



# **DesignWare ARC EM Software Development Platform User Guide**

---

Version 5795-004 January 2019

## Copyright Notice and Proprietary Information Notice

© 2019 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### Destination Control Statement

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

### Disclaimer

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

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/company/legal/trademarks-brands.html>. All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

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

Synopsys, Inc.  
[www.synopsys.com](http://www.synopsys.com)

# Contents

List of Figures .....	5
List of Tables.....	6
Accessing SolvNet .....	7
Contacting the Synopsys Technical Support Center .....	7
1 Introduction .....	8
1.1 Package Content.....	8
1.2 Getting Started .....	8
1.2.1 Installing Device Drivers.....	8
1.2.2 Checking Default Board Settings .....	9
1.2.3 Installing and Configuring PuTTY.....	9
1.2.4 Programming the FPGA Device .....	12
1.2.5 Starting U-Boot.....	13
1.3 Location of Components On ARC EM SDP .....	14
1.4 Software Packages .....	15
2 Hardware Description .....	16
2.1 Overview of the ARC EM SDP Board.....	16
2.2 Overview of the ARC EM Software Development Platform .....	18
2.3 Clocks and Resets .....	19
2.3.1 Clocks .....	19
2.3.2 Reset.....	23
2.4 Interrupts .....	24
2.5 DMA .....	24
2.6 Debug and Trace .....	24
2.6.1 Debug .....	25
2.6.2 ARC Real-Time Trace.....	26
2.7 Configuration and Boot Modes.....	27
2.7.1 Boot Switches .....	28
2.7.2 VCCIO Voltage Selection Switches.....	30
2.7.3 Miscellaneous Switches .....	31

2.7.4 On-Board LEDs.....	31
2.8 Memories .....	32
2.9 USB Interface.....	32
2.10SD Card Interface.....	33
2.11Audio Interface .....	33
2.12On-Board I2C Control Bus .....	34
2.13ADC.....	34
2.14Redpine® WiFi, Bluetooth, Zigbee Interface .....	35
2.15External Bus Interface.....	35
2.15.1 Timing Diagrams .....	36
2.16Extension Interfaces.....	37
2.16.1 Digilent Pmod™ .....	37
2.16.2 Mikrobus .....	42
2.16.3 Arduino.....	43
2.16.4 Generic Header.....	46
3 Programmer's Reference .....	49
3.1 Memory Map .....	49
3.1.1 APB Peripheral Address Map .....	50
3.1.2 Auxiliary Based Peripherals .....	52
3.2 Software Interfaces .....	53
3.2.1 Clock Registers.....	53
3.2.2 Control Registers .....	71
3.2.3 EBI Registers .....	80
Glossary and References .....	86
Glossary .....	86
References .....	87

## List of Figures

---

Figure 1 Identification of COM Port .....	10
Figure 2 PuTTY Configuration .....	11
Figure 3 ARC EM SDP Programming of the Bitstream .....	12
Figure 4 Configuration LEDs .....	13
Figure 5 Default Boot with ARC EM Initialization.....	13
Figure 6 ARC EM SDP Components - Top View.....	14
Figure 7 ARC EM SDP Components - Bottom View .....	15
Figure 8 ARC EM SDP Block Diagram.....	16
Figure 9 ARC EM Software Development Platform Top-Level Diagram .....	18
Figure 10 ARC EM SDP Clock Architecture .....	19
Figure 11 Graphical Overview of the Clock Domains .....	22
Figure 12 Reset Architecture .....	23
Figure 13 ARC EM SDP Debug and Trace Headers.....	25
Figure 14 Pinout of 10-Pin Debug Header.....	25
Figure 15 10-Pin to 20-Pin JTAG Adapter .....	26
Figure 16 Nexus Mictor 38 Interface.....	27
Figure 17 ARC EM SDP Configuration - Boot Switches and Buttons.....	27
Figure 18 Normal Read (Without External Wait) .....	36
Figure 19 Normal Write (Without External Wait).....	36
Figure 20 ARC EM SDP Peripheral Extension Interfaces .....	37
Figure 21 Pinout Diagram of the Pmod_A, Pmod_B and Pmod_C Connectors .....	38
Figure 22 MikroBus Headers .....	42
Figure 23 Arduino Shield Interface .....	44
Figure 24 Generic Pin Header Interface .....	46

## List of Tables

---

Table 1	Overview of ARC EM SDP Clock Components .....	20
Table 2	Audio Clock Divider Settings .....	34
Table 3	ARC EM SDP On-board I2C Slave Addresses .....	34
Table 4	ADC Channel Usage .....	35
Table 5	Pin Description of the Pmod_A Connector .....	39
Table 6	Pin Description of the Pmod_B Connector .....	40
Table 7	Pin Description of the Pmod_C Connector .....	41
Table 8	Pin Description of the MikroBUS Connectors .....	43
Table 9	Pin Description of the Arduino Shield Interface .....	45
Table 10	ADC Input .....	46
Table 11	Pin Description of the Generic Pin Interface .....	47
Table 12	ARC EM SDP Memory Map .....	49
Table 13	APB Peripheral Address Map .....	50
Table 14	Auxiliary Based IO Peripherals .....	52
Table 15	CGU Clock Register Overview .....	54
Table 16	CREG Control Register Overview .....	72

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](mailto: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>.

## 1.1 Package Content

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

- DesignWare ARC EM Software Development Platform (ARC EM SDP).
- A 100-240 V AC power adapter (including adapters for the US, UK, and EU outlets).
- USB cable.



### Warning

The DesignWare ARC EM Development Platform contains static-sensitive devices.

---

## 1.2 Getting Started

This section includes instructions for the following tasks:

1. [Installing device drivers](#)
2. [Checking default board settings](#)
3. [Installing and configuring PuTTY](#)
4. [Starting U-Boot](#)

---

### 1.2.1 Installing Device Drivers

Before the USB-JTAG and the USB-UART interfaces are used, you must install the required drivers on the computer where you intend to run the MetaWare debugger or another serial debug console (such as PuTTY or other hyper-terminals). For more information on the MetaWare debugger, see *MetaWare Debugger User's Guide for ARC*.

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.



---

## 1.2.2 Checking Default Board Settings

Check if the boot switches and jumpers are set to their default positions. For an overview of the configuration options and the default settings, see [Configuration and Boot Modes](#) on page 27.

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

Connect the power supply included in the product package to the ARC EM SDP.



The ARC EM SDP must be powered by an external power adapter.

---

---

## 1.2.3 Installing and Configuring PuTTY

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

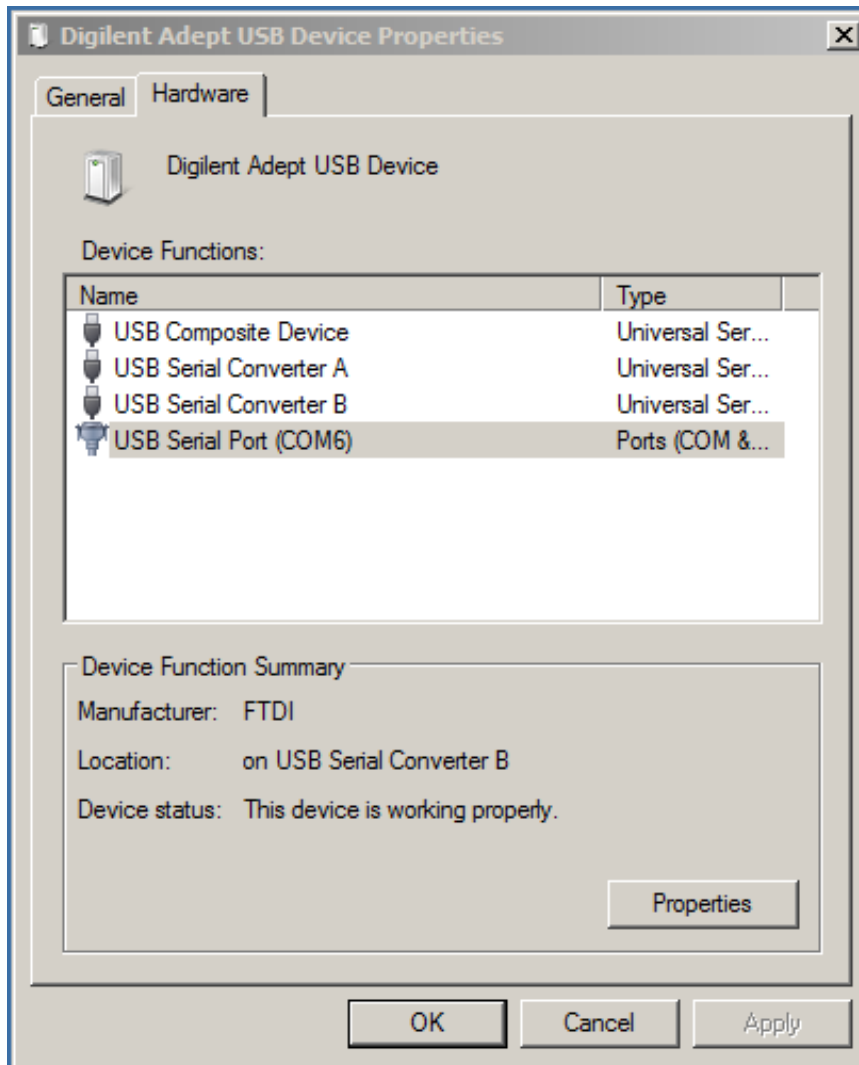
1. Download `putty.exe` from <http://www.putty.org>
2. Open the Windows **Control Panel**.
3. In the category **Hardware and Sound**, click **View devices and printers**, and **Digilent Adept USB Device**.

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

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

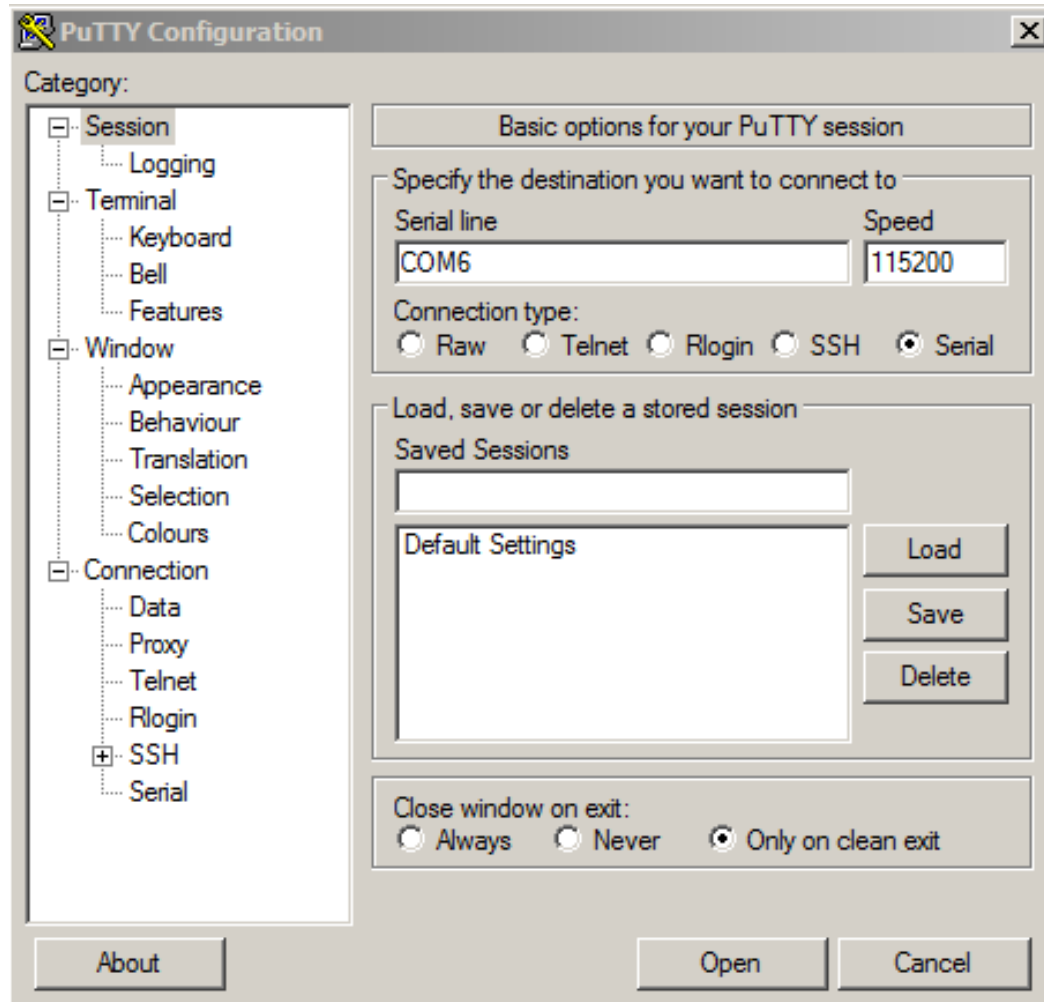
The example in Figure 1 uses the COM6 port:

Figure 1 Identification of COM Port



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

Figure 2 PuTTY Configuration



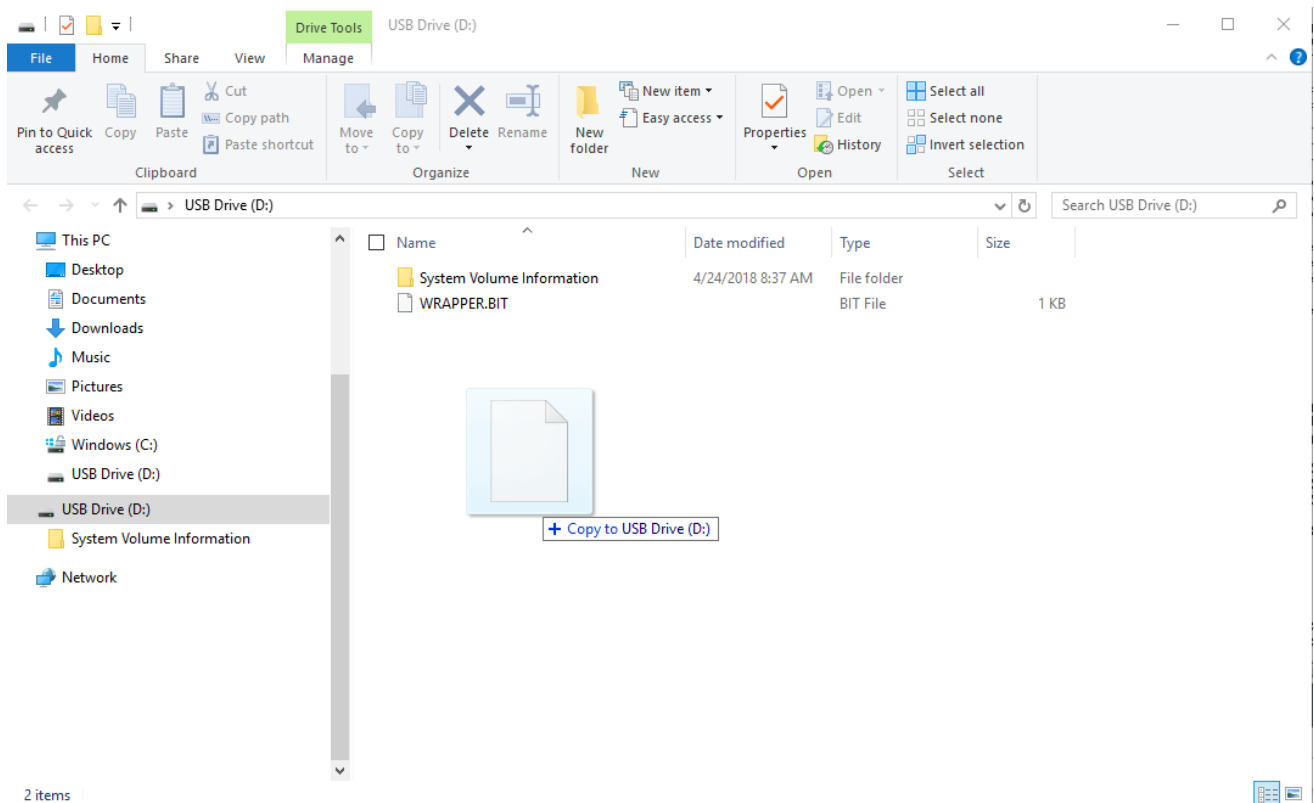
8. Click **Open** to launch the PuTTY terminal.

## 1.2.4 Programming the FPGA Device

The FPGA on the ARC EM SDP board is configured by storing an FPGA bitstream in the configuration memory of the ARC EM SDP. By default, no image is present in the configuration memory and the configuration LED blinks red (see [Figure 4](#)).

The configuration memory can be accessed by connecting the USB cable to the USB data port of the ARC EM SDP and the PC. When the ARC EM SDP is powered (power LED lights green), the ARC EM SDP presents itself as a mass-storage device on the PC (see [Figure 3](#)).

Figure 3 ARC EM SDP Programming of the Bitstream



Drag-and-drop an FPGA bitstream to the USB drive and the bitstream is copied from the PC to the configuration memory and programmed into the FPGA.

While the configuration memory is being written, the status LED blinks green. When a bitstream is stored inside the configuration memory, the bitstream is automatically programmed into the FPGA. During this process the configuration LED blinks red.

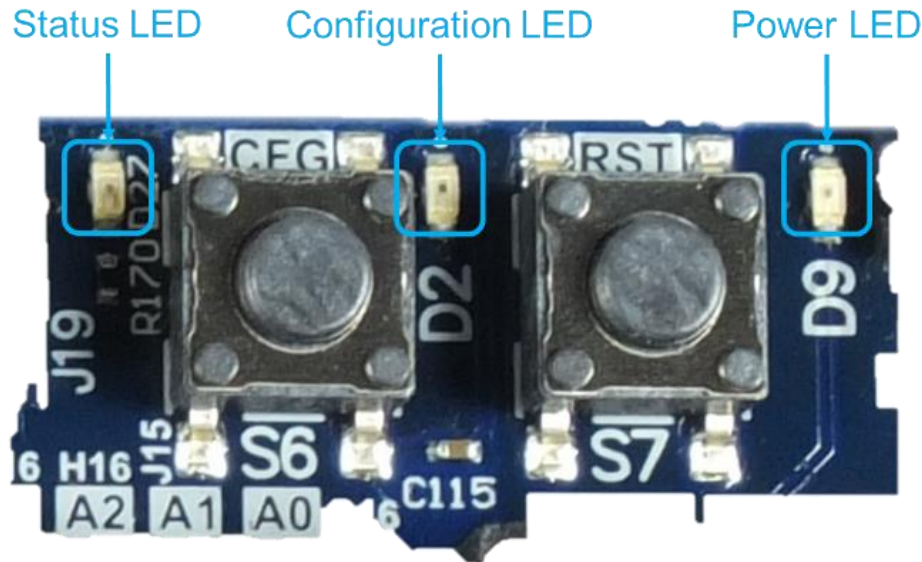
When the configuration is completed, the status LED on the ARC EM SDP stays green. For a full description on the LEDs, see [On-Board LEDs](#) on page 31.



**Note** Updating the content of the configuration memory may take few minutes to complete.

The bitstream remains present in the non-volatile configuration memory until it is overwritten by another bitstream or is explicitly removed from the memory. This means that on a next power cycle, or when pressing the CFG key, the FPGA is automatically programmed with the bitstream present in the configuration memory.

Figure 4 Configuration LEDs



## 1.2.5 Starting U-Boot

Press the start button on the ARC EM SDP for the ARC EM core to start executing the bootloader. After you press the start button, you see text similar as shown in [Figure 5](#). This figure shows a log from default booting, followed by U-Boot commands to initialize the ARC EM SDP board and start an application called `app.bin` that resides on the SD card. The entry point for this example application is `0x1000_0400`.

Figure 5 Default Boot with ARC EM Initialization

```
U-Boot 2018.11 (Nov 27 2018 - 08:23:36 +0100)

CPU:   ARC EM11D v5.0 at 40 MHz
Subsys:ARC Data Fusion IP Subsystem
Model: snps,emsdp
Board: ARC EM Software Development Platform v1.0
DRAM:  16 MiB
MMC:   Synopsys Mobile storage: 0
Loading Environment from FAT... OK
In:    serial0@f0004000
Out:   serial0@f0004000
Err:   serial0@f0004000
emsdp# fatload mmc 0 0x10000000 app.bin
45728 bytes read in 28 ms (1.6 MiB/s)
emsdp# go 0x10000400
## Starting application at 0x10000400 ...
```

For more information on the available U-Boot commands, you can execute the help function in U-Boot.

## 1.3 Location of Components On ARC EM SDP

Figure 6 and Figure 7 show the placement of various components on the ARC EM SDP.

Figure 6 ARC EM SDP Components - Top View

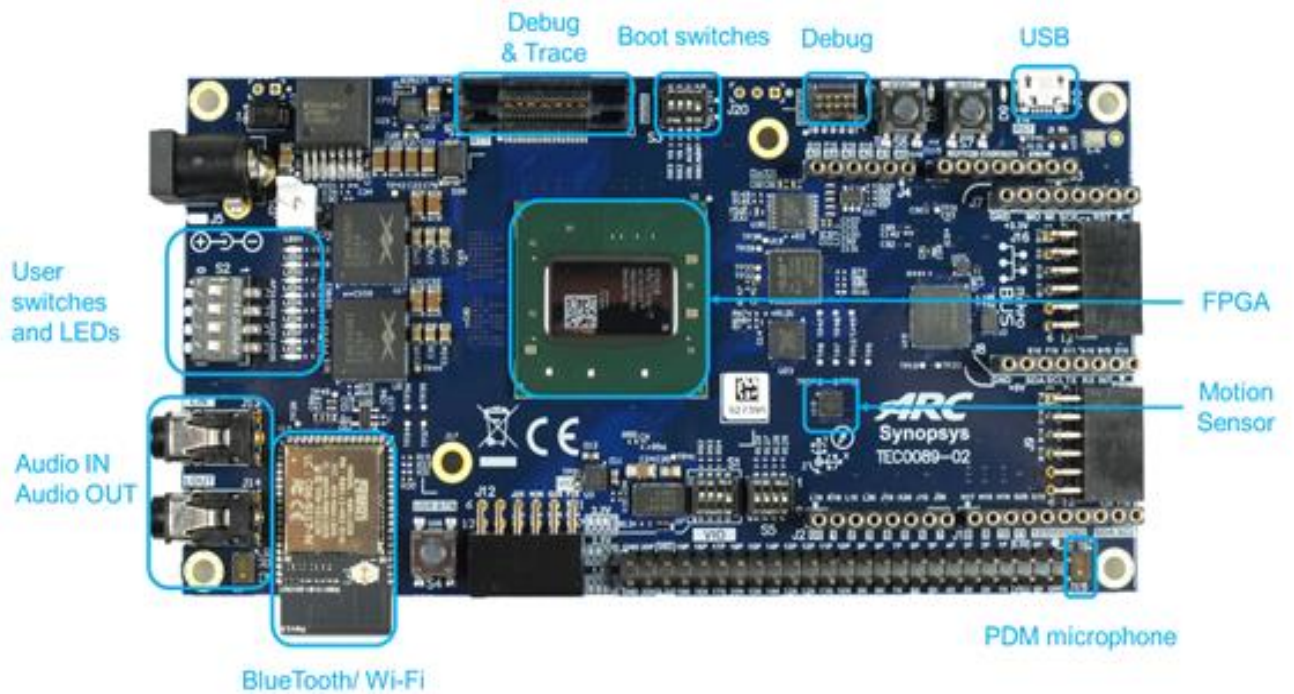
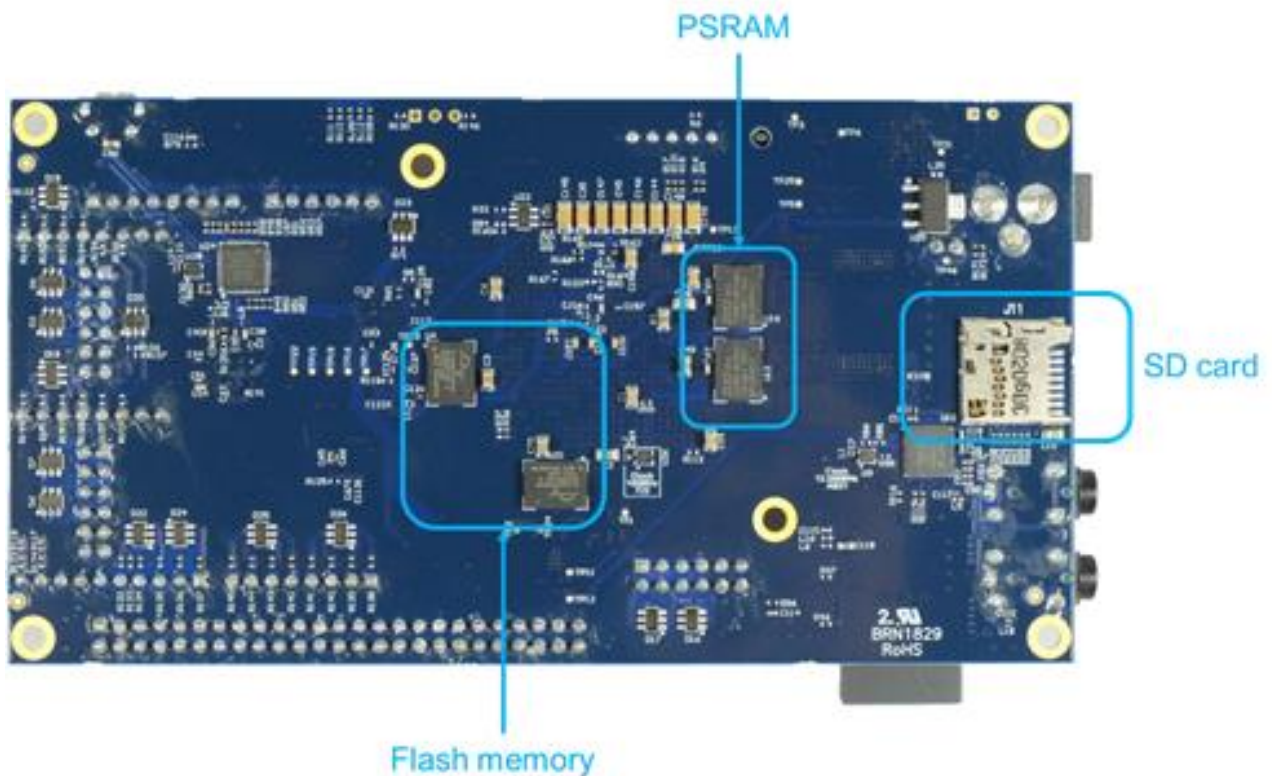




Figure 7 ARC EM SDP Components - Bottom View



## 1.4 Software Packages

See the <http://embarc.org/> portal for information on the available software packages for the ARC EM SDP.

For more details on the embARC software releases for ARC EM SDP, see:

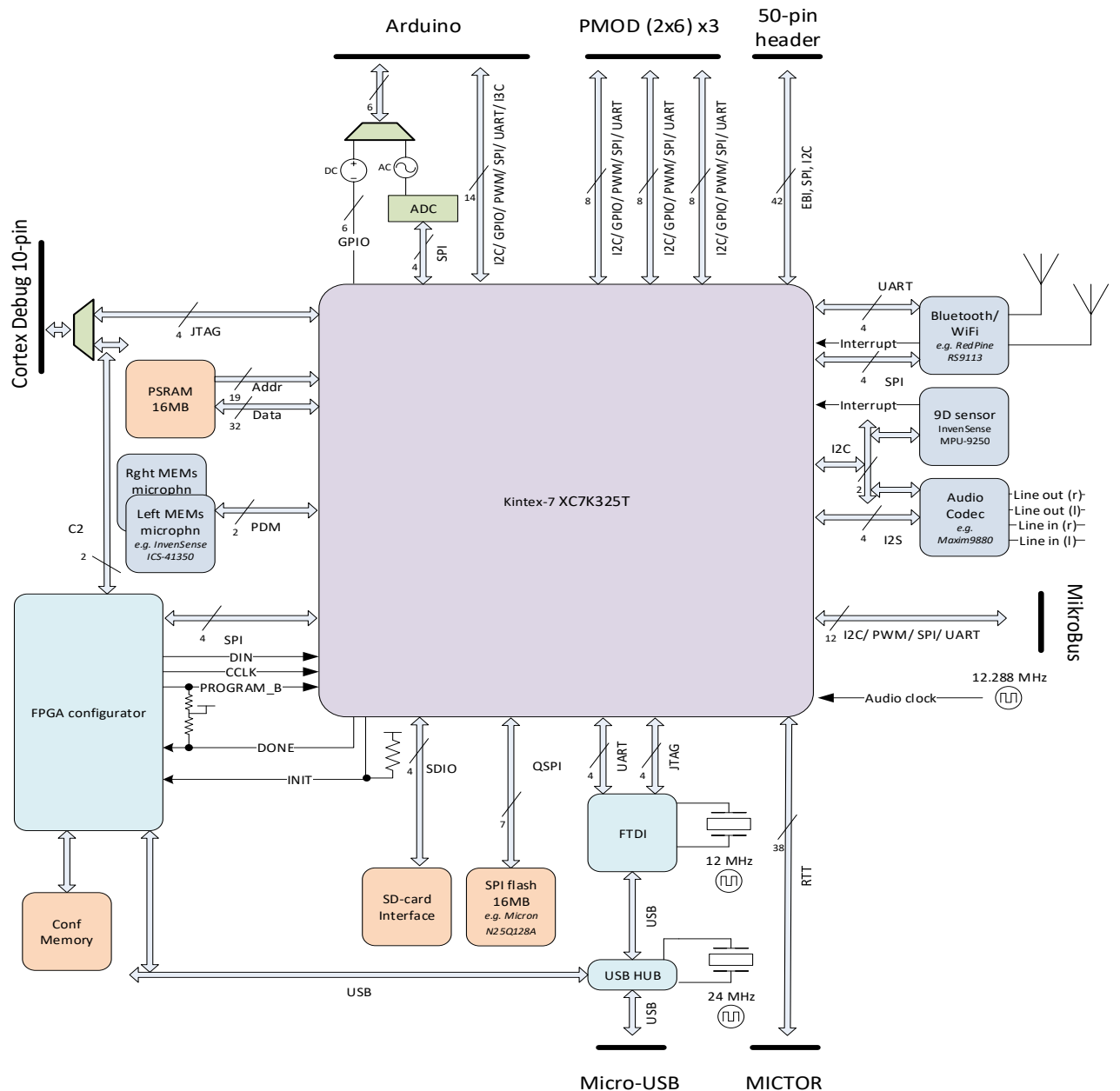
- [https://github.com/foss-for-synopsys-dwc-arc-processors/embarc\\_osp/releases](https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_osp/releases)

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>

### 2.1 Overview of the ARC EM SDP Board

Figure 8 ARC EM SDP Block Diagram





The ARC EM SDP contains the following components:

- FPGA
  - Xilinx® Kintex®-7 XC7325T-2
- Memory
  - PSRAM (16 MB)
  - SPI Flash (16 MB quad-mode XiP)
  - Configuration memory (16 MB SPI flash)
- Interfaces
  - Audio line in/out
  - USB Data port (JTAG/UART/access to configuration memory)
  - Micro- SD Card
  - Wi-Fi®/Bluetooth®/Zigbee® module
  - ADC (eight channels)
  - Motion Sensor
  - Digital MEMs microphone (2x)
  - RTT Nexus, JTAG
- Extensions
  - Arduino® Interface headers (UNO Rev3 compatible)
  - MikroBUS™ headers
  - Digilent Pmod™ Interfaces (3x)
  - Generic Pin Header (50 pins)



- I<sup>2</sup>C interfaces
- SPI interfaces
- PWM interfaces
- JTAG interface

## 2.3 Clocks and Resets

### 2.3.1 Clocks

The ARC EM SDP uses a single 100 MHz reference clock from which all the system clocks are generated. The clock generation is centralized in the Clock and Reset Unit (CRU). The CRU implements several PLLs and integer dividers that allow for accurate fine-tuning of the system clocks to the desired frequency. Figure 10 shows the ARC EM SDP clock architecture.

Figure 10 ARC EM SDP Clock Architecture

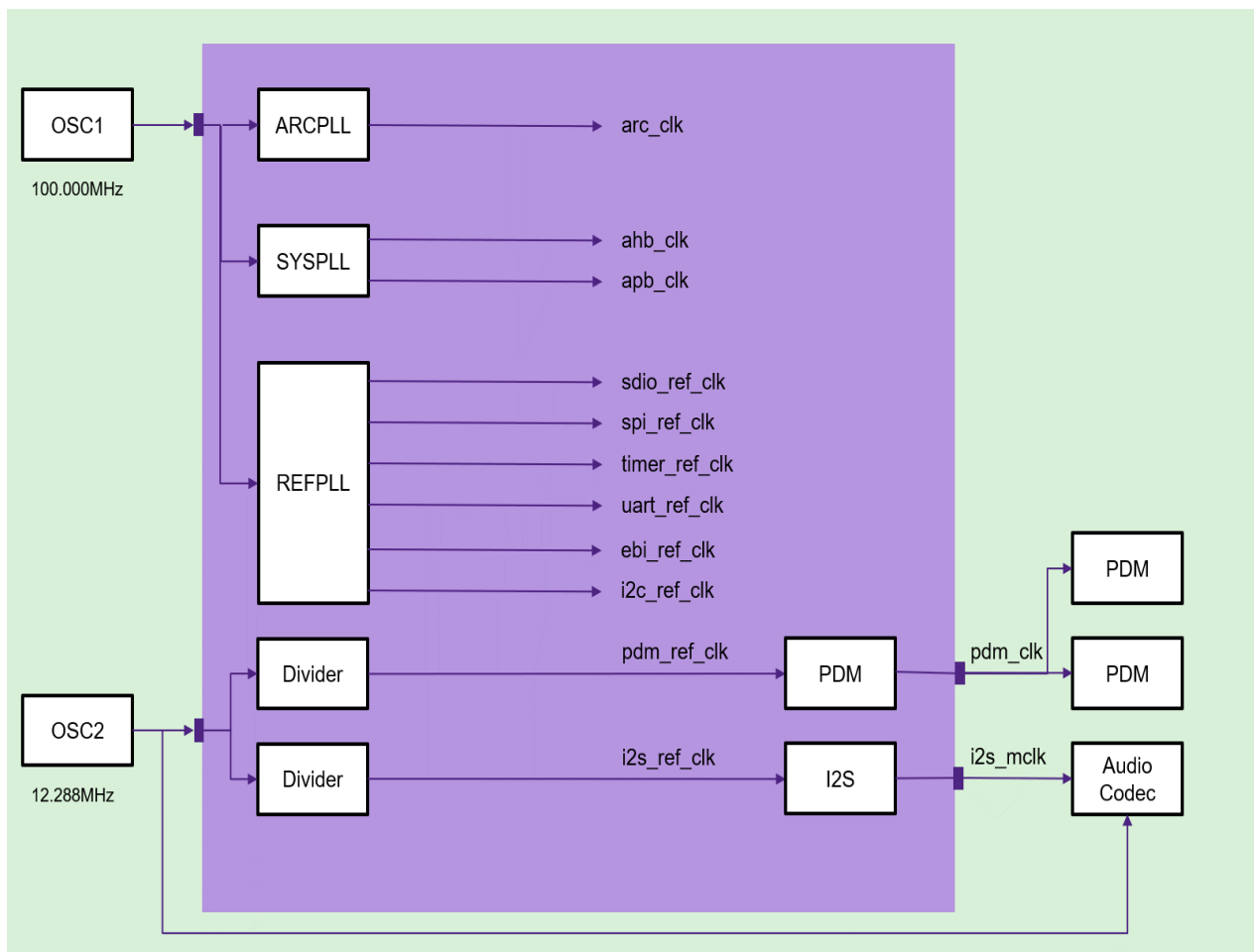


Table 1 lists the summary of all the clocks and their sources.

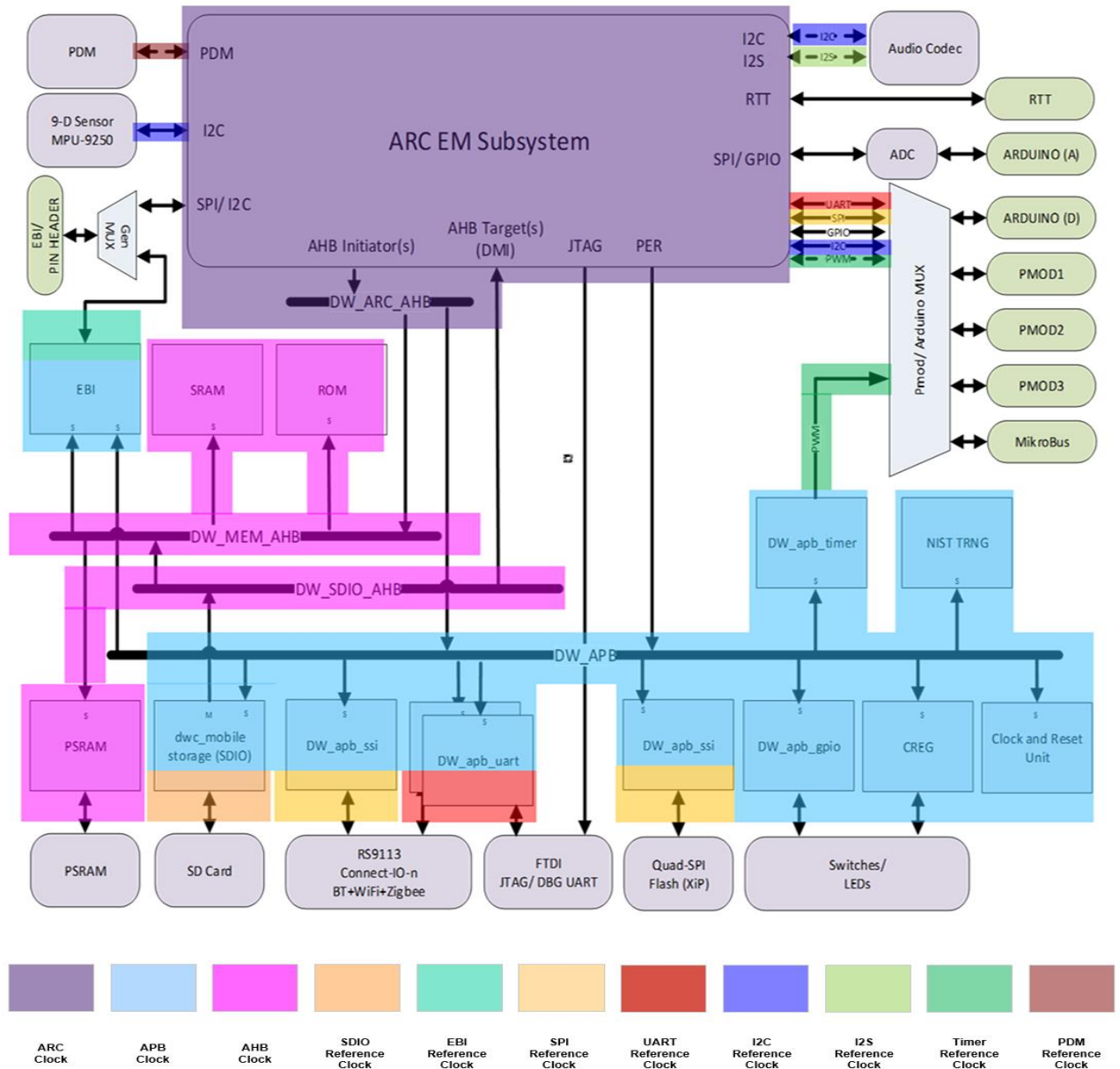
Table 1 Overview of ARC EM SDP Clock Components

PLL	Base Clock	Clock Name	Description	Default Frequency
<b>Core Clock Domain</b>				
0	0	arc_clk	ARC core clock driving EM subsystem	50 MHz (see table footnote)
<b>AMBA Clock Domain</b>				
1	0	ahb_clk	AHB clock for bus infrastructure	100 MHz
	0	apb_clk	APB clock for bus infrastructure	50 MHz
<b>Reference Clock Domain</b>				
2	0	sdio_ref_clk	SDIO reference clock	100 MHz
	1	spi_ref_clk	SPI reference clock	20 MHz
	2	timer_ref_clk	Timer reference clock	100 MHz
	3	uart_ref_clk	UART reference clock	100 MHz
	4	ebi_ref_clk	EBI reference clock	100 MHz
	5	i2c_ref_clk	I2C reference clock	100 MHz

PLL	Base Clock	Clock Name	Description	Default Frequency
<b>Audio Clock Domain</b>				
3	0	i2s_ref_clk	I2S reference clock	12.228 MHz
	1	pdm_ref_clk	PDM reference clock	12.288 MHz
<p>Note: The default ARC clock frequency may be different based on the configuration. For the actual frequency values, see the build configuration description in the subsystem reference document that is delivered as part of the platform package or the board header file.</p>				

[Figure 11](#) shows a graphical representation of the different clock domains.

Figure 11 Graphical Overview of the Clock Domains



### 2.3.2 Reset

Figure 12 shows the top-level reset architecture of the ARC EM SDP.

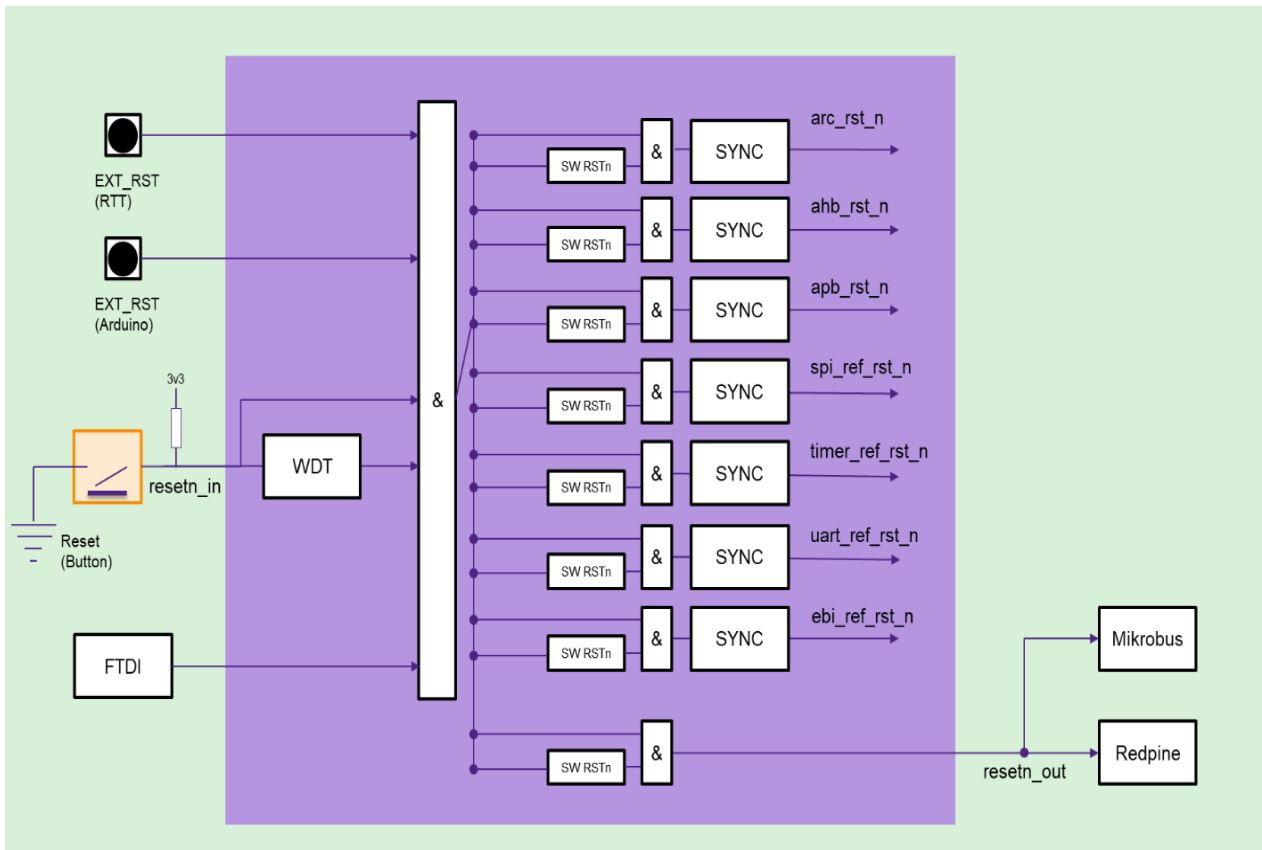
When the ARC EM SDP is in reset, the `resetn_out` is asserted. This output pin resets all components on the board that have a reset input pin (for example, the Redpine communication module). The other board components have a power-on reset that assure the correct sequence of de-asserting the board resets.

The ARC EM SDP has an 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 'ACBUS6' GPIO pin. This allows for the board to be reset from a PC. For more information on FTDI Chip, see [FT2232H Datasheet](#).

After the ARC EM SDP is out of reset, `resetn_out` is asserted resetting all the ARC EM SDP components. The `resetn_out` is also routed to the Mikrobus headers.

Figure 12 Reset Architecture



---

## 2.4 Interrupts

The interrupt architecture of the ARC EM SDP depends on the design with which the FPGA is programmed. Each interrupt in the design is assigned to a dedicated interrupt pin on the ARC EM processor.

The assignment of the interrupt pins can be derived from the software header files delivered as part of the product. For more information on the assignment of the interrupt pins, see the subsystem reference document that is delivered as part of the platform package.

---

## 2.5 DMA

The DMA architecture of the ARC EM SDP depends on the design with which the FPGA is programmed. Each DMA channel interface in the design is assigned to a dedicated DMA interface on the ARC EM processors DMA controller.

The assignment of the DMA interface pins can be derived from the software header files delivered as part of the product. For more information on the assignment of the DMA interface pins, see the subsystem reference document that is delivered as part of the platform package.

---

## 2.6 Debug and Trace

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

- Debug:
  - USB cable connected to the on-board 2-channel FTDI chip:
    - Compatible with MetaWare debugger and GDB.
  - Ashling Opella-XD, Lauterbach Trace-32, Digilent HS1 and HS2 support through:
    - A 10-pin 5 mm debug header and an adapter cable.
- Trace:
  - Both on-chip through system memory or off-chip through the Nexus interface into external host trace off-load is available.



**Note** The availability of the trace functionality depends on the build configuration. For more details on the build configuration, see the subsystem reference document that is delivered as part of the platform package.

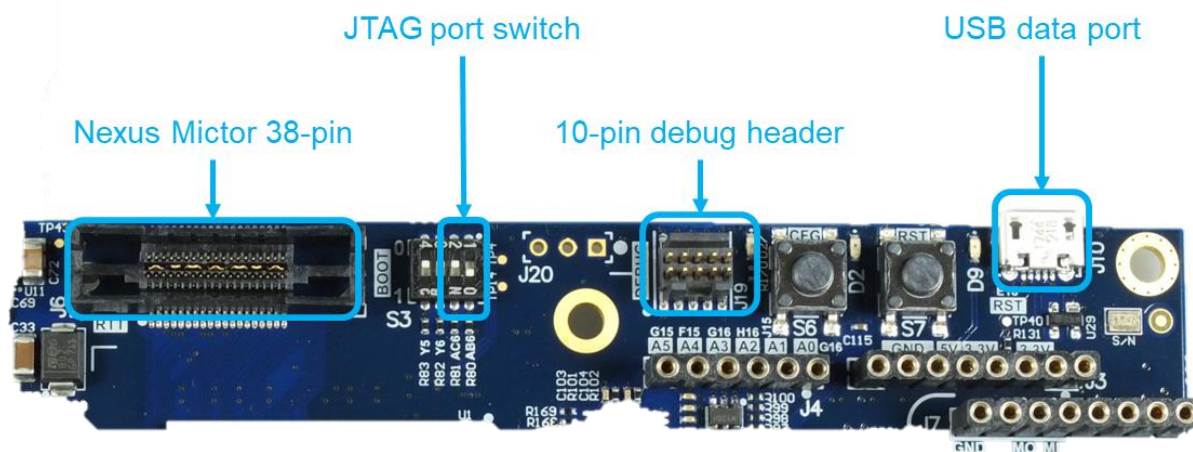
---

- Ashling Ultra-XD and Lauterbach Trace-32 are supported.



Figure 13 shows the various debug and trace headers.

Figure 13 ARC EM SDP Debug and Trace Headers



## 2.6.1 Debug

The ARC EM core provides debug access through an IEEE 1149.1 JTAG port.

### 2.6.1.1 USB Dataport

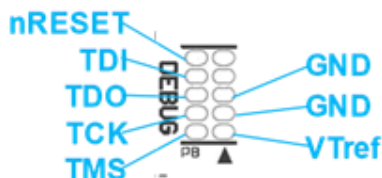
The USB Dataport can be connected to your PC using the USB cable included in the product package. A USB converter from FTDI (FT2232HL) converts one channel to a serial communication protocol (UART). The other channel is converted to JTAG.

The JTAG channel offered over this Dataport is compatible with the MetaWare debugger. The serial communication channel is used as a console and can be monitored using a standard hyper terminal application for example PuTTY. For more information, see [Getting Started](#) on page 8.

### 2.6.1.2 10-Pin Header Pinout

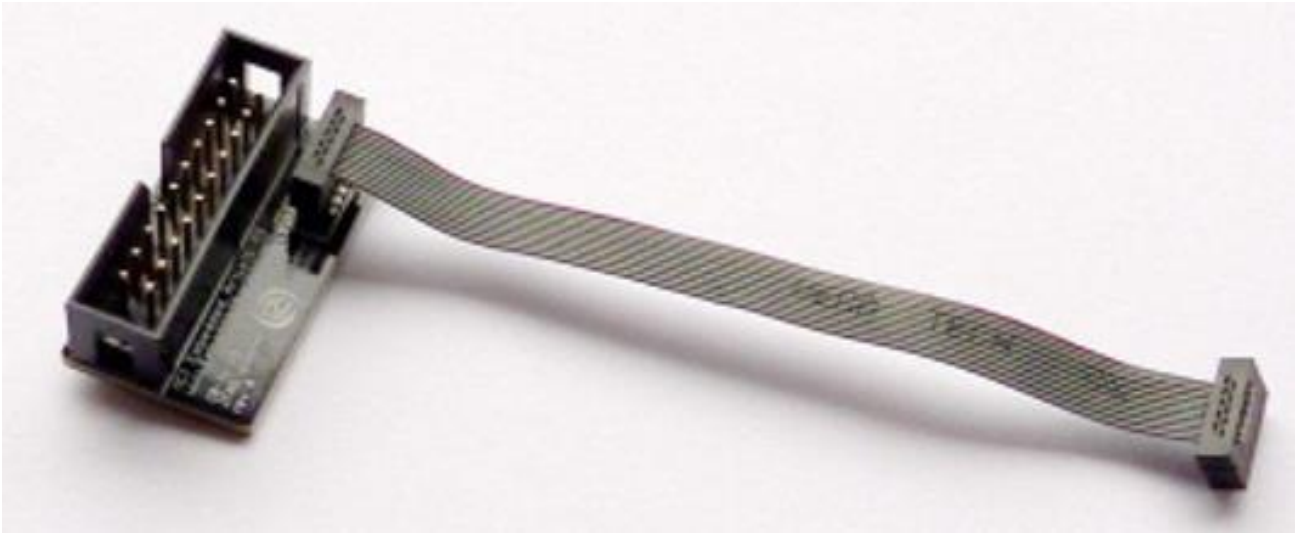
The ARC EM SDP includes a 10-pin debug header that can be used for a JTAG connection towards the core.

Figure 14 Pinout of 10-Pin Debug Header



This header can be used to connect a standard 20-pin Ashling or Lauterbach probe header using an adapter. Figure 15 shows the 10-pin to 20-pin JTAG adapter.

Figure 15 10-Pin to 20-Pin JTAG Adapter



The adapter can be purchased from many distributors such as Mouser, Embedded Artists, and Digikey.

---

## 2.6.2 ARC Real-Time Trace

The ARC EM SDP implements Real-Time-Trace capability (ARC RTT). When ARC RTT is present in the configuration, it allows configurable/ programmable monitoring of:

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

**Note**

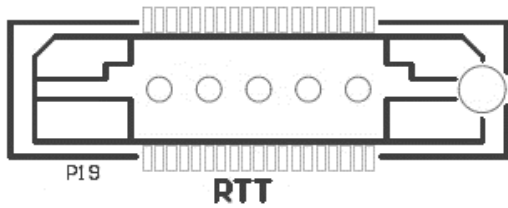
The trace functionality is only present when ARC RTT has been configured for the subsystem.

---

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

The Nexus 5001 interface is a 16-bit high-speed interface and for the ARC EM SDP, it supports a trace clock up to 50 MHz.

Figure 16 Nexus Mictor 38 Interface



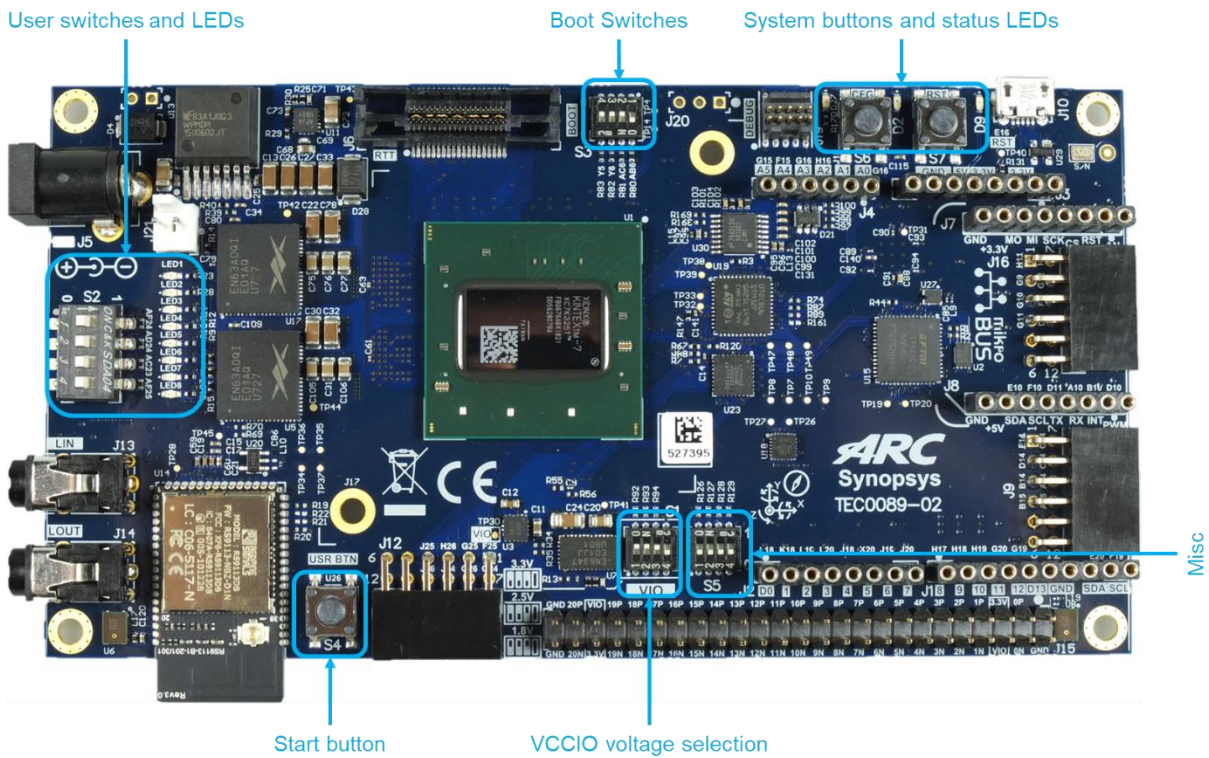
The ARC RTT interface is supported with the Ashling Ultra-XD and Lauterbach Trace-32 products.

## 2.7 Configuration and Boot Modes

Figure 17 shows the configuration switches, jumpers, and LEDs available on the ARC EM SDP. In addition to jumpers and standard reset and start buttons there are also switches to control the board boot mode and debug mode.

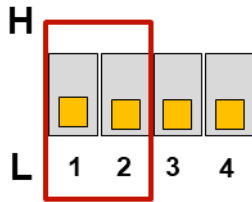
For further details on the jumpers and switches used in ARC EM SDP configuration, see the following sections.

Figure 17 ARC EM SDP Configuration - Boot Switches and Buttons

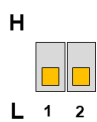


## 2.7.1 Boot Switches

### 2.7.1.1 JTAG Port Switches

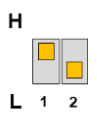


These switches are used to control the way the JTAG signals are routed. This can be either through the USB dataport, the 10-pin debug header or through the Nexus RTT interface.



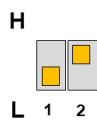
USB mode.

JTAG signals are routed through the FTDI USB port. The Digilent plugin of the MetaWare debugger can be used to access the core. No external probe is required. This is the default setting.



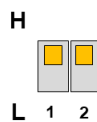
JTAG mode.

JTAG signals are routed over the 10-pin debug header. An external probe can be used to access to core.



RTT mode.

JTAG signals are routed over the 38-pin Mictor header. An external probe can be used to access to core.



Reserved.

### 2.7.1.2 Boot Start Mode Switch

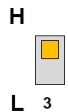


This switch controls manual or automatic booting of the ARC EM SDP.



Manual mode.

The ARC EM SDP only starts booting after the START button is pushed. This is the default setting.



Automatic mode.

The ARC EM SDP automatically starts booting after Reset.

### 2.7.1.3 Boot Image Location Switch



The following table provides an overview of the boot image location switch settings.



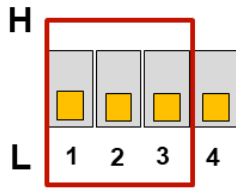
Ensure that these switches are set as depicted on the left (default position). This setting ensures that the pre-bootloader starts executing the bootloader that is stored in ROM.



Reserved for future use.

## 2.7.2 VCCIO Voltage Selection Switches

### 2.7.2.1 VCCIO Voltage Selection



The following table provides an overview of the reference voltage selection for the generic pin header.

	<p>1.8V. VIO pins of the generic pin header output - 1.8 volts.</p>
	<p>2.5V. VIO pins of the generic pin header output - 2.5 volts.</p>
	<p>3.3V. VIO pins of the generic pin header output - 3.3 volts.</p>

### 2.7.2.2 Factory Update



The following table provides an overview of the factory update mode.

	<p>Normal mode. The JTAG port is connected to the program interface of the FPGA or the JTAG chain of the ARC core depending on the board status and the JTAG overwrite switch. This is the default setting.</p>
	<p>Factory mode. This mode allows the firmware for the FPGA configurator to be updated.</p>

## 2.7.3 Miscellaneous Switches

### 2.7.3.1 USB JTAG Overwrite



This switch controls the muxing of the JTAG chain.

**H** Normal mode.



While the status LED stays green, the USB JTAG is connected to the debug interface of the ARC core. When the status LED indicates that no design is present, the USB JTAG is connected to the program interface of the FPGA. This is the default setting.

**H** Program mode.



Independent of the status LED, the USB JTAG is connected to the program interface of the FPGA.

## 2.7.4 On-Board LEDs

The ARC EM SDP includes the following LEDs:

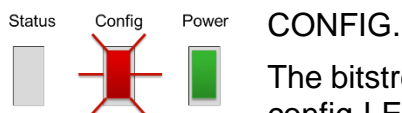
- Eight user LEDs
- Three status LEDs

The status LEDs indicate the status of the board. The following table lists an overview of the status LEDs.



**ON.**

Power is supplied to the board.






**CONFIG.**




The bitstream configuration is being programmed in the FPGA while the config LED is blinking. When the config LED remains blinking, there is no (valid) bitstream present inside the configuration memory.



---

Status	Config	Power	
			<b>READY.</b> The bitstream located inside the configuration memory has been successfully loaded into the FPGA.

---

Status	Config	Power	
			<b>UPDATE.</b> When the status LED blinks green, the content of the configuration memory is updated. Upon completion, the new content is automatically programmed in the FPGA.

---

The user LEDs can be controlled through the DesignWare GPIO. For more information, see [DesignWare DW\\_apb\\_gpio Databook](#).

---

## 2.8 Memories

The ARC EM SDP features the following memories:

- 16 MB PSRAM memory
- 16 MB user available SPI flash
  - This SPI flash supports Quad-mode SPI and execute-in-place
- 16 MB dedicated SPI flash storage for FPGA bitstream

---

## 2.9 USB Interface

The ARC EM SDP offers a single USB 2.0 host port, which is both used to access the FPGAs configuration memory and as a DEBUG/ UART port as described in [USB Dataport](#) on page 25.

When connected using the USB cable to a PC, the ARC EM SDP presents itself as a mass-storage device. This allows an FPGA configuration bitstream to be dragged and dropped into the configuration memory. The FPGA bitstream is automatically loaded into the FPGA device upon power-on reset, or when the configuration button is pressed.



## 2.10 SD Card Interface

The ARC EM SDP features a micro SD-card interface supporting the following micro SD cards:

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

After U-Boot, the default settings operate the SD card in the SDR25 speed mode.

## 2.11 Audio Interface

The ARC EM SDP features stereo audio jacks for `Line Out` and `Line In`. The stereo audio input and output signals are converted using a MAX9880A stereo codec from Maxim Integrated, which provides the interface between the stereo I<sup>2</sup>S ports of the ARC EM SDP and drives the `Line Out` and `Line In` interfaces. The I<sup>2</sup>S is controlled by the `io_i2s_tx` and `io_i2s_rx` auxiliary based peripheral controllers in the subsystem. For more information, see [Dual I2S Stereo Audio Codec MAX9880A](#).

**Note**

The audio codec is controlled by the I<sup>2</sup>S controllers inside the subsystem, and this functionality is therefore only available when I<sup>2</sup>S auxiliary based controller is present in the build configuration of the subsystem.

The I<sup>2</sup>S ports operate in master mode, which means that the I<sup>2</sup>S IPs inside the ARC EM SDP initialize and drive the I<sup>2</sup>S word select and a serial clock signal. The I<sup>2</sup>S serial clocks are generated by integer dividers that run at a fixed audio reference clock frequency of 12.288 MHz.

In addition to the stereo audio jacks, the ARC EM SDP also features two PDM MEMs microphones (left and right channel), which are controlled by the `io_pdm_rx` auxiliary based PDM peripheral controller when configured in the subsystem. The PDM clock is also generated by integer dividers that run on the audio reference clock of 12.288 MHz.

**Note**

The PDM microphones are controlled by the PDM controller inside the subsystem, and this functionality is therefore only available when the PDM auxiliary based controller is present in the build configuration of the subsystem.

The ARC EM SDP supports the following sampling frequencies: 16 kHz, 32 kHz, 48 kHz, 96 kHz, and 192 kHz. [Table 2](#) shows the divider settings for 16-bit stereo audio.

Table 2 Audio Clock Divider Settings

Audio Sample Rate				
16 kHz	32 kHz	48 kHz	96 kHz	192 kHz
24	12	8	4	2

## 2.12 On-Board I2C Control Bus

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

- Audio codec
- 9D motion sensor

Table 3 lists an overview of the I<sup>2</sup>C bus slave addresses. The on-board I<sup>2</sup>C bus is controlled by the `io_i2c_mst0` controller in the subsystem.



### Note

The on-board I<sup>2</sup>C bus is controlled by the I<sup>2</sup>C controller inside the subsystem, and this functionality is therefore only available when the I<sup>2</sup>C auxiliary based controller is present in the build configuration of the subsystem.

Table 3 ARC EM SDP On-board I2C Slave Addresses

Device Name	Description	7-Bit I <sup>2</sup> C Address
MPU 9250	9D motion sensor	1101000 = 0x68
MAX 9880A	Audio codec	0010000 = 0x10

## 2.13 ADC

The ARC EM SDP board includes the 8-input, 8-bit ADC088S022 from Texas Instruments. The conversion rate ranges from 500 kSPS to 1 MSPS. The analog input range is 0 to 5 volts. The analog input values are read using the `io_spi_mst2` auxiliary peripheral. For more information on 8-input 8-bit ADC088S022, see [ADC088S022 Datasheet](#).



### Note

The ADC is controlled by the SPI controller inside the subsystem, and this functionality is therefore only available when the SPI auxiliary based controller is present in the build configuration of the subsystem.

[Table 4](#) lists the various ADC channels and their usage.

Table 4 ADC Channel Usage

ADC IN	Usage
0	mikroBUS AN
1	Arduino AD0
2	Arduino AD1
3	Arduino AD2
4	Arduino AD3
5	Arduino AD4
6	Arduino AD5
7	Not Used

## 2.14 Redpine® WiFi, Bluetooth, Zigbee Interface

The ARC EM SDP board includes the Redpine Signals RS9113 module for communication using Wi-Fi, Bluetooth, and Zigbee. For more information, see [Redpine Signals RS9113-WiseConnect-API-Guide-v1.7.1](#).

The SPI and UART host interfaces for running the embedded software stack are connected to dedicated controllers (see [APB Peripheral Address Map](#) on page 50). By default, the RS9113 is set in SPI host interface mode.

## 2.15 External Bus Interface

An external bus interface (EBI) controller provides access to external MMIO peripherals through an SRAM-like interface with separate address and data bus plus the most commonly used control signals (that is chip select, read/write strobes and optional signal for wait state insertion). In the ARC EM SDP the main purpose of the EBI is to provide access to external peripherals through the generic pin header.

Access to external peripherals is initiated using an AHB slave data port. The EBI controller translates the AHB read/write requests into a sequence of memory read/writes. All critical memory timing parameters are programmable for compatibility with different speed grades and/or different vendors. The EBI controller provides an AHB slave configuration port for access to the timing and control and status registers.



**Note** The EBI controller FSM operates on the EBI reference clock (`ebi_ref_clk`). All the memory timing parameters, such as read cycle wait time specified in the `EBI_CS_WCR1` register, are counted in `ebi_ref_clk` cycles.

### 2.15.1 Timing Diagrams

Figure 18 Normal Read (Without External Wait)

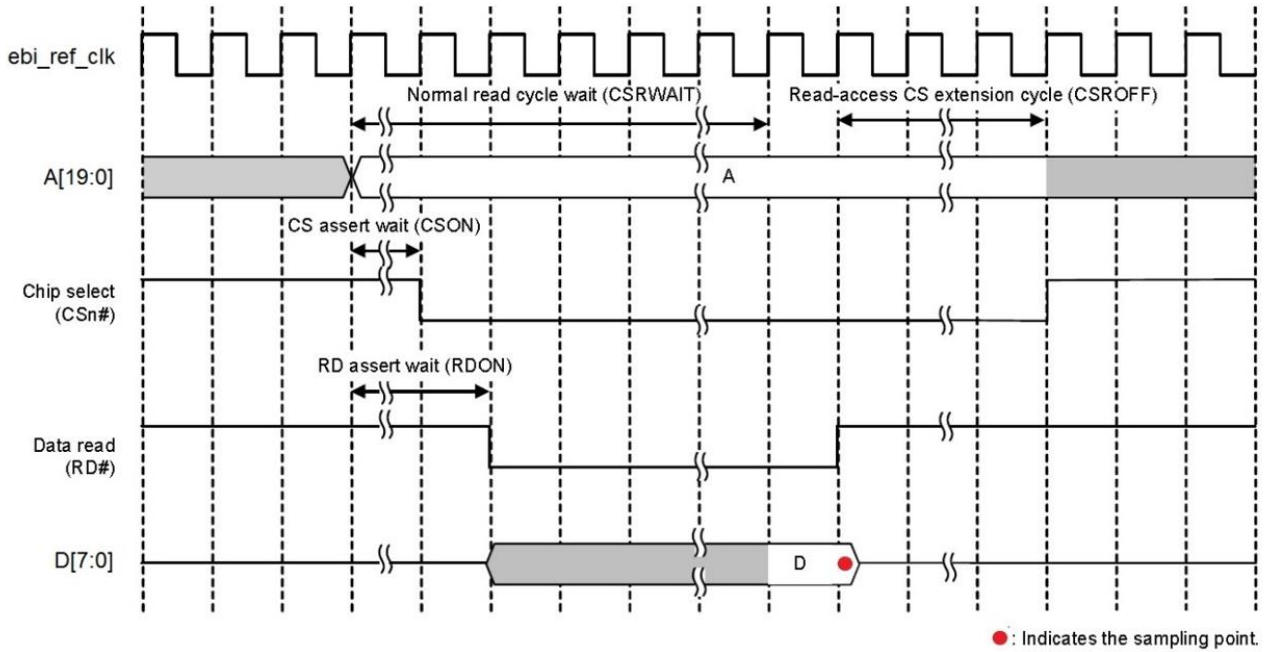
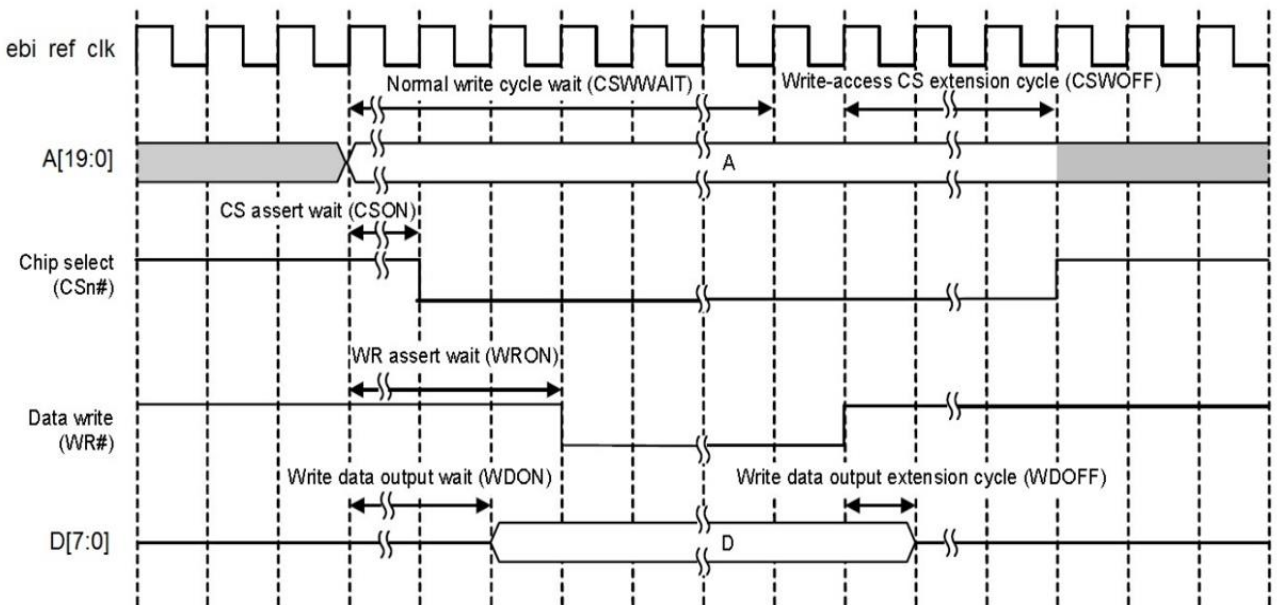


Figure 19 Normal Write (Without External Wait)

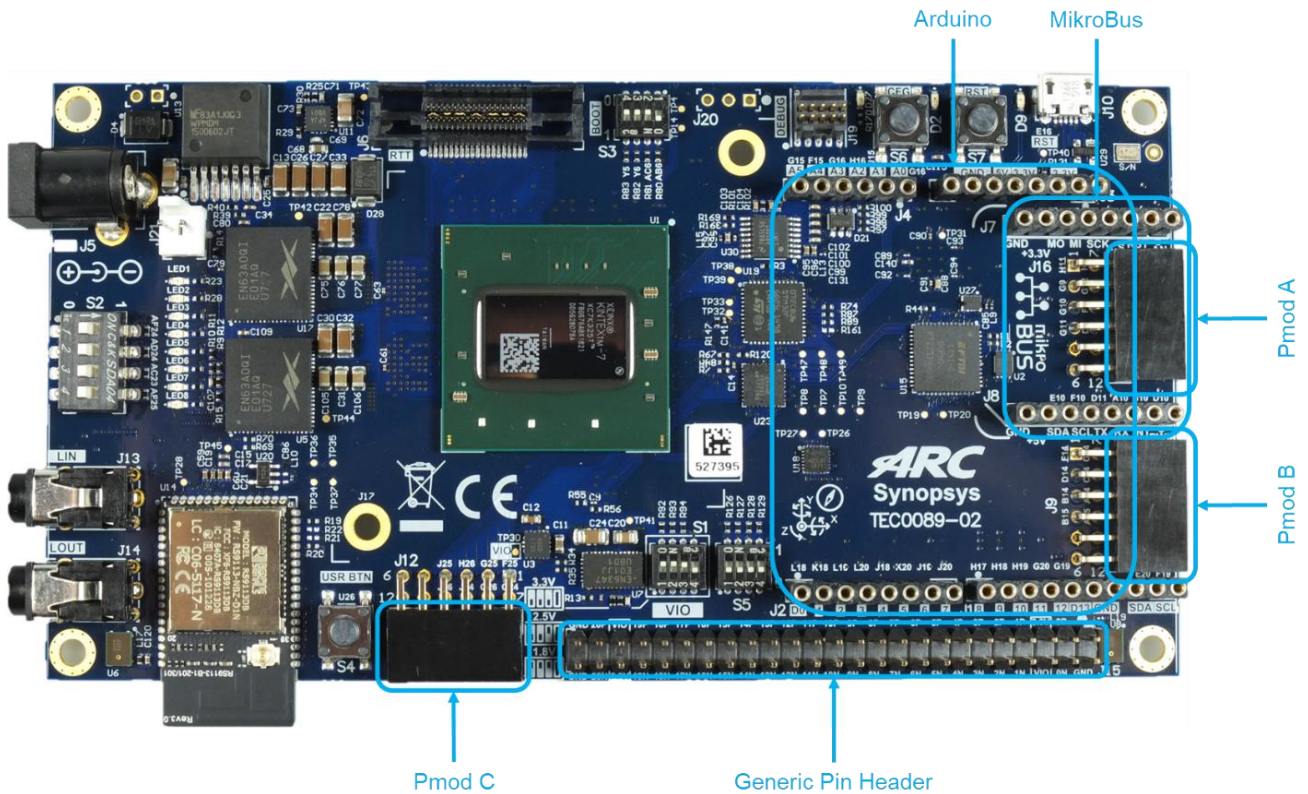


## 2.16 Extension Interfaces

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

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

Figure 20 ARC EM SDP Peripheral Extension Interfaces



Furthermore, the ARC EM SDP features a single 50-pin generic pin header (42 IO pins, four supply pins, and four ground pins).

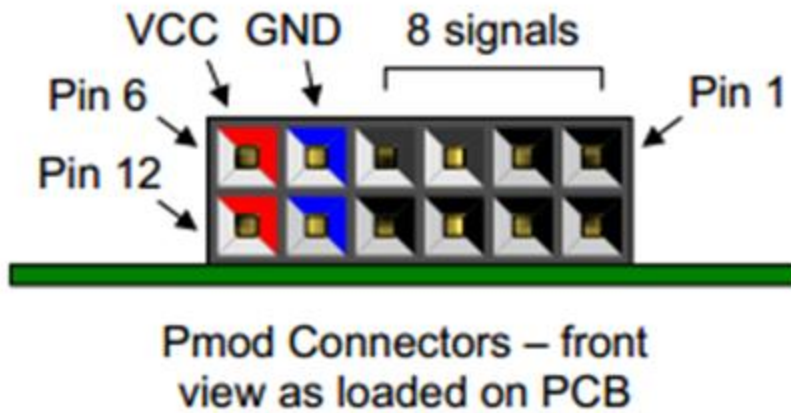
### 2.16.1 Digilent Pmod™

The ARC EM SDP features three 12-pin Pmod connectors: Pmod\_A, Pmod\_B, and Pmod\_C.

The functionality of the Pmod connectors is programmable and includes GPIO, UART, SPI, I<sup>2</sup>C, and PWM. Multiplexing is controlled by software using the PMOD\_MUX\_CTRL register (see [MUX Registers](#) on page 74MUX ). After a reset, all ports are configured as GPIO inputs.

Figure 21 shows the location of the pins on the Pmod connectors. Detailed pin descriptions depending on the pin multiplexer settings are provided in the subsequent sections.

Figure 21 Pinout Diagram of the Pmod\_A, Pmod\_B and Pmod\_C Connectors



#### Note

The Pmod is controlled by the controllers inside the subsystem, and this functionality is therefore only available when the auxiliary based controllers are present in the build configuration of the subsystem.

When the auxiliary based PWM IO peripherals are not available in the build configuration of the subsystem, the PWM functionality of `io_pwm0` is implemented using `dw_timer0` and the functionality of `io_pwm1` is implemented using `dw_timer1`.



### 2.16.1.1 Pmod\_A Connector

Table 5 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 5 Pin Description of the Pmod\_A Connector

MUX	Pin	GPIO	I2C	SPI	UART_1	UART_2	PWM_1	PWM_2
CFG0	A1	io_gpio1[0]	io_gpio1[0]	io_spi_mst1_cs[0]	io_uart1_cts	io_uart1_cts	io_gpio1[0]	io_gpio1[0]
	A2	io_gpio1[1]	io_gpio1[1]	io_spi_mst1_mosi	io_uart1_rts	io_uart1_txd	io_pwm1_ch[0]	io_pwm1_ch[0]
	A3	io_gpio1[2]	io_i2c_mst2_scl	io_spi_mst1_miso	io_uart1_rxd	io_uart1_rxd	io_gpio1[2]	io_gpio1[2]
	A4	io_gpio1[3]	io_i2c_mst2_sda	io_spi_mst1_clk	io_uart1_txd	io_uart1_rts	io_gpio1[2]	io_pwm1_ch[1]
	A5	GND	GND	GND	GND	GND	GND	GND
	A6	3V3	3V3	3V3	3V3	3V3	3V3	3V3
CFG1	A7	io_gpio1[4]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	A8	io_gpio1[5]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	A9	io_gpio1[6]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	A10	io_gpio1[7]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	A11	GND	GND	GND	GND	GND	GND	GND
	A12	3V3	3V3	3V3o	3V3	3V3	3V3	3V3

### 2.16.1.2 Pmod\_B Connector

Table 6 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 6 Pin Description of the Pmod\_B Connector

MUX	Pin	GPIO	I2C	SPI	UART_1	UART_2	PWM_1	PWM_2
CFG0	B1	io_gpio1[8]	io_gpio1[8]	io_spi_mst1_cs[1]	io_uart2_cts	io_uart2_cts	io_gpio1[8]	io_gpio1[8]
	B2	io_gpio1[9]	io_gpio1[9]	io_spi_mst1_mosi	io_uart2_rts	io_uart2_txd	io_pwm1_ch[2]	io_pwm1_ch[2]
	B3	io_gpio1[10]	io_i2c_mst2_scl	io_spi_mst1_miso	io_uart2_rxd	io_uart2_rxd	io_gpio1[10]	io_gpio1[10]
	B4	io_gpio1[11]	io_i2c_mst2_sda	io_spi_mst1_clk	io_uart2_txd	io_uart2_rts	io_gpio1[11]	io_pwm1_ch[3]
	B5	GND	GND	GND	GND	GND	GND	GND
	B6	3V3	3V3	3V3	3V3	3V3	3V3	3V3
CFG1	B7	io_gpio1[12]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	B8	io_gpio1[13]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	B9	io_gpio1[14]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	B10	io_gpio1[15]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	B11	GND	GND	GND	GND	GND	GND	GND
	B12	3V3	3V3	3V3	3V3	3V3	3V3	3V3



### 2.16.1.3 Pmod\_C Connector

Table 7 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 7 Pin Description of the Pmod\_C Connector

MUX	Pin	GPIO	I2C	SPI	UART_1	UART_2	PWM_1	PWM_2
CFG0	C1	io_gpio1[16]	io_gpio1[16]	io_spi_mst1_cs[2]	io_uart3_cts	io_uart3_cts	io_gpio1[16]	io_gpio1[16]
	C2	io_gpio1[17]	io_gpio1[17]	io_spi_mst1_mosi	io_uart3_rts	io_uart3_txd	io_pwm1_ch[4]	io_pwm1_ch[4]
	C3	io_gpio1[18]	io_i2c_mst2_scl	io_spi_mst1_miso	io_uart3_rxd	io_uart3_rxd	io_gpio1[18]	io_gpio1[18]
	C4	io_gpio1[19]	io_i2c_mst2_sda	io_spi_mst1_clk	io_uart3_txd	io_uart3_rts	io_gpio1[19]	io_pwm1_ch[5]
	C5	GND	GND	GND	GND	GND	GND	GND
	C6	3V3	3V3	3V3	3V3	3V3	3V3	3V3
CFG1	C7	io_gpio1[20]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	C8	io_gpio1[21]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	C9	io_gpio1[22]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	C10	io_gpio1[23]	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
	C11	GND	GND	GND	GND	GND	GND	GND
	C12	3V3	3V3	3V3	3V3	3V3	3V3	3V3

### 2.16.2 Mikrobus

The ARC EM SDP features a set of MikroBUS headers. [Figure 22](#) shows the relevant function assignments, fully compatible with the MikroBUS standard. For more information, see the [MikroBUS standard specification](#).

The MikroBUS headers enable the addition of Click boards. Click boards are developed by the company MikroElektronica ([www.mikroe.com](http://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](http://www.mikroe.com/click) for a full list.

Multiplexing to get the right function assignment of the auxiliary based peripheral controllers on the MikroBUS headers is controlled by software using the `ARDUINO_MUX_CTRL` register (see [MUX Registers](#) on page 74).

Note that since the controllers that are mapped to the MikroBUS are shared with the Arduino controllers, and therefore the MikroBUS functions are only available when the Arduino multiplexer `ARDUINO_MUX_CTRL` is in the default mode (GPIO).



**Note** The MikroBUS is controlled by the controllers inside the subsystem, and this functionality is therefore only available when auxiliary based controllers are present in the build configuration of the subsystem.

Figure 22 MikroBus Headers

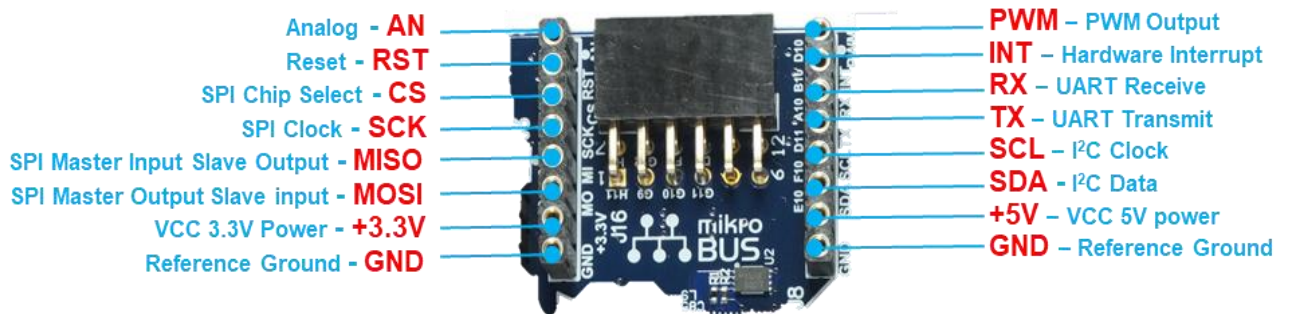


Table 8 shows the pin assignment on the I/O multiplexer.

Table 8 Pin Description of the MikroBUS Connectors

Pin	I/O	Pin	I/O
AN	ADC VIN0*	PWM	io_pwm0_ch[0]
RST	io_gpio0[20]	INT	io_gpio0[21]
CS	io_spi_mst0_cs[0]	RX	io_uart0_rxd
SCK	io_spi_mst0_clk	TX	io_uart0_txd
MISO	io_spi_mst0_rxd	SCL	io_i2c_mst1_scl
MOSI	io_spi_mst0_txd	SDA	io_i2c_mst1_sda

\*: \*ADC VIN0 is available through the on-board ADC and is read though io\_spi\_mst2.

### 2.16.3 Arduino

The ARC EM SDP provides an Arduino shield interface. Figure 23 shows the pin assignments. The Arduino shield interface is compatible with the Arduino UNO Rev3 with the following exceptions:

- 5-volt shields are not supported.
- The IOREF voltage on the ARC EM SDP board is fixed to 3.3 volts.



#### Note

The ICSP header is 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 23 Arduino Shield Interface

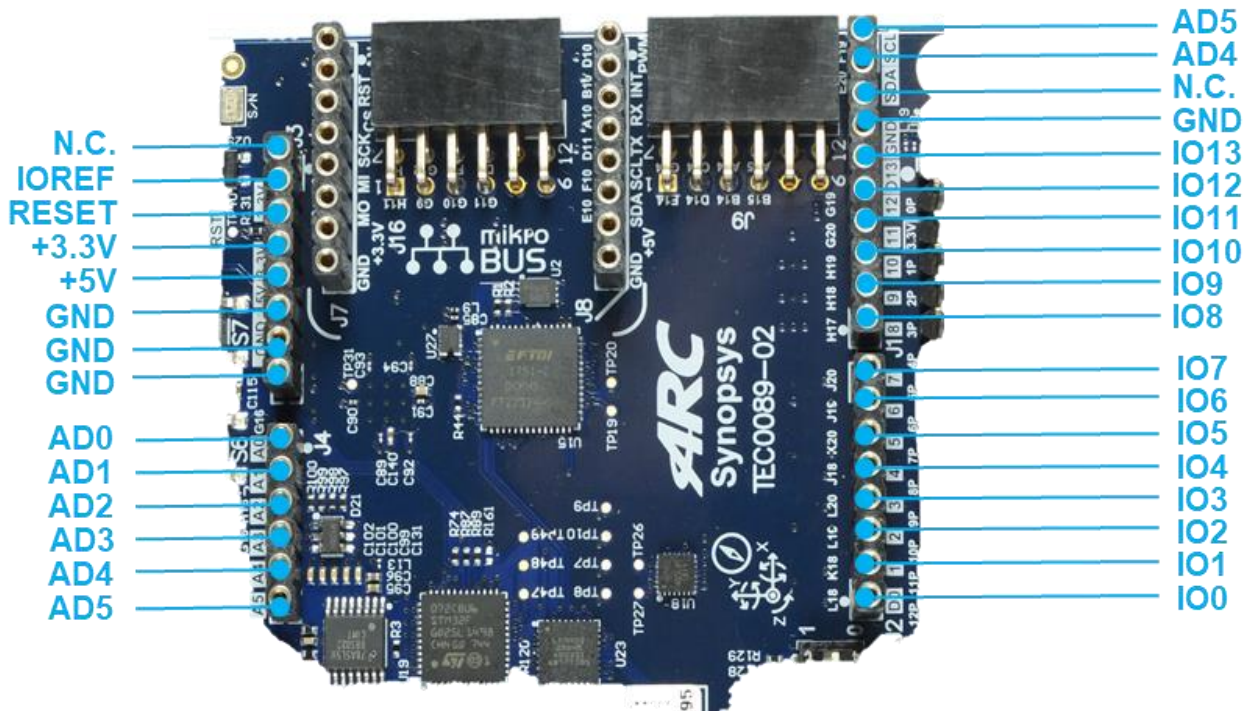


Table 9 shows the pin assignment on the I/O multiplexer. Multiplexing is controlled by software using the `ARDUINO_MUX_CTRL` register (see MUX Registers on page 74). After a reset, all the ports are configured as GPIO inputs.

Table 9 Pin Description of the Arduino Shield Interface

MUX	Pin	I/O-1	I/O-2	I/O-3	I/O-4	I/O-5
x	AD0	io_gpio0[14]	-			-
	AD1	io_gpio0[15]	-			-
	AD2	io_gpio0[16]	-			-
	AD3	io_gpio0[17]	-			-
CFG6	AD4	io_gpio0[18]	io_i2c_mst1_sda			-
	AD5	io_gpio0[19]	io_i2c_mst1_scl			-
CFG0	IO0	io_gpio0[0]	io_uart0_rxd			-
	IO1	io_gpio0[1]	io_uart0_txd			-
CFG1	IO2	io_gpio0[2]	io_gpio[2]			-
	IO3	io_gpio0[3]	io_pwm0_ch[0]			-
CFG2	IO4	io_gpio0[4]	io_gpio[4]			-
	IO5	io_gpio0[5]	io_pwm0_ch[1]			-
CFG3	IO6	io_gpio0[6]	io_pwm0_ch[2]			-
	IO7	io_gpio0[7]	io_gpio[7]			-
CFG4	IO8	io_gpio0[8]	io_gpio[8]			-
	IO9	io_gpio0[9]	io_pwm0_ch[3]			-
CFG5	IO10	io_gpio0[10]	io_spi_mst0_cs[0]	io_pwm0_ch[4]	io_gpio[10]	io_pwm0_ch[4]
	IO11	io_gpio0[11]	io_spi_mst0_mosi	io_pwm0_ch[5]	io_pwm0_ch[5]	io_gpio[11]
	IO12	io_gpio0[12]	io_spi_mst0_miso	io_gpio[12]	io_gpio[12]	io_gpio[12]
	IO13	io_gpio0[13]	io_spi_mst0_clk	io_gpio[13]	io_gpio[13]	io_gpio[13]

The analog input pins AD0 – 5 are also directly connected to the on-board ADC and are read through `io_spi_mst2`. The assignment is shown in [Table 10](#).

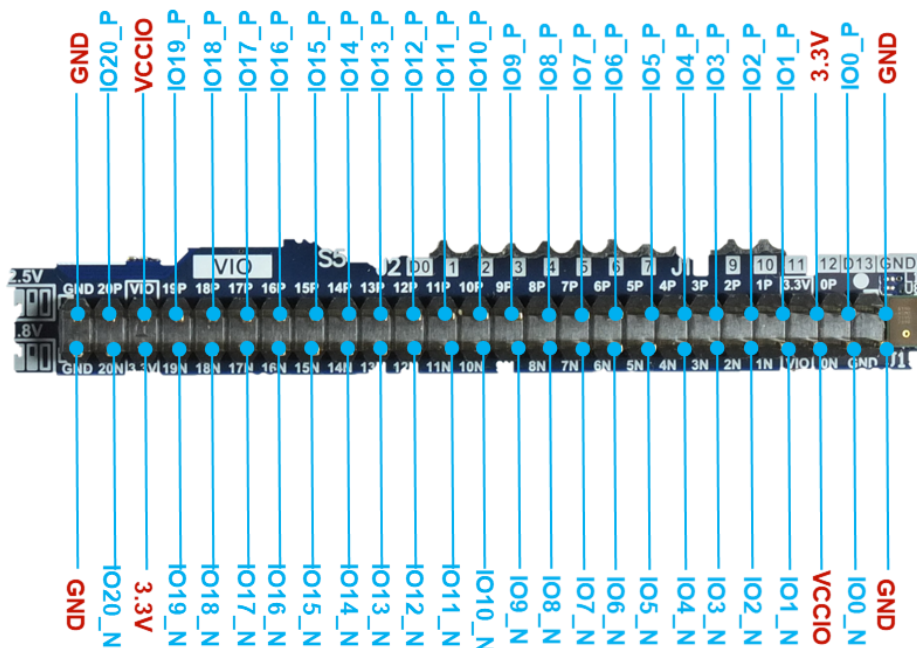
Table 10 ADC Input

Pin	I/O-1
AD0	ADC VIN1
AD1	ADC VIN2
AD2	ADC VIN3
AD3	ADC VIN4
AD4	ADC VIN5
AD5	ADC VIN6

### 2.16.4 Generic Header

In addition to the standard interfaces, the ARC EM SDP also features a generic pin header. Figure 24 shows the pin assignments. The pin header can be used as an EBI interface towards an external peripheral device or as a host interface.

Figure 24 Generic Pin Header Interface



Multiplexing is controlled by software using the `GENERIC_MUX_CTRL` register (see [MUX Registers](#) on page 74).

Table 11 Pin Description of the Generic Pin Interface

Pin	GPIO	EBI	Host-IF	DBG	Pin	GPIO	EBI	Host-IF	DBG
IO0_P		-	-	-	IO0_N		-	-	-
IO1_P	io_gpio2 [0]	CSn	splv0_sclk	-	IO1_N	io_gpio2 [4]	A0	-	-
IO2_P	io_gpio2 [1]	WAIT	splv0_miso	-	IO2_N	io_gpio2[5]	A1	-	-
IO3_P	io_gpio2 [2]	RDn	splv0_mosi	-	IO3_N	io_gpio2[6]	A2	-	-
IO4_P	io_gpio2 [3]	WRn	splv0_csn	-	IO4_N	io_gpio2[7]	A3	-	-
IO5_P	io_gpio2 [24]	D0	-	-	IO5_N	io_gpio2[8]	A4	i2c_slv0_sda	-
IO6_P	io_gpio2 [25]	D1	-	-	IO6_N	io_gpio2[9]	A5	i2c_slv0_scl	-
IO7_P	io_gpio2 [26]	D2	-	-	IO7_N	io_gpio2 [10]	A6	-	-
IO8_P	io_gpio2 [27]	D3	-	-	IO8_N	io_gpio2 [11]	A7	-	-
IO9_P	io_gpio2 [28]	D4	-	-	IO9_N	io_gpio2 [12]	A8	-	-
IO10_P	io_gpio2 [29]	D5	-	-	IO10_N	io_gpio2 [13]	A8	-	-
IO11_P	io_gpio2 [30]	D6	-	-	IO11_N	io_gpio2 [14]	A10	-	-
IO12_P	io_gpio2 [31]	D7	-	-	IO12_N	io_gpio2 [15]	A11	-	-
IO13_P		-	-	-	IO13_N	io_gpio2 [16]	A12	-	-
IO14_P		-	-	-	IO14_N	io_gpio2 [17]	A13	-	-
IO15_P		-	-	-	IO15_N	io_gpio2 [18]	A14	-	-
IO16_P		-	-	-	IO16_N	io_gpio2 [19]	A15	-	-
IO17_P		-	-	-	IO17_N	io_gpio2 [20]	A16	-	redpine_uart_cts

Pin	GPIO	EBI	Host-IF	DBG	Pin	GPIO	EBI	Host-IF	DBG
IO18_P		-	-	-	IO18_N	io_gpio2 [21]	A17	-	redpine_uart_txd
IO19_P		-	-	-	IO19_N	io_gpio2 [22]	A18	-	redpine_uart_rxd
IO20_P		-	-	-	IO20_N	io_gpio2 [23]	A19	-	redpine_uart_rts



## 3.1 Memory Map

Table 12 shows the memory map overview of the ARC EM SDP.

Table 12 ARC EM SDP Memory Map

Address Start Offset	Aperture	Slave
0xF000_0000	256 MB	AHB2APB bridge
0xE000_0000		RESERVED
0xD000_0000		RESERVED
0xC000_0000		RESERVED
0xB000_0000		RESERVED
0xA000_0000	16 KB	YCCM <sup>1</sup>
0x9000_0000	16 KB	XCCM <sup>1</sup>
0x8000_0000	128 KB	DCCM
0x7000_0000	128 KB	ICCM1 <sup>2</sup>
0x6000_0000	128 KB	ICCM0
0x5000_0000		RESERVED
0x4000_0000	16 MB	EBI data read/ write
0x3000_0000		RESERVED

Address Start Offset	Aperture	Slave
0x2000_0000	256 KB	SRAM
0x1000_0000	16 MB	PSRAM
0x0000_0000	256 KB	ROM

1. Only available if X/Y memory is present in the configuration.
2. Only available if ICCM1 memory is present in the configuration.

### 3.1.1 APB Peripheral Address Map

Table 13 lists all the APB peripherals that are accessible through the AHB2APB bridge 1. The total reserved address space for the AHB2APB bridge is 256 MB. Accessing a non-existing APB peripheral results in an AHB error response.

Table 13 APB Peripheral Address Map

Peripheral Name	Base	Size	Description
dw_cru	0xF000_0000	4 KB	Clock and Reset control. See <a href="#">Clock Registers</a> on page 53 for register details.
dw_creg	0xF000_1000	4 KB	Configuration register. See <a href="#">Control Registers</a> on page 71 for register details.
dw_gpio	0xF000_2000	4 KB	LED and switch control. For more information on the register details, see <a href="#">DesignWare DW_apb_gpio Databook</a> .
dw_nist_trng	0xF000_3000	1 KB	True Random Number Generator. For more information on the register details, see <a href="#">DesignWare DWC_trng_nist Databook</a> .
dw_dbg_uart	0xF000_4000	4 KB	UART for FTDI terminal. For more information on the register details, see <a href="#">DesignWare DW_apb_uart Databook</a> .

Peripheral Name	Base	Size	Description
dw_wdt	0xF000_5000	4 KB	Watchdog Timer. For more information on the register details, see <a href="#">DesignWare DW_apb_wdt Databook</a> .
dw_timers0	0xF000_6000	4 KB	Timers 0. For more information on the register details, see <a href="#">DesignWare DW_apb_timers Databook</a> .
dw_timers1	0xF000_7000	4 KB	Timers 1. For more information on the register details, see <a href="#">DesignWare DW_apb_timers Databook</a> .
dw_spi_mst0	0xF000_8000	4 KB	Redpine SPI host interface. For more information on the register details, see <a href="#">DesignWare DW_apb_ssi Databook</a> .
dw_uart	0xF000_9000	4 KB	Redpine UART host interface. For more information on the register details, see <a href="#">DesignWare DW_apb_uart Databook</a> .
Reserved	0xF000_A000	4 KB	
dw_sdio	0xF001_0000	4 KB	SD-card controller. For more information on the register details, see <a href="#">DesignWare DWC Mobile Storage Host Databook</a> .
Reserved	0xF001_1000		
dw_spi_mst1	0xF100_0000	16 MB	Quad-SPI flash controller. For more information on the register details, see <a href="#">DesignWare DW_apb_ssi Databook</a> .
dw_ebi	0xF200_0000	4 KB	External Bus Interface Controller. See <a href="#">EBI Registers</a> on page 80 for register details.
xilinx_axi_emc	0xF200_1000	4 KB	PSRAM controller. For more information on the register details, see <a href="#">Xilinx AXI EMC</a> .

### 3.1.2 Auxiliary Based Peripherals

Depending on the selected build configuration of the subsystem, a set of auxiliary based IO peripherals is available. The peripherals are not controlled over a bus but are directly controlled using the core's internal auxiliary registers. Table 14 shows the list of auxiliary based IO peripherals.

For more information about the individual IO peripherals see:

- [DesignWare ARC Data Fusion IP Subsystem I/O Databook](#)
- [DesignWare Smart Sensor and Control IP Subsystem I/O Databook](#).

Table 14 Auxiliary Based IO Peripherals

Peripheral Name	AUX Base	Size	Description
io_spi_mst0	0x8001_0000	256 byte	SPI master for Arduino and MikroBUS
io_spi_mst1	0x8001_0100	256 byte	SPI master for Pmod_A, Pmod_B, Pmod_C
io_spi_mst2	0x8001_0200	256 byte	SPI master for on-board ADC
io_spi_slv0	0x8001_1000	256 byte	SPI slave for Generic Pin Header
io_i2c_mst0	0x8001_2000	256 byte	I2C master for on-board I2C bus (Codec, 9D sensor)
io_i2c_mst1	0x8001_2100	256 byte	I2C master for Arduino
io_i2c_mst2	0x8001_2200	256 byte	I2C master for Pmod
io_i2c_slv0	0x8001_3000	256 byte	I2C slave host interface for generic pin Header
io_uart0	0x8001_4000	256 byte	UART for Arduino and MikroBUS
io_uart1	0x8001_4100	256 byte	UART for Pmod_A

Peripheral Name	AUX Base	Size	Description
io_uart2	0x8001_4200	256 byte	UART for Pmod_B
io_uart3	0x8001_4300	256 byte	UART for Pmod_C
io_gpio0	0x8001_7000	256 byte	GPIO for Arduino
io_gpio1	0x8001_7100	256 byte	GPIO for Pmod_A, Pmod_B and Pmod_C
io_gpio2	0x8001_7200	256 byte	GPIO for Generic Pin Header
io_i2s_tx	0x8001_9000	256 byte	I2S TX for on-board audio CODEC
io_i2s_rx	0x8001_a000	256 byte	I2S RX for on-board audio CODEC
io_pdm_rx	0x8001_b000	256 byte	PDM for on-board MEMs microphones
io_pwm0	0x8001_d000	256 byte	PWM for Arduino and MikroBUS
io_pwm1	0x8001_d100	256 byte	PWM for Pmod_A, Pmod_B and Pmod_C

## 3.2 Software Interfaces

This section describes the software interfaces for the custom IP inside the ARC EM Software Development Platform. For more information on the software interfaces for the DesignWare IP and the DesignWare ARC subsystems, see the Databooks listed in [References](#) on page 87.

### 3.2.1 Clock Registers

The clock registers and the associated clock circuitry (that is, the PLLs plus clock dividers) inside the ARC EM SDP are implemented by the Clock and Reset Unit (CRU) module. The clock registers are used to program the PLLs and clock dividers. The CRU 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 (for example, AHB, APB, and IP core clocks)
- REF PLL:
  - Used to generate reference clocks

In addition to the PLLs, the CRU module includes integer dividers for generating audio reference clocks based on a fixed 12.288 MHz audio input clock.

Additionally, the CRU module implements a measurement logic that can be used to verify that PLL and dividers have been programmed correctly. Guidelines for programming the PLL can be found in [PLL Programming](#) on page 59, and for clock measurements in [Frequency Measurement](#) on page 60

[Table 15](#) lists the registers for the CRU module including a brief description and their offset to the base address of the CRU (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.

Table 15 CGU Clock Register Overview

Name	Address Offset <sup>[1]</sup>	Access	Description
<b>PLL Control and Status Registers</b>			
<b>- ARC PLL</b>			
CGU_ARC_PLL_IDIV_CTRL	0x0000	RW	ARC PLL input divider control register
CGU_ARC_PLL_STATUS	0x0004	R	ARC PLL status register
CGU_ARC_PLL_FMEAS	0x0008	RW	ARC PLL frequency measurement register

Name	Address Offset <sup>[1]</sup>	Access	Description
CGU_ARC_PLL_FBDIV_CTRL	0x000C	RW	ARC PLL feedback divider control register
<b>- SYS PLL</b>			
CGU_SYS_PLL_IDIV_CTRL	0x0010	RW	SYS PLL input divider 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_FBDIV_CTRL	0x001C	RW	SYS PLL feedback divider control register
<b>- REF PLL</b>			
CGU_REF_PLL_IDIV_CTRL	0x0020	RW	REF PLL input divider control register
CGU_REF_PLL_STATUS	0x0024	R	REF PLL status register
CGU_REF_PLL_FMEAS	0x0028	RW	REF PLL frequency measurement register
CGU_REF_PLL_FBDIV_CTRL	0x002C	RW	REF PLL feedback divider control register

Name	Address Offset <sup>[1]</sup>	Access	Description
<b>Clock Control and Status Registers</b>			
<b>ARC PLL</b>			
CGU_ARC_ODIV_ARC	0x0080	RW	Clock output divider register for ARC EM clock
CGU_ARC_FMEAS_ARC	0x0084	RW	Clock measurement register for ARC EM clock
<b>SYS PLL</b>			
CGU_SYS_ODIV_AHB	0x00E0	RW	Clock output divider register for AHB clock
CGU_SYS_FMEAS_AHB	0x00E4	RW	Clock measurement register for AHB clock
CGU_SYS_ODIV_APB	0x00F0	RW	Clock output divider register for APB clock
CGU_SYS_FMEAS_APB	0x00F4	RW	Clock measurement register for APB clock
<b>REF PLL</b>			
CGU_REF_ODIV_SDIO	0x0140	RW	Clock output divider register for SDIO clock
CGU_REF_FMEAS_SDIO	0x0144	RW	Clock measurement register for SDIO clock



Name	Address Offset <sup>[1]</sup>	Access	Description
CGU_REF_ODIV_SPI	0x0150	RW	Clock output divider register for SPI clock
CGU_REF_FMEAS_SPI	0x0154	RW	Clock measurement register for SPI clock
CGU_REF_ODIV_TIMER	0x0160	RW	Clock output divider register for timer clock
CGU_REF_FMEAS_TIMER	0x0164	RW	Clock measurement register for timer clock
CGU_REF_ODIV_UART	0x0170	RW	Clock output divider register for UART clock
CGU_REF_FMEAS_UART	0x0174	RW	Clock measurement register for UART clock
CGU_REF_ODIV_EBI	0x0180	RW	Clock output divider register for EBI clock
CGU_REF_FMEAS_EBI	0x0184	RW	Clock measurement register for EBI clock
CGU_REF_ODIV_I2C	0x0190	RW	Clock output divider register for I2C clock
CGU_REF_FMEAS_I2C	0x0194	RW	Clock measurement register for I2C clock

Name	Address Offset <sup>[1]</sup>	Access	Description
<b>AUDIO</b>			
CGU_AUDIO_FMEAS_I2S	0x01A4	RW	Clock measurement register for I2S clock
CGU_AUDIO_IDIV_I2S	0x01A8	RW	Clock integer divider register for I2S clock
CGU_AUDIO_FMEAS_PDM	0x01B4	RW	Clock measurement register for PDM clock
CGU_AUDIO_IDIV_PDM	0x01B8	RW	Clock integer divider register for PDM clock
<b>Reset Control Registers</b>			
CGU_ARC_RST_CTRL	0x0380	RW1C	Reset control register for ARC clock
CGU_SYS_RST_CTRL	0x0384	RW1C	Reset control register for system clocks
CGU_REF_RST_CTRL	0x0388	RW1C	Reset control register for reference clocks
CGU_AUDIO_RST_CTRL	0x038C	RW1C	Reset control register for audio clocks
CGU_SYS_RST_OUT_CTRL	0x03A0	RW1C	Reset control register for system reset output

Name	Address Offset <sup>[1]</sup>	Access	Description
<b>Standard Registers</b>			
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
<p>[1]: The following access types are defined:</p> <ul style="list-style-type: none"> <li>• RW            Read/Write register</li> <li>• R             Read-only register</li> <li>• W             Write-only register</li> <li>• RW1C        Read-only, Write-1-to-Clear register</li> </ul>			

### 3.2.1.1 PLL Programming

The Clock and Reset Unit (CRU) implements logic (FSM plus a clock switch) that allows glitchless reprogramming of the PLLs with minimal software interaction. The PLLs are implemented using the Xilinx MMCM primitive, which allows for dynamic reconfiguration.

Software must program the desired PLL output frequency using a set of three counter dividers. Each PLL clock input has a programmable counter divider (D). Each PLL clock output has a programmable counter divider (O). In addition, the feedback clock has a fractional counter divider (M).

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

$$F_{vco} = F_{clk_{in}} * (M/D)$$

$$F_{out} = F_{clk_{in}} * M / (D * O)$$

where:

- D = input divider value
- M = feedback divider value
- O = output divider value

The input clock frequency  $f_{clk_{in}}$  of the ARC EM SDP is fixed to 100 MHz. The VCO frequency  $f_{VCO}$  needs to be between 600 MHz and 1440 MHz in accordance with the Xilinx datasheet.

To detect completion of the PLL reprogramming sequence, software polls the `CGU_*_PLL_STATUS` register.

### 3.2.1.2 Frequency Measurement

The frequency measurement module measures the frequency of a clock relative to the 100 MHz input-reference clock. The frequency measurement modules can be controlled through the `CRU_*_FMEAS` registers.

When the `START` bit in the `CRU_*_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 `CRU_*_FMEAS` register is set to 1.

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

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



#### Note

- By default, both counters start counting from zero. However, the frequency calculation `RCNT` can be initialized with a non-zero value. When `RCNT` is initialized with a value equal to `215 - 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 1000 MHz versus 1 kHz) because one counter saturates while the other counter only has a small count value.
- Due to synchronization, the 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 as long as possible.

### 3.2.1.3 Register Descriptions

#### 3.2.1.3.1 PLL Control and Status Registers

#### CGU\_m\_PLL\_IDIV\_CTRL – PLL Control Register (Address Offset = 0x0000 + p\*0x10)

Bit	Name	Access	Value	Description
0	BYPASS	RW	1*	Bypass PLL input divider counter
1	HIGHEDGE	RW	0*	Force high time counter to transition on a falling edge (add half a VCO cycle)
3:2	Reserved			
9:4	HIGHTIME	RW	see below**	Set number of VCO cycles that the clock remains high (D <sub>HIGH</sub> )
11:10	Reserved			
17:12	LOWTIME	RW	see below**	Set number of VCO cycles that the clock remains low (D <sub>LOW</sub> )
30:18	Reserved		*	
31	UPDATE	RW1C	0*	Update the PLL with the configured values

\*: Reset value

\*\*: The values of m and p are:

p = 0	m = ARC	D <sub>HIGH</sub> =NA	D <sub>LOW</sub> =NA	M <sub>HIGH</sub> =5	M <sub>LOW</sub> =5	F <sub>VCO</sub> =1000 MHz
p = 1	m = SYS	D <sub>HIGH</sub> =NA	D <sub>LOW</sub> =NA	M <sub>HIGH</sub> =5	M <sub>LOW</sub> =5	F <sub>VCO</sub> =1000 MHz
p = 2	m = REF	D <sub>HIGH</sub> =NA	D <sub>LOW</sub> =NA	M <sub>HIGH</sub> =5	M <sub>LOW</sub> =5	F <sub>VCO</sub> =1000 MHz

**CGU\_m\_PLL\_STATUS – PLL Status Register (Address Offset = 0x0004 + p\*0x10)**

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

\*: Reset value

The values of m and p are:

p = 0	m =ARC
p = 1	m =SYS
p = 2	m =REF

**CGU\_m\_PLL\_FMEAS – PLL Measurement Register (Address Offset = 0x0008 + p\*0x10)**

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. <ul style="list-style-type: none"> <li>Measured frequency = (FCNT / RCNT) * f<sub>ref</sub></li> </ul>

\*: Reset value

The values of m and p are:

p = 0	m =ARC
p = 1	m =SYS
p = 2	m =REF

**CGU\_m\_FB\_DIV\_CTRL – PLL FB Control Register (Address Offset = 0x000C + p\*0x10)**

Bit	Name	Access	Value	Description
0	BYPASS	RW	1*	Bypass PLL input divider counter
1	HIGHEDGE	RW	0*	Force high time counter to transition on a falling edge (add half a VCO cycle)
3:2	Reserved			
9:4	HIGHTIME	RW	See below**	Set number of VCO cycles that the clock remains high (M <sub>HIGH</sub> )
11:10	Reserved			
17:12	LOWTIME	RW	See below**	Set number of VCO cycles that the clock remains low (M <sub>LOW</sub> )
30:18	Reserved		*	
31	UPDATE	RW1C	0*	Update the PLL with the configured values

\*: Reset value

\*\*: The values of m and p are:

p = 0    m =ARC    D<sub>HIGH</sub>=NA    D<sub>LOW</sub>=NA    M<sub>HIGH</sub>=5    M<sub>LOW</sub>=5    F<sub>VCO</sub>=1000 MHz

p = 1    m =SYS    D<sub>HIGH</sub>=NA    D<sub>LOW</sub>=NA    M<sub>HIGH</sub>=5    M<sub>LOW</sub>=5    F<sub>VCO</sub>=1000 MHz

p = 2    m =REF    D<sub>HIGH</sub>=NA    D<sub>LOW</sub>=NA    M<sub>HIGH</sub>=5    M<sub>LOW</sub>=5    F<sub>VCO</sub>=1000 MHz

**3.2.1.3.2 Clock Control and Status Registers****CGU\_m\_ODIV\_c – Clock Divider Register (Address Offset = 0x0080 + p\*0x60 + n\*0x10)**

Bit	Name	Access	Value	Description
0	BYPASS	RW	0*	Bypass PLL input divider counter
1	HIGHEDGE	RW	0*	Force high time counter to transition on a falling edge (add half a VCO cycle)
3:2	Reserved			
9:4	HIGHTIME	RW	See below**	Set number of VCO cycles that the clock remains high (O <sub>HIGH</sub> )
11:10	Reserved			
17:12	LOWTIME	RW	See below**	Set number of VCO cycles that the clock remains low (O <sub>LOW</sub> )
30:18	Reserved		*	
31	UPDATE	RW1C	0*	Update the PLL with the configured values

\*: Reset value

\*\* : The values of c, m, n, and p are:

m = ARC	p = 0	c = ARC	ARC HS clock	n = 0	O <sub>HIGH</sub> =10	O <sub>LOW</sub> =10	F <sub>out</sub> =50 MHz
		c = AHB	AHB clock	n = 0	O <sub>HIGH</sub> =10	O <sub>LOW</sub> =10	F <sub>out</sub> =50 MHz
m = SYS	p = 1	c = APB	APB clock	n = 1	O <sub>HIGH</sub> =10	O <sub>LOW</sub> =10	F <sub>out</sub> =50 MHz
		c = SDIO	SDIO clock	n = 0	O <sub>HIGH</sub> =2	O <sub>LOW</sub> =3	F <sub>out</sub> =200 MHz
		c = SPI	SPI clock	n = 1	O <sub>HIGH</sub> =25	O <sub>LOW</sub> =25	F <sub>out</sub> =20 MHz
m = REF	p = 2	c = TIMER	TIMER core clock	n = 2	O <sub>HIGH</sub> =5	O <sub>LOW</sub> =5	F <sub>out</sub> =100 MHz
		c = UART	UART core clock	n = 3	O <sub>HIGH</sub> =5	O <sub>LOW</sub> =5	F <sub>out</sub> =100 MHz
		c = EBI	EBI core clock	n = 4	O <sub>HIGH</sub> =5	O <sub>LOW</sub> =5	F <sub>out</sub> =100 MHz
		c = I2C	I2C core clock	n = 5	O <sub>HIGH</sub> =5	O <sub>LOW</sub> =5	F <sub>out</sub> =100 MHz



**CGU\_m\_FMEAS\_c – Clock Measurement Register (Address Offset = 0x0084 + p\*0x60 + n\*0x10)**

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 is complete.
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 completes: <ul style="list-style-type: none"> <li>Measured frequency = (FCNT / RCNT) * f<sub>ref</sub></li> </ul>

\*: Reset value

The values of c, m, and p are:

m = ARC	p = 0	c = ARC	ARC HS clock	n = 0
		c = AHB	AHB clock	n = 0
m = SYS	p = 1	c = APB	APB clock	n = 1
		c = SDIO	SDIO clock	n = 0
		c = SPI	SPI clock	n = 1
m = REF	p = 2	c = TIMER	TIMER core clock	n = 2
		c = UART	UART core clock	n = 3
		c = EBI	EBI core clock	n = 4

**CGU\_AUDIO\_FMEAS\_c – Clock Measurement Register (Address Offset = 0x01A4 + p\*0x10)**

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 is complete.
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 completes: <ul style="list-style-type: none"> <li>Measured frequency = (FCNT / RCNT) * f<sub>ref</sub></li> </ul>

\*: Reset value  
The values of c and p are:

p = 0	c = I2S	I2S audio reference clock
p = 1	c = PDM	PDM audio reference clock

**CGU\_AUDIO\_IDIV\_c – Clock Measurement Register (Address Offset = 0x01A8 + n\*0x10)**

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

\*: Reset value  
\*\* The values of c and p are:

p = 0	c = I2S	I2S audio reference clock
p = 1	c = PDM	PDM audio reference clock

**3.2.1.3.3 Reset Control Registers****CGU\_ARC\_RST\_CTRL – ARC Reset Control Register (Address Offset = 0x380)**

Bit	Name	Access	Description
0	ARC_RST	RW	Reset the ARC clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
31:1	-	RW	Reserved; must be written as 0 Value: 0*
*: Reset value			

**CGU\_SYS\_RST\_CTRL – ARC Reset Control Register (Address Offset = 0x384)**

Bit	Name	Access	Description
0	AHB_RST	RW	Reset the AHB clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
1	APB_RST	RW	Reset the APB clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
31:2	-	RW	Reserved; must be written as 0 Value: 0*
*: Reset value			

**CGU\_REF\_RST\_CTRL – ARC Reset Control Register (Address Offset = 0x388)**

Bit	Name	Access	Description
0	SDIO_RST	RW	Reset the SDIO clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
1	SPI_RST	RW	Reset the SPI clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
2	TIMER_RST	RW	Reset the timers clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
3	UART_RST	RW	Reset the UART clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
4	EBI_RST	RW	Reset the EBI clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
5	I2C_RST	RW	Reset the I2C clock domain Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
31:6	Reserved	RW	Reserved; must be written as 0 Value: 0*
*: Reset value			

**CGU\_SYS\_RST\_OUT\_CTRL – System Reset Output Control Register (Address Offset = 0x3A0)**

Bit	Name	Access	Description
0	SYS_RST_OUT	RW	Toggle external reset output Value: <ul style="list-style-type: none"> <li>• 0*: No</li> <li>• 1: Yes</li> </ul>
31:1	-	RW	Reserved; must be written as 0 Value: 0*

\*: Reset value

**3.2.1.3.4 Standard Registers****CGU\_IP\_SW\_RESET – CGU Software Reset Register (Address Offset = 0xFF0)**

Bit	Name	Access	Description
0	SW_RESET	RW1C	Writing a 1 to this bit initiates a software reset of the full ARC EM Software Development Platform system.
31:16	SW_RESET_DELAY	RW	Delay between software reset command and reset assertion. Value: 0x0000*

\*: Reset value

**CGU\_IP\_VERSION – CGU Version Register (Address Offset = 0xFF8)**

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

**CGU\_IP\_TYPE – CGU Type Register (Address Offset = 0xFFC)**

Bit	Name	Access	Description
15:0	ID	R	IP code <ul style="list-style-type: none"> <li>• 0x0100 = CGU</li> </ul> Value: 0x1*
31:16	DW	R	Hex code for the two ASCII letters AD (ARC Development) Value: 0x4144*
*: Reset value			

## 3.2.2 Control Registers

The global control registers inside the ARC EM SDP are implemented by the CREG module. The global control registers control configuration settings for sub-modules that do not have a software interface (for example: the GPIO mux). Additionally, 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 the software interrupts
- Identification registers for the ARC EM SDP

[Table 16](#) lists the registers for the CREG module, including a brief description and their offset to the base address of the CREG (`0xF000_1000`). All the registers are 32 bits 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.

Table 16 CREG Control Register Overview

Name	Address Offset	Access <sup>[1]</sup>	Description
<b>Boot Registers</b>			
CREG_BOOT	0x0	RW	Boot register
CREG_START	0x10	RW	Start register
<b>Multiplexer Registers</b>			
PMOD_MUX_CTRL	0x30	RW	Arduino MUX register
ARDUINO_MUX_CTRL	0x34	RW	Pmod MUX register
GENERIC_MUX_CTRL	0x38	RW	Generic pin header MUX register
<b>Standard Registers</b>			
CREG_IP_SW_RESET	0xFF0	RW1C	CREG IP software reset register
CREG_IP_PRODUCT	0xFF4	R	CREG IP product register
CREG_IP_VERSION	0xFF8	R	CREG IP version register
CREG_IP_TYPE	0xFFC	R	CREG IP type register
<p>[1] The following access types are defined:</p> <p>RW     Read/Write register</p> <p>R       Read-only register</p> <p>W       Write-only register</p> <p>RW1C   Read-only, Write-1-to-Clear Register</p>			



### 3.2.2.1 Boot Registers

#### CREG\_BOOT – Boot Register (Address Offset = 0x0)

Bit	Name	Access	Description
3:0	Reserved	RW	
5:4	IMAGE <sup>[1]</sup>	RW	Bootloader image location Value: <ul style="list-style-type: none"> <li>• 0*: ROM</li> <li>• 1: SPI flash</li> <li>• 2: EBI</li> </ul>
6	REDPINE_HOST_IF	RW	Redpine host interface selection Value: <ul style="list-style-type: none"> <li>• 0*: ROM</li> <li>• 1: SPI flash</li> </ul>
7	XIP_EN	RW	Control the XIP_EN pin of the <code>dw_spi_mst1</code> <ul style="list-style-type: none"> <li>• 0*: XIP not asserted</li> <li>• 1: XIP asserted</li> </ul>
8	WP	RW	Write protect for ROM <ul style="list-style-type: none"> <li>• 0: No write protect, ROM operates as RAM</li> <li>• 1*: Write protect enabled</li> </ul>
<p>*: Reset value</p> <p>[1]: Do not attempt to write to EBI_CS_MOD register while access to CS area is in progress.</p>			

**CREG\_CPU\_START – CPU Start Register (Address Offset = 0x10)**

Bit	Name	Access	Description
0	START	RW1C	Writing a 1 to this bit generates a <code>cpu_start</code> pulse for ARC EM. Value: 0x0*
3:1	Reserved		
4	START_MODE	RW	Boot start mode <ul style="list-style-type: none"> <li>0x0<sup>[1]</sup>: Start ARC core manually (CREG, external start button or debugger).</li> <li>0x1: Start ARC core autonomously after reset.</li> </ul>
8	POL	RW	Polarity of <code>cpu_start</code> pulse. Value: <ul style="list-style-type: none"> <li>0x0: Active low</li> <li>0x1*: Active high</li> </ul>
31:9	Reserved		
*: Reset value			
[1]: Reset value for START_MODE is sampled from <code>boot_start_mode</code> pin during power-on-reset.			

**3.2.2.2 MUX Registers****PMOD\_MUX\_CTRL – Pmod Mux Register (Address Offset = 0x30)**

Bit	Name	Access	Description
3:0	PMOD_A_CFG1	RW	<ul style="list-style-type: none"> <li>• GPIO Value: 0x0*</li> <li>• I2C Value: 0x1</li> <li>• SPI Value: 0x2</li> <li>• UART1 Value: 0x3</li> <li>• UART2 Value: 0x4</li> <li>• PWM1 Value: 0x5</li> <li>• PWM2 Value: 0x6</li> <li>• Reserved Value: 0x7</li> </ul>
7:4	PMOD_A_CFG2	RW	<ul style="list-style-type: none"> <li>• GPIO Value: 0x0*</li> </ul>
11:8	PMOD_B_CFG1	RW	<ul style="list-style-type: none"> <li>• GPIO Value: 0x0*</li> <li>• I2C Value: 0x1</li> <li>• SPI Value: 0x2</li> <li>• UART1 Value: 0x3</li> <li>• UART2 Value: 0x4</li> <li>• PWM1 Value: 0x5</li> <li>• PWM2 Value: 0x6</li> <li>• Reserved Value: 0x7</li> </ul>

Bit	Name	Access	Description
15:12	PMOD_B_CFG2	RW	<ul style="list-style-type: none"> <li>GPIO Value: 0x0*</li> </ul>
19:16	PMOD_C_CFG1	RW	<ul style="list-style-type: none"> <li>GPIO Value: 0x0*</li> <li>I2C Value: 0x1</li> <li>SPI Value: 0x2</li> <li>UART1 Value: 0x3</li> <li>UART2 Value: 0x4</li> <li>PWM1 Value: 0x5</li> <li>PWM2 Value: 0x6</li> <li>Reserved Value: 0x7</li> </ul>
23:20	PMOD_C_CFG2	RW	<ul style="list-style-type: none"> <li>GPIO Value: 0x0*</li> </ul>
*: Reset value			

### ARDUINO\_MUX\_CTRL – ARDUINO Mux Register (Address Offset = 0x34)

Bit	Name	Access	Description
3:0	ARDUINO_CFG0	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
7:4	ARDUINO_CFG1	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
11:8	ARDUINO_CFG2	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
15:12	ARDUINO_CFG3	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
19:16	ARDUINO_CFG4	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
23:20	ARDUINO_CFG5	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0</li> <li>I/O-2 Value: 0x1</li> <li>I/O-3 Value: 0x2</li> <li>I/O-4 Value: 0x3</li> <li>I/O-5 Value: 0x4</li> </ul>

Bit	Name	Access	Description
27:24	ARDUINO_CFG6	RW	<ul style="list-style-type: none"> <li>I/O-1 Value: 0x0*</li> <li>I/O-2 Value: 0x1</li> </ul>
*: Reset value			

### GENERIC\_MUX\_CTRL – Generic Mux Register (Address Offset = 0x38)

Bit	Name	Access	Description
3:0	GENERIC_CFG0	RW	<ul style="list-style-type: none"> <li>GPIO Value: 0x0*</li> <li>EBI Value: 0x1</li> <li>HOST-IF Value: 0x2</li> <li>DBG Value: 0x3</li> </ul>
*: Reset value			

### 3.2.2.3 Standard Registers

#### CREG\_IP\_SW\_RESET – CREG Software Reset Register (Address Offset = 0xFF0)

Bit	Name	Access	Description
0	RESET	RW1C	<p>Software reset.</p> <p>Writing a 1 to this bit initiates a software reset of all 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.</p>

**CREG\_IP\_PRODUCT – CREG Product Register (Address Offset = 0xFF4)**

Bit	Name	Access	Description
31:0	PRODUCT_NAME	R	Hex code for the two ASCII letters ARC EM Software Development Platform (ESDP) Value: 0x45534450

**CREG\_IP\_VERSION – CREG Version Register (Address Offset = 0xFF8)**

Bit	Name	Access	Description
15:0	MINOR_REV	R	Minor version of the ARC EM SDP Value: 0x0000*
31:16	MAJOR_REV	R	Major version of the ARC EM SDP Value: 0x0001*
*: Reset value			

**CREG\_IP\_TYPE – CREG Type Register (Address Offset = 0xFFC)**

Bit	Name	Access	Description
15:0	ID	R	IP code <ul style="list-style-type: none"> <li>0x0200 = CREG</li> </ul> Value: 0x0200*
31:16	DW	R	Hex code for the two ASCII letters ARC Development (AD) Value: 0x4144*
*: Reset value			

### 3.2.3 EBI Registers

Name	Address Offset	Access <sup>[1]</sup>	Description
EBI_CS_MOD	0x002	RW	CS mode register
EBI_CS_WCR1	0x004	RW	CS wait control register 1
EBI_CS_WCR2	0x008	RW	CS wait control register 2
EBI_CS_CR	0x802	RW	CS control register
EBI_CS_REC	0x80A	RW	CS recovery cycle register
EBI_CS_RECEN	0x880	RW	CS recovery cycle insertion enable 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

#### 3.2.3.1 EBI Registers

##### EBI\_CS\_MOD – CS Mode Register (Address Offset = 0x002)

Bit	Name	Access	Description
0	WRMOD	RW	Write access mode select Value: <ul style="list-style-type: none"> <li>0x0*: Byte strobe mode</li> <li>0x1: Single write strobe mode</li> </ul>
3	EWENB	RW	External wait enable/disable Value: <ul style="list-style-type: none"> <li>0x0*: Disabled</li> <li>0x1: Enabled</li> </ul>



Bit	Name	Access	Description
8	PRENB	RW	Page read access enable/disable Value: <ul style="list-style-type: none"> <li>0x0*: Disabled</li> <li>0x1: Enabled</li> </ul>
9	PWENB	RW	Page write access enable/disable Value: <ul style="list-style-type: none"> <li>0x0*: Disabled</li> <li>0x1: Enabled</li> </ul>
15	PRMOD	RW	Page read access mode select Value: <ul style="list-style-type: none"> <li>0x0*: Normal access compatible mode</li> <li>0x1: External data read continuous assertion mode</li> </ul>
<p>*: Reset value</p> <p>[1]: Do not attempt to write to EBI_CS_MOD register while access to CS area is in progress.</p>			

**EBI\_CS\_WCR1 – CS Wait Control Register 1 (Address Offset = 0x004)**

Bit	Name	Access	Description
2:0	CSPWAIT [2]	RW	Page write cycle wait time Value: <ul style="list-style-type: none"> <li>• 0x0: No wait is inserted</li> <li>• 0x1: Wait time is 1 EBI clock cycle</li> <li>• .... ..</li> <li>• 0x7*: Wait time is 7 EBI clock cycles</li> </ul>
10:8	CSPRWAIT [3]	RW	Page read cycle wait time Value: <ul style="list-style-type: none"> <li>• 0x7*</li> </ul>
20:16	CSWAIT	RW	Normal write cycle wait time Value: <ul style="list-style-type: none"> <li>• 0x1F*</li> </ul>
28:24	CSRWAIT	RW	Normal read cycle wait time Value: <ul style="list-style-type: none"> <li>• 0x1F*</li> </ul>
<p>*: Reset value</p> <p>[1]: Do not attempt to write to the EBI_CS_WCR1 register while the external bus is being accessed.</p> <p>[2]: The CSPWAIT value is only valid when PWENB bit in EBI_CS_MOD is set to 1.</p> <p>[3]: The CSPRWAIT value is only valid when PRENB bit in EBI_CS_MOD is set to 1.</p>			

**EBI\_CS\_WCR2 – CS Wait Control Register 2 (Address Offset = 0x008)**

Bit	Name	Access	Description
2:0	CSROFF	RW	Read access CS extension cycles Value: <ul style="list-style-type: none"> <li>• 0x0: No wait is inserted</li> <li>• 0x1: Wait duration is 1 EBI clock cycle</li> <li>• 0x2: Wait duration is 2 EBI clock cycles</li> <li>• ....</li> <li>• ..... .....</li> <li>• 0x7*: Wait duration is 7 EBI clock cycles</li> </ul>
6:4	CSWOFF	RW	Write access CS extension cycles Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>
10:8	WDOFF	RW	Write data output extension cycles Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>
13:12	AWAIT	RW	Address cycle wait time Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>
18:16	RDON	RW	RD assert wait time Value: <ul style="list-style-type: none"> <li>• 0x7*</li> </ul>
22:20	WRON	RW	WR assert wait time Value: <ul style="list-style-type: none"> <li>• 0x7*</li> </ul>

Bit	Name	Access	Description
26:24	WDON	RW	Write data output wait time Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>
30:28	CSON	RW	CS assert wait time Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>

\*: Reset value  
[1]: Do not attempt to write to EBI\_CS\_WCR2 register while the external bus is being accessed.

### EBI\_CS\_CR – CS Control Register (Address Offset = 0x802)

Bit	Name	Access	Description
0	EXENB	RW	Operation enable / disable Value: <ul style="list-style-type: none"> <li>• 0x0: Disabled</li> <li>• 0x1*: Enabled</li> </ul>
5:4	BSIZE	R	External bus width select Value: <ul style="list-style-type: none"> <li>• 0x2*: 8-Bit</li> </ul>
8	EMODE	RW	Endian mode Value: <ul style="list-style-type: none"> <li>• 0x0*: Little endian</li> <li>• 0x1: Big endian</li> </ul>
12	MPXEN	R	Address/data bus multiplex mode Value: <ul style="list-style-type: none"> <li>• 0x0*: Separate address/data bus</li> </ul>

Bit	Name	Access	Description
*: Reset value			
[1]: Do not attempt to write to EBI_CS_CR register while the external bus is being accessed.			

**EBI\_CS\_REC – CS Recovery Control Register (Address Offset = 0x80A)**

Bit	Name	Access	Description
3:0	RRCV	RW	Read recovery cycles Value: <ul style="list-style-type: none"> <li>• 0x0*: No recovery cycle is inserted</li> <li>• 0x1: One recovery cycle</li> <li>• 0x2: Two recovery cycles</li> <li>• ....</li> <li>• 0x7: Seven recovery cycles</li> </ul>
11:8	WRCV	RW	Write recovery cycles Value: <ul style="list-style-type: none"> <li>• 0x0*</li> </ul>
*: Reset value			
[1]: Do not attempt to write to EBI_CS_REC register while the external bus is being accessed.			

**EBI\_CS\_RECEN – CS Recovery Insertion Enable Register (Address Offset = 0x880)**

Bit	Name	Access	Description
0	RCVEN	RW	Separate bus recovery cycle insertion enable Value: <ul style="list-style-type: none"> <li>• 0x0*: Disabled</li> <li>• 0x1: Enabled</li> </ul>
*: Reset value			
[1]: Do not attempt to write to EBI_CS_RECEN register while the external bus is being accessed.			

# Glossary and References

---

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

---

## Glossary

**AHB**

Advanced High Performance Bus

**CRU**

Clock Reset Unit

**GPIO**

General Purpose Input/Output

**HW**

Hardware

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

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

**JTAG**

Joint Test Action Group

**R**

Read-only register

**RW**

Read-write register

**RW1C**

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

**PSRAM**

Pseudo Static Random Access Memory

**SRAM**

Static Random Access Memory

**SW**

Software

## References

1. Xilinx Kintex-7 data sheet:  
[https://www.xilinx.com/support/documentation/data\\_sheets/ds182\\_Kintex\\_7\\_Data\\_Sheet.pdf](https://www.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf)
2. MikroBUS Standard Specification:  
<https://download.mikroe.com/documents/standards/mikrobus/mikrobus-standard-specification-v200.pdf>
3. 8-input 8-bit ADC088S022: <http://www.ti.com/lit/ds/symlink/adc088s022.pdf>
4. FTDI Chip FT2232H Data sheet:  
[https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS\\_FT2232H.pdf](https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232H.pdf)
5. Dual I2S Stereo Audio Codec MAX9880A:  
<https://datasheets.maximintegrated.com/en/ds/MAX9880A.pdf>
6. *DesignWare® MetaWare Debugger User's Guide for ARC®*:  
[https://www.synopsys.com/dw/doc.php/arc\\_tools/dw\\_arc\\_metaware/dbgr\\_gd\\_arc.pdf](https://www.synopsys.com/dw/doc.php/arc_tools/dw_arc_metaware/dbgr_gd_arc.pdf)
7. DesignWare DW\_apb\_gpio Databook:  
[https://www.synopsys.com/dw/doc.php/ds/c/DW\\_apb\\_gpio\\_databook.pdf](https://www.synopsys.com/dw/doc.php/ds/c/DW_apb_gpio_databook.pdf)
8. DesignWare ARC EM Series Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DWC\\_ARC\\_EM4\\_Core/latest/doc/ARC\\_EM\\_Databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DWC_ARC_EM4_Core/latest/doc/ARC_EM_Databook.pdf)
9. *DesignWare ARCv2 ISA Programmer's Reference Manual for ARC EM Processors*:  
[https://www.synopsys.com/dw/doc.php/iip/DWC\\_ARC\\_EM4\\_Core/latest/doc/ARC\\_V2\\_ProgrammersReference.pdf](https://www.synopsys.com/dw/doc.php/iip/DWC_ARC_EM4_Core/latest/doc/ARC_V2_ProgrammersReference.pdf)
10. DesignWare DWC Mobile Storage Host Databook:  
[https://www.synopsys.com/dw/doc.php/ds/c/dwc\\_mobile\\_storage.pdf](https://www.synopsys.com/dw/doc.php/ds/c/dwc_mobile_storage.pdf)
11. DesignWare DW\_apb\_uart Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DW\\_apb\\_uart/latest/doc/DW\\_apb\\_uart\\_databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DW_apb_uart/latest/doc/DW_apb_uart_databook.pdf)

12. DesignWare DW\_apb\_ssi Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DW\\_apb\\_ssi/latest/doc/DW\\_apb\\_ssi\\_databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DW_apb_ssi/latest/doc/DW_apb_ssi_databook.pdf)
13. DesignWare DW\_apb\_timers Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DW\\_apb\\_timers/latest/doc/DW\\_apb\\_timers\\_databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DW_apb_timers/latest/doc/DW_apb_timers_databook.pdf)
14. DesignWare DWC\_trng\_nist Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DWC\\_trng\\_nist/latest/doc/DWC\\_trng\\_nist\\_sp800\\_90c\\_databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DWC_trng_nist/latest/doc/DWC_trng_nist_sp800_90c_databook.pdf)
15. DesignWare DW\_apb\_wdt Databook:  
[https://www.synopsys.com/dw/doc.php/iip/DW\\_apb\\_wdt/latest/doc/DW\\_apb\\_wdt\\_databook.pdf](https://www.synopsys.com/dw/doc.php/iip/DW_apb_wdt/latest/doc/DW_apb_wdt_databook.pdf)
16. DesignWare ARC Data Fusion IP Subsystem I/O Databook:  
[https://www.synopsys.com/dw/doc.php/iip/dwc\\_data\\_fusion\\_subsystem/latest/doc/DataFusionSubsystem\\_IO\\_Databook.pdf](https://www.synopsys.com/dw/doc.php/iip/dwc_data_fusion_subsystem/latest/doc/DataFusionSubsystem_IO_Databook.pdf)
17. DesignWare Smart Sensor and Control IP Subsystem I/O Databook:  
[https://www.synopsys.com/dw/doc.php/iip/dwc\\_sensor\\_subsystem/latest/doc/SensorSubsystem\\_IO\\_Databook.pdf](https://www.synopsys.com/dw/doc.php/iip/dwc_sensor_subsystem/latest/doc/SensorSubsystem_IO_Databook.pdf)
18. Redpine Signals RS9113-WiseConnect-API-Guide-v1.7.1:  
[www.redpinenetworks.com](http://www.redpinenetworks.com)
19. Xilinx AXI EMC:  
[https://www.xilinx.com/support/documentation/ip\\_documentation/axi\\_emc/v3\\_0/pg100-axi-emc.pdf](https://www.xilinx.com/support/documentation/ip_documentation/axi_emc/v3_0/pg100-axi-emc.pdf)