



DesignWare ARC HS4xD Development Kit User Guide

Version 5818-001 April 2020

Copyright Notice and Proprietary Information Notice

© 2020 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

<https://www.synopsys.com/company/legal/trademarks-brands.html>.

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.
www.synopsys.com

Contents

Contents.....	3
List of Figures	5
List of Tables.....	6
1 Customer Support.....	7
1.1 Accessing SolvNetPlus	7
1.2 Accessing ARC Application Notes	7
1.3 Contacting the Synopsys Technical Support Center	8
2 Introduction	10
2.1 Package Content.....	10
2.2 Getting Started	10
2.2.1 Installing Device Drivers.....	10
2.2.3 Checking Default Board Settings	11
2.2.4 Installing and Configuring PuTTY.....	11
2.2.5 Starting U-Boot.....	13
2.3 Location of Components On HSDK-4xD	14
2.4 Software Packages	16
3 Hardware Description	17
3.1 Overview of ARC HSDK-4xD	17
3.2 Overview of ARC HS4xD Development System SoC.....	19
3.3 Clocks and Resets	21
3.3.1 Clocks	21
3.3.2 Reset.....	24
3.4 Interrupts	26
3.5 Debug and Trace	32
3.5.1 Debug	33
3.5.2 ARC Real-Time Trace.....	35
3.6 Configuration and Boot Modes.....	36
3.6.1 Boot Switches	37
3.6.2 Jumpers	38
3.6.3 Other Switches.....	39
3.6.4 On-board LEDs	39

3.7 Memories	40
3.8 USB Interface	40
3.9 Ethernet Interface	40
3.10 SD Card Interface	41
3.11 Redpine® WiFi / Bluetooth Interface	41
3.12 Audio Interface	41
3.13 On-board I2C Control Bus	42
3.14 ADC	42
3.15 I/O Expander	43
3.16 Extension Interfaces	44
3.16.1 Digilent Pmod™	45
3.16.2 Mikrobus	48
3.16.3 Arduino	49
3.16.4 HapsTrak3 Extension	51
4 Programmer's Reference	53
4.1 Memory Map	53
4.1.1 APB Peripheral Address Map	54
4.1.2 PAE	56
4.1.3 I/O Coherency	57
4.2 Software Interfaces	58
4.2.1 Control Registers	58
4.2.2 Clock Registers	86
4.2.3 PWM Registers	99
Appendix A HapsTrak 3 Extension Connector Pins	116
A.1 HapsTrak 3 Extension Connector Pin Description	116
Glossary and References	118
Glossary	118
References	119

List of Figures

Figure 1 Identification of COM Port	11
Figure 2 PuTTY Configuration	12
Figure 3 Application Boot with ARC HS47D Initialization	13
Figure 4 Linux Boot with ARC HS48x4 Initialization	14
Figure 5 HSDK-4xD Components- Top View	15
Figure 6 HSDK-4xD Components: Bottom View	15
Figure 7 ARC HSDK-4xD Block Diagram	17
Figure 8 ARC HS4xD Development System SoC Top-Level Diagram	19
Figure 9 ARC HSDK-4xD Clock Architecture	21
Figure 10 ARC HSDK-4xD Reset Architecture	25
Figure 11 ARC HSDK-4xD Interrupt Architecture	27
Figure 12 ARC HSDK-4xD Debug and Trace Headers	33
Figure 13 JTAG Daisy Chain	33
Figure 14 10-Pin to 20-Pin JTAG Adapter	34
Figure 15 ARC HSDK-4xD Configuration and Boot Switches and Buttons	36
Figure 16 ARC HSDK-4xD Boot Switches	37
Figure 17 ARC HSDK-4xD Peripheral Extension Interfaces	44
Figure 18 ARC HSDK-4xD HAPS Extension Interface	44
Figure 19 Pinout Diagram of the Pmod_A, Pmod_B and Pmod_C Connectors	45
Figure 20 MikroBus Headers	49
Figure 21 Arduino Shield Interface	50
Figure 22 ARC HSDK-4xD HAPS Extension: Typical Use-case	52
Figure 23 ARC HSDK-4xD Memory Map (After U-Boot)	53
Figure 24 ARC HS4xD Development System SoC Memory Map – High Level Overview	56
Figure 25 I/O Coherency Architecture	57

List of Tables

Table 1	Overview of ARC HSDK-4xD Clock Components.....	22
Table 2	ARC HSDK-4xD Interrupt Mapping.....	28
Table 3	ARC ID Codes.....	34
Table 4	ARC HSDK-4xD Status and System LEDs	39
Table 5	Ethernet Link Speed Indication	40
Table 6	I2S SCLK Divider Settings	41
Table 7	ARC HSDK-4xD On-board I2C Save Addresses	42
Table 8	ADC Channel Usage.....	42
Table 9	I/O Expander I/O Overview	43
Table 10	Available Protocol Options	45
Table 11	Pin Description of the Pmod_A Connector.....	46
Table 12	Pin Description of the Pmod_B Connector.....	47
Table 13	Pin Description of the Pmod_C Connector	48
Table 14	Pin Description of the MikroBUS Connectors	49
Table 15	Pin Description of the Arduino Shield Interface.....	51
Table 16	APB Peripheral Address Map	54
Table 17	CREG Control Register Overview.....	59
Table 18	CREG Address Decoder Register Reset Values (After U-Boot)	68
Table 19	GPIO Mux	77
Table 20	CGU Clock Register Overview.....	86
Table 21	PWM Control Register overview	99
Table 22	Pin Description of the HapsTrak 3 Extension Connectors J3 And J4	116

Customer support is available through SolvNetPlus online customer support and through contacting the Synopsys Technical Support Center.

1.1 Accessing SolvNetPlus

SolvNetPlus includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNetPlus also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation on the Web, and entering a call to the Support Center.

To access SolvNetPlus:

1. Go to the SolvNetPlus Web page at <http://solvnetplus.synopsys.com/>.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNetPlus.)

If you need help using SolvNetPlus, click **SolvNetPlus Help** in the **Support Resources** section.

1.2 Accessing ARC Application Notes

SolvNetPlus articles include useful application notes to help you do your job using DesignWare ARC products. To find application notes for your DesignWare ARC product, do the following:

1. Log into SolvNetPlus at <https://solvnetplus.synopsys.com/>.
2. Enter the name of your product in the **Search** field.
3. Click **Search**.

The figure below shows example results.



Note Application notes are updated frequently, so results can vary.

The screenshot shows the Synopsys SolvNetPlus search interface. The search bar contains 'ARC HS' and the search button is visible. The navigation menu includes 'Cases', 'STARs', 'Knowledge Base', 'Legacy Docs Search', and 'Feedback'. The search results are displayed under the 'Knowledge' tab, showing 5+ results sorted by relevance. The results list includes:

- Software Bootloading on ARC EM and ARC HS** (ID: 000024517, Last Modified: Dec 12, 2019, 7:28 PM). Question: How to I initialize ICCM and DCCM from ROM on an ARC EM or ARC HS processor?
- Power Management on ARC HS** (ID: 000030883, Last Modified: Dec 25, 2019, 1:54 AM). HS processor family and gives guidance on using them. These features include functional clock
- ARCv2 and EV Hardware ID Core Versions and Compiler Options** (ID: 000025590, Last Modified: Dec 12, 2019, 9:24 PM). 0x43 32 bit 3 stage ocao -oore3 -av2em ARC HS Note: ARC HS has two versions: ARC HS3x and ARC
- Useful Signals for Debugging ARC HS3x RTL Simulations** (ID: 000025072, Last Modified: Dec 12, 2019, 7:06 PM). HS3x hierarchy for helping debug RTL simulations.
- DesignWare ARC Processor Training** (ID: 000015152, Last Modified: Dec 16, 2019, 12:38 AM). Direct links to the forms are listed here: ARC 600 ARC 700 ARG EM ARC HS ARC MQX RTOS

At the bottom of the search results, there is a link 'Not finding what you are looking for?' and a button 'Create a New Case'.

1.3 Contacting the Synopsys Technical Support Center

Synopsys provides various methods for contacting Customer Support, as follows:

- For the fastest response, enter a case through SolvNetPlus:
 - a. <https://solvnetplus.synopsys.com>



Note SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

- b. Click the **Cases** menu and then click **Create a New Case** (below the list of cases).
- c. Complete the mandatory fields that are marked with an asterisk and click **Save**.

Make sure to include the following:

- **Product L1:** Select one of the following:
 - **DesignWare Cores ARC Tools**

- **DesignWare Cores ARC Processors**
- **DesignWare Subsystems**
- **Product L2:** (select the name that best matches your product)

d. After creating the case, attach any debug files you created.

For general usage information, refer to the following article in SolvNetPlus:

<https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources>

**Note**

You must be logged in to SolvNetPlus before clicking this link:
<https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources>

- Or, send an e-mail message to support_center@synopsys.com (your email is queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
 - Include the full product name and version number in your e-mail so it can be routed correctly.
 - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood.
 - Attach any debug files you created.
- Or, telephone your local support center:
 - North America:
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
 - All other countries:
<https://www.synopsys.com/support/global-support-centers.html>

2.1 Package Content

The DesignWare ARC HS4xD Development Kit package contains the following items:

- DesignWare ARC HS4xD Development Kit (herein referred to as ARC HSDK-4xD in this document)
- A 100-240 V AC power adapter (including adapters for U.S., UK, and EU outlets)
- USB cable



Warning

The DesignWare ARC HS4xD Development Kit contain static-sensitive devices.

2.2 Getting Started

This section includes instructions for the following tasks:

1. Installing device drivers
2. Checking default board settings
3. Installing and configuring PuTTY
4. Starting U-Boot

2.2.1 Installing Device Drivers

Before the USB-JTAG and the USB-UART interfaces can be used, you must install the required drivers on the computer where you intend to run the MetaWare debugger [5] or another serial debug console (such as PuTTY or other hyper-terminals).

The driver is a part of the Digilent Adept tool. You can download the most recent version of the Digilent Adept tool from the Digilent website at <http://www.digilentinc.com>, and follow the installation instructions provided by Digilent.

2.2.3 Checking Default Board Settings

Check if the boot switches and jumpers are set to their default positions.

For an overview of the configuration options and the default settings, see [Configuration and Boot Modes](#) on page 36. Connect the ARC HSDK-4xD to your PC by connecting the USB cable to the USB data port of the ARC HSDK-4xD and the PC.

Connect the power supply included in the product package to the ARC HSDK-4xD.



Note The ARC HSDK-4xD must be powered by an external power adapter. Also, when the ARC HSDK-4xD is mounted on a HAPS system, the board ARC HSDK requires this external power adapter.

2.2.4 Installing and Configuring PuTTY

PuTTY is a serial console that can be used as a simple debug console.

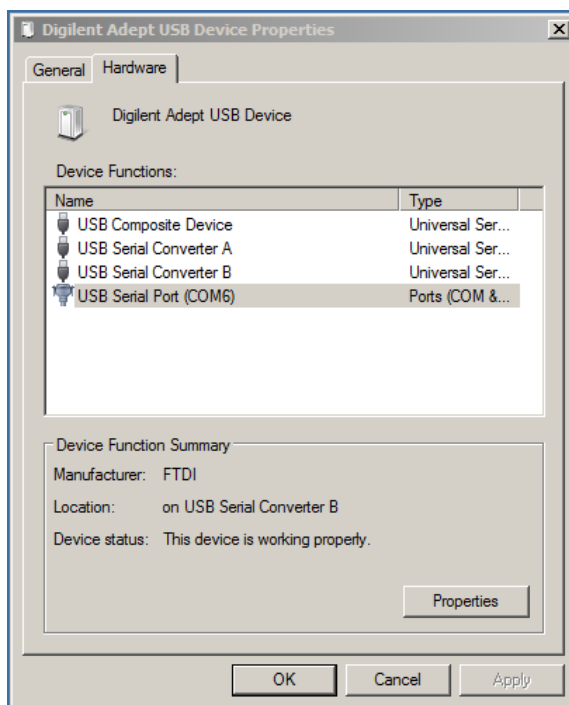
1. Download `putty.exe` from <http://www.putty.org>
2. Open the Windows **Control Panel**.
3. In the category **Hardware and Sound**, click **View devices and printers**, and **Digilent Adept USB Device**.

The **Digilent Adept USB Device Properties** windows opens.

Select the **Hardware** tab and note the COM port assigned to the USB Serial Port.

The example in [Figure 1](#) uses the COM6 port:

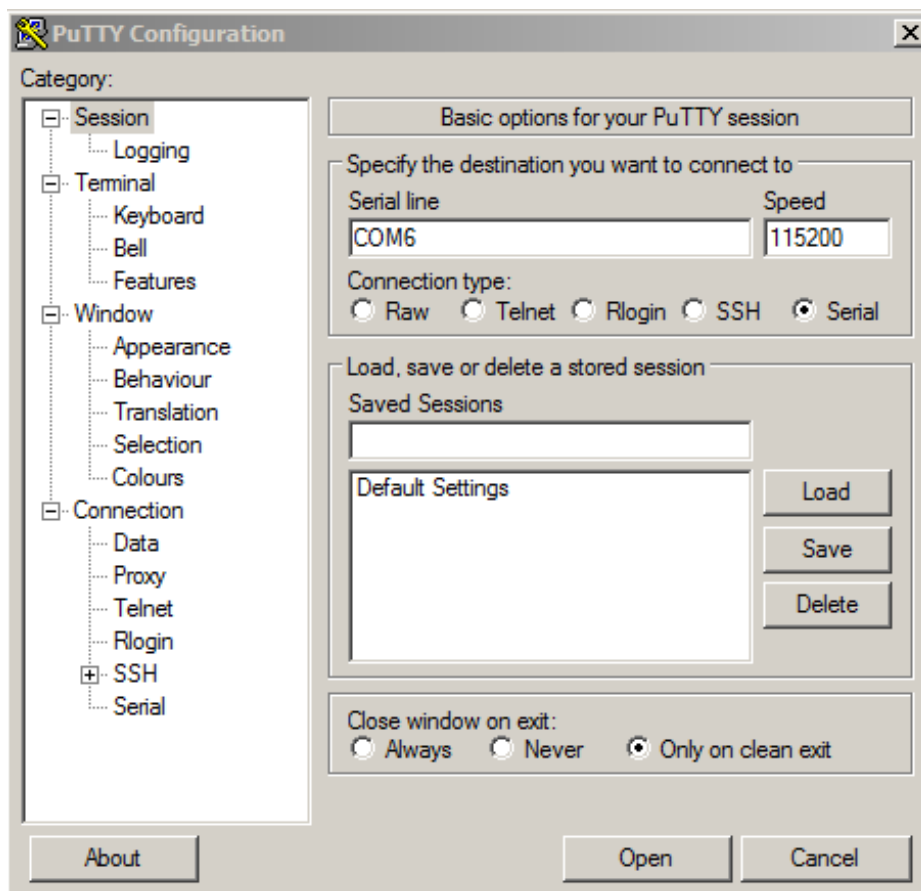
Figure 1 Identification of COM Port



4. Execute `putty.exe`.

- The **PuTTY Configuration** window appears.
5. Set the **Connection type** to **Serial**.
 6. Enter the name of the COM port in the **Serial line** field.
 7. Set the **Speed** field to **115200** as shown in [Figure 2](#).

Figure 2 PuTTY Configuration



8. Click **Open** to launch the PuTTY terminal.

2.2.5 Starting U-Boot

By default, you press the start button on the HSDK-4xD for the ARC HS core to start executing the bootloader. After you press the start button, you see text like the following figures:

- Figure 3 shows a log from default booting, followed by U-Boot commands to initialize the HSDK-4xD board in single HS47D mode and start an application called `app.bin` that resides on the SD card.
- Figure 4 shows a log from default booting followed by U-Boot commands to initialize the HSDK-4xD board in HS48x4 mode and start Linux from an image on the SD card (`uImage`).

Figure 3 Application Boot with ARC HS47D Initialization

```

COM19 - PuTTY
*****
**      Synopsys, Inc.      **
**   ARC HS Development Kit   **
*****
** IC revision: Rev 2.0
** Bootloader verbosity: Normal
** Starting HS Core 1
** HS Core running @ 500 MHz
fptr = 8** HS Core fetching application from SPI flash
** HS Core starting application
<debug_uart>

U-Boot 2020.01-00017-g862d528623a (Mar 25 2020 - 13:27:20 +0300)

CPU:   ARC HS v4.0 at 500 MHz
Model: snps,hsdk-4xd
Board: Synopsys ARC HS4x/4xD Development Kit
DRAM:  1 GiB
Relocation Offset is: 3ef8b000
MMC:   mmc0@f000a000: 0
Loading Environment from FAT... OK
In:    serial0@f0005000
Out:   serial0@f0005000
Err:   serial0@f0005000
Clock values are saved to environment
Net:
Warning: ethernet@f0008000 (eth0) using random MAC address - 5e:df:01:a9:c3:9a
eth0: ethernet@f0008000
hsdk-4xd# run hsdk_hs47d
hsdk-4xd# hsdk_init
CPU start mask is 0x1
hsdk-4xd# fatload mmc 0:1 0x80000000 app.bin
79312 bytes read in 13 ms (5.8 MiB/s)
hsdk-4xd# setenv core_entry_0 0x80000200
hsdk-4xd# hsdk_go
** selftest - Application **
Mar 24 2020
10:17:22
IC revision (dd-mm-yyyy): 6-2-2019, Time (hh:mm): 14:2
Core: ARCHS47Dx4_1
Operating mode = OP_MODE_SLOW_DDR_333:
- ARC clock freq      = 500 MHz
- ARC-AXI clock divider = 1
- AXI clock freq      = 400 MHz
- DDR clock freq      = 333 MHz
- TUN clock freq      = 49 MHz
- HDMI clock freq     = 27 MHz
application finished

```

Figure 4 Linux Boot with ARC HS48x4 Initialization

```

COM19 - PuTTY
*****
**      Synopsys, Inc.      **
**   ARC HS Development Kit   **
*****
** IC revision: Rev 2.0
** Bootloader verbosity: Normal
** Starting HS Core 1
** HS Core running @ 500 MHz
fptr = 8** HS Core fetching application from SPI flash
** HS Core starting application
<debug_uart>

U-Boot 2020.01-00017-g862d528623a (Mar 25 2020 - 13:27:20 +0300)

CPU:   ARC HS v4.0 at 500 MHz
Model: snps,hsdk-4xd
Board: Synopsys ARC HS4x/4xD Development Kit
DRAM:  1 GiB
Relocation Offset is: 3ef8b000
MMC:   mmc0@f000a000: 0
Loading Environment from FAT... OK
In:    serial0@f0005000
Out:   serial0@f0005000
Err:   serial0@f0005000
Clock values are saved to environment
Net:
Warning: ethernet@f0008000 (eth0) using random MAC address - c2:15:73:94:aa:8d
eth0: ethernet@f0008000
hsdk-4xd# run hsdk_hs48x4
hsdk-4xd# hsdk_init
CPU start mask is 0xf
hsdk-4xd# hsdk_clock set cpu_freq 1000 axi_freq 800 tun_freq 150
Set clocks to values specified in args
hsdk-4xd# hsdk_clock print
HSDK: clock 'cpu-clk' rate 1000 MHz
HSDK: clock 'tun-clk' rate 150 MHz
HSDK: clock 'axi-clk' rate 800 MHz
HSDK: clock 'ddr-clk' rate 334 MHz
hsdk-4xd# fatload mmc 0:1 0x82000000 uImage
22073325 bytes read in 4137 ms (5.1 MiB/s)
hsdk-4xd# setenv loadaddr 0x82000000
hsdk-4xd# bootm
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-5.4.0-00023-gac57d0971774-
   Image Type:   ARC Linux Kernel Image (gzip compressed)
   Data Size:    22073261 Bytes = 21.1 MiB
   Load Address: 90000000
   Entry Point:  910cc000
   Verifying Checksum ... OK
   Uncompressing Kernel Image
CPU start mask is 0xf

Starting kernel ...

```

2.3 Location of Components On HSDK-4xD

Figure 5 and Figure 6 show the placement of various components on the HSDK-4xD board.

Figure 5 HSDK-4xD Components- Top View

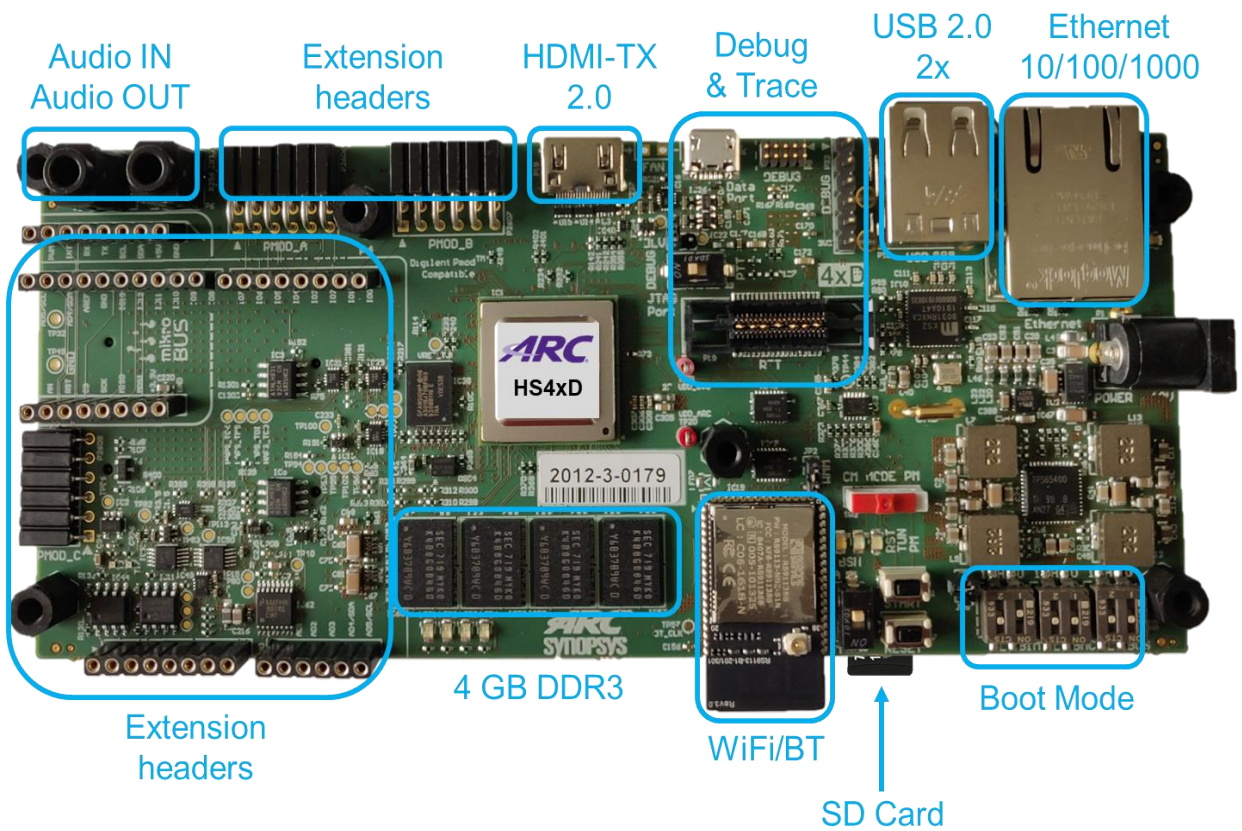
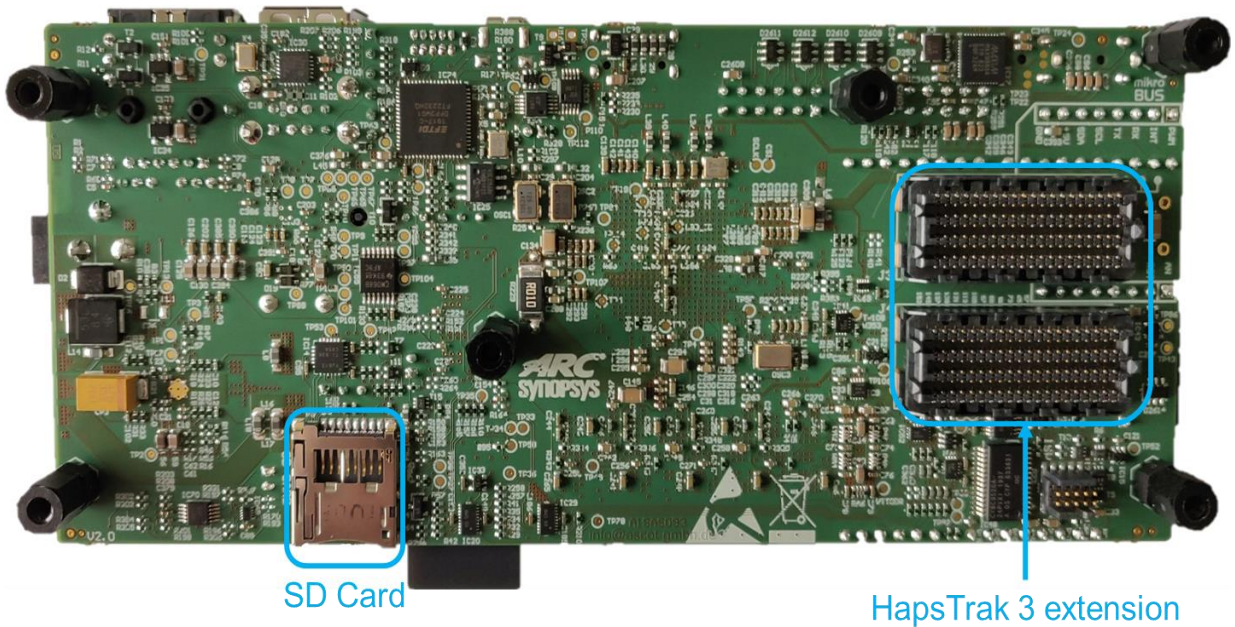


Figure 6 HSDK-4xD Components: Bottom View



2.4 Software Packages

See the <http://embarc.org/> portal for information on the available software packages for the ARC HSDK-4xD.

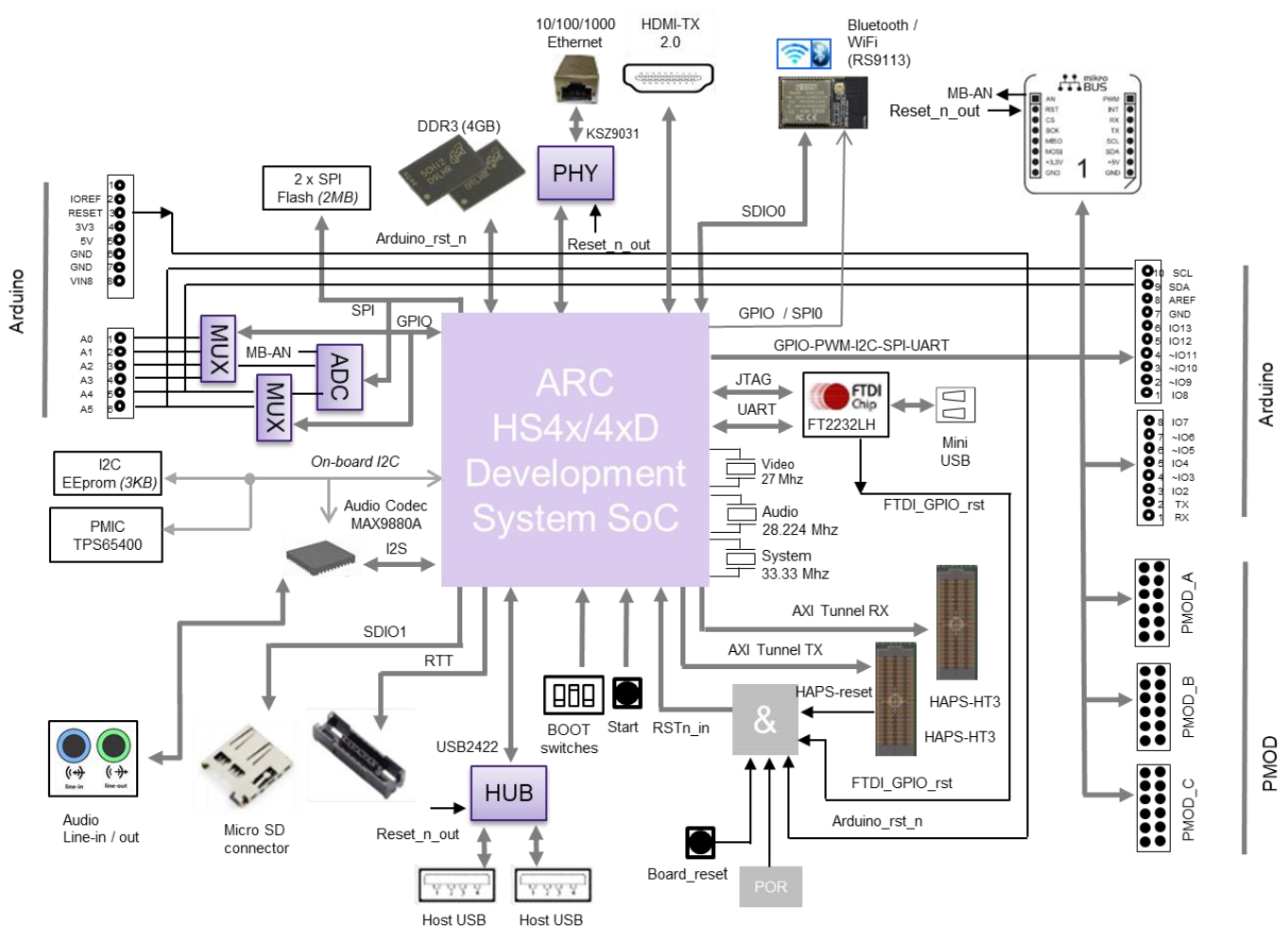
Direct links to the software release is available here:

- Linux software release for ARC HSDK-4xD are available here:
<https://github.com/foss-for-synopsys-dwc-arc-processors/buildroot/releases>
- embARC software releases for ARC HSDK-4xD are available here:
https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_osp/releases

Additional documentation on how to get started can be found here:
<https://github.com/foss-for-synopsys-dwc-arc-processors/ARC-Development-Systems-Forum/wiki/ARC-Development-Systems-Forum-Wiki-Home#arc-hs-development-kit-1>

3.1 Overview of ARC HSDK-4xD

Figure 7 ARC HSDK-4xD Block Diagram



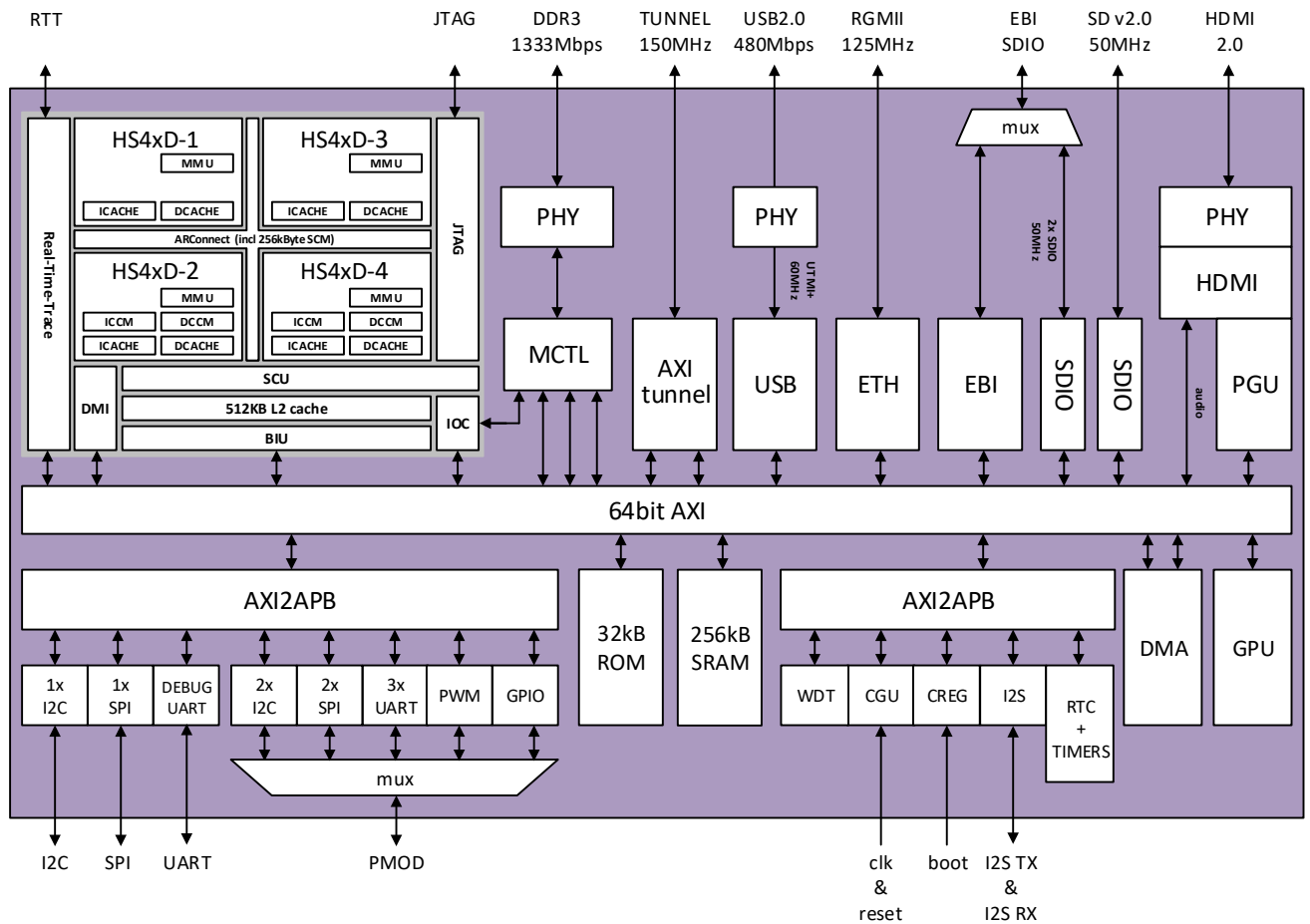
DesignWare ARC HS4xD Development Kit contains the following components:

- ARC HS4xD Development System SoC
 - Quad Core ARC HS4x/4xD
 - DDR3 memory interface
 - GPU
 - USB, Ethernet, SDIO
 - Several APB Peripherals

- Memory
 - DDR3-1333 (4 GB)
 - NOR Flash (1 MB)
 - 2x SPI Flash (2 MB)
 - I2C EEPROM (3 KB)
- Interfaces
 - HDMI-TX 2.0
 - USB2 (2x)
 - Ethernet (10/100/1000)
 - Audio line in/out
 - USB Data port (JTAG/UART)
 - Micro- SD Card
 - WiFi/Bluetooth module
 - ADC (6 channels)
 - RTT Nexus, JTAG
- Extensions
 - AXI Tunnel (32-bit, max 150 MHz)
 - Arduino Interface headers (UNO R3 compatible)
 - mikroBUS headers
 - Pmod Interfaces (3x)

3.2 Overview of ARC HS4xD Development System SoC

Figure 8 ARC HS4xD Development System SoC Top-Level Diagram



The ARC HS4xD Development System SoC provides the following main features:

- Flexible, customizable IC architecture
 - Configurable/programmable boot scenarios
 - Configurable/programmable memory map
- DesignWare ARC HS4xD quad-core @ 1GHz
 - 64 KB instruction cache
 - 64 KB data cache
 - 256 KB ICCM (two cores)
 - 256 KB DCCM (two cores)
 - Memory Management Unit
 - Physical Address Extension (PAE)
 - 256 KB Cluster Shared Memory
 - 512 KB L2 cache
 - Support for I/O coherency

- Support for ARC Real-Time Trace
- Think Silicon NemaT Graphics Processing Unit @ 400 MHz
- Flexible clock generation
 - System locks
 - 33 MHz system reference clock input
 - PLL for DDR clock
 - PLL for TUNNEL clock
 - PLL for ARC clock
 - PLL for all other system clocks
 - Audio
 - Audio reference clock input (24.576 MHz)
 - Integer divider(s) for audio serial clock
- 256 KB SRAM
- DDR3 interface
 - Maximum speed grade DDR-1333 (400 MHz)
 - 32-bit data width
 - Maximum row address width of 16-bits
 - Maximum bank address width: 3-bits
 - Maximum two memory ranks
 - Maximum supported DDR memory size is 4 GB
- I2S TX/RX interface
- USB 2.0 Host interface
- SD card interface
- SDIO interfaces
- 10/100/1000Mbps Ethernet RGMII interface
- AXI tunnel interface
 - Source-synchronous
 - Maximum frequency of 150 MHz
 - Maximum data throughput of 600 MB per second
- UART interfaces
- I2C interfaces
- SPI interfaces
- PWM interfaces
- JTAG interface

3.3 Clocks and Resets

3.3.1 Clocks

The ARC HS4xD Development System SoC uses a single 33 MHz reference clock from which all the system clocks are generated. The clock generation is centralized in the Clock Generation Unit (CGU). The CGU implements several PLLs and integer dividers that allow for accurate fine-tuning of the system clocks to the desired frequency. For more details on the CGU, see [Clock Registers](#) on page 86. [Figure 9](#) shows the HSDK-4xD clock architecture.

Figure 9 ARC HSDK-4xD Clock Architecture

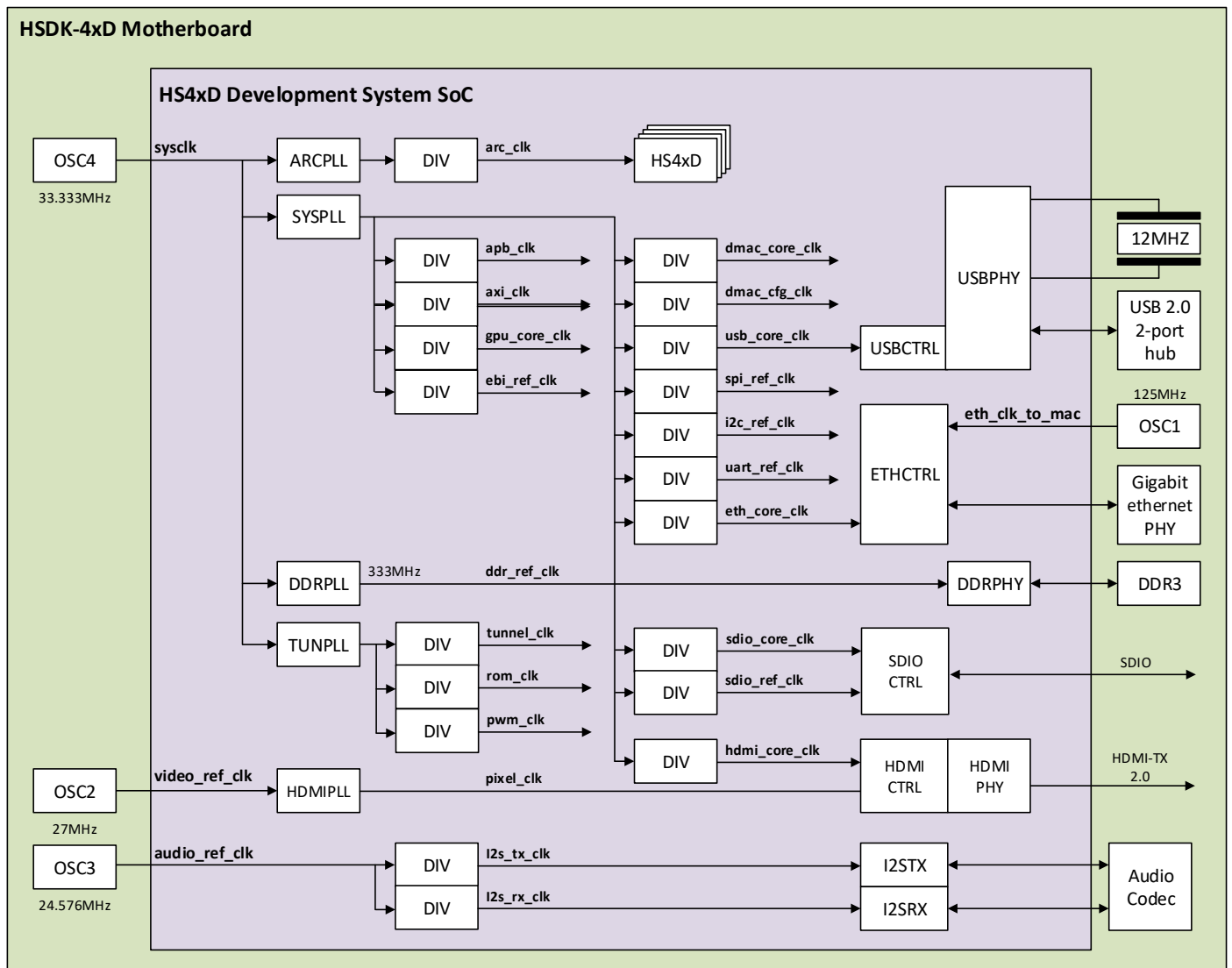


Table 1 lists the summary of all the clocks and their sources.

Table 1 Overview of ARC HSDK-4xD Clock Components

HS IC Clock	Clock Source	Value After U-Boot	Description
arc_clk	sysclk	500	Clock for ARC HS4xD quad-core (including ARC RTT and ARConnect)
apb_clk	sysclk	200	Clock for APB peripherals
axi_clk	sysclk	400	Clock for AXI network
eth_core_clk	sysclk	400	Core clock for Ethernet MAC
usb_core_clk	sysclk	400	Core clock for USB-HOST controller
sdio_core_clk	sysclk	400	Core clock for SDIO controller
hdmi_core_clk	sysclk	400	Core clock HDMI
gpu_core_clk	sysclk	400	Core clock GPU
dmac_core_clk	sysclk	400	Core clock for DMAC
dmac_cfg_clk	sysclk	200	Configuration clock for DMAC
sdio_ref_clk	sysclk	100	Reference clock for SD card interface
spi_ref_clk	sysclk	33.33	Reference clock for SPI baud rate. SPI reference clock must be less than or equal to the APB clock.
i2c_ref_clk	sysclk	200	Reference clock for I2C baud rate.

HS IC Clock	Clock Source	Value After U-Boot	Description
			I2C reference clock must be greater than or equal to the APB clock
uart_ref_clk	sysclk	33.33	Reference clock for UART baud rate. UART reference clock must be less than or equal to APB clock
ddr_ref_clk	sysclk	333	Reference clock for DDR controller + PHY (333 MHz for DDR-1333). This clock is driven directly by the PLL; all the integer dividers inside the CGU are bypassed.
tunnel_clk	sysclk	50	Clock for AXI tunnel
rom_clk	sysclk	150	Clock for ROM controller
pwm_clk	sysclk	75	Clock for PWM controller
timer_clk	sysclk	50	Clock for TIMERS / RTC
i2s_tx_sclk	audio_ref_clk	24.576	Audio serial clock for Tx channel
i2s_rx_sclk	audio_ref_clk	24.576	Audio serial clock for Rx channel
eth_clk	OSC1	125	Fixed clock for Ethernet MAC
video_ref_clk	OSC2	27	Fixed clock for PGU and HDMI
audio_ref_clk	OSC3	24.576	Fixed clock frequency for I2S interfaces and audio codec
sysclk	OSC4	33.333	Fixed system clock for ARC HS4xD Development System SoC

3.3.2 Reset

Figure 10 shows the top-level reset architecture of the ARC HSDK-4xD.

When the ARC HS4xD Development System SoC is in reset the `resetn_out` is asserted. This output pin resets all components on the board that have a reset input pin (for example: Gigabit Ethernet PHY, USB 2.0 2-port hub, and so on).

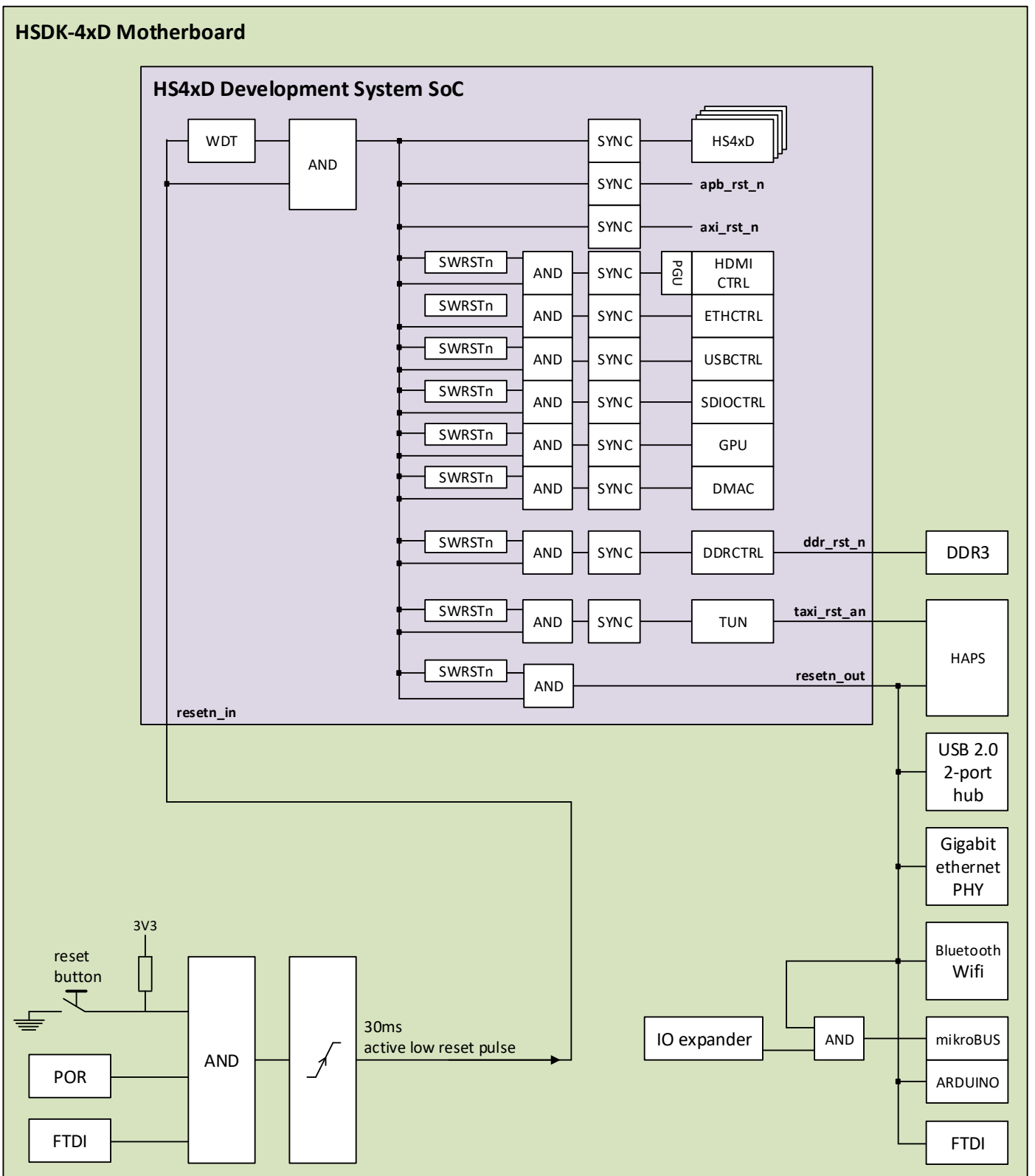
The ARC HS4xD Development System SoC has one external reset pin (`resetn_in`) that serves as an active low, hardware reset. When the external hardware reset is active, the entire chip is reset. The chip does not have an on-chip power-on-reset module so it relies on the external circuitry to keep `resetn_in` asserted low until all the chip power supplies and the system input clock are stabilized.

The reset generated by the reset button is merged with a Power on Reset circuit and the `FTDI_gpio` reset.

After the ARC HS4xD Development System SoC is out of reset, `resetn_out` is asserted resetting all the HSDK-4xD components, `resetn_out` is also routed to the mikrobus and Arduino headers and to the HapsTrak 3 connector. The reset input of the Mikrobus/Click shield is asserted by `resetn_out` but can also be asserted through a software-controllable GPIO output from the on board I2C I/O expander.

Further, an AXI tunnel reset signal `taxi_rst_an` is available on the HapsTrak 3 connector. This signal allows you to reset the AXI tunnel on the HAPS side independently from the ARC HS4xD Development System SoC reset.

Figure 10 ARC HSDK-4xD Reset Architecture



3.4 Interrupts

Figure 11 shows the top-level interrupt architecture of the ARC HSDK-4xD. The ARC HSDK-4xD distinguishes between the following three interrupt sources:

- Software (SW) interrupts: a software interrupt can be generated by writing a 1 to the corresponding interrupt bit in the in the CREG module.
- External hardware interrupts: generated by off-chip interrupt sources (for example: external host CPU). The interrupt requests are received by the GPIO module and forwarded to the ARC HS4xD quad-core.
- Internal hardware interrupts: generated by the on-chip interrupt sources (for example: I2C, UART, SPI).

Table 2 lists the interrupt mapping for the ARC HS4xD quad-core. All the above motioned interrupts sources are connected as an external common interrupt to the interrupt distribution unit (IDU). The IDU distributes the external common interrupts to the cores in the ARC HS4xD quad-core. All interrupts are active high and level sensitive unless noted differently.

Figure 11 ARC HSDK-4xD Interrupt Architecture

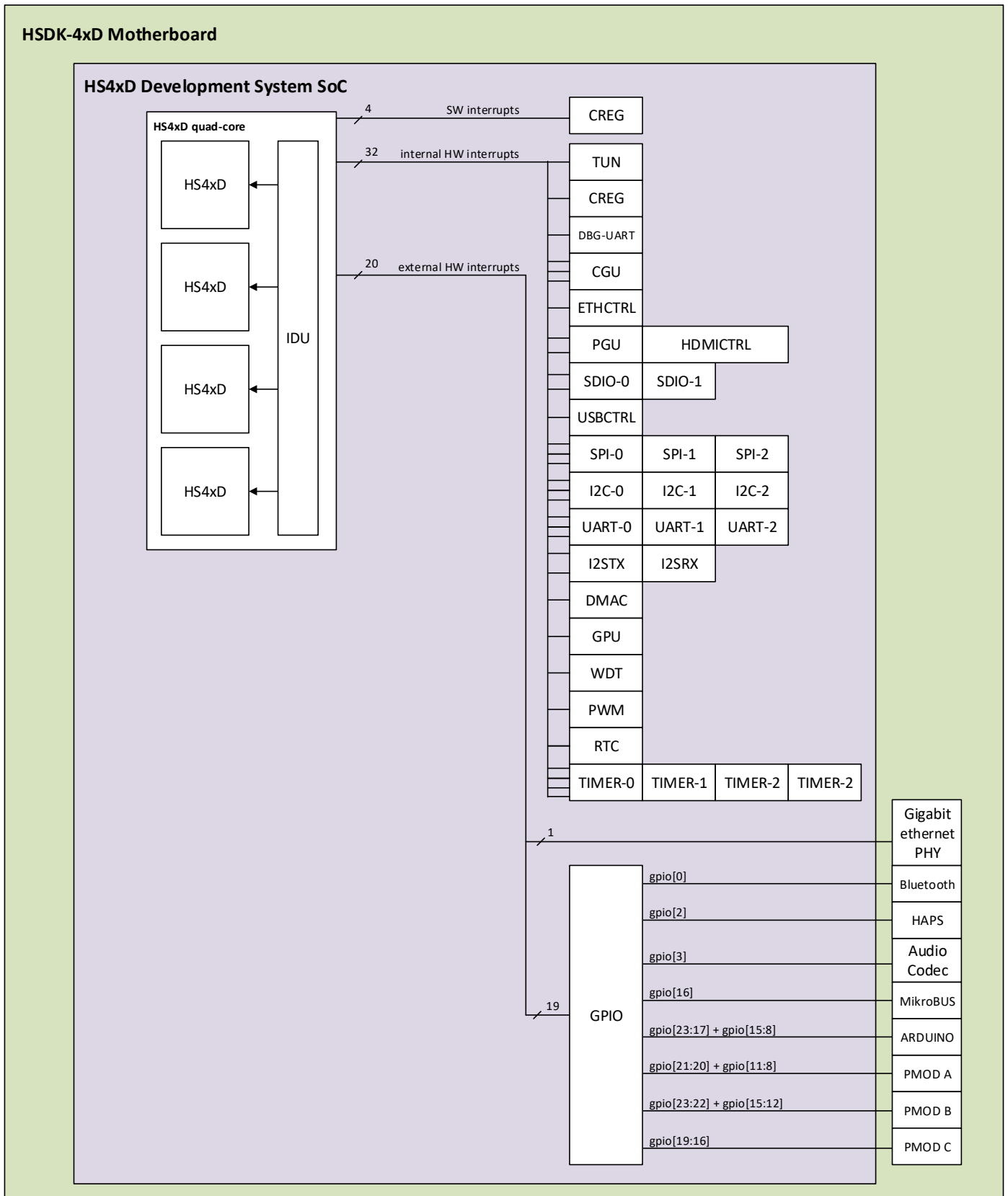


Table 2 ARC HSDK-4xD Interrupt Mapping

IRQ #		Interrupt Source	Remarks
IDU	CORE		
ARC Internal Interrupts			
	irq0	reset	
	irq1	memory error	
	irq2	instruction error	
	irq16	timer0	
	irq17	timer1	
ARC External Private Interrupts			
	irq18_a	reserved	
	irq19_a	ARConnect inter-core interrupt	
	irq20_a	reserved	
	irq21_a	reserved	
	irq22_a	reserved	
	irq23_a	reserved	
ARC External Common Interrupts (Routed through ARConnect IDU)			
cirq0_a	irq24_a	SW interrupt from CREG INT bit [0]	
cirq1_a	irq25_a	SW interrupt from CREG INT bit [1]	
cirq2_a	irq26_a	SW interrupt from CREG INT bit [2]	
cirq3_a	irq27_a	SW interrupt from CREG INT bit [3]	
cirq4_a	irq28_a	internal HW interrupt from TUNNEL	Parity error
cirq5_a	irq29_a	internal HW interrupt from CREG	Address decoder updated; edge sensitive

cirq6_a	irq30_a	internal HW interrupt from DEBUG-UART	
cirq7_a	irq31_a	internal HW interrupt from CGU	PLL locked; edge sensitive
cirq8_a	irq32_a	internal HW interrupt from CGU	PLL unlocked; edge sensitive
cirq9_a	irq33_a	internal HW interrupt from CGU	PLL lock error; edge sensitive
cirq10_a	irq34_a	internal HW interrupt from ETH MAC	
cirq11_a	irq35_a	external HW interrupt from ETH PHY	
cirq12_a	irq36_a	internal HW interrupt from SDIO0	
cirq13_a	irq37_a	internal HW interrupt from PGU	
cirq14_a	irq38_a	internal HW interrupt from HDMI	
cirq15_a	irq39_a	internal HW interrupt from USB-HOST	
cirq16_a	irq40_a	internal HW interrupt from SPI0	
cirq17_a	irq41_a	internal HW interrupt from SPI1	
cirq18_a	irq42_a	internal HW interrupt from SPI-2	
cirq19_a	irq43_a	internal HW interrupt from I2C0	
cirq20_a	irq44_a	internal HW interrupt from I2C1	
cirq21_a	irq45_a	internal HW interrupt from I2C2	
cirq22_a	irq46_a	internal HW interrupt from UART0	
cirq23_a	irq47_a	internal HW interrupt from UART1	

cirq24_a	irq48_a	internal HW interrupt from UART2	
cirq25_a	irq49_a	internal HW interrupt from I2S-TX	
cirq26_a	irq50_a	internal HW interrupt from I2S-RX	
cirq27_a	irq51_a	internal HW interrupt from DMAC	
cirq28_a	irq52_a	internal HW interrupt from GPU	
cirq29_a	irq53_a	internal HW interrupt from WDT	
cirq30_a	irq54_a	internal HW interrupt from PWM	
cirq31_a	irq55_a	reserved	
cirq56_a	irq80_a	internal HW interrupt from SDIO1	
cirq57_a	irq81_a	internal HW interrupt from TIMER1	
cirq58_a	irq82_a	internal HW interrupt from TIMER2	
cirq59_a	irq83_a	internal HW interrupt from TIMER3	
cirq60_a	irq84_a	internal HW interrupt from TIMER4	
cirq61_a	irq85_a	internal HW interrupt from RTC	
cirq32_a	irq56_a	external HW interrupt from GPIO[0]	Bluetooth interrupt of RS9113 module
cirq33_a	irq57_a	reserved	N/A, GPIO[1] is used as output
cirq34_a	irq58_a	external HW interrupt from GPIO[2]	HAPS interrupt (on HapsTrak 3 connector)
cirq35_a	irq59_a	external HW interrupt from GPIO[3]	Audio codec (MAX9880A) interrupt
cirq36_a	irq60_a	reserved	N/A, GPIO[4] is not connected

cirq37_a	irq61_a	reserved	N/A, GPIO[5] is not connected
cirq38_a	irq62_a	reserved	N/A, GPIO[6] is not connected
cirq39_a	irq63_a	reserved	N/A, GPIO[7] is not connected
cirq40_a	irq64_a	external HW interrupt from GPIO[8]	Available on Arduino and PMOD_A header
cirq41_a	irq65_a	external HW interrupt from GPIO[9]	Used on Arduino and PMOD_A header
cirq42_a	irq66_a	external HW interrupt from GPIO[10]	Available on Arduino and PMOD_A header
cirq43_a	irq67_a	external HW interrupt from GPIO[11]	Available on Arduino and PMOD_A header
cirq44_a	irq68_a	external HW interrupt from GPIO[12]	Available on Arduino and PMOD_B header
cirq45_a	irq69_a	external HW interrupt from GPIO[13]	Available on Arduino and PMOD_B header
cirq46_a	irq70_a	external HW interrupt from GPIO[14]	Available on Arduino and PMOD_B header
cirq47_a	irq71_a	external HW interrupt from GPIO[15]	Available on Arduino and PMOD_B header
cirq48_a	irq72_a	external HW interrupt from GPIO[16]	Available on MikroBUS and PMOD_C header
cirq49_a	irq73_a	external HW interrupt from GPIO[17]	Available on Arduino and PMOD_C header
cirq50_a	irq74_a	external HW interrupt from GPIO[18]	Available on Arduino and PMOD_C header
cirq51_a	irq75_a	external HW interrupt from GPIO[19]	Available on Arduino and PMOD_C header
cirq52_a	irq76_a	external HW interrupt from GPIO[20]	Available on Arduino and PMOD_A header
cirq53_a	irq77_a	external HW interrupt from GPIO[21]	Available on Arduino and PMOD_A header
cirq54_a	irq78_a	external HW interrupt from GPIO[22]	Available on Arduino and PMOD_B header
cirq55_a	irq79_a	external HW interrupt from GPIO[23]	Available on Arduino and PMOD_B header

cirq62_a	irq80_a	reserved	
cirq63_a	irq81_a	reserved	

3.5 Debug and Trace

The HSDK-4xD offers a rich set of debug and trace options:

Debug:

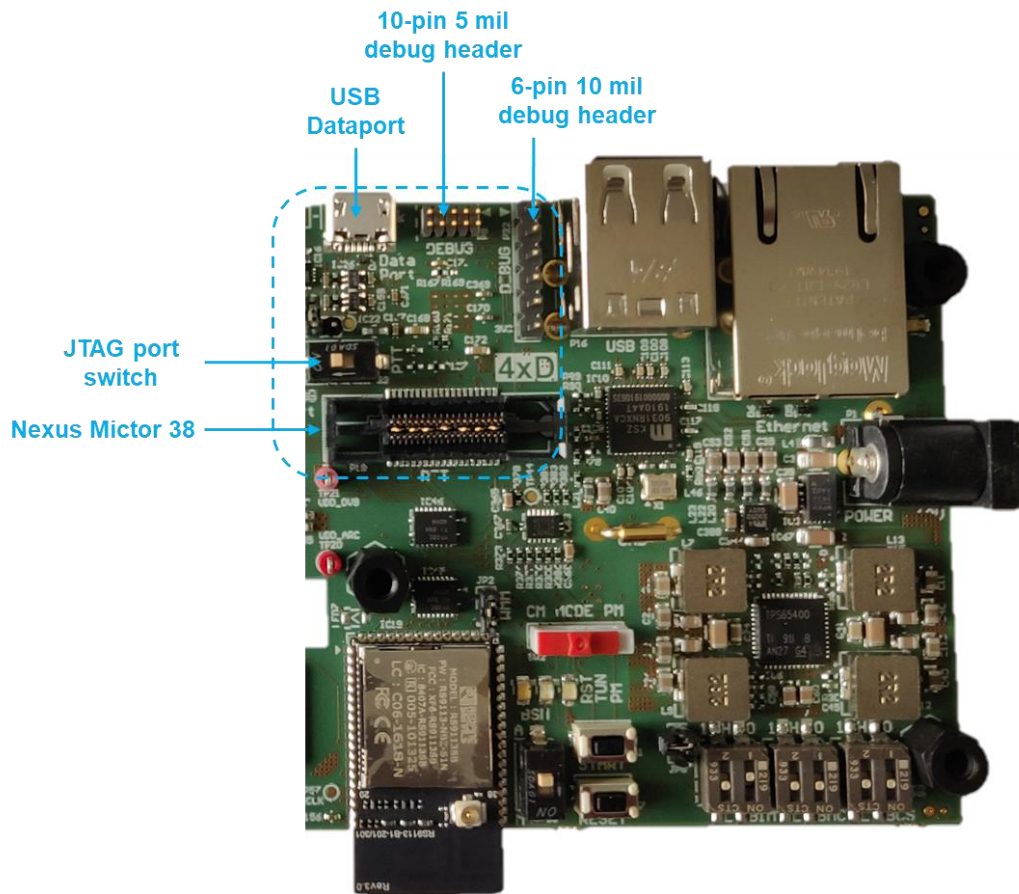
- USB cable connected to on-board 2-channel FTDI chip
 - Compatible with MetaWare debugger and GDB
- Ashling Opella-XD, Lauterbach Trace-32, Digilent HS1 and 2 support through:
 - A 10-pin 5 mil debug header and an adapter cable
 - A 6-pin 10 mil debug header and flying leads

Trace:

- Both on-chip through DDR or off-chip through the Nexus interface into external host trace off-load is available
- Ashling Ultra-XD and Lauterbach Trace-32 are supported.

Figure 12 shows the various debug and trace headers.

Figure 12 ARC HSDK-4xD Debug and Trace Headers



3.5.1 Debug

The HSDK-4xD provides debug access through an IEEE 1149.1 JTAG port. The four ARC HS cores in the ARC HS4xD quad-core cluster are daisy-chained into a JTAG chain. In a JTAG chain, the data output from the first core becomes the data input to the second core and so forth; the control and clock signals are common to all the cores in the chain. The JTAG chain for the ARC HS4xD Development System SoC is shown in Figure 13. To distinguish between the individual cores in the JTAG chain each core has a unique JTAG IDCODE.

Figure 13 JTAG Daisy Chain

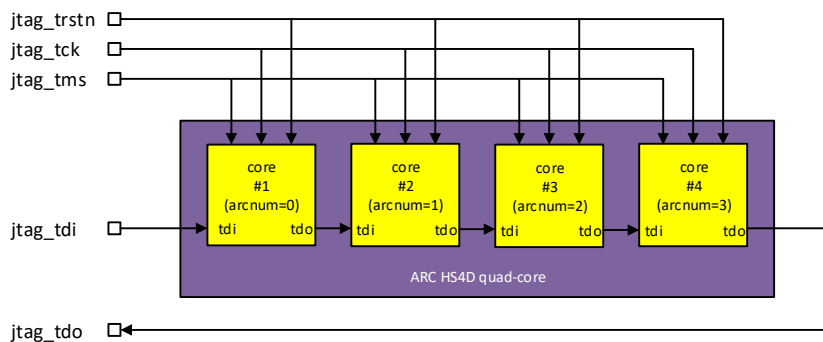


Table 3 ARC ID Codes

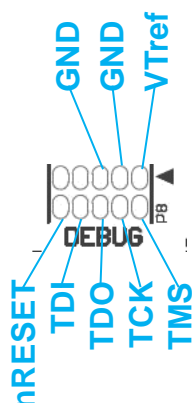
Core	ARC ID	CPUNUM in MDB
ARC HS4xD #1	0x054	1
ARC HS4xD #2	0x154	2
ARC HS4xD #3	0x254	3
ARC HS4xD #4	0x354	4

3.5.1.1 USB Dataport

The USB Dataport can be connected to your PC using the USB cable included in the product package. A USB converter from FTDI (FT2232HL) converts one channel to a serial communication protocol (UART). The other channel is converted to JTAG.

The JTAG channel offered over this Dataport is compatible with the MetaWare debugger. The serial communication channel is used as console and can be monitored using a standard hyper terminal application for example: PuTTY. For more information, see [Getting Started](#).

3.5.1.2 10-Pin Header Pinout



This header can be used to connect a standard 20-pin Ashling or Lauterbach probe header using an adapter as depicted below

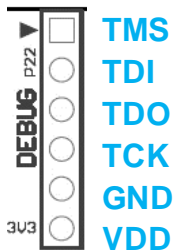
Figure 14 10-Pin to 20-Pin JTAG Adapter



The company Embedded Artists offers such a 10-pin to 20-pin JTAG Adapter.

Article code: EA-ACC-040. The adapter can be purchased from many known distributors such as Mouser, Digikey, and so on.

3.5.1.3 6-Pin Header Pinout



This 6-pin header is compatible with the standard Digilent HS1 and HS2 probes.

3.5.2 ARC Real-Time Trace

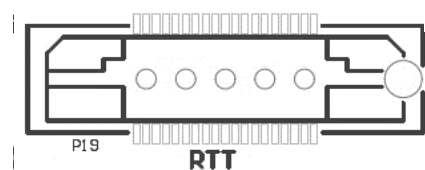
The ARC HS4xD Development System SoC implements Real-Time-Trace capability (ARC RTT). ARC RTT allows configurable/programmable monitoring of:

- Program flow and instruction execution
- Data reads and writes
- Auxiliary reads and writes
- Core register writes

Trace data can either be off-loaded from internal ARC RTT buffers to an on-chip memory (that is, DDR), or to an off-chip memory in an external host using the Nexus 5001 interface.

The Nexus 5001 interface is a 16-bit high-speed interface and for the ARC HS4xD Development System SoC supports up to 100 MHz trace clock.

3.5.2.1 Nexus Mictor 38 Interface

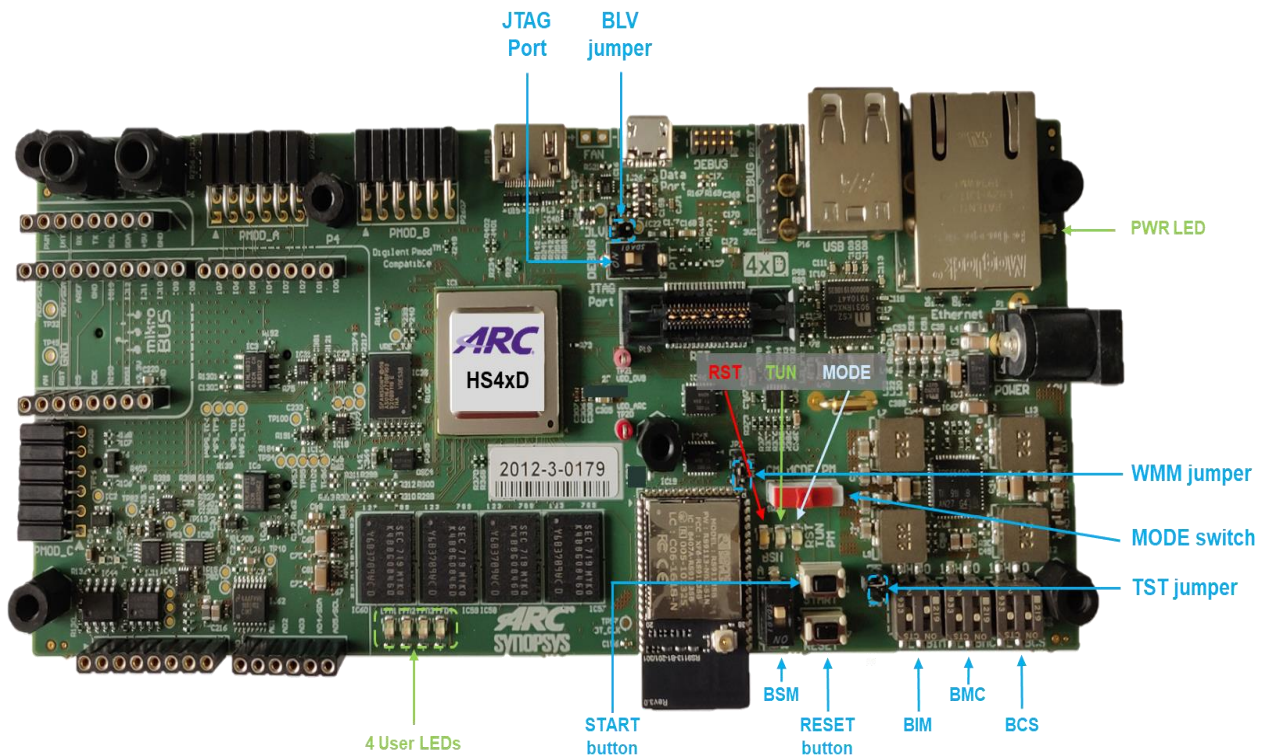


The ARC RTT interface is supported with the Ashling Ultra-XD and Lauterbach Trace-32 products.

3.6 Configuration and Boot Modes

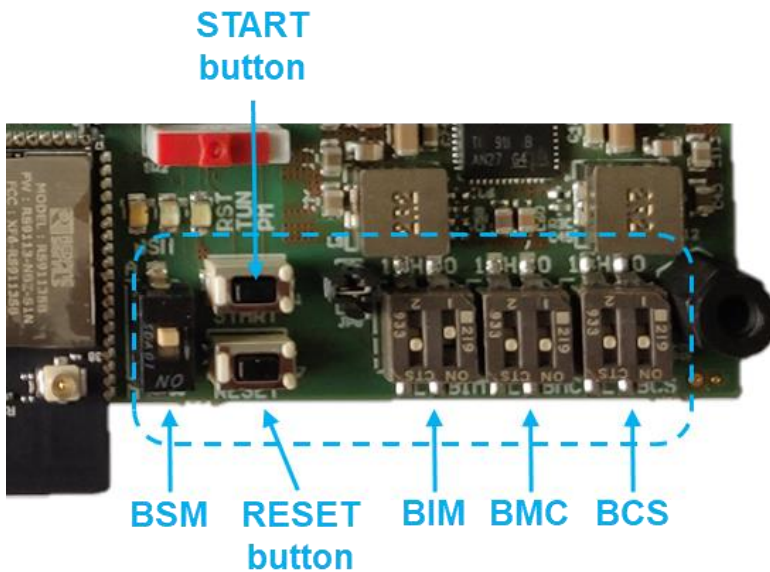
Figure 15 depicts the configuration switches, jumpers, and LEDs available on the HSDK-4xD. Besides jumpers and standard Reset and Start buttons there are also switches to control the board boot mode and debug mode. More detail of each of the jumpers and switches are explained in the following sections.

Figure 15 ARC HSDK-4xD Configuration and Boot Switches and Buttons

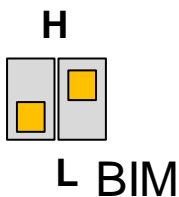


3.6.1 Boot Switches

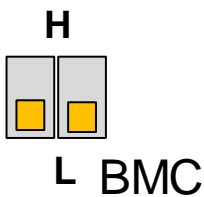
Figure 16 ARC HSDK-4xD Boot Switches



Boot Image location (BIM) Switches



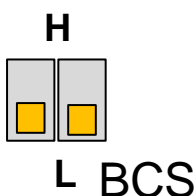
Ensure that these switches are set as depicted on the left (**default position**). This setting ensures that the pre-bootloader starts executing the U-Boot bootloader that is stored in SPI Flash. All other settings for these switches are reserved and must not be used.



Boot Multi-Core (BMC) Switches

These switches are all reserved. The setting of these switches is currently not used by the pre-bootloader. Note that selecting multi-core operating modes can be controlled directly from the U-Boot bootloader.



Boot Core Select (BCS) switches.



Ensure that these switches are set as depicted on the left (**default position**). This setting ensures that the ARC HS4xD core 1 starts executing U-Boot. All other settings for these switches are reserved and not be used.

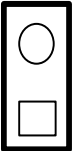
Boot Start Mode (BSM) switch.

This switch controls manual or automatic booting of the ARC HSDK-4xD:


- A**
 *Manual* mode. The HSDK-4xD only starts booting after the *START* button is pushed. This is the **default setting**
- M**
- A**
 *Automatic* mode. The HSDK-4xD automatically starts booting after *Reset*.
- M**

3.6.2 Jumpers

The HSDK-4xD includes the following jumpers: TST, BLV, and WMM.


 The TST jumper (JP8) is used for production purposes only and must not be used in the normal course of operation. **Default position:** open

JP8

 The WMM jumper (Wireless Module Mode) is used to control the mode of the RS9113 Wireless Module. If the jumper is *closed*, the module is set to operate in the hosted mode using the SDIO interface. In this mode, the networking stacks are running on the ARC HS4xD Core. If the jumper is *open*, the module operates in the embedded mode using the SPI interface, and hence the full networking stack is running on the RS9113 module itself.

Default position: closed

WMM

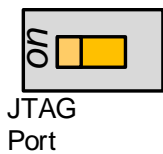
 The BLV jumper (Boot Loader Verbosity) is used to control the verbosity of the bootloader messages. If the jumper is *open*, the boot loader prints the normal boot messages without additional system/debug info. If the jumper is closed, the boot loader prints additional messages during execution.

Default position: open

BLV

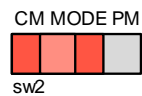
3.6.3 Other Switches

Besides the bootmode switches, the HSDK-4xD includes the following switches:



JTAG
Port

JTAG-port switch: This switch is used to control how the JTAG signals are routed. If the switch is in the on position (left), the JTAG signals are available on the 6-pin and 10-pin debug headers as well as through the USB Dataport. If the switch is in the off position (right), the JTAG signals are available on the ARC RTT connector. **Default position:** on.



sw2

MODE switch: This switch is used to select the operating mode of the HSDK-4xD. Currently only the *Core Mode*; switch in CM position is supported. **Default position:** CM

3.6.4 On-board LEDs

The HSDK-4xD includes the following LEDs: 4 user LEDs and 4 status/system LEDs. Table 4 summarizes the status/system LEDs. The user LEDs can be controlled through the I/O expander using the on-board I2C bus (see sections: [On-board I2C Control Bus](#) and [I/O Expander](#)).

Table 4 ARC HSDK-4xD Status and System LEDs

LED	Color	Description
RST	red	Reset LED, this LED turns red if the ARC HSDK-4xD is in Reset.
TUN	green	Tunnel BIST, indicates that the AXI Tunnel self-test is OK after connecting HSDK-4xD on a HAPS system through the HapsTrak 3 connectors.
PM	blue	The HSDK-4xD is in Peripheral MODE. This mode is currently not available.
PWR	green	Indicates that the power supplies for the HSDK-4xD are OK.

3.7 Memories

The HSDK-4xD features the following memories

- 4 GB DDR3-1333 memory
- 2 MB SPI Flash
 - This SPI Flash is pre-loaded with the U-Boot bootloader
- 3 KB I2C EEprom

3.8 USB Interface

The HSDK-4xD offers two USB 2.0 host ports. The ARC HS4xD Development System SoC includes a single DesignWare USB 2.0 host controller and PHY and through an on-board USB Hub (USB2422) two USB ports are available. The controller supports high-speed (480 Mbps) transfers using an EHCI Host Controller, as well as full (12 Mbps) and low (1.5 Mbps) speeds through the integrated OHCI Host Controller. See references [11] and [12] for more info on the USB Host controller and PHY.

3.9 Ethernet Interface

On the HSDK-4xD, Ethernet access is provided through a combination of the Synopsys DesignWare Ethernet MAC [10] and an external KSZ9031 triple-speed Ethernet Physical Layer from Microchip and supports the supports the 10/100/1000 Mbps link speeds. The PHY is connected to the ARC HS4xD Development System SoC using the RGMII interface. An RJ45 connector with LED indicators is used as external interface.

On the RJ45 connector, only the LED on the left side is used and this LED indicates the link speed, see Table 5. The LED on the right side is not connected.

Table 5 Ethernet Link Speed Indication

Link Speed	Color
10 Mbps	Yellow
100 Mbps	Orange
1000 Mbps	Green

3.10 SD Card Interface

The HSDK-4xD features a micro SD-card interface supporting the following micro SD cards:

- 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)

After U-Boot, the default settings operate the SD card in the SDR25 speed mode. The micro SD-card interface is controlled by SDIO0 controller.

3.11 Redpine® WiFi / Bluetooth Interface

The HSDK-4xD includes the Redpine® Signals RS9113 module for communication using Wi-Fi, Bluetooth, and Zigbee. For more information, see *Redpine Signals RS9113-WiseConnect-API-Guide-v1.7.1* in [References](#).

The SDIO host interface of the Redpine® Signals RS9113 module is connected to the SDIO1 controller. The SPI host interface is connected to the SPI0 controller. The operating mode of the RS9113 module is controlled by the WMM jumper, see [Jumpers](#) for details. By default, the RS9113 is set in SDIO host interface mode.

3.12 Audio Interface

The ARC HSDK-4xD board features stereo audio jacks for `Line Out` and `Line In`. The stereo audio input and output signals are converted using a MAX9880A stereo codecs from Maxim Integrated, which provides the interface between the stereo I²S [17] ports of the ARC HS4xD Development System SoC and drives the `Line Out` and `Line In` interfaces.

The I²S ports operate in master mode, which means that the I²S IPs inside the ARC HS4xD Development System SoC initialize and drive the I²S word select and serial clock signal. The I²S serial clocks are generated by integer dividers that run at a fixed audio reference clock frequency of 24.576 MHz.

The HSDK-4xD supports the following sampling frequencies: 16 kHz, 32 kHz, 48 kHz, 96 kHz, and 192 kHz. I²S SCLK divider settings are shown in Table 6.

Table 6 I²S SCLK Divider Settings

Audio Sample Rate				
16 kHz	32 kHz	48 kHz	96 kHz	192 kHz

24	12	8	4	2
----	----	---	---	---

3.13 On-board I2C Control Bus

The HSDK-4xD board offers an on-board I2C bus [16] to control the following on-board devices:

- Power management IC (PMIC)
- Audio Codec
- I2C EEPROM (inside I/O Expander)
- I/O Expander

An overview of the I2C bus slave addresses can be found in Table 7. The on-board I2C bus is controlled by I2C0.

Table 7 ARC HSDK-4xD On-board I2C Slave Addresses

Device Name	Description	7-bit I ² C Address
TPS65400	Power Management IC (PMIC)	1101001 = 0x69
MAX 9880A	Audio Codec	0010000 = 0x10
CY8C9520A	I/O Expander part	0100000 = 0x20
	3 KB EEPROM part	1000000 = 0x40

3.14 ADC

The HSDK-4xD board includes the 8-input 10-bit ADC108S102 from Texas Instruments [3]. The conversion rate ranges from 500 kSPS to 1 MSPS. The analog input range is 0 to 5 Volt. The analog input values are read using the SPI0 peripheral using SPI chip select 1. Table 8 lists the various ADC channels and their usage.

Table 8 ADC Channel Usage

ADC IN	Usage
0	Arduino AD0
1	Arduino AD1
2	Arduino AD2
3	Arduino AD3

4	Arduino AD4
5	Arduino AD5
6	mikroBUS AN
7	not used

3.15 I/O Expander

The HSDK-4xD board includes a CY8C9520A I/O expander from Cypress [4]. The I/O expander offers additional GPIO signals and board control signals and can be accessed through the on-board I2C bus, see section [On-board I2C Control Bus](#). Table 9 gives an overview of relevant I/O signals.

Table 9 I/O Expander I/O Overview

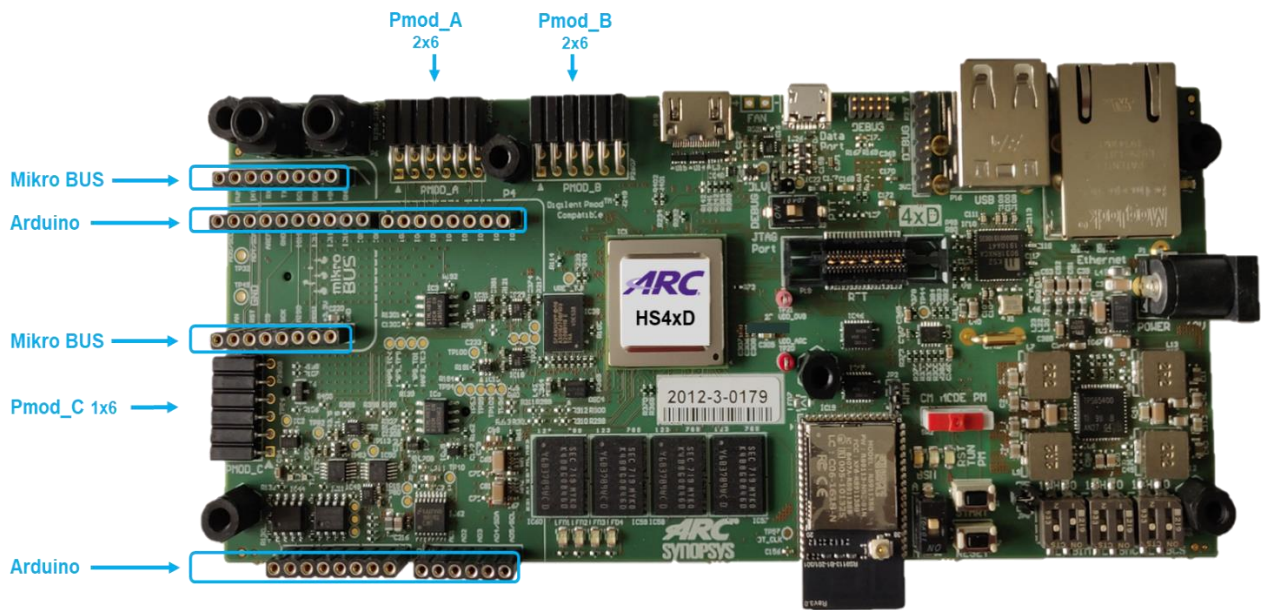
Port0	Usage	Port1	Usage
bit0	RS9113 Bluetooth I2S RX enable (active low)	bit0..3	reserved
bit1	mikroBUS Reset (active low)	bit4	On-board user LED 1
bit2	GPIO for Arduino AD0	bit5	On-board user LED 2
bit3	GPIO for Arduino AD1	bit6	On-board user LED 3
bit4	GPIO for Arduino AD2	bit7	On-board user LED 4
bit5	GPIO for Arduino AD3		
bit6,7	Reserved		

3.16 Extension Interfaces

To build your application context around the ARC HSDK-4xD, the following peripheral module standards are supported:

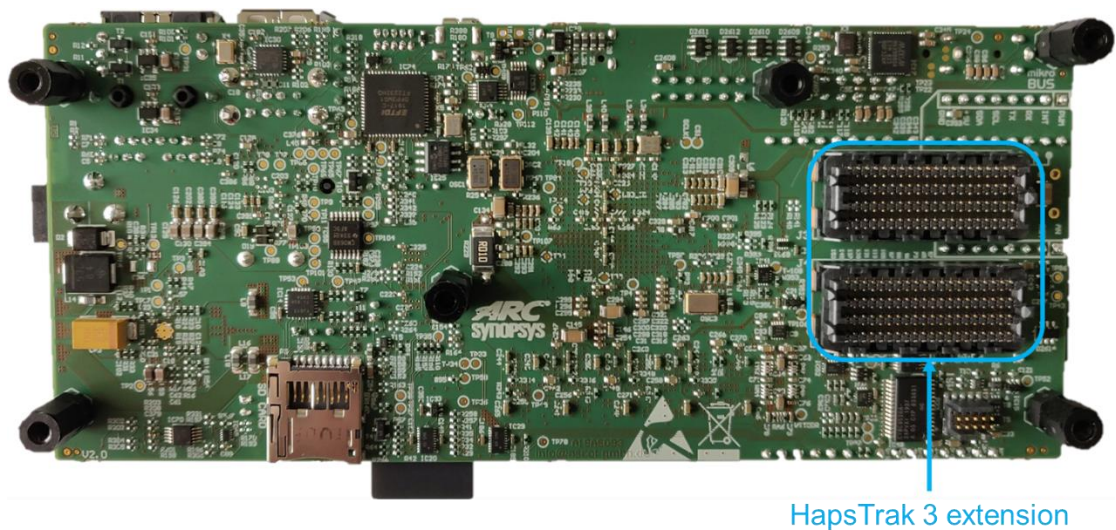
- Digilent Pmod™ (3x)
- MikroBUS (1x)
- Arduino (1x)

Figure 17 ARC HSDK-4xD Peripheral Extension Interfaces



Additionally, the ARC HSDK-4xD features two HapsTrak 3 connectors that allow you to connect the ARC HSDK-4xD to a HAPS prototyping system. See [HapsTrak3 Extension](#) on page 51 for more details regarding this extension option.

Figure 18 ARC HSDK-4xD HAPS Extension Interface



3.16.1 Digilent Pmod™

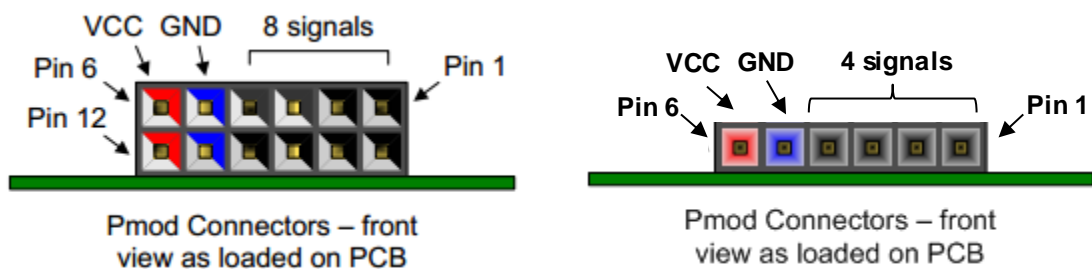
The HSDK-4xD features two 12-pin Pmod connectors P_{mod_A} and P_{mod_B} and one 6-pin Pmod connector P_{mod_C} . The functionality of the Pmod connectors is programmable and includes GPIO [6], UART [14], SPI [15], I2C [16] and PWM. The available options are summarized in Table 10. The available options are limited if the Pmod header is used in conjunction with a MikroBUS Click or Arduino shield. Multiplexing is controlled by software using the $PMOD_MUX_CTRL$ register (see [Mux](#)). After a reset, all ports are configured as GPIO inputs [6].

Table 10 Available Protocol Options

		PMOD_A			PMOD_B			PMOD_C		
	Mode	PMOD Only	PMOD+ Click	PMOD+ Arduino	PMOD Only	PMOD+ Click	PMOD+ Arduino	PMOD Only	PMOD+ Click	PMOD+ Arduino
Protocol	gpio	x	x		x	x		x	x	x
	uart	x	x	x	x	x	x	x		
	spi	x	x	x	x			x		
	i2c	x	x	x	x	x	x	x	x	x
	pwm	x	x		x	x		x		

The location of the pins on the Pmod connectors is shown in [Figure 19](#). Detailed pin descriptions depending on the pin multiplexer settings are provided in the subsequent sections.

Figure 19 Pinout Diagram of the Pmod_A, Pmod_B and Pmod_C Connectors



3.16.1.1 Pmod_A Connector

Table 11 lists the pin assignment of valid protocols that can be multiplexed on the Pmod_A connector. The **GPIO** column is the default assignment after Reset.

Table 11 Pin Description of the Pmod_A Connector

Pin	GPIO	UART	SPI	I2C	PWM_1	PWM_2
A1	gpio[8]	uart2_cts	spi1_cs[1]	gpio[8]	gpio[8]	pwm_ch[2]
A2	gpio[9]	uart2_txd	spi1_mosi	gpio[9]	pwm_ch[4]	gpio[9]
A3	gpio[10]	uart2_rxd	spi1_miso	i2c1_scl	gpio[10]	gpio[10]
A4	gpio[11]	uart2_rts	spi1_clk	i2c1_sda	gpio[11]	gpio[11]
A5	GND	GND	GND	GND	GND	GND
A6	3V3	3V3	3V3	3V3	3V3	3V3
A7	gpio[20]	gpio[20]	gpio[20]	gpio[20]	gpio[20]	gpio[20]
A8	gpio[21]	gpio[21]	gpio[21]	gpio[21]	gpio[21]	gpio[21]
A9	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
A10	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
A11	GND	GND	GND	GND	GND	GND
A12	3V3	3V3	3V3	3V3	3V3	3V3

3.16.1.3 Pmod_B Connector

Table 12 lists the pin assignment of valid protocols that can be multiplexed on the Pmod_B connector. The **GPIO** column is the default assignment after Reset.

Table 12 Pin Description of the Pmod_B Connector

Pin	GPIO	UART	SPI	I2C	PWM_1	PWM_2
B1	gpio[12]	uart0_cts	spi2_cs[1]	gpio[12]	gpio[12]	pwm_ch[0]
B2	gpio[13]	uart0_txd	spi2_mosi	gpio[13]	pwm_ch[6]	gpio[13]
B3	gpio[14]	uart0_rxd	spi2_miso	i2c2_scl	gpio[14]	gpio[14]
B4	gpio[15]	uart0_rts	spi2_clk	i2c2_sda	gpio[15]	gpio[15]
B5	GND	GND	GND	GND	GND	GND
B6	3V3	3V3	3V3	3V3	3V3	3V3
B7	gpio[22]	gpio[22]	gpio[22]	gpio[22]	gpio[22]	gpio[22]
B8	gpio[23]	gpio[23]	gpio[23]	gpio[23]	gpio[23]	gpio[23]
B9	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
B10	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
B11	GND	GND	GND	GND	GND	GND
B12	3V3	3V3	3V3	3V3	3V3	3V3

3.16.1.5 Pmod_C Connector

Table 13 lists the pin assignment of valid protocols that can be multiplexed on the Pmod_C connector. The **GPIO** column is the default assignment after Reset.

Table 13 Pin Description of the Pmod_C Connector

Pin	GPIO	UART	SPI	I2C	PWM
C1	gpio[16]	uart1_txd	spi1_cs[2]	i2c1_scl	gpio[16]
C2	gpio[17]	uart1_rxd	spi1_mosi	i2c1_sda	pwm_ch[0]
C3	gpio[18]	uart2_txd	spi1_miso	i2c2_scl	gpio[18]
C4	gpio[19]	uart2_rxd	spi1_clk	i2c2_sda	gpio[19]
C5	GND	GND	GND	GND	GND
C6	3V3	3V3	3V3	3V3	3V3

3.16.2 Mikrobus

The ARC HSDK-4xD features a set of MikroBUS headers. **Figure 20** shows the relevant function assignments, fully compatible with the MikroBUS standard [2]. The MikroBUS headers enable the addition of Click boards. Click boards are developed by the company MikroElektronika (www.mikroe.com) and are a range of hundreds of add on boards for interfacing with peripheral sensors and transceivers. Click boards include wireless and wired connectivity modules, sensor modules, display modules, interface modules, and miscellaneous modules and accessories. See www.mikroe.com/click for a full list. Multiplexing to get the right function assignment on the MikroBUS headers is controlled by software using the PMOD_MUX_CTRL register (see Mux Registers).

Figure 20 MikroBus Headers

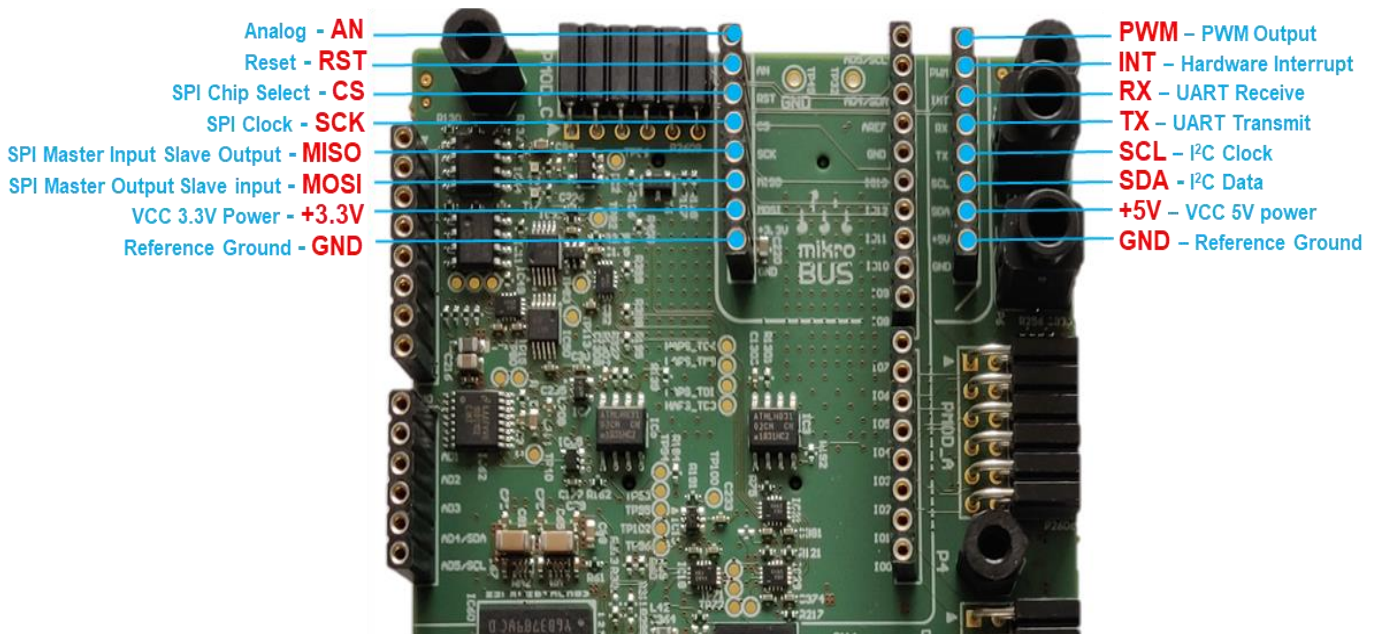


Table 14 shows the pin assignment on the I/O Multiplexer.

Table 14 Pin Description of the MikroBUS Connectors

Pin	I/O	Pin	I/O
AN	ADC VIN6*	PWM	pwm_ch[0]
RST	GPX_Port0_bit1	INT	gpio[16]
CS	spi2_cs[1]	RX	uart2_rxd
SCK	spi2_clk	TX	uart2_txd
MISO	spi2_miso	SCL	i2c2_scl
MOSI	spi2_mosi	SDA	i2c2_sda

*ADC VIN6 is available through the on-board ADC and is read though SPI0 using SPI chip select 1.

3.16.3 Arduino

The HSDK-4xD board provides an Arduino shield interface. Figure 21 shows the relevant function assignments. The Arduino shield interface is compatible with the Arduino UNO R3 with the following exceptions: 5 Volt shields are not supported, the IOREF voltage on the HSDK-4xD board is fixed to 3V3. Note that the ICSP header is also not available. Most

shields do not require this ICSP header as the SPI master interface on this ICSP header is also available on the IO10 to IO13 pins.

Figure 21 Arduino Shield Interface

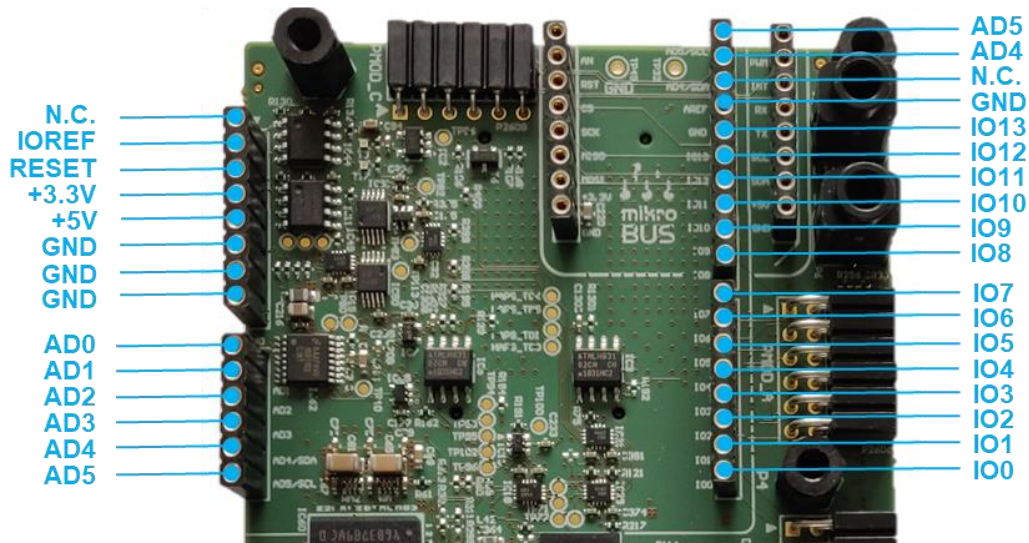


Table 15 shows the pin assignment on the I/O Multiplexer. Multiplexing is controlled by software using the `PMOD_MUX_CTRL` register (see Mux Registers). After a reset, all ports are configured as GPIO inputs.

Table 15 Pin Description of the Arduino Shield Interface

Pin	I/O-1	I/O-2	I/O-3
AD0	ADC VIN0*	GPX_port0_bit2	-
AD1	ADC VIN1*	GPX_port0_bit3	-
AD2	ADC VIN2*	GPX_port0_bit4	-
AD3	ADC VIN3*	GPX_port0_bit5	-
AD4	ADC VIN4*	gpio[18]	i2c2_sda
AD5	ADC VIN5*	gpio[19]	i2c2_scl
IO0	gpio[23]	uart2_rxd	-
IO1	gpio[22]	uart2_txd	-
IO2	gpio[16]	-	-
IO3	gpio[17]	pwm_ch[5]	-
IO4	gpio[11]		-
IO5	gpio[9]	pwm_ch[4]	-
IO6	gpio[21]	pwm_ch[3]	-
IO7	gpio[20]	-	-
IO8	gpio[10]	-	-
IO9	gpio[8]	pwm_ch[2]	-
IO10	gpio[12]	pwm_ch[0]	spi2_cs[1]
IO11	gpio[13]	pwm_ch[6]	spi2_mosi
IO12	gpio[14]	-	spi2_miso
IO13	gpio[15]	-	spi2_clk

*ADC VIN0 – 6 are available through the on-board ADC and are read through SPI0 using SPI chip select 1.

3.16.4 HapsTrak3 Extension

The DesignWare ARC HS4xD Development Kit includes two HapsTrak-3 [1] connectors to allow further extensions of the HSDK-4xD with additional FPGA resources using the

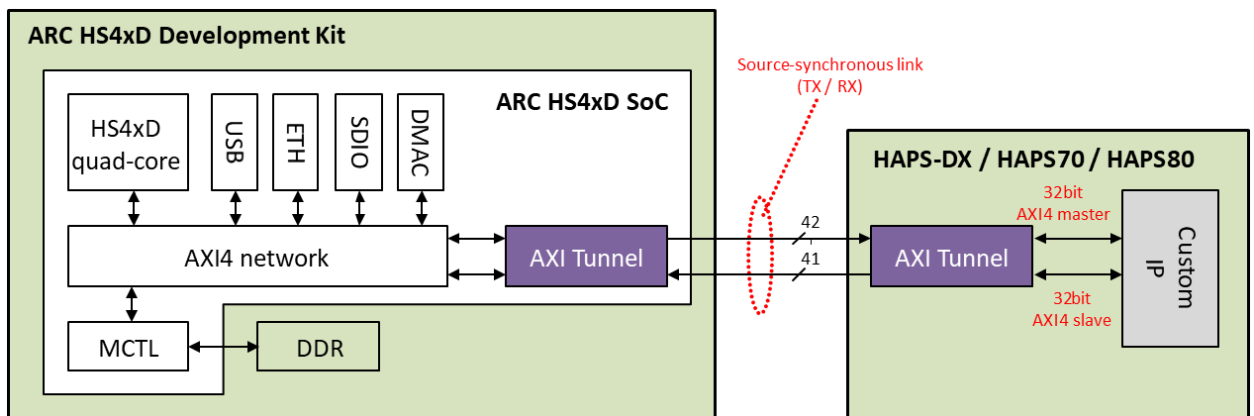
HAPS family of prototyping solutions. This interface allows adding system extensions at various levels of complexity. Main features of the extension connector are:

- Carried signals:
 - 82 AXI tunnel signals, carrying the AXI address and data
 - Tunnel Reset, to reset the AXI Tunnel
 - Tunnel BIST, connected to a LED
 - HAPS Resetrn-Out
 - Interrupt, connected to an HSDK-4xD GPIO input.
- Match trace length connections, as the 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 signal mapping, see appendix A.1.

In [Figure 22](#), a typical usage of the AXI tunnel to add custom IP into the HSDK-4xD context is depicted. Note that the AXI Tunnel provides a Bi-directional 32-bit AXI4 tunnel running at maximum 150 MHz, so total available bandwidth is 600 MB/s.

Figure 22 ARC HSDK-4xD HAPS Extension: Typical Use-case



Users that like to use other FPGA boards that feature an FMC connector can use an HapsTrak-3 to FMC adapter to connect the HSDK-4xD to a 3rd party FPGA featuring FMC extension headers.

4.1 Memory Map

Figure 23 shows the memory map overview depending on the ARC HSDK-4xD Core boot type.

Figure 23 ARC HSDK-4xD Memory Map (After U-Boot)

				hs48					
				address offset	slave				
				0x1_F000_0000	DDR [7]	PAE region			
				0x1_E000_0000	DDR [6]				
				0x1_D000_0000	DDR [5]				
				0x1_C000_0000	DDR [4]				
				0x1_B000_0000	DDR [3]				
				0x1_A000_0000	DDR [2]				
				0x1_9000_0000	DDR [1]				
				0x1_8000_0000	DDR [0]				
				0x1_7000_0000	DDR [15]				
				0x1_6000_0000	DDR [14]				
				0x1_5000_0000	DDR [13]				
				0x1_4000_0000	DDR [12]				
				0x1_3000_0000	DDR [11]				
				0x1_2000_0000	DDR [10]				
				0x1_1000_0000	DDR [9]				
hs45D			hs47D						
address offset	slave	address offset	slave	address offset	slave				
0x0_F000_0000	HSDK + HAPS APB	0x0_F000_0000	HSDK + HAPS APB	0x0_F000_0000	HSDK + HAPS APB	HW IOC region			
0x0_E000_0000	DDR[6]	0x0_E000_0000	DDR[6]	0x0_E000_0000	DDR[6]				
0x0_D000_0000	DDR [5]	0x0_D000_0000	DDR [5]	0x0_D000_0000	DDR [5]				
0x0_C000_0000	DDR [4]	0x0_C000_0000	DDR [4]	0x0_C000_0000	DDR [4]				
0x0_B000_0000	DDR [3]	0x0_B000_0000	DDR [3]	0x0_B000_0000	DDR [3]				
0x0_A000_0000	DDR [2]	0x0_A000_0000	DDR [2]	0x0_A000_0000	DDR [2]				
0x0_9000_0000	DDR [1]	0x0_9000_0000	DDR [1]	0x0_9000_0000	DDR [1]				
0x0_8000_0000	DCCM	0x0_8000_0000	DDR [0]	0x0_8000_0000	DDR [0]				
0x0_7000_0000	ICCM	0x0_7000_0000	DDR [15]	0x0_7000_0000	DDR [15]				
0x0_6000_0000	DDR [14]	0x0_6000_0000	DDR [14]	0x0_6000_0000	DDR [14]				
0x0_5000_0000	DDR [13]	0x0_5000_0000	DDR [13]	0x0_5000_0000	DDR [13]				
0x0_4000_0000	DDR [12]	0x0_4000_0000	DDR [12]	0x0_4000_0000	DDR [12]				
0x0_3000_0000	DDR [11]	0x0_3000_0000	DDR [11]	0x0_3000_0000	DDR [11]				
0x0_2000_0000	DDR [10]	0x0_2000_0000	DDR [10]	0x0_2000_0000	DDR [10]				
0x0_1000_0000	DDR [9]	0x0_1000_0000	DDR [9]	0x0_1000_0000	DDR [9]				
0x0_0000_0000	DDR [8]	0x0_0000_0000	DDR [8]	0x0_0000_0000	DDR [8]				

4.1.1 APB Peripheral Address Map

Table 16 lists all the APB peripherals that are accessible through the AXI2APB bridge. The total reserved address space for the AXI2APB bridge is 256 MB. Accessing a non-existing APB peripheral results in an AXI error response.

Table 16 APB Peripheral Address Map

Peripheral Name	Base	Size	Description
CGU	0xF0000000	4 KB	Clock Generation Unit
CREG	0xF0001000	4 KB	Control registers
GPIO	0xF0003000	128Byte	General purpose I/O [6]
MCTL	0xF0004000	2 KB	DDR memory controller [9]
PUB	0xF0004800	2 KB	DDR PHY Utility Block
DEBUG-UART	0xF0005000	256Byte	Debug UART [14]
WDT	0xF0006000	256Byte	Watchdog Timer [18]
ETH-GMAC	0xF0008000	8 KB	Ethernet MA <ul style="list-style-type: none"> • 0x00008000 => control/status registers • 0x00009000 => DMA registers
SDIO0	0xF000A000	1 KB	SDIO controller [13] ; used for SD card
SDIO1	0xF000A800	1 KB	SDIO controller [13] ; used for Wifi/BT
PGU	0xF000B000	1 KB	PGU controller
TIMERS	0xF000C000	256Byte	Timers
RTC	0xF000D000	64Byte	Real Time Counter
HDMI	0xF0010000	32 KB	HDMI-TX controller
SPI0	0xF0020000	256Byte	SPI
SPI1	0xF0021000	256Byte	SPI
SPI2	0xF0022000	256Byte	SPI
I2C0	0xF0023000	256Byte	I2C
I2C1	0xF0024000	256Byte	I2C

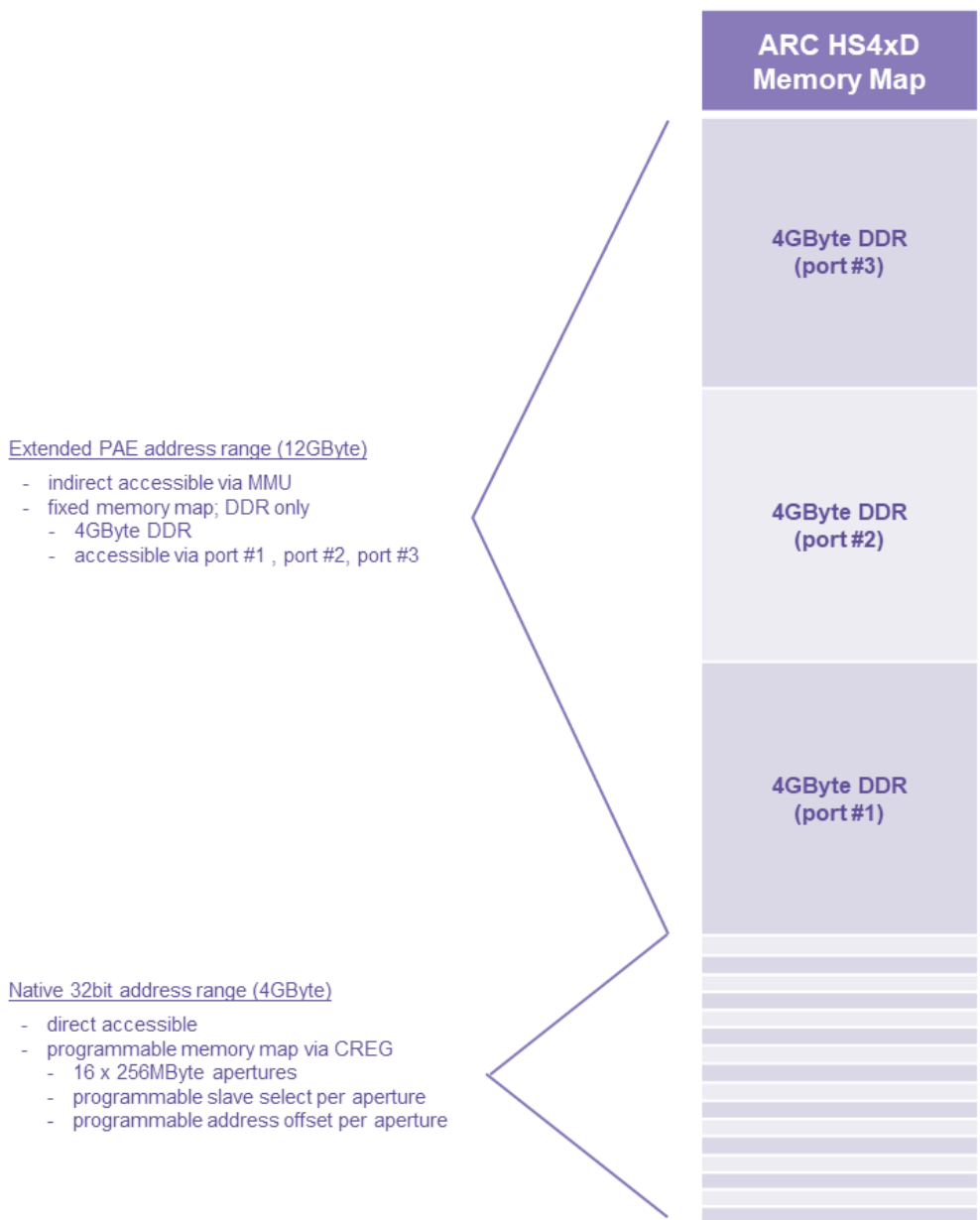
I2C2	0xF0025000	256Byte	I2C
UART0	0xF0026000	256Byte	UART
UART1	0xF0027000	256Byte	UART
UART2	0xF0028000	256 Byte	UART
I2S_TX	0xF0029000	256 Byte	I2S TX
I2S_RX	0xF002A000	256 Byte	I2S RX
PWM	0xF002B000	256 Byte	PWM
USB-HOST	0xF0040000	256 KB	USB-Host controller <ul style="list-style-type: none"> • 0x00040000 => EHCI control/status registers • 0x00060000 => OHCI control/status registers
DMAC	0xF0080000	1 KB	DMAC [19]
GPU	0xF0090000	256 KB	GPU
HAPS	0xF8000000	128 MB	HAPS APB peripherals

4.1.2 PAE

The ARC HS4xD core supports Physical Address Extension (PAE). PAE extends the physical address range beyond the core’s native 32-bit, 4 GB address range. In the ARC HS4xD Development System SoC, the PAE functionality is used to extend the physical address range from 4 GB to 16 GB.

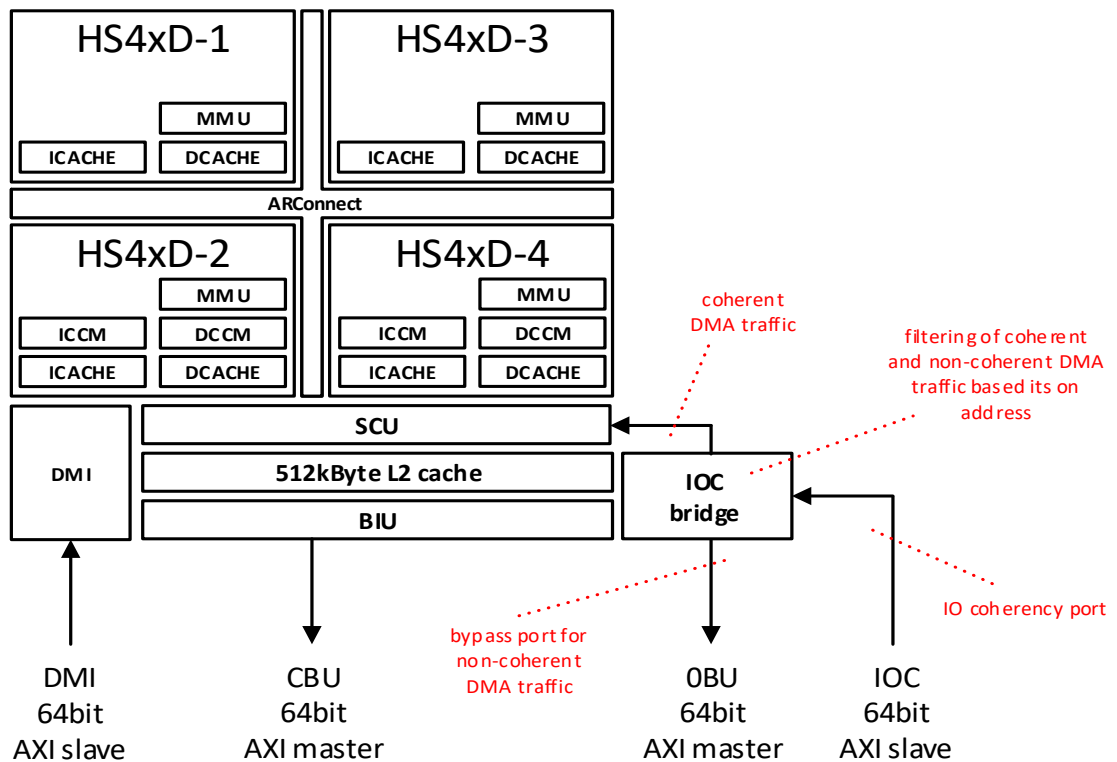
Figure 24 shows a high-level overview of the ARC HS4xD Development System SoC memory map. The memory map for the lower 4 GB of the physical address range is fully programmable through the CREG, see [Control Registers](#). The memory map for the extended part of the physical address range is fixed and used for DDR only. The additional 12 GB is divided into three 4 GB regions that all map to the same 4 GB of DDR but through different DDR ports. Thus, it is still possible to map PAE traffic to the different DDR ports.

Figure 24 ARC HS4xD Development System SoC Memory Map – High Level Overview



4.1.3 I/O Coherency

Figure 25 I/O Coherency Architecture



The ARC HS4xD quad-core is equipped with 512 KB L2 system level cache that is shared among the different ARC cores. L1 and L2 cache coherency is maintained by the Shared Coherency Unit (SCU). Problems may occur when an ARC core and a non-cached DMA client operate on a piece of shared data such as an Ethernet packet header.

Without a proper coherency mechanism, the ARC core and DMA client may have inconsistent views of the shared data. Conventional systems leave the responsibility of maintaining I/O coherency to software: the OS must ensure that the cache lines are cleaned before an outgoing DMA transfer is started and invalidated before a memory range affected by an incoming DMA transfer is accessed. This introduces some overhead for every DMA operation.



Note IO coherency: term used to indicate consistency of data that is shared between ARC cores and DMA clients.

The ARC HS4xD Development System SoC (or more specifically the ARC HS4xD quad-core) implements a hardware mechanism for maintaining I/O coherency.

Figure 25 shows the I/O coherency architecture. The ARC HS4xD quad-core provides an additional AXI slave port (aka I/O coherency port) that can be used by any DMA client that needs to operate on cacheable data that is shared with the ARC core. Traffic on the I/O coherency port is filtered by the IOC bridge based on its address. When it falls within a certain (programmable) address window, the traffic is forwarded to the SCU. When the address is outside the address window, the traffic is forwarded to its destination without being snooped.

Detailed description of I/O coherency and programming guidelines can be found in [7] and [8]. Following are some of the important programming:

- Although permitted by the programmable memory mapping feature of the ARC HS4xD Development System SoC provides, (cached) transactions from the ARC core shall never be routed to the I/O coherency port. Doing so may result in a deadlock condition.
- The ARC cores must not be engaged in coherent memory operations during times when the I/O coherency and/or cache-ability aperture registers are being modified. If this rule is violated, memory operations occurring during that time may not observe the coherency protocols as expected.

To avoid this situation, all cores must be either inoperative, or they must operate with data caches disabled, when the I/O coherency and/or cache-ability aperture register are being programmed. In addition, all cores' data caches must be flushed before their caches are disabled.

4.2 Software Interfaces

This section describes the software interfaces for the custom IP inside the ARC HS4xD Development System SoC. The software interfaces for the DesignWare IP (for example, USB-HOST) are described in the corresponding Databooks.

4.2.1 Control Registers

The global control registers inside the ARC HS4xD Development System SoC are implemented by the CREG module. The global control registers are used to control configuration settings for sub-modules that do not have a software interface (for example: the GPIO mux). Further, the CREG module implements the following functionality:

- Logic to sample boot mode configuration values from BOOT pins during power-on-reset
- Logic to generate `cpu_start` signal for ARC cores
- Logic to generate software interrupts

[Table 17](#) lists the registers for the CREG module including a brief description and their offset to the base address of the AXI2APB bridge (default = `0xF000_0000`). All registers are 32-bit wide. Read/write access to undefined registers is ignored, and an APB error response is generated. All unused bits within a register are non-writable and return zero when read. A detailed register description can be found in [Register Descriptions](#) section on page 65.

Table 17 CREG Control Register Overview

Name	Address Offset	Access ^[1]	Description
Address Decoder Registers			
ARC HS4xD (CORE)			
CREG_AXI_M_ARC_SLV0	0x1000	RW	Address decoder slave select register
CREG_AXI_M_ARC_SLV1	0x1004	RW	Address decoder slave select register
CREG_AXI_M_ARC_OFFSET0	0x1008	RW	Address decoder slave offset register
CREG_AXI_M_ARC_OFFSET1	0x100C	RW	Address decoder slave offset register
CREG_AXI_M_ARC_UPDATE	0x1014	RW1C	Address decoder update
ARC HS4xD (RTT)			
CREG_AXI_M_RTT_SLV0	0x1020	RW	Address decoder slave select register
CREG_AXI_M_RTT_SLV1	0x1024	RW	Address decoder slave select register
CREG_AXI_M_RTT_OFFSET0	0x1028	RW	Address decoder slave offset register
CREG_AXI_M_RTT_OFFSET1	0x102C	RW	Address decoder slave offset register
CREG_AXI_M_RTT_UPDATE	0x1034	RW1C	Address decoder update
AXI Tunnel			
CREG_AXI_M_TUN_SLV0	0x1040	RW	Address decoder slave select register
CREG_AXI_M_TUN_SLV1	0x1044	RW	Address decoder slave select register
CREG_AXI_M_TUN_OFFSET0	0x1048	RW	Address decoder slave offset register
CREG_AXI_M_TUN_OFFSET1	0x104C	RW	Address decoder slave offset register

CREG_AXI_M_TUN_UPDATE	0x1054	RW1C	Address decoder update
HDMI-V			
CREG_AXI_M_TUN_SLV0	0x1060	RW	Address decoder slave select register
CREG_AXI_M_TUN_SLV1	0x1064	RW	Address decoder slave select register
CREG_AXI_M_TUN_OFFSET0	0x1068	RW	Address decoder slave offset register
CREG_AXI_M_TUN_OFFSET1	0x106C	RW	Address decoder slave offset register
CREG_AXI_M_TUN_UPDATE	0x1074	RW1C	Address decoder update
HDMI-A			
CREG_AXI_M_TUN_SLV0	0x1080	RW	Address decoder slave select register
CREG_AXI_M_TUN_SLV1	0x1084	RW	Address decoder slave select register
CREG_AXI_M_TUN_OFFSET0	0x1088	RW	Address decoder slave offset register
CREG_AXI_M_TUN_OFFSET1	0x108C	RW	Address decoder slave offset register
CREG_AXI_M_TUN_UPDATE	0x1094	RW1C	Address decoder update
USB			
CREG_AXI_M_USB_SLV0	0x10A0	RW	Address decoder slave select register
CREG_AXI_M_USB_SLV1	0x10A4	RW	Address decoder slave select register
CREG_AXI_M_USB_OFFSET0	0x10A8	RW	Address decoder slave offset register
CREG_AXI_M_USB_OFFSET1	0x10AC	RW	Address decoder slave offset register
CREG_AXI_M_USB_UPDATE	0x10B4	RW1C	Address decoder update

ETH			
CREG_AXI_M_ETH_SLV0	0x10C0	RW	Address decoder slave select register
CREG_AXI_M_ETH_SLV1	0x10C4	RW	Address decoder slave select register
CREG_AXI_M_ETH_OFFSET0	0x10C8	RW	Address decoder slave offset register
CREG_AXI_M_ETH_OFFSET1	0x10CC	RW	Address decoder slave offset register
CREG_AXI_M_ETH_UPDATE	0x10D4	RW1C	Address decoder update
SDIO0 / SDIO1			
CREG_AXI_M_SDIO_SLV0	0x10E0	RW	Address decoder slave select register
CREG_AXI_M_SDIO_SLV1	0x10E4	RW	Address decoder slave select register
CREG_AXI_M_SDIO_OFFSET0	0x10E8	RW	Address decoder slave offset register
CREG_AXI_M_SDIO_OFFSET1	0x10EC	RW	Address decoder slave offset register
CREG_AXI_M_SDIO_UPDATE	0x10F4	RW1C	Address decoder update
GPU			
CREG_AXI_M_GPU_SLV0	0x1100	RW	Address decoder slave select register
CREG_AXI_M_GPU_SLV1	0x1104	RW	Address decoder slave select register
CREG_AXI_M_GPU_OFFSET0	0x1108	RW	Address decoder slave offset register
CREG_AXI_M_GPU_OFFSET1	0x110C	RW	Address decoder slave offset register
CREG_AXI_M_GPU_UPDATE	0x1114	RW1C	Address decoder update
DMAC			
CREG_AXI_M_DMACH0_SLV0	0x1120	RW	Address decoder slave select register

CREG_AXI_M_DMACH0_SLV1	0x1124	RW	Address decoder slave select register
CREG_AXI_M_DMACH0_OFFSET0	0x1128	RW	Address decoder slave offset register
CREG_AXI_M_DMACH0_OFFSET1	0x112C	RW	Address decoder slave offset register
CREG_AXI_M_DMACH0_UPDATE	0x1134	RW1C	Address decoder update
CREG_AXI_M_DMACH1_SLV0	0x1140	RW	Address decoder slave select register
CREG_AXI_M_DMACH1_SLV1	0x1144	RW	Address decoder slave select register
CREG_AXI_M_DMACH1_OFFSET0	0x1148	RW	Address decoder slave offset register
CREG_AXI_M_DMACH1_OFFSET1	0x114C	RW	Address decoder slave offset register
CREG_AXI_M_DMACH1_UPDATE	0x1154	RW1C	Address decoder update
Latency Registers			
CREG_DDR_LATENCY	0x1200	RW	Latency register for DDR
CREG_SRAM_LATENCY	0x1204	RW	Latency register for SRAM
Base Address Registers			
CREG_CSM_BASE	0x1210	RW	CSM Base Address register
CREG_DCCM_BASE	0x1214	RW	DCCM Base Address register
Timer Halt Registers			
CREG_TIMERS_HALT	0x1220	RW	TIMERS halt register
CREG_RTC_HALT	0x1224	RW	RTC halt Address register
Boot Registers			
CREG_BOOT	0x1010	RW	Boot register
CREG_CPU_START	0x1400	RW	CPU start register
Mux Registers			
CREG_GPIO_DBG_MUX	0x1480	RW	GPIO debug mux register

CREG_GPIO_MUX	0x1484	RW	GPIO mux register
CREG_DMA_MUX	0x1488	RW	DMA mux register
TUNNEL Control Registers			
CREG_TUN_CTRL	0x14A0	RW	AXI tunnel control register
CREG_TUN_STAT	0x14A4	R	AXI tunnel status register
DDR Control and Status Registers			
CREG_DDR_CTRL	0x14A8	RW	DDR control register
CREG_DDR_STAT	0x14AC	R	DDR status register
SPI Control Registers			
CREG_SPI_CS_CTRL	0x14B0	RW	SPI chip-select control register
Clock Control Registers			
CREG_GPIO_CLK_CTRL	0x14B4	RW	GPIO clock control register
CREG_ARC_CLK_CTRL	0x14B8	RW	ARC clock control register
CREG_I2S_CLK_CTRL	0x14BC	RW	I2S clock control register
RTT Reset Control Register			
CREG_RTT_RST_CTRL	0x14C0	RW	RTT reset control register
Debug Registers			
CREG_DBG_SCRATCH	0x14F8	RW	Debug scratch register
CREG_DBG_PRINT	0x14FC	RW	Debug print register (simulation only)
Interrupt Registers			
CREG_IRQ_CLR_ENABLE	0x1500	W	Interrupt clear enable register
CREG_IRQ_SET_ENABLE	0x1504	W	Interrupt set enable register
CREG_IRQ_STATUS	0x1508	R	Interrupt status register
CREG_IRQ_ENABLE	0x150C	R	Interrupt enable register
CREG_IRQ_CLR_STATUS	0x1510	W	Interrupt clear status register
CREG_IRQ_SET_STATUS	0x1514	W	Interrupt set status register

Standard Registers			
CREG_IP_SW_RESET	0x1FF0	RW1C	CREG IP software reset register
CREG_IP_VERSION	0x1FF8	R	CREG IP version register
CREG_IP_TYPE	0x1FFC	R	CREG IP type register

[1] The following access types are defined:

- RW Read/Write register
- R Read-only register
- W Write-only register
- RW1C Read-only, Write-1-to-Clear register

4.2.1.1 Register Descriptions

4.2.1.1.1 Address Decoder Registers

CREG_AXI_M_m_SLV0 – Slave Select Register (Address Offset = 0x1000 + m*0x20)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
3:0	SLV_SELO	RW	[2] *	Slave select for address aperture[0]
			0	No slave selected
			1	Slave 1 selected (=> DDR controller port #1)
			2	Slave 2 selected (=> SRAM controller)
			3	Slave 3 selected (=> AXI tunnel)
			4	Illegal
			5	Slave 5 selected (=> ROM controller)
			6	Slave 6 selected (=> AXI2APB bridge)
			7	Slave 7 selected (=> DDR controller port #2)
			8	Slave 8 selected (=> DDR controller port #3)
			9	Slave 9 selected (=> HS4xD IOC)
10	Slave 10 selected (=> HS4xD DMI) ^[1]			
7:4	SLV_SEL1	RW	[2] *	Slave select for address aperture[1]
11:8	SLV_SEL2	RW	[2] *	Slave select for address aperture[2]
15:12	SLV_SEL3	RW	[2] *	Slave select for address aperture[3]
19:16	SLV_SEL4	RW	[2] *	Slave select for address aperture[4]
23:20	SLV_SEL5	RW	[2] *	Slave select for address aperture[5]
27:24	SLV_SEL6	RW	[2] *	Slave select for address aperture[6]
31:28	SLV_SEL7	RW	[2] *	Slave select for address aperture[7]

^[1]	m = 0	ARC HS4xD (CORE)	m = 5	USB
	m = 1	ARC HS4xD (RTT)	m = 6	ETH
	m = 2	AXI Tunnel	m = 7	SDIO
	m = 3	HDMI-V	m = 8	GPU
	m = 4	HDMI-A	m = 9/10	DMAC

^[2] See Table 18 for reset values after U-Boot.

^[3] when "ARC HS4xD DMI" is selected as a slave for a certain aperture, the associated address offset in CREG_AXI_m_OFFSET0 register is used for selection of ICCM / DCCM:

- OFFSET = 3 : CORE 2 / ICCM

- OFFSET = 5 : CORE 2 / DCCM
- OFFSET = 9 : CORE 4 / ICCM
- OFFSET = 11 : CORE 4 / DCCM

CREG_AXI_M_m_SLV1 – Slave Select Register (Address Offset = 0x1004 + m*0x20)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
3:0	SLV_SEL8	RW	[2] *	Slave select for address aperture[8]
7:4	SLV_SEL9	RW	[2] *	Slave select for address aperture[9]
11:8	SLV_SEL10	RW	[2] *	Slave select for address aperture[10]
15:12	SLV_SEL11	RW	[2] *	Slave select for address aperture[11]
19:16	SLV_SEL12	RW	[2] *	Slave select for address aperture[12]
23:20	SLV_SEL13	RW	[2] *	Slave select for address aperture[13]
27:24	SLV_SEL14	RW	[2] *	Slave select for address aperture[14]
31:28	SLV_SEL15	RW	[1] *	Slave select for address aperture[15]

^[1]	m = 0	ARC HS4xD (CORE)	m = 5	USB
	m = 1	ARC HS4xD (RTT)	m = 6	ETH
	m = 2	AXI Tunnel	m = 7	SDIO
	m = 3	HDMI-V	m = 8	GPU
	m = 4	HDMI-A	m = 9/10	DMAC

^[2] See Table 18 for reset values after U-Boot.

CREG_AXI_M_m_OFFSET0 – Address Offset Register (Address Offset = 0x1008 + m*0x20)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
3:0	OFFSET0	RW	[2] *	Address offset for address aperture[0]
			0	0 * 256 MB
			1	1 * 256 MB
		
			15	15 * 256 MB
7:4	OFFSET1	RW	[2] *	Address offset for address aperture[1]
11:8	OFFSET2	RW	[2] *	Address offset for address aperture[2]
15:12	OFFSET3	RW	[2] *	Address offset for address aperture[3]
19:16	OFFSET4	RW	[2] *	Address offset for address aperture[4]

23:20	OFFSET5	RW	[2] *	Address offset for address aperture[5]
27:24	OFFSET6	RW	[2] *	Address offset for address aperture[6]
31:28	OFFSET7	RW	[2] *	Address offset for address aperture[7]

[1]	m = 0	ARC HS4xD (CORE)	m = 5	USB
	m = 1	ARC HS4xD (RTT)	m = 6	ETH
	m = 2	AXI Tunnel	m = 7	SDIO
	m = 3	HDMI-V	m = 8	GPU
	m = 4	HDMI-A	m = 9/10	DMAC

[2] See Table 18 for reset values after U-Boot.

CREG_AXI_M_m_OFFSET1 – Address Offset Register (Address Offset = 0x100C + m*0x20)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
3:0	OFFSET8	RW	[2] *	Address offset for address aperture[8]
			0	0 * 256 MB
			1	1 * 256 MB
		
			15	15 * 256 MB
7:4	OFFSET9	RW	[2] *	Address offset for address aperture[9]
11:8	OFFSET10	RW	[2] *	Address offset for address aperture[10]
15:12	OFFSET11	RW	[2] *	Address offset for address aperture[11]
19:16	OFFSET12	RW	[2] *	Address offset for address aperture[12]
23:20	OFFSET13	RW	[2] *	Address offset for address aperture[13]
27:24	OFFSET14	RW	[2] *	Address offset for address aperture[14]
31:28	OFFSET15	RW	[2] *	Address offset for address aperture[15]

[1]	m = 0	ARC HS4xD (CORE)	m = 5	USB
	m = 1	ARC HS4xD (RTT)	m = 6	ETH
	m = 2	AXI Tunnel	m = 7	SDIO
	m = 3	HDMI-V	m = 8	GPU
	m = 4	HDMI-A	m = 9/10	DMAC

[2] See Table 18 for reset values after U-Boot

CREG_AXI_M_m_UPDATE – Update register (address offset = 0x1014 + m*0x20)

Legend: * reset value

Bit	Name	Access	Value	Description
0	UPDATE	RW1C		All the address aperture configuration registers (that is, *_SLV and *_OFFSET) are double-buffered. The newly programmed values are only be forwarded to the address decoder after writing a '1' to this bit.

^[1]	m = 0	ARC HS4xD (CORE)	m = 5	USB
	m = 1	ARC HS4xD (RTT)	m = 6	ETH
	m = 2	AXI Tunnel	m = 7	SDIO
	m = 3	HDMI-V	m = 8	GPU
	m = 4	HDMI-A	m = 9/10	DMAC

Table 18 CREG Address Decoder Register Reset Values (After U-Boot)

m	Master	AXI_M_m_ SLV0	AXI_M_m_ SLV1	AXI_M_m_ OFFSET0	AXI_M_m_ OFFSET0
0	HS4xD (CORE)	0x1111_1111	0x6111_1111	0xFEDC_BA98	0x0654_3210
1	HS4xD (RTT)	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
2	AXI Tunnel	0x8888_8888	0x8888_8888	0xFEDC_BA98	0x7654_3210
3	HDMI-V	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
4	HDMI-A	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
5	USB-HOST	0x7777_7777	0x7777_9999	0xFEDC_BA98	0x7654_BA98
6	ETHERNET	0x7777_7777	0x7777_9999	0xFEDC_BA98	0x7654_BA98
7	SDIO	0x7777_7777	0x7777_9999	0xFEDC_BA98	0x7654_BA98
8	GPU	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
9	DMAC (port #1)	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
10	DMAC (port #2)	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210

4.2.1.1.3 Latency Registers

CREG_DDR_LATENCY – DDR Latency Register (Address Offset = 0x1200)

Legend: * reset value

Bit	Name	Access	Value	Description
7:0	LATENCY_0	RW		Latency value for DDR controller (host port 0)
			0 *	0 clocks cycles (latency unit is disabled)
			1	1 clocks cycle
			2	2 clocks cycles
			..	
			255	255 clocks cycles
15:8	LATENCY_1	RW	0 *	Latency value for DDR controller (host port 1)
23:16	LATENCY_2	RW	0 *	Latency value for DDR controller (host port 2)
31:24	LATENCY_3	RW	0 *	Latency value for DDR controller (host port 3)

CREG_SRAM_LATENCY – SRAM Latency Register (Address Offset = 0x1204)

Legend: * reset value

Bit	Name	Access	Value	Description
7:0	LATENCY	RW		Latency value for SRAM controller
			0 *	0 clocks cycles (latency unit is disabled)
			1	1 clocks cycle
			2	2 clocks cycles
			..	
			255	255 clocks cycles

4.2.1.1.4 Base Address Registers

CREG_CSM_BASE – CSM Base address Register (Address Offset = 0x1210)

Legend: * reset value

Bit	Name	Access	Value	Description
17:2	CSM_BASE	RW		CSM base address (256 KB aligned)
			0x0000	CSM base address = 0x0_0000_0000

0x0001	CSM base address = 0x0_0004_0000
0x0002	CSM base address = 0x0_0008_0000
0x0003	CSM base address = 0x0_000C_0000
...	
0x0800	CSM base address = 0x0_2000_0000 *
...	
0xFFFF	CSM base address = 0x3_FFFC_0000

CREG_DCCM_BASE – DCCM Base address Register (Address Offset = 0x1214)

Legend: * reset value

Bit	Name	Access	Value	Description
7:4	DCCM_BASE_2	RW	0x6 *	DCCM base address ARC HS4xD_2 (256 MB aligned)
15:12	DCCM_BASE_4	RW	0x6 *	DCCM base address ARC HS4xD_4 (256 MB aligned)
			0x0	DCCM base address = 0x0_0000_0000
			0x1	DCCM base address = 0x0_1000_0000
			0x2	DCCM base address = 0x0_2000_0000
			0x3	DCCM base address = 0x0_3000_0000
			..	
			0xF	DCCM base address = 0x0_F000_0000

4.2.1.1.5 Timer Halt Address Registers

CREG_TIMERS_HALT – TIMERS Halt Register (Address Offset = 0x1220)

Legend: * reset value

Bit	Name	Access	Value	Description
0	TIMER_1_HALT	RW		Enable / disable halt mode for 1 st timer of TIMERS module

			0 *	timer 1 free running
			1	timer 1 halted when ARC HS4xD_1 is halted
1	TIMER_2_HALT	RW		Enable / disable halt mode for 2 nd timer of TIMERS module
			0 *	timer 2 free running
			1	timer 2 halted when ARC HS4xD_2 is halted
2	TIMER_3_HALT	RW		Enable / disable halt mode for 3 rd timer of TIMERS module
			0 *	timer 3 free running
			1	timer 3 halted when ARC HS4xD_3 is halted
3	TIMER_3_HALT	RW		Enable / disable halt mode for 4 th timer of TIMERS module
			0 *	timer 4 free running
			1	timer 4 halted when ARC HS4xD_4 is halted

CREG_RTC_HALT – RTC Halt Register (Address Offset = 0x1224)

Legend: * reset value

Bit	Name	Access	Value	Description
3:0	RTC_HALT_OR ^[1]	RW		Enable / disable halt mode for RTC timer (OR mask)
			4'b0000 *	RTC timer free running
			4'b0001	RTC timer halted when ARC HS4xD_1 is halted
			4'b0010	RTC timer halted when ARC HS4xD_2 is halted
			4'b0011	RTC timer halted when ARC HS4xD_1, or ARC HS4xD_2 is halted
			
			4'b1111	RTC timer halted when ARC HS4xD_1, ARC HS4xD_2, ARC HS4xD_3, or ARC HS4xD_4 is halted
7:4	RTC_HALT_AND	RW		Enable / disable halt mode for RTC timer (AND mask)

4'b0000 *	RTC timer free running
4'b0001	RTC timer halted when ARC HS4xD_1 is halted
4'b0010	RTC timer halted when ARC HS4xD_2 is halted
4'b0011	RTC timer halted when ARC HS4xD_1, and ARC HS4xD_2 are halted
.....	
4'b1111	RTC timer halted when ARC HS4xD_1, ARC HS4xD_2, ARC HS4xD_3, and ARC HS4xD_4 are halted

[1] RTC_HALT_OR is ignored when RTC_HALT_AND is set (that is, not equal to zero)

4.2.1.1.6 Boot Registers

CREG_BOOT – Boot Register (Address Offset = 0x1010)

Legend: * reset value

Bit	Name	Access	Value	Description
1:0	MIRROR [1]	RW		Boot mirror (location of pre-bootloader)
			0	Disabled
			1 *	Internal rom
			2	Illegal
			3	Illegal
5:4	IMAGE [2]	RW		Image location
			0	Bypass (. arc core enters halt state after pre-boot)
			1 *	SPI flash
			2	Illegal
			3	Illegal
9:6	AUX [3]	RW		Auxiliary boot settings
			4'h12 * [3]	Speed grade=sg125

[1] reset value for MIRROR[1:0] is sampled from “boot_mirror[1:0]” pin during power-on-reset

[2] reset value for IMAGE[1:0] is sampled from “boot_image[1:0]” pin during power-on-reset

[3] reset value for AUX[3:0] is sampled from “boot_aux[3:0]” pin during power-on-reset

CREG_CPU_START – CPU Start Register (Address Offset = 0x1400)*Legend: * reset value*

Bit	Name	Access	Value	Description
0	START_1	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS4xD_1
1	START_2	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS4xD_2
2	START_3	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS4xD_3
3	START_4	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS4xD_4
4	START_MODE	RW		Boot start mode
			0x0 ^[1]	Start ARC core manually (CREG, external start button or debugger). The core that is started is selected by CORE_SEL, or by START_1/2/3/4
			0x1	Start ARC core autonomously after reset. The core that is started is selected by CORE_SEL
8	POL	RW		Polarity of cpu_start pulse
			0x0	active low
			0x1 *	active high
10:9	CORE_SEL	RW		Boot Core Select
			0x0 ^[2]	HS4xD_1
			0x1	HS4xD_2
			0x2	HS4xD_3
			0x3	HS4xD_4
13:12	MULTI_CORE	RW		Multi Core Mode
			0x0 ^[3]	single-core
			0x1	dual-core
			0x2	triple-core
			0x3	quad-core
16	DEBUG_UART_MODE	RW		Debug UART mode

			0x0 ^[4]	normal mode
			0x1	debug mode
24	HALT_1	RW1C	0x0 *	Writing a '1' to this bit generates a cpu_halt pulse for ARC HS4xD_1
25	HALT_2	RW1C	0x0 *	Writing a '1' to this bit generates a cpu_halt pulse for ARC HS4xD_2
26	HALT_3	RW1C	0x0 *	Writing a '1' to this bit generates a cpu_halt pulse for ARC HS4xD_3
27	HALT_4	RW1C	0x0 *	Writing a '1' to this bit generates a cpu_halt pulse for ARC HS4xD_4
28	STATUS_1	R		ARC HS4xD_1 status
			0x0	running
			0x1 *	halted
29	STATUS_2	R	0x1 *	ARC HS4xD_2 status
30	STATUS_3	R	0x1 *	ARC HS4xD_3 status
31	STATUS_4	R	0x1 *	ARC HS4xD_4 status

^[1] reset value for START_MODE is sampled from "boot_start_mode" pin during power-on-reset

^[2] reset value for CORE_SEL is sampled from "boot_core_sel[1:0]" pins during power-on-reset

^[3] reset value for MUTLI_CORE is sampled from "boot_multi_core[1:0]" pins during power-on-reset

^[4] reset value for DEBUG_UART_MODE is sampled from "debug_uart_mode" pin during power-on-reset

4.2.1.1.7 Mux Registers

CREG_GPIO_DBG_MUX – GPIO DEBUG Mux register (Address Offset = 0x1480)

Legend: * reset value

Bit	Name	Access	Value	Description
0	RESERVED			
1	DBG_SEL_1	RW		GPIO debug mux select for GPIO[5:4]
			0x0*	gpio[5:4] is used for normal gpio
			0x1	gpio[4] is used for ARC PLL clock gpio[5] is used for ARC PLL lock
2	DBG_SEL_2	RW		GPIO debug mux select for GPIO[9:8]
			0x0*	gpio[9:8] is used for normal gpio

			0x1	gpio[8] is used for SYS PLL clock gpio[9] is used for SYS PLL lock
3	DBG_SEL_3	RW		GPIO debug mux select for GPIO[13:12]
			0x0*	gpio[13:12] is used for normal gpio
			0x1	gpio[12] is used for DDR PLL clock gpio[13] is used for DDR PLL lock
4	DBG_SEL_4	RW		GPIO debug mux select for GPIO[17:16]
			0x0*	gpio[17:16] is used for normal gpio
			0x1	gpio[16] is used for TUNNEL PLL clock gpio[17] is used for TUNNEL PLL lock

CREG_GPIO_MUX – GPIO Mux Register (Address Offset = 0x1484)

Legend: * reset value

Bit	Name	Access	Value	Description
2:0	GPIO_SEL_0	RW	0x0*	GPIO mux select for gpio[3:0]
5:3	GPIO_SEL_1	RW	0x0*	GPIO mux select for gpio[7:4]
8:6	GPIO_SEL_2	RW	0x0*	GPIO mux select for gpio[11:8]
11:9	GPIO_SEL_3	RW	0x0*	GPIO mux select for gpio[15:12]
14:12	GPIO_SEL_4	RW	0x0*	GPIO mux select for gpio[17:16]
17:15	GPIO_SEL_5	RW	0x0*	GPIO mux select for gpio[19:18]
20:18	GPIO_SEL_6	RW	0x0*	GPIO mux select for gpio[21:20]
23:21	GPIO_SEL_7	RW	0x0*	GPIO mux select for gpio[23:22]

^[1] see Table 19 value for detailed GPIO mux description

CREG_DMA_MUX – DMA Mux Register (Address Offset = 0x1488)

Legend: * reset value

Bit	Name	Access	Value	Description
3:0	DMA_FC_SEL0	RW		DMA mux select for DMA flow control interface[1:0] ^[1]
			0x0 *	SPI0
7:4	DMA_FC_SEL1	RW		DMA mux select for DMA flow control interface[3:2] ^[1]
			0x3 *	I2C0
11:8	DMA_FC_SEL2	RW		DMA mux select for DMA flow control interface[5:4] ^[1]
			0x0	SPI0
			0x1	SPI1
			0x2	SPI2
			0x3	I2C0
			0x4	I2C1
			0x5	I2C2
			0x6 *	UART0
			0x7	UART1
			0x8	UART2
			0x9-0xF	reserved (=> no peripheral is selected)

^[1] the DMA mux does not implement logic that prevents incorrect assignment of DMA flow control interfaces. Hence, it is the users' responsibility to ensure that a single peripheral is not assigned to multiple DMA flow control interfaces. Assigning a peripheral to multiple DMA flow control interfaces results in incorrect behavior.

Table 19 GPIO Mux

GPIO	GPIO_MUX							
	0	1	2	3	4	5	6	7
0	gpio[0]	uart0_cts	spi1_cs[0]	gpio[0]	gpio[0]	pwm_ch[6]	pwm_ch[6]	pwm_ch[1]
1	gpio[1]	uart0_txd	spi1_mosi	gpio[1]	pwm_ch[0]	gpio[1]	pwm_ch[0]	pwm_ch[0]
2	gpio[2]	uart0_rxd	spi1_miso	i2c1_scl	gpio[2]	gpio[2]	gpio[2]	gpio[2]
3	gpio[3]	uart0_rts	spi1_clk	i2c1_sda	gpio[3]	gpio[3]	gpio[3]	gpio[3]
4	gpio[4]	uart1_cts	spi2_cs[0]	gpio[4]	gpio[4]	pwm_ch[4]	pwm_ch[4]	pwm_ch[3]
5	gpio[5]	uart1_txd	spi2_mosi	gpio[5]	pwm_ch[2]	gpio[5]	pwm_ch[2]	pwm_ch[2]
6	gpio[6]	uart1_rxd	spi2_miso	i2c2_scl	gpio[6]	gpio[6]	gpio[6]	gpio[6]
7	gpio[7]	uart1_rts	spi2_clk	i2c2_sda	gpio[7]	gpio[7]	gpio[7]	gpio[7]
8	gpio[8]	uart2_cts	spi1_cs[1]	gpio[8]	gpio[8]	pwm_ch[2]	pwm_ch[2]	pwm_ch[5]
9	gpio[9]	uart2_txd	spi1_mosi	gpio[9]	pwm_ch[4]	gpio[9]	pwm_ch[4]	pwm_ch[4]
10	gpio[10]	uart2_rxd	spi1_miso	i2c1_scl	gpio[10]	gpio[10]	gpio[10]	gpio[10]
11	gpio[11]	uart2_rts	spi1_clk	i2c1_sda	gpio[11]	gpio[11]	gpio[11]	gpio[11]
12	gpio[12]	uart0_cts	spi2_cs[1]	gpio[12]	gpio[12]	pwm_ch[0]	pwm_ch[0]	pwm_ch[7]
13	gpio[13]	uart0_txd	spi2_mosi	gpio[13]	pwm_ch[6]	gpio[13]	pwm_ch[6]	pwm_ch[6]
14	gpio[14]	uart0_rxd	spi2_miso	i2c2_scl	gpio[14]	gpio[14]	gpio[14]	gpio[14]
15	gpio[15]	uart0_rts	spi2_clk	i2c2_sda	gpio[15]	gpio[15]	gpio[15]	gpio[15]
16	gpio[16]	uart1_txd	spi1_cs[2]	i2c1_scl	gpio[16]	pwm_fault_0	gpio[16]	pwm_fault_0
17	gpio[17]	uart1_rxd	spi1_mosi	i2c1_sda	pwm_ch[0]	pwm_ch[0]	pwm_ch[5]	pwm_ch[5]
18	gpio[18]	uart2_txd	spi1_miso	i2c2_scl	gpio[18]	gpio[18]	gpio[18]	gpio[18]
19	gpio[19]	uart2_rxd	spi1_clk	i2c2_sda	gpio[19]	gpio[19]	gpio[19]	gpio[19]
20	gpio[20]	uart0_txd	spi2_cs[2]	i2c1_scl	gpio[20]	pwm_fault_1	gpio[20]	pwm_fault_1
21	gpio[21]	uart0_rxd	spi2_mosi	i2c1_sda	pwm_ch[6]	pwm_ch[6]	pwm_ch[3]	pwm_ch[3]
22	gpio[22]	uart2_txd	spi2_miso	i2c2_scl	gpio[22]	gpio[22]	gpio[22]	gpio[22]
23	gpio[23]	uart2_rxd	spi2_clk	i2c2_sda	gpio[23]	gpio[23]	gpio[23]	gpio[23]

4.2.1.1.8 TUNNEL Control and Status Registers**CREG_TUN_CTRL – Tunnel Control Register (Address Offset = 0x14A0)**

Legend: * reset value

Bit	Name	Access	Value	Description
4	PD	RW		Power down tunnel. This bit can be used to power down the tunnel through software if taxi_pd='0'
			0x0 *	normal mode (only if taxi_pd='0')
			0x1	Power-down mode
1:0	PRIO	RW		Priority setting for arbitration
			0x0 *	AXI master and AXI slave have equal priority (round-robin)
			0x1	AXI master has highest priority
			0x2	AXI slave has highest priority

CREG_TUN_STATUS – Tunnel Status Register (Address Offset = 0x14A4)

Legend: * reset value

Bit	Name	Access	Value	Description
0	INIT_DONE	R		Initialization done
			0x0	Initialization sequence is not yet completed
			0x1 *	Initialization sequence is completed
1	INIT_ERROR	R		Initialization error (only valid when INIT_DONE='1')
			0x0 *	Initialization sequence completed successfully
			0x1	Initialization sequence completed with error
2	BIST_DONE	R		BiST done
			0x0	BiST sequence is not yet completed
			0x1 *	BiST sequence completed
3	BIST_ERROR	R		BiST error (only valid when BIST_DONE='1')
			0x0 *	BiST sequence is completed successfully
			0x1	BiST sequence is completed with error

4.2.1.1.9 DDR Control and Status Registers**CREG_DDR_CTRL – DDR Control Register (Address Offset = 0x14A8)***Legend: * reset value after U-Boot*

Bit	Name	Access	Value	Description
0	MCTL_ENA	RW		Enable for DDR memory controller
			0x0	Disable DDR memory controller
			0x1 *	Enable (release reset) DDR memory controller
1	PUB_ENA	RW		Enable for DDR PHY Utility Block
			0x0	Disable DDR PHY Utility Block
			0x1 *	Enable (release reset) DDR PHY Utility Block

CREG_DDR_STATUS – DDR Status Register (Address Offset = 0x14AC)*Legend: * reset value after U-Boot*

Bit	Name	Access	Value	Description
0	DDR_INIT_DONE	R		DDR initialization done
			0x0	DDR initialization sequence is not yet completed
			0x1 *	DDR initialization sequence is completed

4.2.1.1.10 SPI Control Registers**CREG_SPI_CS_CTRL – SPI Chip Select ctrl Register (Address Offset = 0x14B0)**

Legend: * reset value

Bit	Name	Access	Value	Description
1:0	SPI-0_CS0	RW		Control for SPI [0] 1 st chip select
			0x0 *	CS_n controlled by SPI-0
			0x1	CS_n controlled by SPI-0
			0x2	CS_n controlled by CREG => SPI-0_CS_n[0] = 0
			0x3	CS_n controlled by CREG => SPI-0_CS_n[0] = 1
3:2	SPI-0_CS1	RW		Control for SPI [0] 2 nd chip select
			0x0 *	CS_n controlled by SPI-0
			0x1	CS_n controlled by SPI-0
			0x2	CS_n controlled by CREG => SPI-0_CS_n[1] = 0
			0x3	CS_n controlled by CREG => SPI-0_CS_n[1] = 1
5:4	SPI-0_CS2	RW		Control for SPI [0] 3 rd chip select
			0x0 *	CS_n controlled by SPI-0
			0x1	CS_n controlled by SPI-0
			0x2	CS_n controlled by CREG => SPI-0_CS_n[2] = 0
			0x3	CS_n controlled by CREG => SPI-0_CS_n[2] = 1
7:6	SPI-0_CS3	RW		Control for SPI [0] 4 th chip select
			0x0 *	CS_n controlled by SPI-0
			0x1	CS_n controlled by SPI-0
			0x2	CS_n controlled by CREG => SPI-0_CS_n[3] = 0
			0x3	CS_n controlled by CREG => SPI-0_CS_n[3] = 1
9:8	SPI-1_CS0	RW		Control for SPI [1] 1 st chip select
			0x0 *	CS_n controlled by SPI-1
			0x1	CS_n controlled by SPI-1
			0x2	CS_n controlled by CREG => SPI-1_CS_n[0] = 0
			0x3	CS_n controlled by CREG => SPI-1_CS_n[0] = 1

11:10	SPI-1_CS1	RW		Control for SPI [1] 2 nd chip select
			0x0 *	CS_n controlled by SPI-1
			0x1	CS_n controlled by SPI-1
			0x2	CS_n controlled by CREG => SPI-1_CS_n[1] = 0
			0x3	CS_n controlled by CREG => SPI-1_CS_n[1] = 1
13:12	SPI-1_CS2	RW		Control for SPI [1] 3 rd chip select
			0x0 *	CS_n controlled by SPI-1
			0x1	CS_n controlled by SPI-1
			0x2	CS_n controlled by CREG => SPI-1_CS_n[2] = 0
			0x3	CS_n controlled by CREG => SPI-1_CS_n[2] = 1
17:16	SPI-2_CS0	RW		Control for SPI [2] 1 st chip select
			0x0 *	CS_n controlled by SPI-2
			0x1	CS_n controlled by SPI-2
			0x2	CS_n controlled by CREG => SPI-2_CS_n[0] = 0
			0x3	CS_n controlled by CREG => SPI-2_CS_n[0] = 1
19:18	SPI-2_CS1	RW		Control for SPI [2] 2 nd chip select
			0x0 *	CS_n controlled by SPI-2
			0x1	CS_n controlled by SPI-2
			0x2	CS_n controlled by CREG => SPI-2_CS_n[1] = 0
			0x3	CS_n controlled by CREG => SPI-2_CS_n[1] = 1
21:20	SPI-2_CS2	RW		Control for SPI [2] 3 rd chip select
			0x0 *	CS_n controlled by SPI-2
			0x1	CS_n controlled by SPI-2
			0x2	CS_n controlled by CREG => SPI-2_CS_n[2] = 0
			0x3	CS_n controlled by CREG => SPI-2_CS_n[2] = 1

4.2.1.1.11 Clock Control registers**CREG_GPIO_CLK_CTRL – GPIO Clock Control Register (Address Offset = 0x14B4)***Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	DIV	RW		Clock control register for GPIO de-bounce clock
			0x0 *	clk_in divide-by-1
			0x1	clk_in divide-by-2
			...	
			0xFF	clk_in divide-by-256

CREG_ARC_CLK_CTRL – ARC Clock Control Register (Address Offset = 0x14B8)*Legend: * reset value*

Bit	Name	Access	Value	Description
0	DIV	RW		Clock control register for ARC interface clock
			0x0 *	arc_clk divide-by-1
			0x1	arc_clk divide-by-2

CREG_I2S_CLK_CTRL – I2S Clock Control Register (Address Offset = 0x14BC)*Legend: * reset value*

Bit	Name	Access	Value	Description
0	DIV	RW		Clock control register for I2S sclk
			0x0 *	Separate sclk for I2S TX and RX
			0x1	Combined sclk for I2S TX and RX

4.2.1.1.12 RTT Reset Control Registers**CREG_RTT_RST_CTRL – RTT Reset Control Register (Address Offset = 0x14C0)***Legend: * reset value*

Bit	Name	Access	Value	Description
1:0	RST_CTRL	RW		ARC RTT IC reset (rtt_mictor_p9) control register
			0x0	ARC RTT IC reset is active-low
			0x1	ARC RTT IC reset is active-high

			0x2 *	ARC RTT IC reset is ignored
			0x3	Illegal

4.2.1.1.13 Interrupt Registers

CREG_INT_CLR_ENABLE – Interrupt Enable Clear Register (Address Offset = 0x1500)

Bit	Name	Access	Value	Description
5:0	INT_CLR_ENABLE	W		Writing a '1' to an INT_CLR_ENABLE bit resets the corresponding interrupt enable bit in the INT_ENABLE register to '0'. bit[3:0] => HS4xD core 4 / 3 / 2 / 1

CREG_INT_SET_ENABLE – Interrupt Enable Set Register (Address Offset = 0x1504)

Bit	Name	Access	Value	Description
5:0	INT_SET_ENABLE	W		Writing a 1 to an INT_SET_ENABLE bit sets the corresponding interrupt enable bit in the INT_ENABLE register to 1.

CREG_INT_STATUS – Interrupt Status Register (Address Offset = 0x1508)

Legend: * reset value

Bit	Name	Access	Value	Description
5:0	INT_STATUS	R	0x0*	Each INT_STATUS bit indicates the status of the corresponding interrupt event.

CREG_INT_ENABLE – Interrupt Enable Register (Address Offset = 0x150C)*Legend: * reset value*

Bit	Name	Access	Value	Description
5:0	INT_ENABLE	R	0x0*	Each INT_ENABLE bit indicates whether the corresponding interrupt event is enabled or disabled.

CREG_INT_CLR_STATUS – Interrupt Status Clear Register (Address Offset = 0x1510)

Bit	Name	Access	Value	Description
5:0	INT_CLR_STATUS	W		Writing a 1 to an INT_CLR_STATUS bit resets the corresponding interrupt status bit in the INT_STATUS register to 0.

CREG_INT_SET_STATUS – Interrupt Status Set Register (Address Offset = 0x1514)

Bit	Name	Access	Value	Description
11:0	INT_SET_STATUS	W		Writing a 1 to an INT_SET_STATUS bit sets the corresponding interrupt status bit in the INT_STATUS register to 1.

4.2.1.1.14 Standard Registers**CREG_IP_SW_RESET – CREG Software Reset Register (Address Offset = 0x1FF0)***Legend: * reset value*

Bit	Name	Access	Value	Description
0	RESET	RW1C		Software reset. Writing a 1 to this bit initiates a software reset if the entire IP. After initiating the software reset, software can read the software reset register and when the read completes, software can determine that the IP has been reset.

CREG_IP_VERSION – CREG Version Register (Address Offset = 0x1FF8)*Legend: * reset value*

Bit	Name	Access	Value	Description
				Version of the ARC HS4xD Development System SoC
4:0	DD	R	06	- date : day
8:5	MM	R	02	- date : month

20:9	YYYY	R	2019	- date : year
26:21	TM	R	02	- time : minutes
31:27	TH	R	14	- time : hours

CREG_IP_TYPE – CREG Type Register (Address Offset = 0x1FFC)

Legend: * reset value

Bit	Name	Access	Value	Description
7:0	PRODUCT	R	0x4 *	Product code 0x4 = ARC HS4xD Development Kit
15:8	IP	R	0x2 *	IP code 0x1 = CGU 0x2 = CREG 0x5 = PWM
31:16	DW	R	0x4144 *	Hex code for the two ASCII letters “AD” (Arc Development)

4.2.2 Clock Registers

The clock registers and the associated clock circuitry (that is, PLLs + clock dividers) inside the ARC HS4xD Development System SoC are implemented by the CGU module. The clock registers are used to program the PLLs and clock dividers. The CGU implements the following PLLs:

- **ARC PLL**
Used to generate the clock for the ARC HS4xD quad-core
- **SYS PLL**
Used to generate all the other clocks in the system (for example, AXI, APB, and IP core clocks)
- **DDR PLL**
Used to generate reference clock for the DDR PHY
- **TUN PLL**
Used to generate the clock for the AXI tunnel
- **HDMI PLL**
Used to generate pixel clock for the HDMI PHY

Further, the CGU module also implements measurement logic that can be used to verify that PLL and dividers have been programmed correctly. Guidelines for programming PLLs and measuring clocks are provided in the following sections.

[Table 20](#) lists the registers for the CGU module including a brief description and their offset to the base address of the AXI2APB bridge (default = 0xF000_0000). All registers are 32-bit wide. Read/write access to undefined registers is ignored, and an APB error response is generated. All unused bits within a register are non-writable and return zero when read. A detailed register description can be found in [Register Descriptions on page 92](#).

Table 20 CGU Clock Register Overview

Name	Address Offset	Access [1]	Description
PLL Control and Status Registers			
-ARC PLL			
CGU_ARC_PLL_CTRL	0x0000	RW	ARC PLL control register
CGU_ARC_PLL_STATUS	0x0004	R	ARC PLL status register
CGU_ARC_PLL_FMEAS	0x0008	RW	ARC PLL frequency measurement register
CGU_ARC_PLL_MON	0x000C	RW	ARC PLL monitor register

SYS PLL			
CGU_SYS_PLL_CTRL	0x0010	RW	SYS PLL control register
CGU_SYS_PLL_STATUS	0x0014	R	SYS PLL status register
CGU_SYS_PLL_FMEAS	0x0018	RW	SYS PLL frequency measurement register
CGU_SYS_PLL_MON	0x001C	RW	SYS PLL monitor register
DDR PLL			
CGU_DDR_PLL_CTRL	0x0020	RW	DDR PLL control register
CGU_DDR_PLL_STATUS	0x0024	R	DDR PLL status register
CGU_DDR_PLL_FMEAS	0x0028	RW	DDR PLL frequency measurement register
CGU_DDR_PLL_MON	0x002C	RW	DDR PLL monitor register
TUN PLL			
CGU_TUN_PLL_CTRL	0x0030	RW	Tunnel PLL control register
CGU_TUN_PLL_STATUS	0x0034	R	Tunnel PLL status register
CGU_TUN_PLL_FMEAS	0x0038	RW	Tunnel PLL frequency measurement register
CGU_TUN_PLL_MON	0x003C	RW	Tunnel PLL monitor register
HDMI PLL			
CGU_HDMI_PLL_CTRL	0x0040	RW	HDMI PLL control register
CGU_HDMI_PLL_STATUS	0x0044	R	HDMI PLL status register
CGU_HDMI_PLL_FMEAS	0x0048	RW	HDMI PLL frequency measurement register
CGU_HDMI_PLL_MON	0x004C	RW	HDMI PLL monitor register
Clock Control and Status Registers			
ARC PLL			
CGU_ARC_IDIV	0x0080	RW	Clock divider register for ARC HS4xD clock
CGU_ARC_FMEAS	0x0084	RW	Clock measurement register for ARC HS4xD clock

SYS PLL			
CGU_SYS_IDIV_APB	0x0180	RW	Clock divider register for APB clock
CGU_SYS_FMEAS_APB	0x0184	RW	Clock measurement register for APB clock
CGU_SYS_IDIV_AXI	0x0190	RW	Clock divider register for AXI clock
CGU_SYS_FMEAS_AXI	0x0194	RW	Clock measurement register for AXI clock
CGU_SYS_IDIV_ETH	0x01A0	RW	Clock divider register for ETH clock
CGU_SYS_FMEAS_ETH	0x01A4	RW	Clock measurement register for ETH clock
CGU_SYS_IDIV_USB	0x01B0	RW	Clock divider register for USB clock
CGU_SYS_FMEAS_USB	0x01B4	RW	Clock measurement register for USB clock
CGU_SYS_IDIV_SDIO	0x01C0	RW	Clock divider register for SDIO clock
CGU_SYS_FMEAS_SDIO	0x01C4	RW	Clock measurement register for SDIO clock
CGU_SYS_IDIV_GPU_CORE	0x01E0	RW	Clock divider register for GPU core clock
CGU_SYS_FMEAS_GPU_CORE	0x01E4	RW	Clock measurement register for GPU core clock
CGU_SYS_IDIV_GPU_DMA	0x01F0	RW	Clock divider register for GPU dma clock
CGU_SYS_FMEAS_GPU_DMA	0x01F4	RW	Clock measurement register for GPU dma clock
CGU_SYS_IDIV_GPU_CFG	0x0200	RW	Clock divider register for GPU config clock
CGU_SYS_FMEAS_GPU_CFG	0x0204	RW	Clock measurement register for GPU config clock
CGU_SYS_IDIV_DMAC_CORE	0x0210	RW	Clock divider register for DMAC clock
CGU_SYS_FMEAS_DMAC_CORE	0x0214	RW	Clock measurement register for DMAC clock

CGU_SYS_IDIV_DMAC_CFG	0x0220	RW	Clock divider register for DMAC config clock
CGU_SYS_FMEAS_DMAC_CFG	0x0224	RW	Clock measurement register for DMAC config clock
CGU_SYS_IDIV_SDIO_REF	0x0230	RW	Clock divider register for SDIO reference clock
CGU_SYS_FMEAS_SDIO_REF	0x0234	RW	Clock measurement register for SDIO reference clock
CGU_SYS_IDIV_SPI_REF	0x0240	RW	Clock divider register for SPI reference clock
CGU_SYS_FMEAS_SPI_REF	0x0244	RW	Clock measurement register for SPI reference clock
CGU_SYS_IDIV_I2C_REF	0x0250	RW	Clock divider register for I2C reference clock
CGU_SYS_FMEAS_I2C_REF	0x0254	RW	Clock measurement register for I2C reference clock
CGU_SYS_IDIV_UART_REF	0x0260	RW	Clock divider register for UART reference clock
CGU_SYS_FMEAS_UART_REF	0x0264	RW	Clock measurement register for UART reference clock
TUN PLL			
CGU_TUN_IDIV	0x0380	RW	Clock divider register for HDMI clock
CGU_TUN_FMEAS	0x0384	RW	Clock measurement register for HDMI clock
HDMI PLL			
CGU_HDMI_IDIV	0x0480	RW	Clock divider register for Tunnel clock
CGU_HDMI_FMEAS	0x0484	RW	Clock measurement register for Tunnel clock
I2S			
CGU_I2S_IDIV_TX	0x0580	RW	Clock divider register for I2S TX clock

CGU_I2S_FMEAS_TX	0x0584	RW	Clock measurement register for I2S TX clock
CGU_I2S_IDIV_RX	0x0590	RW	Clock divider register for I2S RX clock
CGU_I2S_FMEAS_RX	0x0594	RW	Clock measurement register for I2S RX clock
Reset Control Registers			
CGU_SYS_RST_CTRL	0x08A0	RW	Reset control register for System peripherals
CGU_DDR_RST_CTRL	0x08C0	RW	Reset control register for DDR
CGU_TUN_RST_CTRL	0x08E0	RW	Reset control register for Tunnel
CGU_HSDK_RST_CTRL	0x0980	RW	Reset control register for HSDK
Standard Registers			
CGU_IP_SW_RESET	0x0FF0	RW1C	CGU IP software reset register
CGU_IP_VERSION	0x0FF8	R	CGU IP version register
CGU_IP_TYPE	0x0FFC	R	CGU IP type register

[1] The following access types are defined:

RW	Read/Write register
R	Read-only register
W	Write-only register
RW1C	Read-only, Write-1-to-Clear Register

4.2.2.1 PLL Programming

The CGU implements logic (FSM + clock switch) that allows glitch less reprogramming of the PLLs with minimal software interaction. Software must program the desired PLL divider settings and the CGU autonomously cycle through the following steps:

- 1) Switch all integer dividers to **sysclk** (33 MHz)
- 2) Apply new PLL divider settings
- 3) Wait for PLL lock
- 4) Switch all fractional dividers back to PLL

To detect completion of the PLL reprogramming sequence, software can either monitor the PLL locked interrupt or poll the `CGU_*_PLL_STATUS` register.

The divider settings for a certain PLL output clock frequency, F_{out} , can be calculated as follows:

$$F_{ref} = F_{in} / NR$$

$$F_{vco} = F_{out} * NO$$

$$F_{out} = F_{in} * NF / (NR * NO)$$

with:

NR = input divider value (1-32)

NF = feedback divider value (16-160)

NO = output divider value (1, 2, 4, or 8)

For proper operation in normal mode, the following constraints *must* be satisfied:

<u>Low-band</u>		<u>High-band</u>	
10 MHz	$\leq \text{Fref} \leq 50 \text{ MHz}$	25 MHz	$\leq \text{Fref} \leq 50 \text{ MHz}$
800 MHz	$\leq \text{Fvco}$	$\leq 1600 \text{ MHz}$	$1500 \text{ MHz} \leq \text{Fvco} \leq 3000$
100 MHz	$\leq \text{Fout}$	$\leq 1600 \text{ MHz}$	$188 \text{ MHz} \leq \text{Fout} \leq 3000$
MHz			

4.2.2.2 Frequency Measurement

The frequency measurement module measures the frequency of a clock relative to the 33 MHz input reference clock. The frequency measurement modules can be controlled through the `CGU_*_FMEAS` registers.

When the START bit in the `CGU_*_FMEAS` register is set to 1, a 15-bit counter, FCNT, starts counting the number of cycles of the clock to be measured and simultaneously a second 15-bit counter, RCNT, starts counting the number of cycles of the reference clock.

When either counter reaches its maximum count, both counters are disabled, and the DONE bit in the `CGU_*_FMEAS` register is set to 1.

The current values of the counters can then be read out and the measured frequency can be obtained by the following equation:

$$f_{\text{meas}} = (\text{FCNT} / \text{RCNT}) * f_{\text{ref}}$$

Few remarks:

- By default, both counters start counting from zero. However, to simplify the frequency calculation, RCNT can be initialized with a non-zero value. When RCNT is initialized with a value equal to 2^{15} – reference clock frequency in MHz and reaches its maximum count before the FCNT counter saturates, the value stored in FCNT then shows the measured clock's frequency in MHz without the need for any further calculation.
- The measured clock frequency can only be as known to the level of precision of the reference clock frequency.
- Quantization error is noticeable if the ratio between the two clocks is large (for example 1000 MHz versus 1 kHz), because one counter saturates while the other counter only has a small count value.
- Due to synchronization, both counters are not started and stopped at the same time. This affects the accuracy of the frequency measurement. This effect can be minimized by running the counters if possible.

4.2.2.3 Register Descriptions

4.2.2.3.1 PLL Control and Status Registers

CGU_p_PLL_CTRL – PLL Control Register (Address Offset = 0x0000 + p*0x10)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
0	PD	RW	0*	Power down PLL (0=normal mode, 1=power down mode)
1	BYPASS	RW	0*	Bypass PLL (0=normal mode, 1=bypass mode)
3:2	OD	RW	see below*	Output divider value $NO = 2^{**}OD$
8:4	R	RW	see below*	Input divider value $NR = R+1$
15:9	F	RW	see below*	Feedback divider value $NF = 2^{*}(F+1)$
				$F_{out} = F_{in} * NF / (NR * NO)$ $F_{out} = F_{in} * 24 / (1 * 8) = 33.33 * 24 / 8 = 100 \text{ MHz}$
20	BS	RW	see below*	Band selection (0=low-band, 1=high-band)

p = 0	ARC PLL	OD=1	R=0	F=14	BS=0	freq=500 MHz
p = 1	SYS PLL	OD=1	R=0	F=23	BS=0	freq=800 MHz
p = 2	DDR PLL	OD=2	R=1	F=39	BS=0	freq=333 MHz
p = 3	TUN PLL	OD=1	R=0	F=17	BS=0	freq=600 MHz
p = 4	HDMI PLL	OD=2	R=0	F=21	BS=0	freq=297 MHz

CGU_p_PLL_STATUS – PLL Status Register (Address Offset = 0x0004 + p*0x10)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
0	LOCK	R	1*	PLL lock indication
1	ERROR	R	0*	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.

p = 0	ARC PLL
p = 1	SYS PLL
p = 2	DDR PLL
p = 3	TUN PLL
p = 4	HDMI PLL

CGU_p_PLL_FMEAS – PLL Measurement Register (Address Offset = 0x0008 + p*0x10)*Legend: * reset value*

Bit	Name	Access	Value	Description
14:0	RCNT	RW	0*	Value of the reference counter.
29:15	FCNT	R	0*	Value of the frequency counter
30	DONE	R	0*	Asserted high to indicate that the frequency measurement has completed.
31	START	RW1C	0*	Writing a 1 to the START bit starts a frequency measurement. - measured frequency = (FCNT / RCNT) * f _{ref}

p = 0	ARC PLL
p = 1	SYS PLL
p = 2	DDR PLL
p = 3	TUN PLL
p = 4	HDMI PLL

CGU_p_PLL_MON – PLL Monitor Register (Address Offset = 0x000C + p*0x10)*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	N	RW		Integer divide-by-N value for PLL clock monitor output
			0*	disabled
			1	divide-by-1
			2	divide-by-2
		
			255	divide-by-255

p = 0	ARC PLL
p = 1	SYS PLL
p = 2	DDR PLL
p = 3	TUN PLL
p = 4	HDMI PLL

4.2.2.3.2 Clock Control and Status Registers

CGU_p_IDIV_c – Clock Divider Register (Address Offset = 0x0080 + p*0x100 + c*0x10)

Legend: * reset value after U-Boot

Bit	Name	Access	Value	Description
7:0	N	RW	see below*	Integer divide-by-N value
			0	disabled
			1	divide-by-1
			2	divide-by-2
		
			255	divide-by-255

ARC PLL	p = 0	c = 0	ARC HS4xD clock	N = 1	freq=500/1=500 MHz
SYS PLL	p = 1	c = 0	APB clock	N = 4	freq=800/4=200 MHz
		c = 1	AXI clock	N = 2	freq=800/2=400 MHz
		c = 2	ETH core clock	N = 2	freq=800/2=400 MHz
		c = 3	USB core clock	N = 2	freq=800/2=400 MHz
		c = 4	SDIO core clock	N = 2	freq=800/2=400 MHz
		c = 5	HDMI core clock	N = 2	freq=800/2=400 MHz
		c = 6	GPU core clock	N = 2	freq=800/2=400 MHz
		c = 9	DMAC core clock	N = 2	freq=800/2=400 MHz
		c = 10	DMAC cfg core clock	N = 4	freq=800/4=200 MHz
		c = 11	SDIO reference clock	N = 8	freq=800/8=100 MHz
		c = 12	SPI reference clock	N = 24	freq=800/24=33 MHz
		c = 13	I2C reference clock	N = 4	freq=800/4=200 MHz
		c = 14	UART reference clock	N = 24	freq=800/24=33 MHz
TUN PLL	p = 3	c = 0	Tunnel clock	N = 12	freq=600/12=50 MHz
		c = 1	ROM clock	N = 4	freq=600/4=150 MHz
		c = 2	PWM clock	N = 8	freq=600/8=75 MHz
		c = 3	TIMERS / RTC clock	N = 12	freq=600/12=50 MHz
HDMI PLL	p = 4	c = 0	HDMI pixel clock	N = 1	freq=297/1=297 MHz
I2S	p = 5	c = 0	I2S TX clock	N = 1	freq=297/1=297 MHz
		c = 1	I2S RX clock	N = 1	freq=297/1=297 MHz

CGU_p_FMEAS_c – Clock Measurement Register (Address Offset = 0x0084 + p*0x100 + c*0x10)

Legend: * reset value

Bit	Name	Access	Value	Description
14:0	RCNT	RW	0*	Value of the reference counter.
29:15	FCNT	R	0*	Value of the frequency counter
30	DONE	R	0*	Asserted high to indicate that the frequency measurement has completed.

31	START	W	0*	<p>Writing a 1 to the START bit starts a frequency measurement. The START bit resets to 0 when the frequency measurement has completed</p> <p>- measured frequency = $(FCNT / RCNT) * f_{ref}$</p>
----	-------	---	----	---

ARC PLL	p = 0	c = 0	ARC HS4xD clock
SYS PLL	p = 1	c = 0	APB clock
		c = 1	AXI clock
		c = 2	ETH core clock
		c = 3	USB core clock
		c = 4	SDIO core clock
		c = 5	HDMI core clock
		c = 6	GPU core clock
		c = 9	DMAC core clock
		c = 10	DMAC cfg core clock
		c = 11	SDIO reference clock
		c = 12	SPI reference clock
		c = 13	I2C reference clock
		c = 14	UART reference clock
TUN PLL	p = 3	c = 0	Tunnel clock
		c = 1	ROM clock
		c = 2	PWM clock
		c = 3	TIMERS / RTC clock
HDMI PLL	p = 4	c = 0	HDMI pixel clock
I2S	p = 5	c = 0	I2S TX clock
		c = 1	I2S RX clock

4.2.2.3.3 Reset Control Registers**CGU_SYS_RST_CTRL – SYS Reset Control Register (Address Offset = 0x08A0)**

Legend: * reset value

Bit	Name	Access	Value	Description
0	-	RW	0*	Reserved; must be written as 0
16	-	RW	0*	Reserved; must be written as 0
17	-	RW	0*	Reserved; must be written as 0
18	ETH_RST	RW		Include ETH in software reset
			0*	no
			1	yes
19	USB_RST	RW	0*	Include USB in software reset
			0*	no
			1	yes
20	SDIO_RST	RW	0*	Include SDIO in software reset
			0*	no
			1	yes
21	HDMI_RST	RW	0 *	Include HDMI in SW reset
22	GPU_RST	RW	0*	Include GPU in software reset
			0*	no
			1	yes
23	-	RW	0*	Reserved; must be written as 0
24	-	RW	0*	Reserved; must be written as 0
25	DMAC_RST	RW	0*	Include DMAC in software reset
			0*	no
			1	yes
26	-	RW	0*	Reserved; must be written as 0
27	-	RW	0*	Reserved; must be written as 0
28	-	RW	0*	Reserved; must be written as 0
29	-	RW	0*	Reserved; must be written as 0

30	-	RW	0*	Reserved; must be written as 0
31	-	RW	0*	Reserved; must be written as 0

CGU_DDR_RST_CTRL – ARC Reset Control Register (Address Offset = 0x08C0)

Legend: * reset value

Bit	Name	Access	Value	Description
0	-	RW	0*	Reserved; must be written as 0
16	DDR_RST	RW		Include DDR in software reset
			0*	no
			1	yes

CGU_TUN_RST_CTRL – ARC Reset Control Register (Address Offset = 0x08E0)

Legend: * reset value

Bit	Name	Access	Value	Description
0	-	RW	0*	Reserved; must be written as 0
16	TUN_RST	RW		Include AXI tunnel in software reset
			0*	no
			1	yes
17	-	RW	0*	Reserved; must be written as 0
18	-	RW	0*	Reserved; must be written as 0

CGU_HSDK_RST_CTRL – HSDK Reset Control Register (Address Offset = 0x0980)*Legend: * reset value*

Bit	Name	Access	Value	Description
0	RESETN_OUT	RW	0 *	Include resetn_out in SW reset
1	RESETN_IN	RW	0 *	Include resetn_in in SW reset (that is, reset entire HSDK-4xD board)

4.2.2.3.4 Standard Registers**CGU_IP_SW_RESET – CGU Software Reset Register (Address Offset = 0x0FF0)***Legend: * reset value*

Bit	Name	Access	Value	Description
0	SW_RESET	RW1C		Writing a 1 to this bit initiates a software reset of the ARC HS4xD Development System SoC. After initiating the software reset, software can read the software reset register and when SW_RESET bit is reset to 0, the software reset is completed. The clock domains that are included in the software reset sequence can be selected through the CGU_*_RST_CTRL registers.
31:16	SW_RESET_DELAY	RW	0x0000 *	Delay between software reset command and reset assertion

CGU_IP_VERSION – CGU Version Register (Address Offset = 0x0FF8)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	MINOR_REV	R	0x0000 *	Minor version of the CGU IP
31:16	MAJOR_REV	R	0x0001 *	Major version of the CGU IP

CGU_IP_TYPE – CGU Type Register (Address Offset = 0x0FFC)*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	PRODUCT	R	0x4 *	Product code 0x4 = ARC HS4xD Development Kit
15:8	IP	R	0x1 *	IP code 0x1 = CGU 0x2 = CREG 0x5 = PWM
31:16	DW	R	0x4144 *	Hex code for the two ASCII letters “AD” (Arc Development)

4.2.3 PWM Registers

[Table 21](#) lists the registers for the PWM module including a brief description and their offset to the base address of the AXI2APB bridge (default = 0xF000_0000). All the registers are 32-bit wide. Read/write access to undefined registers is ignored, and an APB error response is generated. All unused bits within a register are non-writable and return zero when read. A detailed register description can be found in section [PWM Register Descriptions](#).

Table 21 PWM Control Register overview

Name	Address Offset	Access ^[1]	Description
Control / Status Registers			
PWM_CTRL	0x2B000	RW	Timer control register
PWM_CHN_CONFIG	0x2B004	RW	Channel configuration register
PWM_TRIGGER	0x2B008	RW	Trigger control register
PWM_FAULT	0x2B00C	RW	Fault control register
PWM_EVENTS	0x2B010	RW1C	Event trigger register
Interrupt Registers			
PWM_INTCTRL	0x2B014	RW	Interrupt control register
PWM_INTSTAT	0x2B018	R	Interrupt status register

PWM_INTCLR	0x2B01C	RW	Interrupt clear register
PWM Channel Registers			
PWM_THRESHOLD_01	0x2B020	RW	PWM threshold register for complementary channel 0 and 1
PWM_THRESHOLD_23	0x2B024	RW	PWM threshold register for complementary channel 2 and 3
PWM_THRESHOLD_45	0x2B028	RW	PWM threshold register for complementary channel 4 and 5
PWM_THRESHOLD_67	0x2B02C	RW	PWM threshold register for complementary channel 6 and 7
PWM_DEADZONE_01	0x2B030	RW	PWM deadzone register for complementary channel 0 and 1
PWM_DEADZONE_23	0x2B034	RW	PWM deadzone register for complementary channel 2 and 3
PWM_DEADZONE_45	0x2B038	RW	PWM deadzone register for complementary channel 4 and 5
PWM_DEADZONE_67	0x2B03C	RW	PWM deadzone register for complementary channel 6 and 7
PWM Timer Registers			
PWM_TIMER_MAX_01	0x2B040	RW	Maximum timer value register for PWM timer 01
PWM_TIMER_MAX_23	0x2B044	RW	Maximum timer value register for PWM timer 23
PWM_TIMER_MAX_45	0x2B048	RW	Maximum timer value register for PWM timer 45
PWM_TIMER_MAX_67	0x2B04C	RW	Maximum timer value register for PWM timer 67
PWM_NPERIODS_01	0x2B050	RW	NPeriods register for PWM timer 01
PWM_NPERIODS_23	0x2B054	RW	NPeriods register for PWM timer 23
PWM_NPERIODS_45	0x2B058	RW	NPeriods register for PWM timer 45
PWM_NPERIODS_67	0x2B05C	RW	NPeriods register for PWM timer 67
PWM_CLK_DIV_01	0x2B060	RW	Clock divider register for PWM timer 01
PWM_CLK_DIV_23	0x2B064	RW	Clock divider register for PWM timer 23
PWM_CLK_DIV_45	0x2B068	RW	Clock divider register for PWM timer 45
PWM_CLK_DIV_67	0x2B06C	RW	Clock divider register for PWM timer 67

Standard Registers

PWM_IP_TYPE	0x2B0FC	R	PWM IP Type register
-------------	---------	---	----------------------

[1] The following access types are defined:

RW	Read/Write register
R	Read-only register
W	Write-only register
RW1C	Read-only, Write-1-to-Clear Register

4.2.3.1 PWM Register Descriptions

4.2.3.1.1 Control and Status Registers

PWM_CTRL – PWM Control Register (Address Offset = 0x2B000)

Legend: * reset value

Bit	Name	Access	Value	Description
1	TIMER_ENABLE_01	RW	0 *	Enable / disable PWM timer 01
			0 *	disabled
			1	enabled
3:2	TIMER_MODE_01	RW		Operation mode for PWM timer 01
			0 *	up-counting
			1	down-counting
			2	up/down-counting
	3	down/up-counting		
5	TIMER_ENABLE_23	RW	0 *	Enable / disable PWM timer 23
7:6	TIMER_MODE_23	RW	0 *	Operation mode for PWM timer 23
9	TIMER_ENABLE_45	RW	0 *	Enable / disable PWM timer 45
11:10	TIMER_MODE_45	RW	0 *	Operation mode for PWM timer 45
13	TIMER_ENABLE_67	RW	0 *	Enable / disable PWM timer 67

15:14	TIMER_MODE_67	RW	0 *	Operation mode for PWM timer 67
-------	---------------	----	-----	---------------------------------

PWM_CHN_CONFIG – PWM Channel Config Register (Address Offset = 0x2B004)

Legend: * reset value

Bit	Name	Access	Value	Description
8	MASK_CHN_0	RW		Mask PWM channel 0
			0 *	normal PWM value
			1	masked (inactive) PWM value
9	MASK_CHN_1	RW	0 *	Mask PWM channel 1
10	MASK_CHN_2	RW	0 *	Mask PWM channel 2
11	MASK_CHN_3	RW	0 *	Mask PWM channel 3
12	MASK_CHN_4	RW	0 *	Mask PWM channel 4
13	MASK_CHN_5	RW	0 *	Mask PWM channel 5
14	MASK_CHN_6	RW	0 *	Mask PWM channel 6
15	MASK_CHN_7	RW	0 *	Mask PWM channel 7
16	POL_CHN_0	RW		Polarity of PWM channel 0
			0 *	normal (active high)
			1	inverted (active low)
			2	up/down-counting
			3	down/up-counting
17	POL_CHN_1	RW	0 *	Polarity of PWM channel 1
18	POL_CHN_2	RW	0 *	Polarity of PWM channel 2
19	POL_CHN_3	RW	0 *	Polarity of PWM channel 3
20	POL_CHN_4	RW	0 *	Polarity of PWM channel 4
21	POL_CHN_5	RW	0 *	Polarity of PWM channel 5
22	POL_CHN_6	RW	0 *	Polarity of PWM channel 6
23	POL_CHN_7	RW	0 *	Polarity of PWM channel 7

PWM_TRIGGER – PWM Trigger Control Register (Address Offset = 0x2B008)

Legend: * reset value

Bit	Name	Access	Value	Description
5:4	DIR_TRG_01	RW		Select the PWM timer 01 count direction on which the trigger interrupt can be triggered
			0 *	count-up
			1	count-down
			2	count-up and count down
7:6	DIR_TRG_23	RW		Select the PWM timer 23 count direction on which trigger interrupt can be triggered
			3	count-up and count down
9:8	DIR_TRG_45	RW		Select the PWM timer 45 count direction on which trigger interrupt can be triggered
11:10	DIR_TRG_67	RW		Select the PWM timer 67 count direction on which trigger interrupt can be triggered

PWM_FAULT – PWM Fault Control Register (Address Offset = 0x2B00C)

Legend: * reset value

Bit	Name	Access	Value	Description	
1:0	FAULT_ENA_01	RW		pwm_fault inputs for complementary channel pair 01	
				pwm_fault[1]	pwm_fault[0]
			b00 *	ignore	ignore
			b01	ignore	react
			b10	react	ignore
			b11	react	react
4:3	FAULT_ENA_23	RW	b00 *	pwm_fault inputs for complementary channel pair 23	
				pwm_fault[3]	pwm_fault[2]
			b00 *	ignore	ignore
			b01	ignore	react
			b10	react	ignore
			b11	react	react

7:6	FAULT_ENA_45	RW	b00 *	pwm_fault inputs for complementary channel pair 45.	
				pwm_fault[5]	pwm_fault[4]
			b00 *	ignore	ignore
			b01	ignore	react
			b10	react	ignore
			b11	react	react
10:9	FAULT_ENA_67	RW	b00 *	pwm_fault inputs for complementary channel pair 67.	
				pwm_fault[5]	pwm_fault[4]
			b00 *	ignore	ignore
			b01	ignore	react
			b10	react	ignore
			b11	react	react
13:12	FAULT_MODE_01	RW		Behavior of complementary channel pair 01 on fault events	
			0 *	PWM channels ignore fault events	
			1	PWM channels are forced to inactive value	
			2	PWM channels are forced to logic 0	
			3	PWM channels are forced to logic 1	
15:14	FAULT_MODE_23	RW		Behavior of complementary channel pair 23 on fault events; the values are similar to bits[13:12].	
17:16	FAULT_MODE_45	RW		Behavior of complementary channel pair 45 on fault events; the values are similar to bits[13:12].	
19:18	FAULT_MODE_67	RW		Behavior of complementary channel pair 67 on fault events; the values are similar to bits[13:12].	
20	FAULT_POL_0	RW		Polarity of pwm_fault_0	
			0 *	normal (active high)	
			1	inverted (active low)	
21	FAULT_POL_1	RW	0 *	Polarity of pwm_fault_1; ; the values are similar to bit[20].	

23	RECOVER	RW		Fault recover mode
			0 *	manual (see RECOVER field in PWM_EVENTS register)
			1	auto (PWM channel output mode restored in the next PWM period if and only if all fault inputs on which the complementary channel pair reacts are inactive again.

PWM_EVENTS – PWM Event Trigger Register (Address offset = 0x2B010)

Legend: * reset value

Bit	Name	Access	Value	Description
0	SW_RST	RW1C	0 *	Software reset. When 1 is written to this field, all PWM registers are reset.
1	PWM_PERIOD_RST	RW1C	0 *	Reset all the PWM timers to their initial value and start a new PWM period.
2	FORCE_UPDATE	RW1C	0 *	Force update all shadow register in the next PWM period. If PWM_PERIOD_RST field is set simultaneously, the Shadow registers are updated, the PWM timer is initialized, and a new PWM period starts.
3	RECOVER	RW1C	0 *	Recover from the activated fault behavior. Complementary channel pairs output the normal PWM output values instead of the fault output (see FAULT_MODE fields in the PWM_FAULT register) in the <i>next</i> PWM period if and only if all fault inputs on which the complementary channel pair reacts are inactive again

4.2.3.1.1 Interrupt Registers

PWM_INTCTRL – PWM Interrupt Control Register (Address Offset = 0x2B014)

Legend: * reset value

Bit	Name	Access	Value	Description
0	ENA_IRQ_NEW_NPERIODS_01	RW	0 *	Enable the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 01
1	ENA_IRQ_NEW_NPERIODS_23	RW	0 *	Enable the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 23

2	ENA_IRQ_NEW_NPERIODS_45	RW	0 *	Enable the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 45
3	ENA_IRQ_NEW_NPERIODS_67	RW	0 *	Enable the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 67
4	ENA_IRQ_UPD_MISSED_01	RW	0 *	Enable the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 01
5	ENA_IRQ_UPD_MISSED_23	RW	0 *	Enable the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 23
6	ENA_IRQ_UPD_MISSED_45	RW	0 *	Enable the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 45
7	ENA_IRQ_UPD_MISSED_67	RW	0 *	Enable the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 67
8	ENA_IRQ_TRIG_01	RW	0 *	Enable the <code>pwm_irq_trg</code> interrupt for PWM generator 01
9	ENA_IRQ_TRIG_23	RW	0 *	Enable the <code>pwm_irq_trg</code> interrupt for PWM generator 23
10	ENA_IRQ_TRIG_45	RW	0 *	Enable the <code>pwm_irq_trg</code> interrupt for PWM generator 45
11	ENA_IRQ_TRIG_67	RW	0 *	Enable the <code>pwm_irq_trg</code> interrupt for PWM generator 67
12	ENA_IRQ_FAULT_0	RW	0 *	Enable the <code>pwm_irq_fault</code> interrupt on an active fault on <code>pwm_fault_0_a</code> .
13	ENA_IRQ_FAULT_1	RW	0 *	Enable the <code>pwm_irq_fault</code> interrupt on an active fault on <code>pwm_fault_1_a</code> .

PWM_INTSTAT – PWM Interrupt Status Register (Address Offset = 0x2B018)

Legend: * reset value

Bit	Name	Access	Value	Description
0	IRQ_NEW_NPERIODS_01	R	0 *	Interrupt on the start of a new sequence of (NPERIODS) PWM periods for PWM timer 01. This interrupt can be used to indicate to the core that new threshold and <code>timer_max</code> register settings can be programmed for the next sequence of PWM periods.
1	IRQ_NEW_NPERIODS_23	R	0 *	Interrupt on the start of a new sequence of (NPERIODS) PWM periods for PWM timer 23. This interrupt can be used to indicate to

				the core that new threshold and timer_max register settings can be programmed for the next sequence of PWM periods.
2	IRQ_NEW_NPERIODS_45	R	0 *	Interrupt on the start of a new sequence of (NPERIODS) PWM periods for PWM timer 45. This interrupt can be used to indicate to the core that new threshold and timer_max register settings can be programmed for the next sequence of PWM periods.
3	IRQ_NEW_NPERIODS_67	R	0 *	Interrupt on the start of a new sequence of (NPERIODS) PWM periods for PWM timer 67. This interrupt can be used to indicate to the core that new threshold and timer_max register settings can be programmed for the next sequence of PWM periods.
4	IRQ_UPD_MISSED_01	R	0 *	Error event: In the previous sequence of (SHD_NPERIODS) PWM timer periods, the THRESHOLD_01 register has not been written.
5	IRQ_UPD_MISSED_23	R	0 *	Error event: In the previous sequence of (SHD_NPERIODS) PWM timer periods, the THRESHOLD_23 register has not been written.
6	IRQ_UPD_MISSED_45	R	0 *	Error event: In the previous sequence of (SHD_NPERIODS) PWM timer periods, the THRESHOLD_45 register has not been written.
7	IRQ_UPD_MISSED_67	R	0 *	Error event: In the previous sequence of (SHD_NPERIODS) PWM timer periods, the THRESHOLD_67 register has not been written.
8	IRQ_TRIG_01	R	0 *	Interrupt when the PWM timer 01 value matches the SHD_THRESHOLD_01 register, and the PWM timer is running in a direction enabled in the DIR_TRG_01 field in the TRIGGER register.
9	IRQ_TRIG_23	R	0 *	Interrupt when the PWM timer 23 value matches the SHD_THRESHOLD_23 register, and the PWM timer is running in a direction enabled in the DIR_TRG_23 field in the TRIGGER register.
10	IRQ_TRIG_45	R	0 *	Interrupt when the PWM timer 45 value matches the SHD_THRESHOLD_45

				register and the PWM timer is running in a direction enabled in the <code>DIR_TRG_45</code> field in the <code>TRIGGER</code> register.
11	<code>IRQ_TRIG_67</code>	R	0 *	Interrupt on a match between the PWM timer 67 value and the <code>SHD_THRESHOLD_67</code> register and the PWM timer is running in a direction enabled in the <code>DIR_TRG_67</code> field in the <code>TRIGGER</code> register.
12	<code>IRQ_FAULT_0</code>	R	0 *	Interrupt on an active fault on <code>pwm_fault_0_a</code> .
13	<code>IRQ_FAULT_1</code>	R	0 *	Interrupt on an active fault on <code>pwm_fault_1_a</code> .

PWM_INTCLR – PWM Interrupt Clear Register (Address Offset = 0x2B01C)

Legend: * reset value

Bit	Name	Access	Value	Description
0	<code>CLR_IRQ_NEW_NPERIODS_01</code>	RW1C	0 *	Clear the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 01
1	<code>CLR_IRQ_NEW_NPERIODS_23</code>	RW1C	0 *	Clear the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 23
2	<code>CLR_IRQ_NEW_NPERIODS_45</code>	RW1C	0 *	Clear the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 45
3	<code>CLR_IRQ_NEW_NPERIODS_67</code>	RW1C	0 *	Clear the <code>pwm_irq_new_nperiods</code> interrupt from PWM timer 67
4	<code>CLR_IRQ_UPD_MISSED_01</code>	RW1C	0 *	Clear the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 01
5	<code>CLR_IRQ_UPD_MISSED_23</code>	RW1C	0 *	Clear the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 23
6	<code>CLR_IRQ_UPD_MISSED_45</code>	RW1C	0 *	Clear the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 45
7	<code>CLR_IRQ_UPD_MISSED_67</code>	RW1C	0 *	Clear the <code>pwm_irq_upd_missed</code> interrupt from PWM generator 67
8	<code>CLR_IRQ_TRIG_01</code>	RW1C	0 *	Clear the <code>pwm_irq_trg</code> interrupt for PWM generator 01
9	<code>CLR_IRQ_TRIG_23</code>	RW1C	0 *	Clear the <code>pwm_irq_trg</code> interrupt for PWM generator 23

10	CLR_IRQ_TRIG_45	RW1C	0 *	Clear the <code>pwm_irq_trg</code> interrupt for PWM generator 45
11	CLR_IRQ_TRIG_67	RW1C	0 *	Clear the <code>pwm_irq_trg</code> interrupt for PWM generator 67
12	CLR_IRQ_FAULT_0	RW1C	0 *	Clear the <code>pwm_irq_fault</code> interrupt on an active fault on <code>pwm_fault_0_a</code> .
13	CLR_IRQ_FAULT_1	RW1C	0 *	Clear the <code>pwm_irq_fault</code> interrupt on an active fault on <code>pwm_fault_1_a</code> .

4.2.3.1.2 PWM Channel Registers

PWM_THRESHOLD_01 – PWM Threshold Register (Address Offset = 0x2B020)

Legend: * reset value

Bit	Name	Access	Value	Description
15:0	THRESHOLD_01	RW	0 *	Threshold value for complementary channels 0 and 1. The threshold value is used by the PWM Generator to drive the proper PWM output value for complementary channels 0 and 1, and used to raise <code>pwm_trq_irq_01</code> interrupts

PWM_THRESHOLD_23 – PWM Threshold Register (Address Offset = 0x2B024)

Legend: * reset value

Bit	Name	Access	Value	Description
15:0	THRESHOLD_23	RW	0 *	Threshold value for complementary channels 2 and 3. The threshold value is used by the PWM Generator to drive the proper PWM output value for complementary channels 2 and 3, and used to raise <code>pwm_trq_irq_23</code> interrupts

PWM_THRESHOLD_45 – PWM Threshold Register (Address Offset = 0x2B028)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	THRESHOLD_45	RW	0 *	Threshold value for complementary channels 4 and 5. The threshold value is used by the PWM Generator to drive the proper PWM output value for complementary channels 4 and 5, and used to raise <code>pwm_trq_irq_45</code> interrupts

PWM_THRESHOLD_67 – PWM Threshold Register (Address Offset = 0x2B02C)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	THRESHOLD_67	RW	0 *	Threshold value for complementary channels 6 and 7. The threshold value is used by the PWM Generator to drive the proper PWM output value for complementary channels 6 and 7, and used to raise <code>pwm_trq_irq_67</code> interrupts

PWM_DEADZONE_01 – PWM Deadzone Register (Address Offset = 0x2B030)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	DEADZONE_01	RW	0 *	The number of additional <code>pwm_clk</code> cycles <code>pwm_ch[0]</code> (<code>pwm_ch[1]</code>) must be 'inactive' when according to the PWM Timer 01 and corresponding <code>THRESHOLD_01</code> value <code>pwm_ch[0]</code> (<code>pwm_ch[1]</code>) must have switched from 'inactive' to 'active'.

PWM_DEADZONE_23 – PWM Deadzone Register (Address Offset = 0x2B034)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	DEADZONE_23	RW	0 *	The number of additional pwm_clk cycles pwm_ch[2] (pwm_ch[3]) must be 'inactive' when according to the PWM Timer 23 and corresponding THRESHOLD_23 value pwm_ch[2] (pwm_ch[3]) must have switched from 'inactive' to 'active'.

PWM_DEADZONE_45 – PWM Deadzone Register (Address Offset = 0x2B038)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	DEADZONE_45	RW	0 *	The number of additional pwm_clk cycles pwm_ch[4] (pwm_ch[5]) must be 'inactive' when according to the PWM Timer 45 and corresponding THRESHOLD_45 value pwm_ch[4] (pwm_ch[5]) must have itched from 'inactive' to 'active'.

PWM_DEADZONE_67 – PWM Deadzone Register (Address Offset = 0x2B03C)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	DEADZONE_67	RW	0 *	The number of additional pwm_clk cycles pwm_ch[6] (pwm_ch[7]) must be 'inactive' when according to the PWM Timer 67 and corresponding THRESHOLD_67 value pwm_ch[6] (pwm_ch[7]) must have switched from 'inactive' to 'active'.

4.2.3.1.3 PWM Timer Registers**PWM_TIMER_MAX_01 – PWM Max Timer Value Register (Address Offset = 0x2B040)***Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	TIMER_MAX_01	RW	0 *	Maximum timer value for PWM timer 01.

PWM_TIMER_MAX_23 – PWM Max Timer Value Register (Address Offset = 0x2B044)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	TIMER_MAX_23	RW	0 *	Maximum timer value for PWM timer 23.

PWM_TIMER_MAX_45 – PWM Max Timer Value Register (Address Offset = 0x2B048)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	TIMER_MAX_45	RW	0 *	Maximum timer value for PWM timer 45.

PWM_TIMER_MAX_67 – PWM Max Timer Value Register (Address Offset = 0x2B04C)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	TIMER_MAX_67	RW	0 *	Maximum timer value for PWM timer 67.

PWM_NPERIODS_01 – PWM nperiods Value Register (Address Offset = 0x2B050)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	NPERIODS_01	RW		The number of PWM periods THRESHOLD_01 and TIMER_MAX register settings are valid for PWM timer 01. <code>pwm_irq_new_nperiods</code> and <code>pwm_irq_upd_missed</code> interrupts do not occur within a sequence of NPERIODS PWM periods.
			0 *	1 PWM period
			1	2 PWM periods
			2	3 PWM periods
			

PWM_NPERIODS_23 – PWM nperiods Value Register (Address Offset = 0x2B054)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	NPERIODS_23	RW	0 *	The number of PWM periods THRESHOLD_23 and TIMER_MAX register settings are valid for PWM timer 23. <code>pwm_irq_new_nperiods</code> and <code>pwm_irq_upd_missed</code> interrupts do not occur within a sequence of NPERIODS PWM periods.

PWM_NPERIODS_45 – PWM nperiods Value Register (Address Offset = 0x2B058)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	NPERIODS_45	RW	0 *	The number of PWM periods THRESHOLD_45 and TIMER_MAX register settings are valid for PWM timer 45. <code>pwm_irq_new_nperiods</code> and <code>pwm_irq_upd_missed</code> interrupts do not occur within a sequence of NPERIODS PWM periods.

PWM_NPERIODS_67 – PWM nperiods Value Register (Address Offset = 0x2B05C)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	NPERIODS_67	RW	0 *	The number of PWM periods THRESHOLD_67 and TIMER_MAX register settings are valid for PWM timer 67. pwm_irq_new_nperiods and pwm_irq_upd_missed interrupts do not occur within a sequence of NPERIODS PWM periods.

PWM_CLK_DIV_01 – PWM Clock Divider Register (Address Offset = 0x2B060)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	CLK_DIV_01	RW		Clock divider value for PWM timer 01. Used to derive the internal used pwm_clk_en signal from the external pwm_clk. The PWM Timer counter only updates (counts up/ down) on a pwm_clk pulse when pwm_clk_en is high.
			0 *	divide-by-1
			1	divide-by-2
			2	divide-by-3
			

PWM_CLK_DIV_23 – PWM Clock Divider Register (Address Offset = 0x2B064)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	CLK_DIV_23	RW		Clock divider value for PWM timer 23. Used to derive the internal used pwm_clk_en signal from the external pwm_clk. The PWM Timer counter only updates (counts up/ down) on a pwm_clk pulse when pwm_clk_en is high.

PWM_CLK_DIV_45 – PWM Clock Divider Register (Address Offset = 0x2B068)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	CLK_DIV_45	RW		Clock divider value for PWM timer 45 Used to derive the internal used <code>pwm_clk_en</code> signal from the external <code>pwm_clk</code> . The PWM Timer counter only updates (counts up/ down) on a <code>pwm_clk</code> pulse when <code>pwm_clk_en</code> is high.

PWM_CLK_DIV_67 – PWM Clock Divider Register (Address Offset = 0x2B06C)*Legend: * reset value*

Bit	Name	Access	Value	Description
15:0	CLK_DIV_67	RW		Clock divider value for PWM timer 67. Used to derive the internal used <code>pwm_clk_en</code> signal from the external <code>pwm_clk</code> . The PWM Timer counter only updates (counts up/ down) on a <code>pwm_clk</code> pulse when <code>pwm_clk_en</code> is high.

4.2.3.1.4 Standard Registers**PWM_IP_TYPE – PWM Type Register (Address Offset = 0x2B0FC)***Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	PRODUCT	R	0x4 *	Product code 0x4 = ARC HS4xD Development Kit
15:8	IP	R	0x5 *	IP code 0x1 = CGU 0x2 = CREG 0x5 = PWM
31:16	DW	R	0x4144 *	Hex code for the two ASCII letters “AD” (Arc Development)

Appendix A

HapsTrak 3 Extension Connector Pins

This appendix provides the pin description of the HapsTrak 3 Extension Connector.

A.1 HapsTrak 3 Extension Connector Pin Description

Table 22 Pin Description of the HapsTrak 3 Extension Connectors J3 And J4

J3-Pin Number	J3-Signal	J4-Pin Number	J4-Signal
A0	he_tunnel_rx_clk	A0	he_tunnel_tx_clk
A1	haps_arc_start	A1	he_tunnel_tx[30]
A2	he_tunnel_rx[2]	A2	he_tunnel_tx[31]
A3	he_tunnel_rx[10]	A3	he_tunnel_tx[32]
A4	haps_boot_start_mode	A4	haps_tdi
A5	haps_boot_core_sel[0]	A5	haps_tdo
A6	he_tunnel_rx[11]	A6	he_tunnel_tx[9]
A7	he_tunnel_rx[19]	A7	he_tunnel_tx_ctrl[1]
A8	haps_boot_core_sel[1]	A8	haps_tms
A9	he_tunnel_rst_an	A9	haps_resetrn_out
A10	haps_boot_multi_core[0]	A10	he_tunnel_bist
A11	haps_boot_multi_core[1]	A11	haps_int
B0	he_tunnel_rx[15]	B0	haps_tck
B1	he_tunnel_rx_ctrl[2]	B1	he_tunnel_tx_ctrl[2]
B2	he_tunnel_rx[4]	B2	he_tunnel_tx[20]
B3	he_tunnel_rx[8]	B3	he_tunnel_tx[24]
B4	he_tunnel_rx[14]	B4	he_tunnel_tx[34]
B5	he_tunnel_rx[0]	B5	he_tunnel_tx[23]
B6	he_tunnel_rx[7]	B6	he_tunnel_tx[25]
B7	he_tunnel_rx[31]	B7	he_tunnel_tx[27]
B8	he_tunnel_rx[1]	B8	he_tunnel_tx[26]
B9	he_tunnel_rx[16]	B9	he_tunnel_tx[21]

J3-Pin Number	J3-Signal	J4-Pin Number	J4-Signal
B10	he_tunnel_rx[33]	B10	he_tunnel_tx[14]
B11	he_tunnel_rx[28]	B11	he_tunnel_tx[15]
C0	he_tunnel_rx_valid	C0	he_tunnel_tx_valid
C1	he_tunnel_rx[3]	C1	he_tunnel_tx[11]
C2	he_tunnel_rx_ctrl[1]	C2	he_tunnel_tx[7]
C3	he_tunnel_rx[9]	C3	he_tunnel_tx[19]
C4	he_tunnel_rx[13]	C4	he_tunnel_tx[18]
C5	he_tunnel_rx[5]	C5	he_tunnel_tx[28]
C6	he_tunnel_rx_ctrl[0]	C6	he_tunnel_tx_ctrl[0]
C7	he_tunnel_rx[18]	C7	he_tunnel_tx[4]
C8	he_tunnel_rx[34]	C8	he_tunnel_tx[33]
C9	he_tunnel_rx[35]	C9	he_tunnel_tx[35]
<u>C10</u>	he_tunnel_rx[17]	<u>C10</u>	he_tunnel_tx[1]
C11	he_tunnel_rx[6]	C11	he_tunnel_tx[22]
D0	he_tunnel_rx[20]	D0	he_tunnel_tx[6]
D1	he_tunnel_rx[23]	D1	he_tunnel_tx[2]
D2	he_tunnel_rx[26]	D2	he_tunnel_tx[5]
D3	he_tunnel_rx[29]	D3	he_tunnel_tx[8]
D4	he_tunnel_rx[21]	D4	he_tunnel_tx[10]
D5	he_tunnel_rx[30]	D5	he_tunnel_tx[0]
D6	he_tunnel_rx[32]	D6	he_tunnel_tx[12]
D7	he_tunnel_rx[25]	D7	he_tunnel_tx[3]
D8	he_tunnel_rx[27]	D8	he_tunnel_tx[13]
D9	he_tunnel_rx[12]	D9	he_tunnel_tx[16]
D10	he_tunnel_rx[24]	D10	he_tunnel_tx[17]
D11	he_tunnel_rx[22]	D11	he_tunnel_tx[29]

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

CGU

Clock Generator Unit

DDR3

Double Data Rate 2

GPIO

General Purpose Input/Output

HW

Hardware

HAPS

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

HapsTrak 3

Standard (SAMTEC) connector type used on HAPS

HSDK

HS Development Kit

HSDC

HS Development SoC

IC

Integrated Circuit

I²S

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

JTAG

Joint Test Action Group

R

Read-only register

RW

Read-write register

RW1C

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

SDRAM

Synchronous Dynamic Random Access Memory

SRAM

Static Random Access Memory

SW

Software

References

- [1] *HapsTrak 3 standard, Connector type:* <https://www.samtec.com/products/seam-20-02.0-s-08-2-a-k-tr>
- [2] *MikroBUS™ Standard Specification,* <https://download.mikroe.com/documents/standards/mikrobus/mikrobus-standard-specification-v200.pdf>
- [3] 8-input 10-bit ADC108S102 <http://www.ti.com/product/adc108s102>
- [4] CY8C9520A I/O expander <http://www.cypress.com/documentation/datasheets/cy8c9520a-cy8c9540a-cy8c9560a-20-40-and-60-bit-io-expander-eeeprom>
- [5] *DesignWare® MetaWare Debugger User's Guide for ARC®*
- [6] *Synopsys DesignWare dw_apb_gpio Databook* <http://www.synopsys.com>
- [7] DesignWare ARC HS Series Databook
- [8] DesignWare ARCV2 ISA Programmer's Reference Manual for ARC HS processors
- [9] DesignWare Cores Enhanced Universal DDR Memory Controller (uMCTL2) Databook
- [10] DesignWare Cores Ethernet MAC Universal Databook
- [11] DesignWare Cores USB 2.0 Host AHB Controller Databook
- [12] DesignWare Cores USB 2.0 picoPHY for TSMC28nm HPM Databook
- [13] DesignWare Cores Mobile Storage Host Databook
- [14] DesignWare DW_apb_uart Databook
- [15] DesignWare DW_apb_ssi Databook
- [16] DesignWare DW_apb_i2c Databook
- [17] DesignWare DW_apb_i2s Databook
- [18] DesignWare DW_apb_wdt Databook
- [19] DesignWare DW_axi_dmac Databook
- [20] Redpine Signals RS9113-WiseConnect-API-Guide-v1.7.1: www.redpinenetworks.com