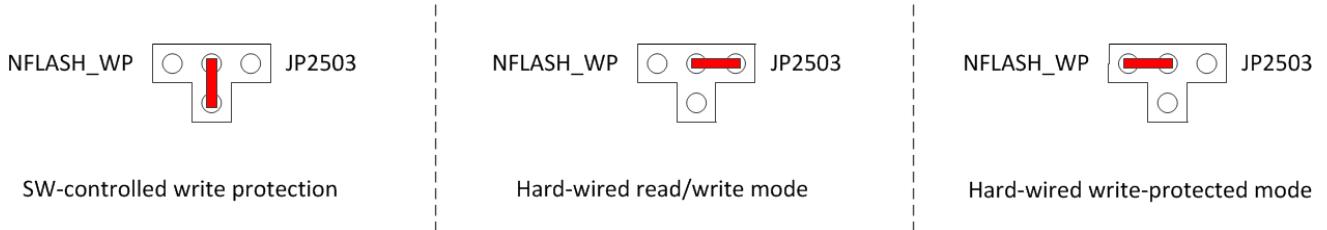


Figure 61 Jumper settings for NAND flash write protection

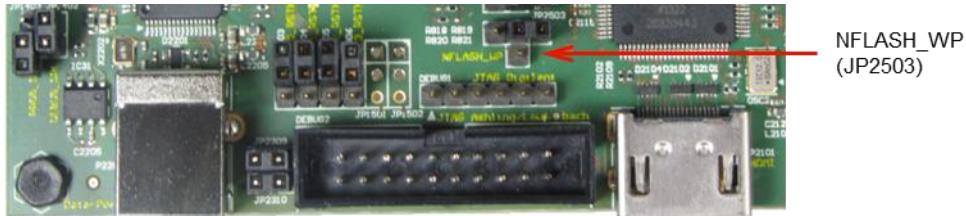


Figure 62 Location of the NAND Flash write protection header

## 2.21.5 RAM

The peripheral subsystem implemented in the FPGA includes 256 Kbyte of RAM. This memory is initialized during FPGA configuration, i.e. the initial content is included in the FPGA image, which is stored in the FPGA serial flash memory. By default, this memory contains the boot loader code. After booting the memory can be used as RAM.

Note that pushing the `Board RESET` button does not re-load the boot code. If this memory is used for other purposes, then the `Reconfigure` button has to be used to re-load the boot code.

## 2.22 Power Supply

The main power of the board is provided by means of a 12V DC power inlet. The ARC SDP Mainboard package includes a 100-240 V AC power adapter.

The switch `SW2701` allows to switch the board ON or OFF. The LED `+12V` shines green when the 12V source is ON.

All required voltage levels are generated on the board, and a CPLD board supervisor monitors the power supply. The **POWER** LED shines green, if the generated voltage levels are OK.

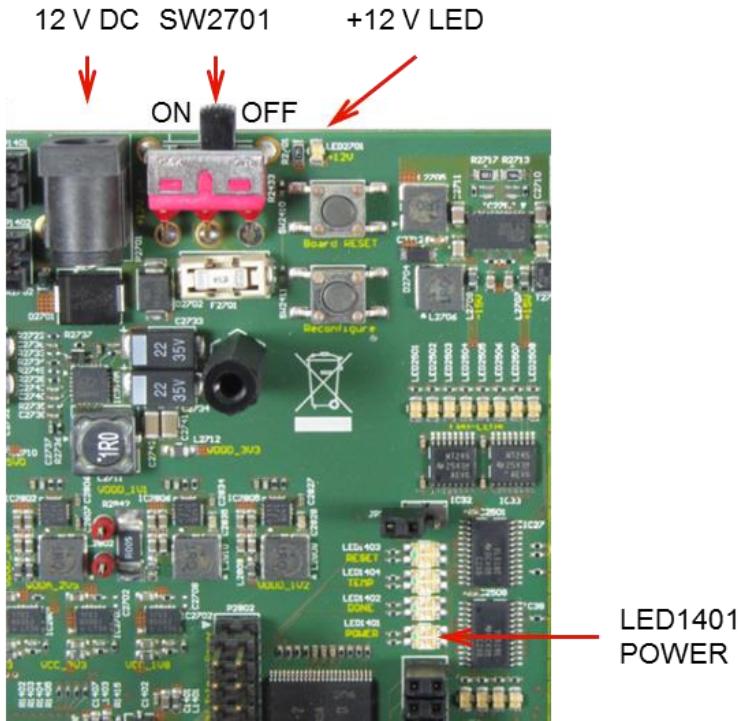


Figure 63 Location of power inlet, power switch and power status LEDs

## 2.23 Peripheral Subsystem FPGA Overview

The peripheral subsystem is based on an AXI interconnect network and has two AXI tunnels to the ARC CPU Card and to the HAPS system.

It includes controllers for the on-board USB and Ethernet PHYs, HDMI Transmitter, SD-card, NAND Flash and RAM. Additionally, it includes an Interrupt Controller, a DMA Controller, a clock generator unit, and control registers. Multiple instances of serial peripherals ( $I^2C$ , SPI, UART) and general-purpose I/O (GPIO) are available.

The peripheral subsystem also includes 256 Kbyte of RAM and the ability to perform pin-multiplexing.

Figure 64 shows a principle block diagram of the peripheral subsystem.

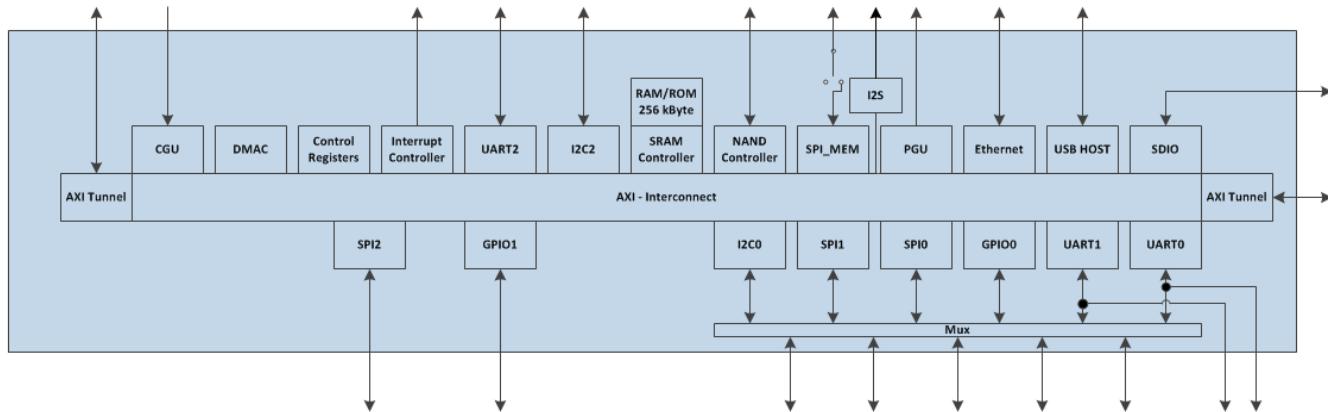


Figure 64 Principle block diagram of the peripheral subsystem

### 2.23.1 AXI Tunnel

An AXI tunnel allows connecting AXI buses from different boards or chips using a reduced set of wires. The two AXI tunnels of the peripheral subsystem allow seamless integration of the peripheral subsystem with the AXI systems on the ARC CPU Card and on a HAPS system. The status of the AXI tunnels is indicated by the two LEDs `TUNNEL0` and `TUNNEL1`, which shine green when the status of the respective tunnel is OK, or red in case of an error. `TUNNEL0` reflects the status of the AXI tunnel to the ARC CPU Card and `TUNNEL1` reflects the status of the tunnel to the HAPS system.

### 2.23.2 Interrupt Controller (ICTL)

All interrupts generated by the internal interrupt sources (such as USB, SPI) are routed to the ICTL module, which combines them into a single interrupt request. The interrupt source responsible for generating the interrupt request can be determined by reading back the interrupt status register in the ICTL module.

The interrupt output of the ICTL module is routed to the `GPIO_A[12]` input of the ARC CPU Card Connector.

The Synopsys DesignWare `DW_apb_ictl` IP is used to implement the ICTL. Use the corresponding driver to control the interrupts and to read the interrupt status.

The mapping of the interrupt sources onto ICTL registers is listed in Table 35. All interrupts are active high and level sensitive.

Table 35 ICTL Interrupt mapping

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

### 2.23.3 Clock Generator Unit (CGU)

Both 27 Mhz and 25 MHz reference clocks are supplied to the CGU, which generates all the internal clocks. See “[Clock Generation](#)” for more details on the generated clocks and “[Clock Generation Registers](#)” for information on how to control the CGU by software.

### 2.23.4 DMA Controller (DMAC)

The main IP blocks inside the FPGA, i.e. GMAC, USB, SDIO, NAND, and PGU are all DMA capable and do not require an external DMA controller (or a host CPU) to transfer data to/from the main memory. However, the serial connectivity peripherals in the peripheral subsystem do not have DMA capability. Therefore, the FPGA is provided with a DMA controller, which allows efficient data transfer to/from the serial connectivity peripherals with minimum loading of the host CPU.

The Synopsys DesignWare DW\_ahb\_dmac IP [11] is used to implement the DMAC. The DMAC has two 64-bit AHB master ports for the data transfers, and a 32-bit AHB slave port for access to the control and status registers. Furthermore, the DMAC is configured with four DMA channels and DMA flow control interfaces for 16 DMA peripherals. The allocation of the DMA peripherals onto the flow control interfaces is shown in the table below:

*Table 36 DMA flow control interface mapping*

DMA flow control interface	DMA peripheral	TX/RX	Remarks
0 1	SPI_MEM	TX	Used for SPI flash
		RX	
2 3	SPI0	TX	
		RX	
4 5	SPI1	TX	
		RX	
6 7	I2S	TX	Used for I2S output to HDMI
		n/a	
8 9	I2C0	TX	
		RX	
10 11	UART0	TX	Supports 16570 modem flow control
		RX	
12 13	UART1	TX	
		RX	

DMA flow control interface	DMA peripheral	TX/RX	Remarks
14	UART2	TX	16550 compatible
15		RX	

## 2.23.5 SPI Controller for SPI Flash

During normal operation the SPI\_MEM module of the peripheral subsystem is used to control the SPI Flash. However, the system also supports programming the SPI Flash through the USB Dataport. When the USB Dataport is configured to operate in SPI mode, a multiplexer inside the peripheral subsystem disconnects the SPI Flash from the SPI\_MEM module and connects it to the USB Dataport.

## 2.23.6 GPIO Modules

The peripheral subsystem includes two GPIO modules. All GPIO ports can be programmed as inputs or outputs. Inputs can be configured to synchronize the incoming systems to the internal clock to avoid meta-stability problems. Deglitching is supported as well.

### 2.23.6.1 GPIO0

The GPIO0 module is interfacing to the Pmod connectors and the Extension headers. Port A of GPIO0 can be used to generate interrupts. Pin multiplexing is performed between GPIO0 ports and the interfaces of peripherals for serial communications. For detailed information about the pin-multiplexing, refer to the sections "[Pmod Extension Options](#)" and "[Other Extension Options](#)."

### 2.23.6.2 GPIO1

The GPIO1 module is connected to on-board switches and push-buttons. It also controls eight on-board LEDs.

Table 37 GPIO1 I/O register function

Register	Usage
GPIO1 EXT_PORTA[24]	Push button GPIO0
GPIO1 EXT_PORTA[25]	Push button GPIO1
GPIO1 EXT_PORTA[26]	Push button GPIO2
GPIO1 EXT_PORTA[27]	Push button GPIO3
GPIO1 EXT_PORTA[28]	Push button GPIO4
GPIO1 EXT_PORTA[29]	Push button GPIO5

Register	Usage
GPIO1 EXT_PORTB[9:0]	GPIO DIP switches SW2401
GPIO1 SWPORTC_DR[7:0]	GPIO LED[7:0]

For further details, refer to the "[Push Buttons, Switches and LEDs](#)" section.

## 2.24 HAPS HapsTrak-3 Extension

The ARC SDP Mainboard includes two HapsTrak-3 connectors to allow further extensions of the ARC Software Development Platform with additional FPGA resources using the HAPS family of prototyping solutions. This interface allows adding system extensions at various levels of complexity. Main features of the extension connector are:

- Carried signals: 82 AXI tunnel signals + 2 interrupts + 1 reset.
- Match trace length connections, since AXI tunnel is source synchronous
- Voltage level swing (FPGA bank voltage) 1.8 Volt to support all HAPS systems

For more information on the HAPS extension interface, refer to "[Appendix D](#)"

## 2.25 Pmod Extension Options

The ARC SDP Mainboard features five 12-pin Pmod connectors Pmod0, Pmod1, Pmod2, Pmod3 and Pmod4. The functionality of the Pmod connectors is programmable. The available options are summarized in

Table 38. Multiplexing is controlled by software using the PMOD\_MUX\_CTRL register (see "[Control Registers](#)"). After a reset, all ports are configured as GPIO inputs.

All data signals at the Pmod connectors have pull-up resistors to 3V3.

The Pmods and extensions are on GPIO0.

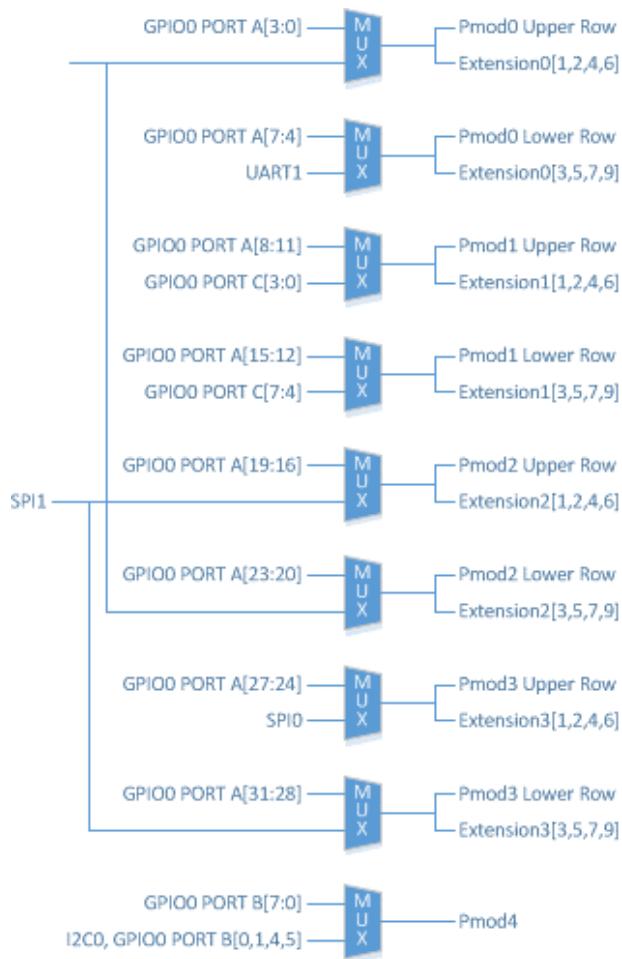


Figure 65 P Devices on GPIO0



**Note** The Pmod0 connector shares its pin-multiplexer with the Extension0 connector. This means that (with the exception of the reset output at Extension0) the connectors Pmod0 and Extension0 provide the same data signals. However, the voltage levels may be different.

Likewise the connectors Pmod1, Pmod2 and Pmod3 share their pin-multiplexers with the connectors Extension1, Extension2 and Extension3 respectively.

Table 38 Pmod pin multiplexing overview

Connector		Control Register	Peripherals	
Name	Pins	PMOD_MUX_CTRL Control Bit	MUX option 0	MUX option 1
Pmod0	upper row	PM0 [0]	GPIO0 A[3:0]	UART0
	lower row	PM0 [2]	GPIO0 A[7:4]	UART1

Connector		Control Register	Peripherals	
Name	Pins	PMOD_MUX_CTRL Control Bit	MUX option 0	MUX option 1
Pmod1	upper row	PM1 [0]	GPIO0 A[11:8]	GPIO0 C[3:0]
	lower row	PM1 [2]	GPIO0 A[15:12]	GPIO0 C[7:4]
Pmod2	upper row	PM2 [0]	GPIO0 A[19:16]	SPI1
	lower row	PM2 [2]	GPIO0 A[23:20]	UART0
Pmod3	upper row	PM3 [0]	GPIO0 A[27:24]	SPI0
	lower row	PM3 [2]	GPIO0 A[31:28]	SPI1
Pmod4	upper row	PM4 [0]	GPIO0 B[3:0]	I2C0
	lower row	PM4 [2]	GPIO0 B[7:4]	I2C0

Table 39, Table 40, and Table 41 describe the signals that are used in the pin description tables of the Pmod connectors.

Table 39 *UART signal description*

Name	Description	Direction
CTS_n	Clear to send, active low Handshake signal from external peripheral to on-board host	Input
RTS_n	Ready To Send, active low Handshake signal from on-board host to external peripheral	Output
RXD	Receive data Data from external peripheral to on-board host	Input
TXD	Transmit data Data clock from on-board host to external peripheral	Output

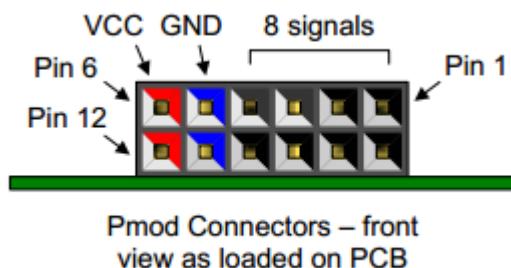
*Table 40 SPI Signal Description*

Name	Description	Direction
SS	Slave Select Active low to enable slave device	Output
MISO	Master Out Slave In Data from master to slave	Output
MOSI	Master In Slave Out Data from slave to master	Input
SCK	Serial clock Data clock from master to slave	Output

*Table 41 I<sup>2</sup>C Signal Description*

Name	Description	Direction
SCL	Serial Clock	Input / Output
SDA	Serial Data	Input / Output

The location of the pins on the Pmod connectors is shown in Figure 66. Detailed pin descriptions depending on the pin multiplexer settings are provided in Table 42, Table 43, Table 44, Table 45, and Table 46.

*Figure 66 Pinout diagram of the Pmod0, Pmod1, Pmod2, Pmod3 and Pmod4 connectors*

## 2.25.1 Pmod0 Connector

Table 42 Pin description of the Pmod0 connector

Pin	Standard Setting		Alternative Setting	
1	PM0[0]=0	GPIO0 A[0]	PM0[0]=1	UART0 RTS_N
2	PM0[0]=0	GPIO0 A[1]	PM0[0]=1	UART0 TXD
3	PM0[0]=0	GPIO0 A[2]	PM0[0]=1	UART0 RXD
4	PM0[0]=0	GPIO0 A[3]	PM0[0]=1	UART0 CTS_N
5	PM0[0]=0	GND	PM0[0]=1	GND
6	PM0[0]=0	3V3	PM0[0]=1	3V3
7	PM0[2]=0	GPIO0 A[4]	PM0[2]=1	UART1 RTS_N
8	PM0[2]=0	GPIO0 A[5]	PM0[2]=1	UART1 TXD
9	PM0[2]=0	GPIO0 A[6]	PM0[2]=1	UART1 RXD
10	PM0[2]=0	GPIO0 A[7]	PM0[2]=1	UART1 CTS_N
11	PM0[2]=0	GND	PM0[2]=1	GND
12	PM0[2]=0	3V3	PM0[2]=1	3V3

## 2.25.2 Pmod1 Connector

*Table 43 Pin description of the Pmod1 connector*

Pin	Standard Setting		Alternative Setting	
1	PM1[0]=0	GPIO0 A[8]	PM1[0]=1	GPIO0 C[0]
2	PM1[0]=0	GPIO0 A[9]	PM1[0]=1	GPIO0 C[1]
3	PM1[0]=0	GPIO0 A[10]	PM1[0]=1	GPIO0 C[2]
4	PM1[0]=0	GPIO0 A[11]	PM1[0]=1	GPIO0 C[3]
5	PM1[0]=0	GND	PM1[0]=1	GND
6	PM1[0]=0	3V3	PM1[0]=1	3V3
7	PM1[2]=0	GPIO0 A[12]	PM1[2]=1	GPIO0 C[4]
8	PM1[2]=0	GPIO0 A[13]	PM1[2]=1	GPIO0 C[5]
9	PM1[2]=0	GPIO0 A[14]	PM1[2]=1	GPIO0 C[6]
10	PM1[2]=0	GPIO0 A[15]	PM1[2]=1	GPIO0 C[7]
11	PM1[2]=0	GND	PM1[2]=1	GND
12	PM1[2]=0	3V3	PM1[2]=1	3V3

## 2.25.3 Pmod2 Connector

*Table 44 Pin description of the Pmod2 connector*

Pin	Standard Setting		Alternative Setting	
1	PM2[0]=0	GPIO0 A[16]	PM2[0]=1	SPI1_SS
2	PM2[0]=0	GPIO0 A[17]	PM2[0]=1	SPI1_MOSI
3	PM2[0]=0	GPIO0 A[18]	PM2[0]=1	SPI1_MISO
4	PM2[0]=0	GPIO0 A[19]	PM2[0]=1	SPI1_SCK
5	PM2[0]=0	GND	PM2[0]=1	GND
6	PM2[0]=0	3V3	PM2[0]=1	3V3
7	PM2[2]=0	GPIO0 A[20]	PM2[2]=1	UART0_RTS_N
8	PM2[2]=0	GPIO0 A[21]	PM2[2]=1	UART0_TXD
9	PM2[2]=0	GPIO0 A[22]	PM2[2]=1	UART0_RXD
10	PM2[2]=0	GPIO0 A[23]	PM2[2]=1	UART0_CTS_N
11	PM2[2]=0	GND	PM2[2]=1	GND
12	PM2[2]=0	3V3	PM2[2]=1	3V3

## 2.25.4 Pmod3 Connector

*Table 45 Pin description of the Pmod3 connector*

Pin	Standard Setting		Alternative Setting	
1	PM3[0]=0	GPIO0 A[24]	PM3[0]=1	SPI0 SS
2	PM3[0]=0	GPIO0 A[25]	PM3[0]=1	SPI0 MOSI
3	PM3[0]=0	GPIO0 A[26]	PM3[0]=1	SPI0 MISO
4	PM3[0]=0	GPIO0 A[27]	PM3[0]=1	SPI0 SCK
5	PM3[0]=0	GND	PM3[0]=1	GND
6	PM3[0]=0	3V3	PM3[0]=1	3V3
7	PM3[2]=0	GPIO0 A[28]	PM3[2]=1	SPI1 SS
8	PM3[2]=0	GPIO0 A[29]	PM3[2]=1	SPI1 MOSI
9	PM3[2]=0	GPIO0 A[30]	PM3[2]=1	SPI1 MISO
10	PM3[2]=0	GPIO0 A[31]	PM3[2]=1	SPI1 SCK
11	PM3[2]=0	GND	PM3[2]=1	GND
12	PM3[2]=0	3V3	PM3[2]=1	3V3

## 2.25.5 Pmod4 Connector

*Table 46 Pin description of the Pmod4 connector*

Pin	Standard Setting		Alternative Setting	
1	PM4[0]=0	GPIO0 B[0]	PM4[0]=1	GPIO0 B[0]
2	PM4[0]=0	GPIO0 B[1]	PM4[0]=1	GPIO0 B[1]
3	PM4[0]=0	GPIO0 B[2]	PM4[0]=1	I2C0 SCL
4	PM4[0]=0	GPIO0 B[3]	PM4[0]=1	I2C0 SDA
5	PM4[0]=0	GND	PM4[0]=1	GND
6	PM4[0]=0	3V3	PM4[0]=1	3V3
7	PM4[2]=0	GPIO0 B[4]	PM4[2]=1	GPIO0 B[4]
8	PM4[2]=0	GPIO0 B[5]	PM4[2]=1	GPIO0 B[5]
9	PM4[2]=0	GPIO0 B[6]	PM4[2]=1	I2C0 SCL
10	PM4[2]=0	GPIO0 B[7]	PM4[2]=1	I2C0 SDA
11	PM4[2]=0	GND	PM4[2]=1	GND
12	PM4[2]=0	3V3	PM4[2]=1	3V3

## 2.25.6 Pmod Interface Type 1 (GPIO)

By default, all Pmod connectors are configured as GPIO interfaces and are compatible with the Pmod Interface Type 1 (GPIO) as specified in [3].

For a Pmod interface type 1 at `Pmodn` use the following settings:

- program the `PMOD_MUX_CTRL` setting  $PMn[0] = 0$ ,  $PMn[2] = 0$   
where `n` equals 0, 1, 2, 3 or 4.

---

## 2.25.7 Pmod Interface Type 2 (SPI)

The connectors `Pmod2` and `Pmod3` can be programmed to operate as SPI interfaces compatible with the Pmod Interface Type 2 (SPI) as specified in [3]. Refer to the “[SPI Interfaces](#)” section for more details.

---

## 2.25.8 Pmod Interface Type 2a (Expanded SPI)

The connectors `Pmod2` and `Pmod3` can be used to implement extended SPI interfaces compatible with the Pmod Interface Type 2a (Expanded SPI) as specified in [3].

For a Pmod interface type 2a at `Pmod2` take the following steps:

- program the `PMOD_MUX_CTRL` setting  $PM2[0] = 1$ ,  $PM2[2] = 0$
- use GPIO A[20] as an interrupt input
- use GPIO A[21] as a reset output.

For a Pmod interface type 2a at `Pmod3` take the following steps:

- program the `PMOD_MUX_CTRL` setting  $PM3[0] = 1$ ,  $PM3[2] = 0$
- use GPIO A[28] as an interrupt input
- use GPIO A[29] as a reset output.

---

## 2.25.9 Pmod Interface Type 3 (UART)

Peripheral modules that comply with the deprecated Pmod interface type 3 (UART) can be connected to the Pmod connectors of the ARC SDP Mainboard using a crossover cable or a flying lead cable (see [3]), if the selected connector is configured as a Pmod Interface Type 4 (UART).

---

## 2.25.10 Pmod Interface Type 4 (UART)

The connectors `Pmod0` and `Pmod2` can be programmed to operate as UART interfaces compatible with the Pmod Interface Type 4 (UART) as specified in [3]. Refer to the “[UART \(RS232\) Interfaces](#)” section for more details.

---

## 2.25.11 Pmod Interface Type 4a (Expanded UART)

The connector Pmod2 can be used to implement a UART interface compatible with the Pmod Extended UART Interface (Type 4a) as specified in [3].

For a Pmod interface type 4a (expanded UART) at Pmod0 take the following steps:

- program the PMOD\_MUX\_CTRL setting PM0[0] = 1, PM0[2] = 0
- use GPIO A[4] as an interrupt input
- use GPIO A[5] as a reset output.

---

## 2.25.12 Pmod I<sup>2</sup>C Interface

The connector Pmod4 can be programmed to operate as I<sup>2</sup>C interface compatible with the Pmod I<sup>2</sup>C Interface as specified in [3]. Refer to the “[I<sup>2</sup>C Interfaces](#)” section for more details.

---

## 2.26 Other Extension Options

The ARC SDP Mainboard features four 10-pin extension connectors Extension0, Extension1, Extension2 and Extension4. The voltage level at the extension connectors can be selected by jumpers as 1V8, 2V5, 3V3, or 5V0. In addition to the I/O signals available at the Pmod connectors, the extension connectors feature reset outputs. The polarity of the reset signal can be individually controlled by the jumpers JP1503, JP1504, JP1505 and JP1506.

The functionality of these extension connectors is programmable. The available options are summarized in Table 47. Multiplexing is controlled by software using the PMOD\_MUX\_CTRL register (see "[Control Registers](#)"). After a reset, all ports are configured as GPIO inputs.

All signals at the extension connectors have pull-up resistors to the selected voltage levels.



**Note** The Pmod0 connector shares its pin-multiplexer with the Extension0 connector. This means that (with the exception of the reset output at Extension0) the connectors Pmod0 and Extension0 provide the same data signals. However, the voltage levels may be different.

Likewise the connectors Pmod1, Pmod2 and Pmod3 share their pin-multiplexers with the connectors Extension1, Extension2 and Extension3 respectively.

---

*Table 47 Extension connector pin multiplexing overview*

Connector		Control Register	Peripherals	
Name	Pins	PMOD_MUX_CTR L Control Bit	MUX option 0	MUX option 1
Extension0	1, 2, 4, 6	PM0 [0]	GPIO0 A[3:0]	UART0
	3, 5, 7, 9	PM0 [2]	GPIO0 A[7:4]	UART1
Extension1	1, 2, 4, 6	PM1 [0]	GPIO0 A[11:8]	GPIO C[3:0]
	3, 5, 7, 9	PM1 [2]	GPIO0 A[15:12]	GPIO C[7:4]
Extension2	1, 2, 4, 6	PM2 [0]	GPIO0 A[19:16]	SPI1
	3, 5, 7, 9	PM2 [2]	GPIO0 A[23:20]	UART0
Extension3	1, 2, 4, 6	PM3 [0]	GPIO0 A[27:24]	SPI0
	3, 5, 7, 9	PM3 [2]	GPIO0 A[31:28]	SPI1

Table 48 and Table 49 describe the signals that are used in the pin description tables of the extension connectors.

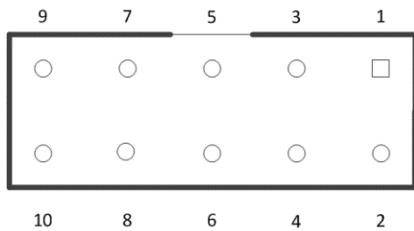
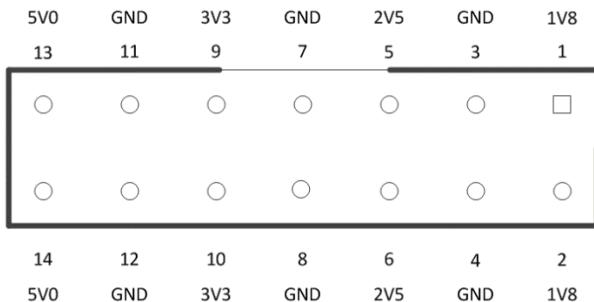
*Table 48 UART signal description*

Name	Description	Direction
CTS_n	Clear to send, active low Handshake signal from external peripheral to on-board host	Input
RTS_n	Ready To Send, active low Handshake signal from on-board host to external peripheral	Output
RXD	Receive data Data from external peripheral to on-board host	Input
TXD	Transmit data Data clock from on-board host to external peripheral	Output

*Table 49 SPI Signal Description*

Name	Description	Direction
SS	Slave Select Active low to enable slave device	Output
MISO	Master Out Slave In Data from master to slave	Output
MOSI	Master In Slave Out Data from slave to master	Input
SCK	Serial clock Data clock from master to slave	Output

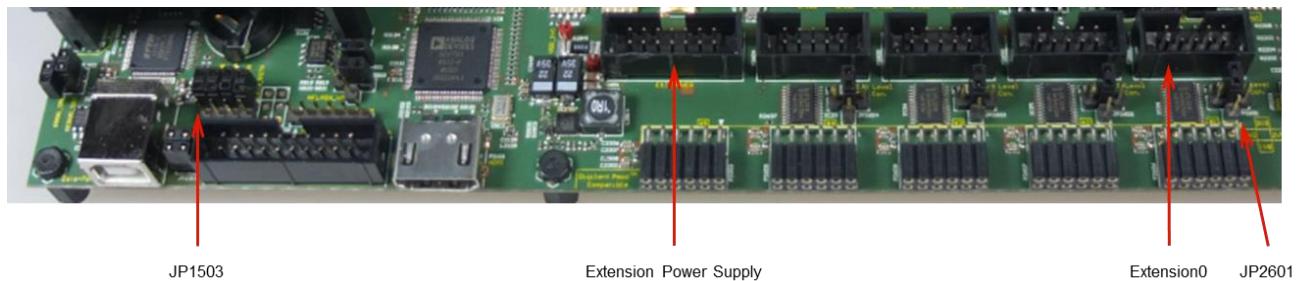
The location of the pins on the Extension connectors is shown in Figure 67. The detailed pin descriptions depending on the pin multiplexer settings are provided in Table 50, Table 51, Table 52, and Table 53.

*Figure 67 Pinout diagram of the Extension0, Extension1, Extension2 and Extension3 connectors**Figure 68 Pinout diagram of the Extension Power Supply connector*

## 2.26.1 Extension0 Connector

Figure 69 shows the location of the Extension0 connector on the ARC SDP Mainboard. The jumpers for setting the voltage levels and the reset polarities at Extension0 are shown

as well. Pin 8 of Extension0 supplies power to the extension board. The Extension Power Supply connector provides ground pins and additional voltage levels. This is useful if your extension board requires multiple voltage levels.



*Figure 69 Location of the Extension0 connector and its associated jumpers*

*Table 50 Pin description of the Extension0 connector*

Pin	Standard Setting	Alternative Setting
1	PM0 [0]=0	GPIO0 A[3]
2	PM0 [0]=0	GPIO0 A[2]
3	PM0 [2]=0	GPIO0 A[4]
4	PM0 [0]=0	GPIO0 A[1]
5	PM0 [2]=0	GPIO0 A[5]
6	PM0 [0]=0	GPIO0 A[0]
7	PM0 [2]=0	GPIO0 A[6]
8	VDD_ext0 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2601	VDD_ext0 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2601
9	PM0 [2]=0	GPIO0 A[7]
10		EXT0_RST / EXT0_RSTn

The reset signal on pin 10 of Extension0 is active high when the jumper JP1503 is in place and active low when this jumper is removed.

The voltage level at Extension0 is selected by jumper JP2601 as shown in Figure 70.

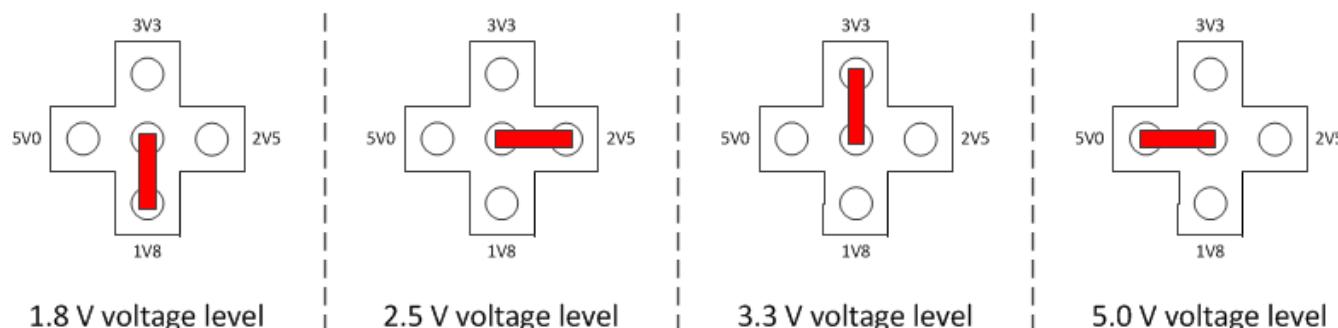


Figure 70 Voltage level selection for the Extension0 connector

## 2.26.2 Extension1 Connector

Figure 71 shows the location of the Extension1 connector on the ARC SDP Mainboard. The jumpers for setting the voltage levels and the reset polarities at Extension1 are shown as well. Pin 8 of Extension1 supplies power to the extension board. The Extension Power Supply connector provides ground pins and additional voltage levels. This is useful if your extension board requires multiple voltage levels.

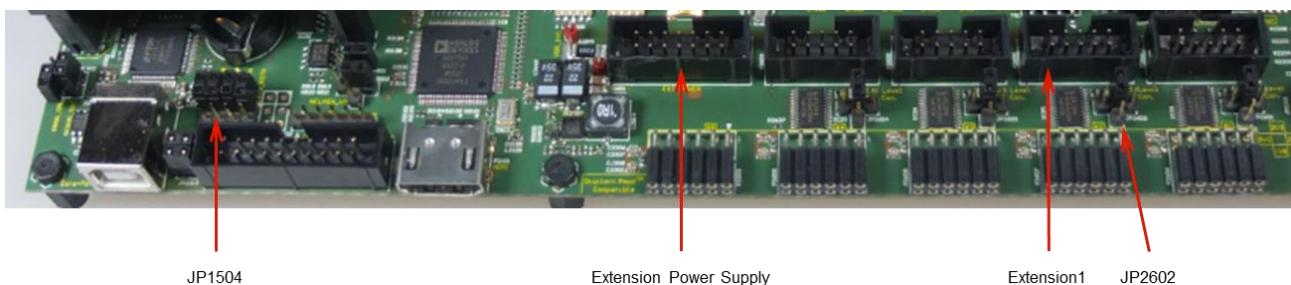


Figure 71 Location of the Extension1 connector and its associated jumpers

Table 51 Pin description of the Extension1 connector

Pin	Standard Setting	Alternative Setting
1	PM1[0]=0 GPIO0 A[11]	PM1[0]=1 GPIO0 C[3]
2	PM1[0]=0 GPIO0 A[10]	PM1[0]=1 GPIO0 C[2]
3	PM1[2]=0 GPIO0 A[12]	PM1[2]=1 GPIO0 C[4]
4	PM1[0]=0 GPIO0 A[9]	PM1[0]=1 GPIO0 C[1]
5	PM1[2]=0 GPIO0 A[13]	PM1[2]=1 GPIO0 C[5]
6	PM1[0]=0 GPIO0 A[8]	PM1[0]=1 GPIO0 C[0]
7	PM1[2]=0 GPIO0 A[14]	PM1[2]=1 GPIO0 C[6]
8	VDD_ext1 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2602	VDD_ext1 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2602
9	PM1[2]=0 GPIO0 A[15]	PM1[2]=1 GPIO0 C[7]
10	EXT1_RST / EXT1_RSTn	EXT1_RST / EXT1_RSTn

The reset signal at pin 10 of Extension1 is active high when the jumper JP1504 is in place and active low when this jumper is removed.

The voltage level on Extension1 is selected by jumper JP2602 as shown in Figure 72.

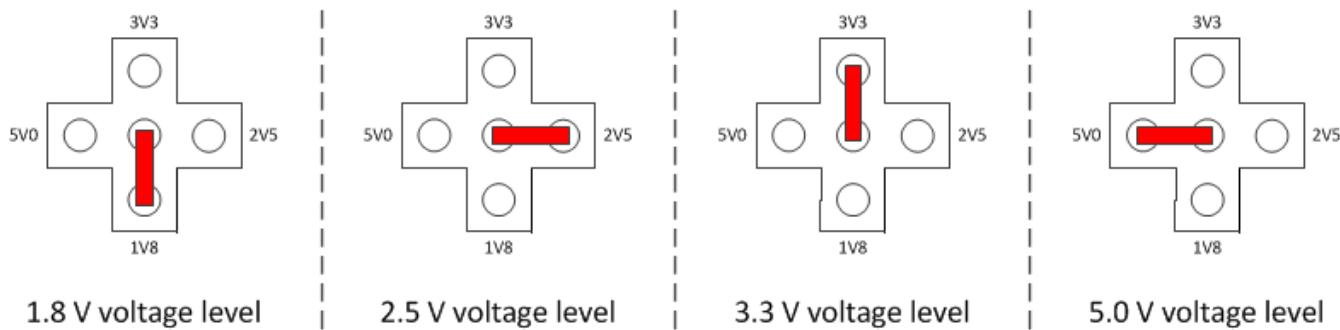


Figure 72 Voltage level selection for the Extension1 connector

### 2.26.3 Extension2 Connector

Figure 73 shows the location of the Extension2 connector on the ARC SDP Mainboard. The jumpers for setting the voltage levels and the reset polarities at Extension2 are shown as well. Pin 8 of Extension2 supplies power to the extension board. The Extension Power Supply connector provides ground pins and additional voltage levels. This is useful if your extension board requires multiple voltage levels.

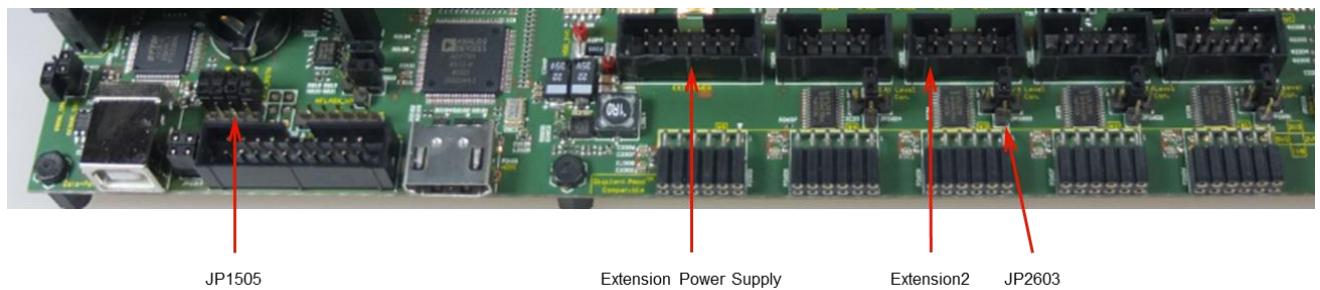


Figure 73 Location of the Extension2 connector and its associated jumpers

Table 52 Pin description of the Extension2 connector

<b>Pin</b>	<b>Standard Setting</b>		<b>Alternative Setting</b>	
1	PM2 [0]=0	GPIO0 A[19]	PM2 [0]=1	SPI1 SCK
2	PM2 [0]=0	GPIO0 A[18]	PM2 [0]=1	SPI1 MISO
3	PM2 [2]=0	GPIO0 A[20]	PM2 [2]=1	UART0 RTS_N
4	PM2 [0]=0	GPIO0 A[17]	PM2 [0]=1	SPI1 MOSI
5	PM2 [2]=0	GPIO0 A[21]	PM2 [2]=1	UART0 TXD
6	PM2 [0]=0	GPIO0 A[16]	PM2 [0]=1	SPI1 SS
7	PM2 [2]=0	GPIO0 A[22]	PM2 [2]=1	UART0 RXD
8		VDD_ext2 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2603		VDD_ext2 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2603
9	PM2 [2]=0	GPIO0 A[23]	PM2 [2]=1	UART0 CTS_N
10		EXT2_RST / EXT2_RSTn		EXT2_RST / EXT2_RSTn

The reset signal on pin 10 of Extension2 is active high when the jumper JP1505 is in place and active low when this jumper is removed.

The voltage level on Extension2 is selected by jumper JP2603 as shown in Figure 74.

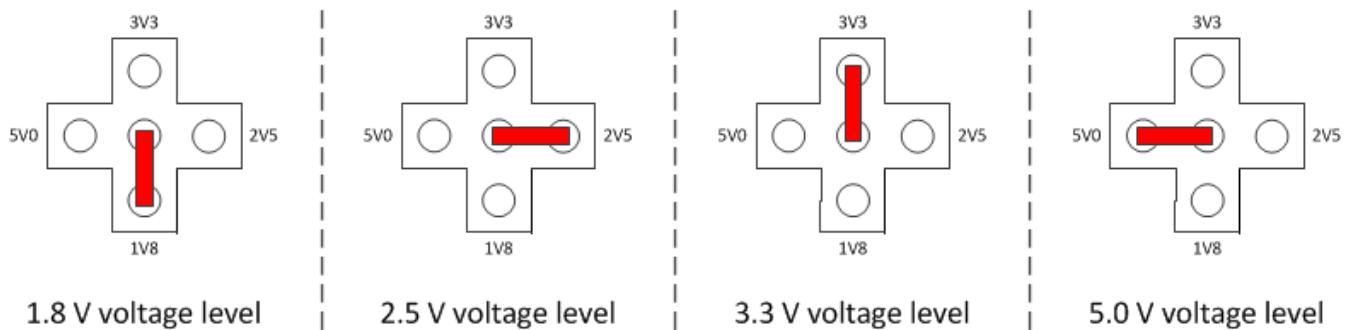


Figure 74 Voltage level selection for the Extension2 connector

## 2.26.4 Extension3 Connector

Figure 75 shows the location of the Extension3 connector on the ARC SDP Mainboard. The jumpers for setting the voltage levels and the reset polarities at Extension3 are shown as well. Pin 8 of Extension3 supplies power to the extension board. The Extension Power Supply connector provides ground pins and additional voltage levels. This is useful if your extension board requires multiple voltage levels.

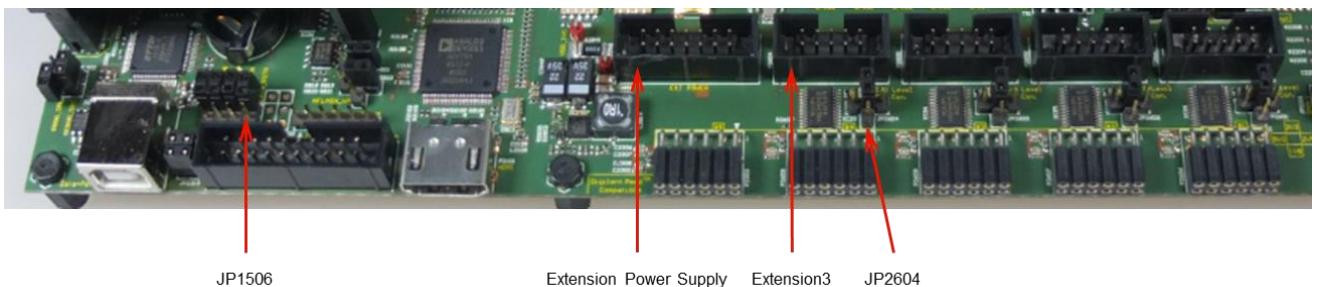


Figure 75 Location of the Extension3 connector and its associated jumpers

Table 53 Pin description of the Extension3 connector

Pin	Standard Setting	Alternative Setting
1	PM3[0]=0 GPIO0 A[27]	PM3[0]=1 SPI0 SCK
2	PM3[0]=0 GPIO0 A[26]	PM3[0]=1 SPI0 MISO
3	PM3[2]=0 GPIO0 A[28]	PM3[2]=1 SPI1 SS
4	PM3[0]=0 GPIO0 A[25]	PM3[0]=1 SPI0 MOSI
5	PM3[2]=0 GPIO0 A[29]	PM3[2]=1 SPI1 MOSI
6	PM3[0]=0 GPIO0 A[24]	PM3[0]=1 SPI0 SS
7	PM3[2]=0 GPIO0 A[30]	PM3[2]=1 SPI1 MISO
8	VDD_ext3 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2604	VDD_ext3 Selected voltage level 1V8, 2V5, 3V3 or 5V0 according to JP2604
9	PM3[2]=0 GPIO0 A[31]	PM3[2]=1 SPI1 SCK
10	EXT3_RST / EXT3_RSTn	EXT3_RST / EXT3_RSTn

The reset signal on pin 10 of Extension3 is active high when the jumper JP1506 is in place and active low when this jumper is removed.

The voltage level on Extension3 is selected by jumper JP2604 as shown in Figure 76.

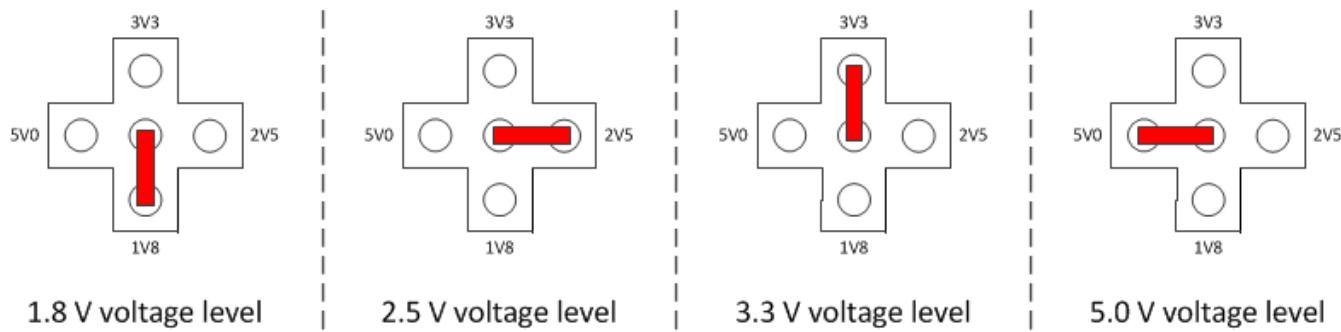


Figure 76 Voltage level selection for the Extension3 connector

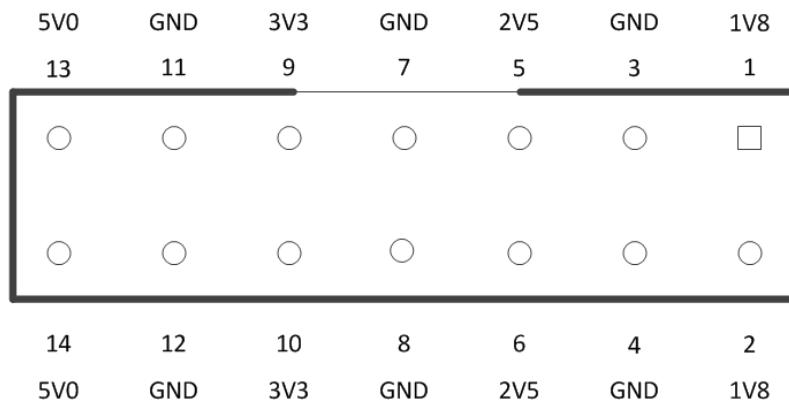
## 2.27 Power Supply for Extension Boards

According to the Pmod Specification [3] the Pmod connectors have dedicated power and ground pins. They operate at a fixed voltage level of 3V3.

The extension ports with variable voltage level (Extension0 to Extension3) do have a VDD pin but no GND pin. The VDD pin provides power to the extension board at the voltage level selected for the respective connector. A separate power supply connector Extension Power Supply is available, which shall be used to connect the extension module to GND. Additionally, it provides additional VDD pins with different voltage levels, which are useful if your extension module requires multiple voltage levels. A pin description of this interface is provided in Table 54 and the pinout diagram of the connector is shown in Figure 77.

*Table 54 Pin description of the extension power supply connector*

Pin	Name	Description	Direction
1, 2	VDDD_1V8	1.8 Volt power supply pin	
3, 4	GND	Ground supply pin	
5, 6	VDDD_2V5	2.5 Volt power supply pin	
7, 8	GND	Ground supply pin	
9, 10	VDDD_3V3	3.3 Volt power supply pin	
11, 12	GND	Ground supply pin	
13, 14	VDDD_5V0	5.0 Volt power supply pin	



*Figure 77 Pinout diagram of the Extension Power Supply connector*

## 2.28 Mounting an ARC CPU Card

Switch the power off before mounting or removing an ARC CPU Card. The CPU Card is plugged into the two HapsTrak-3 CPU Card Connectors on the left side of the board and into the CPU Card Power Supply connector.

---

## 2.29 Backup Battery

The ARC SDP Mainboard includes a battery, which backs the real time clock. See section “[Appendix G](#)” for additional information.

# System Memory Map

*This chapter provides information about the system memory map.*

## 3.1 Controlling the Memory Map

Control registers are available for each AXI master to customize its memory map. Table 55 lists the available AXI masters.

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

Table 55 AXI Masters

Master Number	AXI Master
0	TUNNEL0 (AXI tunnel to/from ARC CPU Card)
1	TUNNEL1 (AXI tunnel to/from HAPS System)
2	GMAC (Ethernet controller)
3	PGU (HDMI)
4	NAND Controller
5	SDIO
6	DMAC0 (DMAC port 0)
7	DMAC1 (DMAC port 1)
8	Reserved
9	USB-EHCI
10	USB-OHCI

For each 256 MByte aperture within the AXI address space of an AXI master the address map configuration consists of two steps:

First, a target slave is selected from the list shown in Table 56. In a second step 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 offset specified refers to the address offset within the target slave only, i.e. the base address of the aperture is not taken into account.

Table 56 Target slaves

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

Table 57 lists the registers available for selecting the target slave and the address offset for each master.

Table 57 Slave Select and Address Offset Registers

Master	Slave Selection Registers	Address Offset Registers
TUNNEL0	AXI_0_SLV_SEL0 AXI_0_SLV_SEL1	AXI_0_SLV_OFFSET0 AXI_0_SLV_OFFSET1
TUNNEL1	AXI_1_SLV_SEL0 AXI_1_SLV_SEL1	AXI_1_SLV_OFFSET0 AXI_1_SLV_OFFSET1
GMAC	AXI_2_SLV_SEL0 AXI_2_SLV_SEL1	AXI_2_SLV_OFFSET0 AXI_2_SLV_OFFSET1
PGU	AXI_3_SLV_SEL0 AXI_3_SLV_SEL1	AXI_3_SLV_OFFSET0 AXI_3_SLV_OFFSET1
NAND	AXI_4_SLV_SEL0 AXI_4_SLV_SEL1	AXI_4_SLV_OFFSET0 AXI_4_SLV_OFFSET1
SDIO	AXI_5_SLV_SEL0 AXI_5_SLV_SEL1	AXI_5_SLV_OFFSET0 AXI_5_SLV_OFFSET1
DMAC0	AXI_6_SLV_SEL0 AXI_6_SLV_SEL1	AXI_6_SLV_OFFSET0 AXI_6_SLV_OFFSET1
DMAC1	AXI_7_SLV_SEL0 AXI_7_SLV_SEL1	AXI_7_SLV_OFFSET0 AXI_7_SLV_OFFSET1
USB-EHCI	AXI_9_SLV_SEL0 AXI_9_SLV_SEL1	AXI_9_SLV_OFFSET0 AXI_9_SLV_OFFSET1
USB-OHCI	AXI_10_SLV_SEL0 AXI_10_SLV_SEL1	AXI_10_SLV_OFFSET0 AXI_10_SLV_OFFSET1

These registers are double-buffered, which means that writing to these registers does not take effect immediately. The address decoder starts using the newly-programmed values only after the corresponding control bit in the `AXI_UPDATE` register has been set to 1.

### **Example 1:**

If the offset for aperture[3] is programmed to 0 MByte, then the master's AXI aperture ranging from `0x3000_0000` to `0x3FFF_FFFF` is mapped to the address range `0x0000_0000` to `0x0FFF_FFFF` of the selected target slave.

### **Example 2:**

If the offset for aperture[3] is programmed to 256 MByte, then the master's AXI aperture ranging from `0x3000_0000` to `0x3FFF_FFFF` is mapped to the address range `0x1000_0000` to `0x1FFF_FFFF` of the selected target slave.

### **Example 3:**

If a target slave has an address aperture wider than 256 Mbytes (such as the TUNNEL0), then this memory should be selected as the target slave for multiple address apertures, where each aperture is assigned a different address offset. For example aperture[1], aperture[2], aperture[3] and aperture[4] could all be programmed to select TUNNEL0. If the address offsets are programmed as shown in Table 58, these settings make a contiguous range of 1 GByte of the ARC CPU Card memory map visible in the AXI master's address range from `0x1000_0000` to `0x4FFF_FFFF`.

*Table 58 Address Offset Settings for Example 3*

<b>Aperture</b>	<b>Slave Address Offset</b>	<b>SLV_OFFSET used in register</b>
aperture[1]	0 MByte	0
aperture[2]	256 MByte	1
aperture[3]	512 MByte	2
aperture[4]	768 MByte	3

## 3.2 Memory Map After Reset

The memory map of the ARC SDP Mainboard after reset is listed in Table 59. This memory map is valid for all AXI masters on the Mainboard. This memory map may be changed by the pre-boot loader of your ARC CPU Card or the initialization sequence of an operating system running on a core of that CPU Card. Refer to the documentation of your CPU Card for details.

Table 59 Memory map after reset

Aperture #	Master Address	SLV_SEL	SLV_OFFSET	Selected Slave	Slave Address
15	0xFFFF_FFFF 0xF000_0000	1	0xF	TUNNEL0 (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 (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 RAM has a size of 256 kByte. The lower 256 kByte within the master's 256 MByte aperture access the RAM, the remaining 261888 kByte are not accessible.



**Note** Aperture #0 and aperture #1 both access the entire RAM. For example, the master addresses `0x0000_0000` and `0x1000_0000` both access the RAM address `0x0000_0000`.

The slave address of the AXI TUNNEL0 slave on the Mainboard, as listed in Table 59, is transparently forwarded via the AXI Tunnel. Following this, it becomes the address issued by the AXI Tunnel master on the ARC CPU Card. It is then decoded according to the memory map programmed for the AXI Tunnel master on the CPU Card.

Likewise, the slave address of the AXI TUNNEL1 slave on the Mainboard is forwarded to the AXI Tunnel master on the HAPS system. It is then decoded according to your custom design. The memory map can be modified for each master individually by altering the settings of the corresponding control registers. Refer to “[Controlling the Memory Map](#)” and “[Control Registers](#)” for more detailed information. Refer to the documentation of your CPU Card for additional information.

### 3.3 Memory Map of Peripheral Subsystem

All peripherals of the peripheral subsystem are mapped into the Control/Status section of the system memory map, which has the default base address `0xE000_0000`. Table 60 shows the address offsets of the individual peripherals and the corresponding aperture within the Control/Status section. This means that the address offset listed in Table 60 has to be added to the base address of the Control/Status section to obtain the correct base address of the peripheral.

Example: If the Control/Status section is located at its default address `0xE000_0000`, then the CGU base address within the address space of the master is `0xE001_0000`.

*Table 60 Peripheral memory map*

Peripheral	Address Offset	Aperture (bytes)	Description
SPI_MEM	<code>0x0000_0000</code>	256	SPI Master for Flash memory
CGU	<code>0x0001_0000</code>	4096	Clock Generator Unit
CREG	<code>0x0001_1000</code>	4096	Control Registers
ICTL	<code>0x0001_2000</code>	512	Interrupt Controller
GPIO0	<code>0x0001_3000</code>	128	General Purpose I/O for Pmod and Extension connectors
GPIO1	<code>0x0001_4000</code>	128	General Purpose I/O for push buttons, switches and LEDs

<b>Peripheral</b>	<b>Address Offset</b>	<b>Aperture (bytes)</b>	<b>Description</b>
SDIO	0x0001_5000	1024	SDIO Controller for SD-Card
NAND	0x0001_6000	4096	NAND flash controller
PGU	0x0001_7000	1024	Pixel Graphics Unit for HDMI Transmitter
Ethernet	0x0001_8000	4096	Ethernet MAC Control and status registers
	0x0001_9000	4096	Ethernet MAC DMA registers
SPI0	0x0001_A000	256	SPI0 Pmod interface
SPI1	0x0001_B000	256	SPI1 Pmod Interface
SPI2	0x0001_C000	256	SPI for seven-segment display
I2C0	0x0001_D000	256	I2C0 interface Pmod connector
I2S	0x0001_E000	256	I2S interface to HDMI
I2C2	0x0001_F000	256	On-board I <sup>2</sup> C bus
UART0	0x0002_0000	256	UART0 DB9 or Pmod connector
UART1	0x0002_1000	256	UART1 6-pin header or Pmod connector
UART2	0x0002_2000	256	UART connected to USB Dataport
USB Host	0x0004_0000	131072 (128K)	USB-HOST EHCI control/status
	0x0006_0000	131072 (128K)	USB-HOST OHCI control/status
DMAC	0x0008_0000	1024	DMA Controller control/status

## Software Interfaces

*This chapter provides information about the software interfaces of the peripheral subsystem on the ARC SDP Mainboard.*

Table 61 Peripheral registers overview

Name	Address Offset	R/W	Description
<b>Clock Generation Registers</b>			
TUN_PLL_IDIV	0x0001_0020	RW	Tunnel Input divider register
TUN_PLL_FBDIV	0x0001_0024	RW	Tunnel Feedback divider register
TUN_PLL_ODIV0	0x0001_0028	RW	Tunnel Output divider register
TUN_PLL_ODIV1	0x0001_002C	RW	Tunnel Output divider register
TUN_PLL_LOCK	0x0001_0104	R	Tunnel PLL Lock register
PGU_PLL_IDIV	0x0001_0080	RW	PGU Input divider register
PGU_PLL_FBDIV	0x0001_0084	RW	PGU Feedback divider register
PGU_PLL_ODIV0	0x0001_0088	RW	PGU Output divider register
PGU_PLL_LOCK	0x0001_0110	R	PGU PLL Lock register
AUDIO_I2S_SCLK_DIV	0x0001_00A8	RW	Audio I2S sclk divider register
AUDIO_I2S_MCLK_DIV	0x0001_00AC	RW	Audio I2S mclk divider register
<b>Control Registers</b>			
AXI_m_SLV_SEL0	0x0001_1000 + m*0x10	RW	Address decoder slave select register for AXI master [m]
AXI_m_SLV_SEL1	0x0001_1004 + m*0x10	RW	Address decoder slave select register for AXI master [m]
AXI_m_SLV_OFFSET0	0x0001_1008 + m*0x10	RW	Address decoder offset register for AXI master [m]
AXI_m_SLV_OFFSET0	0x0001_100C + m*0x10	RW	Address decoder offset register for AXI master [m]

Name	Address Offset	R/W	Description
AXI_UPDATE	0x0001_1100	RW1C	Address decoder update register
AXI_UPDATE_CLR	0x0001_1104	RW1C	Address decoder update clear register
AXI_UPDATE_STAT	0x0001_1108	R	Address decoder update status register
TUN_CTRL	0x0001_1200	RW	AXI tunnel control register
TUN_STAT	0x0001_1204	R	AXI tunnel status register
PMOD_MUX_CTRL	0x0001_1210	RW	Pmod mux register
AUDIO_CLK_MUX_CTRL	0x0001_1214	RW	Audio clock mux register
SPI_FLASH_MUX_CTRL	0x0001_1218	RW	SPI_FLASH mux register
SW_RESET	0x0001_1108	RW1C	SW reset register for resetting HW modules on the Mainboard like e.g. Ethernet PHY

## 4.1 Clock Generation Registers

### 4.1.1 TUNNEL PLL

Reference input clock for Tunnel PLL is 25Mhz

Minimum input clock frequency is 19MHz

VCO range for Tunnel PLL is 800 - 1866MHz

#### 4.1.1.1 TUN\_PLL\_IDIV Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		NOUPDATE	BYPASS	EDGE	HIGHTIME		LOWTIME									

Address offset: 0x0001\_0020

Reset Value: 0x0000\_2041 (0x0000\_3001 after pre-boot)

Access: RW

Register to control setting of the Tunnel PLL input divider

LOWTIME[5:0] sets the amount of time in input cycles that the divided input clock remains low

HIGHTIME[5:0] sets the amount of time in input cycles that the divided input clock remains high

$$\text{IDIV} = \text{LOWTIME} + \text{HIGHTIME}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypass the input divider

NOUPDATE prevent update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider shall be programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.1.2 TUN\_PLL\_FBDIV Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NOUPDATE	BYPASS	EDGE	HIGHTIME			LOWTIME							

Address offset: 0x0001\_0024

Reset Value: 0x0000\_0492

Access: RW

Register to control setting of the Tunnel PLL feedback divider

LOWTIME[5:0] sets the amount of time in VCO cycles that the feedback clock remains low

HIGHTIME[5:0] sets the amount of time in VCO cycles that the feedback clock remains high

$$\text{FBDIV} = \text{LOWTIME} + \text{HIGHTIME}$$

$$\text{VCOFREQ} = (25\text{MHz} / \text{IDIV}) * \text{FBDIV}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypass the feedback divider

NOUPDATE prevent update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider shall be programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.1.3 TUN\_PLL\_ODIV0 Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			NOUPDATE			BYPASS			EDGE			HIGHTIME			LOWTIME		

Address offset: 0x0001\_0028

Reset Value: 0x0000\_0492 (0x0000\_0186 after pre-boot)

Access: RW

Register for controlling the clock setting for the ARC CPU Card tunnel.

LOWTIME[5:0] sets the amount of time in VCO cycles that the output clock remains low

HIGHTIME[5:0] sets the amount of time in VCO cycles that the output clock remains high

$$\text{ODIV} = \text{LOWTIME} + \text{HIGHTIME}$$

$$\text{OFREQ} = \text{VCOFREQ} / \text{ODIV}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypass the output divider

NOUPDATE prevents update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider shall be programmed as follows:

- even divider ratio =>  $\text{LOWTIME} = \text{HIGHTIME}$   
 $\text{EDGE} = 0$
- odd divider ratio =>  $\text{LOWTIME} = \text{HIGHTIME} + 1$   
 $\text{EDGE} = 1$

#### 4.1.1.4 TUN\_PLL\_ODIV1 Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		NOUPDATE	BYPASS	EDGE	HIGHTIME		LOWTIME									

Address offset: 0x0001\_002C

Reset Value: 0x0000\_0492

Access: RW

Register to control setting of the clock for the tunnel to the HAPS System

LOWTIME[5:0] sets the amount of time in VCO cycles that the output clock remains low

HIGHTIME[5:0] sets the amount of time in VCO cycles that the output clock remains high

$$\text{ODIV} = \text{LOWTIME} + \text{HIGHTIME}$$

$$\text{OFREQ} = \text{VCOFREQ} / \text{ODIV}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypass the output divider

NOUPDATE prevent update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider shall be programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.1.5 TUN\_PLL\_LOCK Register

31	2	1	0
Reserved		ERROR	LOCK

Address offset: 0x0104

Reset Value: 0x0000\_0001

Access: R

Register for Tunnel PLL lock status

LOCK PLL lock indication

0 = PLL is unlocked

1 = PLL is locked

ERROR PLL error indication. Asserted high to indicate that PLL was programmed with an illegal value. PLL can be re-programmed once the ERROR status bit is reset back to '0'

## 4.1.2 PGU PLL

Reference input clock for PGU PLL is 27Mhz

VCO range for Tunnel PGU is 800 - 1866MHz

### 4.1.2.1 PGU\_PLL\_IDIV Register

31		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			NOUPDATE	BYPASS	EDGE		HIGHTIME		LOWTIME							

Address offset: 0x0001\_0080

Reset Value: 0x0000\_2041

Access: RW

Register to control setting of the PGU PLL input divider

LOWTIME[5:0] sets the amount of time in input cycles that the divided input clock remains low

HIGHTIME[5:0] sets the amount of time in input cycles that the divided input clock remains high

$$\text{IDIV} = \text{LOWTIME} + \text{HIGHTIME}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypass the input divider

NOUPDATE prevent update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle, the divider is programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.2.2 PGU\_PLL\_FBDIV Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		NOUPDATE	BYPASS	EDGE	HIGHTIME		LOWTIME									

Address offset: 0x0001\_0084

Reset Value: 0x0000\_0596

Access: RW

Register to control setting of the PGU PLL feedback divider

LOWTIME[5:0] sets the amount of time in VCO cycles that the feedback clock remains low

HIGHTIME[5:0] sets the amount of time in VCO cycles that the feedback clock remains high  
FBDIV = LOWTIME + HIGHTIME

$$\text{VCOFREQ} = (27\text{MHz} / \text{IDIV}) * \text{FBDIV}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypasses the feedback divider

NOUPDATE prevents update of the PLL with new settings. Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider is programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.2.3 PGU\_PLL\_ODIV Register

31	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		NOUPDATE		BYPASS		EDGE		HIGHTIME		LOWTIME							

Address offset: 0x0001\_0088

Reset Value: 0x0001\_0208

Access: RW

Register to control setting of the PGU reference clock

LOWTIME[5:0] sets the amount of time in VCO cycles that the output clock remains low

HIGHTIME[5:0] sets the amount of time in VCO cycles that the output clock remains high

$$\text{ODIV} = \text{LOWTIME} + \text{HIGHTIME}$$

$$\text{OFREQ} = \text{VCOFREQ} / \text{ODIV}$$

EDGE chooses the edge that the High Time counter transitions on (0=rising, 1=falling)

BYPASS bypasses the feedback divider

NOUPDATE prevents update of the PLL with new settings.  
Debug only; can be used for register RW test

To obtain a 50% duty-cycle the divider is programmed as follows:

- even divider ratio => LOWTIME = HIGHTIME  
EDGE = 0
- odd divider ratio => LOWTIME = HIGHTIME + 1  
EDGE = 1

#### 4.1.2.4 PGU\_PLL\_LOCK Register

31	2	1	0
Reserved		ERROR	LOCK

Address offset: 0x0110

Reset Value: 0x0000\_0001

Access: R

Register for PGU PLL lock status

LOCK PLL lock indication

0 = PLL is unlocked

1 = PLL is locked

ERROR PLL error indication. Asserted high to indicate that the PLL was programmed with an illegal value. The PLL can be re-programmed after the ERROR status bit is reset to 0.

### 4.1.3 Audio I2S Clock

A PLL inside the clock generation unit (CGU) generates a fixed audio reference clock of 12.288 or 28.224MHz. The desired audio reference clock can be selected by software using the `AUDIO_CLK_MUX_CTRL` register (see "Control Registers").

The I2S serial and oversampling clock (sclk and mclk) are generated by integer dividers inside the CGU. Table 62 lists the example sclk divider settings for commonly used audio reference clock frequencies.

*Table 62 I2S sclk Divider Settings*

Audio Reference Clock (MHz)	Audio Sample Rate					
	16 kHz	32 kHz	44.1 kHz	48 kHz	96 kHz	192 kHz
12.288	768/32/2	384/32/2		256/32/2	128/32/2	64/32/2
28.224	1764/32/2	882/32/2	640/32/2	588/32/2	294/32/2	147/32/2

#### 4.1.3.1 AUDIO\_I2S\_SCLK\_DIV Register

31	24	23	16	15	0
Reserved		SCLK_DIV[7:0]		Reserved	

Address offset: 0x0001\_00A8

Reset Value: 0x0001\_0000

Access: RW

Register to control setting of the I2S sclk

SCLK\_DIV[7:0] sclk divider setting (0=disabled, 1=div-by-1, 2=div-by-2, ....)

- sclk = audio\_ref\_clk / SCLK\_DIV

#### 4.1.3.2 AUDIO\_I2S\_MCLK\_DIV Register

31	24	23	16	15	0
Reserved		MCLK_DIV[7:0]		Reserved	

Address offset: 0x0001\_00AC

Reset Value: 0x0001\_0000

Access: RW

Register to control setting of the I2S mclk (0=disabled, 1=div-by-1, 2=div-by-2, ....)

MCLK\_DIV[7:0] mclk divider setting

$$\text{- mclk} = \text{audio\_ref\_clk} / \text{MCLK\_DIV}$$

## 4.2 Control Registers

### 4.2.1 AXI\_m\_SLV\_SEL0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SLVSEL7	SLVSEL6		SLVSEL5	SLVSEL4	SLVSEL3	SLVSEL2	SLVSEL1	SLVSEL0																								

Address offset: 0x0001\_1000 +  $m$  \* 0x010

Reset Value: 0x0000\_0033

Access: RW

SLVSEL $x$ [3:0] selects target slave for address aperture  $x$  for AXI master  $m$

SLVSEL $x$  [3:0] = 0 no slave selected

SLVSEL $x$  [3:0] = 1 AXI tunnel to/from ARC CPU Card

SLVSEL $x$  [3:0] = 2 AXI tunnel to/from HAPS System

SLVSEL $x$  [3:0] = 3 SRAM controller (Mainboard RAM)

SLVSEL $x$  [3:0] = 4 control/status peripherals

AXI masters:

$m$  = 0 TUNNEL0 (AXI tunnel to/from ARC CPU Card)

$m$  = 1 TUNNEL1 (AXI tunnel to/from HAPS System)

$m$  = 2 GMAC (Ethernet controller)

$m$  = 3 PGU (HDMI)

$m$  = 4 NAND

$m$  = 5 SDIO

$m$  = 6 DMAC0 (DMAC port 0)

$m$  = 7 DMAC1 (DMAC port 1)

$m$  = 8 Reserved

$m$  = 9 USB-EHCI

$m$  = 10 USB-OHCI

## 4.2.2 AXI\_ $m$ \_SLV\_SEL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SLVSEL15	SLVSEL14	SLVSEL13	SLVSEL12	SLVSEL11	SLVSEL10	SLVSEL9	SLVSEL8																										

Address offset: 0x0001\_1004 +  $m$  \* 0x010

Reset Value: 0x1420\_1111

Access: RW

SLVSEL $x$ [3:0] selects target slave for address aperture  $x$  for AXI master  $m$

SLVSEL $x$  [3:0] = 0 no slave selected

SLVSEL $x$  [3:0] = 1 AXI tunnel to/from ARC CPU Card

SLVSEL $x$  [3:0] = 2 AXI tunnel to/from HAPS System

SLVSEL $x$  [3:0] = 3 SRAM controller (Mainboard RAM)

SLVSEL $x$  [3:0] = 4 control/status peripherals

AXI masters:

$m$  = 0 TUNNEL0 (AXI tunnel to/from ARC CPU Card)

$m$  = 1 TUNNEL1 (AXI tunnel to/from HAPS System)

$m$  = 2 GMAC (Ethernet controller)

$m$  = 3 PGU (HDMI)

$m$  = 4 NAND

$m$  = 5 SDIO

$m$  = 6 DMAC0 (DMAC port 0)

$m$  = 7 DMAC1 (DMAC port 1)

$m$  = 8 Reserved

$m$  = 9 USB-EHCI

$m$  = 10 USB-OHCI

### 4.2.3 AXI\_ *m*\_SLV\_OFFSET0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OFFSET7	OFFSET6	OFFSET5	OFFSET4	OFFSET3	OFFSET2	OFFSET1	OFFSET0																									

Address offset: 0x0001\_1008 + *m* \* 0x010

Reset Value: 0x0000\_0000

Access: RW

OFFSET*x*[3:0] selects offset for address aperture *x* for AXI master *m*

OFFSET*x* [3:0] = 0 → 0\*256MB

OFFSET*x* [3:0] = 1 → 1\*256MB

....

OFFSET*x* [3:0] = 15 → 15\*256MB

AXI masters:

*m* = 0 TUNNEL0 (AXI tunnel to/from ARC CPU Card)

*m* = 1 TUNNEL1 (AXI tunnel to/from HAPS System)

*m* = 2 GMAC (Ethernet controller)

*m* = 3 PGU (HDMI)

*m* = 4 NAND

*m* = 5 SDIO

*m* = 6 DMAC0 (DMAC port 0)

*m* = 7 DMAC1 (DMAC port 1)

*m* = 8 Reserved

*m* = 9 USB-EHCI

*m* = 10 USB-OHCI

#### 4.2.4 AXI\_ $m$ \_SLV\_OFFSET1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OFFSET15	OFFSET14	OFFSET13	OFFSET12	OFFSET11	OFFSET10	OFFSET9	OFFSET8																									

Address offset: 0x0001\_100C +  $m$  \* 0x010

Reset Value: 0xF0D0\_BA98

Access: RW

OFFSET $x$ [3:0] selects the offset for address aperture  $x$  for AXI master  $m$

OFFSET $x$  [3:0] = 0 → 0\*256MB

OFFSET $x$  [3:0] = 1 → 1\*256MB

....

OFFSET $x$  [3:0] = 15 → 15\*256MB

AXI masters:

$m$  = 0 TUNNEL0 (AXI tunnel to/from ARC CPU Card)

$m$  = 1 TUNNEL1 (AXI tunnel to/from HAPS System)

$m$  = 2 GMAC (Ethernet)

$m$  = 3 PGU (HDMI)

$m$  = 4 NAND

$m$  = 5 SDIO

$m$  = 6 DMAC0 (DMAC port 0)

$m$  = 7 DMAC1 (DMAC port 1)

$m$  = 8 Reserved

$m$  = 9 USB-EHCI

$m$  = 10 USB-OHCI

## 4.2.5 AXI\_UPDATE Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																UPDATE[10:0]															

Address offset: 0x0001\_1100

Reset Value: 0x000

Access: RW1C

All address aperture configuration registers (i.e. \*\_SLV\_SEL and \*\_SLV\_OFFSET) are double-buffered. The newly programmed values for a master will only be forwarded to the address decoder after writing a '1' to the associated UPDATE bit (i.e. bit[0] for master[0], bit[1] for master[1] etc.). Writing a '1' to one or multiple UPDATE bit(s) asserts the update signal towards the corresponding address decoder(s), sets the corresponding bit(s) of the AXI\_UPDATE\_STAT register and raises an interrupt request to the ICTL.

## 4.2.6 AXI\_UPDATE\_CLR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																UPDATE_CLR[10:0]															

Address offset: 0x0001\_1104

Reset Value: 0x000

Access: RW1C

Writing a '1' to one or multiple UPDATE\_CLR[m] clears the update signal to the corresponding address decoder(s), clears the corresponding bit(s) of the AXI\_UPDATE\_STAT register and clears the interrupt request to the ICTL.

#### 4.2.7 AXI\_UPDATE\_STAT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																UPDATE_STAT[10:0]															

Address offset: 0x0001\_1108

Reset Value: 0x000

Access: R

UPDATE\_STAT[x] reflects the status of the update signal to the associated address decoder. Writing a '1' to one or multiple bits of the AXI\_UPDATE register sets the corresponding bit(s) of the AXI\_UPDATE\_STAT register. Writing a '1' to one or multiple bits of the AXI\_UPDATE\_CLR register clears the corresponding bit(s) of the AXI\_UPDATE\_STAT register. The interrupt service routine can use this register to determine the root cause of the interrupt (i.e. to find out which address decoders have just been modified).

#### 4.2.8 TUN\_CTRL Register

31	9	8	7	6	5	4	3	2	1	0
Reserved		LEGACY		Reserved		PRIO1[1:0]		Reserved		PRIO0[1:0]

Address offset: 0x0001\_1200

Reset Value: 0x0000\_0100

Access: RW

PRIO<sub>x</sub>[1:0] controls the priority setting for the tunnel arbitration

PRIO<sub>x</sub> = 0 → axi master and slave have equal priority (round-robin)

PRIO<sub>x</sub> = 1 → axi master has the highest priority

PRIO<sub>x</sub> = 2 → axi slave has the highest priority

PRIO<sub>x</sub> = 3] → illegal

AXI tunnels

<sub>x</sub> = 0 AXI tunnel to/from ARC CPU Card

<sub>x</sub> = 1 AXI tunnel to/from HAPS System

## 4.2.9 TUN\_STAT Register

31	8	7	4	3	0
Reserved		STAT1[3:0]		STAT0[3:0]	

Address offset: 0x0001\_1204

Reset Value: 0x00

Access: R

STAT<sub>X</sub>[3:0] reflects the status of the tunnel after reset completion

STAT <sub>X</sub> [0]	→ initialization sequence done	(1=done, 0=not yet done)
STAT <sub>X</sub> [1]	→ initialization sequence error	(1=error, 0=no error)
STAT <sub>X</sub> [2]	→ bist sequence done	(1=done, 0=not yet done)
STAT <sub>X</sub> [3]	→ bist sequence error	(1=error, 0=no error)

AXI tunnels

<sub>X</sub> = 0 AXI tunnel to/from ARC CPU Card

<sub>X</sub> = 1 AXI tunnel to/from HAPS System

## 4.2.10 PMOD\_MUX\_CTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					PM4[3:0]		PM3[3:0]		PM2[3:0]		PM1[3:0]		PM0[3:0]																		

Address offset: 0x0001\_1210

Reset Value: 0x0000\_0000

Access: RW

The fields PM<sub>X</sub>[3:0] control the multiplexing of the Pmod and Extension connectors:

PM0[1:0] = 0 Pmod0 upper row and Extension0 pins 1,2,4,6 → GPIO0 A[3:0]

PM0[1:0] = 1 Pmod0 upper row and Extension0 pins 3,5,7,9 → UART0

PM0[1:0] > 1 reserved

PM0[3:2] = 0 Pmod0 lower row and Extension0 pins 3,5,7,9 → GPIO0 A [7:4]

PM0[3:2] = 1 Pmod0 lower row and Extension0 pins 3,5,7,9 → UART1

PM0[3:2] > 1 reserved

PM1[1:0] = 0 Pmod1 upper row and Extension1 pins 1,2,4,6 → GPIO0 A[11:8]

PM1[1:0] = 1	Pmod1 upper row and Extension1 pins 3,5,7,9 → GPIO0 C [3:0]
PM1[1:0] > 1	reserved
PM1[3:2] = 0	Pmod1 lower row and Extension1 pins 3,5,7,9 → GPIO0 A [15:12]
PM1[3:2] = 1	Pmod1 lower row and Extension1 pins 3,5,7,9 → GPIO0 C [7:4]
PM1[3:2] > 1	reserved
PM2[1:0] = 0	Pmod2 upper row and Extension2 pins 1,2,4,6 → GPIO0 A[19:16]
PM2[1:0] = 1	Pmod2 upper row and Extension2 pins 3,5,7,9 → SPI1
PM2[1:0] > 1	reserved
PM2[3:2] = 0	Pmod2 lower row and Extension2 pins 3,5,7,9 → GPIO0 A [23:20]
PM2[3:2] = 1	Pmod2 lower row and Extension2 pins 3,5,7,9 → UART0
PM2[3:2] > 1	reserved
PM3[1:0] = 0	Pmod3 upper row and Extension3 pins 1,2,4,6 → GPIO0 A[27:24]
PM3[1:0] = 1	Pmod3 upper row and Extension3 pins 3,5,7,9 → SPI0
PM3[1:0] > 1	reserved
PM3[3:2] = 0	Pmod3 lower row and Extension3 pins 3,5,7,9 → GPIO0 A [31:28]
PM3[3:2] = 1	Pmod3 lower row and Extension3 pins 3,5,7,9 → SPI1
PM3[3:2] > 1	reserved
PM4[1:0] = 0	Pmod4 upper row → GPIO0 B[3:0]
PM4[1:0] = 1	Pmod4 upper row → GPIO0 B[1:0] and I2C0
PM4[1:0] > 1	reserved
PM4[3:2] = 0	Pmod4 lower row → GPIO0 B [7:4]
PM4[3:2] = 1	Pmod4 lower row → GPIO0 B [5:4] and I2C1
PM4[3:2] > 1	reserved

#### 4.2.11 AUDIO\_CLK\_MUX\_CTRL Register

31	1	0
Reserved		AUDIO_CLK_SEL

Address offset: 0x0001\_1214

Reset Value: 0x0000\_0000

Access: RW

Register to select the HDMI audio reference clock

AUDIO\_CLK\_SEL      0 = audio reference clock is 12.288 MHz

                        1 = audio reference clock is 28.224 MHz

#### 4.2.12 SPI\_FLASH\_MUX\_CTRL Register

31	2	1	0
Reserved	SPI_FLASH_CS_N	SPI_FLASH_MUX_SEL	

Address offset: 0x0001\_1218

Reset Value: 0x0000\_0002 (0x0000\_0003 after pre-boot)

Access: RW

Register to control the SPI FLASH chip-select

SPI\_FLASH\_MUX\_SEL control source of SPI-FLASH chip-select

0 = chip-select controlled by SPI controller

1 = chip-select controlled by SPI\_FLASH\_MUX\_CTRL[1]

SPI\_FLASH\_CS\_N value of SPI-FLASH chip-select

- valid when SPI\_FLASH\_MUX\_CTRL[0]=1

- ignored when SPI\_FLASH\_MUX\_CTRL[0]=0

#### 4.2.13 SW\_RESET Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SW_RESET[7:0]	

Address offset: 0x0001\_1220

Reset Value: 0x00

Access: RW1C

Writing a ‘1’ to one or multiple bits within the SW\_RESET register initiates a software reset for the associated HW module(s). The software reset has a fixed duration of 32 APB clock cycles. Any additional requirements w.r.t. reset duration and/or polarity have been implemented in the CPLD supervisor.

SW_RESET [0]	→ reserved
SW_RESET [1]	→ audio stereo DAC
SW_RESET [2]	→ audio multi-channel DAC
SW_RESET [3]	→ Ethernet PHY
SW_RESET [4]	→ USB PHY

SW_RESET [5]	→ reserved
SW_RESET [6]	→ reserved
SW_RESET [7]	→ reserved

Note: When the USB Dataport operates in SPI mode for programming the SPI flash device, then the software reset functionality for the extension headers, the USB PHY, the Ethernet PHY and the audio codecs is disabled.

## 4.3 ICTL Registers

### 4.3.1 ICTL\_INT\_STATUS: Interrupt Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		Peripheral interrupts																													

Address offset: 0x0001\_2030

Reset Value: 0x0000\_0000

Access: R

This register provides the (masked) interrupt status.

ICTL_INT_STATUS [0]	→ CGU: PLL lock interrupt
ICTL_INT_STATUS [1]	→ CGU: PLL unlock interrupt
ICTL_INT_STATUS [2]	→ CGU: PLL lock error interrupt
ICTL_INT_STATUS [3]	→ CREG interrupt
ICTL_INT_STATUS [4]	→ Ethernet interrupt
ICTL_INT_STATUS [5]	→ PGU interrupt
ICTL_INT_STATUS [6]	→ NAND interrupt
ICTL_INT_STATUS [7]	→ SDIO interrupt
ICTL_INT_STATUS [8]	→ USB_HOST interrupt
ICTL_INT_STATUS [9]	→ DMAc interrupt
ICTL_INT_STATUS [10]	→ SPI_MEM interrupt
ICTL_INT_STATUS [11]	→ SPI0 interrupt
ICTL_INT_STATUS [12]	→ SPI1 interrupt
ICTL_INT_STATUS [13]	→ SPI2 interrupt
ICTL_INT_STATUS [14]	→ I2C0 interrupt
ICTL_INT_STATUS [15]	→ I2C1 interrupt
ICTL_INT_STATUS [16]	→ I2C2 interrupt
ICTL_INT_STATUS [17]	→ UART0 interrupt
ICTL_INT_STATUS [18]	→ UART1 interrupt
ICTL_INT_STATUS [19]	→ UART2 interrupt

ICTL_INT_STATUS [20]	→ GPIO0 interrupt
ICTL_INT_STATUS [21]	→ GPIO1 interrupt
ICTL_INT_STATUS [22]	→ Ethernet PHY interrupt
ICTL_INT_STATUS [23]	→ Reserved
ICTL_INT_STATUS [24]	→ HAPS Extension 0 interrupt
ICTL_INT_STATUS [25]	→ (signal HE_intr[0] at connector) → HAPS Extension 1 interrupt → (signal HE_intr[1] at connector)

## 4.4 GPIO Registers

### 4.4.1 GPIO0\_SWPORTA\_DR: GPIO0 Port A Output Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pmod3 / Extension3								Pmod2 / Extension2								Pmod1 / Extension1								Pmod0 / Extension0							

Address offset: 0x0001\_3000

Reset Value: 0x0000\_0000

Access: RW

This register controls the GPIO output signals of the Pmod connectors and the Extension headers when the corresponding GPIO0 bit is programmed as an output. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0_SWPORTA_DR [0]	→ Pmod0 pin 1	and Extension0 pin 6 (when PM0[0]=0)
GPIO0_SWPORTA_DR [1]	→ Pmod0 pin 2	and Extension0 pin 4 (when PM0[0]=0)
GPIO0_SWPORTA_DR [2]	→ Pmod0 pin 3	and Extension0 pin 2 (when PM0[0]=0)
GPIO0_SWPORTA_DR [3]	→ Pmod0 pin 4	and Extension0 pin 1 (when PM0[0]=0)
GPIO0_SWPORTA_DR [4]	→ Pmod0 pin 7	and Extension0 pin 3 (when PM0[2]=0)
GPIO0_SWPORTA_DR [5]	→ Pmod0 pin 8	and Extension0 pin 5 (when PM0[2]=0)
GPIO0_SWPORTA_DR [6]	→ Pmod0 pin 9	and Extension0 pin 7 (when PM0[2]=0)
GPIO0_SWPORTA_DR [7]	→ Pmod0 pin 10	and Extension0 pin 9 (when PM0[2]=0)
GPIO0_SWPORTA_DR [8]	→ Pmod1 pin 1	and Extension1 pin 6 (when PM1[0]=0)
GPIO0_SWPORTA_DR [9]	→ Pmod1 pin 2	and Extension1 pin 4 (when PM1[0]=0)
GPIO0_SWPORTA_DR [10]	→ Pmod1 pin 3	and Extension1 pin 2 (when PM1[0]=0)
GPIO0_SWPORTA_DR [11]	→ Pmod1 pin 4	and Extension1 pin 1 (when PM1[0]=0)
GPIO0_SWPORTA_DR [12]	→ Pmod1 pin 7	and Extension1 pin 3 (when PM1[2]=0)
GPIO0_SWPORTA_DR [13]	→ Pmod1 pin 8	and Extension1 pin 5 (when PM1[2]=0)
GPIO0_SWPORTA_DR [14]	→ Pmod1 pin 9	and Extension1 pin 7 (when PM1[2]=0)
GPIO0_SWPORTA_DR [15]	→ Pmod1 pin 10	and Extension1 pin 9 (when PM1[2]=0)
GPIO0_SWPORTA_DR [16]	→ Pmod2 pin 1	and Extension2 pin 6 (when PM2[0]=0)
GPIO0_SWPORTA_DR [17]	→ Pmod2 pin 2	and Extension2 pin 4 (when PM2[0]=0)

GPIO0 SWPORTA_DR [18]	→ Pmod2 pin 3	and Extension2 pin 2 (when PM2[0]=0)
GPIO0 SWPORTA_DR [19]	→ Pmod2 pin 4	and Extension2 pin 1 (when PM2[0]=0)
GPIO0 SWPORTA_DR [20]	→ Pmod2 pin 7	and Extension2 pin 3 (when PM2[2]=0)
GPIO0 SWPORTA_DR [21]	→ Pmod2 pin 8	and Extension2 pin 5 (when PM2[2]=0)
GPIO0 SWPORTA_DR [22]	→ Pmod2 pin 9	and Extension2 pin 7 (when PM2[2]=0)
GPIO0 SWPORTA_DR [23]	→ Pmod2 pin 10	and Extension2 pin 9 (when PM2[2]=0)
GPIO0 SWPORTA_DR [24]	→ Pmod3 pin 1	and Extension3 pin 6 (when PM3[0]=0)
GPIO0 SWPORTA_DR [25]	→ Pmod3 pin 2	and Extension3 pin 4 (when PM3[0]=0)
GPIO0 SWPORTA_DR [26]	→ Pmod3 pin 3	and Extension3 pin 2 (when PM3[0]=0)
GPIO0 SWPORTA_DR [27]	→ Pmod3 pin 4	and Extension3 pin 1 (when PM3[0]=0)
GPIO0 SWPORTA_DR [28]	→ Pmod3 pin 7	and Extension3 pin 3 (when PM3[2]=0)
GPIO0 SWPORTA_DR [29]	→ Pmod3 pin 8	and Extension3 pin 5 (when PM3[2]=0)
GPIO0 SWPORTA_DR [30]	→ Pmod3 pin 9	and Extension3 pin 7 (when PM3[2]=0)
GPIO0 SWPORTA_DR [31]	→ Pmod3 pin 10	and Extension3 pin 9 (when PM3[2]=0)

#### 4.4.2 GPIO0 SWPORTB\_DR: GPIO0 Port B Output Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Pmod4															

Address offset: 0x0001\_300C

Reset Value: 0x0000\_0000

Access: RW

This register controls the GPIO output signals of the Pmod4 connector when the corresponding GPIO0 bit is programmed as an output. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0 SWPORTB_DR [0]	→ Pmod4 pin 1
GPIO0 SWPORTB_DR [1]	→ Pmod4 pin 2
GPIO0 SWPORTB_DR [2]	→ Pmod4 pin 3 (when PM4[0]=0)
GPIO0 SWPORTB_DR [3]	→ Pmod4 pin 4 (when PM4[0]=0)
GPIO0 SWPORTB_DR [4]	→ Pmod4 pin 7
GPIO0 SWPORTB_DR [5]	→ Pmod4 pin 8
GPIO0 SWPORTB_DR [6]	→ Pmod4 pin 9 (when PM4[2]=0)
GPIO0 SWPORTB_DR [7]	→ Pmod4 pin 10 (when PM4[2]=0)

#### 4.4.3 GPIO0\_SWPORTC\_DR: GPIO0 Port C Output Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	Pmod1 / Extension1
----------	--------------------

Address offset: 0x0001\_3018

Reset Value: 0x0000\_0000

Access: RW

This register controls the GPIO output signals of the Pmod connectors and the Extension headers when the corresponding GPIO0 bit is programmed as an output. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0_SWPORTC_DR [0]	→ Pmod1 pin 1 and Extension1 pin 6 (when PM1[0]=1)
GPIO0_SWPORTC_DR [1]	→ Pmod1 pin 2 and Extension1 pin 4 (when PM1[0]=1)
GPIO0_SWPORTC_DR [2]	→ Pmod1 pin 3 and Extension1 pin 2 (when PM1[0]=1)
GPIO0_SWPORTC_DR [3]	→ Pmod1 pin 4 and Extension1 pin 1 (when PM1[0]=1)
GPIO0_SWPORTC_DR [4]	→ Pmod1 pin 7 and Extension1 pin 3 (when PM1[2]=1)
GPIO0_SWPORTC_DR [5]	→ Pmod1 pin 8 and Extension1 pin 5 (when PM1[2]=1)
GPIO0_SWPORTC_DR [6]	→ Pmod1 pin 9 and Extension1 pin 7 (when PM1[2]=1)
GPIO0_SWPORTC_DR [7]	→ Pmod1 pin 10 and Extension1 pin 9 (when PM1[2]=1)

#### 4.4.4 GPIO0 EXT\_PORTA: GPIO0 Port A Input Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Pmod3 / Extension3	Pmod2 / Extension2	Pmod1 / Extension1	Pmod 0 / Extension0
--------------------	--------------------	--------------------	---------------------

Address offset: 0x0001\_3050

Reset Value: Depends on external signals

Access: R

This register reflects the GPIO input signals of the Pmod connectors and the Extension headers when the corresponding GPIO0 bit is programmed as an input. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0 EXT_PORTA [0]	→ Pmod0 pin 1	and Extension0 pin 6 (when PM0[0]=0)
GPIO0 EXT_PORTA [1]	→ Pmod0 pin 2	and Extension0 pin 4 (when PM0[0]=0)
GPIO0 EXT_PORTA [2]	→ Pmod0 pin 3	and Extension0 pin 2 (when PM0[0]=0)
GPIO0 EXT_PORTA [3]	→ Pmod0 pin 4	and Extension0 pin 1 (when PM0[0]=0)
GPIO0 EXT_PORTA [4]	→ Pmod0 pin 7	and Extension0 pin 3 (when PM0[2]=0)
GPIO0 EXT_PORTA [5]	→ Pmod0 pin 8	and Extension0 pin 5 (when PM0[2]=0)
GPIO0 EXT_PORTA [6]	→ Pmod0 pin 9	and Extension0 pin 7 (when PM0[2]=0)
GPIO0 EXT_PORTA [7]	→ Pmod0 pin 10	and Extension0 pin 9 (when PM0[2]=0)
GPIO0 EXT_PORTA [8]	→ Pmod1 pin 1	and Extension1 pin 6 (when PM1[0]=0)
GPIO0 EXT_PORTA [9]	→ Pmod1 pin 2	and Extension1 pin 4 (when PM1[0]=0)
GPIO0 EXT_PORTA [10]	→ Pmod1 pin 3	and Extension1 pin 2 (when PM1[0]=0)
GPIO0 EXT_PORTA [11]	→ Pmod1 pin 4	and Extension1 pin 1 (when PM1[0]=0)
GPIO0 EXT_PORTA [12]	→ Pmod1 pin 7	and Extension1 pin 3 (when PM1[2]=0)
GPIO0 EXT_PORTA [13]	→ Pmod1 pin 8	and Extension1 pin 5 (when PM1[2]=0)
GPIO0 EXT_PORTA [14]	→ Pmod1 pin 9	and Extension1 pin 7 (when PM1[2]=0)
GPIO0 EXT_PORTA [15]	→ Pmod1 pin 10	and Extension1 pin 9 (when PM1[2]=0)
GPIO0 EXT_PORTA [16]	→ Pmod2 pin 1	and Extension2 pin 6 (when PM2[0]=0)
GPIO0 EXT_PORTA [17]	→ Pmod2 pin 2	and Extension2 pin 4 (when PM2[0]=0)
GPIO0 EXT_PORTA [18]	→ Pmod2 pin 3	and Extension2 pin 2 (when PM2[0]=0)
GPIO0 EXT_PORTA [19]	→ Pmod2 pin 4	and Extension2 pin 1 (when PM2[0]=0)
GPIO0 EXT_PORTA [20]	→ Pmod2 pin 7	and Extension2 pin 3 (when PM2[2]=0)
GPIO0 EXT_PORTA [21]	→ Pmod2 pin 8	and Extension2 pin 5 (when PM2[2]=0)
GPIO0 EXT_PORTA [22]	→ Pmod2 pin 9	and Extension2 pin 7 (when PM2[2]=0)
GPIO0 EXT_PORTA [23]	→ Pmod2 pin 10	and Extension2 pin 9 (when PM2[2]=0)
GPIO0 EXT_PORTA [24]	→ Pmod3 pin 1	and Extension3 pin 6 (when PM3[0]=0)
GPIO0 EXT_PORTA [25]	→ Pmod3 pin 2	and Extension3 pin 4 (when PM3[0]=0)
GPIO0 EXT_PORTA [26]	→ Pmod3 pin 3	and Extension3 pin 2 (when PM3[0]=0)
GPIO0 EXT_PORTA [27]	→ Pmod3 pin 4	and Extension3 pin 1 (when PM3[0]=0)
GPIO0 EXT_PORTA [28]	→ Pmod3 pin 7	and Extension3 pin 3 (when PM3[2]=0)
GPIO0 EXT_PORTA [29]	→ Pmod3 pin 8	and Extension3 pin 5 (when PM3[2]=0)
GPIO0 EXT_PORTA [30]	→ Pmod3 pin 9	and Extension3 pin 7 (when PM3[2]=0)
GPIO0 EXT_PORTA [31]	→ Pmod3 pin 10	and Extension3 pin 9 (when PM3[2]=0)

#### 4.4.5 GPIO0 EXT\_PORTB: GPIO0 Port B Input Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Pmod4															

Address offset: 0x0001\_3054

Reset Value: Depends on external signals

Access: R

This register reflects the GPIO input signals of the Pmod4 connector when the corresponding GPIO0 bit is programmed as an input. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0 EXT_PORTB [0]	→ Pmod4 pin 1
GPIO0 EXT_PORTB [1]	→ Pmod4 pin 2
GPIO0 EXT_PORTB [2]	→ Pmod4 pin 3 (when PM4[0]=0)
GPIO0 EXT_PORTB [3]	→ Pmod4 pin 4 (when PM4[0]=0)
GPIO0 EXT_PORTB [4]	→ Pmod4 pin 7
GPIO0 EXT_PORTB [5]	→ Pmod4 pin 8
GPIO0 EXT_PORTB [6]	→ Pmod4 pin 9 (when PM4[2]=0)
GPIO0 EXT_PORTB [7]	→ Pmod4 pin 10 (when PM4[2]=0)

#### 4.4.6 GPIO0 EXT\_PORTC: GPIO0 Port C Input Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Reserved	Pmod1 / Extension1
----------	--------------------

Address offset: 0x0001\_3058

Reset Value: Depends on external signals

Access: R

This register reflects the GPIO input signals of the Pmod connectors and the Extension headers when the corresponding GPIO0 bit is programmed as an input. Additionally, the settings of the control bits in the PMOD\_MUX\_CTRL register determine the operation. The relevant control bit PMx[y] is listed below.

GPIO0 EXT_PORTC [0]	→ Pmod1 pin 1	and Extension1 pin 6 (when PM1[0]=1)
GPIO0 EXT_PORTC [1]	→ Pmod1 pin 2	and Extension1 pin 4 (when PM1[0]=1)
GPIO0 EXT_PORTC [2]	→ Pmod1 pin 3	and Extension1 pin 2 (when PM1[0]=1)
GPIO0 EXT_PORTC [3]	→ Pmod1 pin 4	and Extension1 pin 1 (when PM1[0]=1)
GPIO0 EXT_PORTC [4]	→ Pmod1 pin 7	and Extension1 pin 3 (when PM1[2]=1)
GPIO0 EXT_PORTC [5]	→ Pmod1 pin 8	and Extension1 pin 5 (when PM1[2]=1)
GPIO0 EXT_PORTC [6]	→ Pmod1 pin 9	and Extension1 pin 7 (when PM1[2]=1)
GPIO0 EXT_PORTC [7]	→ Pmod1 pin 10	and Extension1 pin 9 (when PM1[2]=1)

#### 4.4.7 GPIO1\_SWPORTC\_DR: GPIO1 Port C Output Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GPIO LEDs															

Address offset: 0x0001\_4018

Reset Value: 0x0000\_0000

Access: RW

This register controls the GPIO LEDs. A LED is ON when its control bit is set to '1'

GPIO1_SWPORTC_DR [0]	→ GPIO LED0
GPIO1_SWPORTC_DR [1]	→ GPIO LED1
GPIO1_SWPORTC_DR [2]	→ GPIO LED2
GPIO1_SWPORTC_DR [3]	→ GPIO LED3
GPIO1_SWPORTC_DR [4]	→ GPIO LED4
GPIO1_SWPORTC_DR [5]	→ GPIO LED5
GPIO1_SWPORTC_DR [6]	→ GPIO LED6
GPIO1_SWPORTC_DR [7]	→ GPIO LED7

#### 4.4.8 GPIO1 EXT\_PORTA: GPIO1 Port A Input Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	GPIO push buttons																						Reserved								

Address offset: 0x0001\_4050

Reset Value: Depends on settings on the ARC SDP Mainboard

Access: R

This register reflects the status of the push buttons.

GPIO1 EXT_PORTA [24]	→ Push button GPIO0
GPIO1 EXT_PORTA [25]	→ Push button GPIO1
GPIO1 EXT_PORTA [26]	→ Push button GPIO2
GPIO1 EXT_PORTA [27]	→ Push button GPIO3
GPIO1 EXT_PORTA [28]	→ Push button GPIO4
GPIO1 EXT_PORTA [29]	→ Push button GPIO5

#### 4.4.9 GPIO1 EXT\_PORTB: GPIO1 Port B Input Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DIP switch SW2401															

Address offset: 0x0001\_4054

Reset Value: Depends on settings on the ARC SDP Mainboard

Access: R

This register reflects the status of the GPIO DIP switch SW2401. The first 9 bits of this switch are available for user applications. The last bit EXT\_PORTA[9] is used by the Pre-Bootloader (refer to the documentation of your ARC CPU Card).

GPIO1 EXT_PORTB [0]	→ SW2401, bit 1
GPIO1 EXT_PORTB [1]	→ SW2401, bit 2
GPIO1 EXT_PORTB [2]	→ SW2401, bit 3
GPIO1 EXT_PORTB [3]	→ SW2401, bit 4
GPIO1 EXT_PORTB [4]	→ SW2401, bit 5
GPIO1 EXT_PORTB [5]	→ SW2401, bit 6
GPIO1 EXT_PORTB [6]	→ SW2401, bit 7
GPIO1 EXT_PORTB [7]	→ SW2401, bit 8
GPIO1 EXT_PORTB [8]	→ SW2401, bit 9
GPIO1 EXT_PORTB [9]	→ SW2401, bit 10

*This chapter describes how to install the drivers for the USB Dataport and the AXS communicator tool `axs_comm`.*

## 5.1 USB-JTAG and USB-UART Driver Installation

Before the USB-JTAG and the USB-UART interfaces can be used, a driver needs to be installed on the computer where you intend to run the MetaWare debugger or another serial debug console (such as PuTTY or other hyperterminals).

The driver is a part of the Digilent Adept tool.

The Adept driver is included in the zip file `axs<main_board_number>_tools_<version>.zip` which is available at the ARC SDP download webpage [12].

Take the following steps for installing the driver:

1. Unzip `axs<main_board_number>_tools_<version>.zip`
2. Change to the `/tools/digilent_adept` sub-directory
3. Double-click on the `digilent.adept.system_<version>.exe` executable

You can also download the most recent version of the Digilent Adept tool from the Digilent website at <http://www.digilentinc.com>. Then follow the installation instructions provided by Digilent.

## 5.2 AXS Communicator Tool – `axs_comm`

### 5.2.1 Tool Overview and Installation Instructions

The `axs<main_board_number>_tools_<version>.zip` package, which is available at the ARC SDP download webpage [12], includes `axs_comm`, a PC-DOS-based tool for accessing peripherals via the USB Dataport. The tool is able to access the internal I<sup>2</sup>C bus of the ARC SDP Mainboard for programming the I<sup>2</sup>C EEPROM, or configuring the Real Time Clock, HDMI Transceiver and audio codecs. However, the tool is also able to program an image in the SPI-flash or to read the power status of the ARC SDP Mainboard from the CPLD.

Follow the steps below to install the `axs_comm` tool:

1. Download and unzip the `axs<main_board_number>_tools_<version>.zip` file
2. Change to the `/tools/axs_comm` sub-directory
3. Double-click the `axs_comm_install.bat` file in a Windows Explorer window or execute the batch file from a Windows Command Prompt

During installation `axs_comm` is added to the path and can be executed from any directory afterwards.

---

## 5.2.2 Usage

Before using the `axs_comm` tool, make sure that you connect your PC to the ARC SDP Mainboard using the USB Dataport connector. Additionally, any tools that might use the USB/UART channel of the USB Dataport (such as PuTTY or other hyperterminals) need to be closed.

Open a Windows command prompt and issue the `axs_comm -h` command to display all available options.

# Appendix A

This appendix provides a full overview of the board jumpers including their default settings and location on the printed circuit board.

## A.1 Jumper Overview

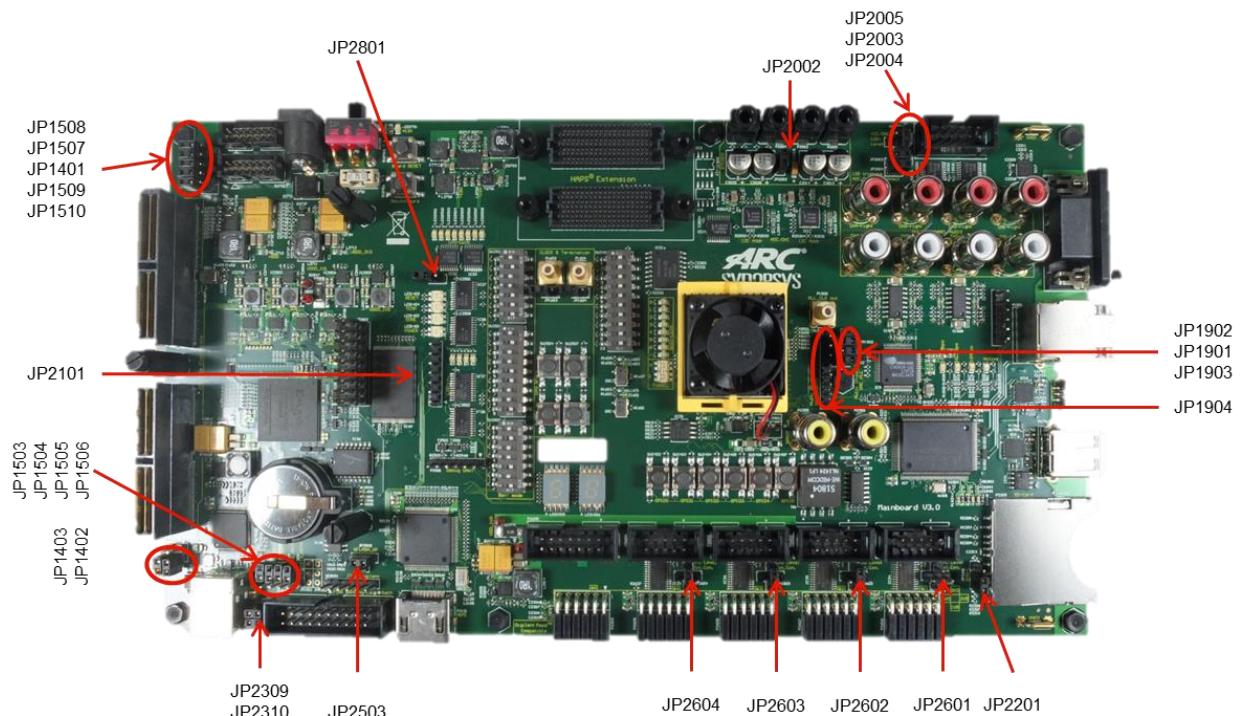


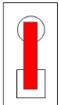
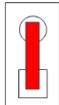
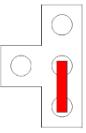
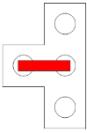
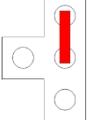
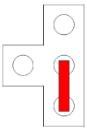
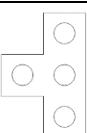
Figure 78 Jumper location overview

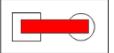
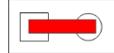
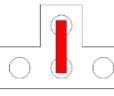
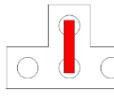
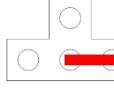
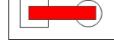
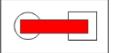
Table 63 Jumper functional overview and default settings

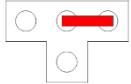
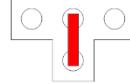
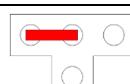
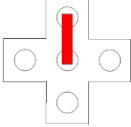
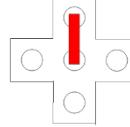
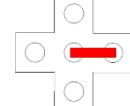
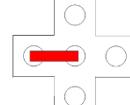
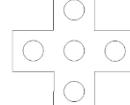
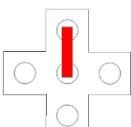
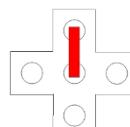
Jumper Name	Default Setting	Description
<b>CPU Debug</b>		
JP1402 JP1403		Debugger connected to USB Dataport

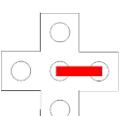
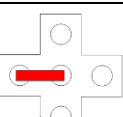
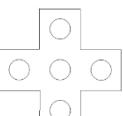
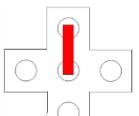
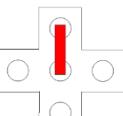
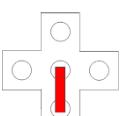
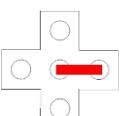
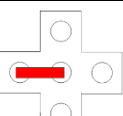
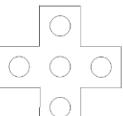
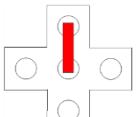
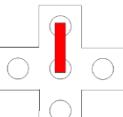
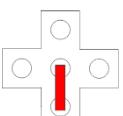
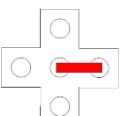
Jumper Name	Default Setting	Description	
			Debugger connected via one of the debug connectors CPU Debug Ashling / Lauterbach or CPU Debug Digilent
JP2309 JP2310	 		20-pin debug connector configured for Ashling probe
			20-pin debug connector configured for Lauterbach probe
<b>Reset Polarity at Extension Interfaces</b>			
JP1503			Active High RST at Extension0 header
			Active Low RST at Extension0 header
JP1504			Active High RST at Extension1 header
			Active Low RST at Extension1 header
JP1505			Active High RST at Extension2 header
			Active Low RST at Extension2 header
JP1506			Active High RST at Extension3 header
			Active Low RST at Extension3 header
<b>Reserved</b>			
JP1507			Reserved

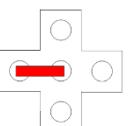
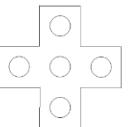
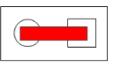
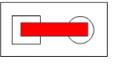
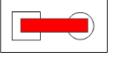
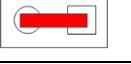
Jumper Name	Default Setting	Description	
			Default setting
JP1508			Reserved
			Default setting
JP1401			Reserved
			Default setting
JP1402			Default setting
			Reserved
<b>SD Card Mode</b>			
JP2201			SD-Card with authentication
			SD-card without authentication
JP1603			Reserved
			Default setting
JP1604			Reserved
			Default setting

Jumper Name	Default Setting	Description	
<b>Stereo ADC/DAC</b>			
JP2002			Dynamic Microphones (without battery) supported on MIC-in (P2002).
			Condenser Microphones (with battery) supported on MIC-in (P2002).
JP2005			Select 2V5 Level for External Stereo Codec connector (P2005)
			Select 1V8 Level for External Stereo Codec connector (P2005)
			Select 3V3 Level for External Stereo Codec connector (P2005). This setting is required for using the on-board stereo audio codecs.
			Not allowed

Jumper Name	Default Setting	Description	
<b>8-Channel Audio Codec</b>			
JP1901	 	CPU Card PLL_CLK (TC_audio_pll_clk) and the PLL_CLK Out connector are driven by the RMCK output of the 8-channel audio codec.	
		CPU Card PLL_CLK (TC_audio_pll_clk) and the PLL_CLK Out connector (P1909) are disconnected from the RMCK output of the 8-channel audio codec.	
JP1902	   	Use CPU Card PLL reference clock (TC_audio_pll_ref_clk) as input clock for the 8-channel audio codec PLL.	
		Reserved	
		Use I <sup>2</sup> S left/right clock of CPU Card (TC_mc8_o_ws) as input clock for the 8-channel audio codec PLL (if JP1904 is put in place)	
		PLL of 8-channel audio codec not driven externally.	
JP1903	 	CPU Card PLL_LOCK signal (TC_audio_pll_lock) is driven by on-board 8-channel audio codec.	
		Reserved	
JP1904a-h	 	Connect 8-channel I <sup>2</sup> S interface of CPU Card (TC_i2s_mc8_o_*) to on-board 8-channel audio codec.	
		Disconnect on-board 8-channel audio codec and connect an external audio codec to the CPU Card using the pins on the right side of JP1904 (see Figure 26).	

Jumper Name	Default Setting	Description
<b>NAND Flash Write Protection</b>		
JP2503		SW-controlled write protection mode
		 Read / write mode
		 Write protected mode
<b>Voltage Levels at Extension Connectors</b>		
JP2601		3V3 I/O levels on Extension0 connector
		 1V8 I/O levels on Extension0 connector
		 2V5 I/O levels on Extension0 connector
		 5V0 I/O levels on Extension0 connector
		 Not allowed
JP2602		3V3 I/O levels on Extension1 connector
		 1V8 I/O levels on Extension1 connector

Jumper Name	Default Setting	Description	
			2V5 I/O levels on Extension1 connector
			5V0 I/O levels on Extension1 connector
			Not allowed
JP2603			3V3 I/O levels on Extension2 connector
			1V8 I/O levels on Extension2 connector
			2V5 I/O levels on Extension2 connector
			5V0 I/O levels on Extension2 connector
			Not allowed
JP2604			3V3 I/O levels on Extension3 connector
			1V8 I/O levels on Extension3 connector
			2V5 I/O levels on Extension3 connector

<b>Jumper Name</b>	<b>Default Setting</b>	<b>Description</b>	
			5V0 I/O levels on Extension3 connector
			Not allowed
<b>Miscellaneous</b>			
JP801			Reserved
			RTC_MFP output
JP1509			Reserved
			Normal operation
JP1510			Reserved
			Normal operation
JP2801			Reserved
			Normal operation

# Appendix B

---

*This appendix describes how to store an application in the SPI flash memory. The SPI flash memory is application memory and not FPGA configuration memory.*

---

## B.1 Programming the SPI Flash Memory

Connect the ARC SDP Mainboard to your computer plugging the USB cable to the USB Dataport connector. Close any application that is using the USB/UART channel of the USB Dataport, such as PuTTY or other hyperterminals.

On your computer open a Command Prompt, change to the directory that contains the `axs_comm` tool (see “[AXS Communicator Tool – axs\\_comm](#)”) and program the SPI Flash memory as described below.

---

### B.1.1 Using `axs_comm` for SPI FLASH programming

`axs_comm` has direct access to the SPI Flash device on the ARC SDP Mainboard via the USB Dataport. Note that this is the application memory rather than the FPGA configuration memory.

`axs_comm` tool allows flashing an image in the device. `Axs_comm` automatically erases as many pages as necessary.

Next to the image the batch file generates and stores a header based on the options provided on the command line. This header controls the operation of the pre-boot loader (refer to the user guide of your ARC CPU Card).

After programming the image the data stored in the SPI Flash memory is also verified.



**Note** Instructions for restoring the factory default content of the SPI Flash device are provided in the documentation of your ARC CPU Card.

---

### B.1.1.1 Command Line Options

```
axs_comm -c <arc_id> -p <memory_address> -f <filename>
```

- <arc\_id>

This is the ARC ID of the CPU core for which the image has been compiled. Refer to the user guide of your ARC CPU Card for applicable ARC IDs.

For example, the ARC ID of an ARC HS36 core is 0551 and the ARC ID of an ARC 770D is 0434.

- <memory\_address>

Specifies the target address in the system memory to which the pre-bootloader shall copy the image. The value has to be provided in hexadecimal format, but without a leading “0x”. For example, enter 80000000 for selecting the address 0x80000000.

- <filename>

This is the filename of the \*.bin file to be flashed into the on-board SPI Flash memory.

### B.1.1.2 Example for an ARC AXC 003 CPU Card

For programming the image, `selftest_axs<main_board_number>_<CPU_core>.bin`, to the SPI Flash starting at the first page of the first sector.

Use the following command to program the CPU card:

```
axs_comm -c 0552 -p 80000000 -f selftest_axs103_archs36.bin
```

In this example, the generated header informs the pre-bootloader that this image has been compiled for the ARC HS36 (ARC ID = 0552) and the image has to be copied to the memory address 0x80000000. This is the start address of the DDR3 SDRAM on the ARC AXC003 CPU Card.

Alternatively, you can run the batch file `axs_comm_program_selftest.bat`.



**Note** The `selftest_axs<main_board_number>_<CPU_core>.bin` image file is included in the `axs<main_board_number>_selftest_firmware_<version>.zip` package.

### B.1.1.3 Example for an ARC AXC 001 CPU Card

For programming the image, `selftest_axs<main_board_number>_<CPU_core>.bin`, to the SPI Flash starting at the first page of the first sector.

Use the following command to program the CPU card:

```
axs_comm -c 0434 -p 80000000 -f selftest_axs101_arc770.bin
```

In this example, the generated header informs the pre-bootloader that this image has been compiled for the ARC 770D (ARC ID = 0434) and the image has to be copied to the memory address 0x80000000. This is the start address of the DDR2 SDRAM on the ARC AXC001 CPU Card.

Alternatively, you can run the batch file `axs_comm_program_selftest.bat`.



**Note** The `selftest_axs<main_board_number>_<CPU_core>.bin` image file is included in the `axs<main_board_number>_selftest_firmware_<version>.zip` package.

---

# Appendix C

---

*This appendix describes how the time and date of the on-board RTC can be set to the current date and time of your PC.*

---

## C.1 Configuring the Real Time Clock

Connect the ARC SDP Mainboard to your computer plugging the USB cable to the USB Dataport connector. Close any application that may be using the USB/UART channel of the USB Dataport. For example, PuTTY or any other hyperterminals.

On your computer open a Command Prompt, change to the directory where you unzipped the `axs<main_board_number>_tools_<version>.zip` package (see “[AXS Communicator Tool – axs\\_comm](#)”). Then change to the `/tools/axs_comm` sub-directory and enter the following command:

```
axs_comm -st
```

This command sets the RTC time and date registers according to the `DATE` and `TIME` environment variables of your PC. Additionally, it enables the battery backup mode of the RTC.

To verify the time setting enter the following command:

```
axs_comm -gt
```

This command displays the date and time in the following format:

`day_of_the_week month/day/year hours:minutes:seconds`

Example:

```
Wed 08/28/2013 14:23:12
```

# Appendix D

This appendix provides the pin description of the HAPS Trak-3 Extension Connector.

## D.1 HAPS Trak-3 Extension Connector Pins

Table 64 Pin description of the HAPS Trak-3 extension connectors J3 and J4

J3-Pin Number	J3-Signal	J4-Pin Number	J4-Signal
A0	HE_taxi_ss_tx[31]	A0	HE_taxi_ss_rx_clk
A1		A1	HE_taxi_ss_rx_valid
A2	HE_taxi_ss_tx[25]	A2	
A3	HE_taxi_ss_tx[24]	A3	HE_taxi_ss_rx[13]
A4	HE_taxi_ss_tx[21]	A4	HE_taxi_ss_rx[7]
A5	HE_taxi_ss_tx[10]	A5	HE_taxi_ss_rx[30]
A6	HE_taxi_ss_tx[16]	A6	HE_taxi_ss_rx[10]
A7	HE_taxi_ss_tx[1]	A7	HE_taxi_ss_rx[1]
A8	HE_taxi_ss_tx[12]	A8	HE_taxi_ss_rx[8]
A9		A9	HE_taxi_ss_rx[9]
A10	HE_taxi_ss_tx[7]	A10	HE_taxi_ss_rx[28]
A11	HE_taxi_ss_tx[8]	A11	HE_taxi_ss_rx[29]
B0		B0	
B1	HE_taxi_ss_tx[27]	B1	HE_taxi_ss_rx[14]
B2	HE_taxi_ss_tx[30]	B2	
B3	HE_taxi_ss_tx[23]	B3	HE_taxi_ss_rx[11]
B4	HE_taxi_ss_tx[20]	B4	HE_taxi_ss_rx[35]
<u>B5</u>	HE_taxi_ss_tx[11]	<u>B5</u>	HE_taxi_ss_rx[6]
B6	HE_taxi_ss_tx[14]	B6	HE_taxi_ss_rx[33]
B7	HE_taxi_ss_tx[5]	B7	HE_taxi_ss_rx[5]
B8	HE_taxi_ss_tx[9]	B8	HE_taxi_ss_rx[26]

<b>J3-Pin Number</b>	<b>J3-Signal</b>	<b>J4-Pin Number</b>	<b>J4-Signal</b>
B9	HE_taxi_ss_tx[6]	B9	HE_taxi_ss_rx[27]
B10	HE_intr[1]	B10	HE_taxi_ss_rx[24]
B11	HE_RSTn	B11	HE_taxi_ss_rx[25]
C0		C0	
C1	HE_taxi_ss_tx[28]	C1	HE_taxi_ss_rx[15]
C2	HE_taxi_ss_tx[22]	C2	
C3	HE_taxi_ss_tx[13]	C3	HE_taxi_ss_rx[12]
C4	HE_taxi_ss_tx[19]	C4	HE_taxi_ss_rx[34]
C5	HE_taxi_ss_tx[3]	C5	HE_taxi_ss_rx[4]
C6	HE_taxi_ss_tx[15]	C6	HE_taxi_ss_rx[32]
C7	HE_taxi_ss_tx_ctrl[2]	C7	HE_taxi_ss_rx[3]
C8	HE_taxi_ss_tx[2]	C8	HE_taxi_ss_rx[22]
C9	HE_taxi_ss_tx[0]	C9	HE_taxi_ss_rx[23]
<u>C10</u>	HE_taxi_ss_tx_ctrl[1]	<u>C10</u>	HE_taxi_ss_rx[20]
C11	HE_intr[0]	C11	HE_taxi_ss_rx[21]
D0	HE_taxi_ss_tx[17]	D0	HE_taxi_ss_rx[31]
D1	HE_taxi_ss_tx[29]	D1	
D2	HE_taxi_ss_tx_clk	D2	HE_taxi_ss_rx_ctrl[2]
D3	HE_taxi_ss_tx[26]	D3	HE_taxi_ss_rx_ctrl[1]
D4	HE_taxi_ss_tx[4]	D4	
D5	HE_taxi_ss_tx_ctrl[0]	D5	HE_taxi_ss_rx[17]
D6	HE_taxi_ss_tx[18]	D6	HE_taxi_ss_rx_ctrl[0]
D7	HE_taxi_ss_tx[34]	D7	HE_taxi_ss_rx[0]
D8	HE_taxi_ss_tx_valid	D8	HE_taxi_ss_rx[18]
D9	HE_taxi_ss_tx[35]	D9	HE_taxi_ss_rx[19]
D10	HE_taxi_ss_tx[33]	D10	HE_taxi_ss_rx[2]
D11	HE_taxi_ss_tx[32]	D11	HE_taxi_ss_rx[16]

# Appendix E

This appendix provides information on the CPU Card connectors.

## E.1 Using GPIO Pins at the ARC CPU Card Connectors

The CPU Card connectors include 24 GPIO pins, which are connected to the DIP switches SW2501, SW2502, and SW2503 during reset. These switches are typically used to configure the boot mode of the ARC CPU Card. The values are latched by the CPU Card at the end of the reset. After reset these signals have different functions, which are described in Table 65.

Table 65 GPIO field of ARC CPU Card Interface

Pin Name	Description
TC_gpio_a[0]	Connected to LED2501
TC_gpio_a[1]	Connected to LED2502
TC_gpio_a[2]	Reserved (pulled-down on Mainboard)
TC_gpio_a[3]	Reserved (pulled-down on Mainboard)
TC_gpio_a[4]	Reserved (pulled-down on Mainboard)
TC_gpio_a[5]	Connected to LED2503
TC_gpio_a[6]	Connected to LED2504
TC_gpio_a[7]	Reserved (pulled-down on Mainboard)
TC_gpio_a[8]	Reserved (pulled-down on Mainboard)
TC_gpio_a[9]	Reserved (pulled-down on Mainboard)
TC_gpio_a[10]	Connected to LED2505
TC_gpio_a[11]	Connected to LED2506
TC_gpio_a[12]	Connected to the output of the interrupt controller. The GPIO peripheral of the CPU Card needs to be programmed as an input. Can be used to provide an interrupt from the peripheral subsystem to the ARC CPU Card.
TC_gpio_a[13]	Reserved (pulled-down on Mainboard)
TC_gpio_a[14]	Reserved (pulled-down on Mainboard)
TC_gpio_a[15]	Connected to LED2507
TC_gpio_a[16]	Connected to LED2508

Pin Name	Description
TC_gpio_a[17]	Reserved (pulled-down on Mainboard)
TC_gpio_a[18]	Reserved
TC_gpio_a[19]	Reserved
TC_gpio_a[20]	Connected to push button SW2504
TC_gpio_a[21]	Connected to push button SW2506
TC_gpio_a[22]	Connected to push button SW2505
TC_gpio_a[23]	Connected to push button SW2507

## E.2 ARC CPU Card Power Supply Connector

The ARC SDP Mainboard supplies power to the ARC CPU Card through a separate connector. A pin description of this connector is provided in Table 66 and the pinout is shown in Figure 79.

Table 66 Pin description of the ARC CPU Card power supply connector

Pin	Name	Description	Direction
1,2	12V	12 Volt power supply pin	
3,4	GND	Ground supply pin	
5,6	1V1	1.1 Volt power supply pin	
7,8	GND	Ground supply pin	
9,10	1V8	1.8 Volt power supply pin	
11,12	GND	Ground supply pin	
13,14	2V5	2.5 Volt power supply pin	
15,16	GND	Ground supply pin	
17,18	3V3	3.3 Volt power supply pin	

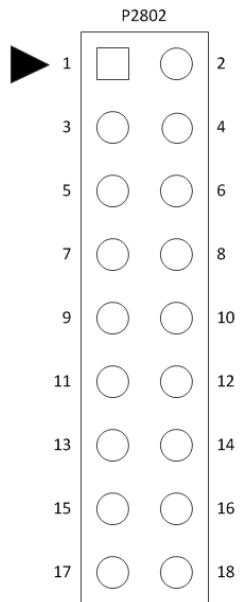


Figure 79 Power supply of the ARC CPU Card

# Appendix F

This appendix provides information on using the ARC SDP Mainboard with a HAPS system.

## F.1 Adding System Extensions via HAPS Extension Interface

The ARC SDP Mainboard includes two HapsTrak-3 connectors for connecting prototyping systems from the HAPS product family. See the “[HAPS HapsTrak-3 Extension](#)” section for a description of the HAPS HapsTrak-3 extension interface.

The peripheral subsystem of the ARC SDP Mainboard maps an AXI tunnel onto the HAPS HapsTrak-3 extension interface. This AXI tunnel allows seamless integration of the AXI system of the peripheral subsystem with a secondary AXI system on a HAPS prototyping system. The AXI tunnel consists of two complementary, identical modules; one on each side of the HAPS extension interface. The two complementary counter-parts of the AXI tunnel communicate with each other via a bi-directional source-synchronous link. A pin description of the source-synchronous link via the HapsTrak-3 connector is provided in “[Appendix D](#)”

The HAPS extension hardware package (`axs<main_board_number>_haps_extension_<version>.zip`) contains an example design that can be used as a starting point for adding system extensions via the HAPS extension interface. The example design instantiates an AXI tunnel alongside the other basic functions like e.g. clock generation. In the example design the slave port of the AXI tunnel is immediately forwarded to the master port of the AXI tunnel. This implements a simple loopback that can be used for initial bring-up of the AXI tunnel. Once the loopback functionality has been verified, the master and slave ports of the AXI tunnel can be connected to an AXI bus and the extension peripherals can be added.

Step-by-step instructions for adding systems extensions via the HAPS extension interface:

1) Unzip the HAPS extension hardware package (`axs<main_board_number>_haps_extension_<version>.zip`) and you will get all the RTL sources of the HAPS extension example design including scripts for simulation, FPGA synthesis and P&R. The top-level directory structure will look as follows:

- /haps\_extension/data/ip\_sc  
Sub-IP instantiated in the design
- /haps\_extension/data/top/arc\_dev\_sys\_he/RTL  
RTL source files
- /haps\_extension/data/top/arc\_dev\_sys\_he/setup  
Setup scripts

- `/haps_extension/data/top/arc_dev_sys_he/synopsys_fpga`  
FPGA synthesis / P&R scripts
- 2) Go to the “`/haps_extension/data/top/arc_dev_sys_he/synopsys_fpga`” directory. In this directory you will find the following three sub-directories:
- `/syn`  
FPGA synthesis scripts (using Synplify)
  - `/vivado`  
FPGA P&R scripts for HAPSDX and HAPS70 (using Vivado)
- 3) Go to the “`/haps_extension/data/top/arc_dev_sys_he/synopsys_fpga/syn`” directory. In this directory you will find a run script (`run_syn`) to synthesize the example design. The target prototyping system (hapsdx or haps7) and which HT-3 connectors to be used, can be selected via command-line options of the run script. Type `run_syn -h` for more information on the command-line options. This example provides pinning information using several HT-3 connector options for the HAPSDX as well as for the HAPS70 system. The pin assignments are located in the following file:
- `arc_dev_sys_he_pin_loc_<name>_<ht-3_con>.fdc`  
where `<name>` is the name of the prototyping system (hapsdx or haps7)  
where `<ht-3_con>` are the names of the selected HT-3 connectors (a1\_a2, a3\_a4, a5\_a6, a7\_a8, a9\_a10 for hapsdx, a1\_a2, a3\_a4, a5\_a6, a7\_a8, a9\_a10, a11\_a12, a13\_a14, a21\_a22 for haps7)
- With the command line options `-con1` and `-con2` the HT-3 connectors at the HAPS side will be selected which will be used to make an AXI tunnel connection with the SDP Main board. Execute the `run_syn` script with the appropriate command-line options, for an HAPSDX example using HT-3 connectors J1 and J2 use the following command line options (make sure that the `-con1` / `-con2` parameters are equal to the `<ht-3_con>` connector options given above) :
- `run_syn -target hapsdx -con1 a1 -con2 a2`
- 4) Go to the “`/haps_extension/data/top/arc_dev_sys_he/synopsys_fpga/vivado`”. In this directory you will find a run script (`run_vivado`) to P&R the example design. The target prototyping system can be selected via command-line options of the run script. Type `run_vivado -h` for more info on the command-line options. After successful completion of P&R (using the same command line options for `run_vivado` as used for `run_syn`) you will find a `.bit` and `.bin` file in the following directory:
- `/haps_extension/data/top/arc_dev_sys_he/synopsys_fpga/vivado/rev_1/bit`
- 5) Program the bit-file (or bin-file) into your HAPS prototyping system

- 6) Configure the I/O voltage (for the selected HT-3 connectors) on your HAPS prototyping system to 1V8
- 7) Configure the system clock (GCLK1) on your HAPS prototyping system to 25 MHz
- 8) Connect your prototyping system to the ARC SDP Mainboard using the HapsTrak-3 extension interface. For this connection the following HAPS cables can be used (use two of the same cables):
  - H\_CON\_CABLE\_25\_HT3 (cable length of 25 cm)
  - H\_CON\_CABLE\_50\_HT3 (cable length of 50 cm)

For the example given above, the SDP extension connector J3 must be connected with the HAPS connector J1 since `-con1 a1` was used. Furthermore, SDP extension connector J4 must be connected with the HAPS connector J2 since `-con2 a2` was used.

- 9) Switch-on the ARC SDP Mainboard. Press Reset. The status LED for the HAPS extension interface on the ARC SDP Mainboard (i.e. TUNNEL1) should now turn green. This indicates that the AXI tunnel for the HAPS extension interface has been properly initialized.
- 10) After successful completion of step 9) you can extend the example design with your AXI subsystem.

# Appendix G

This appendix provides information on board maintenance.

## G.1 Replacing the Fuse

The ARC SDP Mainboard is protected by a 5A type T fuse. The location of the fuse is shown in Figure 80.

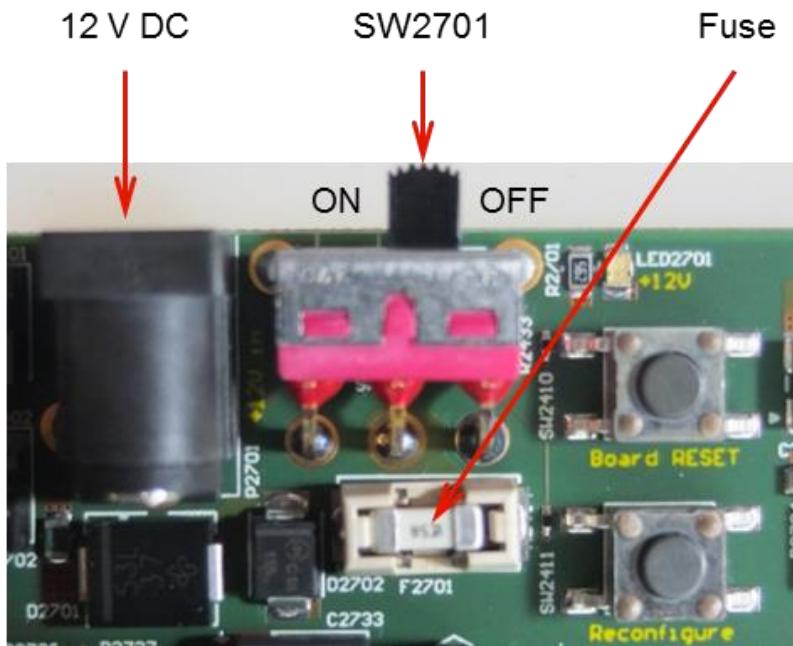


Figure 80 Location of the fuse

For replacing the fuse switch off the ARC SDP Mainboard and remove any ARC CPU Card.

Remove the fuse from the holder using a small screwdriver as a lever.

Insert the new fuse by pressing it in the holder.

## G.2 Backup Battery

The encryption key (when programmed) and the real time clock are backed by a 3.6V 40 mAh Li-Ion Rechargeable button cell battery (LIR2032)

Since this battery will be charged when the Mainboard is powered-up, replacement of this battery is not required.

In case the board is not used for more than one year, you might need to set the real-time clock once again as described in "[Appendix C](#)"

# Glossary and References

---

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

---

## Glossary

**AHB**

Advanced High Performance Bus

**AXI**

Advanced eXtensible Interface

**CPLD**

Complex Programmable Logic Device

**FPGA**

Field Programmable Gate Array

**GPIO**

General Purpose Input/Output

**HDMI**

High Definition Multimedia Interface

**HW**

Hardware

**HAPS**

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

**HAPSTrak-II**

Standard (SAMTEC) connector type used in HAPS systems

**HAPSTrak-3**

Standard (SAMTEC) connector type used in HAPS systems

**Kintex**

Xilinx FPGA series

**RTC**

Real Time Clock

**R**

Read-only register

**RW**

Read-write register

**RW1C**

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

**SDP**

Software Development Package

**S/PDIF**

Sony/Philips Digital Interface

**SW**

Software

**USB**

Universal Serial Bus

---

## References

- [1] *HAPS Trak-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] *HAPS Trak-3 standard*
- [3] *Digilent Pmod Interface Specification*, <http://digilentinc.com/>
- [4] *Xilinx Kintex-7 FPGA*, <http://www.xilinx.com/products/silicon-devices/fpga/kintex-7/index.htm>
- [5] *IDT Frequency Synthesizer (clock generator)*, <http://www.idt.com>
- [6] *USB to Serial cable*, [http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS\\_TTL-232R\\_CABLES.pdf](http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf)
- [7] *Ethernet PHY DP83865 Datasheet*, <http://www.ti.com>
- [8] *HDMI Transmitter ADV7511 Hardware User Guide*, <http://www.analog.com>
- [9] *HDMI Transmitter ADV7511 Programming Guide*, <http://www.analog.com>
- [10] *USB Transceiver Chip TUSB1210 Datasheet*, <http://www.ti.com>
- [11] *DW\_ahb\_dmac Databook*, <http://www.synopsys.com>
- [12] ARC SDP download webpage  
*You have received the corresponding URL during the purchasing process*