DesignWare ARC IoT Development Kit
User Guide

Version 5796-001 November 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 About the DesignWare® ARC® IoT Development Kit

The DesignWare® ARC® IoT Development Kit (ARC IoTDK) is a versatile platform that includes the necessary hardware and software to accelerate software development and debugging of sensor fusion, voice recognition, and face detection designs. The ARC IoTDK includes a silicon implementation of the ARC Data Fusion IP Subsystem as well as a rich set of peripherals commonly used in IoT designs such as Bluetooth, USB, an analog-to-digital converter (ADC), pulse width modulator (PWM), and onboard nine axis sensor.

The ARC IoTDK is supported by a robust ecosystem of development tools and software including the MetaWare Development Toolkit, which enables the development and debugging of highly optimized, high-density code. The embARC Open Software Platform gives developers online access to device drivers, application examples, and a suite of free and open-source software that speeds software development for ARC-based embedded systems.

2.2 Tool Requirements

The DesignWare® ARC® IoT Development Kit requires the following tool to be installed on your host:

- Digilent Adept software — Software driver for the Digilent JTAG-USB cable

And one of the following toolchains:

- MetaWare Development Toolkit — DesignWare ARC tools to run and debug applications on the DesignWare ARC processors.
- MetaWare Lite — A free downloadable version of the MetaWare Development Toolkit for developing applications 32 KB or smaller on Windows.
- GNU Toolchain for ARC Processors — An open-source development environment to run and debug applications for the DesignWare ARC processors. For more information on the GNU Toolchain for ARC Processors and Eclipse IDE for the GNU toolchain for ARC Processors, see: https://github.com/foss-for-synopsys-dwc-arc-processors/toolchain/releases

2.3 Target Applications

The following is the list of ARC IoTDK target applications:

- Evaluation of ARC EM9(D) processors
- Evaluation of the ARC Data Fusion IP Subsystem
- Application software development on ARC EM9(D) processors for the following domains:
  - IoT end nodes
  - Sensor Fusion
  - Low-power voice/audio

The ARC IoTDK is further extensible through available Arduino, MikroBus[1], and Pmod[3] connectors.

You can connect the following end-devices to the ARC IoTDK:

- Sensors
- Actuators
- Displays
- Buttons and Switches
- Communication devices

3 Getting Started

3.1 Package Content

The ARC IoTDK package contains the following items:

DesignWare ARC IoT Development Kit hardware board

Warning

The DesignWare® ARC® IoT Development Kit contains static-sensitive devices.

3.2 Setting up the ARC IoTDK

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

3.2.1 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 [2][2] 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.

3.2.2 Checking Default Board Settings

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

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

Note The ARC IoTDK is powered over USB. Note that the ARC IoTDK needs to be powered by an external power adapter if additional devices are connected to the extension interfaces.

3.2.3 Installing and Configuring PuTTY

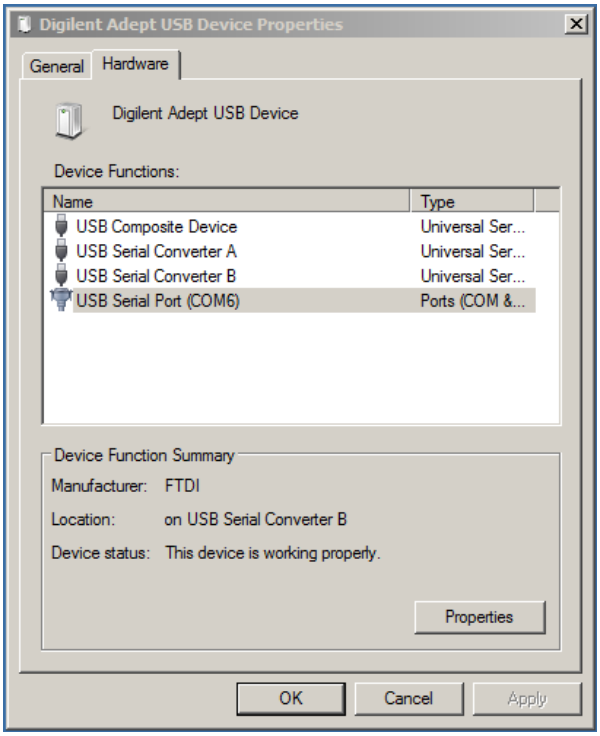
PuTTY is a serial console that can be used as a simple debug console.

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

   The example in Figure 1 uses the COM6 port:

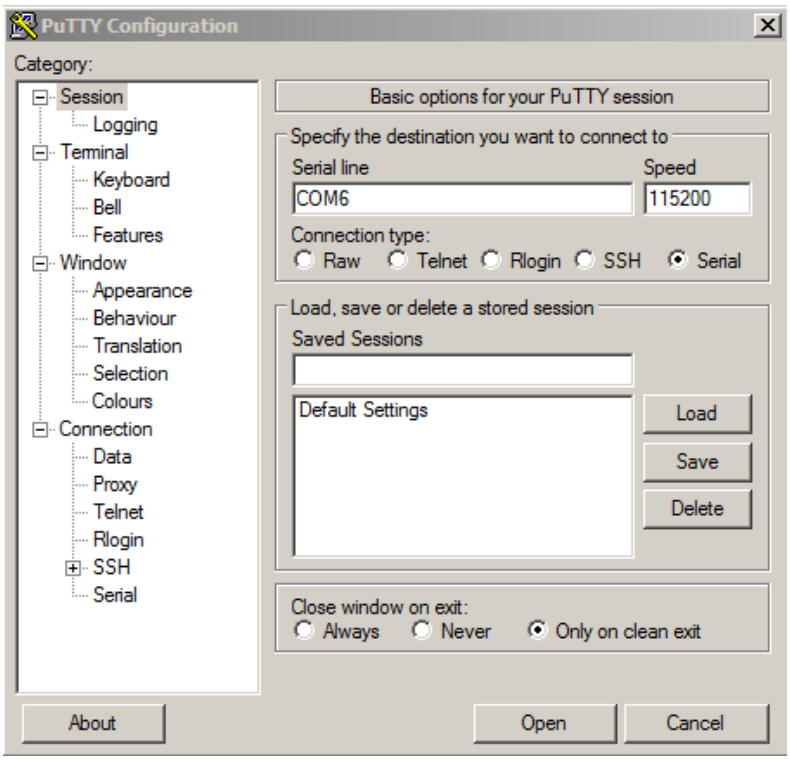
Figure 1 Identification of COM Port



5. Execute `putty.exe`.

   The **PuTTY Configuration** window appears.
6. Set the **Connection type** to **Serial**.
7. Enter the name of the COM port in the **Serial line** field.
8. Set the **Speed** field to **115200** as shown in Figure 2.

Figure 2 PuTTY Configuration



9. Click **Open** to start the PuTTY terminal.

3.2.4 Starting Uboot

By default, press the start button on the ARC IoTDK for the ARC EM core to start executing the bootloader. You see text as shown in Figure 3.

Figure 3 Default Boot

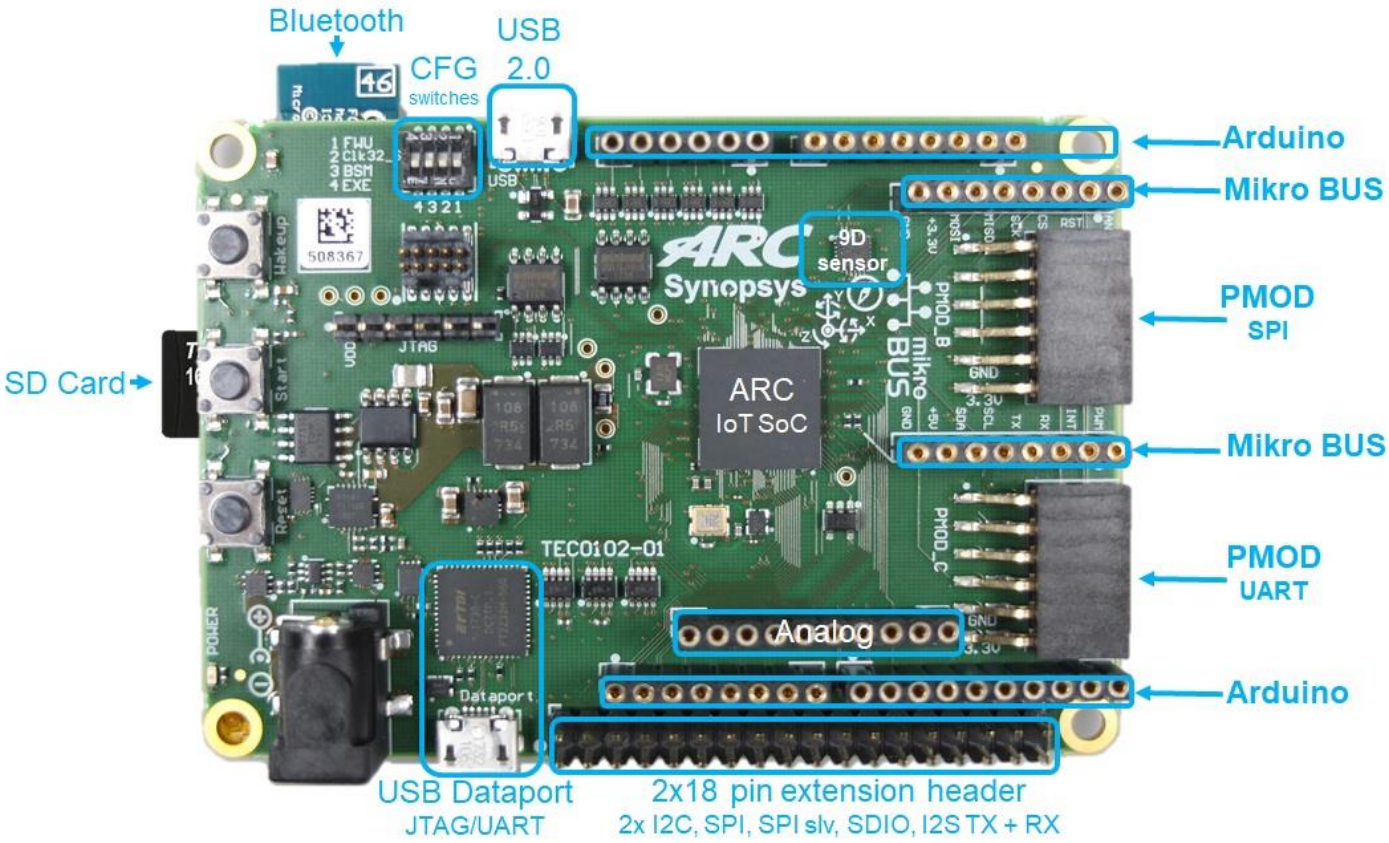
```
-----------------------------------------------------------------

   U-Boot 2018.11-rc1-iotdk-1.0 (Oct 10 2018 - 17:16:47 +0300)


   CPU:   ARC EM9D at 144 MHz
   Board: Synopsys IoT Development Kit
   DRAM:  128 KiB
   MMC:   Synopsys Mobile storage: 0
   Loading Environment from FAT... OK
   In:    serial0@80014000
```

```
    Out:    serial0@80014000
    Err:    serial0@80014000
    IoTDK#
-----------------------------------------------------------------
```

3.3 Location of Components On the ARC IoTDK

Figure 4 shows the placement of various components on the DesignWare® ARC® IoT Development Kit.

Figure 4 ARC IoTDK Components: Top View



3.4 Software Packages

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

Direct links to the software release are available here:

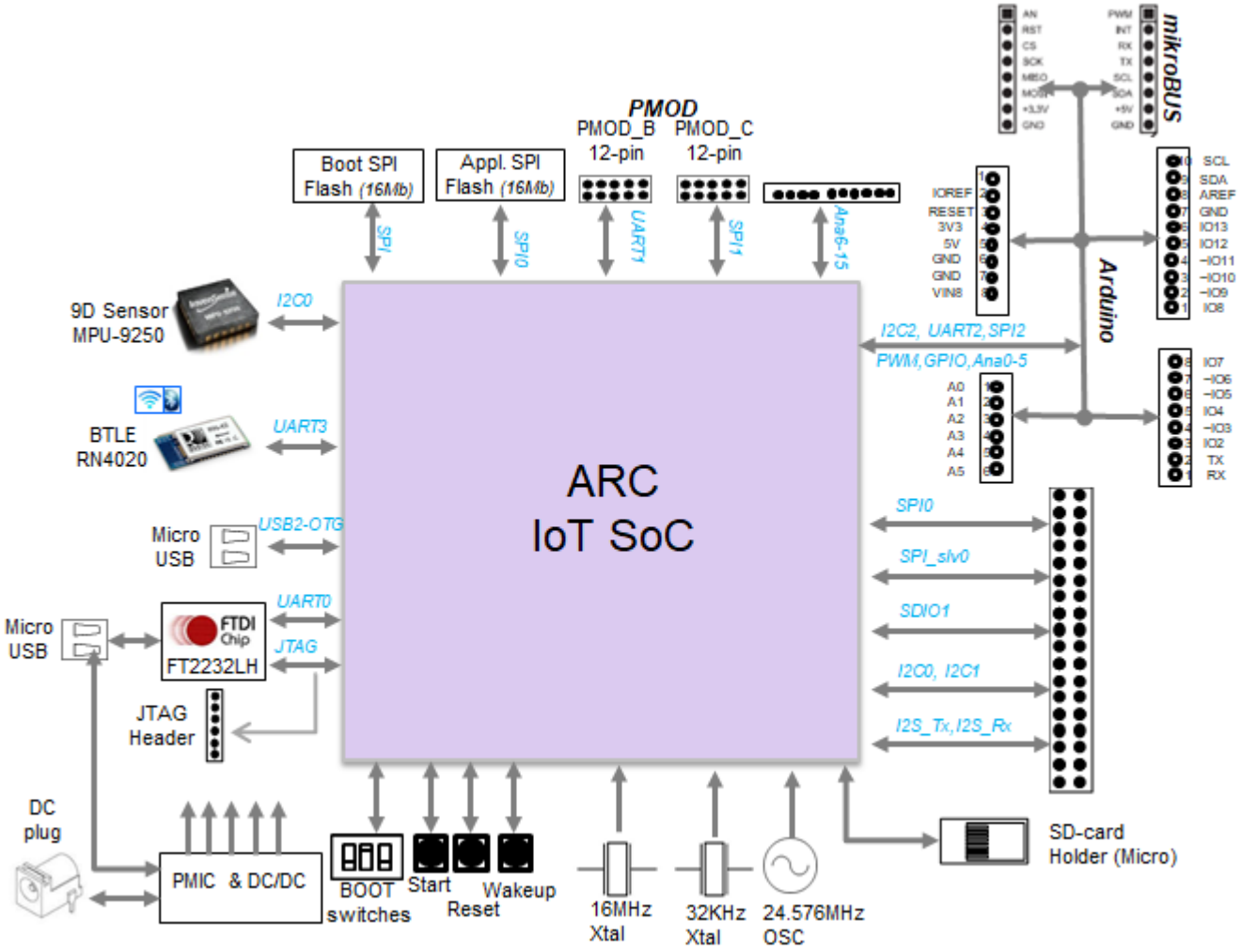
https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_osp/releases

For additional documentation on how to get started, see https://github.com/foss-for-synopsys-dwc-arc-processors/ARC-Development-Systems-Forum/wiki/ARC-Development-Systems-Forum-Wiki-Home

4 Hardware Description

4.1 Overview of ARC IoTDK

Figure 5 ARC IoTDK Block Diagram



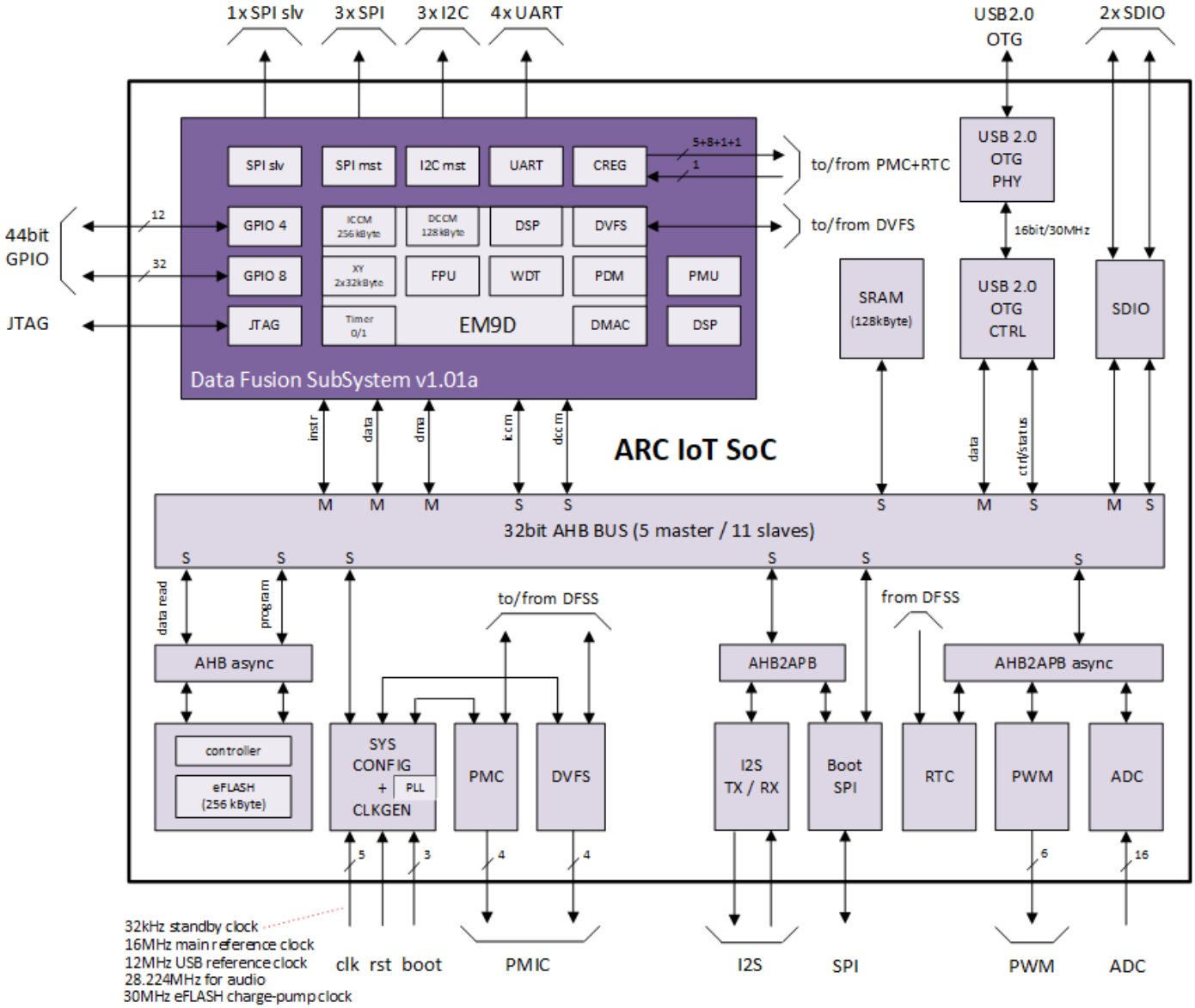
The DesignWare® ARC® IoT Development Kit contains the following components:

- ARC IoT SoC
  - EM9D based Data Fusion Subsystem
  - USB 2.0 OTG, 2x SDIO
  - eFLASH (256 KB), SRAM (128 KB)
  - Several APB Peripherals

- Memory
  - 2x SPI Flash (2 MB)
- Interfaces
  - USB 2.0 OTG
  - USB Data port (JTAG/UART)
  - microSD Card
  - 9D Sensor (Invensense MPU-9250)
  - BTLE module (Microchip RN-4020)
  - ADC (16 channels)
  - JTAG
  - PMIC with dynamic voltage control
- Extensions
  - Arduino Interface headers (UNO R3 compatible)
  - MikroBUS headers
  - Pmod Interfaces (2x)
  - Extension header (2x 18 pin)

4.2 Overview of ARC IoT SoC

Figure 6 ARC IoT SoC Top-Level Diagram



The ARC IoT SoC provides the following main features:

- DesignWare ARC Data Fusion Subsystem
  - EM9D Core at 144 MHz[1]
    - FPU
    - 256 KB ICCM
    - 128 KB DCCM

[1] 144 MHz at a temperature of 25 degrees celcius

        - 32 KB Xmemory
        - 32 KB Ymemory
        - Memory Protection Unit
        - DMA controller
    - DSP trigonometric accelerator functions
    - Advanced power management
        - DVFS
        - Clock and voltage switching. On-chip voltage switches for digital core logic supplies
        - Off-chip voltage switches for all other supplies and DFSS tightly coupled peripherals
    - 4 x UART
    - 3 x SPI master
    - 1 x SPI slave
    - 3 x I2C master
    - 44-bit GPIO (4 x 8-bit + 3 x 4-bit)
- 128 KB SRAM
- 256 KB eFLASH (50 MHz read operation)
- 2 MB Boot SPI
- Real Time Clock (RTC) running at a reference clock of 32.678 kHz
- I2S TX/RX interface
- SDIO interface (2x)
- USB 2.0 OTG interface
- JTAG interface for debug
- PWM interface
    - 6 PWM channels
    - 2 PWM timers (1 timer for channel 1/2/3 and 1 timer for channel 4/5/6)
- ADC interface, 16-channel single-ended 12-bit ADC

4.3 Clocks and Resets

4.3.1 Clocks

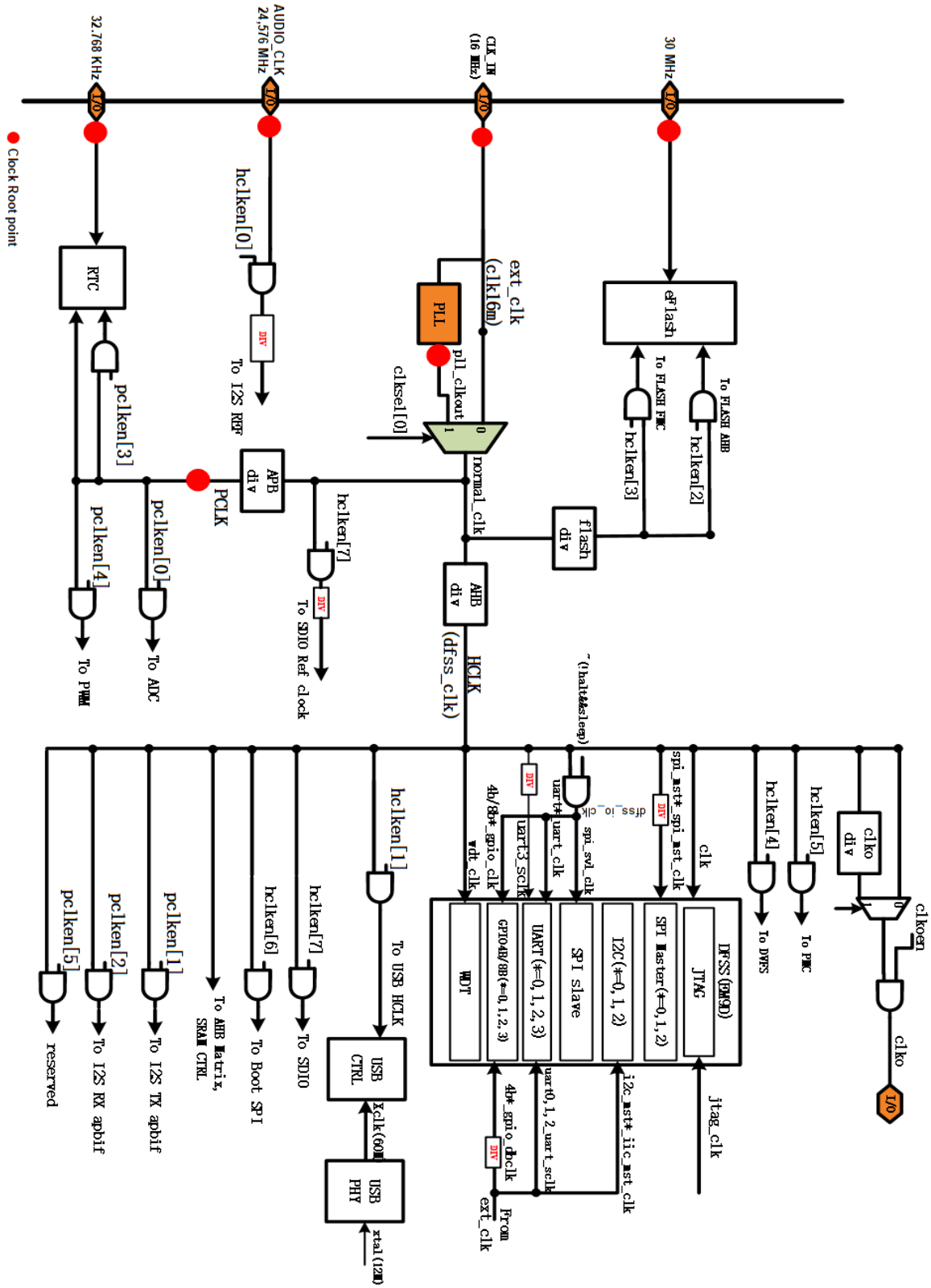
Figure 7 shows the top-level clock architecture of the ARC IoT SoC. The ARC IoT SoC uses a single 16 MHz reference clock from which all the main system clocks are generated. The clock generation is centralized in the SYSCONFIG module. The SYSCONFIG module implements a single PLL and integer dividers, which allows for accurate fine-tuning of the system clocks to the required frequency. See SYSCONFIG Control Registers for more details on the SYSCONFIG module.

Besides the 16MHz reference clock the ARC IoT SoC requires the following input clocks:

- 30 MHz charge-pump clock for eFLASH
- 12 MHz reference clock for USB
- 4.096 - 28.224 MHz reference clock for audio
- 32.768 kHz standby clock

An overview of the various system clock domains and their mapping onto the individual IP's is shown in Table 1.

Figure 7 ARC IoT SoC Clock Architecture



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

Table 1 Overview of ARC IoT SoC Clock Components

Clock Domain	Clock Name	Clock Type	Default Value (MHz)	Description
JTAG	jtag_clk	input	??	Input clock used to clock the ARC EM9D JTAG debug interface. JTAG tck can only run at maximum ½ freq of DFSS clk
32K	32k_clk	input	32.768 KHz	Input clock used for RTC
Audio	audio_clk	input	24.576	Input clock used as a reference for Audio I2S
	i2s_tx_clk	scalable	24.576	
	i2s_rx_clk	scalable	24.576	
CLK_12M	ext_clk_12M	input	12	Input Clock used to generate 60MHz USB PHY clock
CLK_30M	ext_clk_30M	input	30	Input clock used for eFlash
CLK_IN	ext_clk_16M	input	16	Input clock used to generate all the main system clocks, listed in the remaining part of this table
DFSS	dfss_core_clk (HCLK)	scalable	144	Clock used to clock DFSS (including the ARC EM9D).
	dfss_io_clk	scalable	144	Clock used to clock DFSS I/O peripherals: spi_slv0; uart0,1,2; gpio_4b0,1,2; gpio_8b0,1,2,3
	dfss_ahb_clk	scalable	144	Clock used to clock top-level AHB peripherals that run at the same clock as DFSS
	dfss_apb_clk	scalable	144	Clock used to clock top-level APB peripherals that run at the same clock as DFSS
APB	apb_clk (PCLK)	scalable	50	Clock used to clock all the asynchronous, top-level APB peripherals: ADC; RTC;
I2C	i2c_mst*_iic_mst_clk	fixed	16	Clock used to clock I2C master IP (mst0, mst1, mst2) and used as reference clock for the i2c baud-rate generation
SPI	spi_mst*_spi_mst_clk	fixed	--	Clock used to clock SPI master IP (mst0, mst1, mst2) and used as a reference clock for the SPI baud-rate generation
UART0,1,2	uart*_uart_sclk	fixed	16	Clock used as reference clock for the uart baud-rate generation: uart0,1,2
UART3	uart3_uart_sclk	scalable	--	Clock used as reference clock for the uart baud-rate generation: uart3
GPIO_DB	gpio_db_clk	scalable	16	Clock used to remove glitches with a period smaller than the debounce clock period
SDIO	sdio_ref_clk	fixed	--	Clock used to generate the sdio sample and drive clocks
FLASH	flash_clk	fixed	--	Clock used for flash host controller
PWM	pwm_timer_123_clk	scalable	--	Clock used to clock PWM timers 1 / 2 /3
	pwm_timer_456_clk	scalable	--	Clock used to clock PWM timers 4 / 5 / 6
USB PHY	usb_clk_60m	fixed	60	Clock used to transmit data through UTMI interface between USB-PHY and USB CTRL

4.4 Interrupts

All the interrupt sources outside the Data Fusion Subsystem (DFSS) are routed directly to the ARC EM9D interrupt controller without an additional top-level ICTL. The interrupt mapping for the ARC IoT SoC is listed in Table 2. All the interrupts are active-high. Each interrupt can be enabled/disabled individually, and sensitivity can be reprogrammed from level to edge. After reset, all interrupts are enabled and level-sensitive.

Table 2 Interrupt Mapping

IRQ#	Interrupt Source	Remarks
irq0	reset	
irq1	memory error	
irq2	instruction error	
irq16	timer0	
irq17	timer1	
irq18	Watchdog Timer	
irq19	io_gpio	gpio_4b0_gpio_intr_flag
irq20 : irq35	DMA controller	DMA_done[15:0]
irq36 : irq51	DMA controller	DMA_err[15:0]
irq52	io_gpio	gpio_4b1_gpio_intr_flag
irq53	io_gpio	gpio_4b2_gpio_intr_flag
irq54	io_gpio	gpio_8b0_gpio_intr_flag
irq55	io_gpio	gpio_8b1_gpio_intr_flag
irq56	io_gpio	gpio_8b2_gpio_intr_flag
irq57	io_gpio	gpio_8b3_gpio_intr_flag
irq58	i2c_mst0	i2c_mst0_iic_mst_intr_err
irq59	i2c_mst0	i2c_mst0_iic_mst_intr_rx_avail
irq60	i2c_mst0	i2c_mst0_iic_mst_intr_tx_req
irq61	i2c_mst0	i2c_mst0_iic_mst_intr_stop_det
irq62	i2c_mst1	i2c_mst1_iic_mst_intr_err
irq63	i2c_mst1	i2c_mst1_iic_mst_intr_rx_avail
irq64	i2c_mst1	i2c_mst1_iic_mst_intr_tx_req
irq65	i2c_mst1	i2c_mst1_iic_mst_intr_stop_det
irq66	i2c_mst2	i2c_mst2_iic_mst_intr_err
irq67	i2c_mst2	i2c_mst2_iic_mst_intr_rx_avail
irq68	i2c_mst2	i2c_mst2_iic_mst_intr_tx_req
irq69	i2c_mst2	i2c_mst2_iic_mst_intr_stop_det
irq70	spi_mst0	spi_mst0_spi_mst_err_intr
irq71	spi_mst0	spi_mst0_spi_mst_intr_rx_avail
irq72	spi_mst0	spi_mst0_spi_mst_intr_tx_req
irq73	spi_mst0	spi_mst0_spi_mst_intr_idle
irq74	spi_mst1	spi_mst1_spi_mst_err_intr
irq75	spi_mst1	spi_mst1_spi_mst_intr_rx_avail
irq76	spi_mst1	spi_mst1_spi_mst_intr_tx_req
irq77	spi_mst1	spi_mst1_spi_mst_intr_idle
irq78	spi_mst2	spi_mst2_spi_mst_err_intr
irq79	spi_mst2	spi_mst2_spi_mst_intr_rx_avail
irq80	spi_mst2	spi_mst2_spi_mst_intr_tx_req

IRQ#	Interrupt Source	Remarks
irq81	spi_mst2	spi_mst2_spi_mst_intr_idle
irq82	spi_slv0	spi_slv0_spi_slv_err_intr
irq83	spi_slv0	spi_slv0_spi_slv_intr_rx_avail
irq84	spi_slv0	spi_slv0_spi_slv_intr_tx_req
irq85	spi_slv0	spi_slv0_spi_slv_intr_idle
irq86	uart0	uart0_uart_intr
irq87	uart1	uart1_uart_intr
irq88	uart2	uart2_uart_intr
irq89	uart3	uart3_uart_intr
irq90	external	External wakeup interrupt
irq91	sdio	SDIO interrupt
irq92	i2s tx	I2S TX empty interrupt
irq93	i2s tx	I2S TX overrun interrupt
irq94	i2s rx	I2S RX data available interrupt
irq95	i2s rx	I2S RX overrun interrupt
irq96	USB	USB interface interrupt, Active low
irq97	adc_int	ADC convert channels finish interrupt
irq98	i_timer	i_timer interrupt[0]
irq99	i_timer	i_timer interrupt[1]
irq100	i_timer	i_timer interrupt[2]
irq101	i_timer	i_timer interrupt[3]
irq102	i_timer	i_timer interrupt[4]
irq103	i_timer	i_timer interrupt[5]
irq104	i_rtc	i_rtc interrupt
irq105	reserved	
irq106	reserved	
irq107	reserved	
irq108	reserved	
irq109	reserved	
irq110	reserved	

4.5 Debug

The ARC IoTDK offers several debug options:

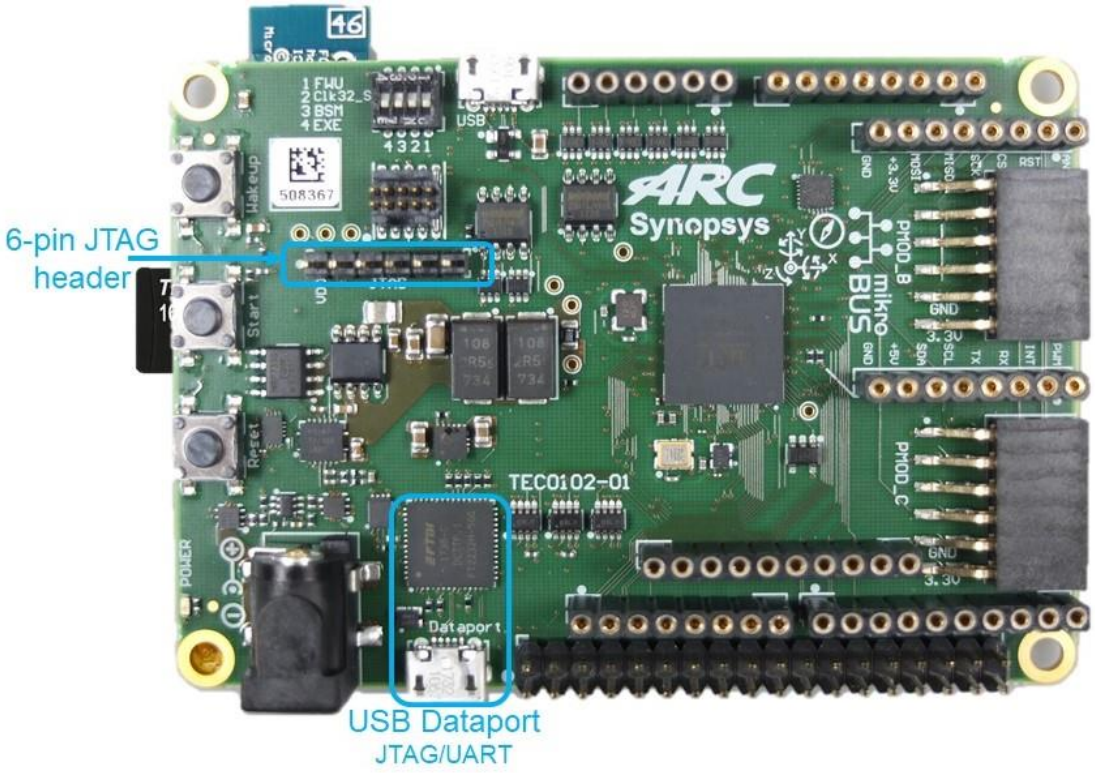
- USB cable connected to the data port
  - Compatible with MetaWare debugger and GDB
- Ashling Opella-XD, Lauterbach Trace-32, Digilent HS1 and HS2 support through:
  - A 6-pin 10 mil debug header and flying leads

Figure 8 shows the various debug headers.

Warning

The 6-pin header and USB data port are both directly connected to the JTAG port of the ARC EM9D Core. If the 6-pin connector is used for debugging, do not drive the JTAG port in parallel over the USB cable.

Figure 8 ARC IoTDK Debug Headers



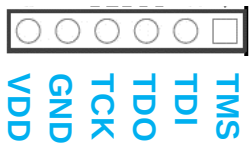
4.5.1 USB Dataport

The USB data port can be connected to your PC using a USB cable. 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 data port is compatible with the MetaWare debugger. The serial communication channel is used as console and can be monitored using a

standard hyperterminal application, for example PuTTY. For more information, see Getting Started.

4.5.2 6-Pin Header Pinout

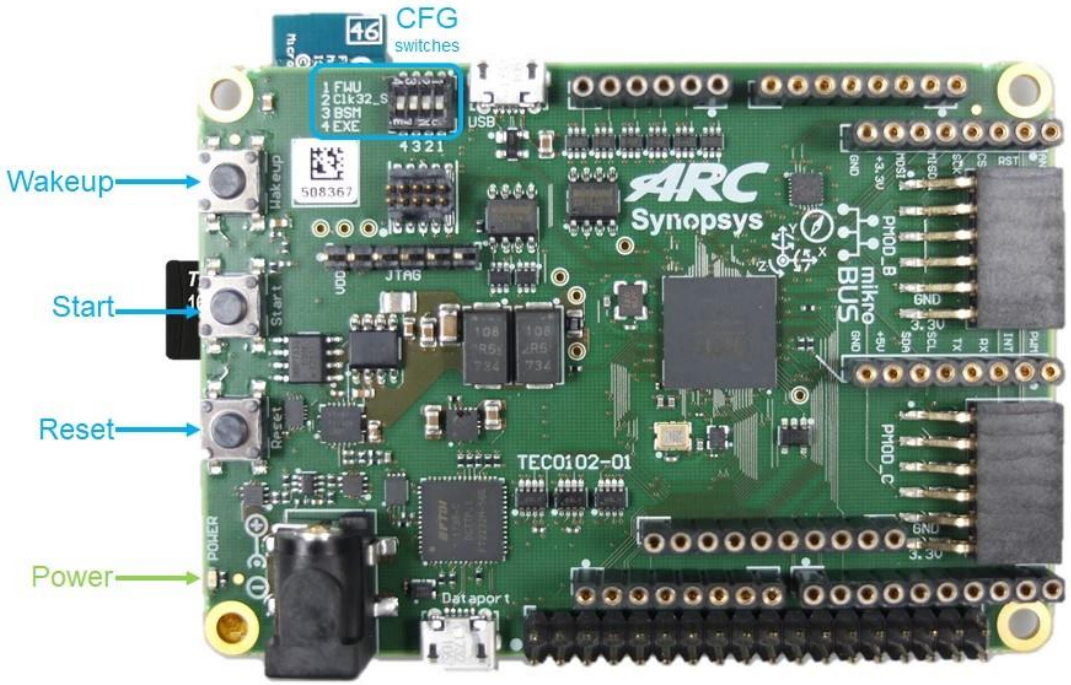


This 6-pin header is compatible with the standard Digilent HS1 and HS2 probes. The Ashling Opella XD and Lauterbach Trace-32 probes can be connected using flying leads or by using a 6-pin to 20-pin header converter.

4.6 Configuration Switches and Buttons

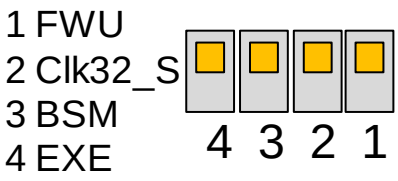
Figure 9 depicts the configuration switches, buttons, and power LED available on the ARC IoTDK. Besides the standard Reset and Start buttons there are also switches to control the board boot and operating mode. More detail of each of the switches is explained in the following sections.

Figure 9 ARC IoTDK Configuration Switches and Buttons



4.6.1 Configuration Switches

Figure 10 ARC IoTDK Configuration Switches



FWU switch

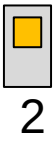
This switch is used for Firmware Updates

Ensure that this switch is set to “off” as depicted on the left (**default position**).

The “off” setting ensures that the EM9D Core starts executing the Uboot bootloader stored in the embedded FLASH. Setting this switch in the “on” position is only required if the Uboot bootloader residing in the embedded FLASH needs to be updated.

Clk32_S switch

This switch is used to select the 32.768 kHz clock source.



Ensure that this switch is set to “off” as depicted on the left (**default position**).

The “off” setting selects the onboard 32.768 KHz Oscillator. The “on” setting selects the 32.768 KHz Crystal (and corresponding Xtal I/O Pads).

BSM switch

This switch controls manual or automatic booting of the ARC IoTDK.

Ensure that this switch is set to “off” as depicted on the left (**default position**). The “off” position indicates *Manual* mode: the ARC IoTDK only starts booting after the *START* button is pushed. This is the **default setting.**

The “on” position selects *Automatic* mode: the ARC IoTDK automatically starts booting after Reset.

EXE switch

This switch controls if the EM9D Core continues with EXEcution or jumps into HALT mode.

Ensure that this switch is set to “off” as depicted on the left (**default position**). The “off” position selects EXEcution mode. The “on” position selects HALT mode.

In case the EM9D continues in EXEcution mode, depending on the FWU switch setting, the core either continues execution of the Uboot bootloader or starts executing the firmware update procedure.

4.6.2 Buttons

The ARC IoTDK includes 3 Buttons: Start, Reset, and Wakeup.

Start: Start the ARC Core if the BSM is in *Manual* Mode.

Reset: Reset the full board and ARC IoT SoC.

Wakeup: Wake up the core from the *Sleep* mode.

4.7 On-Board Memories

The ARC IoTDK features the following onboard memories:

- One 2 MB SPI Flash (U19); Flash type: W25Q16JVSNIQ.
    - One 2 Mbyte SPI Flash device (U19) is connected to the Boot SPI (See Figure 6). This flash is pre-loaded with a firmware upgrade application. Therefore, this flash device should not be used.
- One 2 MB SPI Flash (U35); Flash type: W25Q16JVSNIQ
    - One 2 Mbyte SPI Flash (U35) is connected to SPI Master0 (See Figure 5). This flash is fully available for storing user data.

Note The ARC IoT SoC also has a 256 Kbyte embedded Flash. Note that the Uboot bootloader is stored in this embedded Flash. Therefore, this embedded Flash is not available for storing user data.

4.8 USB Interface

The ARC IoTDK offers a USB 2.0 OTG port. The ARC IoT SoC includes a single DesignWare USB 2.0 OTG controller and a PHY from Brite Semiconductors. The controller supports high-speed (480 Mbps) transfers using an EHCI Host Controller, as well as full (12 Mbps) and low (1.5 Mbps) speeds through the integrated OHCI Host Controller. See references [4] and [5] for more info on the USB OTG controller and PHY.

4.9 SD Card Interface

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

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

After Uboot, the default settings operate the SD card in the SDR25 speed mode.

4.10 Audio Interface

The ARC IoTDK features stereo I2S [7] tx and rx ports. The TX and RX audio signals are available on the 2x18 pin general extension header. See 2x18 Pin Extension Header.

The I2S ports operate in master mode, which means that the I2S IP inside the ARC IoT SoC initializes and drives the I2S word select and serial clock signal. The I2S serial clocks are generated by integer dividers that run at a fixed audio reference clock frequency of 24.576 MHz.

The ARC IoTDK supports the following sampling frequencies: 16 kHz, 32 kHz, 48 kHz, 96 kHz, and 192 kHz. I2S SCLK divider settings are shown in Table 3.

Table 3 I2S SCLK Divider Settings

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

4.11 On-Board I2C Control Bus

The ARC IoTDK offers an onboard I2C bus [6] to control the following onboard device:

- MPU-2950 9D sensor (I2C address: 0xD0)

The onboard I2C bus is connected to I2C Master 0 and also routed to the 2x18 pin general purpose extension header.

4.12 ADC

The ARC IoTDK board includes a 16-channel single ended 12-bit ADC Interface.

The analog input range is 0 to 5 Volt. Channel AD0 – AD5 are available on the Arduino headers. Channels AD6 – AD15 are available on a dedicated 10-pin header (J2). Channel AD6 is also routed to the MikroBUS header AN signal. Table 4 lists the ADC channels and their usage.

Table 4 ADC Channel Usage

ADC IN	Usage
0	Arduino AD0
1	Arduino AD1

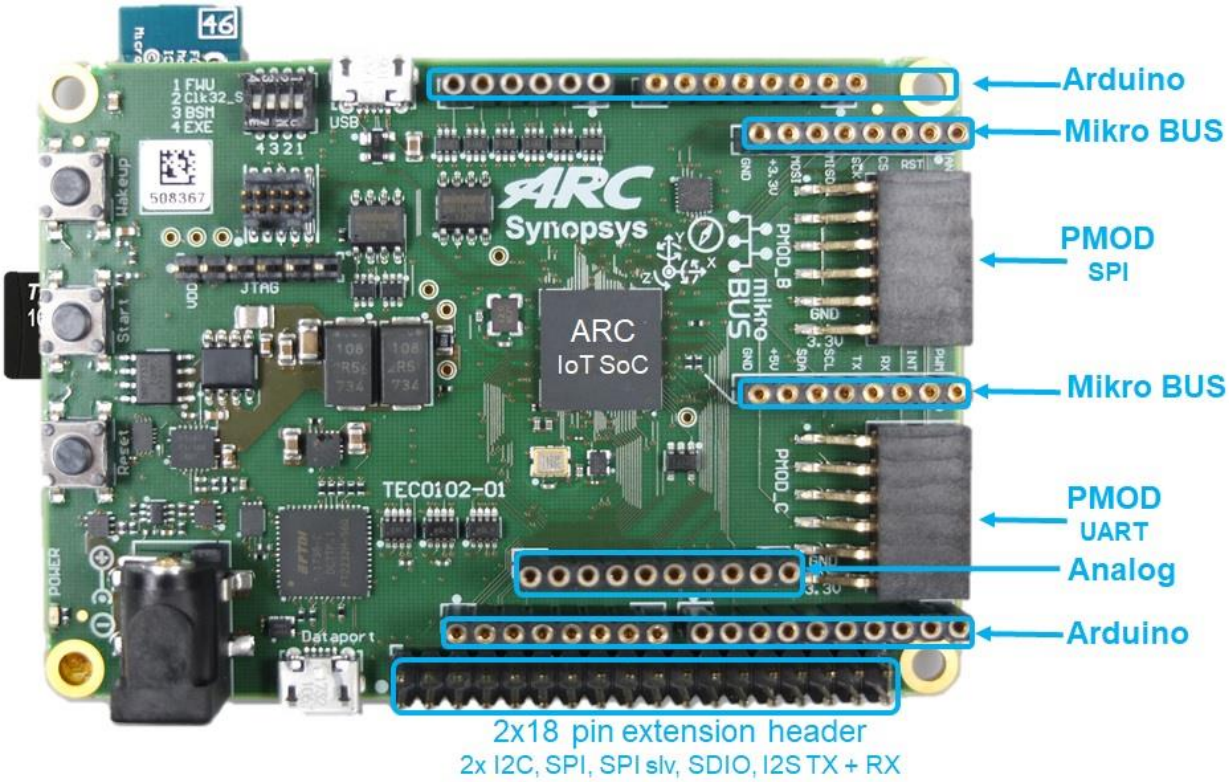
2	Arduino AD2
3	Arduino AD3
4	Arduino AD4
5	Arduino AD5
6	MikroBUS AN
6 - 15	Dedicated header (J2)

4.13 Extension Interfaces

To include peripheral hardware for your application, the following peripheral module standards are supported:

- Digilent Pmod™ (2x), see [3]
- MikroBUS (1x), see [1]
- Arduino (1x)

Figure 11 ARC IoTDK Extension Interfaces



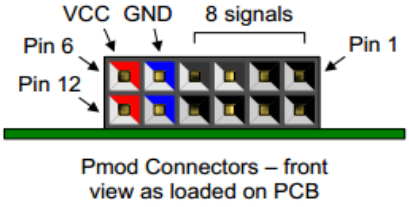
The ARC IoTDK features a 2x18 pin extension header that allows you to extend the ARC IoTDK with additional functionality. See section 2x18 Pin Extension for more details regarding this extension option.

4.13.1 Digilent Pmod™

The ARC IoTDK features two 12-pin Pmod connectors `Pmod_B` and `Pmod_C`. The functionality of the Pmod connectors is programmable and includes GPIO and SPI for `Pmod_B` and GPIO and UART for `Pmod_C`. After a reset, all ports are configured as GPIO inputs.

The location of the pins on the Pmod connectors is shown in Figure 12. Detailed pin descriptions depending on the pin multiplexer settings are provided in the subsequent sections.

Figure 12 Pinout Diagram of the Pmod_B and Pmod_C Connectors



4.13.1.1 Pmod_B Connector

Table 5 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 5 Pin Description of the Pmod_B Connector

Pin	GPIO	SPI
A1	gpio8b_0[0]	spi1_cs_n[0]
A2	gpio8b_0[1]	spi1_mosi
A3	gpio8b_0[2]	spi1_miso
A4	gpio8b_0[3]	spi1_clk
A5	GND	GND
A6	3V3	3V3
A7	gpio8b_0[4]	gpio8b_0[4]
A8	gpio8b_0[5]	gpio8b_0[5]
A9	n.c.	n.c.
A10	n.c.	n.c.
A11	GND	GND
A12	3V3	3V3

4.13.1.2 Pmod_C Connector

Table 6 lists the pin assignment of protocols that can be multiplexed on the `Pmod_C` connector. The **GPIO** column is the default assignment after reset.

Table 6 Pin Description of the Pmod_C Connector

Pin	GPIO	UART
B1	gpio8b_1[0]	uart1_cts
B2	gpio8b_1[1]	uart1_txd
B3	gpio8b_1[2]	uart1_rxd
B4	gpio8b_1[3]	uart1_rts
B5	GND	GND
B6	3V3	3V3
B7	gpio8b_1[4]	gpio8b_1[4]
B8	gpio8b_1[5]	gpio8b_1[5]
B9	n.c.	n.c.
B10	n.c.	n.c.
B11	GND	GND
B12	3V3	3V3

4.13.2 MikroBUS

The ARC IoTDK features a set of MikroBUS headers. Figure 13 shows the relevant function assignments, fully compatible with the MikroBUS standard [1]. The MikroBUS headers enable the addition of click boards. Click boards are developed by the company MikroElektronica (www.mikroe.com) and are 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 `ARDUINO_MUX` register (see Register Descriptions).

Figure 13 MikroBUS Headers

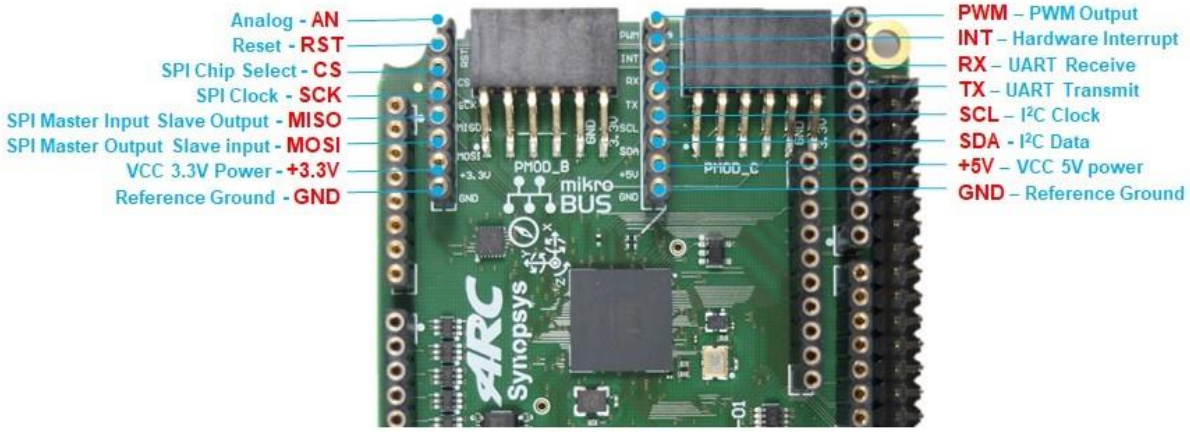


Table 7 shows the pin assignment on the I/O Multiplexer. The “MUX bitfield“ column indicates which bit in the `ARDUINO_MUX` register is used to control the functionality of this pin.

Table 7 Pin Description of the MikroBUS Connectors

Pin	I/O	MUX Bitfield	Pin	I/O	MUX Bitfield
AN	ADC IN6*	-	PWM	pwm0	Bit 2
RST	Reset_N	-	INT	gpio4b_2[2]	-
CS	spi2_cs	Bit 1	RX	uart2_rxd	Bit 0
SCK	spi2_clk	Bit 1	TX	uart2_txd	Bit 0
MISO	spi2_miso	Bit 1	SCL	i2c2_scl	Bit 8
MOSI	spi2_mosi	Bit 1	SDA	i2c2_sda	Bit 8

4.13.3 Arduino

The ARC IoTDK provides an Arduino shield interface. Figure 14 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 and the IOREF voltage on the ARC IoTDK board is fixed to 3V3. Note that 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 14 Arduino Shield Interface

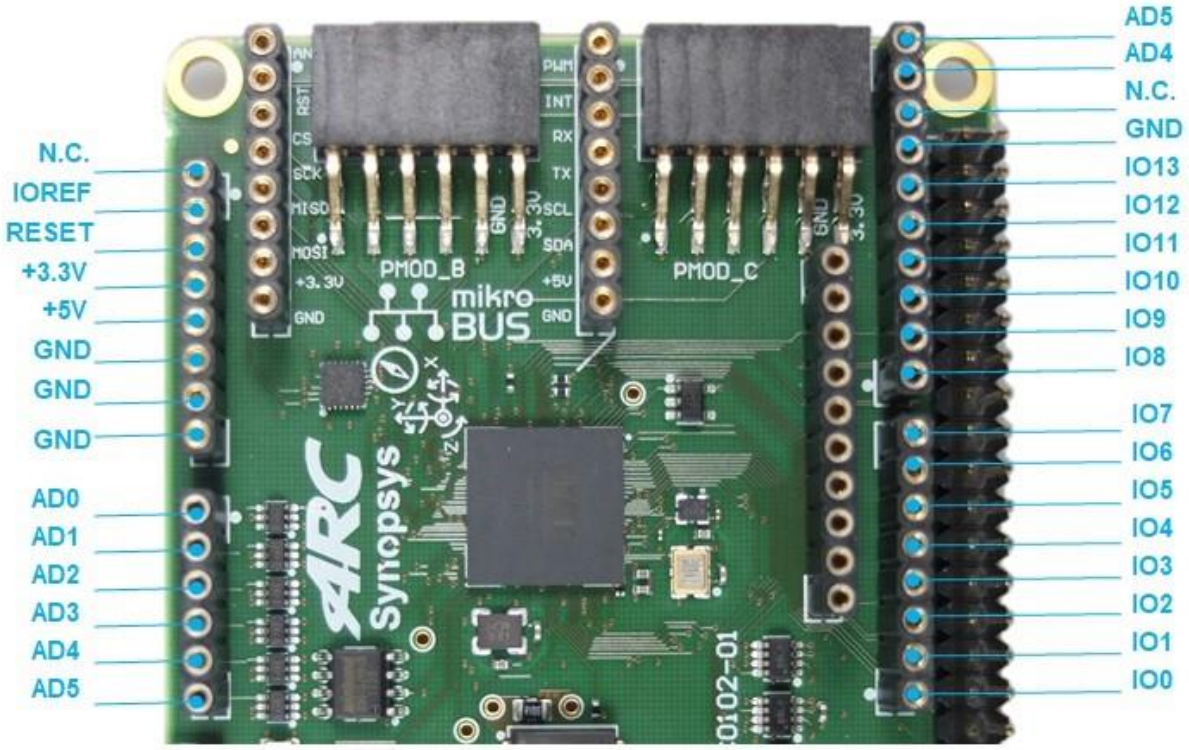


Table 8 shows the pin assignment on the I/O Multiplexer. Multiplexing is controlled by software using the `ARDUINO_MUX` register (see Register Descriptions). After a reset, all ports are configured as GPIO inputs. The “MUX bitfield“ column indicates which bit in the `ARDUINO_MUX` register is used to control the functionality of this pin.

Table 8 Pin Description of the Arduino Shield Interface

Pin	MUX bitfield	I/O-0	I/O-1	I/O-2
AD0	Bit 10	ADC IN0 gpio8b_3[7]	-	
AD1	Bit 11	ADC IN1 gpio8b_3[6]	-	
AD2	Bit 12	ADC IN2 gpio8b_3[5]	-	
AD3	Bit 13	ADC IN3 gpio8b_3[4]	-	
AD4	Bit 8/Bit 14	ADC IN4 gpio8b_3[3]	i2c2_sda	
AD5	Bit 8/Bit 15	ADC IN5 gpio8b_3[2]	i2c2_scl	
IO0	Bit 0	gpio4b_2[0]	uart2_rxd	-

IO1	Bit 0	gpio4b_2[1]	uart2_txd	-
IO2	-	gpio4b_2[2]	-	-
IO3	Bit 2	gpio4b_2[3]	-	pwm0
IO4	-	gpio8b_2[0]	-	
IO5	Bit 3	gpio8b_2[1]	-	pwm1
IO6	Bit 4	gpio8b_2[2]	-	pwm2
IO7	-	gpio8b_2[3]	-	-
IO8	-	gpio8b_2[4]	-	-
IO9	Bit 5	gpio8b_2[5]	-	pwm3
IO10	Bit 1/Bit 6	gpio8b_2[6]	spi2_cs_n	pwm4
IO11	Bit 1/Bit 7	gpio8b_2[7]	spi2_mosi	pwm5
IO12	Bit 1	gpio8b_3[0]	spi2_miso	gpio8b_3[0]
IO13	Bit 1	gpio8b_3[1]	spi2_clk	gpio8b_3[1]

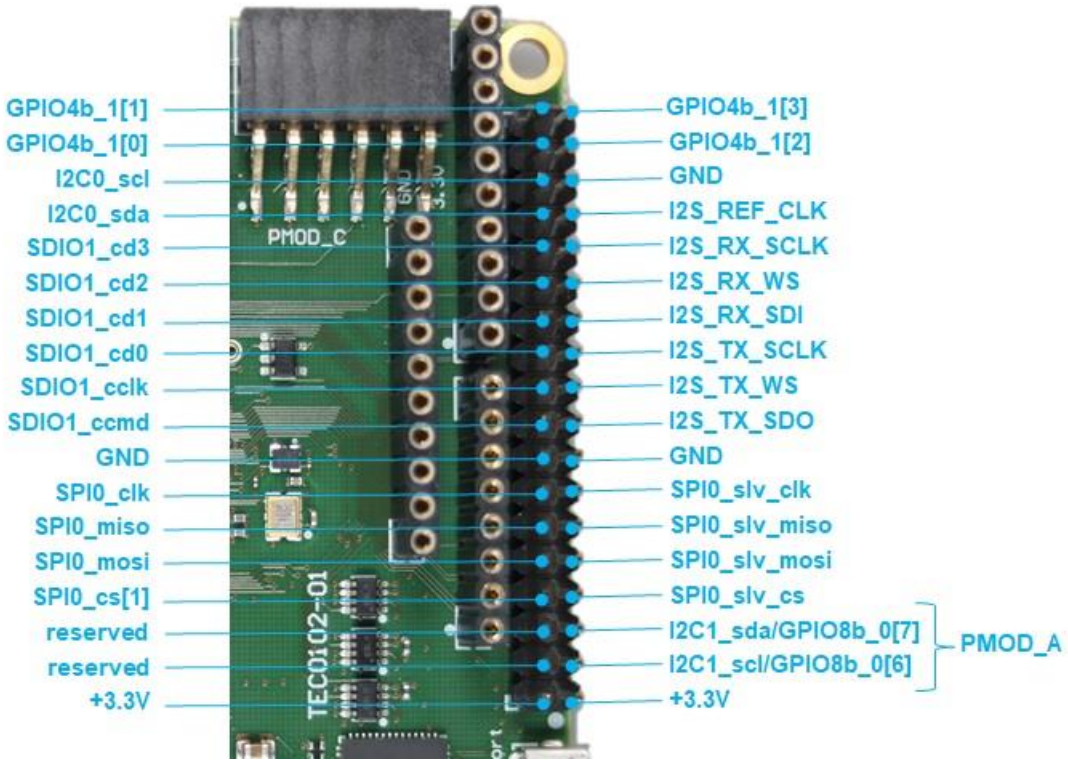
4.13.4 2x18 Pin Extension Header

The DesignWare ARC IoT Development Kit includes a 2 x 18 pin extension header.

- Carried signals:
  - PMOD_A (GPIO, I2C)
  - SPI0_slv
  - I2S TX and I2S RX
  - GPIO4b_1
  - I2C0
  - SDIO1
  - SPI0

A detailed overview of all signals can be found in Figure 15. The selection of the PMOD_A functionality is controlled by software using the `PMOD_MUX` register (see Register Descriptions).

Figure 15 2x 18 Pin Extension Header – Signal List


GPIO4b_1[1]
GPIO4b_1[0]
I2C0_scl
I2C0_sda
SDIO1_cd3
SDIO1_cd2
SDIO1_cd1
SDIO1_cd0
SDIO1_cclk
SDIO1_ccmd
GND
SPI0_clk
SPI0_miso
SPI0_mosi
SPI0_cs[1]
reserved
reserved
+3.3V
PMOD_C
GND
3.3V
TEC0102-01
GPIO4b_1[3]
GPIO4b_1[2]
GND
I2S_REF_CLK
I2S_RX_SCLK
I2S_RX_WS
I2S_RX_SDI
I2S_TX_SCLK
I2S_TX_WS
I2S_TX_SDO
GND
SPI0_slv_clk
SPI0_slv_miso
SPI0_slv_mosi
SPI0_slv_cs
I2C1_sda/GPIO8b_0[7]
I2C1_scl/GPIO8b_0[6]
+3.3V
PMOD_A

5 Programmer's Reference

5.1 Memory Map

Table 9 shows the ARC IoT SoC memory map overview.

Table 9 ARC IoT SoC Memory Map

Offset	Size	Slave
0x0000_0000	256 KB	eFLASH READ
0x1000_0000	2 MB	External BOOT SPI Flash
0x2000_0000	256 KB	ICCM0 RAM
0x3000_0000	128 KB	SRAM
0x8000_0000	128 KB	DCCM
0xC000_0000	32 KB	XCCM
0xE000_0000	32 KB	YCCM
0xF000_0000	256 Byte	I2S TX
0xF000_1000	256 Byte	I2S RX
0xF000_2000	N/A	Reserved
0xF000_3000	256 Byte	BOOT SPI CTRL
0xF000_4000	N/A	Reserved
0xF000_5000	256 Byte	RTC
0xF000_6000	256 Byte	PWM
0xF000_7000	128 Byte	ADC CTRL
0xF000_8000	N/A	Reserved
0xF000_9000	32 Byte	eFlash CTRL
0xF000_a000	144 Byte	SYS Configure
0xF000_b000	1 KB	SDIO CTRL
0xF004_0000	256 KB	USB CTRL

5.2 Software Interfaces

This section describes the software interfaces for the custom IP inside the ARC IoT SoC. The software interfaces for the DesignWare IP (for example, SDIO) are described in the corresponding databooks.

5.2.1 SYSCONFIG Control Registers

The global control registers inside the ARC IoT SoC are implemented by the SYSCONFIG module. The global control registers are used to control configuration settings for sub-modules that do not have a software interface. The SYSCONFIG module implements the following functionality:

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

Table 10 lists the registers for the SYSCONFIG module including a brief description and their offset to the base address (0xF000_a000). All registers are 32 bits wide. A detailed register description can be found in section Register Descriptions.

Table 10 SYSCONFIG Control Register Overview

Name	Address offset	Access*	Description
Control & Status registers			
AHBCLKDIV	0x04	RW	AHB clock divisor register
APBCLKDIV	0x08	RW	APB clock divisor register
APBCLKEN	0x0C	RW	APB module clock enable register
CLKODIV	0x10	RW	AHB clock output enable and divisor set register
RSTCON	0x18	WO	Reset control register
RSTSTAT	0x1C	RW	Reset status register
AHBCLKDIV_SEL	0x20	RW	AHB clock divider select register
CLKSEL	0x24	RW	Main clock source select register
PLLSTAT	0x28	RO	PLL status register
PLLCON	0x2C	RW	PLL control register
AHBCLKEN	0x34	RW	AHB module clock enable register
I2S_TX_SCLKDIV	0x40	RW	I2S TX SCLK divisor register
I2S_RX_SCLKDIV	0x44	RW	I2S RX SCLK divisor register
I2S_RX_SCLKSEL	0x48	RW	I2S RX SCLK source select register
SDIO_REFCLK_DIV	0x4C	RW	SDIO reference clock divisor register
GPIO4B_DBCLK_DIV	0x50	RW	GPIO4B DBCLK divisor register
SPI_MST_CLKDIV	0x5C	RW	SPI master clock divisor register
IMAGE_CHK	0x54	RO	Image pad status register

PROT_RANGE	0x58	RW	PROT range register
PMOD_MUX	0x70	RW	PMOD I/O multiplex
ARDUINO_MUX	0x74	RW	ARDUINO I/O multiplex
USBPHY_PLL	0x78	RW	USBPHY PLL control register
USBCFG	0x7C	RW	USB configuration register
TIMER_PAUSE	0x80	RW	PWM Timer pause register
GPIO8B_DBCLK_DIV	0x84	RW	GPIO8B DBCLK divisor register
UART3SCLK_DIV	0x8C	RW	Uart3 sclk divider

* The following access types are defined:
RW: Read/Write register
RO: Read-Only register
WO: Write-Only register

5.2.1.1 Register Descriptions

AHBCLKDIV register (address offset = 0x04)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	AHBCLKDIV[7:0]	RW		AHB clock divisor setting
			0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
11:8	FLASHCLKDIV[7:0]	RW	0 *	Flash controller clock divider
31:12	reserved	RO	0 *	

APBCLKDIV register (address offset = 0x08)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	APBCLKDIV[7:0]	RW		APB clock divisor setting
			0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
31:8	reserved	RO	0 *	

APBCLKEN register (address offset = 0x0C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	APBCLKEN[0]	RW		ADC controller clock enable bit
			0	Clock disabled
			1*	Clock enabled
1	APBCLKEN[1]	RW		I2S TX module clock enable bit
2	APBCLKEN[2]	RW		I2S RX module clock enable bit
3	APBCLKEN[3]	RW		RTC PCLK enable bit
4	APBCLKEN[4]	RW		PWM PCLK enable bit
5	APBCLKEN[5]	RW		reserved
31:6	reserved	RO	0 *	

CLKODIV register (address offset = 0x10)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7..0	CLKODIV[7:0]	RW		Clock monitor output divisor setting
			0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
8	CLKOEN	RW		Clock monitor output enable
			0 *	Clock monitor output disabled
			1	Monitor clock output enabled
31:9	reserved	RO	0 *	

AHBCLKDIV_SEL register (address offset = 0x20)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	AHBCLKDIV_SEL	RW		AHB clock divider selection
			0 *	AHB clock divider from DVFS & PMC
			1	AHB clock divider from AHBCLKDIV register
1	CLK_16M_ENA	RW		Enable / disable 16MHz oscillator
			0	powered-down
			1 *	enabled
2	XTALPD	RW		Enable / disable external 16MHz oscillator power
			0	powered-down
			1 *	enabled

3	Reserved	RO	0	
5:4	XTALDS	RW	10	16MHz oscillator Drive current source
7:6	Reserved	RO	0	
8	OSC32KEN	RW		Enable / disable 32K oscillator
			0	disable
			1*	enable
11:9	OSC32KDS	RW	100	32K oscillator Drive current source
12	OSC32kRST	RW		32K oscillator
			0	Normal mode
			1	32k oscillator restart (only need a pulse)

RSTCON register (address offset = 0x18)

*Legend: * reset value*

Bit	Name	Access	Value	Description
31:0	MCURSTN [31:0]	WO		Writing 0x55AA6699 to the register generates MCU reset.

RSTSTAT register (address offset = 0x1C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	reserved	RO	0 *	
1	MCURS	RW		
			0 *	Power on reset occurred
			1	MCU reset occurred
31:2	reserved	RO	0 *	

CLKSEL register (address offset = 0x24)

*Legend: * reset value*

Bit	Name	Access	Value	Description
1:0	CLKSEL	RW		Main clock source selection
			0 *	external 16MHz clock
			1	PLL clock
			2	external 32kHz clock
			3	illegal
31:2	reserved	RO	0 *	

PLLSTAT register (address offset = 0x28)

*Legend: * reset value*

Bit	Name	Access	Value	Description
1:0	reserved	RO	0 *	
2	PLLSTB	RO		PLL locked indicator
			0 *	PLL not locked
			1	PLL locked
3	PLLRDY	RO		PLL ready indicator
			0	PLL not ready
			1 *	PLL ready
31:4	reserved	RO	0 *	

PLLCON register (address offset = 0x2C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
3:0	N[3:0]	RW	1 *	PLL input divider (valid range: 1 … 15)
17:4	M[13:0]	RW	13 *	PLL feedback divider (valid range: 4 …16385)
19:18	reserved	RO	0 *	
21:20	OD[1:0]	RW		PLL output divider value NO = 2 ** OD
			0	div-by-1
			1 *	div-by-2
			2	div-by-4
			3	div-by-8
23:22	reserved	RO	0 *	
24	BP	RW		Bypass PLL
			0 *	Not bypass PLL
			1	Bypass PLL
25	reserved			
26	PLLRST	RW		PLL reset control
			0	PLL normal operation
			1 *	PLL reset
31:27	reserved	RO	0 *	
				Fout = Fin * M / (N * NO) ⇨ Fout = 16 * 13 / (1 * 2) = 1004MHz

AHBCLKEN register (address offset = 0x34)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	AHBCLKEN[0]	RW		I2S reference clock enable bit
			0	Clock disabled
			1 *	Clock enabled
1	AHBCLKEN[1]	RW	1 *	USB module clock enable bit
2	AHBCLKEN[2]	RW	1 *	FLASH module AHB clock enable bit
3	AHBCLKEN[3]	RW	1 *	FLASH FMC clock enable bit
4	AHBCLKEN[4]	RW	1 *	DVFS clock enable bit
5	AHBCLKEN[5]	RW	1 *	PMC clock enable bit
6	AHBCLKEN[6]	RW	1 *	Boot SPI module clock enable bit
7	AHBCLKEN[7]	RW	1 *	SDIO module clock enable bit
31:8	reserved	RO	0 *	

I2S_TX_SCLKDIV register (address offset = 0x40)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	I2STXSCLKDIV[7:0]	RW		I2S TX SCLK divisor setting
			0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
31:8	reserved	RO	0 *	

I2S_RX_SCLKDIV register (address offset = 0x44)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	I2SRXSCLKDIV[7:0]	RW		I2S RX SCLK divisor setting
			0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
31:8	reserved	RO	0 *	

I2S_RX_SCLKSEL register (address offset = 0x48)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	I2SRXSCLKSEL	RW		I2S RX SCLK source selection
			0 *	I2S RX SCLK divided output
			1	I2S TX SCLK divided output
31:1	reserved	RO	0*	

SDIO_REFCLK_DIV register (address offset = 0x4C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	SDIOREFCLKDIV [7:0]	RW		SDIO reference clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
31:8	reserved	RO	0*	

GPIO4B_DBCLK_DIV register (address offset = 0x50)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	GPIO4B0DBCLKDIV	RW		GPIO4B0 debounce clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
15:8	GPIO4B1DBCLKDIV	RW	0*	GPIO4B1 debounce clock divisor setting
23:16	GPIO4B2DBCLKDIV	RW	0*	GPIO4B2 debounce clock divisor setting
31:24	reserved	RO	0*	

SPI_MST_CLKDIV register (address offset = 0x5C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	SPI0MSTDIV	RW		SPI0 master clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
15:8	SPI1MSTDIV	RW		SPI1 master clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
23:16	SPI2MSTDIV	RW		SPI2 master clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
31:24	reserved	RO	0*	

IMAGE register (address offset = 0x54)

*Legend: * reset value*

Bit	Name	Access	Value	Description
1	IMAGE	RO		Execute application image yes/no? The value for IMAGE is sampled from the “image” pin and is used by the bootloader to determine whether an application image should be executed yes/no.
			0	no (that is ARC core enters HALT state after bootloader completed)
			1	yes

PROT_RANGE register (address offset = 0x58)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	PROT_RANGE	RW	0x0000 *	*Prot range register*

PMOD_MUX register (address offset = 0x70)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	PMOD_A	RW		Select function of PMOD_A
			0x0 *	0 = I2C
			0x1	1 = GPIO
1	PMOD_B	RW		Select function of PMOD_B
			0x0 *	0 = SPI
			0x1	1 = GPIO
2	PMOD_C	RW		Select function of PMOD_C
			0x0 *	0 = UART
			0x1	1 = GPIO
31:3	reserved	RO	0x0 *	

ARDUINO_MUX register (address offset = 0x74)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	UART	RW		Select function of D[1:0]
			0x0 *	0 = GPIO
			0x1	1 = UART
1	SPI	RW		Select function of D[13:10]
				0 = GPIO / PWM (defined by PWM5 and PWM4 respectively)
				1 = SPI
2	PWM0	RW		Select function of D[3]
			0x0 *	0 = GPIO
			0x1	1 = PWM
3	PWM1	RW		Select function of D[5]
			0x0 *	0 = GPIO
			0x1	1 = PWM
4	PWM2	RW		Select function of D[6]
			0x0 *	0 = GPIO
			0x1	1 = PWM
5	PWM3	RW		Select function of D[9]
			0x0 *	0 = GPIO
			0x1	1 = PWM
6	PWM4	RW		Select function of D[10] (only valid when SPI=0)
			0x0 *	0 = GPIO
			0x1	1 = PWM
7	PWM5	RW		Select function of D[11] (only valid when SPI=0)
			0x0 *	0 = GPIO

			0x1	1 = PWM
8	I2C	RW		Select function of A[5:4]
			0x0 *	0 = GPIO / ADC (defined by ADC5 and ADC4 respectively)
			0x1	1 = I2C
10	ADC0	RW		Select function of A[0]
			0x0 *	0 = GPIO
			0x1	1 = ADC
11	ADC1	RW		Select function of A[1]
			0x0 *	0 = GPIO
			0x1	1 = ADC
12	ADC2	RW		Select function of A[2]
			0x0 *	0 = GPIO
			0x1	1 = ADC
13	ADC3	RW		Select function of A[3]
			0x0 *	0 = GPIO
			0x1	1 = ADC
14	ADC4	RW		Select function of A[4] (only valid when I2C=0)
			0x0 *	0 = GPIO
			0x1	1 = ADC
15	ADC5	RW		Select function of A[5] (only valid when I2C=0)
			0x0 *	0 = GPIO
			0x1	1 = ADC
31:16	reserved	RO	0x0 *	

USBPHY CFG register (address offset = 0x78)

*Legend: * reset value 0x01830680*

Bit	Name	Access	Value	Description
3:0	M	RW		M of USBPHY PLL
9:4	N	RW		N of USBPHY PLL
11:10	C	RW		C of USBPHY PLL
12	FOCLOCK	RW		FOC lock of USBPHY
13	CLKSEL	RW		USBPHY PLL ref clock select
			0	XTAL of USBPHY
			1	EXT CLK of SYSTEM
14	HSDRV0	RW		HSDRV0
15	HSDRV1	RW		HSDRV1
16	USB RSTn	RW		USB reset both controller and usbphy
			0	Reset
			1	Normal active
17	USB SUSPEND	RW		USB suspend

			0	No suspended
			1	suspended
22:18	USB PHY Test MODE	RW		Test mode setting of USB PHY
23	USB PHY RSTn	RW		reset USB PHY
			0	Reset
			1	Normal active
24	USB PLL LOCK	R0		USB PLL LOCK bit
			0	unlocked
			1	locked
25	USB LBKERR	RO		USB PHY LBK Error bit
			0	No Error
			1	Error
31:26	Reserved	RO	0	

USBCFG register (address offset = 0x7C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	DBUSSEL	RW		UTMI databus width selection
			0	8bit width
			1	16bit width
1	TXBITSTUFF	RW		TX bit stuff enable
2	TXBITSTUFFH	RW		TX bit stuff enable High
3	TXEN	RW		TX Enable
4	TXDAT	RW		USBPHY TX data
5	TXSE0	RW		USBPHY TX SE0
6	FSLSMODE	RW		USBPHY FS/LS Serial mode
7	ADPPRBEN	RW		USBPHY ADPPRBEN
8	TESTRSTn	RW		USBPHY TEST reset
10:9	SCALEDOWN	RW		USB scale down mode
11	DBFLTRBP	RW		USB dbnce filter bypass
12	UTMISRP	RW		USB UMTISRP bvalid
31:13	Reserved	RO	0	

PWM Timer PAUSE register (address offset = 0x80)

*Legend: * reset value*

Bit	Name	Access	Value	Description
0	TIM0PS	RW		Timer 0 pause
			0	Run
			1	Pause
1	TIM1PS	RW		Timer 1 pause

			0	Run
			1	Pause
2	TIM2PS	RW		Timer 2 pause
			0	Run
			1	Pause
3	TIM3PS	RW		Timer 3 pause
			0	Run
			1	Pause
4	TIM4PS	RW		Timer 4 pause
			0	Run
			1	Pause
5	TIM5PS	RW		Timer 5 pause
			0	Run
			1	Pause
31:6	Reserved	RO	0	

GPIO8B_DBCLK_DIV register (address offset = 0x84)

*Legend: * reset value*

Bit	Name	Access	Value	Description
7:0	GPIO8B0DBCLKDIV	RW		GPIO8B0 debounce clock divisor setting
			0*	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255
15:8	GPIO8B1DBCLKDIV	RW	0*	GPIO8B1 debounce clock divisor setting
23:16	GPIO8B2DBCLKDIV	RW	0*	GPIO8B2 debounce clock divisor setting
31:24	GPIO8B3DBCLKDIV	RW	0*	GPIO8B3 debounce clock divisor setting

UART3SCLK_DIV register (address offset = 0x8C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
31:8	Reserved	RO	0	
7:0	UART3SCLK_DIV		0 *	divide-by-1
			1	divide-by-1
			2	divide-by-2
			255	divide-by-255

5.2.1.2 PLL Programming

The PLL inside SYSCONFIG can be reprogrammed to a different frequency. The procedure for reprogramming the PLL frequency is as follows:

1) Switch the main clock source to the external 16 MHz clock.
   - Write `0` to the CLKSEL bit in CLKSEL register
2) Update the PLL divider settings (N / M / OD / B) in PLLCON register.
3) Toggle the PLL reset for at least 10 ns.
   - write `1` to the PLLRST bit in the PLLCON register.
   - write `0` to the PLLRST bit in the PLLCON register.
4) Wait for the PLL lock (implement a 0.5 ms wait time or poll the PLLSTAT register).
5) Switch the main clock source to PLL.
   - Write `1` to CLKSEL bit in CLKSEL register.

The PLL output frequency is calculated as follows:

Fref = Fin / N

Fvco = Fref * M

Fout = Fvco / NO

with:

N = input divider value (1, 2, 3 … 15)
M = feedback divider value (4, 5, 6 … 16383)
NO = output divider value (1, 2, 4, or 8)

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

1 MHz ≤ Fref ≤ 50 MHz

200 MHz ≤ Fvco ≤ 400 MHz

Table 11 lists some PLL divider settings for an input clock frequency of 16 MHz.

Table 11 PLL Divider Settings

Fin (MHZ)	Fout (MHz)	PLL Divider Settings			PLLCON Register Settings		
		N	M	NO	R[3:0]	M[13:0]	OD[1:0]
16	100	1	25	4	1	25	2
16	200	1	25	2	1	25	1
16	400	1	25	1	1	25	0

5.2.2 ADC Control

Table 12 ADC Control Register Overview

Name	Address offset	Access	Description
ADCCON	0x0	RW	ADC control register
ADCSCAN	0x4	RW	ADC scan control register
ADCD0	0x8	RW	ADC channel 0 data register
ADCD1	0xC	RW	ADC channel 1 data register
ADCD2	0x10	RW	ADC channel 2 data register
ADCD3	0x14	RW	ADC channel 3 data register
ADCD4	0x18	RW	ADC channel 4 data register
ADCD5	0x1C	RW	ADC channel 5 data register
ADCD6	0x20	RW	ADC channel 6 data register
ADCD7	0x24	RW	ADC channel 7 data register
ADCD8	0x40	RW	ADC channel 8 data register
ADCD9	0x44	RW	ADC channel 9 data register
ADCD10	0x48	RW	ADC channel 10 data register
ADCD11	0x4C	RW	ADC channel 11 data register
ADCD12	0x50	RW	ADC channel 12 data register
ADCD13	0x54	RW	ADC channel 13 data register
ADCD14	0x58	RW	ADC channel 14 data register
ADCD15	0x5C	RW	ADC channel 15 data register
ADCIMSC	0x28	RW	ADC interrupt enable register
ADCRIS	0x2C	RW	ADC raw interrupt status register
ADCMIS	0x30	RW	ADC mask interrupt status register
ADCICLR	0x34	RW	ADC interrupt flag clear register

5.2.2.1 Register Descriptions

ADCCON register (address offset = 0x0)

*Legend: * reset value*

Bit	Name	Access	Value	Description
2~0	ADCFDIV	RW		FCLK divisor bits(The maximum clock frequency is 3.2 MHz). $F_{ADC} = P_{CLK} / 2^{ADCDIV}$ Sampling Rate = F_{ADC} / 16
3	ADCMS	RW		ADC mode
			0x0*	Single mode
			0x1	Continuous mode
4	ADCEN	RW		ADC enable bit

		0x0*	ADC disabled
		0x1	ADC enabled
31~5	Reserved		

ADCSCAN register (address offset = 0x04)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	ADC_CH_EN[15:0]	RW	0x0*	ADC channel enable bits
16	ADCST	RW		ADC start bit
			0x0*	ADC stop
			0x1	ADC start
31~17	Reserved			

5.2.2.2 ADCDx Registers

Following are the offsets for the ADCD registers.

ADCD0 register (address offset = 0x8)

ADCD1 register (address offset = 0xC)

ADCD2 register (address offset = 0x10)

ADCD3 register (address offset = 0x14)

ADCD4 register (address offset = 0x18)

ADCD5 register (address offset = 0x1C)

ADCD6 register (address offset = 0x20)

ADCD7 register (address offset = 0x24)

ADCD8 register (address offset = 0x40)

ADCD9 register (address offset = 0x44)

ADCD10 register (address offset = 0x48)

ADCD11 register (address offset = 0x4C)

ADCD12 register (address offset = 0x50)

ADCD13 register (address offset = 0x54)

ADCD14 register (address offset = 0x58)

ADCD15 register (address offset = 0x5C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
11~0	ADCD[11:0]	RW		ADC channel data register
31~12	Reserved			

ADCIMSC register (address offset = 0x28)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	IMSC[15:0]	RW	0x0*	ADC Channel interrupt enable bits, 0 disabled, 1 enabled
31~16	reserved			

ADCRIS register (address offset = 0x2C)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	RIS[15:0]	RW		Channel raw conversion interrupt flag
			0x0*	ADC conversion in progress or not started
			0x1	ADC conversion end
31~16	reserved			

ADCMIS register (address offset = 0x30)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	MIS[15:0]	RW		Channel mask conversion interrupt flag
			0x0*	No ADC conversion interrupt
			0x1	ADC conversion interrupt occur
31~16	reserved			

ADCICLR register (address offset = 0x34)

*Legend: * reset value*

Bit	Name	Access	Value	Description
15~0	ICLR[15:0]	RW		Channel raw conversion interrupt flag clear, write 1 to ICLR bit clears the raw conversion interrupt flag.
31~16	reserved			

6 References

[1] *MikroBUS™ Standard Specification,* https://download.mikroe.com/documents/standards/mikrobus/mikrobus-standard-specification-v200.pdf

[2] DesignWare® MetaWare Debugger User's Guide for ARC®

[3] Digilent PmodTM Interface Specification, https://www.digilentinc.com/Pmods/Digilent-Pmod_Interface_Specification.pdf

[4] DesignWare Cores USB 2.0 OTG AHB Controller Databook

[5] Brite Semiconductors S55LLUSB20_PHY Datasheet

[6] DesignWare DW_apb_i2c Databook

[7] DesignWare DW_apb_i2s Databook