

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*



ARC EM Starter Kit
User Guide

Version 6280-016 March 2017

Copyright Notice and Proprietary Information Notice

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

Destination Control Statement

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

Disclaimer

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

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at

http://www.synopsys.com/Company/Pages/Trademarks.aspx.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

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

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

Contents

1 Preface 9
Document Structure 9
2 Introduction 10
About the ARC EM Starter Kit 10
Target Applications 10
Tool Requirements 11
Platform Requirements 11
3 Getting Started 12
Package Contents 12
Set up the ARC EM Starter Kit 13
Default Board Settings 13
Board Jumper Settings 13
Default ARC EM Configuration 13
Connect the USB Cable 14
USB Driver Installation 14
Connect the Power Supply 15
Run the Bootloader and Initial Self-Test 16
Bootloader and Initial Self-Test 16
View Self-Test Output on PuTTY Console 17
4 Working with ARC EM Starter Kit 21
Selecting ARC EM Configurations 21
Connecting External Interfaces to the ARC EM Starter Kit 23
Connecting the PmodAD2 Extension Module 24
5 Obtaining embARC Software and Documentation 26
About embARC Software and Documentation 26
6 Creating Applications Using the MetaWare Tools 27
Using MetaWare or MetaWare Lite 27
Using MetaWare 27
Using MetaWare Lite 27
Creating a Simple Application Using the Command-Line Tools 28
Debugging Applications Using the MetaWare Debugger 29

Creating Application Using the MetaWare IDE 30
Create a New Project in the MetaWare IDE 30
Invoking the Debugger and Running the Executable 34
7 Running Applications in Self-Boot Mode 38
Self-Boot Application Concept 38
Building a Self-Boot Application 38
Write the Self-Boot Application Image to SPI Flash 39
Running the Self-Boot Application 39
SPI Flash Memory Structure 40
Appendix: A Hardware Functional Description 41
Board Overview 41
FPGA Design Overview 43
External Hardware Interfaces 43
Connecting Peripheral Controllers 45
Pmod Pin Configuration 46
PMOD_MUX_CTRL Register 46
Pmod1 Configuration 47
UART_MAP_CTRL Register 47
Pmod2 Configuration 49
Pmod3 Configuration 50
Pmod4 Configuration 51
Pmod5 Configuration 52
Pmod6 Configuration 53
Pmod7 Configuration 54
Pmods Configuration Summary 55
Headers J10, J11, J12 57
Peripheral Controllers 59
GPIO 59
I2C 60
SPI Master 61
SPI Slave 64
UART 65
CREG Controller 66

Peripheral Memory Mapping 69
Interrupts Connections 69
On-Board Devices 70
JTAG Connector 70
USB 70
SD Card 71
Man-Machine Interface 71
Power Supply 73
Troubleshooting 73
Appendix: B ARC EM Configurations 75
Programmer's Model 75
Core Configurations and Memory Mapping 75
ARC_EM7D Configuration 76
ARC_EM9D Configuration 78
ARC_EM11D Configuration 80
Detailed Core Configurations 82
Appendix: C FPGA Image Recovery 89
Board Jumper Settings 89
Xilinx Lab Tools Installation 90
Connecting the FPGA Programming Cable 90
FPGA Programming Sequence 91
SPI Flash-Programming Sequence 95
Appendix: D Using a JTAG Debugger 100
Ashling Opella-XD Debugger 100
Lauterbach TRACE32 Debugger 101
Lauterbach TRACE32 Tool Installation 102
Glossary and References 103
Glossary 103
References 104

List of Figures

Figure 1 Board Jumper and Switch Locations....13
Figure 2 Power Supply and USB Cable Connection to the Board....15
Figure 3 COM6 Port....18
Figure 4 PuTTY Configuration Window....19
Figure 5 Self-test Results in a Terminal Window, ARC_EM7D....20
Figure 6 SW1 DIP Switch....22
Figure 7 Peripheral Device Connected Using Pmod....24
Figure 8 PmodAD2 Connection to Pmod2....25
Figure 9 MetaWare Settings Required to Use ARC JTAG through USB....29
Figure 10 MetaWare IDE - Creating a New Project....30
Figure 11 MetaWare IDE - Select Project Configurations....31
Figure 12 MetaWare IDE - Select TCF File....32
Figure 13 MetaWare IDE - Create New Source File....32
Figure 14 MetaWare IDE - Console Output for Test Project....33
Figure 15 MetaWare IDE - Creating New Debug Configuration....34
Figure 16 MetaWare IDE - Configure Debugger....35
Figure 17 MetaWare IDE - Debug Perspective....36
Figure 18 MetaWare IDE - Console window....37
Figure 19 Typical Flash Memory Structure....40
Figure 20 FGPA Design Block Diagram....43
Figure 21 Pmod Pin Numbering....46
Figure 22 Pin Positions at Headers J10, J11, and J12....57
Figure 23 DW I2C Connection....60
Figure 24 DW SPI Master Connection....63
Figure 25 DW UART Components Connection....65
Figure 26 Memory Map of ARC_EM7D Configuration....77
Figure 27 Memory Map of ARC_EM9D Configuration....79
Figure 28 Memory Map of ARC_EM11D Configuration....81
Figure 29 USB FPGA Programming Cable Connection to the Board....91
Figure 30 iMPACT Tool – Main Window View....92
Figure 31 iMPACT Tool – Initialize Chain Menu Option....93

Figure 32 iMPACT Tool – Assign New Configuration File Dialog Box 94
Figure 33 iMPACT View of the Successful FPGA Programming 95
Figure 34 iMPACT Tool – select SPI device 96
Figure 35 iMPACT Tool – Add PROM File Dialog Box 97
Figure 36 iMPACT Tool – Select Attached SPI/BPI Dialog Box 98
Figure 37 iMPACT View of the Successful SPI ROM Programming 99
Figure 38 Ashling XD Pod with Debug Cable 100
Figure 39 ARC JTAG Connection to the Board 101
Figure 40 Lauterbach TRACE32 with Debug Cable 102

List of Tables

Table 1 Predefined ARC Configurations 21
Table 2 Selecting the Configuration 23
Table 3 Peripheral Devices Connection to Pmods 44
Table 4 Board Connections Overview of Peripheral Controllers 45
Table 5 Pin Assignment of Pmod1 (J1) Upper Row Depending on PM1[0] 47
Table 6 Pin Assignment of Pmod1 (J1) Lower Row Depending on PM1[2] 49
Table 7 Pin Assignment of Pmod2 (J2) Depending on PM2[0] 49
Table 8 Pin Assignment of Pmod3 (J3) Depending on PM3[0] 50
Table 9 Pin Assignment of Pmod4 (J4) Depending on PM4[0] 51
Table 10 Pin Assignment of Pmod5 (J5) Upper Row Depending on PM5[0] 52
Table 11 Pin Assignment of Pmod5 (J5) Lower Row Depending on PM5[2] 52
Table 12 Pin Assignment of Pmod6 (J6) Upper Row Depending on PM6[0] 53
Table 13 Pin Assignment of Pmod6 (J6) Lower Row Depending on PM6[2] 54
Table 14 Pin Assignment of Pmod7 (J7) Depending on PM7[0] 54
Table 15 Pmods Configuration Summary 55
Table 16 Pin Assignment of the Header J10 Depending on PM2[0] 57
Table 17 Pin Assignment of the Header J11 Depending on PM3[0] 58
Table 18 Pin Assignment of the Header J12 Depending on PM4[0] 58
Table 19 SPI Master Signals Usage 62
Table 20 SPI_MST_CS_CTRL Register Bits Definition 62
Table 21 DMA Channel Assignment 64
Table 22 Register File Mapping 67
Table 23 Peripheral Memory Mapping 69
Table 24 Interrupts Connections 69
Table 25 Configuration Details 82
Table 26 Board Jumper Settings 89
Table 27 ARC EM Starter Kit JTAG Connector Pin-out 100

1 Preface

This chapter outlines the document structure and provides a brief chapter overview. Additionally, it contains a list of available resources and other supporting documents.

Document Structure

This document consists of the following chapters:

- Preface – Overview of the available documentation and its organization.
- Introduction – Overview of the ARC EM Starter Kit, target applications, and the tool requirements.
- Getting Started – Software required for the ARC EM Starter Kit and step-by-step guide on how to connect it to a power supply and to a debugger.
- Working with ARC EM Starter Kit – Procedures about how to configure the ARC EM Starter Kit.
- Obtaining embARC Software and Documentation – This section describes the embARC demo packages used for ARC EM Starter Kit.
- Creating Applications Using the MetaWare Tools – Procedures to create and debug software applications using the MetaWare Tools.
- Running Applications in Self-Boot Mode – Procedures to run applications in self-boot mode.
- Hardware Functional Description – Information about the hardware used in the ARC EM Starter Kit.
- ARC EM Configurations – Information about the ARC EM configurations packaged with the ARC EM Starter Kit.
- FPGA Image Recovery – Information about how to program the FPGA.
- Using a JTAG Debugger – Information about using a JTAG debugger.
- Glossary and References – List of abbreviations, terms, and external references.

2 Introduction

This document describes the ARC EM Starter Kit and procedures to build and run applications on the ARC EM Starter Kit.

About the ARC EM Starter Kit

The EM Starter Kit enables rapid software development, code porting, software debugging, and profiling for ARC EM processors.

The ARC EM Starter Kit consists of a hardware platform, including pre-installed FPGA images of ARC EM7D, ARC EM9D, and ARC EM11D configurations with peripherals. The ARC EM4, ARC EM5D and ARC EM6 processors are subsets of ARC EM7D processor. As such, applications can be built for ARC EM4, ARC EM5 or ARC EM6 processors and run on the ARC EM7D processor configuration on the starter kit.

The development board is based on a Xilinx Spartan®-6 LX150 FPGA and supports hardware extensions using six 2x6 connectors supporting a total of 48 user I/O pins (plus power and ground pins) that can be used to connect components such as sensors, actuators, memories, displays, buttons, switches, and communication devices. A Digilent Pmod™ compatible extension board containing a four-channel 12-bit A/D converter with an I2C interface and an AC power adapter is included in the package.

Target Applications

The ARC EM Starter Kit is targeted at the following applications:

- Evaluation of ARC EM processors
- Application software development on ARC EM processors
- Performance analysis of ARC EM processors

You can connect the following end-devices to the ARC EM Starter Kit:

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

Tool Requirements

The ARC EM Starter Kit requires the following tools to be installed on your host:

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

  Note

  Make sure that you are using latest version of Adept driver. For minimal version of Adept software, see *DesignWare ARC EM Starter Kit 2.2 Release Notes*.

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 K 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 GNU toolchain for ARC Processors, see:
  - https://github.com/foss-for-synopsys-dwc-arc-processors/toolchain/releases

Platform Requirements

For a list of platforms supported by ARC EM Starter Kit, see the *DesignWare ARC EM Starter Kit Release Notes*.

3
Getting Started

This chapter describes the software required for the ARC EM Starter Kit. It also contains a step-by-step guide on how to connect the ARC EM Starter Kit to a power supply and to a debugger.

Use the following procedure to set up the ARC EM Starter Kit and perform the self-test

1. Unpack the Package Contents
2. Set up the ARC EM Starter Kit
3. Connect the USB Cable
4. Connect the Power Supply
5. Run the Bootloader and Initial Self-Test

Package Contents

The ARC EM Starter Kit package contains the following items:

- FPGA module with Xilinx Spartan-6 FPGA device mounted on base board with extension connectors and peripherals
- 4-channel 12-bit A/D converter extension module
- 100-240V AC power adapter
- USB cable

Next Steps:

- Set up the ARC EM Starter Kit.

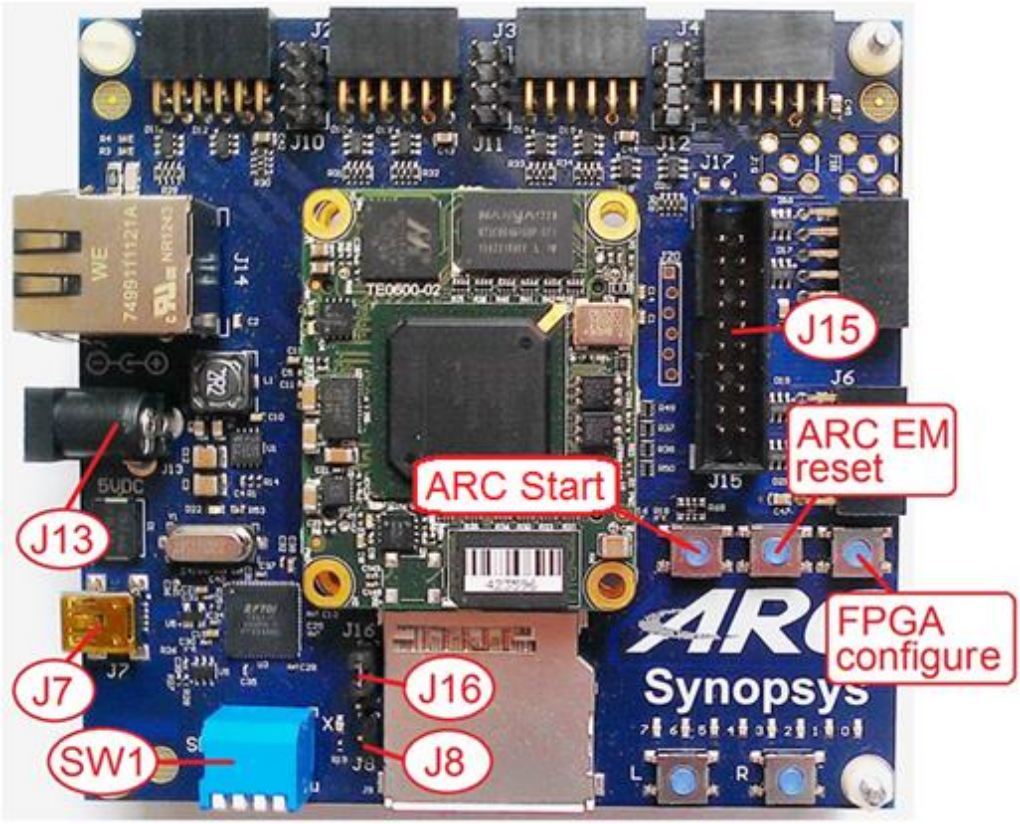
Set up the ARC EM Starter Kit

The following sections explain the procedures to connect the ARC EM Starter Kit to your host.

Default Board Settings

This section describes default settings for links and switches on the board.

Figure 1 Board Jumper and Switch Locations



Board Jumper Settings

The ARC EM Starter Kit hardware contains pre-installed FPGA configurations that are ready for use.

Default ARC EM Configuration

The FPGA board includes an SPI flash storage device pre-programmed with three FPGA images containing different configurations of DesignWare® ARC EM cores. ARC_EM7D is the default processor configuration. See Selecting ARC EM Configurations on page 21 for information on selecting a different configuration.

The default configuration is downloaded from the on-board SPI flash device automatically after the board is powered up or after you press the *FPGA configure* button located above the letter “C” of the ARC logo.

The configuration loading process might take up to five seconds after the board is powered on. The green LED on the FPGA module is on during this period. After the configuration is loaded, the ARC EM processor performs a self-test as described in the Run the bootloader and initial self-test section.

Bits 3 and 4 of the SW1 switch must be in the OFF position to perform a self-test.

Note

After FPGA configuration, the SW1 switch is available for user applications.

For more description of all the supported configurations and instructions on how to select a configuration other than the default configuration, see Working with ARC EM Starter Kit on page 21.

Next Steps:

Connect the USB Cable.

Connect the USB Cable

A single USB cable provides a communication channel between the debugger and the ARC EM Starter Kit board:

- A USB-JTAG interface
- A USB-UART debug console

If you have not installed the MetaWare Development Toolkit to use USB-JTAG channel, you can alternatively use the USB-UART channel to establish communication between the ARC EM core and serial console such as PuTTY.

USB Driver Installation

In order to use USB-JTAG and USB-UART interfaces, a driver is required on the host on which you run the MetaWare debugger or another serial debug console such as PuTTY.

The driver is a part of the Digilent Adept System tool and can be downloaded from the Digilent web site at www.digilentinc.com.

Follow the installation instructions provided by Digilent.

USB Connections

Use the USB cable delivered as part of the ARC EM Starter kit.

1. Connect the mini-USB plug to the connector **J7** on the board as shown on Figure 2.
2. Connect the other end of the cable to the host running the debugger.

Note

The J8 header needs to be open (jumper removed) to enable communication with the debugger via the USB-JTAG channel. This is the factory default setting. The USB-UART channel for a serial console is always enabled.

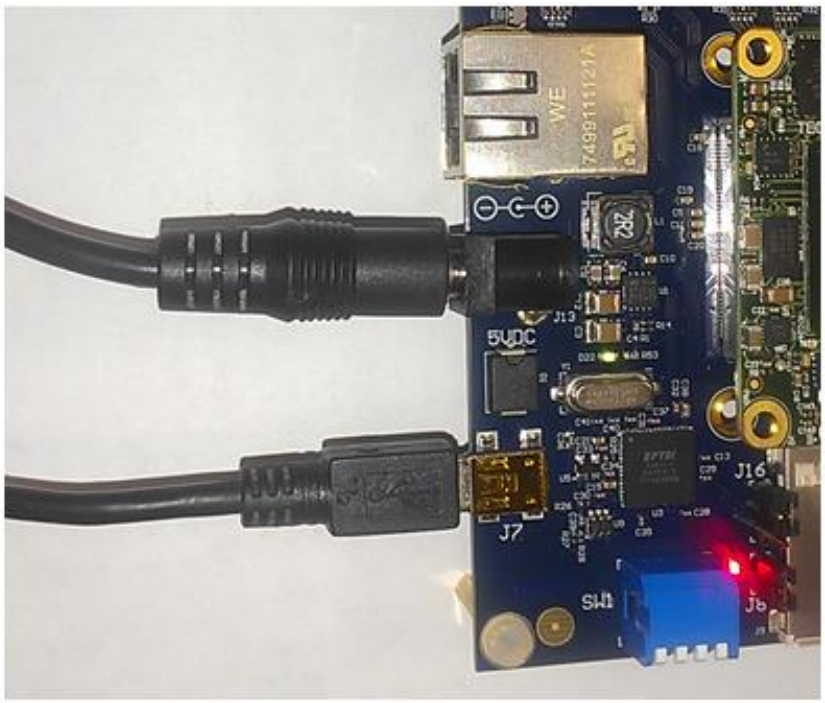
Connect the Power Supply

Use the universal switching power adapter (110-240 Volts AC to 5 Volts DC) provided in the package. Connect the appropriate AC plug for your AC power outlet. Connect the DC plug to the connector **J13** “5V DC” on the board as shown on Figure 2. Finally, connect the AC plug to your AC power outlet.

Note

The ARC EM Starter Kit board operates on 5V DC. It does not have any user-accessible means to change power-supply settings.

Figure 2 Power Supply and USB Cable Connection to the Board



Next Steps:

- Run the bootloader and initial self-test.

Run the Bootloader and Initial Self-Test

Bootloader and Initial Self-Test

After the FPGA image is loaded from the SPI-flash memory, the ICCM of the ARC processor contains a pre-programmed image of the self-test application that is executed after a CPU reset.

The self-test application behavior depends on bit 3 of the on-board DIP switch SW1.

Bit 3 – skip self-test. When this bit is on, the bootloader application does not check CPU ID and does not send any messages to the console.

To run the self-test the Bit 3 of DIP switch SW1 should be off.

The self-test performs the following actions:

1. Runs the startup code, which initializes program counter and stack pointer.
2. Checks the ARC EM `IDENTITY` register.
3. Flashes the LEDs.
4. Tests the SPI flash.
5. Prints the following information to the console:
   - Version of the firmware
   - Version of the bootloader
   - ARC EM configuration number
   - ARC processor ID
   - BCR registers content

Regardless of the position of bit 3, the bootloader enables LED7 to indicate the CPU start and indicates the loaded core configuration as follows:

- LED0 – ARC_EM7D
- LED1 – ARC_EM9D
- LED2 – ARC_EM11D

View Self-Test Output on PuTTY Console

PuTTY is a simple debug console that you can use when the MetaWare Development Toolkit is not installed.

Prerequisites

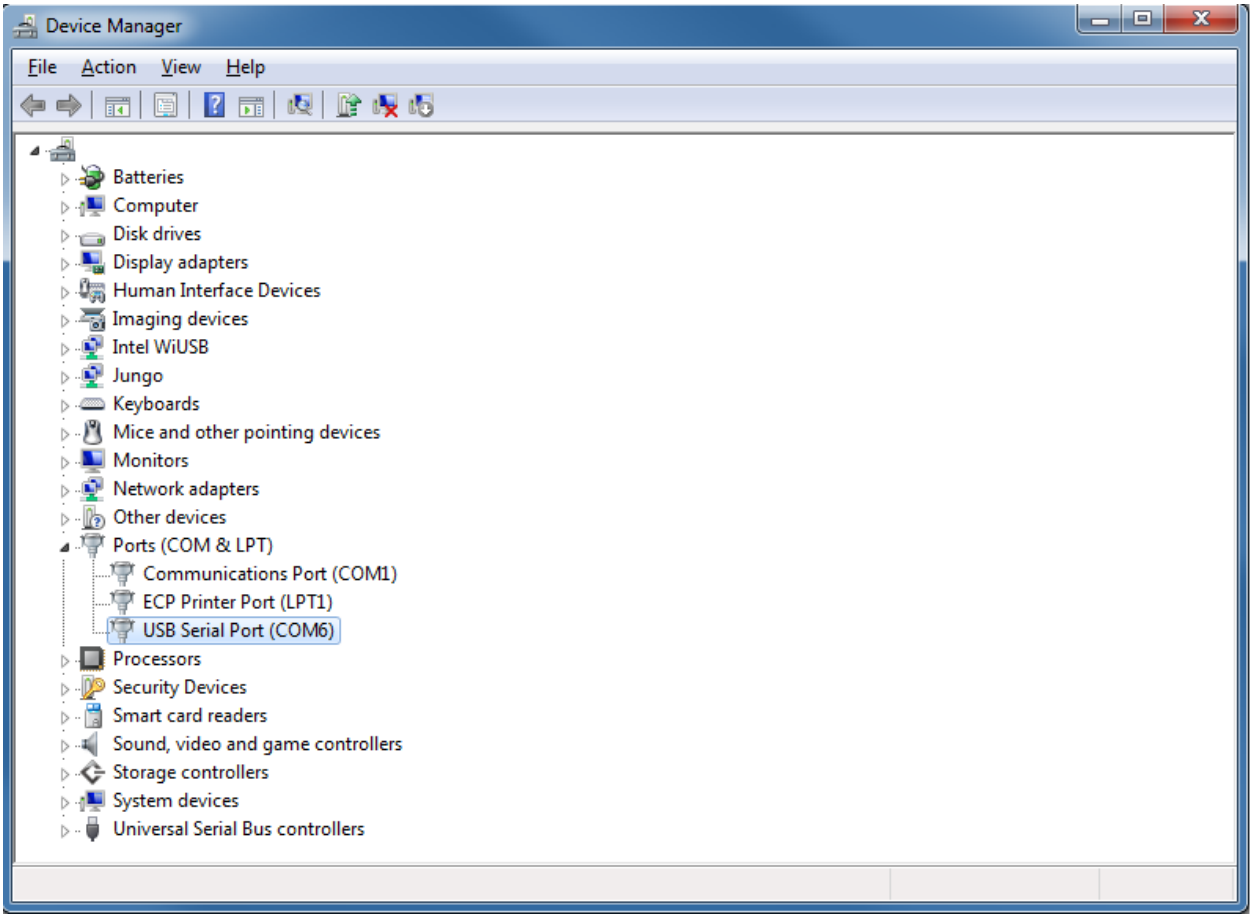
- USB cable is connected to your host and the USB drivers are installed.
- Board is powered up.

Procedure

1. Click **Start > Control Panel > Device Manager**.
2. In the **Device Manager** window, double-click **Ports (COM & LPT)**.
3. Select **USB Serial Port** and take note of the COM port assigned to the USB Serial Port

   Example: Figure 3 shows the port COM6 being used.

Figure 3 COM6 Port

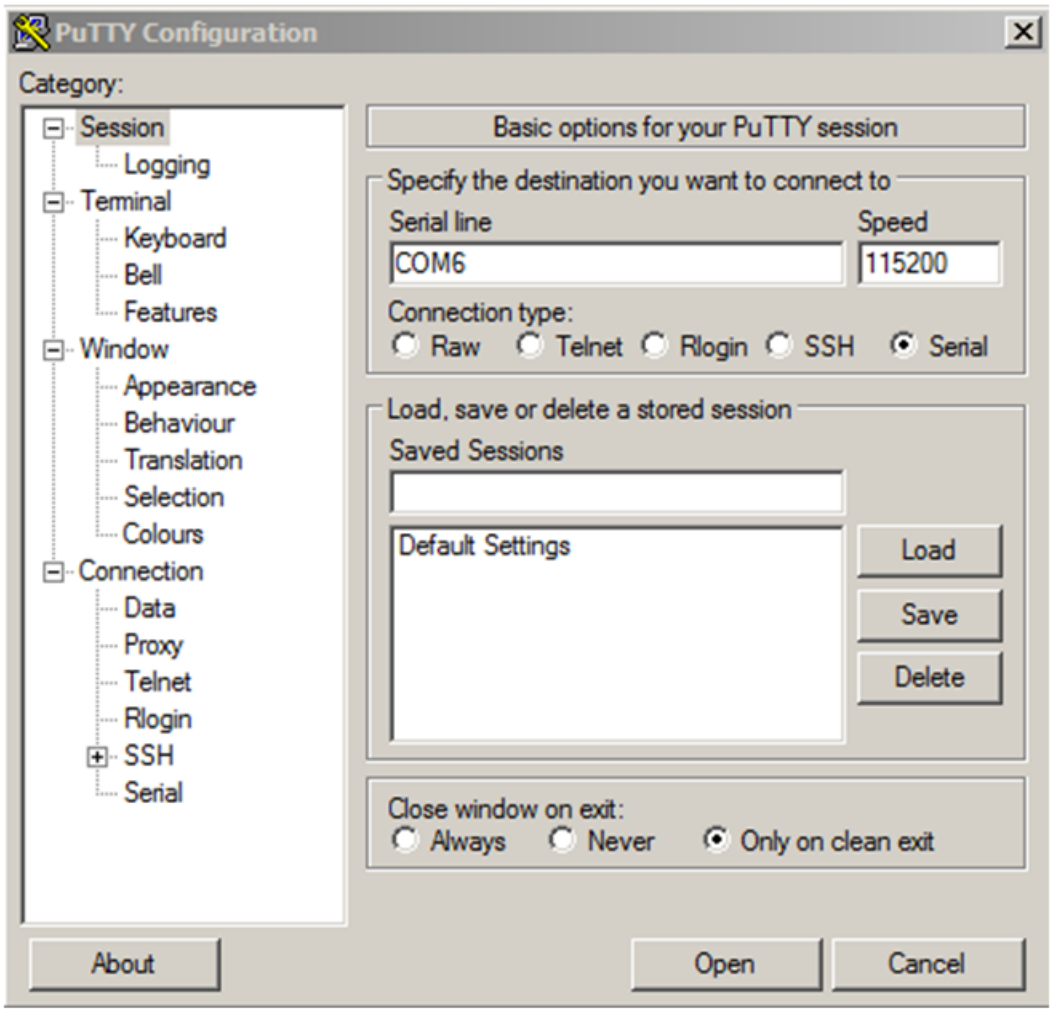


4. Download `putty.exe` from http://www.putty.org.
5. Double-click the downloaded `putty.exe` file.

The **PuTTY Configuration** window appears.

6. Select **Serial** connection type.
7. Enter the COM port name from Step 2 in the **Serial line** field and set the **Speed** field to `115200` as shown in Figure 4.

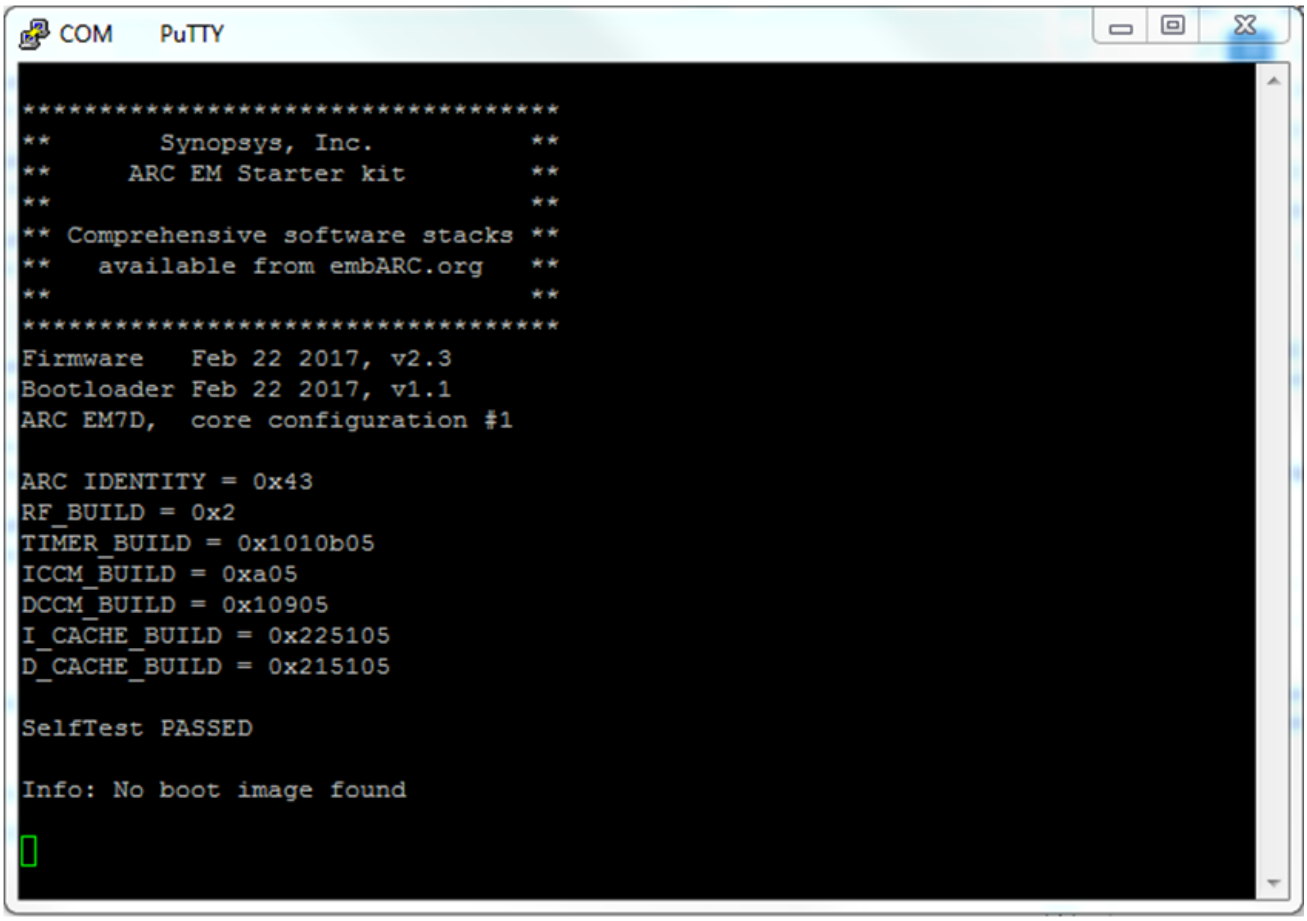
Figure 4 PuTTY Configuration Window



8. Click the **Open** button to launch the PuTTY terminal.
9. Press the **ARC EM Reset** button located above the letter “R” of the ARC logo after launching PuTTY.
10. The self-test is performed.

Figure 5 on page 20 shows the console output after completion of self-test application. This example uses the ARC_EM7D configuration (SW1 bits 1 and 2 both off).

Figure 5 Self-test Results in a Terminal Window, ARC_EM7D



Next Steps:

After the self-test successfully completes, you can start working with the ARC EM Starter Kit. The following chapters explain some of the tasks you can do with the ARC EM Starter Kit:

- Selecting ARC EM Configurations
- Connecting External Interfaces to the ARC EM Starter Kit
- Connecting the PMODAD2 Extension Module
- Obtaining embARC Software and Documentation
- Creating Applications Using the MetaWare Tools

4
Working with ARC EM Starter Kit

Selecting ARC EM Configurations

This section describes the usage of the preloaded FPGA images that are stored in the on-board SPI Flash device.

Spartan-6 FPGAs include a capability called MultiBoot that allows the FPGA to selectively reprogram and reload its bitstream from an attached external memory. The MultiBoot feature allows the FPGA application to load different FPGA bitstreams under the control of the FPGA application.

The FPGA board includes an SPI flash storage device that is pre-programmed with FPGA images with different configurations of DesignWare® ARC EM cores: ARC_EM7D, ARC_EM9D, and ARC_EM11D.

Table 1 describes the main parameters of each configuration.

Table 1 Predefined ARC Configurations

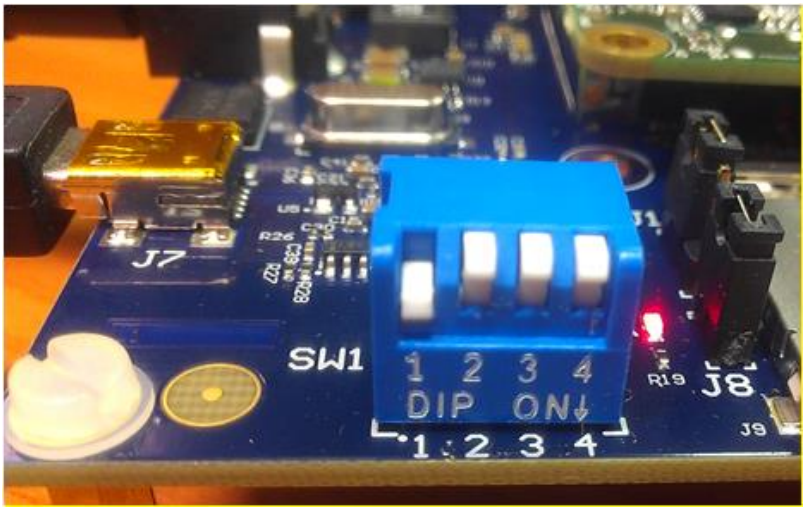
Parameters	Configurations		
	ARC_EM7D	**ARC_EM9D**	**ARC_EM11D**
ICCM	256 KB	256 KB	64 KB
DCCM	128 KB	128 KB	64 KB
DDR2	128 MB	128 MB	128 MB
Instruction Cache	16 KB	-	16 KB
Data cache	16 KB	-	16 KB
Address Width	32	32	32
Timers	2	2	2
XY	-	DCCM+X+Y, 8K	DCCM + Y, 8k
DSP	complex	complex, itu, accshift	complex, itu, accshift

Parameters	Configurations		
	ARC_EM7D	**ARC_EM9D**	**ARC_EM11D**
FPU	-	Single precision	Single and double precision
MPU	True	-	-
Fmax	25 MHz	20 MHz	20 MHz
Note: The ARC EM4, ARC EM5D and ARC EM6 processors are subsets of ARC EM7D processor. So software may be built for ARC EM4, ARC EM5 or ARC EM6 processors but then run on the ARC EM7D processor configuration on the starter kit.			

For a detailed description of the predefined configurations, see Detailed Core Configurations.

The FPGA image can be selected with bits 1 and 2 of the SW1 DIP switch on the baseboard as shown in Figure 6.

Figure 6 SW1 DIP Switch



The definitions of bits 1 and 2 are described in the Table 2.

Table 2 Selecting the Configuration

Bit 1	Bit 2	Configuration
OFF	OFF	ARC_EM7D
ON	OFF	ARC_EM9D
OFF	ON	ARC_EM11D
ON	ON	Reserved

The selected configuration is downloaded from the on-board SPI flash device automatically after power on or after you press the FPGA configure button located above the letter ‘C‘ of the ARC logo on the baseboard.



After the FPGA is programmed, the DIP switches are available for user applications. They must be switched back to the position for the desired predefined FPGA image, before you press the FPGA Configure button and power the board up.

Connecting External Interfaces to the ARC EM Starter Kit

This section describes the connection of peripheral devices using Pmod connectors on the baseboard.

The baseboard has several unified connectors that can be used to connect peripheral devices. These connectors comply with the Pmod™ Interface Specification by Digilent Inc.

Refer to the *Digilent Pmod Interface Specification* for more details.

Figure 7 Peripheral Device Connected Using Pmod



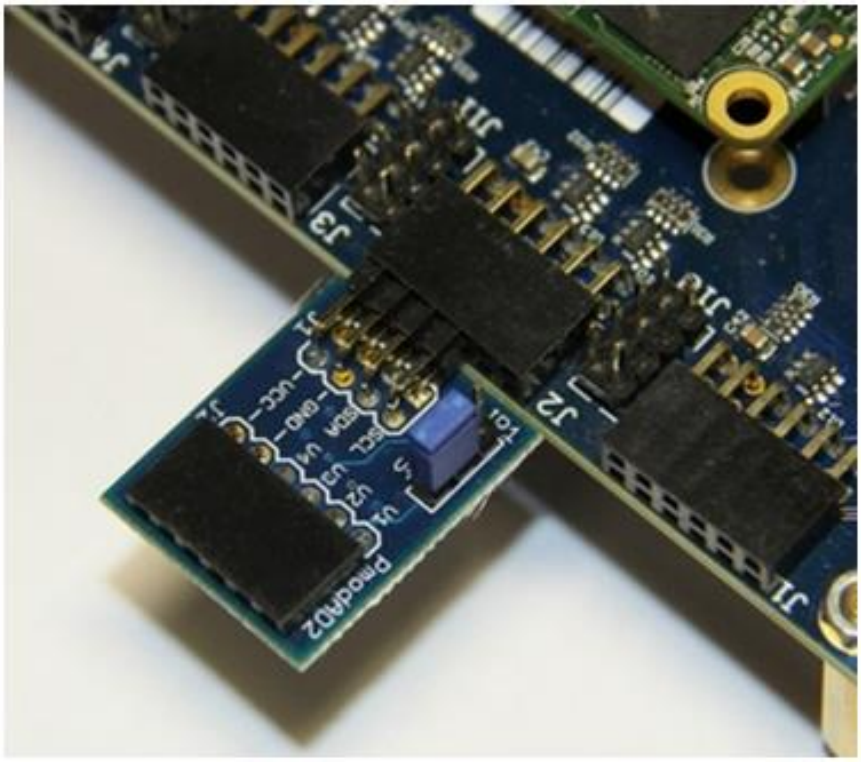
Connecting the PmodAD2 Extension Module

A PmodAD2 module is included in the ARC EM Starter Kit. This is a four-channel 12-bit A/D converter with a Pmod I2C interface.

The module has a 2x4-pin Pmod connector. Insert the module on the left side of connector J2 or J4 using pins 3, 4, 5, 6, 9, 10, 11, 12.

As an example, Figure 8 shows how the PmodAD2 can be connected to Pmod2.

Figure 8 PmodAD2 Connection to Pmod2



For information on how to configure and use Pmod connectors, see section Pmod Pin Configuration in page 46.

5
Obtaining embARC Software and Documentation

This section describes how to obtain the open-source embARC software and documentation for use with the ARC EM Starter Kit.

About embARC Software and Documentation

Drivers, operating systems and middleware ported to the ARC EM family of processors for the EM Starter Kit are provided through embARC. Sign up at https://forums.embarc.org/ to get access to the embARC software downloads and interact with other users.

See https://embarc.org/help.html for access to getting started, FAQ and other quick start information for using embARC.

An extensive collection of application examples is available with corresponding documentation under `/embARC/doc/embARC_Document.html` after you install the embARC zip archive on the development host.

6 Creating Applications Using the MetaWare Tools

This chapter describes how to create and run a simple application using the MetaWare tools in IDE and command-line modes.

Using MetaWare or MetaWare Lite

You can use the MetaWare Development Toolkit or MetaWare Lite.

For how to use the MetaWare IDE or command line, see the MetaWare documentation and the embARC documentation.

Using MetaWare

The DesignWare® MetaWare® Development Toolkit for ARC requires a license from Synopsys.

Follow the instructions provided with DesignWare® MetaWare® Development Toolkit to install the toolkit and set up a license.

Using MetaWare Lite

Download MetaWare Lite

1. You must register at the following website to download MetaWare Lite:

   http://www.synopsys.com/cgi-bin/arcmwtk_lite/reg1.cgi
2. Complete the registration form and click **Continue** to receive the download link.

   A download link with the activation key is sent to the registered email address.
3. Click the download link in the welcome mail to download the software.

Install MetaWare Lite



Note
Ensure you have the following information before proceeding with the installation:

- Email address you registered with Synopsys Inc. to download MetaWare Lite.
- Activation key — you can find this information in the confirmation email from Synopsys Inc.

1. Double-click the `DW_ARC_MetaWare_Lite_2015_12_win.exe` file.
2. Follow the on-screen instructions to complete the installation process.

Creating a Simple Application Using the Command-Line Tools

The following procedure describes the process of building a simple “Hello world” application for the ARC EM Starter Kit. This application can be used as a starting point for creating your own applications to run on the Starter Kit.

Prerequisites:

- Ensure that you have downloaded and installed the MetaWare Development Toolkit. See Download and Install MetaWare.
- Ensure that you have connected the ARC EM Starter Kit to the host. See Quick Connection Guide.

Perform the following steps in order to create an application:

1. Create a simple C program that prints “Hello world” string to the default `stdout` device. For example:

```
#include "stdio.h"

int main(void) {
   printf("Hello world");
   return 0;
}
```

2. Save the code above as `myhello.c`

3. Build your application running the MetaWare compiler in command line mode (`<path to tcf-files>` should point to the TCF files folder provided with the embARC release package):

```
ccac myhello.c -tcf=<path to tcf-files>/em7d.tcf -g -o myhello.elf
```

This command compiles and links `myhello.elf` for EM7D core. The `myhello.elf` file is created in the same folder as the source file. The TCF file depends on the core configuration you selected by onboard DIP-switch SW1.

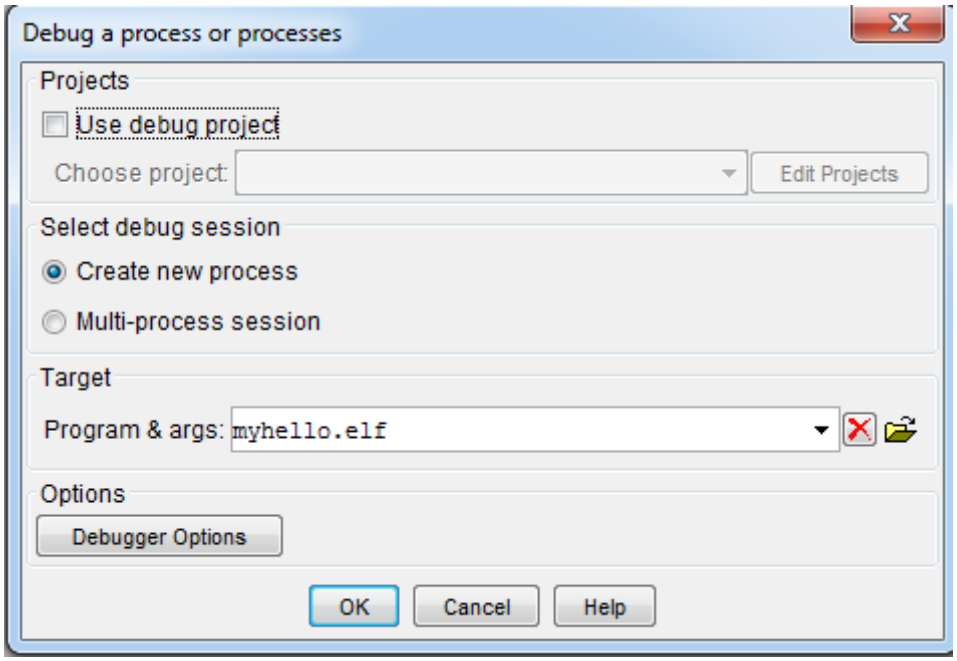
Debugging Applications Using the MetaWare Debugger

1. Use the following command to launch your application in the MetaWare GUI debugger in an interactive mode:

```
mdb -digilent -prop=dig_speed=5000000 -nooptions myhello.elf
```

2. In the Debug a process or processes dialog, click **OK** to run Debugger

Figure 9 MetaWare Settings Required to Use ARC JTAG through USB



3. Click the **Run** button:



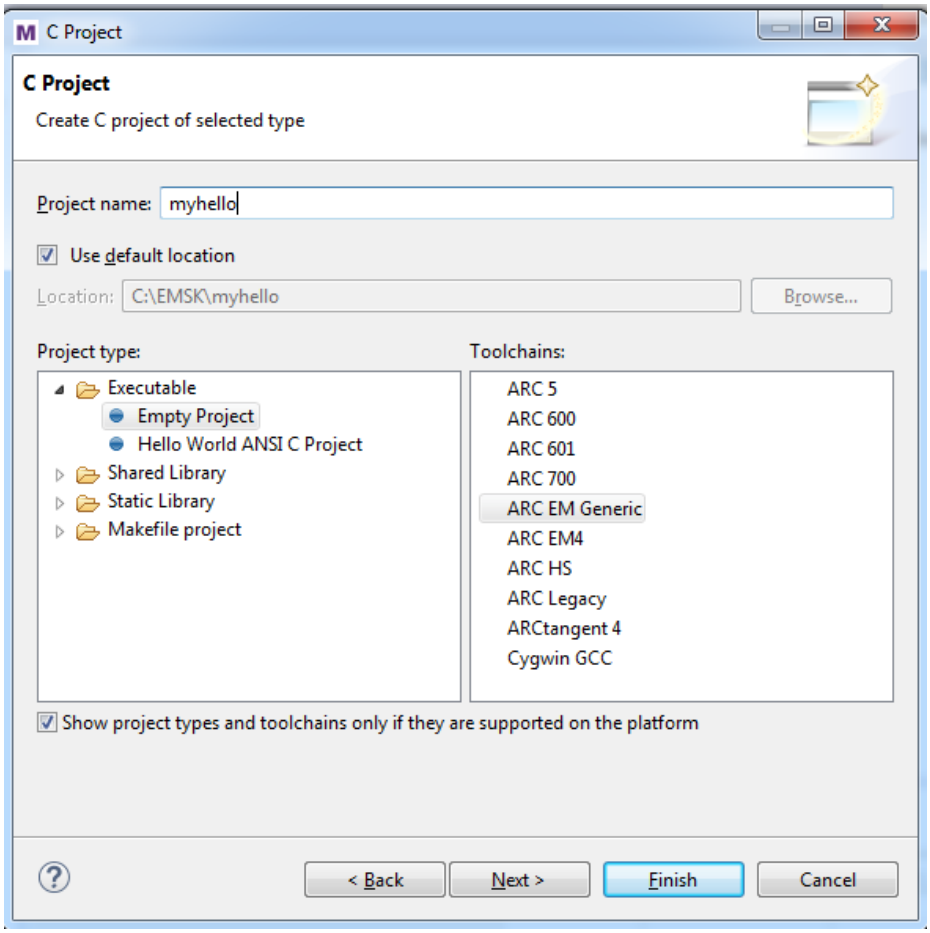
"Hello world" output appears on the command line.

Creating Application Using the MetaWare IDE

Create a New Project in the MetaWare IDE

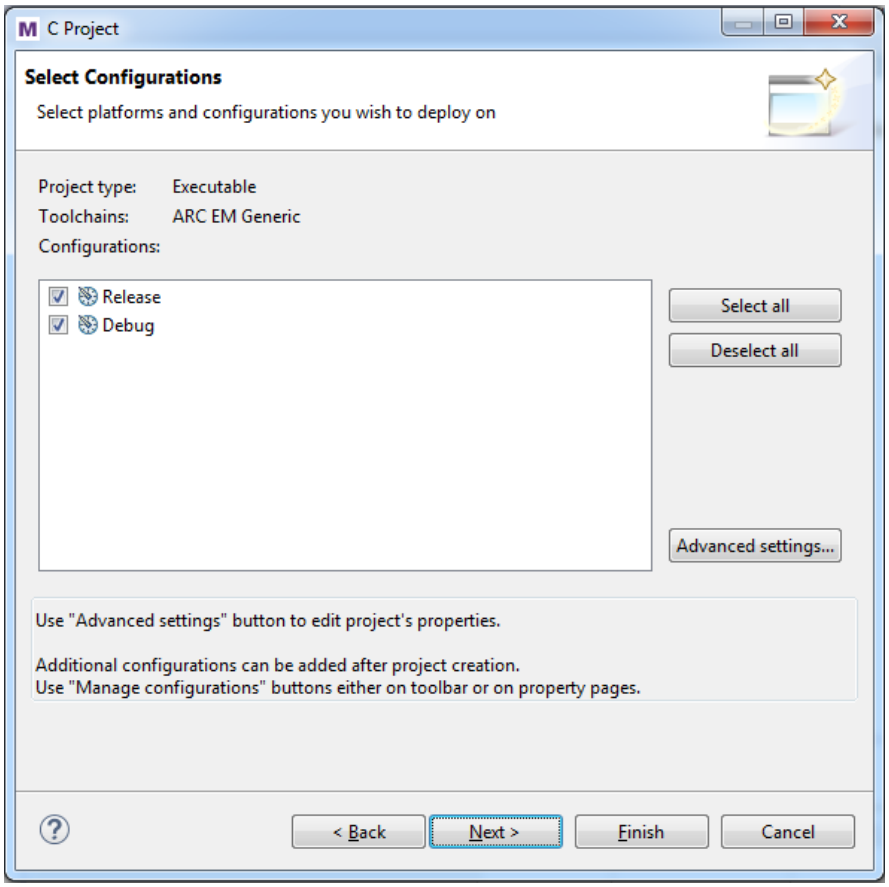
1. Launch the MetaWare IDE.
2. Select **File > Create a New Project...**
3. From the **Select a wizard** dialog box that results, select **C/C++-> C Project** and click **Next**.
4. Add a **Project Name** (for example: `myhello`).
5. To create an **Empty Project** with an **ARC EM Generic** executable, select the appropriate options as shown in Figure 10, and then click **Next**.

Figure 10 MetaWare IDE - Creating a New Project



6. Click **Next** again to create both Debug and Release configurations.

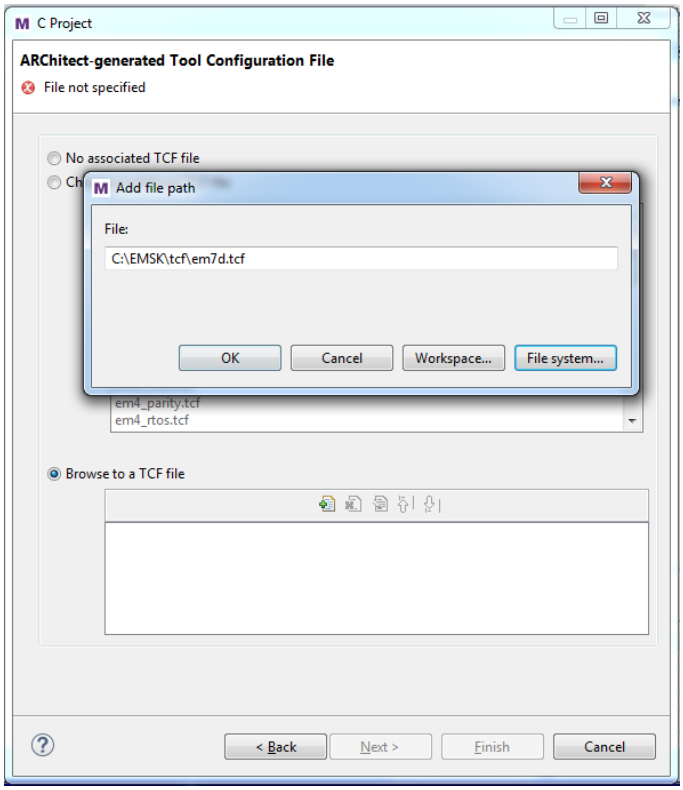
Figure 11 MetaWare IDE - Select Project Configurations



7. Select “Browse to a TCF file” in the **ARChitect-generated Tool Configuration File** dialog, then select the path to the TCF file corresponding to the Starter Kit CPU configuration you are using; this step presets the IDE with the right compile options and memory mapping.

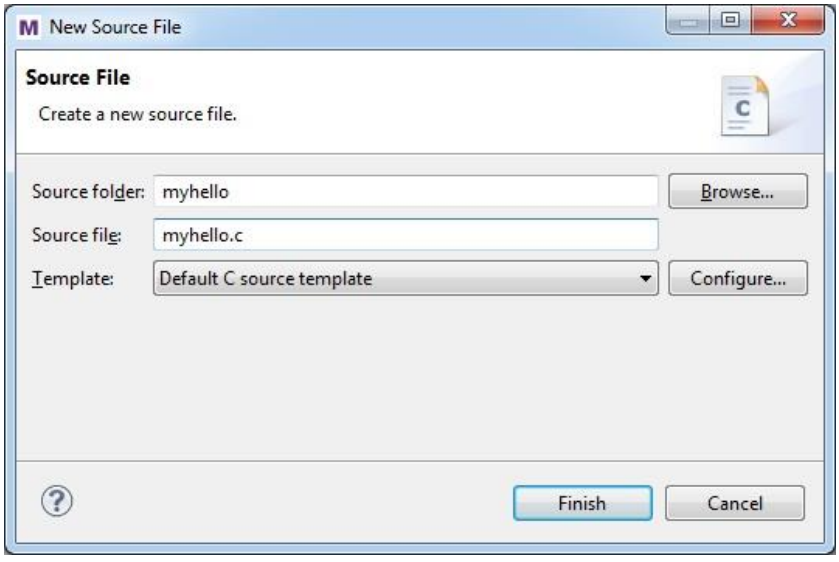
   Note: the TCF file location on your computer may vary. Use the TCF files provided with the EM Starter Kit firmware package.

Figure 12 MetaWare IDE - Select TCF File



8. Click **Finish** to create the project.
9. **Select File > New > Source File** and, create a new source file for your project. For example `myhello.c`

Figure 13 MetaWare IDE - Create New Source File



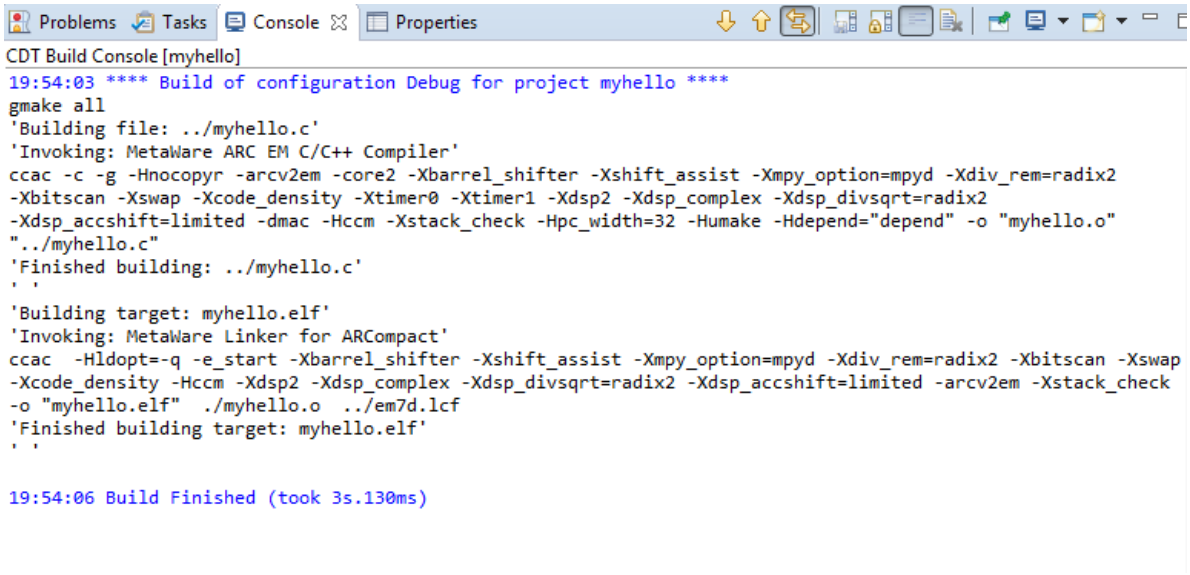
Create a simple C program that prints “Hello world” to the default `stdout` device. For example:

```
#include "stdio.h"

int main(void) {
   printf("Hello world");
   return 0;
}
```

10. Save your file.
11. To create the executable, select **Build Project** under the **Project** main menu, or mouse over the **myhello** project, right-click, and select **Build Project**.
12. Review the messages in the **Problems** and **Console** panes at the bottom of the window to verify a successful build.

Figure 14 MetaWare IDE - Console Output for Test Project



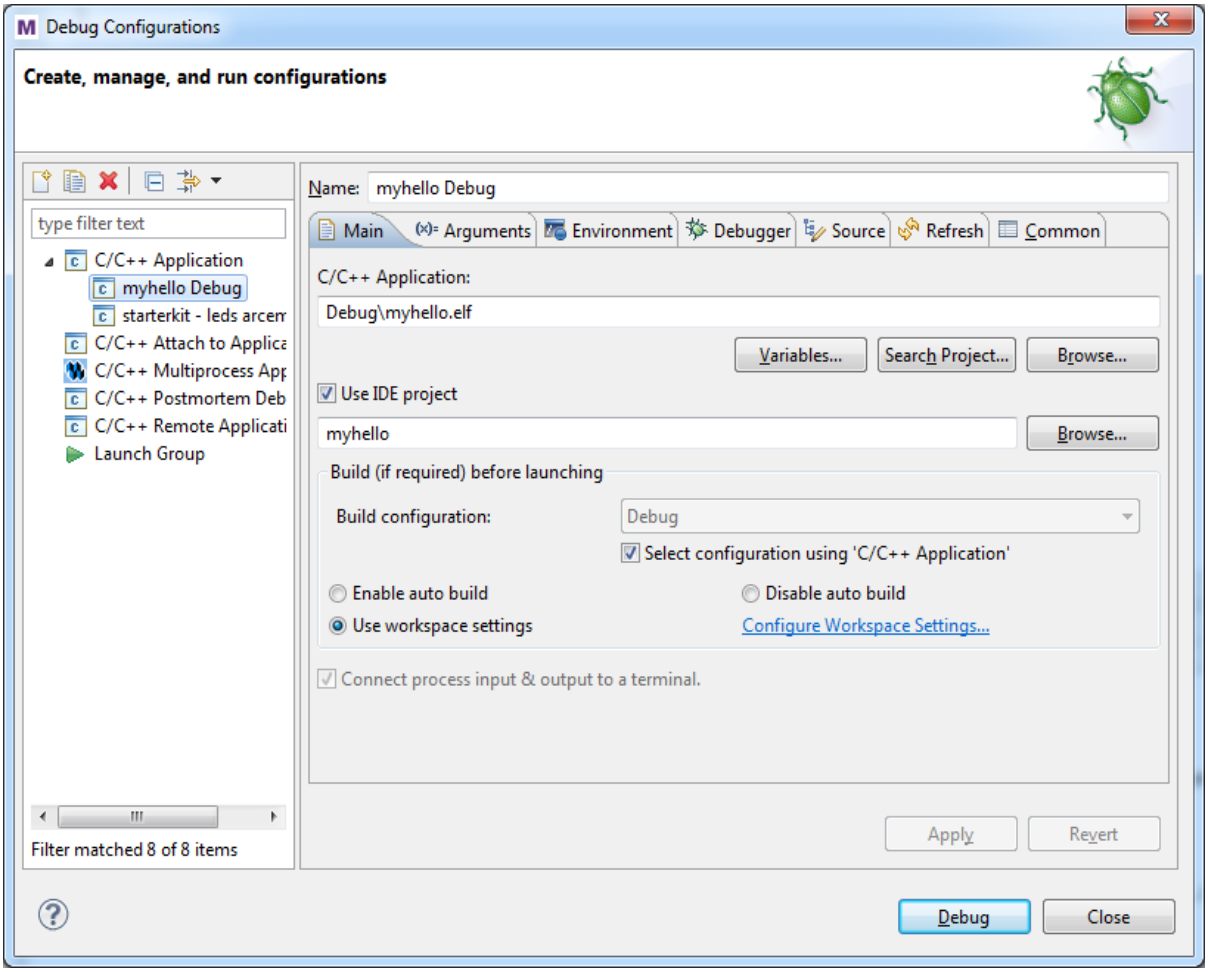
Invoking the Debugger and Running the Executable

Make sure that the ARC EM Starter kit board is connected to your computer by USB cable.

1. Select **Run > Debug Configurations** or right-click **myhello** and select **Debug As > Debug Configurations**.
2. Double click the **C/C++ Application**, or click **New** icon ( ) with **C/C++ Application** selected

The IDE displays a window similar to Figure 15. The **Main** tab is selected by default.

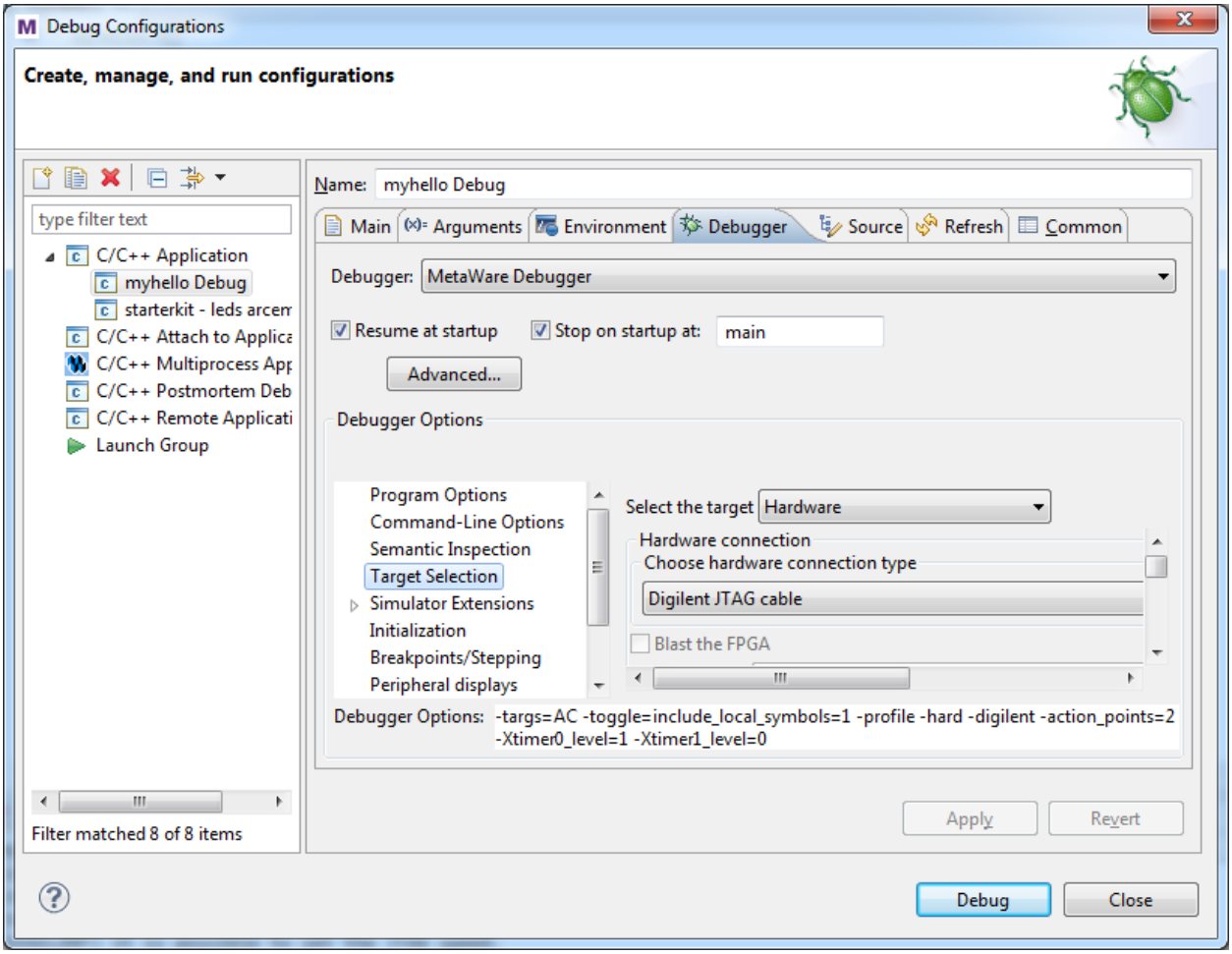
Figure 15 MetaWare IDE - Creating New Debug Configuration



3. Click the **Debugger** tab and set up the debugger to run the application on an ARC EM Starter Kit board.

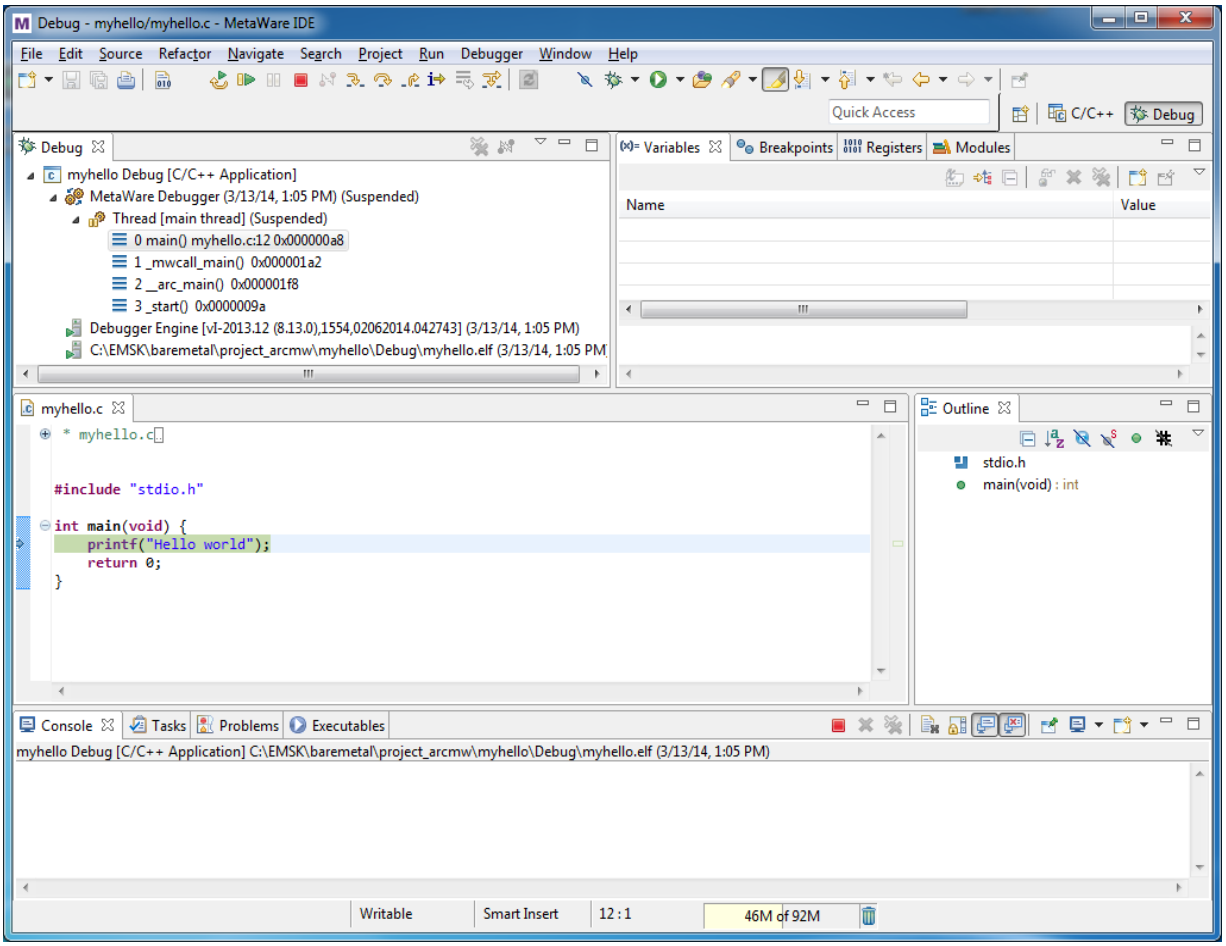
4. Click the **Target Selection** menu in the **Debugger Options** field.
    a. On the right side of **Debugger Options** field, in the **Select the target** field, select **Hardware.**
    b. In the **Hardware connection** field, choose the **Digilent JTAG cable** from the **Choose hardware connection type** drop-down menu. Keep the defaults for all other settings.

Figure 16 MetaWare IDE - Configure Debugger



5. Click **Debug** and confirm the perspective switch by selecting **yes** in the pop-up window. The **Debug** perspective opens as shown in Figure 17.

Figure 17 MetaWare IDE - Debug Perspective

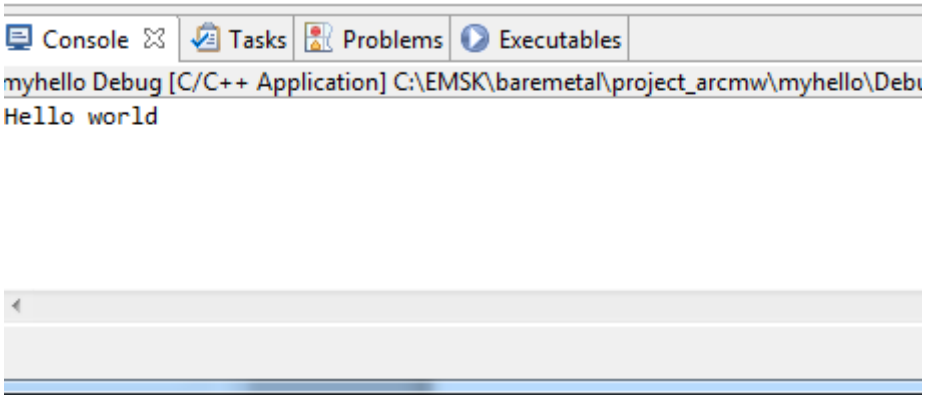


6. Click the **Resume/Run** button:



7. The application prints “Hello world” in the **Console** window.

Figure 18 MetaWare IDE - Console window



7
Running Applications in Self-Boot Mode

This chapter describes how to build a self-booting application, write it into SPI memory, and run it after the board power up or reset.

Self-Boot Application Concept

ARC EM Starter kit provides an ability to store a customer application in SPI flash memory and run it after power up. That application is called a *self-boot application* in this chapter.

The following are the basic stages in preparing and using self-boot applications:

1. Build the self-boot application
2. Write the self-boot application to SPI memory
3. Run the self-boot application

Running the application is a process executed by ARC EM Starterkit board. Preparing application for running in self-boot mode requires connecting the ARC EM Starter Kit to a host with Windows OS and installed MetaWare or MetaWare Lite tools.

Building a Self-Boot Application

The following requirements exist for an application built for running in self-boot mode:

- The application can be built for a target CPU configuration. There are some differences between three configurations of cores supported by ARC EM Starter Kit, and an application built for example for ARC EM7D might not work properly on an ARC EM9D. Make sure that you build your application using the correct core configuration.
- The application must be linked to use a valid memory segment and cannot use the last 6 K of the ICCM region, which is reserved for the bootloader. A valid memory segment means that the memory used by your application must be available for a target core configuration. An application mapped in non-existent memory region does not work.
- The application has to be built without hostlink support. Make sure you are using the `-Hhostlib=` option (without argument) when you build your application with the MetaWare tools.

Write the Self-Boot Application Image to SPI Flash

The SPI interface on the ARC EM Starter Kit board does not connect directly to JTAG and not available through any PMOD. For more information and examples on how to write an application to SPI flash memory, see embARC.org.

Running the Self-Boot Application

After the board is powered up, the ICCM memory has a predefined application that includes self-tests and bootloader parts. After completion of self-tests, this application checks state of on-board DIP switch SW1. If bit 4 of SW1 is ON, the bootloader is starting. The bootloader checks SPI flash memory for a data structure with the self-boot image parameters. This structure includes the following information:

```
typedef struct {
  unsigned int head;        // 0x68656164 ='head'
  unsigned int cpu_type;    // = 0 - all images
  unsigned int start;       // start address of application image in spi flash
  unsigned int size;        // size of image in bytes
  unsigned int ramaddr;     // address of ram for loading image
  unsigned int ramstart;    // start address of application in RAM !!!!
  unsigned int checksum;    // sum of all bytes in image
} image_t;
```

The bootloader looks for a `head` signature in SPI flash memory at address 0xD80000. After the signature is found, the bootloader reads the image parameters from the header and starts loading procedure:

1. The bootloader starts copying the application image from address `start` in SPI-flash to address `ramaddr` in the processor memory. The number of bytes for copying is declared in the `size` parameter of self-boot header. The `ramaddr` can point to a DDR, ICCM or DCCM memory space, except memory region corresponding to last 6K of ICCM memory (actual address depends on configuration), which is reserved for the bootloader.
2. After copying the data, the bootloader compares the checksum of the application image with the checksum from self-boot header, and if they are equal the bootloader start executing the application from the address specified by the `ramaddr` parameter of self-boot header.

Position Bit3 of on-board DIP switch SW1 affects the console output of the bootloader. Setting Bit 3 to the ‘On’ position suppresses console output. When Bit 3 is on the ‘Off’ position the bootloader prints the parameters of the self-boot image and indicates the status of the loading process.

Note The bootloader does not check the compatibility of application stored in SPI flash memory with selected core configuration. Make sure that you run your application on the right core configuration.

SPI Flash Memory Structure

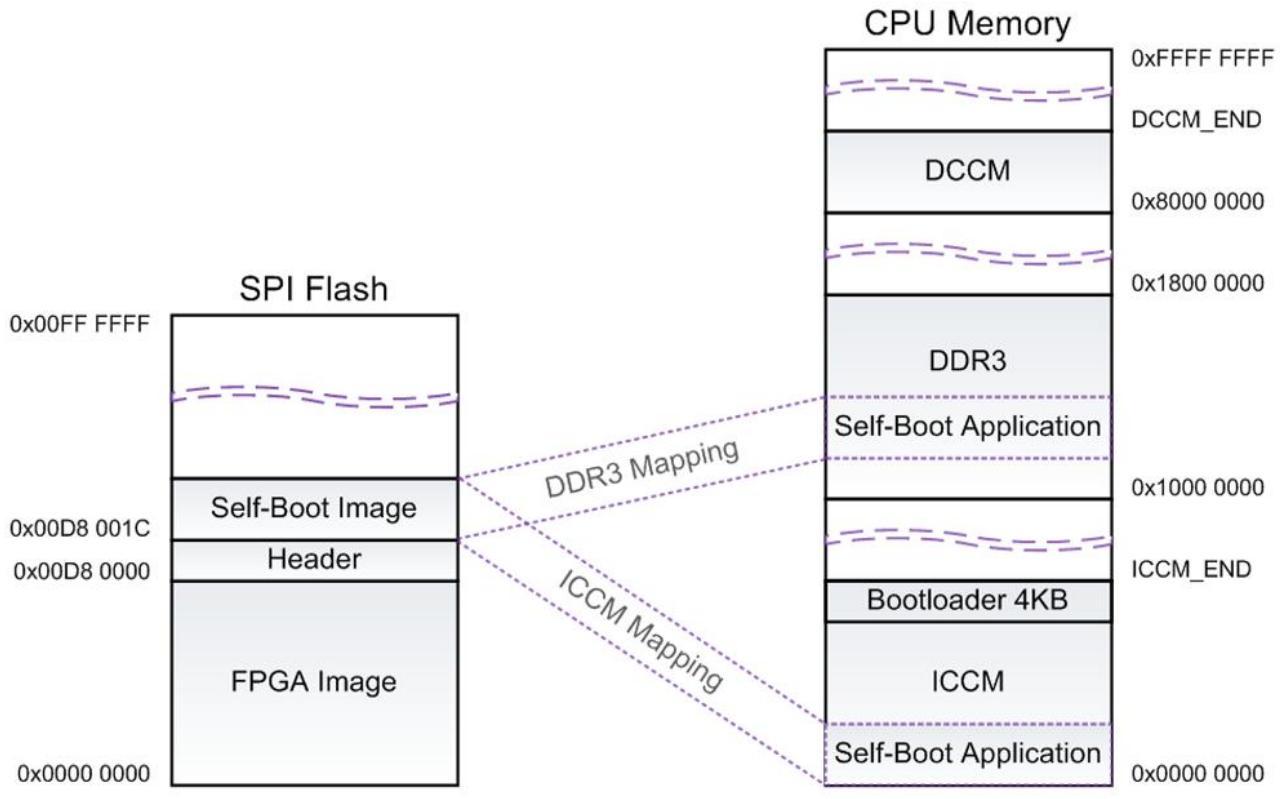
The flash memory is divided in two sections: *FPGA image* and *Application*. The *Application* section contains the main code for the application and header, and the FPGA section contains the default core configurations for FPGA.

The header contains the following information:

- Application image size
- RAM destination address
- Checksum
- Start address from which the CPU starts code execution after loading

A typical memory structure is shown in Figure 19.

Figure 19 Typical Flash Memory Structure



The ICCM_END and DCCM_END addresses depend on the CPU configuration described in the Core Configurations and Memory Mapping section.

Appendix: A
Hardware Functional Description

This appendix provides information about the hardware used in the ARC EM Starter Kit.

Board Overview

The ARC EM Starter Kit consists of two boards:

- FPGA module with a Xilinx Spartan-6 XC6SLX150-3FGG484I FPGA device.
- Base board with extension connectors and peripherals.

The kit has the following on-board peripheral devices:

- 2×16-bit wide 1 Gbit (128 MB) DDR3 SDRAM
- 128 Mbit (16 MB) SPI Flash memory (Winbond W25Q128BV)
- SD card reader
- LEDs
- Push-buttons
- DIP switches
- Seven Pmod connectors
- On-board clock oscillator

The SPI flash contains three pre-programmed FPGA images. One image is loaded after board power up, depending on DIP switch position. Refer to Selecting ARC EM Configurations section for more information.

Every FPGA image includes an ARC EM core and a set of DesignWare peripheral controllers connected to the on-board devices and external interfaces via pin-sharing logic. This allows sharing of 52 I/O pins between GPIO, SPI, I2C and UART devices. The customer application can configure the external connectors as needed (see External Hardware Interfaces section for more information).

The ARC_EM7D core runs at 25 MHz, ARC_EM9D and ARC_EM11D run at 20 MHz.

Depending on the selected configuration, the ARC core supports the following features:

- Different internal memory (SRAM) or external DDR3 memory with different cache parameters
- Floating Point Unit

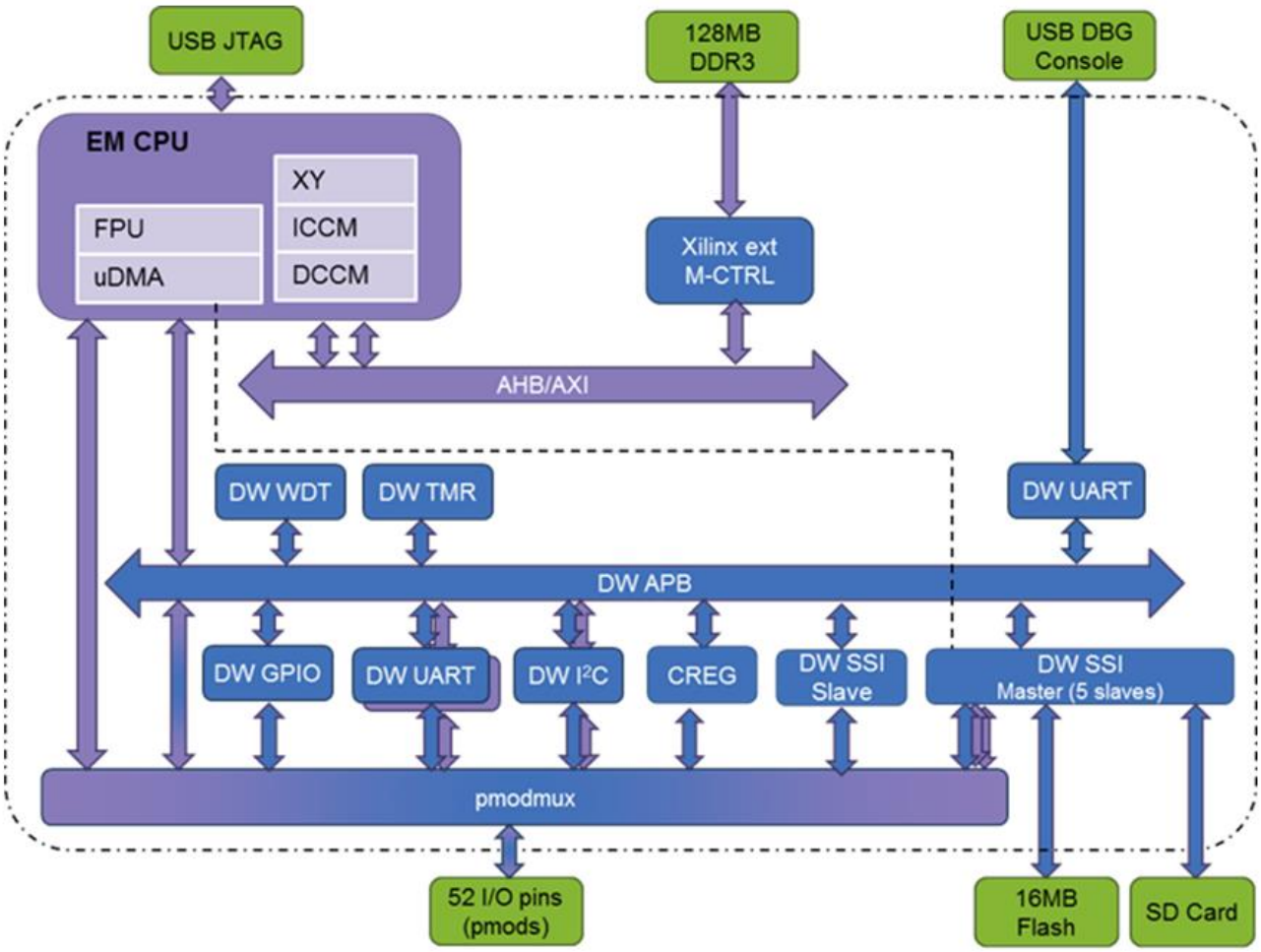
The basic peripheral set includes the following items:

- `GPIO`
  Four ports connected to LEDs, buttons, on-board switch, and external connectors
- `DW UART`
  Three separate ports.
  UART0 is connected to the external interfaces using pin-sharing logic. It can optionally be accessed at the Pmod1 connector.
  UART1 is internally connected to an on-board UART-to-USB converter and used as the default console.
  UART2 is connected to the external interfaces using the pin-sharing logic. It can optionally be accessed at the Pmod5 connector.
- `DW SPI master`
  SPI master controller. It is connected to on-board devices (SPI flash and SD card) and to external interfaces via pin-sharing logic and can optionally be accessed at the connectors Pmod5 or Pmod6. It can also address the SPI slave interface for loop-back testing.
- `DW SPI slave`
  SPI slave controller. It is connected to the external interface using pin-sharing logic and can optionally be accessed at the Pmod1 connector.
- `DW I2C`
  Two separate ports.
  I2C_0 controller is connected to the external interfaces using pin-sharing logic and can optionally be accessed at the connectors Pmod2.
  I2C_1 controller is connected to the external interfaces using the pin-sharing logic and can optionally be accessed at the connector Pmod4.
  The PmodAD2 extension board, provided with the ARC EM Starter kit, connects to the main board using the I2C interface.
- `DW Timers`
  Two separate timers.
- `DW WDT`
  WDT Controller.

FPGA Design Overview

Figure 20 shows the hardware architecture of the FPGA design implemented in the ARC EM Starter kit.

Figure 20 FGPA Design Block Diagram



External Hardware Interfaces

This section describes the external hardware interface devices that can be used with ARC EM Starter Kit.

External hardware interface devices can be connected to the ARC EM Starter Kit using Pmod connectors. Table 3 provides an overview of possible peripheral combinations for each Pmod connector.

Table 3 Peripheral Devices Connection to Pmods

Pmod	Pmod Connector	Peripherals	
		MUX Option 0	MUX Option 1
Pmod1	J1	GPIOs	External SPI master + UART0
Pmod2	J2	GPIOs	ARC EM control/status signals + I2C_0
Pmod3	J3	GPIOs	Reserved
Pmod4	J4	GPIOs	GPIO + I2C_1
Pmod5	J5	GPIOs	External SPI slave 1 + UART2
Pmod6	J6	GPIOs	External SPI slave 0 and CS outputs for all other SPI slaves + ARC EM control/status signals
Pmod7	J20	GPIOs	ARC EM control/status signals

The functionality of the Pmod connectors is programmable. Each connector can operate either as a GPIO or as a dedicated peripheral device. Several connectors provide access to ARC EM status and control pins.

The GPIO ports are configured as inputs or outputs using the GPIO driver.

A PmodAD2 module is included in the ARC EM Starter kit. This is a 4-channel 12-bit A/D converter with a Pmod I2C interface. It can be connected to any of the connectors J2 or J4. Refer to Connecting the PmodAD2 Extension Module section for more details.

The sections below provide information about register programming, possible connections, and other details.

Connecting Peripheral Controllers

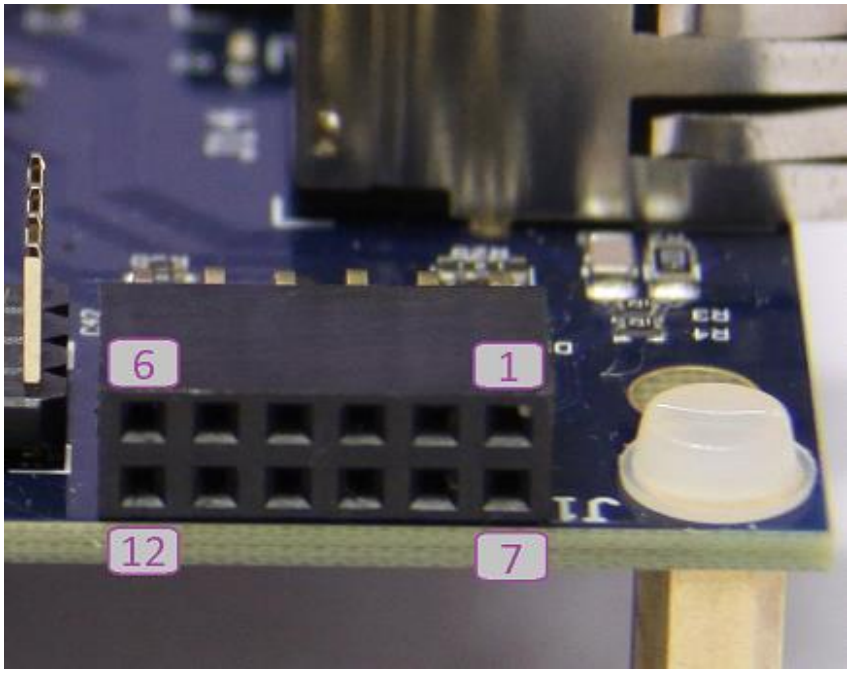
Table 4 Board Connections Overview of Peripheral Controllers

Peripherals	Board Connection
GPIO Port A[2:0]	Buttons: (Above letter 'A' in ARC logo, L, R)
GPIO Port A[31:8]	Pmods (J1, J2, J3, J4, J5, J6) pins [10:7]
GPIO Port B[8:0]	LEDs [8:0] Port B[0] controls LED0 Port B[1] controls LED1 ... Port B[8] controls LED8 The LED is on if its corresponding control bit is programmed to `0`.
GPIO Port C[3:0]	DIP Switch SW1 SW1 switch 1 connected to Port C[0] SW1 switch 2 connected to Port C[1] SW1 switch 3 connected to Port C[2] SW1 switch 4 connected to Port C[3]
GPIO Port C[31:8]	Pmods (J1, J2, J3, J4, J5, J6) pins [4:1]
GPIO Port D[11:0]	Pmods (J4, J20)
I2C_0	Pmod J2
I2C_1	Pmod J4
SPI Master CS0	Pmod J6 pins [4:1]
SPI Master CS1	Pmod J5 pins [4:1]
SPI Master CS3	SD Card
SPI Master CS4	Internal loopback to SPI Slave when register `SPI_MAP_CTRL[0]=1` (for test purposes).
SPI Slave	Pmod J1 [10:7] pins when `SPI_MAP_CTRL[0]=0` Internal loopback to SPI Master when `SPI_MAP_CTRL[0]=1` (for test purposes). For more details, see SPI Slave.
UART 0	Pmod J1 pins [4:1]

Peripherals	Board Connection
UART 1	USB UART (debug console)
UART 2	Pmod J5 pins [10:7]

Figure 21 shows the location of the Pmod pins on the board connectors.

Figure 21 Pmod Pin Numbering



Pmod bits 5 and 11 are connected to GND for all PMOD connectors.

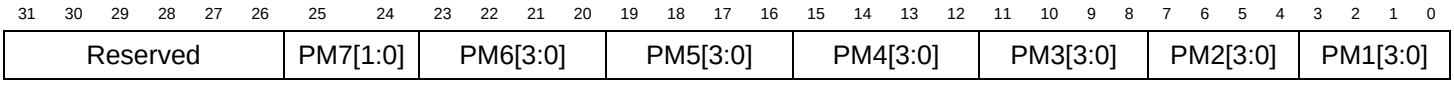
Pmod bits 6 and 12 are connected to 3.3 V for all PMOD connectors.

Pmod Pin Configuration

Pmod pins are configured by the Pin Mux Controller using the `PMOD_MUX_CTRL` register. This register is mapped to the peripheral register file (see Table 22).

PMOD_MUX_CTRL Register

Default	`0x0`
Peripheral memory offset	`0x0`
Access	Read/Write



Pmod1 Configuration

PM1[3:0] – select signals connected to Pmod1 at the J1 connector

Pins 1, 2, 3, 4, and 7 of Pmod1 (J1) are pulled up irrespective of the setting of PM1.

Note: PM1[1] and PM1[3] are reserved.

Using Pmod1 (J1) As a GPIO Port

This is the default setting after a reset.

Set *PM1[0] = 0* – Pmod1[4:1] are connected to DW GPIO Port C[11:8].

Set *PM1[2] = 0* – Pmod1[10:7] are connected to DW GPIO Port A[11:8].

See Table 5 and Table 6 for the detailed pin assignment.

Using Pmod1 (J1) Upper Row As a UART

Set PM1[0] = 1 – Pmod1[4:1] are connected to the DW UART0 signals.

The UART signal mapping to the Pmod1[4:1] port is controlled by the `UART_MAP_CTRL` register.

This register is mapped to the peripheral register file (refer to Table 22).

UART_MAP_CTRL Register

Default	`0xE4`
Peripheral memory offset	`0xC`
Access	`UART_MAP_CTRL[7:0]` bits have R/W access

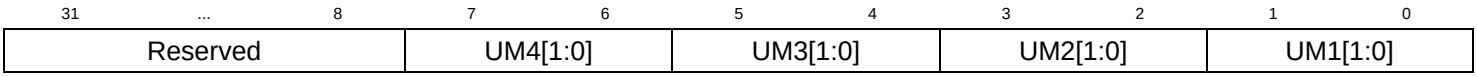


Table 5 Pin Assignment of Pmod1 (J1) Upper Row Depending on PM1[0]

Pmod1 Pins (J1)	PM1[0]=0	PM1[0]=1
	GPIO Signal	UART Signal

4	`Port C[11]`	`RTS_N if UM4="11" (default)`
		`RXD if UM4="10"`
		`TXD if UM4="01"`
		`CTS_N if UM4="00"`

Pmod1 Pins (J1)	PM1[0]=0	PM1[0]=1
	GPIO Signal	UART Signal
3	Port C[10]	RTS_N if UM3="11" RXD if UM3="10" (default) TXD if UM3="01" CTS_N if UM3="00"
2	Port C[9]	RTS_N if UM2="11" RXD if UM2="10" TXD if UM2="01" (default) CTS_N if UM2="00"
1	Port C[8]	RTS_N if UM1="11" RXD if UM1="10" TXD if UM1="01" CTS_N if UM1="00" (default)

UART mapping examples:

- Pmod Interface Type 4 UART

  Set UART_MAP_CTRL=0xE4 (Binary: 11 10 01 00)

  UM1 = "00": Pmod1[1] connected to CTS_N

  UM2 = "01": Pmod1[2] connected to TXD

  UM3 = "01": Pmod1[3] connected to RXD

  UM4 = "11": Pmod1[4] connected to RTS_N

- Pmod Interface Type 3 UART

  Set UART_MAP_CTRL=0x6C (Binary: 01 10 11 00)

  UM1 = "00": Pmod1[1] connected to CTS_N

  UM2 = "11": Pmod1[2] connected to RTS_N

  UM3 = "10": Pmod1[3] connected to RXD

  UM4 = "01": Pmod1[4] connected to TXD

Using Pmod1 (J1) Lower Row As an SPI Slave

Set PM1[2] = 1 – Pmod1[10:7] are connected to DW SPI Slave signals.

Table 6 Pin Assignment of Pmod1 (J1) Lower Row Depending on PM1[2]

Pmod1 Pins (J1)	PM1[2]=0	PM1[2]=1
	GPIO Signals	SPI Signals
10	Port A[11]	SPI SLV SCLK
9	Port A[10]	SPI SLV MISO
8	Port A[9]	SPI SLV MOSI
7	Port A[8]	SPI SLV CS_N

Pmod2 Configuration

PM2[3:0] – select signals connected to Pmod2 at the J2 connector.

Pins 3 and 4 are pulled up, while pins 1 and 7 are pulled down irrespective of the setting of PM2.

Note: PM2[3:1] are reserved.

Using Pmod2 (J2) as a GPIO Port

This is the default setting after a reset.

Set PM2[0] = 0 – Pmod2[4:1] are connected to DW GPIO Port C[15:12].

Pmod2[10:7] are connected to DW GPIO Port A[15:12].

Using Pmod2 (J2) as an I2C port and to access the ARC EM halt/run interface

Set PM2[0] = 1 – Pmod2[4:3] are connected to DW I2C_0 signals,

Pmod2[2:1] and Pmod2[8:7] are connected to the ARC EM halt/run interface.

In this mode the ARC EM halt/run interface is mapped to Pmod2 next to the I2C signals. The halt/run interface signals can be conveniently accessed at the header J10 (see the section *Headers J10, J11, J12*).

Table 7 Pin Assignment of Pmod2 (J2) Depending on PM2[0]

Pmod2 Pins (J2)	PM2[0]=0	PM2[0]=1
	GPIO Signals	Signals
10	Port A[15]	N/C
9	Port A[14]	N/C
8	Port A[13]	halt_ack
7	Port A[12]	halt_req

Pmod2 Pins (J2)	PM2[0]=0	PM2[0]=1
	GPIO Signals	Signals
4	Port C[15]	I2C_0 SDA
3	Port C[14]	I2C_0 SCL
2	Port C[13]	run_ack
1	Port C[12]	run_req

Pmod3 Configuration

PM3[3:0] – select signals connected to Pmod3 at the J3 connector.

Pins 1, 2, 3, 4, 7, and 8 are pulled up irrespective of the PM3[0] setting.

PM3[3:1] are reserved.

Using Pmod3 (J3) As a GPIO Port

This is the default setting after reset.

Set PM3[0] = 0 – Pmod3[4:1] are connected to DW GPIO Port C[19:16].

Pmod3[10:7] are connected to DW GPIO Port A[19:16].

Table 8 Pin Assignment of Pmod3 (J3) Depending on PM3[0]

Pmod3 Pins (J3)	PM3[0]=0	PM3[0]=1
	GPIO Signals	Signals
10	Port A[19]	Reserved
9	Port A[18]	Reserved
8	Port A[17]	Reserved
7	Port A[16]	Reserved
4	Port C[19]	Reserved
3	Port C[18]	Reserved
2	Port C[17]	Reserved
1	Port C[16]	Reserved

Pmod4 Configuration

PM4[3:0] – select signals connected to Pmod4 at the J4 connector.

Pins 1, 2, 3, 4, 7, and 8 are pulled up irrespective of the setting of PM4[0].

Note: PM4[3:1] are reserved.

Using Pmod4 (J4) as a GPIO Port

This is the default setting after a reset.

Set PM4[0] = 0 – Pmod4[4:1] are connected to DW GPIO Port C[23:20].
Pmod4[10:7] are connected to DW GPIO Port A[23:20].

Using Pmod4 (J4) as an I2C Interface

Set PM4[0] = 1 – Pmod4[4:3] are connected to DW I2C_1 signals,
Pmod4[2:1] are connected to DW GPIO Port D[5:4].
Pmod4[8:7] are connected to DW GPIO Port D[7:6].

In this mode, a few GPIO signals are mapped to Pmod4 next to the I2C signals. These GPIO signals can be conveniently accessed at the header J12 (see the section *Headers J10, J11, J12*).

Table 9 Pin Assignment of Pmod4 (J4) Depending on PM4[0]

Pmod4 pins (J4)	PM4[0]=0	PM4[0]=1
	GPIO Signals	Signals
10	Port A[23]	N/C
9	Port A[22]	N/C
8	Port A[21]	Port D[7]
7	Port A[20]	Port D[6]
4	Port C[23]	I2C_1 SDA
3	Port C[22]	I2C_1 SCL
2	Port C[21]	Port D[5]
1	Port C[20]	Port D[4]

Pmod5 Configuration

PM5[3:0] – select signals connected to Pmod5 at the J5 connector.

Pins 3, 7 and 9 are pulled up irrespective of the setting of PM5.

Note: PM5[1] and PM5[3] are reserved.

Using Pmod5 (J5) As a GPIO Port

This is the default setting after a reset.

Set PM5[0] = 0 – Pmod5[4:1] are connected to DW GPIO Port C[27:24].

Set PM5[2] = 0 – Pmod5[10:7] are connected to DW GPIO Port A[27:24].

Using Pmod5 (J5) As an SPI Master (Four-Pin Interfaces Using CS1_N)

Set PM5[0] = 1 – Pmod5[4:1] are connected to DW SPI Master signals using CS1_N.

This allows you to connect an SPI slave to the DW SPI Master. Additional SPI slaves can be connected to the same DW SPI Master at Pmod6.

Using Pmod5 (J5) As a UART

Set PM5[2] = 1 – Pmod5[10:7] are connected to the DW UART2 signals.

Table 10 Pin Assignment of Pmod5 (J5) Upper Row Depending on PM5[0]

Pmod5 Pins (J5)	PM5[0]=0	PM5[0]=1
	GPIO Signals	SPI Signals
4	Port C[27]	SPI MST SCLK
3	Port C[26]	SPI MST MISO
2	Port C[25]	SPI MST MOSI
1	Port C[24]	SPI MST CS1_N

Table 11 Pin Assignment of Pmod5 (J5) Lower Row Depending on PM5[2]

Pmod5 Pins (J5)	PM5[2]=0	PM5[2]=1
	GPIO Signals	UART Signals
10	Port A[27]	UART2 RTS_N
9	Port A[26]	UART2 RXD
8	Port A[25]	UART2 TXD
7	Port A[24]	UART2 CTS_N

Pmod6 Configuration

PM6[3:0] – select signals connected to Pmod6 at J6 connector.

Pin 3 is pulled up irrespective of the setting of PM6.

Note: PM6[1] and PM6[3] are reserved.

Using Pmod6 (J6) As a GPIO Port

Set PM6[0] = 0 – Pmod6[4:1] are connected to DW GPIO Port C[31:28].

Set PM6[2] = 0 – Pmod6[10:7] are connected to DW GPIO Port A[31:28].

Using Pmod6 (J6) As an SPI Master (Four-Pin Interface with One Chip Select CS0_N)

Set PM6[0] = 1 – Pmod6[4:1] are connected to DW SPI Master signals using CS0_N.

This allows you to connect an SPI slave to the DW SPI Master. Additional SPI slave can be connected to the same DW SPI Master at Pmod5. Alternatively, two additional chip select signals can be made available at the lower row of Pmod6.

Using Pmod6 (J6) As an SPI Master (Seven-Pin Interface with Three Chip Selects)

Set PM6[0] = 1 – Pmod6[4:1] are connected to DW SPI Master signals using CS0_N.

Set PM6[2] = 1 – Pmod6[8:7] are connected to the DW SPI Master chip select signals CS1_N and CS2_N

Pmod6[6:5] are connected to the ARC EM halt and sleep status signals

This allows connecting three SPI slaves to the DW SPI Master. In this case do not use Pmod5 as an SPI Master, because it uses the same chip-select signals.

Using Pmod6 (J6) to Monitor the ARC EM Halt and Sleep Status Signals

Set *PM6[2] = 1* – Pmod6[8:7] are connected to the DW SPI Master chip select signals CS1_N and CS2_N

Pmod6[6:5] are connected to the ARC EM halt and sleep status signals

Table 12 Pin Assignment of Pmod6 (J6) Upper Row Depending on PM6[0]

Pmod6 Pins (J6)	PM6[0]=0	PM6[0]=1
	GPIO Signals	SPI Signals
4	Port C[31]	SPI MST SCLK
3	Port C[30]	SPI MST MISO
2	Port C[29]	SPI MST MOSI
1	Port C[28]	SPI MST CS0_N

Table 13 Pin Assignment of Pmod6 (J6) Lower Row Depending on PM6[2]

Pmod6 Pins (J6)	PM6[2]=0	PM6[2]=1
	GPIO Signals	Signals
10	Port A[31]	sleep
9	Port A[30]	halt
8	Port A[29]	SPI MST CS2_N
7	Port A[28]	SPI MST CS1_N

Pmod7 Configuration

PM7[1:0] – select signals connected to Pmod7 at J20 connector.

Note: PM7[1] is reserved.

Using Pmod7 (J20) As a GPIO Port

Set PM7[0] = 0 – Pmod7[4:1] are connected to DW GPIO Port D[11:8].

Using Pmod7 (J20) to Monitor the ARC EM Sleep Status Signals

Set PM7[0] = 1 - Pmod7[4:1] are connected to the ARC EM sleep status signals.

Table 14 Pin Assignment of Pmod7 (J7) Depending on PM7[0]

Pmod7 Pins (J20)	PM7[0]=0	PM7[0]=1
	GPIO Signals	Signals
4	Port D[11]	sleep
3	Port D[10]	sleep_mode[2]
2	Port D[9]	sleep_mode[1]
1	Port D[8]	sleep_mode[0]



Note Pmod7 is implemented using staggered drill-holes. You can plug a standard six-pin Pmod cable into the drill-holes using the six-pin header and gender changer that is typically provided with the Pmod cable. The staggering of the drill-holes ensures a tight and reliable connection. If desired, you can also solder a 1x6 header or a 1x6 female connector.

Pmods Configuration Summary

Table 15 Pmods Configuration Summary

Pmod1 Pins (J1)	PM1[2]=0	PM1[2]=1
10	Port A[11]	SPI SLV SCLK
9	Port A[10]	SPI SLV MISO
8	Port A[9]	SPI SLV MOSI
7	Port A[8]	SPI SLV CS_N
Pmod1 Pins (J1)	**PM1[0]=0**	**PM1[0]=1**
4	Port C[11]	UART0 (Table 5)
3	Port C[10]	UART0 (Table 5)
2	Port C[9]	UART0 (Table 5)
1	Port C[8]	UART0 (Table 5)
Pmod2 Pins (J2)	**PM2[0]=0**	**PM2[0]=1**
10	Port A[15]	N/C
9	Port A[14]	N/C
8	Port A[13]	halt_ack
7	Port A[12]	halt_req
4	Port C[15]	I2C_0 SDA
3	Port C[14]	I2C_0 SCL
2	Port C[13]	run_ack
1	Port C[12]	run_req
Pmod3 Pins (J3)	**PM3[0]=0**	**PM3[0]=1**
10	Port A[19]	Reserved
9	Port A[18]	Reserved
8	Port A[17]	Reserved
7	Port A[16]	Reserved
4	Port C[19]	Reserved
3	Port C[18]	Reserved
2	Port C[17]	Reserved
1	Port C[16]	Reserved

Pmod4 pins (J4)	PM4[0]=0	PM4[0]=1
10	Port A[23]	N/C
9	Port A[22]	N/C
8	Port A[21]	Port D[7]
7	Port A[20]	Port D[6]
4	Port C[23]	I2C_1 SDA
3	Port C[22]	I2C_1 SCL
2	Port C[21]	Port D[5]
1	Port C[20]	Port D[4]
Pmod5 Pins (J5)	**PM5[2]=0**	**PM5[2]=1**
10	Port A[27]	UART2 RTS_N
9	Port A[26]	UART2 RXD
8	Port A[25]	UART2 TXD
7	Port A[24]	UART2 CTS_N
Pmod5 Pins (J5)	**PM5[0]=0**	**PM5[0]=1**
4	Port C[27]	SPI MST SCLK
3	Port C[26]	SPI MST MISO
2	Port C[25]	SPI MST MOSI
1	Port C[24]	SPI MST CS1_N
Pmod6 Pins (J6)	**PM6[2]=0**	**PM6[2]=1**
10	Port A[31]	sleep
9	Port A[30]	halt
8	Port A[29]	SPI MST CS2_N
7	Port A[28]	SPI MST CS1_N
Pmod6 Pins (J6)	**PM6[0]=0**	**PM6[0]=1**
4	Port C[31]	SPI MST SCLK
3	Port C[30]	SPI MST MISO
2	Port C[29]	SPI MST MOSI
1	Port C[28]	SPI MST CS0_N

Pmod7 Pins (J20)	PM7[0]=0	PM7[0]=1
4	Port D[11]	sleep
3	Port D[10]	sleep_mode[2]
2	Port D[9]	sleep_mode[1]
1	Port D[8]	sleep_mode[0]

Headers J10, J11, J12

The Pmod connectors J2 and J4 can be used to plug in peripheral modules with a Pmod I2C interface, which uses only eight out of 12 available Pmod pins.

According to the Pmod signal mapping described in the *Pmod Pin Configuration* section, four pins that are not used for the I2C interface are available for other purposes, such as GPIO or ARC EM control signals. However, when an I2C Pmod module is inserted into the Pmod connector on the board, physical access to the other four Pmod pins might be difficult, depending on the design of the I2C module. To facilitate access to these signals, they are also connected to separate headers J10, J11, and J12.

The pins 5, 6, 7, 8 of J10, J11, and J12 are connected to the ground. Therefore, you can put jumpers in place to tie any signals on pins 1, 2, 3, or 4 to the ground.

Figure 22 Pin Positions at Headers J10, J11, and J12

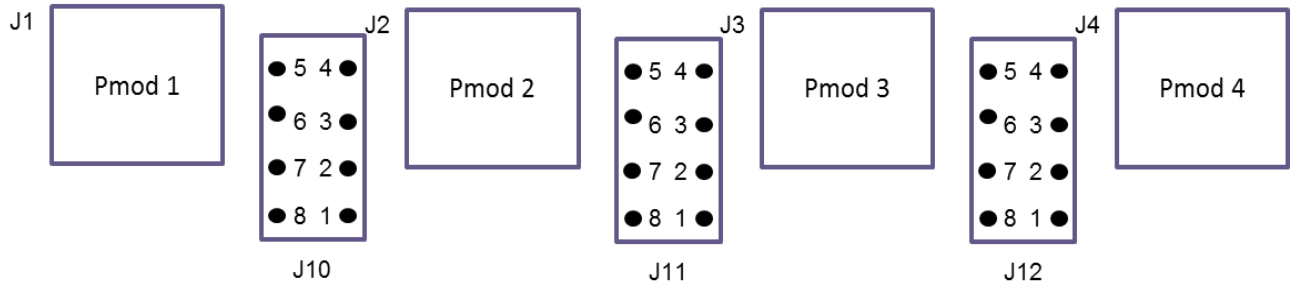


Table 16 lists the pin assignment at the header J10 depending on bit PM2[0] of the PMOD_MUX_CTRL register.

Table 16 Pin Assignment of the Header J10 Depending on PM2[0]

J10	PM2[0]=0	PM2[0]=1
	GPIO Signals	ARC EM Signals
1	Port C[12]	run_req
2	Port A[12]	halt_req
3	Port A[13]	halt_ack

J10	PM2[0]=0	PM2[0]=1
	GPIO Signals	ARC EM Signals
4	Port C[13]	run_ack
5	GND	
6		
7		
8		
9		

The following pin assignment is used at the header J11 depending on bit PM3 of the PMOD_MUX_CTRL Register:

Table 17 Pin Assignment of the Header J11 Depending on PM3[0]

J11	PM3[0]=0	PM3[0]=1
	GPIO Signals	GPIO Signals
1	Port C[16]	Reserved
2	Port A[16]	Reserved
3	Port A[17]	Reserved
4	Port C[17]	Reserved
5	GND	
6		
7		
8		

The following pin assignment is used at the header J12 depending on bit PM4 of the PMOD_MUX_CTRL Register.

Table 18 Pin Assignment of the Header J12 Depending on PM4[0]

J12	PM4[0]=0	PM4[0]=1
	GPIO Signals	GPIO Signals
1	Port C[20]	Port D[4]
2	Port A[20]	Port D[6]
3	Port A[21]	Port D[7]
4	Port C[21]	Port D[5]

J12	PM4[0]=0	PM4[0]=1
	GPIO Signals	GPIO Signals
5	GND	
6		
7		
8		

Peripheral Controllers

This section describes peripheral devices included in the FPGA project of the ARC EM Starter Kit.

The following peripheral devices are available:

- SPI master
- SPI slave
- I2C
- UART
- GPIO
- WDT
- Timers

The following DesignWare IP components were used to implement the peripherals:

- `DW_apb_uart` – Universal Asynchronous Receiver/Transmitter
- `DW_apb_gpio` – General Purpose Programmable I/O
- `DW_apb_ssi` – Synchronous Serial Interface (SPI)
- `DW_apb_i2c` – I2C interface
- `DW_apb_wdt` – WDT
- `DW_apb_timers` – Timers

General information about DesignWare components can be found at http://www.synopsys.com/IP/Pages/default.aspx.

GPIO

Basic GPIO functionality is provided with the Synopsys DesignWare GPIO IP (`DW_apb_gpio`) [4].

This IP controls push buttons, LEDs, and DIP switches and provides the GPIO functionality of Pmod connectors.

The Pmod connectivity is defined by control register settings described in the Connecting Peripheral Controllers section.

The `DW_apb_gpio` is configured to have the following 4 ports:

- Port A width is 32 bits
- Port B width is 9 bits
- Port C width is 32 bits
- Port D width is 12 bits

Port A includes debounce logic and supports interrupt processing. It generates a single, combined interrupt connected to the ARC EM interrupt controller port `irq_22`.

A detailed description of the peripheral can be found in the *DesignWare DW_apb_gpio Databook* [4].

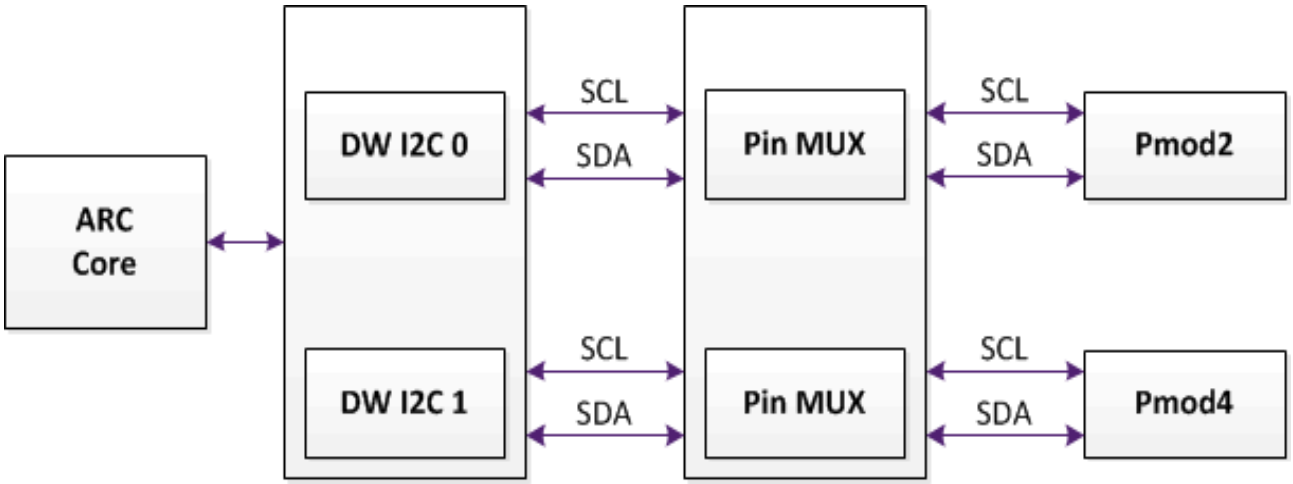
GPIO Controller registers are mapped to the peripheral memory (see *Table 23*).

I2C

Two instances of an I2C peripheral are available. Each of them can be programmed at run time either as a master or slave device. It supports I2C standard and fast mode (up to 400 kHz).

Master mode allows multiple I2C slaves to be connected. Two PMOD connectors (Pmod 2 or Pmod4) are available for direct connection of up to two I2C slaves to the I2C masters; the Pmod connectivity is defined with the control register settings as described in the Connecting Peripheral Controllers section. If needed, additional slaves can be connected using external breadboards that support the necessary wiring to hook up multiple slaves to a single Pmod interface. Figure 23 shows the interconnection between `dw_apb_i2c` and the Pmod connectors.

Figure 23 DW I2C Connection



Signals of I2C 0 controller may be mapped to Pmod2, signals of I2C 1 controller may be mapped to Pmod4. When some Pmod connector is not selected by the pin mux controller, the corresponding signals are connected to high level.

In slave mode, I2C slave address is programmable at run-time. The default I2C slave address is 0x55.

The I2C functionality is based on the Synopsys DesignWare I2C IP (`DW_apb_i2c`). A detailed description can be found in the *DesignWare DW_apb_i2c Databook* [5].

I2C Controller registers are mapped to the peripheral memory (refer to *Table 23*).

SPI Master

The SPI Master has six chip-select outputs. Therefore, it can control up to six SPI devices:

- On-board SPI flash memory
- On-board SD Card
- Internal SPI Slave
- Up to three SPI peripherals connected to Pmod connectors

The SPI Master is implemented using the Synopsys DesignWare `DW_apb_ssi` IP component. Its detailed description can be found in the *DW_apb_ssi Databook* [6].

The SPI Master uses a system clock (`pclk`) and an additional peripheral clock (`ssi_clk`) for the SPI interface. The system clock corresponds with the EM core frequency. The peripheral clock is always 50 MHz.

The SPI Master control registers are mapped to the peripheral memory (refer to *Table 23*).

Note The `DW_apb_ssi` IP supports other serial protocols too. However, the IP configuration selected for the ARC EM Starter Kit supports SPI mode only.

The IP contains separate receive and transmit FIFOs, which have a depth of 32 words each. The width of both transmit and receive FIFO buffers is fixed at 16 bits due to the serial specifications, which state that a serial transfer (data frame) can be from four to 16 bits in length.

Data frames that are less than 16 bits in length must be right-justified when written into the transmit FIFO buffer. The shift control logic automatically right-justifies received data in the receive FIFO buffer. The device supports all four SPI modes (clock polarity and clock phase), which can be programmed at run-time.

The maximum achievable SPI clock frequency is 25 MHz.

The chip-select assignment of the SPI Master is described in Table 19.

Table 19 SPI Master Signals Usage

Chip Select Name	Devices
CS0	Pmod 6 pin 1 (connector J6)
CS1	Pmod 5 pin 1 (connector J5), or Pmod 6 pin 7(connector J6)
CS2	Pmod 6 pin 8 (connector J6)
CS3	On-board SD card
CS4	Internal SPI Slave (when register `SPI_MAP_CTRL[0]=1`)
CS5	On-board SPI Flash memory

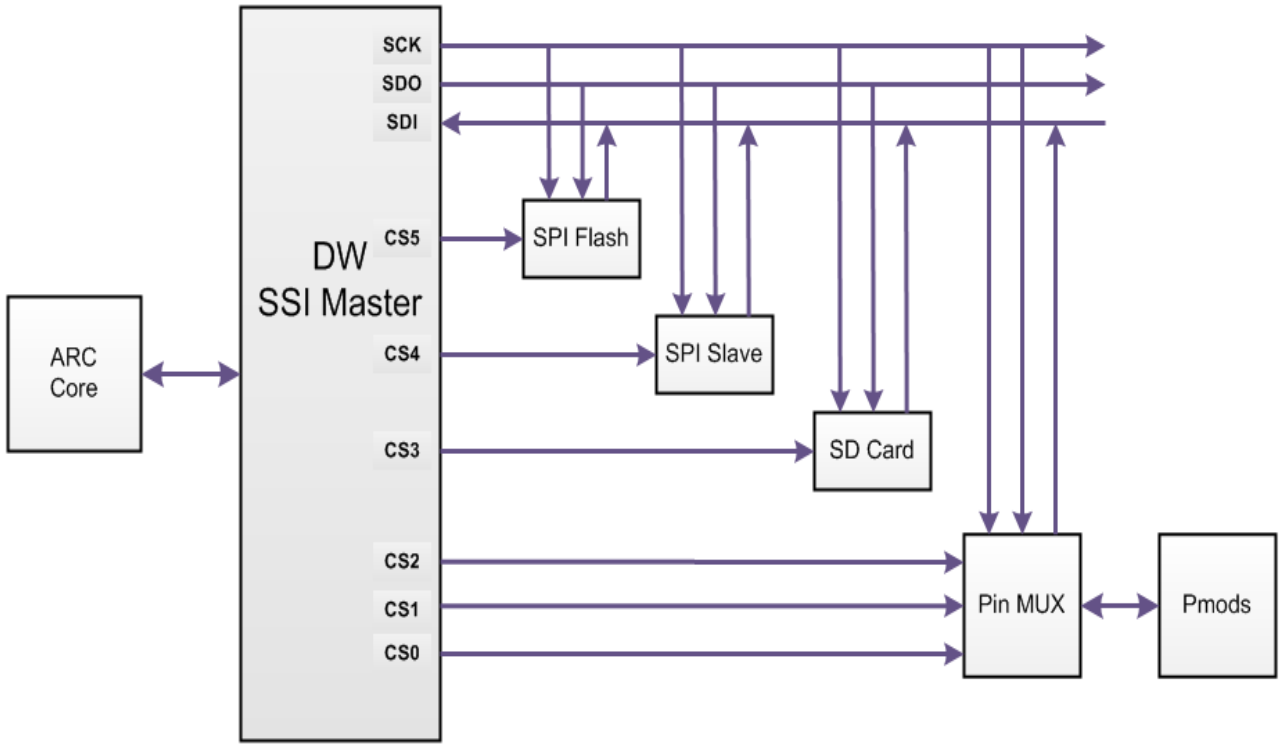
Each Chip Select is masked with the SPI Master Chip Select Control register (SPI_MST_CS_CTRL), which is mapped to the peripheral memory (for more details, see Table 23).

Table 20 describes the SPI_MST_CS_CTRL bits.

Table 20 SPI_MST_CS_CTRL Register Bits Definition

Bit	Default Value	Description
0	0	Forces CS0 to 0 when bit is 1, else pass CS0 to device
1	0	Forces CS1 to 0 when bit is 1, else pass CS1 to device
2	0	Forces CS2 to 0 when bit is 1, else pass CS2 to device
3	0	Forces CS3 to 0 when bit is 1, else pass CS3 to device
4	0	Forces CS4 to 0 when bit is 1, else pass CS4 to device
5	0	Forces CS5 to 0 when bit is 1, else pass CS5 to device
31:6	0	Reserved

Figure 24 DW SPI Master Connection



The Pmod multiplexer allows using the SPI master in two different ways: multiple Pmod SPI mode and pin-count optimized mode.

Multiple Pmod SPI Mode

Each SPI slave has its own Pmod SPI compliant interface with individual clock, data, and chip-select signals. The SPI master generates a single clock and a single data output. These signals are branched to multiple Pmod connectors. The data inputs from the slaves are multiplexed into one signal. This mode is controlled by the chip-select outputs of the master.

Use this mode if you want to connect one or several Pmod-compliant SPI modules in a convenient way. You can directly connect the Pmod SPI modules to the Pmod connectors on the board.

To select this mode, go to the register `PMOD_MUX_CTRL` and set `PM6[0]=1`, `PM5[0]=1` and/or `PM5[2]=1` depending on the desired number and location of SPI interfaces.

Pin-Count Optimized Mode

All SPI slaves share the clock and data signals. However, they have individual chip-select signals. All SPI signals are located in a single Pmod connector, namely Pmod6 at connector J6.

Use this mode if you want to connect extension boards with multiple SPI slaves. This is typically done using wires as this is not a standard Pmod SPI interface.

To select this mode, go to register `PMOD_MUX_CTRL` and set `PM6[0]=1` and `PM6[2]=1`.

Note For more information on the location of individual SPI signals on the Pmod connectors, refer to the Pmod Pin Configuration section.

Micro-DMA Support for SPI Master

The core micro-DMA (µDMA) controller is configured for two channels, with two byte deep FIFOs. The assignment of each of the channels is described in Table 21.

Table 21 DMA Channel Assignment

DMA Channel	Peripheral
1	SPI master TX
2	SPI master RX

SPI Slave

The SPI Slave is implemented using the Synopsys DesignWare DW_apb_ssi IP component. Refer to the *DW_apb_ssi Databook* [6] for a detailed description.

The SPI slave uses a system clock (`pclk`) and a peripheral clock (`ssi_clk`) for the SPI interface. The system clock corresponds with the EM core frequency and the peripheral clock is 50 MHz.

The SPI Slave control registers are mapped to the peripheral memory (refer to Table 23).

Note The `DW_apb_ssi` IP supports other serial protocols. However, the configuration selected for the ARC EM Starter Kit supports SPI mode only.

The IP contains separate receive and transmit FIFOs, which have a depth of 32 words each. The width of both transmits and receives FIFO buffers is fixed at 16 bits due to the serial specifications, which state that a serial transfer (data frame) can be four to 16 bits in length.

Data frames that are less than 16 bits in length must be right-justified when written into the transmit FIFO buffer. The shift-control logic automatically right-justifies receive data in the receive FIFO buffer. The device supports all four SPI modes (clock polarity and clock phase), which can be programmed at run-time.

If the SPI slave is in transmit only mode, the maximum achievable SPI clock frequency is 6.25 MHz.

If the SPI slave is in transmit and receive mode, the maximum achievable SPI clock frequency is 5 MHz.

UART

The UART is implemented using the Synopsys DesignWare UART IP (DW_apb_uart). Refer to *Synopsys DesignWare DW_apb_uart Databook* [8] for more information (including a register description).

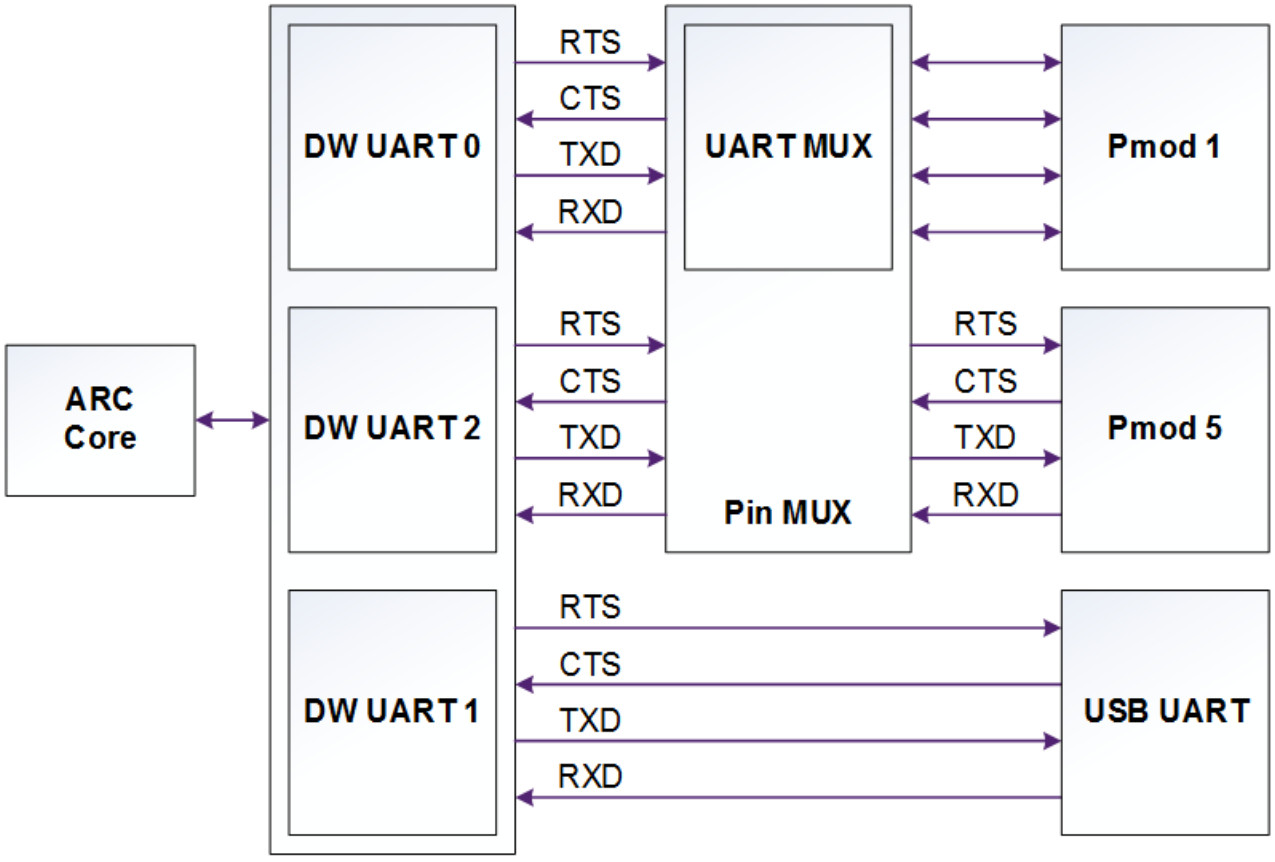
The UART Controller uses a system clock (`pclk`) and a peripheral clock (`sclk`) for the UART interface. The system clock corresponds with the EM core frequency. The peripheral clock is always 50 MHz.

UART Controller registers are mapped to the peripheral memory (refer to Table 23).

The IP contains FIFOs, which have a depth of 32 words. The device supports auto flow control.

The FPGA design includes three UART instances; one of them is connected to the USB Debug Console as shown in the Figure 20. The corresponding USB / UART Interfaces are located on the board.

Figure 25 DW UART Components Connection



Timers

The Timers are implemented using the Synopsys DesignWare Timers IP (DW_apb_timers). Refer to *Synopsys DesignWare DW_apb_timers Databook* [7] for more information (including a register description).

The Timers Controller uses a system clock (`pclk`). It corresponds with the EM core frequency.

Timers controller registers are mapped to the peripheral memory (refer to Table 23).

WDT

The WDT is implemented using the Synopsys DesignWare WDT IP (DW_apb_wdt). Refer to *Synopsys DesignWare DW_apb_wdt Databook* [9] for more information (including a register description).

The WDT controller uses a system clock (`pclk`) and a peripheral clock (`tclk`) as watchdog timer clock. The system clock corresponds with the EM core frequency. The peripheral clock is always 50 MHz.

The WDT controller registers are mapped to the peripheral memory (refer to Table 23).

CREG Controller

Control registers are located starting from address 0xF0000000.

The pin mux controller connects the peripheral controllers to external hardware interfaces and to each other. The Pin Mux Controller registers are mapped to the peripheral memory starting from the pin mux controller base address (refer to Table 23). Registers are listed in Table 22.

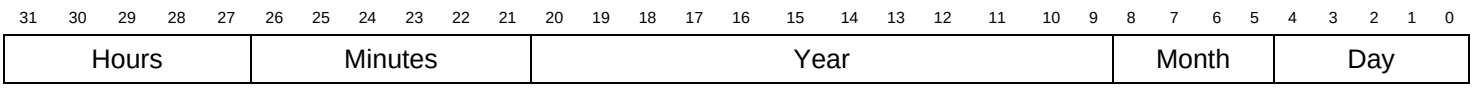
Table 22 Register File Mapping

Offset	Register Name	Default Value	Description
0x0	PMOD_MUX_CTRL	0x0	This register controls mapping of the peripheral device signals on Pmod connectors. Refer to the Pmod Pin Configuration section for more details.
0x4	I2C_MAP_CTRL	0x0	Reserved for future extensions
0x8	SPI_MAP_CTRL	0x0	SPI_MAP_CTRL[0] selects the mode of operation of the SPI Slave: Normal operation, SPI_MAP_CTRL[0]=0: SPI Slave is connected to Pmod1 at connector J1. Loop-back mode, SPI_MAP_CTRL[0]=1: SPI Slave is connected to the SPI Master inside the FPGA using CS4. This mode is for test purposes only.
0xC	UART_MAP_CTRL	0xE4	This register controls mapping of the UART signals on the Pmod1 connector. See the Pmod Pin Configuration section for more details.
0x10	Reserved		Reserved
0x14	SPI_MST_CS_CTRL	0x0	SPI Master select control. For more details, see Table 20.
0x18	BUILD_DATE		Build date and time
0x1C	BUILD_VERSION		Build version

BUILD_DATE Register

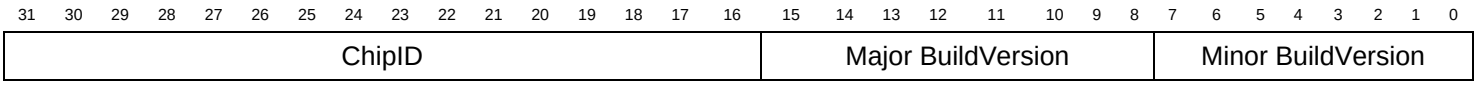
Default	–
Peripheral memory offset	0x18
Access	Read-only

This register defines the date and time of the particular configuration.



BUILD_Version Register

Default	–
Peripheral memory offset	0x1C
Access	Read-only



ChipID defines the configuration of the core:

ChipID	Configuration
1	EM7D
2	EM9D
3	EM11D

Major BuildVersion is 0x2 for the EM StarterKit 2.2

Minor BuildVersion is 0x2 for the EM StarterKit 2.2

Peripheral Memory Mapping

The peripheral memory mapping depends on the AHB address width. The configurations use an address width of 32 bits.

Table 23 Peripheral Memory Mapping

Name	Address Width: 32 Bits		Size
	Start	End	
CREG Controller	0xF0000000	0xF0000FFF	4KB
GPIO	0xF0002000	0xF0002FFF	4KB
Timers	0xF0003000	0xF0003FFF	4KB
I2C_0	0xF0004000	0xF0004FFF	4KB
I2C_1	0xF0005000	0xF0005FFF	4KB
SPI Master	0xF0006000	0xF0006FFF	4KB
SPI Slave	0xF0007000	0xF0007FFF	4KB
UART0	0xF0008000	0xF0008FFF	4KB
UART1	0xF0009000	0xF0009FFF	4KB
UART2	0xF000A000	0xF000AFFF	4KB
WDT	0xF000B000	0xF000BFFF	4KB
Reserved	0xF000C000	0xF000CFFF	4KB
Reserved	0xF000D000	0xF000DFFF	4KB
Reserved	0xF000E000	0xF000FFFF	8KB

Interrupts Connections

Table 24 Interrupts Connections

Interrupt	Component
irq_0…irq_15	Internal EM core interrupts
irq_16	Core Timer 0
irq_17	Core Timer 1

Interrupt	Component
irq_20	Core Secure Timer 0[1]
irq_22	Core DMA controller[1]
irq_23	Core DMA controller[1]
irq_24	GPIO controller
irq_25	I2C_0 controller
irq_26	I2C_1 controller
irq_27	SPI Master controller
irq_28	SPI Slave controller
irq_29	UART0
irq_30	UART1
irq_31	UART2
irq_32	WDT
irq_33	Timer 0
irq_34	Timer 1
irq_35	Reserved
irq_36	Reserved
irq_37	Reserved
Note: 1 - Only available for EM7D + ESP base configuration	

On-Board Devices

JTAG Connector

It is recommended to use the USB interface for JTAG debugging. Alternatively, it is possible to use industry-standard debug probes connected to the 20-pin JTAG connector J15.

The ARC EM Starter Kit supports a standard four-wire JTAG interface. The two-wire IEEE 1149.7 C JTAG is not supported.

Refer to the Appendix D for examples of using JTAG debuggers.

USB

The mini-USB port can be used for several purposes:

- ARC JTAG debugging port

- UART debug console
- FPGA bitfile programming

Put jumper J8 in place for FPGA programming and remove it when the USB port is used for ARC JTAG or as a UART debug console.

Refer to FPGA Image Recovery for details on how to install and use the Digilent Adept and Xilinx iMPACT software.

SD Card

The SD card interface works in SPI mode. It is connected to channel 3 of the SPI Master.

Man-Machine Interface

The following components are available on the ARC EM Starter Kit board:

- Push buttons
- DIP-switches
- Jumpers
- LEDs

Push Buttons

Name	Location	Description
A	Above letter “A” of the ARC logo	ARC start button when processor is halted. Additionally, this button is connected to DW GPIO Port A[2]. Debouncing is implemented inside the DW GPIO IP. Note that when the processor is halted (for example, for debugging), pressing this button continues execution; pay special attention to this button in debug mode; in general mode it may be used by the user application same way as the 'L' and 'R' buttons.
Reset	Above letter “R” of the ARC logo	ARC reset
C	Above letter “C” of the ARC logo	Configure button. Pushing this button has the same effect as a power-on reset and triggers the FPGA to re-load the bitfile from the serial flash memory.
L	Marked 'L'	User button. This button is connected to DW GPIO Port A[0]. Debouncing is implemented inside the GPIO IP.
R	Marked 'R'	User button. This button is connected to DW GPIO Port A[1]. Debouncing is implemented inside the GPIO IP.

Note: three push buttons (A, L, R) are available for user purposes.

DIP-Switches

The four-bit DIP-switch SW1 is used for the following purposes:

1. Controlling input ports for user GPIOs
   - Switch 1 is connected to GPIO Port C[0].
   - Switch 2 is connected to GPIO Port C[1].
   - Switch 3 is connected to GPIO Port C[2].
   - Switch 4 is connected to GPIO Port C[3].
2. Selection of the predefined FPGA image to be loaded at power-on or after pressing the configuration button ‘C‘ (Switch 1 and Switch 2) as described in the section Select ARC EM Configurations .
3. Enabling a Self-test application (Switch 3) as described in the section Bootloader and Initial Self-Test
4. Running user applications from SPI flash (Switch 4) as described in the section Running Applications in Self-Boot Mode.


Note If switches 1 or 2 are used for user applications, ensure that they are switched to a proper position for the predefined FPGA image, before board power on or before the configuration button ’C’ is pressed.

Jumpers

Refer to *ARC EM Starter Kit Getting Started* for more information.

LEDs

Nine LEDs can be used as output ports for user GPIOs. The LEDs LED0 to LED8 are located below the ARC Synopsys logo on the board. LED8 is located next to the DIP switch SW1.

LED GPIO mapping:

- GPIO Port B[0] controls LED0
- GPIO Port B[1] controls LED1
- ...
- GPIO Port B[8] controls LED8

If the LED is on, its corresponding control bit is programmed to `0`.

After a reset, the LEDs indicate the selected ARC EM configuration and display the result of a self-test.

Power Supply

Use a universal power adapter (110-240 Volts AC to 5 Volts DC) from the package. Connect the appropriate AC plug for your AC power outlet. Connect the DC plug to the connector J13 “5V DC” on the board. Finally, connect the AC plug to your AC power outlet.

Troubleshooting

The following message may appear when you try to connect using the MetaWare debugger:

```
[DIGILENT] Device enumeration: #0 is `TE0604-02'=JTAG-ONB4.
[DIGILENT] We choose device  : #0 `TE0604-02' from 1 possible
devices.
[DIGILENT] Product=508 variant=1 fwid=57 firmware-version=108.
[DIGILENT] It is possible to set the JTAG speed.
[DIGILENT] Current speed is 10000000 Hz.
[JTAG] ARC 1 does not exist; the chain has 0 ARCs.
Download failed (2).
No process.
Exiting due to error and because exit_on_error=1.
C:\ARC\MetaWare\arc/bin/cld: Permission denied
```

This means that the J8 jumper was left after programming the FPGA. Remove it to eliminate this message.

There is no output in the PuTTY console window

Make sure that you are using the correct PuTTY console configuration described in the View Self-Test Output on PuTTY Console on page 17 and correct console settings for the `dw_apb_uart` application.

```
Baudrate=115200 Kb/s
Bits=8
Parity=None
Stop=1
```

iMPACT tool does not see Digilent USB JTAG Cable and cannot initialize chain when you do step shown in the Figure 31:

Install drivers from the installed Xilinx LabTools:

```
c:\Xilinx\14.7\LabTools\common\bin\nt\digilent\
install_digilent.exe for 32 bit Windows
c:\Xilinx\14.7\LabTools\common\bin\nt\digilent\
install_digilent.exe for 64 bit Windows 1
```

Appendix: B
ARC EM Configurations

This chapter is intended for programmers of the ARC EM Starter Kit. It explains how to use the ARC EM Starter Kit software development

Programmer's Model

The ARC EM Starter Kit provides three different ARC EM core configurations that can be used for running applications on a 25 MHz CPU clock for ARC_EM7D configuration, 20 MHz clock for ARC_EM9D and ARC_EM11D.

The detailed configurations of the three cores are described in Detailed Core Configurations.

For information about the ARC EM architecture, see the *ARCv2 ISA Programmer's Reference* [12].

For information about building and running applications on the target platform, see the *C/C++ Programmer's Guide for the MetaWare Compiler* [13].

Core Configurations and Memory Mapping

From a programmer's point of view, the ARC EM Starter kit provides following memory maps:

- ARC_EM7D Configuration
- ARC_EM9D Configuration
- ARC_EM11D Configuration

ARC_EM7D Configuration

This is an ARC EM core with 32 bits of address space, 256 KB of code memory (ICCM) and 128 KB of data memory (DCCM).

The MetaWare compiler options for this configuration are:

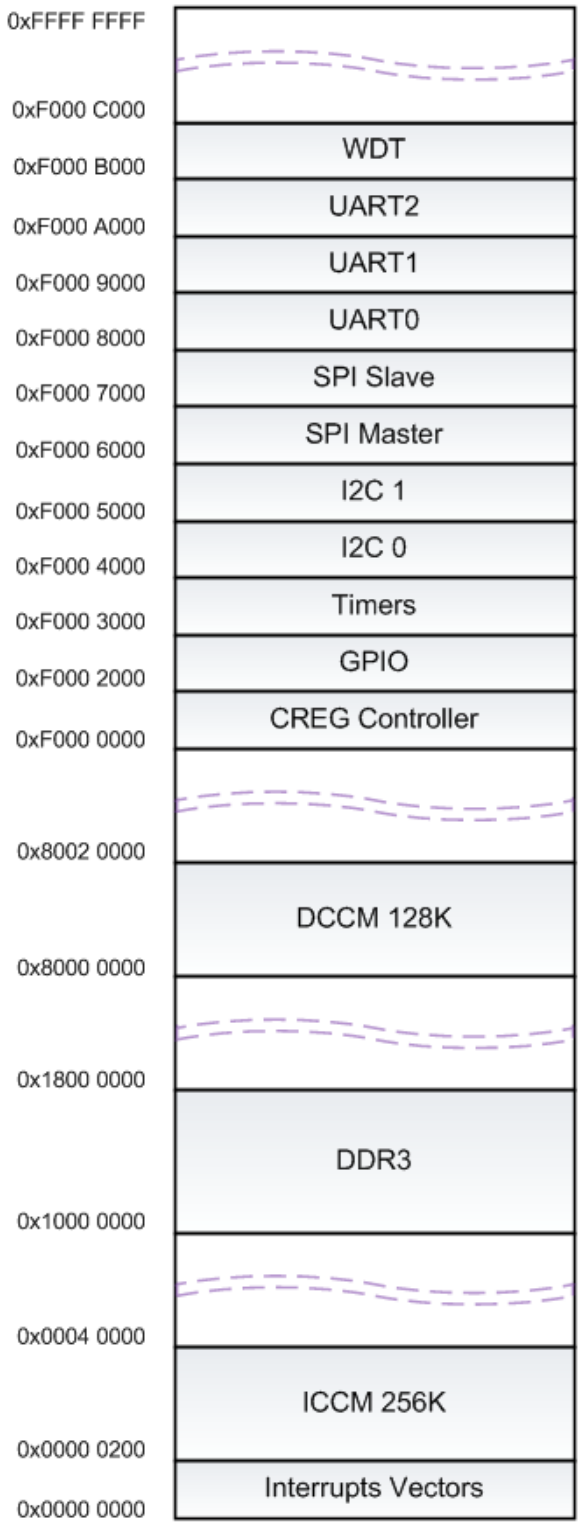
```
-arcv2em -core2 –HL -Xcode_density -Xdiv_rem=radix2 –Xswap
–Xbitscan -Xmpy_option=mpyd -Xshift_assist -Xbarrel_shifter -Xdsp2
-Xdsp_complex -Xdsp_divsqrt=radix2 -Xdsp_accshift=limited -Xtimer0
-Xtimer1 -Xstack_check –Hccm -Xdmac
```

The GCC compiler options for this configuration are:

```
-mcpu=arcem -mlittle-endian -mcode-density -mdiv-rem -mswap –mnorm
-mmpy-option=6 -mbarrel-shifter --param l1-cache-size=16384 --param
l1-cache-line-size=32
```

The memory map is shown in Figure 26.

Figure 26 Memory Map of ARC_EM7D Configuration



ARC_EM9D Configuration

This is an ARC EM core with 32 bits of address space, 256 KB of code memory (ICCM) and 128 KB of data memory (DCCM).

The MetaWare compiler options for this configuration are:

ARC_EM9D:

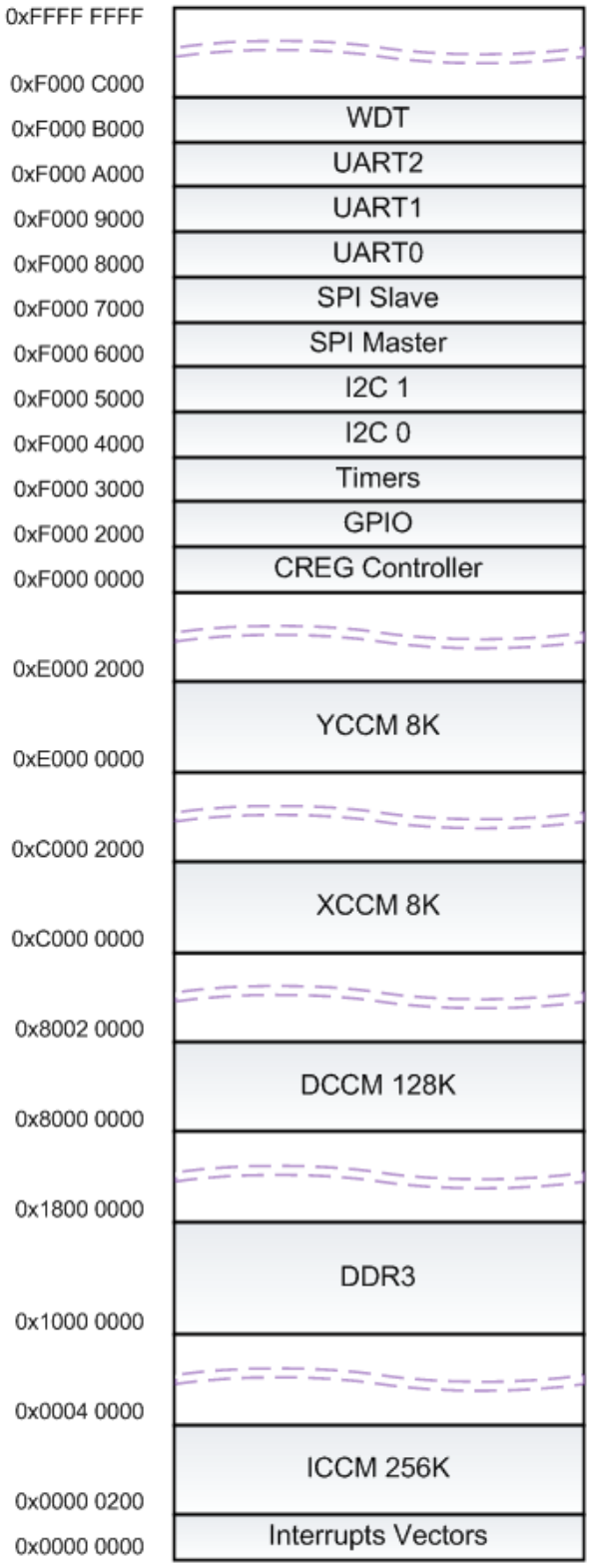
```
-arcv2em -core2 -Hrgf_banked_regs=32 -HL -Xcode_density
-Xdiv_rem=radix2 -Xswap -Xbitscan -Xmpy_option=mpyd -Xshift_assist
-Xbarrel_shifter -Xdsp2 -Xdsp_complex -Xdsp_divsqrt=radix2 -Xdsp_itu
-Xdsp_accshift=full -Xagu_large -Xxy -Xbitstream -Xfpus_div
-Xfpu_mac -Xfpus_mpy_slow -Xfpus_div_slow -Xtimer0 -Xtimer1
-Xstack_check -Hccm -Xdmac
```

The GCC compiler options for this configuration are:

```
-mcpu=arcem -mlittle-endian -mcode-density -mdiv-rem -mswap –mnorm
-mmpy-option=6 -mbarrel-shifter -mfpu=fpus_all
```

The memory map is shown in Figure 27.

Figure 27 Memory Map of ARC_EM9D Configuration



ARC_EM11D Configuration

This is an ARC EM core with 32 bits of address space, 64 KB of code memory (ICCM) and 64 KB of data memory (DCCM).

The MetaWare compiler options for this configuration are:

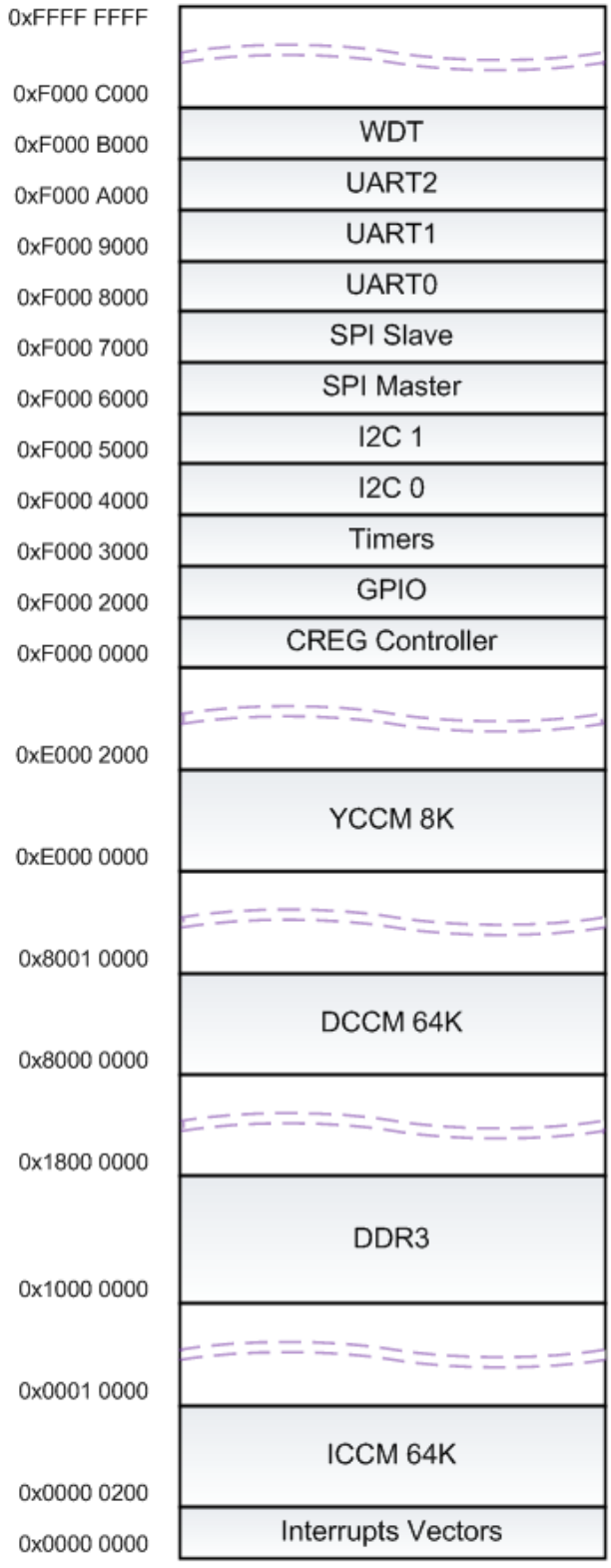
```
-arcv2em -core2 -Hrgf_banked_regs=32 -HL -Xcode_density
-Xdiv_rem=radix2 -Xswap -Xbitscan -Xmpy_option=mpyd -Xshift_assist
-Xbarrel_shifter -Xdsp2 -Xdsp_complex -Xdsp_divsqrt=radix2 -Xdsp_itu
-Xdsp_accshift=full -Xagu_large -Xxy -Xbitstream -Xfpus_div -
Xfpu_mac -Xfpuda -Xfpus_mpy_slow -Xfpus_div_slow -Xtimer0 -Xtimer1
-Xstack_check -Hccm –Xdmac
```

The GCC compiler options for this configuration are:

```
-mcpu=arcem  -mlittle-endian -mcode-density -mdiv-rem –mswap –mnorm
-mmpy-option=6 -mbarrel-shifter -mfpu=fpuda_all
--param l1-cache-size=16384 --param l1-cache-line-size=32
```

The memory map is shown in Figure 28.

Figure 28 Memory Map of ARC_EM11D Configuration



Detailed Core Configurations

Table 25 describes the ARC_EM7D, ARC_EM9D and ARC_EM11D configurations.

Table 25 Configuration Details

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
`instances`	The number of instantiations of this core.	`1`	`1`	`1`
`dsp_complex`	Enable/disable support for single cycle 16b+16b complex instructions and butterfly operations, else 2-cycle complex instructions only without butterfly support.	`True`	`true`	`true`
`dsp_itu`	Enable/disable support for ITU bit-accurate 1 bit fractional shift before accumulation, else 1-bit fractional shift result after accumulation only.	`False`	`True`	`true`
`dsp_divsqrt`	Enable/disable support for divide and square root operations: DIV(U), REM(U), SQRT.	`radix2`	`radix2`	`radix2`
`dsp_accshift`	Select support for accumulator shift operations: no supported, limited shift support only or full shift support and convergent rounding.	`limited`	`Full`	`Full`

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
dsp_impl	Infer DSP data path elements from Verilog operators or use optimized versions.	optimized	optimized	optimized
fpu_dp_assist	Enable/disable support for double-precision acceleration instructions.	-	False	True
fpu_fma_option	Enable/disable support for the fused multiply-add and multiply-subtract instructions.	-	True	True
fpu_mas_cycles	The number of mul/add/sub cycles.	-	2	2
fpu_div_option	Enables divide and square-root acceleration.	-	True	True
fpu_div_cycles	Controls div/sqrt implementation.	-	17	17
byte_order	Endianness.	Little	Little	Little
code_density_option	Code Density ISA extension.	True	True	True
bitscan_option	Bit-Scan ISA extension.	True	True	True
shift_option	Shift option.	3	3	3
swap_option	SWAP instruction.	True	True	True
code_protection	Disables any load or store to the corresponding region.	False	False	False

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
stack_checking	Checks stack accesses and raises an exception when a stack overflow or underflow is detected.	True	True	True
unaligned_ option	Enable/disable unaligned loads and stores.	False	True	True
intvbase_preset	Interrupt vector base.	0	0	0
rgf_num_regs	Number of core registers.	32	32	32
rgf_wr_ports	Register file: number of write ports.	1	2	2
rgf_num_banks	Number of register banks: Two register banks are needed for fast IRQ, but may be selected even without.	1	2	2
rgf_banked_regs	Number of banked registers.	32	32	32
num_actionpoints	Number of trigger events available.	2	2	2
aps_feature	Selects Actionpoint feature set.	min	min	min
smart_stack_ entries	Specifies the number of entries in the trace buffer.	8	8	8
pct_counters	The number of Performance Monitor counters.	8	8	8
dccm_size	Size of the Data Closely Coupled Memory (DCCM) in bytes.	131072	131072	65536

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
dccm_dmi	DCCM has DMI port.	False	false	false
dccm_interleave	Split DCCM into even/odd memory banks.	True	true	true
iccm0_size	Defines the size of ICCM0 in bytes. This ICCM has 0 wait states.	262144	262144	65536
iccm0_base	Sets the initial memory region assignment for ICCM0.	0	0	0
iccm0_wide	Creates ICCM0 as 64b memory to reduce accesses.	False	False	False
iccm0_dmi	ICCM0 has DMI port.	False	False	False
dc_size	D-cache size.	16384	-	16384
dc_ways	D-cache ways.	2	-	2
dc_bsize	D-cache line length.	32	-	32
dc_feature_level	D-cache feature level.	2	-	2
dc_uncached_ region	D-cache uncached region.	false	-	false
ic_size	I-cache size.	16384	-	16384
ic_ways	I-cache ways.	2	-	2
ic_bsize	I-cache line length.	32	-	64

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
ic_pwr_opt_level	I-cache dynamic power optimization.	0	-	0
ic_feature_level	I-cache feature level.	2	-	1
ic_disable_on_reset	Disable I-cache on reset.	False	-	False
number_of_interrupts	The total number of interrupts.	22	22	22
external_interrupts	Number of external interrupt pins.	17	18	18
number_of_levels	Priority levels in the interrupt controller.	4	4	4
firq_option	Fast IRQ enabled.	False	True	True
sec_modes_option	Enable secure shield 2+2 mode.	True	-	-
mpu_sid_option	Enable SID support in Secure Shield.	True		
dmac_channels	The number of data channels implemented.	2	2	2
dmac_fifo_depth	DMA transfer FIFO depth.	2	2	2
dmac_int_config	Configure DMA channel interrupt levels: A separate single interrupt level should be defined for each DMA channel or one defined for all; internal or external.	Single-Internal	Single-Internal	Single-Internal
dmac_registers	Specifies the number of DMA channels with their registers located in auxiliary space.	0	0	0

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
dmac_mem_if	Specifies whether the DMA controller system memory interface is integrated into the existing ARC EM system memory interfaces or has its own interface.	integrated	integrated	integrated
dmac_per_if	Internal vs DW peripheral interface. Specifies (in hex) which channels have the DW interface.	3	3	3
agu_size	Specifies predefined configurations of modifiers, address pointers and offset registers.	-	large	large
agu_accord	Enables/disables the accordion stage if operating frequency is critical.	-	True	True
agu_wb_depth	Specifies the write buffer depth	-	4	4
stimer_0_int_level	Sets the interrupt level of secure timer 0.	1	-	-
secure_debug	Enables secure debug feature	True	False	False
xy_config	Selects the XY memory configuration: DCCM only, DCCM + Y, DCCM + X + Y.	-	dccm_x_y	dccm_y
xy_size	Specifies the size of X and Y memories. X and Y memories both have the same configured size.	-	8192	8192

Configuration Option	Description	ARC_EM7D	ARC_EM9D	ARC_EM11D
xy_interleave	Splits XY memories into odd/even instances to enable single cycle unaligned access.	-	True	True
xy_x_base	Specifies the base region for X memory.	-	12	12
xy_y_base	Specifies the base region for Y memory.	-	14	14
clock_speed	Target clock speed of the system.	25 MHz	20 MHz	20 MHz

Appendix: C
FPGA Image Recovery

ARC EM Starter Kit has pre-programmed configurations stored in SPI flash. However, it is possible to do firmware updates using Xilinx tools.

This section describes software and hardware tools required for FPGA image programming and the actions required to perform it.

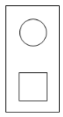
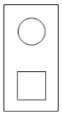
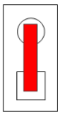
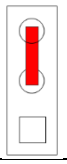
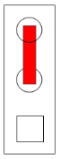
Board Jumper Settings

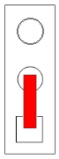
The ARC EM Starter Kit hardware contains pre-installed FPGA configurations. Therefore, it is ready to use with no extra actions needed.

Table 26 describes the board jumper settings. Changing jumper positions is only needed for reprogramming the FPGA or the serial flash memory.

Jumper J8 has two pins. J16 has three pins; pin 1 is at the bottom and it is the closest one to J8.

Table 26 Board Jumper Settings

Reference	Description		Default Jumper Setting
J8	This jumper selects the JTAG target.		
		Select the ARC EM processor Leave the pins open when using the Meta-Ware Debugger	
		Select the FPGA Put a jumper in place during FPGA configuration, for example for programming the bitfile.	
J16	This jumper selects the SDIO mode.		 *Warning:* *Do not alter this setting.*
		normal SDIO mode	

Reference	Description		Default Jumper Setting
		reserved	

Xilinx Lab Tools Installation

The Xilinx Lab Tools utility is optional. This toolset includes the **iMPACT** tool, which can be used to configure the FPGA with bitfiles that may become available on the ARC EM Starter Kit download site. The Xilinx Lab Tools can be downloaded from the following address: http://www.xilinx.com/support/download/index.htm.

Connecting the FPGA Programming Cable

Programming of an FPGA bitfile is normally not required, because pre-programmed images are available in the SPI Flash memory. Refer to the Select ARC EM Configurations section for more details.

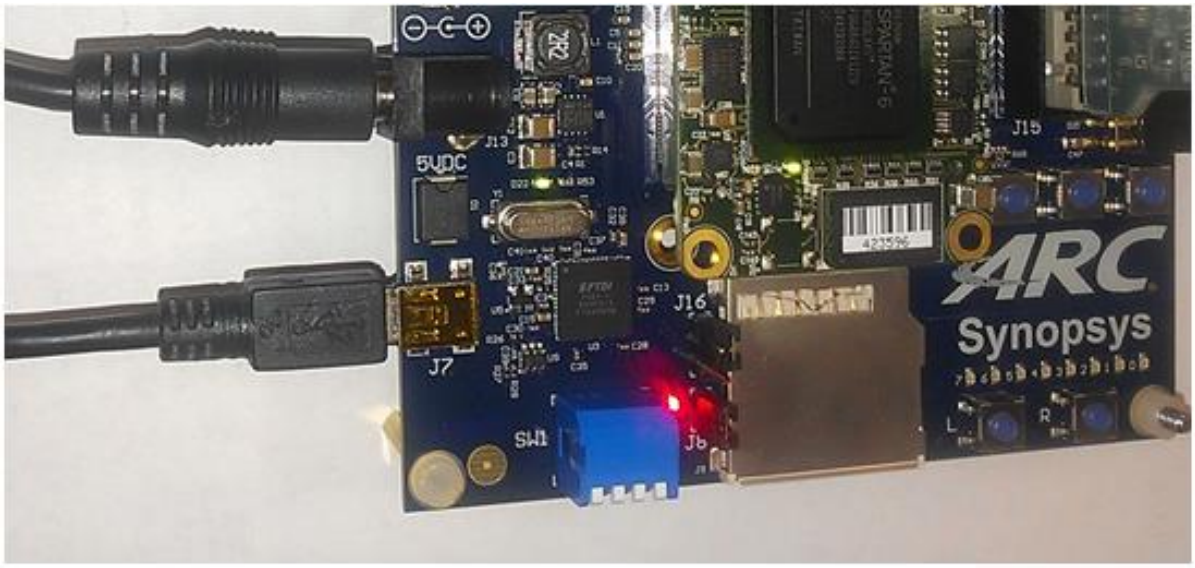
Cabling is required if you wish to program other bitfiles that may become available on the ARC EM Starter Kit web site.

The following cabling is required for programming the board with an FPGA bitfile:

- USB cable with mini-USB connector

Connect this cable to the connector J7 on the board and to your computer.

Figure 29 USB FPGA Programming Cable Connection to the Board



FPGA Programming Sequence

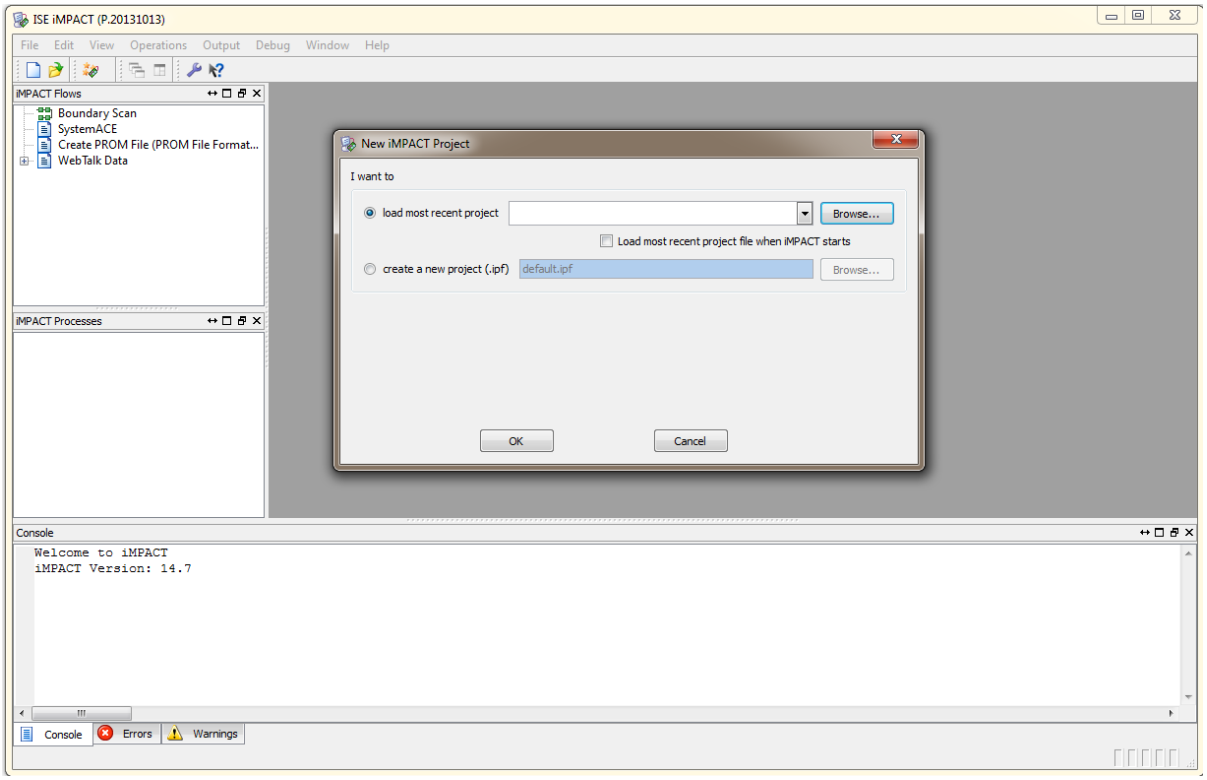
This section describes the process of programming the Xilinx FPGA in the ARC EM Starter Kit with bitfiles using the iMPACT tool.

It is assumed that the USB programming cable is attached to the appropriate connector on the board according to the *Connecting the FPGA Programming Cable* section.

To perform FPGA programming, perform the following steps:

1. Put a jumper in place at header **J8**.
2. Run the Xilinx **iMPACT** tool:

Figure 30 iMPACT Tool – Main Window View

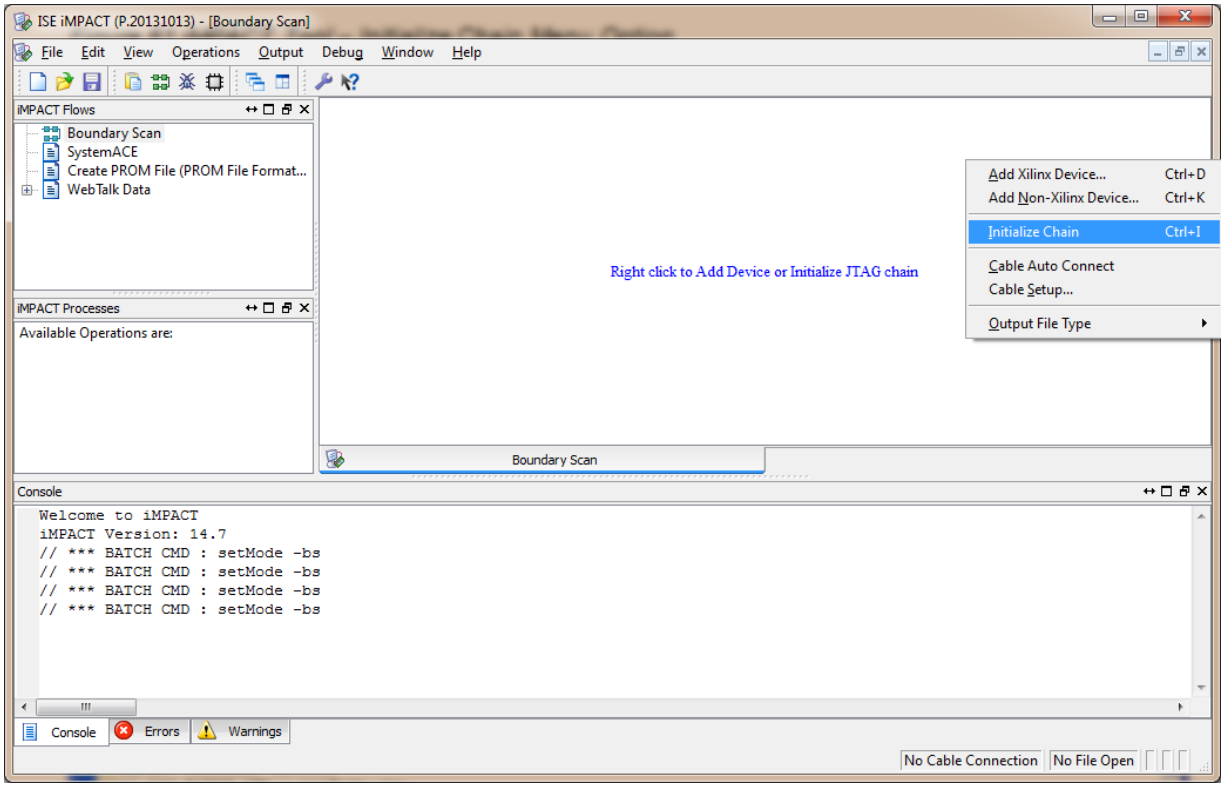


3. Cancel the **iMPACT Project** selection dialog.

4. Select the **Boundary Scan** option from the **Flows** toolbar, situated in the upper left corner of the application.

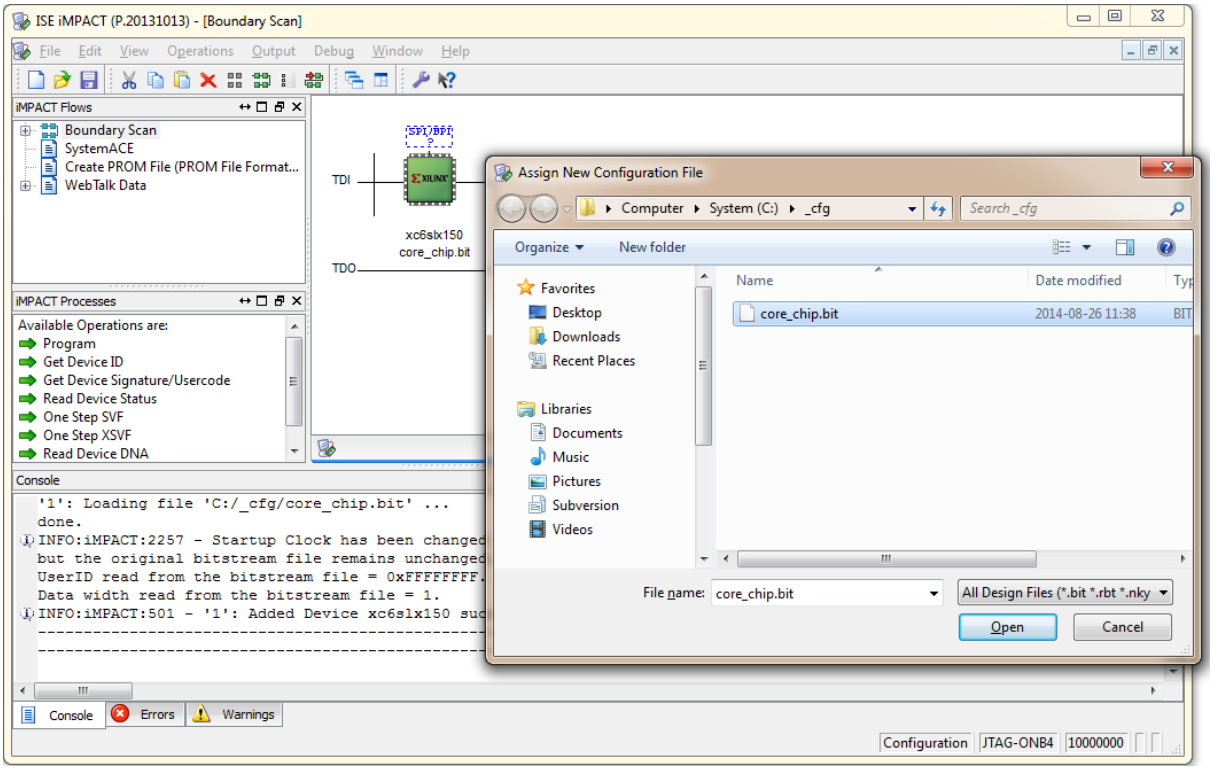
5. When the **Boundary Scan** window appears, right click on it and select **Initialize Chain** option.

Figure 31 iMPACT Tool – Initialize Chain Menu Option



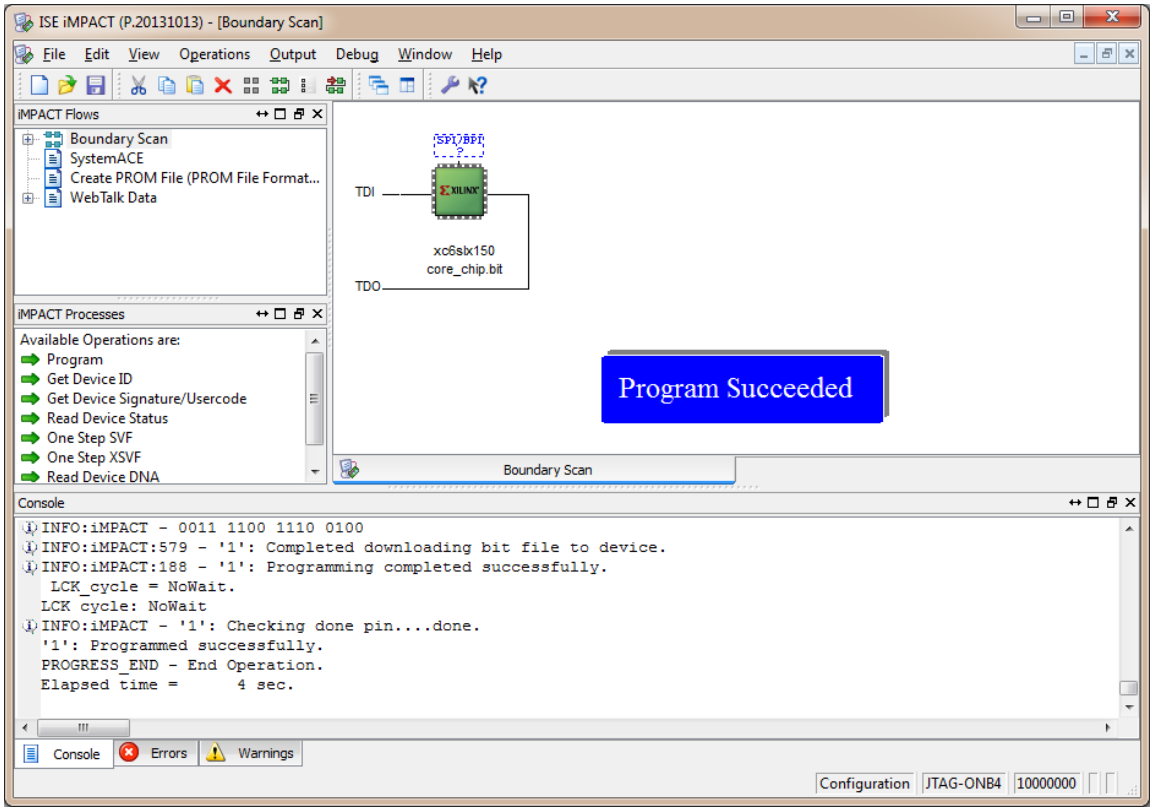
Select the FPGA **XC6SLX150** in the sequence of detected devices. Double-click on it and point to the location of an FPGA image `*.bit` file.

Figure 32 iMPACT Tool – Assign New Configuration File Dialog Box



6. If no SPI flash programming is required, select "**No**" in the window appeared.
7. Right-click on the **FPGA** device and select "**Program**".
8. A screenshot of the iMPACT tool after successful programming is shown in Figure 33.

Figure 33 iMPACT View of the Successful FPGA Programming



9. Remove the jumper from header J8.

SPI Flash-Programming Sequence

This section describes the process of programming the SPI Flash in the ARC EM Starter Kit with MCS files using the iMPACT tool.

It is assumed that the USB programming cable is attached to the appropriate connector on the board according to the *Connecting the FPGA Programming Cable section*.

In order to perform SPI flash programming, perform the following steps:

1. Put a jumper in place at header **J8**.
2. Run the Xilinx **iMPACT** tool.
3. Cancel the project selection dialog box.
4. Select the **Boundary Scan** option from the **Flows** toolbar, situated in the upper left corner of the application.

5. When the **Boundary Scan** window appears, right click on it and select **Initialize Chain** option.

6. Select the SPI/BPI dashed frame above FPGA icon in the sequence of detected devices. Double-click it and point to the location of a flash image `*.mcs` file.

Figure 34 iMPACT Tool – select SPI device

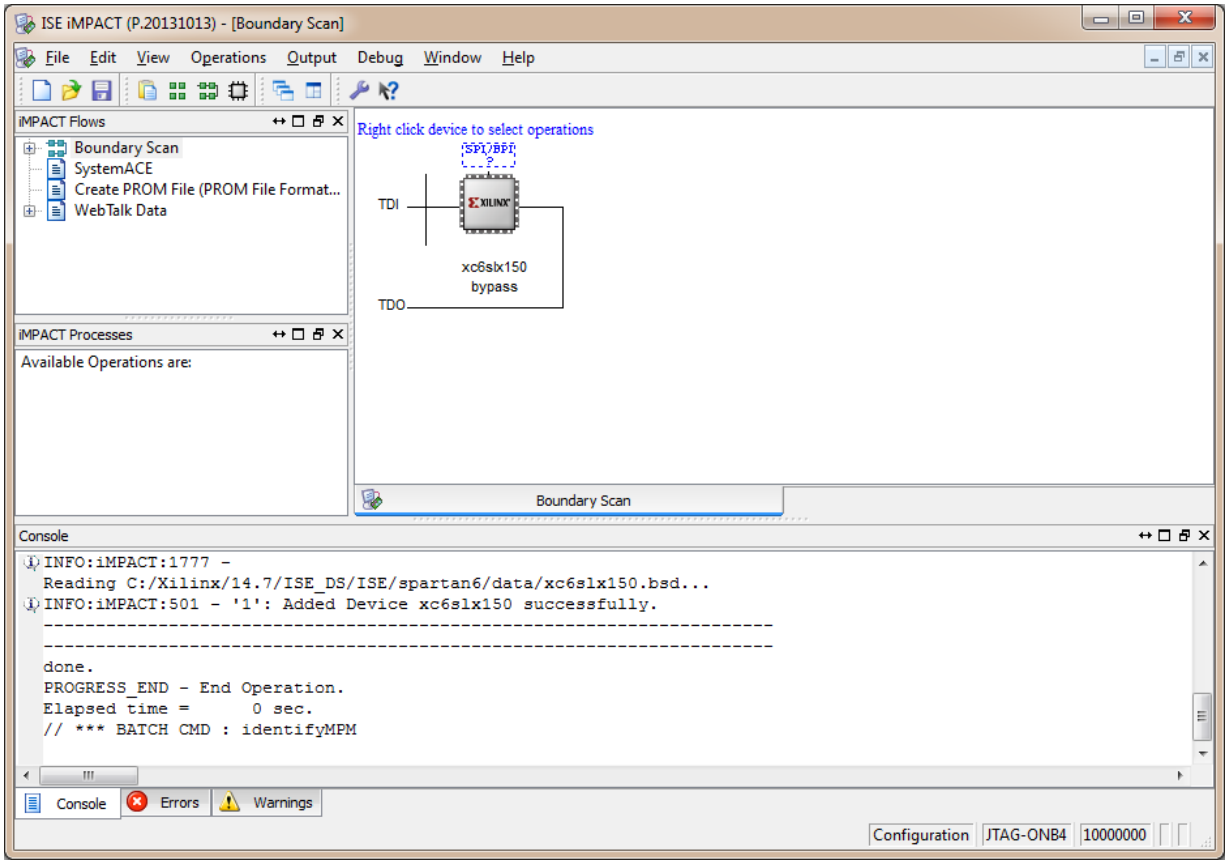
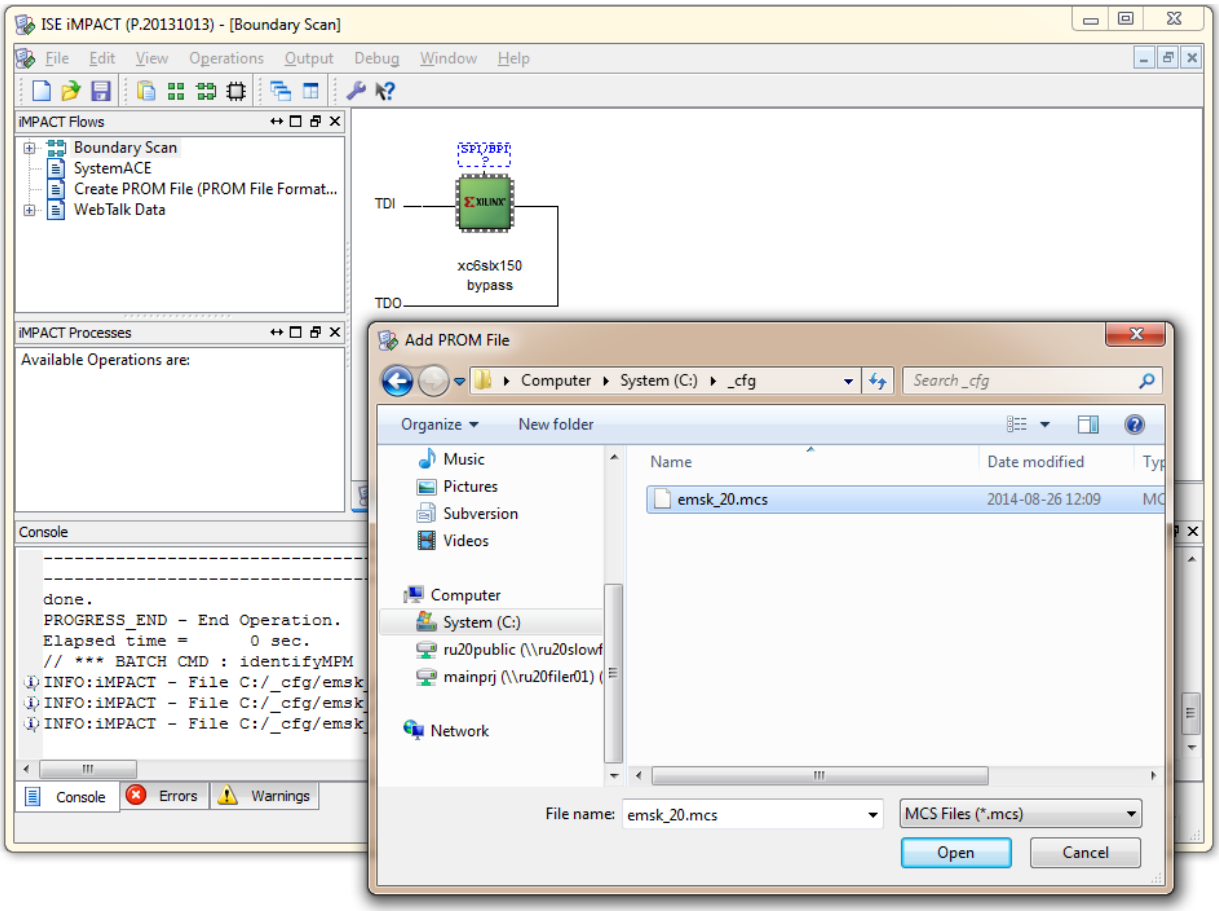
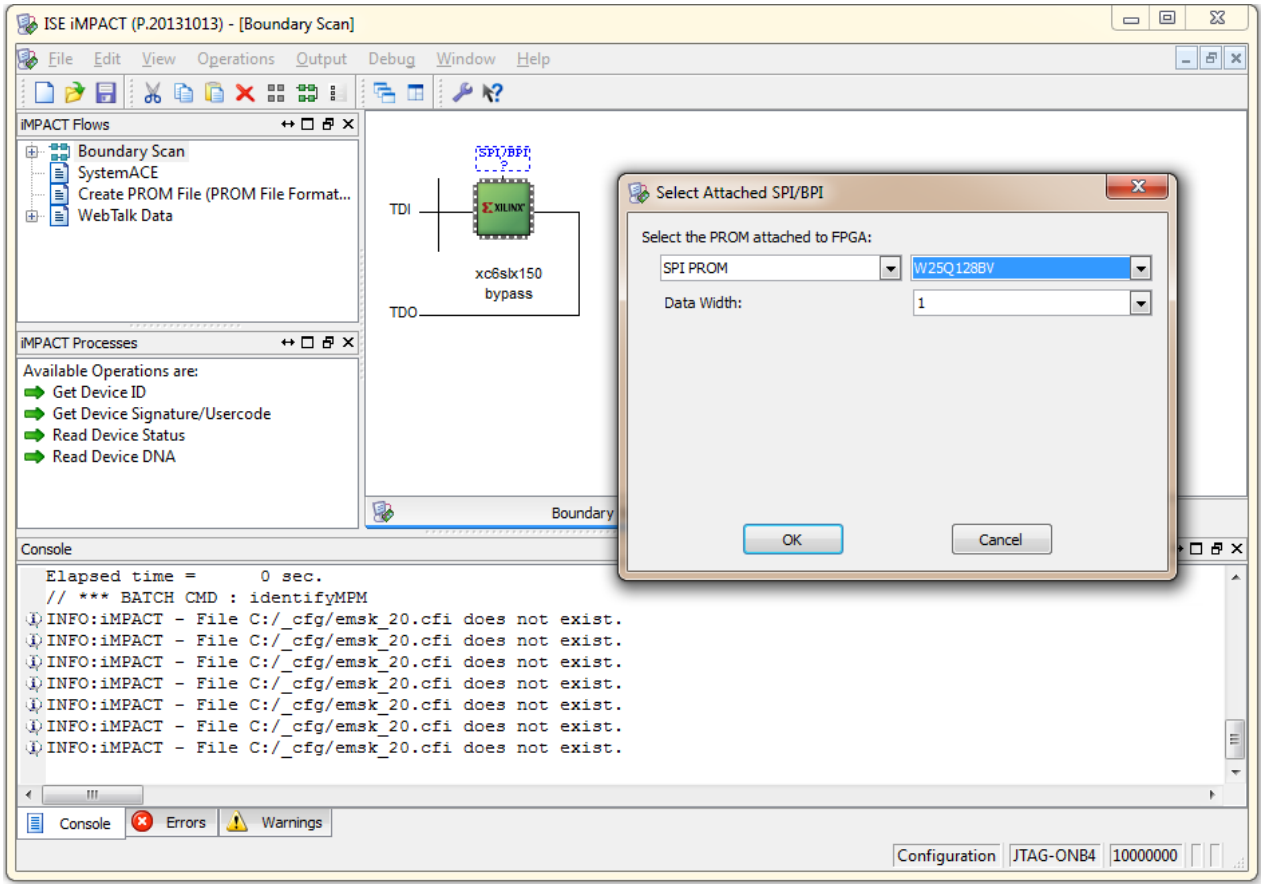


Figure 35 iMPACT Tool – Add PROM File Dialog Box



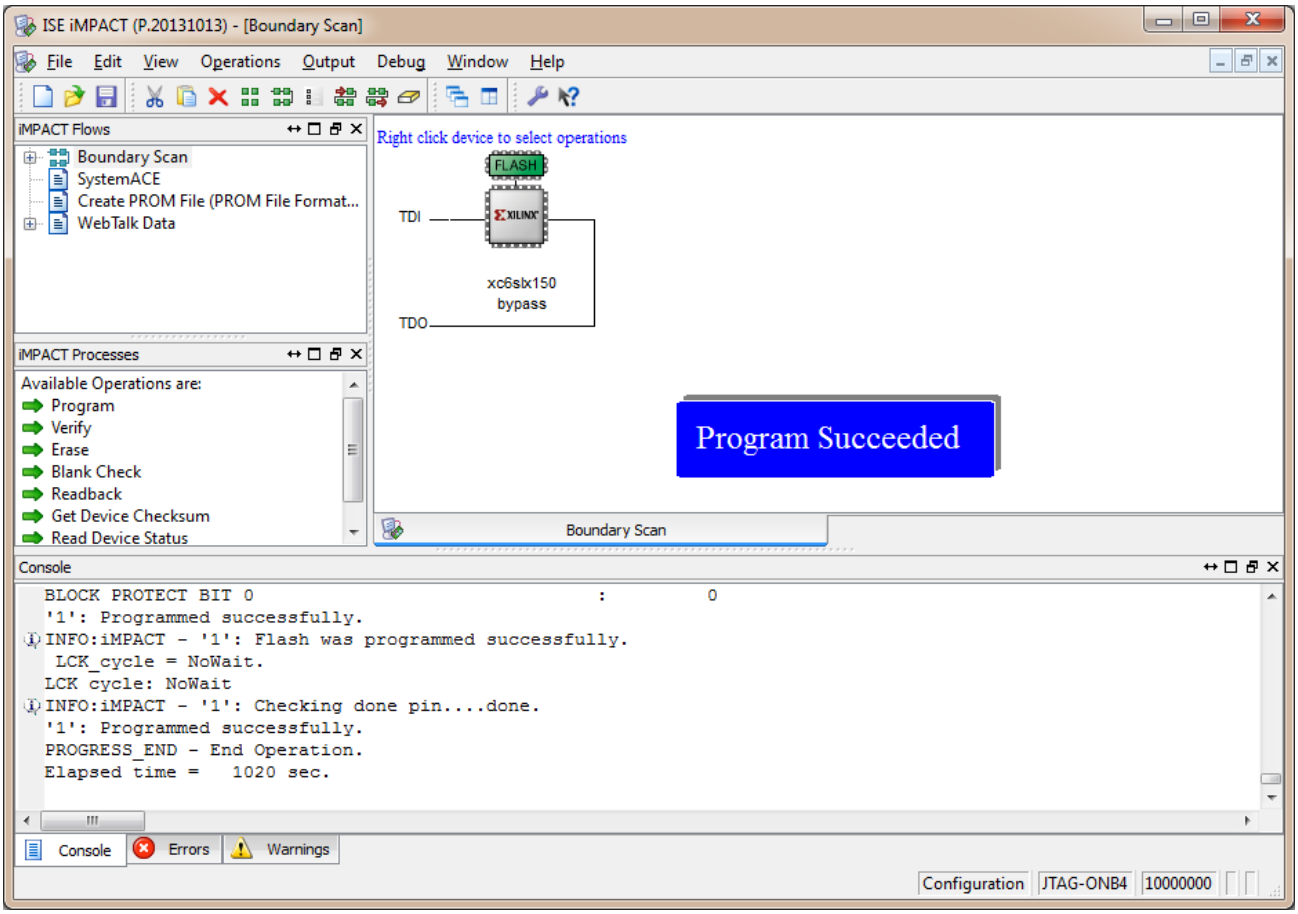
7. Select SPI PROM W25Q128BV flash type, data with data width 1.

Figure 36 iMPACT Tool – Select Attached SPI/BPI Dialog Box



8. Right-click on the **Flash** device and select **Program**.
9. Accept the default programming options.
10. Flash programming takes approximately four minutes.
11. A screen shot of the iMPACT tool after successful programming is shown in Figure 37.

Figure 37 iMPACT View of the Successful SPI ROM Programming



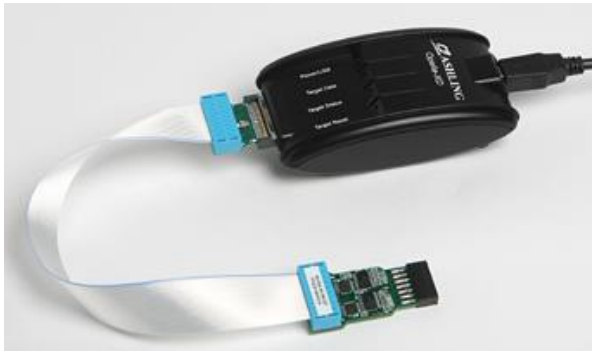
Appendix: D
Using a JTAG Debugger

It is best to use the on-board USB port to connect your debugger. If you wish to use a JTAG debug cable instead, you can connect it to the JTAG debugging port at the J15 connector. This section describes software and hardware tools for using example JTAG debuggers.

Ashling Opella-XD Debugger

The Ashling Opella-XD debugger has two components: The Opella-XD for ARC Debug Probe and the 20-way JTAG Debug cable TPAOP-ARC20. They are shown in Figure 38.

Figure 38 Ashling XD Pod with Debug Cable



For more details, refer to http://www.ashling.com/.

Connect the JTAG debug cable connect to the J15 connector on the baseboard. Match the signal names on the cable and the connector according to Table 27.

Table 27 ARC EM Starter Kit JTAG Connector Pin-out

J15 Pin Number	JTAG Signal
1	V_REF
4	GND
5	TDI
7	TMS
9	TCK
13	TDO

Figure 39 ARC JTAG Connection to the Board



Lauterbach TRACE32 Debugger

The Lauterbach TRACE32 debugger for ARC cores has two components:

- Power debug module
- JTAG debugger for ARC

These components are shown in Figure 40.

Figure 40 Lauterbach TRACE32 with Debug Cable



Connect the debugger to the J15 connector on the baseboard. The JTAG pin assignment is the same as for the Ashling debugger. Refer to Table 27 for more information.

For more information on Lauterbach and the EM Starter Kit, see http://www.lauterbach.com/frames.html?arc_emsk.php.

Lauterbach TRACE32 Tool Installation

This tool is required if the Lauterbach TRACE32 debugger is used for JTAG communication with an ARC EM core.

The software can be downloaded from the Lauterbach Download Center web page at http://www.lauterbach.com/frames.html?downloadcenter.html.

Glossary and References

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

Glossary

APB
AMBA Advanced Peripheral Bus

AHB
AMBA Advanced High Performance Bus

DCCM
Data Closely Coupled Memory

DMP
Data Memory Pipeline (ARC proprietary bus)

DW
Synopsys DesignWare IP solutions

GPIO
General Purpose Input/Output pins

ICCM
Instruction Closely Coupled Memory

Pmod

Digilent Pmod™ socket connector

References

[1] *ARC EM Starter Kit Getting Started*
http://www.synopsys.com/dw/ipdir.php?ds=arc_em_starter_kit

[2] *AMBA Specification (http://www.arm.com)*

[3] *AMBA APB3 Specification (http://www.arm.com)*

[4] *Synopsys DesignWare DW_apb_gpio Databook, version* 2.11a, June 2015

[5] *Synopsys DesignWare DW_apb_i2c Databook, version* 2.00a, June 2015

[6] *Synopsys DesignWare DW_apb_ssi Databook, version* 4.00a, June 2015

[7] *Synopsys DesignWare DW_apb_timers Databook, version* 2.10a, June 2015

[8] *Synopsys DesignWare DW_apb_uart Databook, version* 4.00a, June 2015

[9] *Synopsys DesignWare DW_apb_wdt Databook, version* 1.09a, June 2015

[10] *Synopsys DesignWare DW_apb Databook,* version 3.00a, June 2015

[11] *Xilinx UG380: Spartan-6 FPGA Configuration User Guide*
(http://www.xilinx.com/support/documentation/user_guides/ug380.pdf)

[12] *ARCv2 ISA Programmer's Reference*

[13] *C/C++ Programmer's Guide for the MetaWare Compiler*

[14] *Xilinx XAPP496*
(http://www.xilinx.com/support/documentation/application_notes/xapp496.pdf)

[15] Maxim DS2502-E48 product overview
(http://www.maxim-ic.com/datasheet/index.mvp/id/3748)

[16] Maxim DS2432 product page
(http://www.maximintegrated.com/datasheet/index.mvp/id/2914)

[17] *Xilinx DS160: Spartan-6 Family Overview*
(http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf)

[18] *Xilinx DS162: Spartan-6 FPGA Data Sheet: DC and Switching*
(http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf)

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP