DesignWare ARC HS Development Kit User Guide

Version 5793-002 January 2018

Copyright Notice and Proprietary Information Notice

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.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

List of Figures

List of Tables

1
Customer Support

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

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet 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 SolvNet:

1. Go to the SolvNet Web page at http://solvnet.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 SolvNet.)

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

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to http://solvnet.synopsys.com/ (Synopsys user name and password required), then clicking “Enter a Call to the Support Center.”
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at https://www.synopsys.com/support/global-support-centers.html.
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at https://www.synopsys.com/support/global-support-centers.html.

2 Introduction

2.1 Package Content

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

- DesignWare ARC HS Development Kit (ARC HSDK)
- A 100-240 V AC power adapter (including adapters for U.S., UK, and EU outlets)
- USB cable

Warning

The DesignWare ARC HS 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 Uboot

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.

2.2.2 Checking Default Board Settings

Check if the boot switches and jumpers are set to their default positions. See section Configuration and Boot Modes for an overview of the configuration options and the default settings.

Connect the ARC HSDK to your PC by connecting the USB cable to the USB data port of the ARC HSDK and the PC.

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

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

2.2.3 Installing and Configuring PuTTY

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

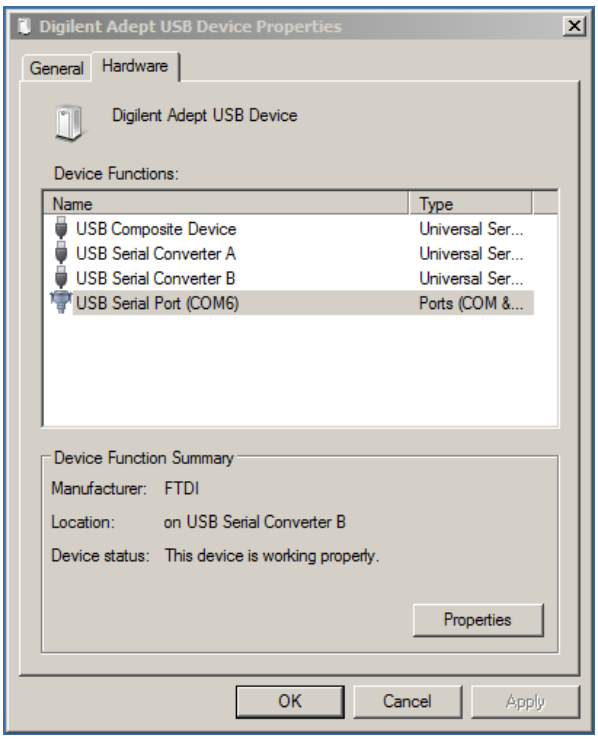
1. Download `putty.exe` from http://www.putty.org
2. Open the Windows **Control Panel**.
3. In the category **Hardware and Sound**, click **View devices and printers**, and **Digilent Adept USB Device**.

   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

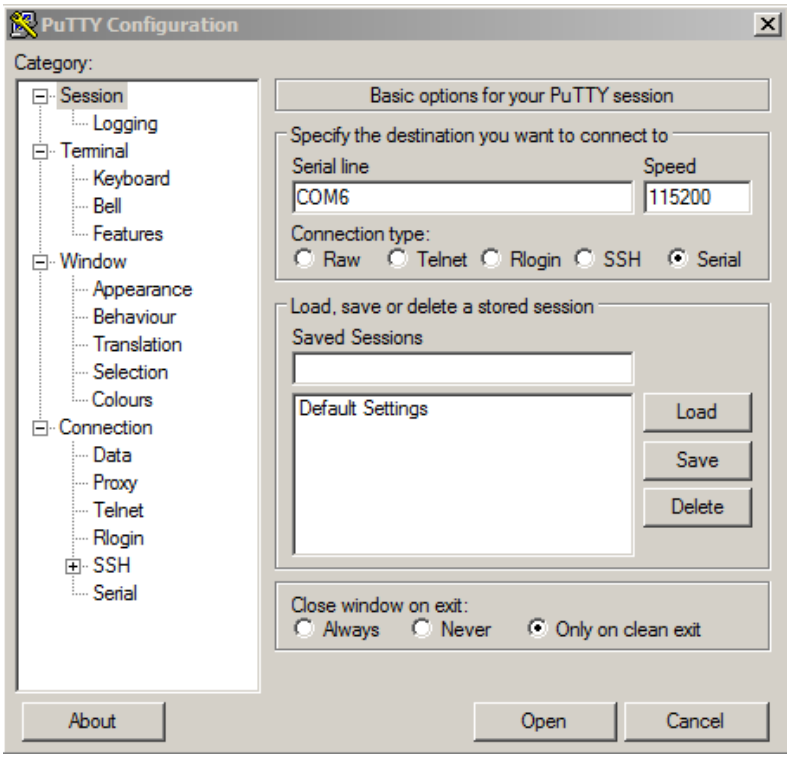


4. Execute `putty.exe`.

   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.

2.2.4 Starting Uboot

By default, you press the start button on the ARC HSDK for the ARC HS core to start executing the bootloader. After you press the start button, you see text similar to the following figures.

- Figure 3 shows a log from default booting, followed by Uboot commands to initialize the HSDK board in single HS38 mode and start an application called `app.bin` that resides on the SD card.
- *Figure 4* on page 12 shows a log from default booting followed by Uboot commands to initialize the HSDK board in HS38x4 mode, re-program several clocks, and start Linux from an image on the SD card (`uImage`).

Figure 3 Default Boot with ARC HS38 Initialization

Figure 4 Default Boot with ARC HS38x4 Initialization and Custom Clock

2.3 Location of Components On ARC HSDK

Figure 5 and Figure 6 show the placement of various components on the DesignWare ARC HS Development Kit.

Figure 5 ARC HSDK Components: Top View

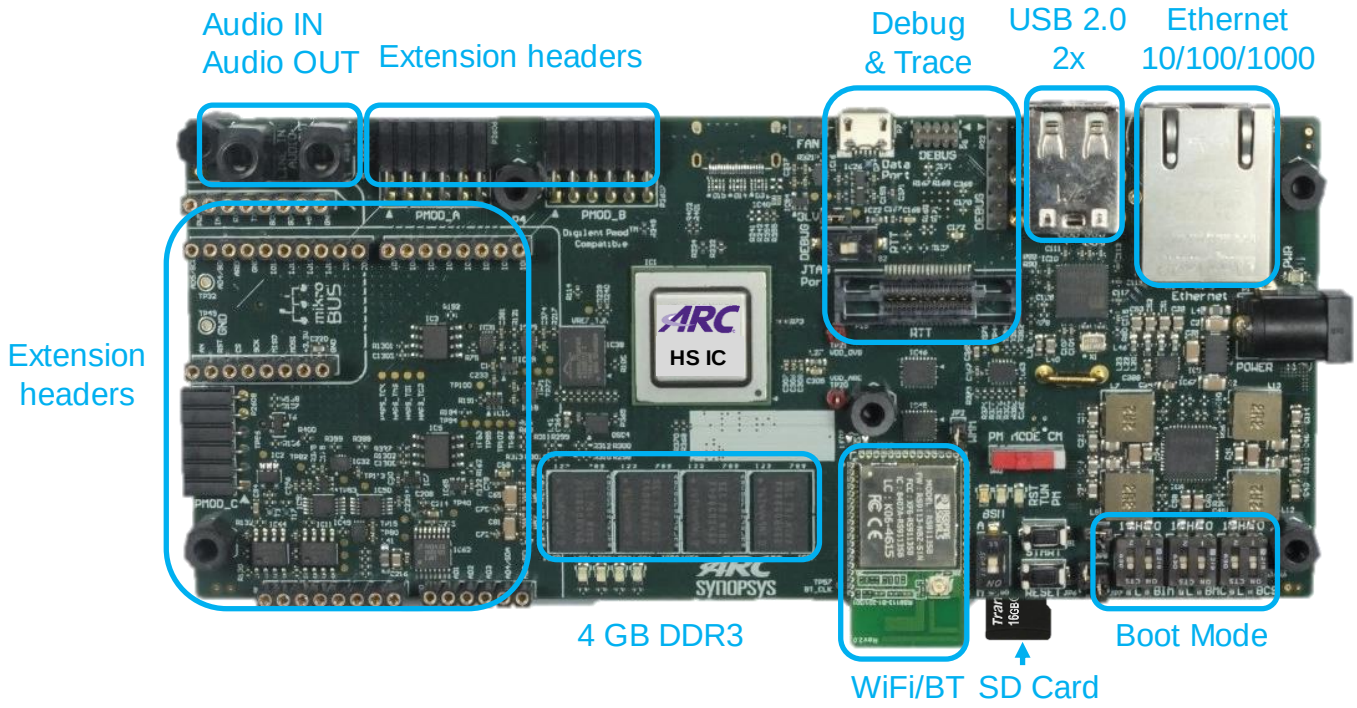
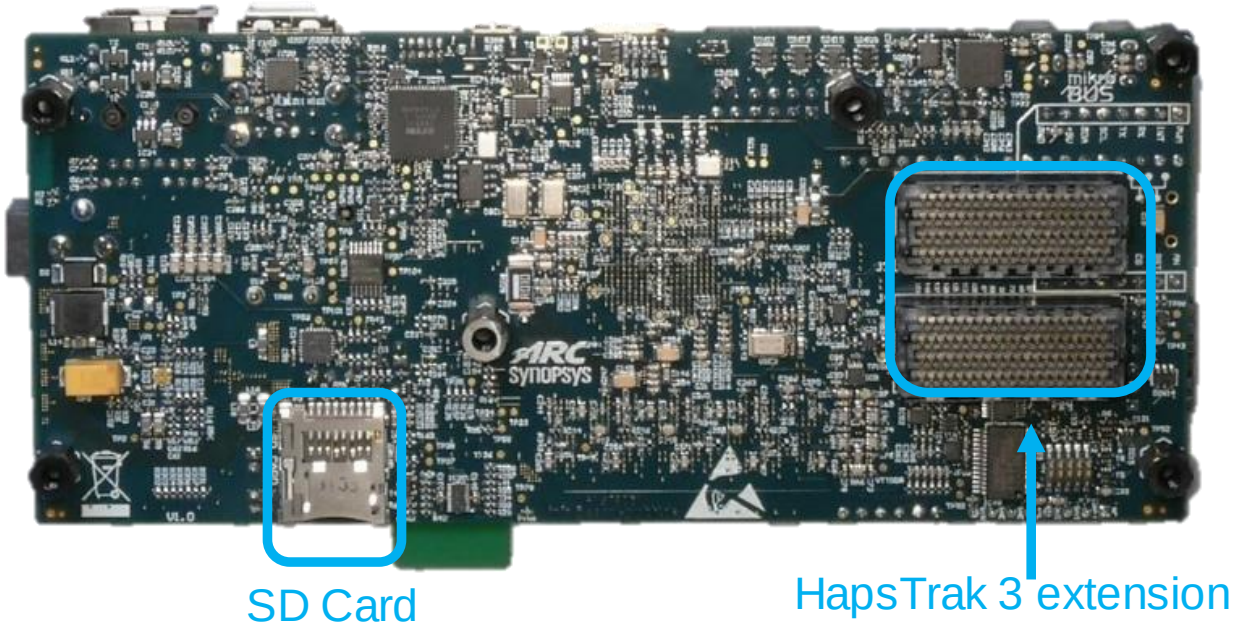


Figure 6 ARC HSDK Components: Bottom View



2.4 Software Packages

See the http://embarc.org/ portal for information on the available software packages for the ARC HSDK.

Direct links to the Software release are available here:

- Linux software release for ARC HSDK are available here:

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

- embARC software releases for ARC HSDK are available here:

  https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_osp/releases

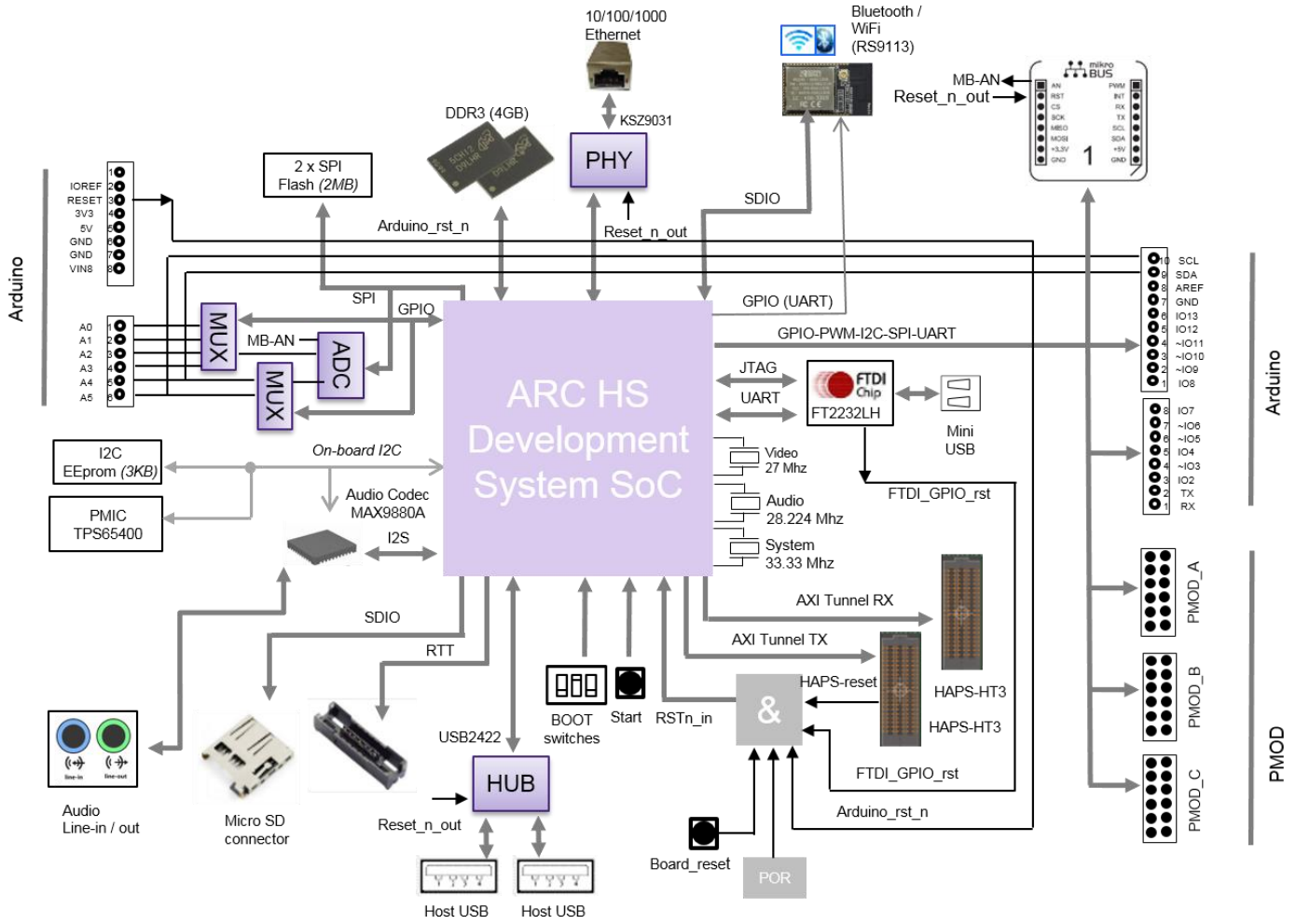
Additional documentation on how to get started can be found here:

https://github.com/foss-for-synopsys-dwc-arc-processors/ARC-Development-Systems-Forum/wiki/ARC-Development-Systems-Forum-Wiki-Home

3 Hardware Description

3.1 Overview of ARC HSDK

Figure 7 ARC HSDK Block Diagram



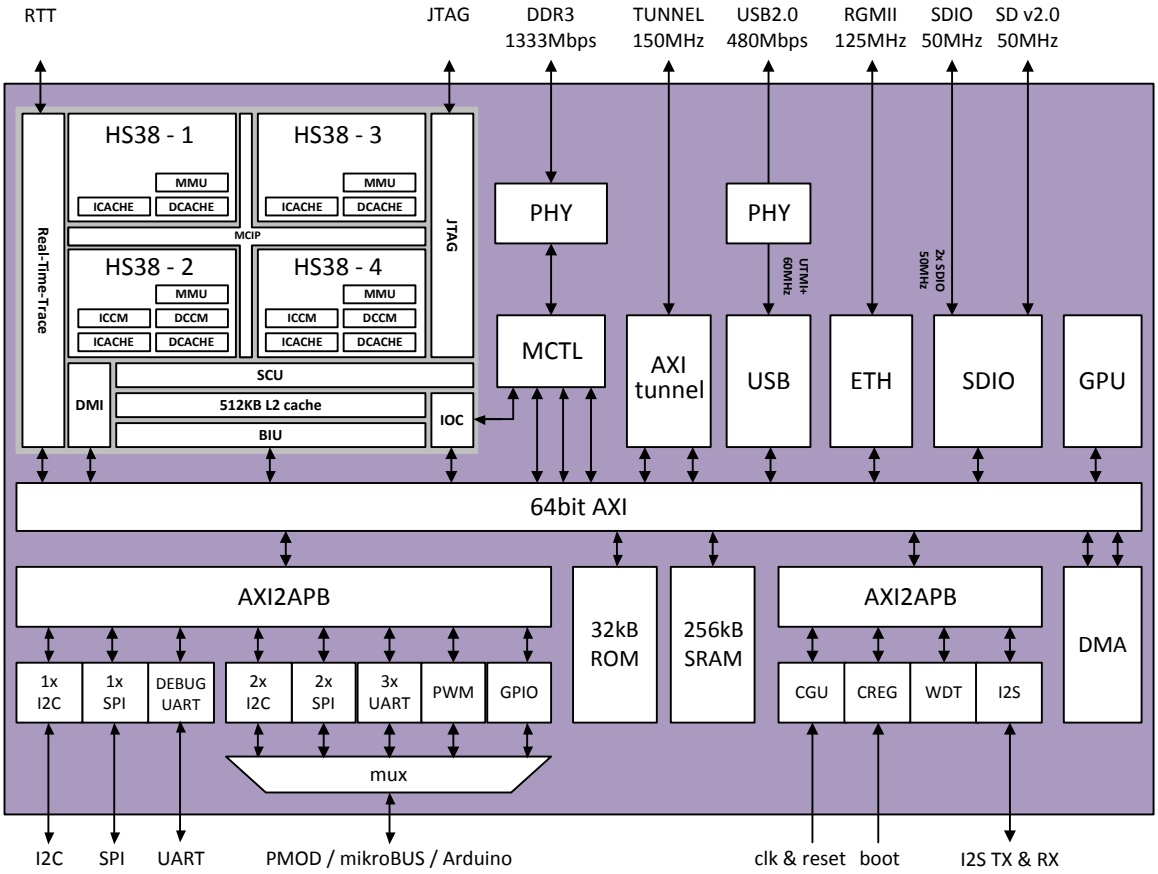
DesignWare ARC HS Development Kit contains the following components:

- ARC HS Development System SoC
  - Quad Core ARC HS38x4
  - 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
  - USB2 (2x)
  - Ethernet (10/100/1000)
  - Audio line in/out
  - USB Data port (JTAG/UART)
  - Micro- SD Card
  - WIFI/BT 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 HS Development System SoC

Figure 8 ARC HS Development System SoC Top-Level Diagram



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

- Flexible, customizable IC architecture
  - Configurable/programmable boot scenarios
  - Configurable/programmable memory map
- DesignWare ARC HS38x4 quad-core @ 1GHz
  - 64kByte instruction cache
  - 64kByte data cache
  - 256kByte ICCM (2 cores)
  - 256kByte DCCM (2 cores)
  - Memory Management Unit
  - Physical Address Extension (PAE)
  - 512kByte L2 cache
  - Support for I/O coherency
  - Support for ARC Real-Time Trace
- Vivante GC7000 NanoUltra3T GPU Processing Unit @ 400Mhz

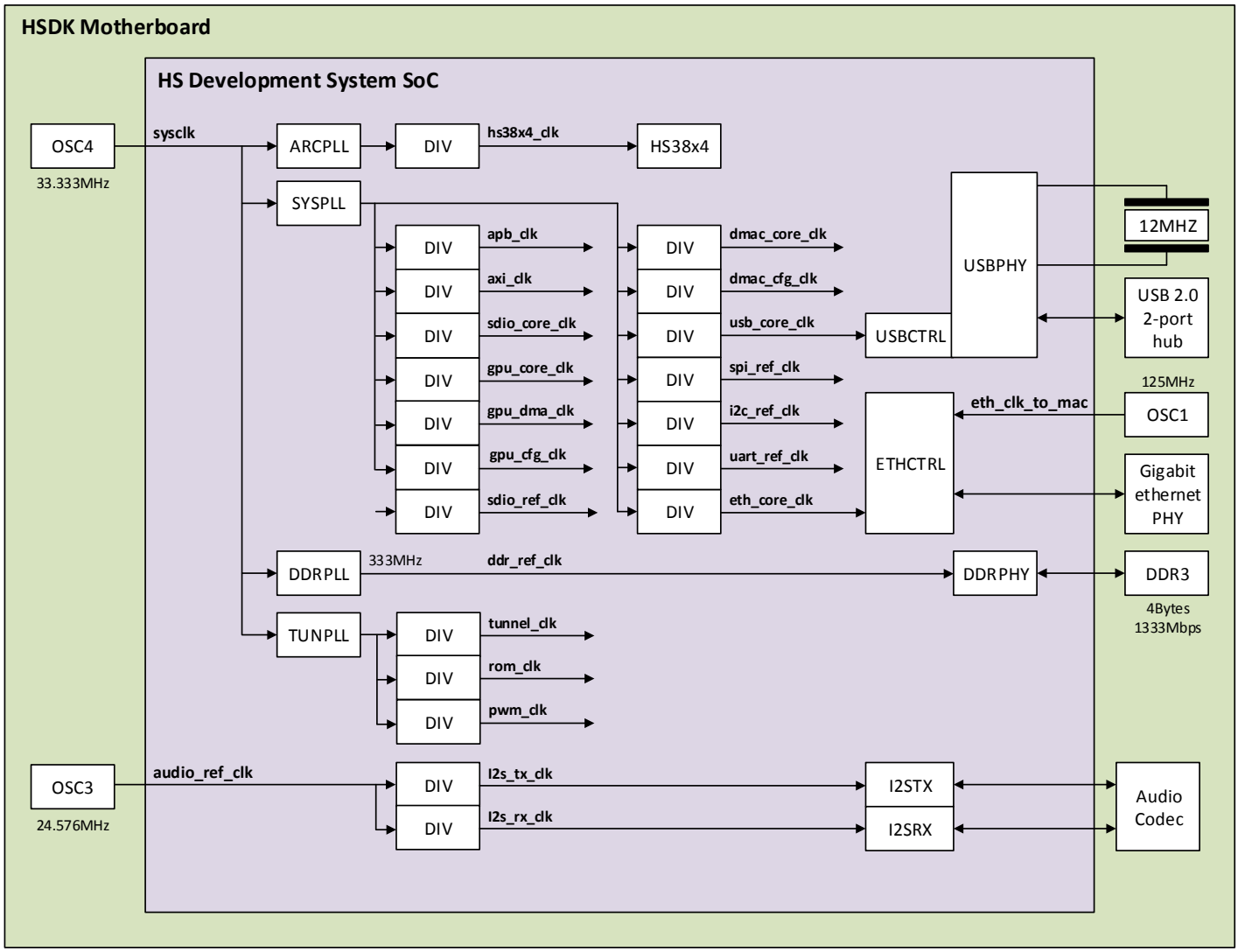
- Flexible clock generation
    - System locks
        - 33MHz system reference clock input
        - PLL for DDR clock
        - PLL for TUNNEL clock
        - PLL for ARC clock
        - PLL for all other system clocks
    - Audio
        - Audio reference clock input (24.576 MHz)
        - Integer divider(s) for audio serial clock
- 256kByte SRAM
- DDR3 interface
    - Max speed grade DDR-1600 (800MHz)
    - 32-bit data width
    - Max row address width of 16 bits
    - Max bank address width: 3 bits
    - Max two memory ranks
    - Max supported DDR memory size is 4GByte
- 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
    - Max freq of 150MHz
    - Max data throughput of 600MByte/s
- UART interfaces
- I2C interfaces
- SPI interfaces
- PWM interfaces
- JTAG interface

3.3 Clocks and Resets

3.3.1 Clocks

The ARC HS Development System SoC uses a single 33MHz 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. See section Clock Registers for more details on the CGU. Figure 9 depicts the ARC HSDK clock architecture.

Figure 9 ARC HSDK Clock Architecture



A summary of all the clocks and their sources is listed in Table 1.

Table 1: Overview Of ARC HSDK Clock Components

HS IC Clock	Clock Source	Default Value	Description
hs38x4_clk	sysclk	1000	Clock for ARC HS38x4 (including ARC RTT and ARConnect)
apb_clk	sysclk	200	Clock for APB peripherals:
axi_clk	sysclk	800	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
gpu_core_clk	sysclk	800	Core clock 2x for GPU
gpu_dma_clk	sysclk	400	DMA clock for GPU
gpu_cfg_clk	sysclk	200	Configuration clock for 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	200	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. I2C reference clock must be greater than or equal to the APB clock
uart_ref_clk	sysclk	200	Reference clock for UART baud rate. UART reference clock must be less than or equal to APB clock
ddr_ref_clk	sysclk	400	Reference clock for DDR controller + PHY (400Mhz for DDR-1600). This clock is driven directly by the PLL; all the integer dividers inside the CGU are bypassed.
tunnel_clk	sysclk	150	Clock for AXI tunnel
rom_clk	sysclk	250	Clock for ROM controller
pwm_clk	sysclk	150	Clock for PWM controller
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
Audio_ref_clk	OSC3	24.576	Fixed clock frequency for I2S interfaces and audio codec
Sys_clk	OSC4	33.333	Fixed system clock for ARC HS Development System SoC

3.3.2 **Reset**

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

When the ARC HS 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).

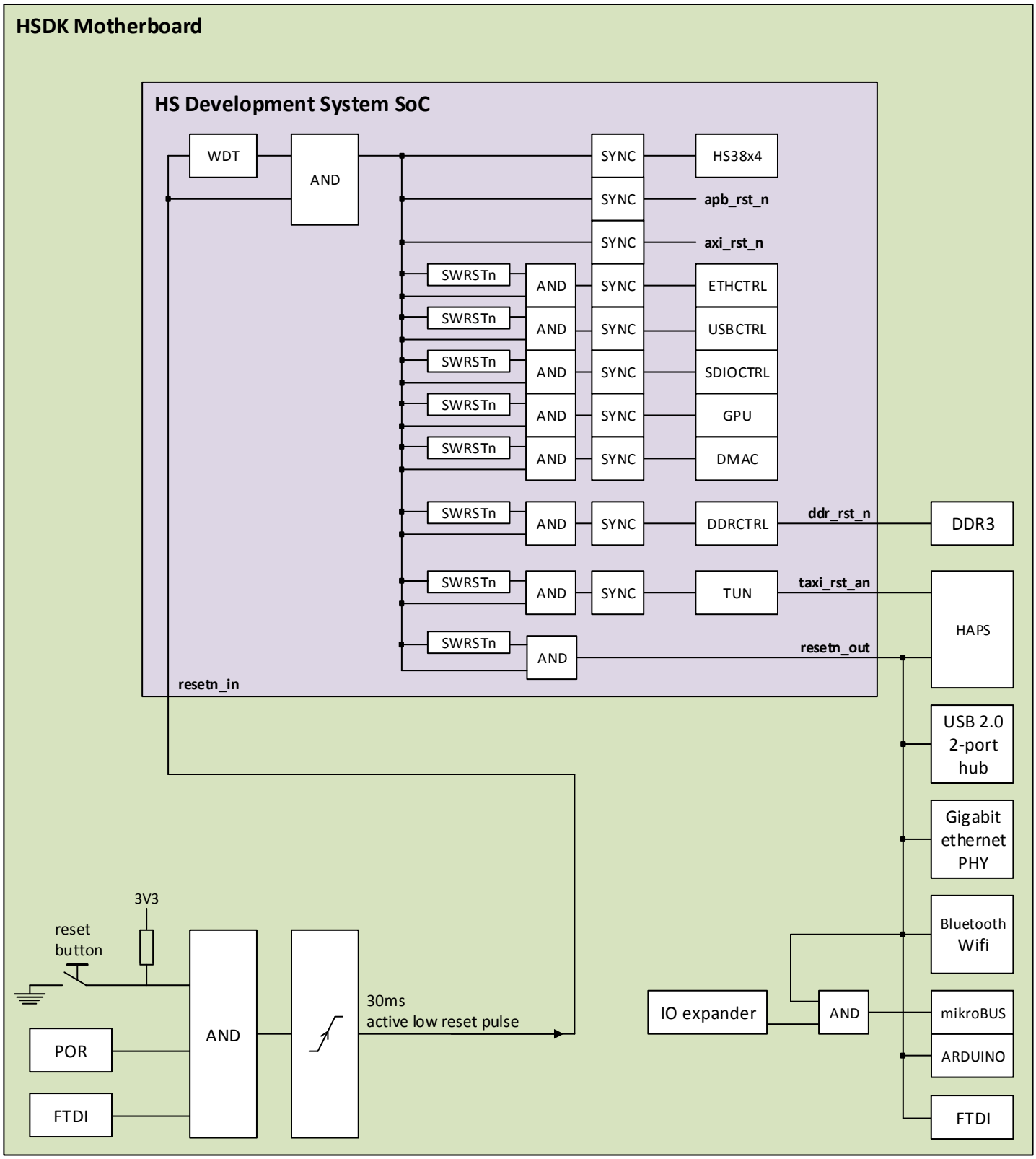
The ARC HS Development System SoC has one external reset pin (`resetn_in`) that serves as an active low, hardware reset. When the external hardware reset is active, the entire chip is reset. The chip does not have an on-chip power-on-reset module so it relies on the external circuitry to keep `resetn_in` 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 HS Development System SoC is out of reset, `resetn_out` is asserted resetting all the ARC HSDK 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 HS Development System SoC reset.

Figure 10 Reset Architecture



3.4 Interrupts

Figure 11 shows the top-level interrupt architecture of the ARC HSDK. The ARC HSDK 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 HS38x4.
- Internal hardware interrupts: generated by the on-chip interrupt sources (for example: I2C, UART, SPI).

The interrupt mapping for the ARC HS38x4 core is listed in Table 2. All the above motioned interrupts sources are connected as external common interrupt to the interrupt distribution unit (IDU). The IDU distributes the external common interrupts to the cores in the ARC HS38x4. All interrupts are active high and level sensitive unless noted differently.

Figure 11 Interrupt Architecture

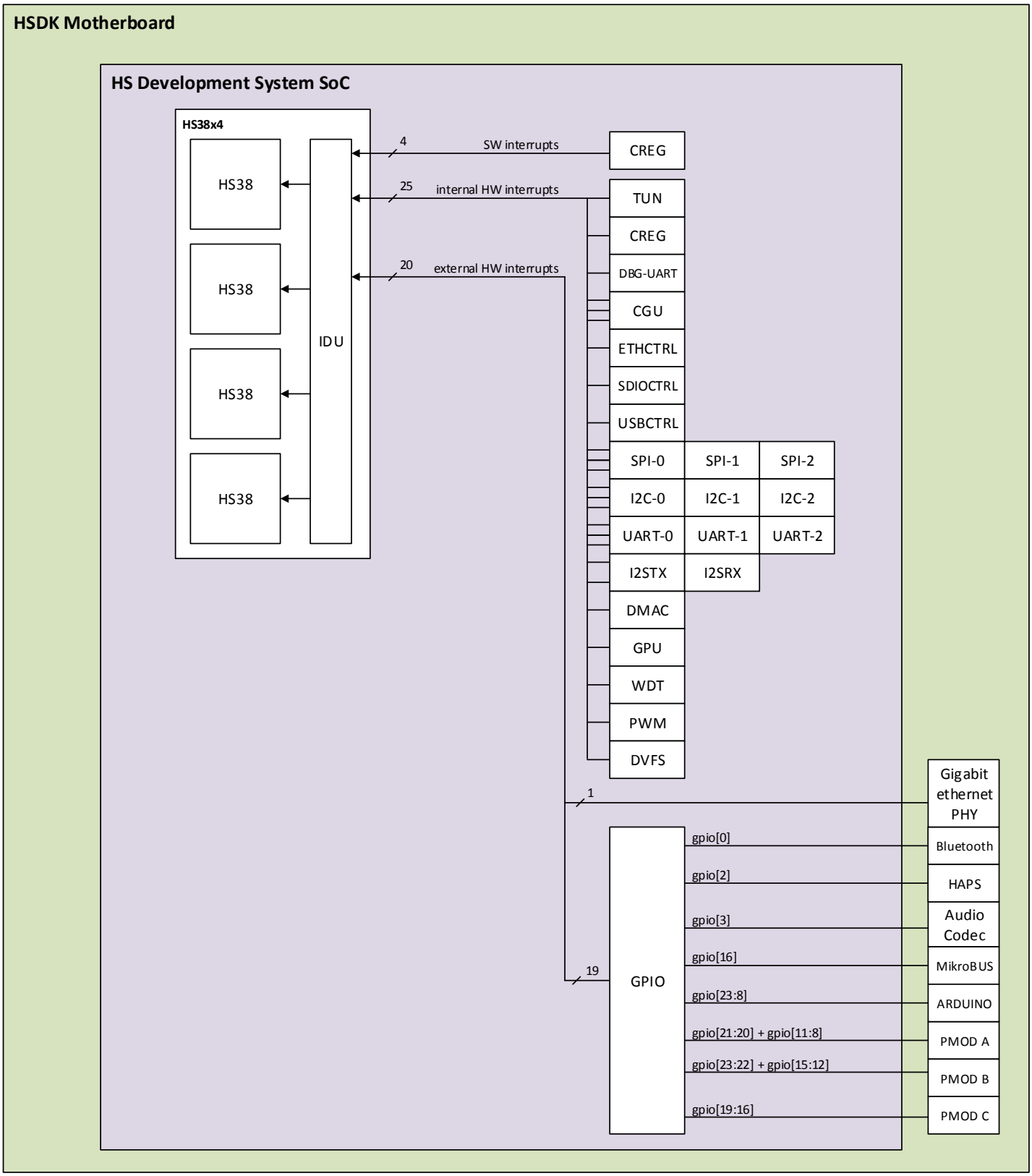


Table 2 Interrupt Mapping

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 SDIO	
cirq13_a	irq37_a	reserved	
cirq14_a	irq38_a	reserved	
cirq15_a	irq39_a	internal HW interrupt from USB-HOST	
cirq16_a	irq40_a	internal HW interrupt from SPI-0	
cirq17_a	irq41_a	internal HW interrupt from SPI-1	
cirq18_a	irq42_a	internal HW interrupt from SPI-2	
cirq19_a	irq43_a	internal HW interrupt from I2C-0	
cirq20_a	irq44_a	internal HW interrupt from I2C-1	
cirq21_a	irq45_a	internal HW interrupt from I2C-2	
cirq22_a	irq46_a	internal HW interrupt from UART-0	
cirq23_a	irq47_a	internal HW interrupt from UART-1	
cirq24_a	irq48_a	internal HW interrupt from UART-2	
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	
cirq32_a	irq56_a	external HW interrupt from GPIO[0]	Bluetooth interrupt of RS9113 module
cirq33_a	irq57_a	external HW interrupt from GPIO[0]	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	external HW interrupt from GPIO[4]	N/A, GPIO[4] is not connected
cirq37_a	irq61_a	external HW interrupt from GPIO[5]	N/A, GPIO[5] is used a UART1 TXD signal for the Bluetooth interface of the RS9113
cirq38_a	irq62_a	external HW interrupt from GPIO[6]	N/A, GPIO[6] is used a UART1 RXD signal for the Bluetooth interface of the RS9113
cirq39_a	irq63_a	external HW interrupt from GPIO[7]	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

3.5 Debug and Trace

The ARC HSDK offers a rich set of debug and trace options:

Debug:

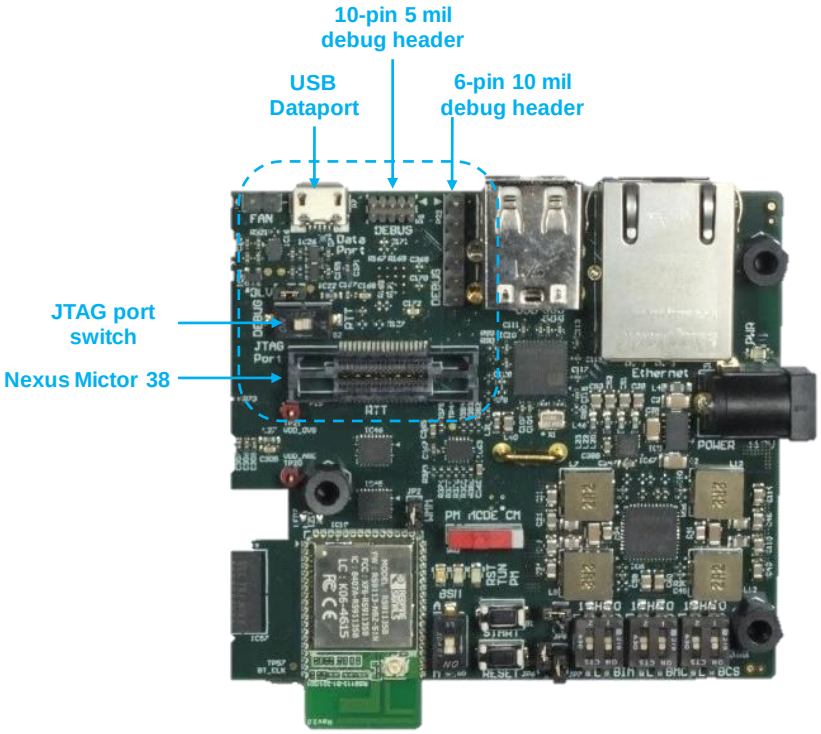
- USB cable connected to on-board 2-channel FTDI chip
  - Compatible with MetaWare debugger and GDB
- Ashling Opella-XD, Lauterbach Trace-32, Digilent HS1 and 2 support through:
  - A 10-pin 5 mil debug header and an adapter cable
  - A 6-pin 10 mil debug header and flying leads

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 Debug and Trace Headers



3.5.1 Debug

The ARC HS core provides debug access through an IEEE 1149.1 JTAG port. The four ARC HS cores in the ARC HS38x4 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 HS Development IC is shown in Figure 13. To

distinguish between the individual cores in the JTAG chain each core has a unique JTAG IDCODE.

Figure 13JTAG Daisy Chain

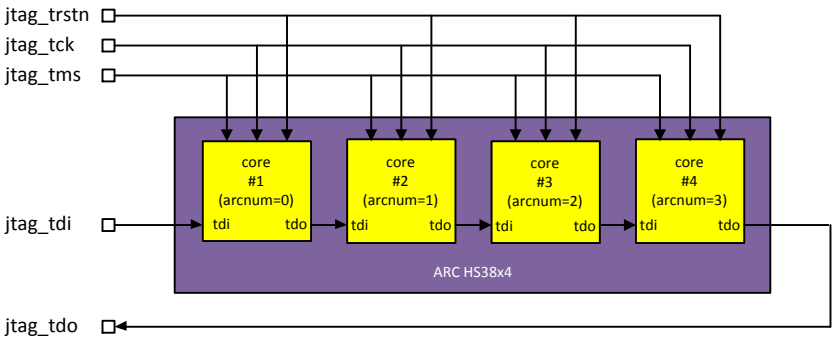


Table 3 ARC ID codes

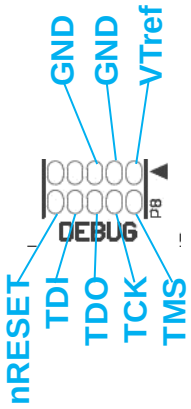
Core	ARC ID	CPUNUM in MDB
ARC HS38x4 #1	0x052	1
ARC HS38x4 #2	0x152	2
ARC HS38x4 #3	0x252	3
ARC HS38x4 #4	0x352	4

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



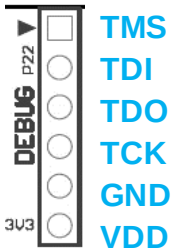
This header can be used to connect a standard 20-pin Ashling or Lauterbach probe header using an adapter as depicted below

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

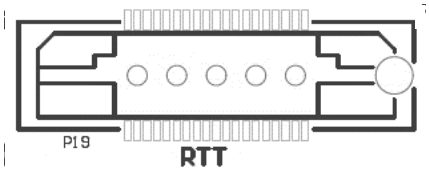
The ARC HS Development IC implements Real-Time-Trace capability (ARC RTT). ARC RTT allows configurable/programmable monitoring of:

- 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 16bit high-speed interface and for the ARC HS Development System SoC supports up to 100MHz trace clock.

3.5.2.1 Nexus Mictor 38 Interface

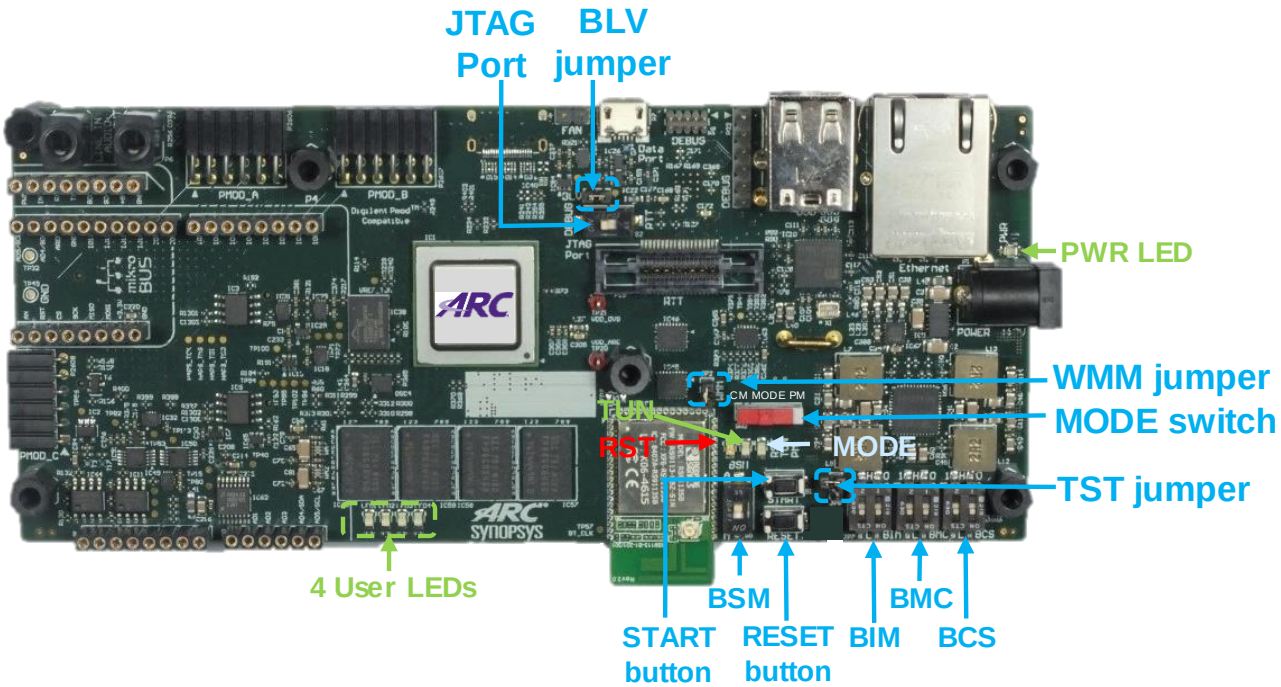


The ARC RTT interface is supported with the Ashling Ultra-XD and Lauterbach Trace-32 products.

3.6 Configuration and Boot Modes

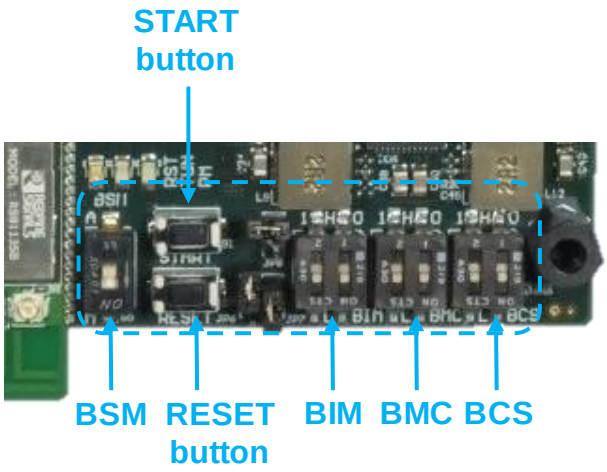
Figure 15 depicts the configuration switches, jumpers, and LEDs available on the ARC HSDK. Besides jumpers and standard Reset and Start buttons there are also switches to control the board boot mode and debug mode. More detail of each of the jumpers and switches are explained in the following sections.

Figure 15 ARC HSDK Configuration and Boot Switches and Buttons

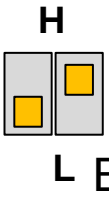


3.6.1 **Boot Switches**

Figure 16 ARC HSDK Boot Switches

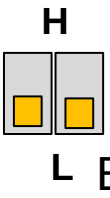


Boot IMage location (BIM) Switches



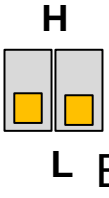
Ensure that these switches are set as depicted on the left (**default position**). This setting ensures that the pre-bootloader starts executing the Uboot bootloader that is stored in SPI Flash. All other settings for these switches are reserved and must not be used.

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 Uboot bootloader.

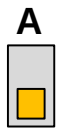
Boot Core Select (BCS) switches.



Ensure that these switches are set as depicted on the left (**default position**). This setting ensures that the ARC HS Core 1 starts executing Uboot. 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:

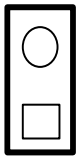


Manual mode. The ARC HSDK only starts booting after the *START* button is pushed. This is the **default setting**

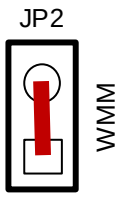
Automatic mode. The ARC HSDK automatically starts booting after *Reset*.

3.6.2 **Jumpers**

The ARC HSDK includes the following jumpers: TST, BLV, and WMM.

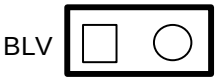


The TST jumper (JP8) is used for production purposes only and must not be used in the normal course of operation. **Default position**: open



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 HS Core. If the jumper is *open,* the module operates in the embedded mode using the UART1 interface,and hence the full networking stack is running on the RS9113 module itself.

Default position: closed

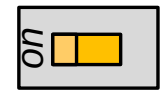


The BLV jumper (Boot Loader Verbosity) is used to control the verbosity of the bootloader messages. If the jumper is *open,* the boot loader prints the normal boot messages without additional system/debug info. If the jumper is closed, the boot loader prints addittional messages during execution.

Default position: open

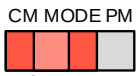
3.6.3 Other Switches

Besides the bootmode switches, the ARC HSDK 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 postion (left), the JTAG signals are availble 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 avialable on the ARC RTT connector. **Default position:** on.



sw2

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

3.6.4 On-board LEDs

The ARC HSDK 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 Status and System LEDs

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

3.7 Memories

The ARC HSDK features the following memories

- 4 GByte DDR3-1333 memory
- 2 Mbyte SPI Flash
  - This SPI Flash is pre-loaded with the Uboot bootloader
- 3 Kbyte I2C EEprom

3.8 USB Interface

The ARC HSDK offers two USB 2.0 host ports. The ARC HS 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 (480Mbps) transfers using an EHCI Host Controller, as well as full (12Mbps) and low (1.5Mbps) 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 ARC HSDK, 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 HS 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 ARC HSDK 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 Uboot, the default settings operate the SD card in the SDR25 speed mode.

3.11 Audio Interface

The ARC HSDK 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 HS 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 HS 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 24.576MHz.

The ARC HSDK 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				
16kHz	32kHz	48kHz	96kHz	192kHz
24	12	8	4	2

3.12 On-board I2C Control Bus

The ARC HSDK 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 offered by I2C0.

Table 7 ARC HSDK On-board I2C Save Addresses

Device Name	Description	7-bit I²C Address
TPS65400	Power Management IC (PMIC)	1101001 = 0x69
MAX 9880A	Audio Codec	0010000 = 0x10
CY8C9520A	I/O Expander part 3 Kbyte EEprom part	0100000 = 0x20 1000000 = 0x40

3.13 ADC

The ARC HSDK 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

3.14 I/O Expander

The ARC HSDK 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

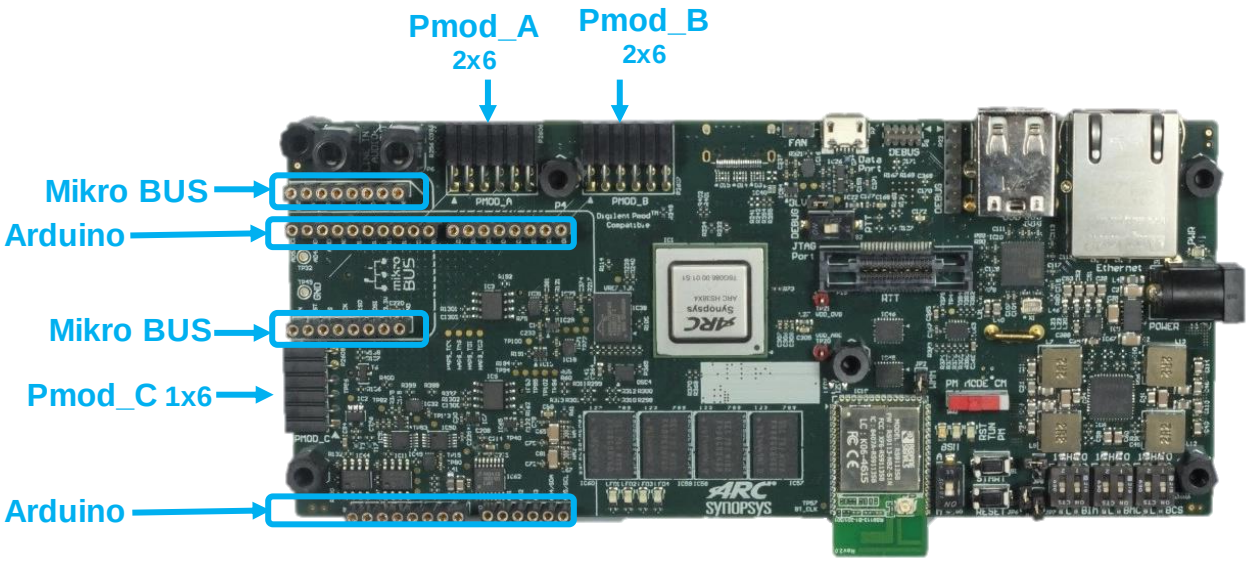
Port0	Usage	Port1	Usage
bit0	RS9113 Bluetooth I2S RX enable (active low)	bit0..3	reserved
bit1	mikroBUS Reset (active low)	bit4	On-board user LED 1
bit2	GPIO for Arduino AD0	bit5	On-board user LED 2
bit3	GPIO for Arduino AD1	bit6	On-board user LED 3
bit4	GPIO for Arduino AD2	bit7	On-board user LED 4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

3.15 Extension Interfaces

To bring your application context around the ARC HSDK, the following peripheral module standards are supported:

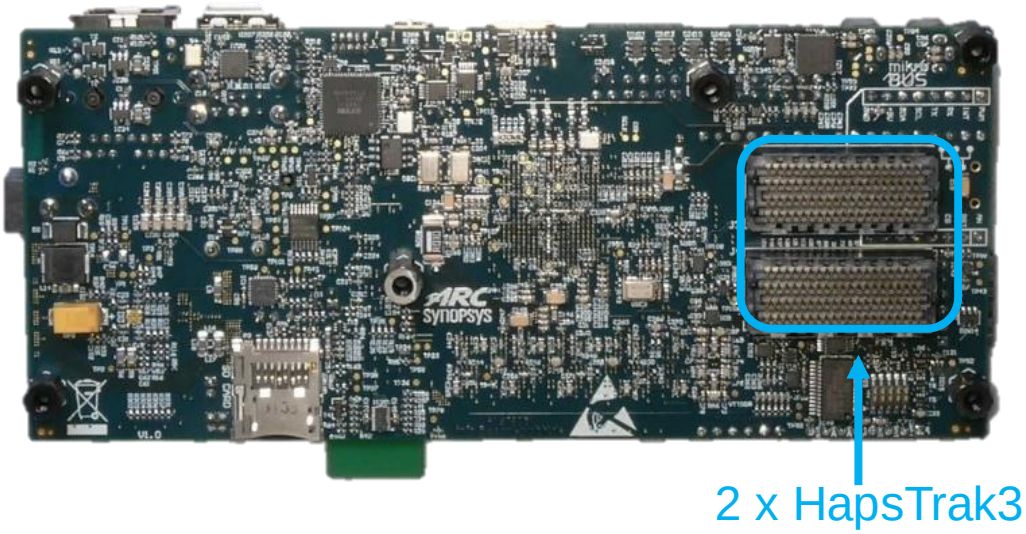
- Digilent Pmod™ (3x)
- MikroBUS (1x)
- Arduino (1x)

Figure 17 ARC HSDK Peripheral Extension Interfaces



Further, the ARC HSDK features two HapsTrak 3 connectors that allow you to connect the ARC HSDK to a HAPS prototyping system. See section HapsTrak 3 Extension for more details regarding this extension option.

Figure 18 ARC HSDK HAPS Extension Interface



3.15.1 Digilent Pmod™

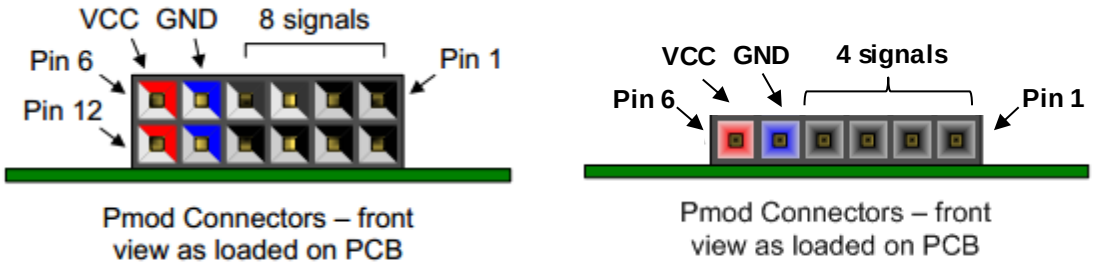
The ARC HSDK features two 12-pin Pmod connectors `Pmod_A` and `Pmod_B` and one 6-pin Pmod connector `Pmod_C`. The functionality of the Pmod connectors is programmable and includes GPIO [6], UART [14], SPI [15], I2C [16] and PWM. The available options are summarized in Table 10. The available options 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



3.15.1.1 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

3.15.1.2 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

3.15.1.3 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

3.15.2 Mikrobus

The ARC HSDK 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 MikroElektronica (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).

Figure 20 MikroBus Headers

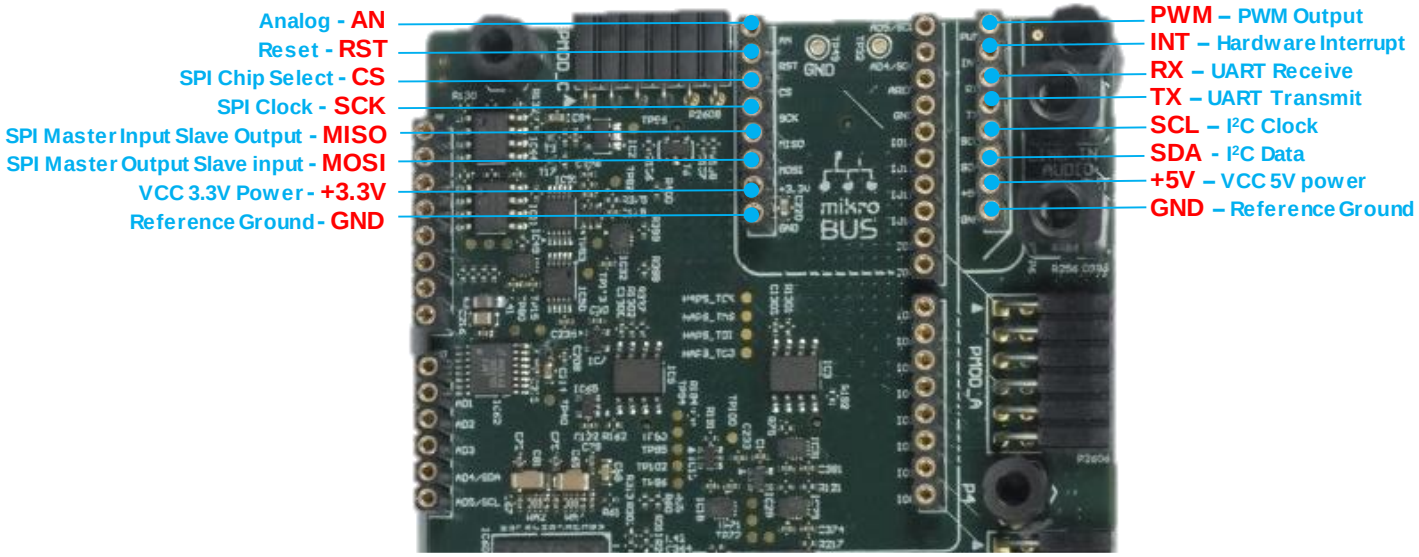


Table 14 shows the pin assignment on the I/O Multiplexer.

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.

3.15.3 Arduino

The ARC HSDK 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 ARC HSDK 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

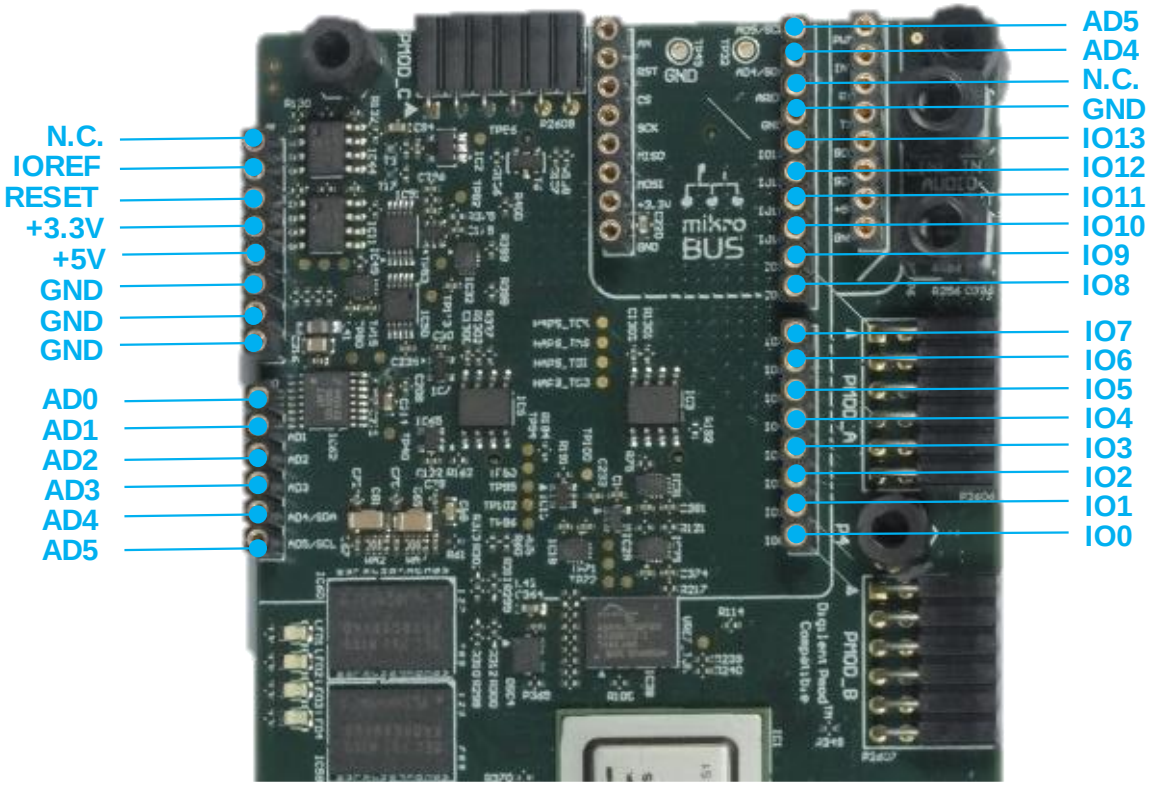


Table 15 shows the pin assignment on the I/O Multiplexer. Multiplexing is controlled by software using the `PMOD_MUX_CTRL` register (see Mux). After a reset, all ports are configured as GPIO inputs.

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.

3.15.4 HapsTrak 3 Extension

The DesignWare ARC HS Development Kit includes two HapsTrak-3 [1] connectors to allow further extensions of the ARC HSDK 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 Resetn-Out

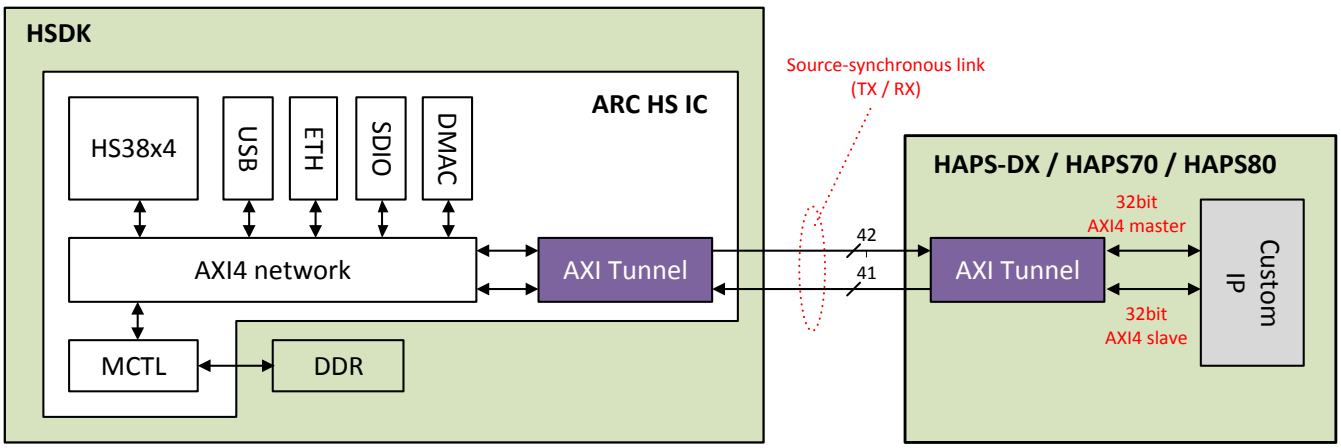
- Interrupt, connected to an ARC HS IC GPIO input.

- 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 0.

In Figure 22, a typical usage of the AXI tunnel to add custom IP into the ARC HSDK 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 Mbyte/s.

Figure 22 ARC HSDK HAPS Extension: Typical Usecase



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

4
Programmer's Reference

4.1 Memory Map

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

Figure 23 ARC HSDK Memory Map

Core address offset	HS34 slave	HS36 slave	HS38 slave
0x1_F000_0000	N/A	N/A	DDR [7]
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]
0x1_0000_0000			DDR [8]
0x0_F000_0000	HSDK APB	HSDK APB	HSDK APB
0x0_E000_0000	HAPS APB	HAPS APB	HAPS APB
0x0_D000_0000	DDR [5]	DDR [5]	DDR [5]
0x0_C000_0000	DDR [4]	DDR [4]	DDR [4]
0x0_B000_0000	DDR [3]	DDR [3]	DDR [3]
0x0_A000_0000	DDR [2]	DDR [2]	DDR [2]
0x0_9000_0000	DDR [1]	DDR [1]	DDR [1]
0x0_8000_0000	DCCM	DDR [0]	DDR [0]
0x0_7000_0000	ICCM	DDR [15]	DDR [15]
0x0_6000_0000	DDR [14]	DDR [14]	DDR [14] or I+DCCM
0x0_5000_0000	DDR [13]	DDR [13]	DDR [13]
0x0_4000_0000	DDR [12]	DDR [12]	DDR [12]
0x0_3000_0000	DDR [11]	DDR [11]	DDR [11]
0x0_2000_0000	DDR [10]	DDR [10]	DDR [10]
0x0_1000_0000	DDR [9]	DDR [9]	DDR [9]
0x0_0000_0000	DDR [8]	DDR [8]	DDR [8]

PAE region

DMA clients address offset	slave
0xF000_0000	DDR [7]
0xE000_0000	DDR [6]
0xD000_0000	DDR [5]
0xC000_0000	DDR [4]
0xB000_0000	DDR [3]
0xA000_0000	DDR [2]
0x9000_0000	DDR [1]
0x8000_0000	DDR [0]
0x7000_0000	DDR [15]
0x6000_0000	DDR [14]
0x5000_0000	DDR [13]
0x4000_0000	DDR [12]
0x3000_0000	DDR [11]
0x2000_0000	DDR [10]
0x1000_0000	DDR [9]
0x0000_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 256MByte. Accessing a non-existing APB peripheral results in an AXI error response.

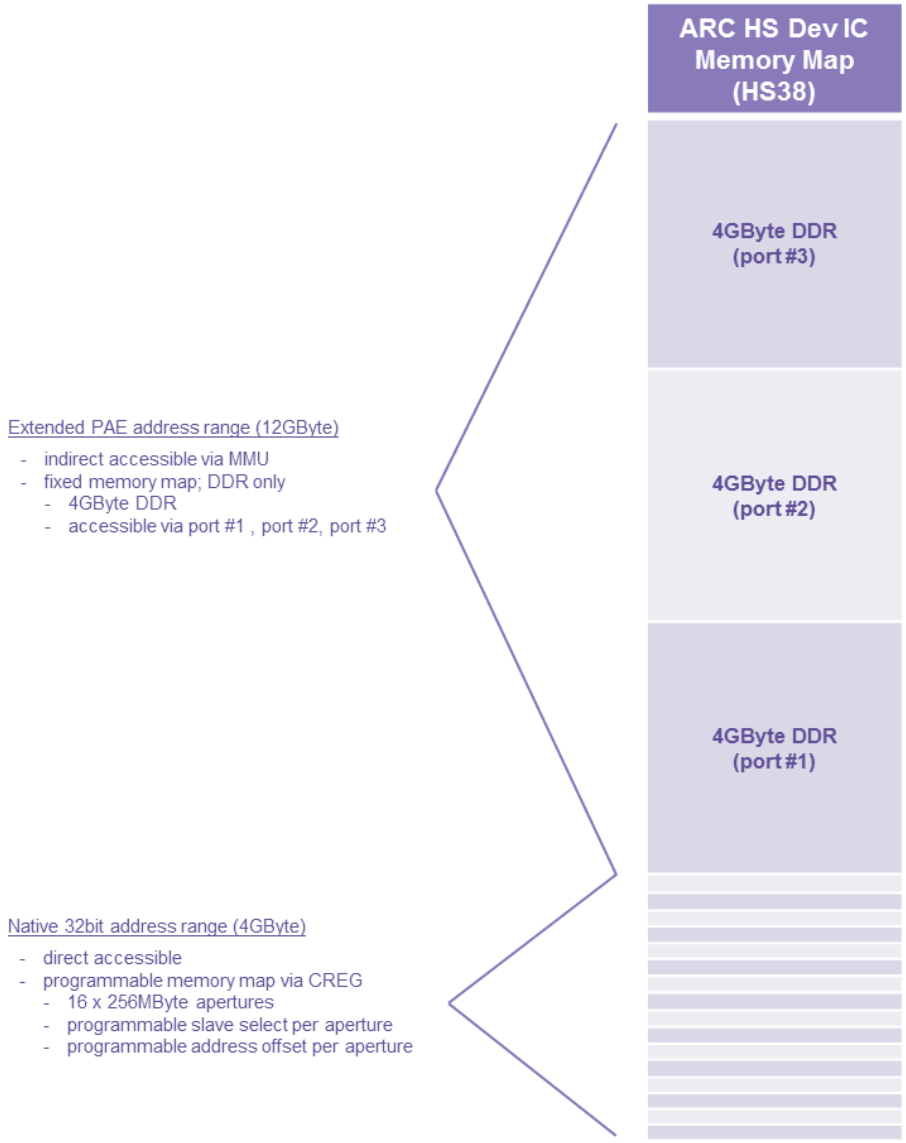
Table 16 APB Peripheral Address Map

Peripheral Name	Base	Size	Description
CGU	0x00000000	4kByte	Clock Generation Unit
CREG	0x00001000	4kByte	Control registers
GPIO	0x00003000	128Byte	General purpose I/O [6]
MCTL	0x00004000	2kByte	DDR memory controller [9]
PUB	0x00004800	2kByte	DDR PHY Utility Block
DEBUG-UART	0x00005000	256Byte	Debug UART [14]
WDT	0x00006000	256Byte	Watchdog Timer [18]
ETH-GMAC	0x00008000	8kByte	Ethernet MAC - 0x00008000 => control/status registers - 0x00009000 => DMA registers
SDIO	0x0000A000	1kByte	SDIO controller [13]
SPI0	0x00020000	256Byte	SPI
SPI1	0x00021000	256Byte	SPI
SPI2	0x00022000	256Byte	SPI
I2C0	0x00023000	256Byte	I2C
I2C1	0x00024000	256Byte	I2C
I2C2	0x00025000	256Byte	I2C
UART0	0x00026000	256Byte	UART
UART1	0x00027000	256Byte	UART
UART2	0x00028000	256Byte	UART
I2S_TX	0x00029000	256Byte	I2S TX
I2S_RX	0x0002A000	256Byte	I2S RX
PWM	0x0002B000	256Byte	PWM
USB-HOST	0x00040000	256kByte	USB-Host controller - 0x00040000 => EHCI control/status registers - 0x00060000 => OHCI control/status registers
DMAC	0x00080000	1kByte	DMAC [19]
GPU	0x00090000	256kByte	GPU

4.1.2 **PAE**

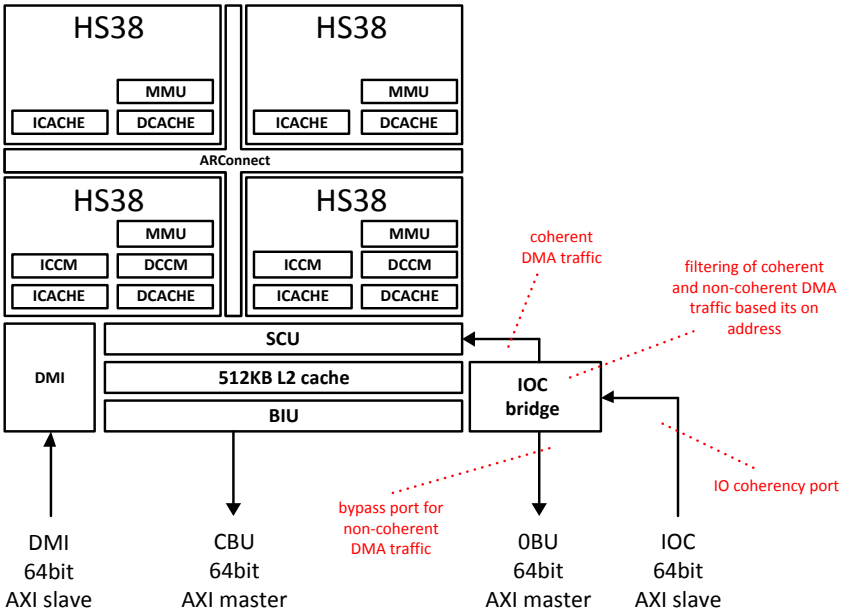
The ARC HS38 core supports Physical Address Extension (PAE). PAE extends the physical address range beyond the core's native 32-bit, 4GByte address range. In the ARC HS Development System SoC the PAE functionality is used to extend the physical address range from 4 to 16GByte. Figure 24 shows a high-level overview of the ARC HS Development System SoC memory map. The memory map for the lower 4GByte 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 12GByte is divided into three 4GByte regions that all map to the same 4GByte of DDR but through different DDR ports. Thus, it is still possible to map PAE traffic to the different DDR ports.

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



4.1.3 **I/O Coherency**

Figure 25 I/O Coherency Architecture



The ARC HS38x4 quad-core is equipped with 1MByte 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 [1] 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.

The ARC HS Development System SoC (or more specifically the ARC HS38x4) implements a hardware mechanism for maintaining I/O coherency. The I/O coherency architecture is illustrated in Figure 25. The ARC HS38x4 provides an additional AXI slave port (that is: 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:

[1] IO coherency: term used to indicate consistency of data that is shared between ARC cores and DMA clients

- Although permitted by the programmable memory mapping feature of the ARC HS Dev IC, (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 HS 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 HS 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 SW 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 section Register Descriptions.

Table 17 CREG Control Register Overview

Name	Address Offset	Access [1]	Description
Address Decoder Registers			
- ARC HS (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 HS (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
- 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
- SDIO			
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_DMAC0_SLV0	0x1120	RW	Address decoder slave select register
CREG_AXI_M_DMAC0_SLV1	0x1124	RW	Address decoder slave select register
CREG_AXI_M_DMAC0_OFFSET0	0x1128	RW	Address decoder slave offset register
CREG_AXI_M_DMAC0_OFFSET1	0x112C	RW	Address decoder slave offset register

CREG_AXI_M_DMAC0_UPDATE	0x1134	RW1C	Address decoder update
CREG_AXI_M_DMAC1_SLV0	0x1140	RW	Address decoder slave select register
CREG_AXI_M_DMAC1_SLV1	0x1144	RW	Address decoder slave select register
CREG_AXI_M_DMAC1_OFFSET0	0x1148	RW	Address decoder slave offset register
CREG_AXI_M_DMAC1_OFFSET1	0x114C	RW	Address decoder slave offset register
CREG_AXI_M_DMAC1_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
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 & 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 uBoot*

Bit	Name	Access	Value	Description
3:0	SLV_SEL0	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 (=> HS38x4 IOC)
			10	Slave 10 selected (=> HS38x4 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 HS (CORE)
m = 1 ARC HS (RTT)
m = 2 AXI Tunnel
m = 5 USB
m = 6 ETH
m = 7 SDIO
m = 8 GPU
m = 9/10 DMAC

[2] See Table 18 for reset values after uBoot

[3] when “ARC HS38x4 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 uBoot*

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 HS (CORE)	m = 6	ETH
m = 1	ARC HS (RTT)	m = 7	SDIO
m = 2	AXI Tunnel	m = 8	GPU
m = 5	USB	m = 9/10	DMAC

[2] See Table 18 for reset values after uBoot

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

*Legend: * reset value after uBoot*

Bit	Name	Access	Value	Description
3:0	OFFSET0	RW	[2] *	Address offset for address aperture[0]
			0	0 * 256MByte
			1	1 * 256MByte
			…	…
			15	15 * 256MByte
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 HS (CORE)	m = 6	ETH
m = 1	ARC HS (RTT)	m = 7	SDIO
m = 2	AXI Tunnel	m = 8	GPU

m = 5 USB m = 9/10 DMAC

[2] See Table 18 for reset values after uBoot

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

*Legend: * reset value after uBoot*

Bit	Name	Access	Value	Description
3:0	OFFSET8	RW	[2] *	Address offset for address aperture[8]
			0	0 * 256MByte
			1	1 * 256MByte
			…	…
			15	15 * 256MByte
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 HS (CORE) m = 6 ETH
m = 1 ARC HS (RTT) m = 7 SDIO
m = 2 AXI Tunnel m = 8 GPU
m = 5 USB m = 9/10 DMAC

[2] See Table 18 for reset values after uBoot

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 (i.e. *_SLV and *_OFFSET) are double-buffered. The newly programmed values will be only be forwarded to the address decoder after writing a '1' to this bit.

[1] m = 0 ARC HS (CORE) m = 6 ETH
m = 1 ARC HS (RTT) m = 7 SDIO
m = 2 AXI Tunnel m = 8 GPU
m = 5 USB m = 9/10 DMAC

Table 18 CREG Address Decoder Register Reset Values (after uBoot)

m	Master	AXI_M_m_SLV0	AXI_M_m_SLV1	AXI_M_m_OFFSET0	AXI_M_m_OFFSET0
0	HS38x4 (CORE)	0x1111_1111	0x6311_1111	0xFEDC_BA98	0x0E54_3210
1	HS38x4 (RTT)	0x7777_7777	0x7777_7777	0xFEDC_BA98	0x7654_3210
2	AXI Tunnel	0x8888_8888	0x8888_8888	0xFEDC_BA98	0x7654_3210
5	USB-HOST	0x7777_7777	0x7799_9999	0xFEDC_BA98	0x76DC_BA98
6	ETHERNET	0x7777_7777	0x7799_9999	0xFEDC_BA98	0x76DC_BA98
7	SDIO	0x7777_7777	0x7799_9999	0xFEDC_BA98	0x76DC_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

4.2.1.1.2 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

4.2.1.1.3 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	density=4gb	width=x8	ranks=1

[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 HS38x4_1
1	START_2	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS38x4_2
2	START_3	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS38x4_3
3	START_4	RW1C	0x0 *	Writing a 1 to this bit generates a cpu_start pulse for ARC HS38x4_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]	HS38x4_1
			0x1	HS38x4_2
			0x2	HS38x4_3
			0x3	HS38x4_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

[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

4.2.1.1.4 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 *	SPI-0
7:4	DMA_FC_SEL1	RW		DMA mux select for dma flow control interface[3:2] [1]
			0x3 *	I2C-0
11:8	DMA_FC_SEL2	RW		DMA mux select for dma flow control interface[5:4] [1]
			0x0	SPI-0
			0x1	SPI-1
			0x2	SPI-2
			0x3	I2C-0
			0x4	I2C-1
			0x5	I2C-2
			0x6 *	UART-0
			0x7	UART-1
			0x8	UART-2
			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 will result 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]

4.2.1.1.5 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

4.2.1.1.6 DDR Control and Status Registers

CREG_DDR_CTRL – DDR Control Register (Address Offset = 0x14A8)

*Legend: * reset value after uBoot*

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 uBoot*

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

4.2.1.1.7 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] 1st 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] 2nd 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] 3rd 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] 4th 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] 1st 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] 2nd 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] 3rd 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] 1st 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] 2nd 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] 3rd 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

4.2.1.1.8 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 *	hs38x4_clk divide-by-1
			0x1	hs38x4_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

4.2.1.1.9 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		ARCRTT 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

4.2.1.1.10 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 will reset the corresponding interrupt enable bit in the INT_ENABLE register to'0'. bit[3:0] => HS38x4 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 will reset 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 will set the corresponding interrupt status bit in the INT_STATUS register to 1.

4.2.1.1.11 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
4:0	DD	R	18	Version of the ARC HS Development IC - date : day
8:5	MM	R	5	- date : month
20:9	YYYY	R	2016	- date : year
26:21	TM	R	25	- time : minutes
31:27	TH	R	16	- 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 HS 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**" (**A**rc **D**evelopment)

4.2.2 Clock Registers

The clock registers and the associated clock circuitry (i.e. PLLs + clock dividers) inside the ARC HS 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 HS

- SYS PLL

  Used to generate all the other clocks in the system (e.g. 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

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

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 section Register Descriptions.

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
Clock Control & Status Registers			
- ARC PLL			
CGU_ARC_IDIV	0x0080	RW	Clock divider register for ARC HS clock
CGU_ARC_FMEAS	0x0084	RW	Clock measurement register for ARC HS 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 Tunnel clock
CGU_TUN_FMEAS	0x0384	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_OUT_RST_CTRL	0x0980	RW	Reset control register for IC reset output
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 has to program the desired PLL divider settings and the CGU autonomously cycle through the following steps:

1) Switch all integer dividers to **sysclk** (33MHz)
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, `Fout`, can be calculated as follows:

```
Fref      = Fin / NR
Fvco = Fout* NO
Fout      = Fin * 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:

Low-band				**High-band**			
10MHz	≤	Fref	≤ 50Mhz	25MHz	≤	Fref	≤ 50Mhz
800MHz	≤	Fvco	≤ 1600Mhz	1500MHz	≤	Fvco	≤ 3000Mhz
100MHz	≤	Fout	≤ 1600Mhz	188MHz	≤	Fout	≤ 3000Mhz

4.2.2.2 Frequency Measurement

The frequency measurement module measures the frequency of a clock relative to the 33MHz 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:

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

Few remarks:

- By default, both counters start counting from zero. However, to simply 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 would then show 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 1000MHz vs. 1kHz), 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 uBoot*

Bit	Name	Access	Value	Description
0	PD	RW	1*	Powerdown PLL (0=normal mode, 1=powerdown mode)
1	BYPASS	RW	1*	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)
				Fout = Fin * NF / (NR*NO) Fout = Fin * 24/(1*8) = 33.33 * 24/8 =100Mhz
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=500MHz
p = 1	SYS PLL	OD=2	R=1	F=47	BS=0	freq=400MHz
p = 2	DDR PLL	OD=2	R=1	F=39	BS=0	freq=333MHz
p = 3	TUN PLL	OD=3	R=0	F=17	BS=0	freq=150MHz

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

*Legend: * reset value after uBoot*

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

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

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

4.2.2.3.2 Clock Control and Status Registers

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

*Legend: * reset value after uBoot*

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 HS clock	N = 1
SYS PLL	p = 1	c = 0	APB clock	N = 2
		c = 1	AXI clock	N = 1
		c = 2	ETH core clock	N = 1
		c = 3	USB core clock	N = 1
		c = 4	SDIO core clock	N = 1
		c = 6	GPU core clock	N = 1
		c = 7	GPU dma clock	N = 1
		c = 8	GPU cfg clock	N = 2
		c = 9	DMAC core clock	N = 1
		c = 10	DMAC cfg core clock	N = 2
		c = 11	SDIO reference clock	N = 1
		c = 12	SPI reference clock	N = 12
		c = 13	I2C reference clock	N = 4
		c = 14	UART reference clock	N = 12
TUN PLL	p = 3	c = 0	Tunnel clock	N = 3
		c = 1	ROM clock	N = 1
		c = 2	PWM core clock	N = 2
I2S	p = 5	c = 0	I2S TX clock	N = 1
		c = 1	I2S RX clock	N = 1

CGU_p_FMEAS_c – Clock Measurement Register (Address offset = 0x0084 + p*0x100 + n*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*	Writing a 1 to the START bit starts a frequency measurement. The START bit resets to 0 when the frequency measurement has completed - measured frequency = (FCNT / RCNT) * f_{ref}

ARC PLL	p = 0	c = 0	ARC HS 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 = 6	GPU core clock
		c = 7	GPU dma clock
		c = 8	GPU cfg 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 core 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 – ARC 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	-	RW	0*	Reserved; must be written as 0
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

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 HS Development IC. 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 HS 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**" (**A**rc **D**evelopment)

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 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
			3	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
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	
7:6	FAULT_ENA_45	RW		pwm_fault[3]	pwm_fault[2]
			b00 *	ignore	ignore
			b01	ignore	react
			b10	react	ignore
			b11	react	react
			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 *next* 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 pwm_irq_new_nperiods interrupt from PWM timer 01
1	ENA_IRQ_NEW_NPERIODS_23	RW	0 *	Enable the pwm_irq_new_nperiods interrupt from PWM timer 23
2	ENA_IRQ_NEW_NPERIODS_45	RW	0 *	Enable the pwm_irq_new_nperiods interrupt from PWM timer 45
3	ENA_IRQ_NEW_NPERIODS_67	RW	0 *	Enable the pwm_irq_new_nperiods interrupt from PWM timer 67
4	ENA_IRQ_UPD_MISSED_01	RW	0 *	Enable the pwm_irq_upd_missed interrupt from PWM generator 01
5	ENA_IRQ_UPD_MISSED_23	RW	0 *	Enable the pwm_irq_upd_missed interrupt from PWM generator 23
6	ENA_IRQ_UPD_MISSED_45	RW	0 *	Enable the pwm_irq_upd_missed interrupt from PWM generator 45
7	ENA_IRQ_UPD_MISSED_67	RW	0 *	Enable the pwm_irq_upd_missed interrupt from PWM generator 67
8	ENA_IRQ_TRIG_01	RW	0 *	Enable the pwm_irq_trg interrupt for PWM generator 01
9	ENA_IRQ_TRIG_23	RW	0 *	Enable the pwm_irq_trg interrupt for PWM generator 23
10	ENA_IRQ_TRIG_45	RW	0 *	Enable the pwm_irq_trg interrupt for PWM generator 45
11	ENA_IRQ_TRIG_67	RW	0 *	Enable the pwm_irq_trg interrupt for PWM generator 67
12	ENA_IRQ_FAULT_0	RW	0 *	Enable the pwm_irq_fault interrupt on an active fault on pwm_fault_0_a.
13	ENA_IRQ_FAULT_1	RW	0 *	Enable the pwm_irq_fault interrupt on an active fault on pwm_fault_1_a.

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

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

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	CLR_IRQ_NEW_NPERIODS_01	RW1C	0 *	Clear the pwm_irq_new_nperiods interrupt from PWM timer 01
1	CLR_IRQ_NEW_NPERIODS_23	RW1C	0 *	Clear the pwm_irq_new_nperiods interrupt from PWM timer 23
2	CLR_IRQ_NEW_NPERIODS_45	RW1C	0 *	Clear the pwm_irq_new_nperiods interrupt from PWM timer 45
3	CLR_IRQ_NEW_NPERIODS_67	RW1C	0 *	Clear the pwm_irq_new_nperiods interrupt from PWM timer 67
4	CLR_IRQ_UPD_MISSED_01	RW1C	0 *	Clear the pwm_irq_upd_missed interrupt from PWM generator 01
5	CLR_IRQ_UPD_MISSED_23	RW1C	0 *	Clear the pwm_irq_upd_missed interrupt from PWM generator 23
6	CLR_IRQ_UPD_MISSED_45	RW1C	0 *	Clear the pwm_irq_upd_missed interrupt from PWM generator 45
7	CLR_IRQ_UPD_MISSED_67	RW1C	0 *	Clear the pwm_irq_upd_missed interrupt from PWM generator 67
8	CLR_IRQ_TRIG_01	RW1C	0 *	Clear the pwm_irq_trg interrupt for PWM generator 01
9	CLR_IRQ_TRIG_23	RW1C	0 *	Clear the pwm_irq_trg interrupt for PWM generator 23
10	CLR_IRQ_TRIG_45	RW1C	0 *	Clear the pwm_irq_trg interrupt for PWM generator 45
11	CLR_IRQ_TRIG_67	RW1C	0 *	Clear the pwm_irq_trg interrupt for PWM generator 67
12	CLR_IRQ_FAULT_0	RW1C	0 *	Clear the pwm_irq_fault interrupt on an active fault on pwm_fault_0_a.
13	CLR_IRQ_FAULT_1	RW1C	0 *	Clear the pwm_irq_fault interrupt on an active fault on pwm_fault_1_a.

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 pwm_trq_irq_01 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 pwm_trq_irq_23 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 pwm_trq_irq_45 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 pwm_trq_irq_67 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 pwm_clk cycles pwm_ch[0] (pwm_ch[1]) must be 'inactive' when according to the PWM Timer 01 and corresponding THRESHOLD_01 value pwm_ch[0] (pwm_ch[1]) 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_xy and TIMER_MAX register settings are valid for PWM timer xy. `pwm_irq_new_nperiods` and `pwm_irq_upd_missed` 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. pwm_irq_new_nperiods and pwm_irq_upd_missed 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. pwm_irq_new_nperiods and pwm_irq_upd_missed 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 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_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 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.

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 HS 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

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

A.1 HapsTrak 3 Extension Connector Pins

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_resetn_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]

J3-Pin Number	J3-Signal	J4-Pin Number	J4-Signal
B9	he_tunnel_rx[16]	B9	he_tunnel_tx[21]
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]
C10	he_tunnel_rx[17]	C10	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]

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

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-eeprom

[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