Mars ZX3 SoC Module

Reference Design for Mars PM3 Base Board User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Mars ZX3 SoC module reference design and to provide the user with a step-by-step guide to the complete Xilinx® SoC design flow used for the Mars ZX3 SoC module.

Summary

This document first gives an overview of the Mars ZX3 SoC module reference design and then guides through the complete Xilinx SoC design flow for the Mars ZX3 SoC module in the getting started section. In addition, the internals and the boot options of the Mars ZX3 SoC module reference design are described.

Product Information	Number	Name
Product	MA-ZX3	Mars ZX3 SoC Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-424-001	04	16.01.2018

Approval Information	Name	Position	Date
Written by	DIUN	Design Engineer	27.04.2016
Verified by	GLAC	Technical Expert	27.04.2016
Approved by	RPAU	Quality Manager	16.01.2018

Document History

Version	Date	Author	Comment
04	16.01.2018	DDUE	Updated to Vivado and SDK 2017.4, other style updates and clarifications
03	10.06.2016	DIUN	Minor style updates and clarifications
02	27.04.2016	DIUN	Minor corrections
01	27.01.2016	DIUN	Version 01

Table of Contents

Overview

Introduction

The Mars ZX3 SoC module reference design demonstrates a system using the Mars ZX3 SoC module in combination with the Mars PM3 base board. It presents the basic configuration of the device and features some example applications.

A troubleshooting section is included at the end of the document, to help the user solve potential issues related to board connectivity and/or system functionality.

An introduction to the Xilinx tools is provided by the documents below:

- Zynq®7000 All Programmable SoC Embedded Design Tutorial [1]
- Vivado Design Suite User Guide, Embedded Processor Hardware Design [2]
- Vivado Design Suite Tutorial, Embedded Processor Hardware Design [3]

More information on the Mars ZX3 SoC module and the Mars PM3 base board can be retrieved from their respective user manuals [5] [6].

Directory Structure

The Mars ZX3 SoC module reference design is delivered as a ZIP archive file with the following directory structure and contents:

- `binaries` — Pre-compiled binaries directory
- `scripts` — Scripts directory required for Vivado project creation
- `SdkExport` — Pre-generated hardware description files required for SDK applications
- `software` — Software projects directory
- `src` — Xilinx pinout and timing constraints and VHDL source code directory
- `Mars_ZX3_Reference_Design_for_Mars_PM3_User_Manual.pdf` — User manual (this document)

Prerequisites

- IT
  - A computer with a microSD card slot (optional[1]) running Windows 7 64-bit (or later)
- Software
  - Xilinx Vivado 2017.4 WebPack, Evaluation, Design or System Edition (check the Mars ZX3 SoC Module User Manual [5] for details on device support in Xilinx tools)
  - Xilinx Software Development Kit (SDK) 2017.4
  - Enclustra Module Configuration Tool (MCT) [7] (optional[2])
  - A terminal emulation program (e.g. Tera Term)
  - PuTTY (optional[3])
- Hardware
  - An Enclustra Mars ZX3 SoC module
  - An Enclustra Mars PM3 base board
- Accessories
  - A standard micro USB cable[4]
  - A JTAG PM3 breakout board or a Xilinx JTAG breakout cable

[1] Only required for SD card boot mode
[2] May be used for flash programming, for SoC device configuration or for Cypress FX3 configuration.
[3] Required for running the Ethernet lwIP example
[4] An extra USB type B cable is required for running the USB storage emulation example.

- A Xilinx JTAG programmer (e.g. Platform Cable USB) download cable
- A microSD card (optional[1])

Reference Design Description

Block Diagram

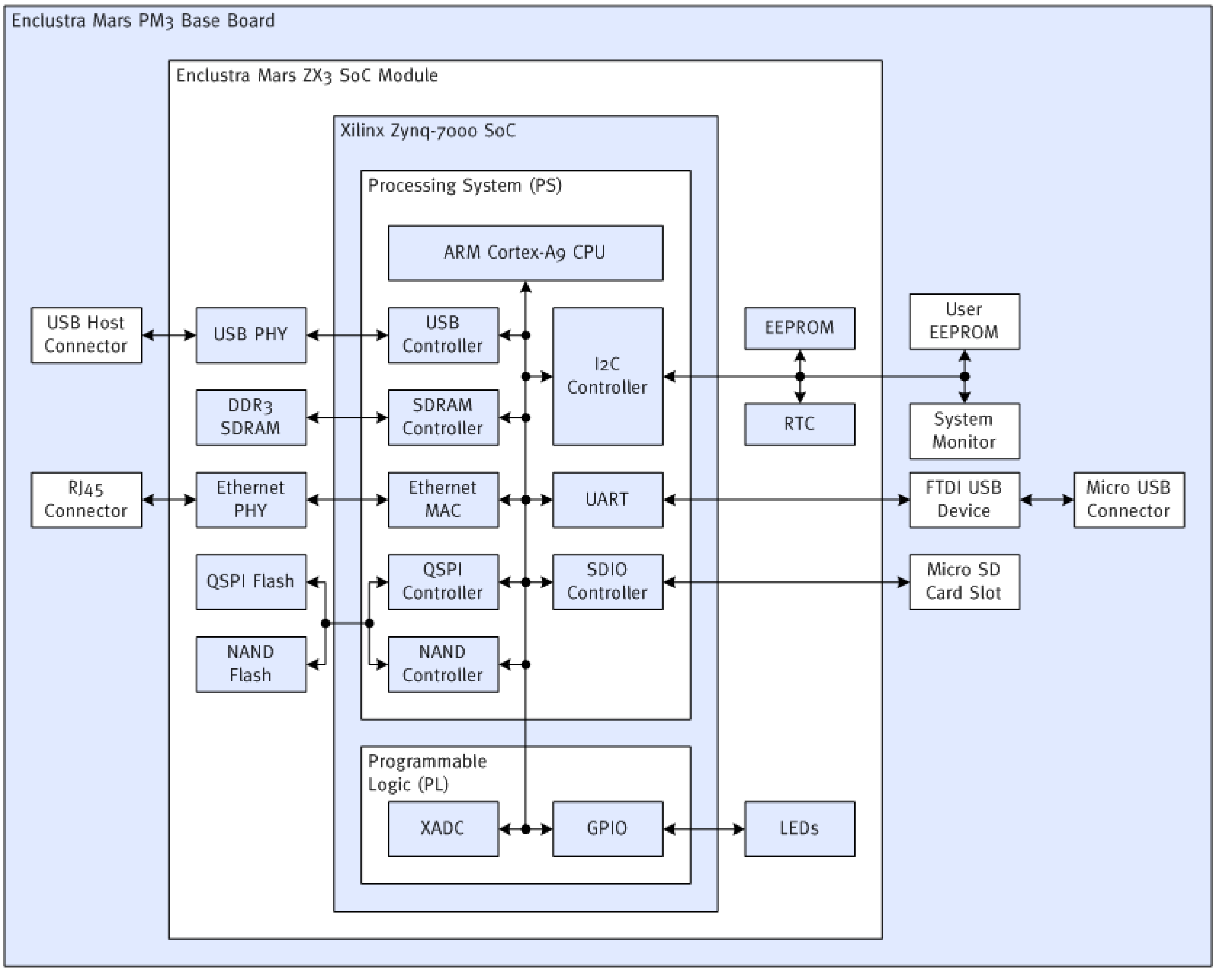


Figure 1: Hardware Block Diagram

Processing System (PS)

Clocks

The PS input clock frequency is configured to 33.33 MHz, while the CPU clock frequency is configured to 666.66 MHz. The CPU clock frequency multiplier can be modified in the Zynq system. A 50 MHz clock is exported from PS to the PL.

DDR3/DDR3L SDRAM

The DDR3/DDR3L SDRAM memory runs at 400 MHz (800 Mbit/sec) at a voltage of 1.35 V by default. These parameters can be modified in the Zynq system. For a voltage of 1.5 V, beside the changes in the PS, it is necessary to change the top level assignment to DDR3_VSEL signal from logic low to high impedance.

The DDR settings in the Zynq system must be configured according to the Mars ZX3 SoC Module User Manual [5].

SD Card

The SD card is configured in the PS to the MIO 40..45 pins. This enables SD card access, as well as booting from the SD card.
To allow the Mars ZX3 SoC module to boot from the SD card, the hardware configuration on the Mars PM3 base board must be done according to Section 4.2.2.

Please note that the SD pins are shared between the PS and the PL.

Warning!

Because the MIOs 40..45 are connected to FPGA pins Y18, AA18, AA17, AB17, U17, V17 in parallel, make sure the FPGA pins are in high impedance state before driving the PS SD pins and vice versa.

In order to be able to access and write to the SD card from Linux, it is recommended to enable the Write Protect (WP) and Card Detect (CD) pins in the PS, map them to EMIO pins and tie them off to high, respectively low state. An example is available in the reference design sources.

I2C

The I2C controller I2C0 is configured to the EMIO pins. Table 1 lists the connected devices on the Mars ZX3 SoC module and Mars PM3 base board.

Device	Address (7-bit)	Vendor	Part Type
Real-time clock	0x6F	Intersil	ISL12020MIRZ
RTC user SRAM	0x57	Intersil	ISL12020MIRZ
Secure EEPROM	0x5C	Maxim	DS28CN01
I2C GPIO expander	0x21	Semtech	SX1505I087TRT
User EEPROM	0x54	Microchip	24AA128T-I/MNY
System monitor	0x2F	Texas Instruments	LM96080CIMT/NOPB

Table 1: I2C Devices

The device vendors or addresses of the I2C devices may change in future revisions of Mars ZX3 SoC module or Mars PM3 base board.

For detailed information on the I2C devices, please refer to the corresponding user manuals [5] [6].

Quad SPI Flash Controller

The quad SPI flash controller is connected to MIO 1..6 and 8 pins in Single Slave Select mode. MIO 2..6 pins are shared between NAND flash and QSPI flash on the Mars ZX3 SoC module. Please refer to the Mars ZX3 SoC Module User Manual [5] for details about flash programming and usage.

To allow the Mars ZX3 SoC module to boot from the QSPI flash, the hardware configuration on the Mars PM3 base board must be done according to Section 4.1.2.

UART

The UART0 is mapped to EMIO pins and connected to the FTDI USB device controller on the Mars PM3 base board. The UART is configured as shown in Table 2.

Parameter	Value
Baud rate	115'200
Data	8 bit
Parity	None
Stop	1 bit
Flow control	None

Table 2: UART Configuration

Ethernet

The Ethernet MAC ENET0 is mapped to MIO 16..27 and MIO 52..53 pins and is connected to a Micrel KSZ9031 Ethernet PHY on the Mars ZX3 SoC module using RGMII interface. The PHY can be configured via the MDIO management interface on PHY address 3.

Please note that the RGMII delays in the Ethernet PHY need to be configured before the Ethernet interface can be used. In the reference design this is done in the First Stage Boot Loader (FSBL) and in the Ethernet lwIP example.

Note that the Ethernet pins are shared between the PS and the PL.

Warning!

Because the MIOs 16..27 and 52..53 are connected to FPGA pins V10, V9, V8, W8, W11, W10, U11, U10, AA12, AB12, Y11, Y9, Y8, U6 in parallel, make sure the FPGA pins are in high impedance state before driving the PS ENET0 pins and vice versa.

USB

The USB controller USB0 on MIO 28..39 pins is connected to a USB3320C USB 2.0 PHY. This interface can be configured for USB host, USB device and USB On-The-Go (OTG) operations.

Depending on the required USB mode, the DIP switches on the Mars PM3 base board must be configured correctly. Please refer to the Mars PM3 Base Board User Manual [6] for details.

GPIOs

The unused MIO pins from the PS are available as GPIOs. They are mapped to MIO pins 0, 7, 9..15, and 46..51 in the PS. The function of the general purpose pins on the Mars ZX3 SoC module is described in Table 3. The shared pins can be used either in the PL or in the PS.

GPIO	Signal	Function
MIO 15	MDIO_SEL_LED1#	MDIO select/LED1# shared signal, see Mars ZX3 SoC Module User Manual [5] for details
MIO 46	GPIO 46	GPIO, shared with FPGA pin U15
MIO 47	GPIO 47	GPIO, shared with FPGA pin U16
MIO 48	GPIO 48	GPIO, shared with FPGA pin W16
MIO 49	GPIO 49	GPIO, shared with FPGA pin Y16
MIO 50	GPIO 50	GPIO, shared with FPGA pin W17
MIO 51	GPIO 51	GPIO, shared with FPGA pin W18

Table 3: PS GPIO Configuration

Warning!
Because the MIOs 46..51 are connected to FPGA pins U15, U16, W16, Y16, W17, W18 in parallel, make sure the FPGA pins are in high impedance state before driving the PS GPIO pins and vice versa.

Warning!
Make sure that all unused FPGA pins are in high impedance state in the top-level, as they may be driven by other devices, or by pull-up/pull-down resistors. An example is given in the VHDL source code and constraints provided in the ZIP archive along with this document.

Programmable Logic (PL)

GPIOs

A Xilinx GPIO controller in the PL is connected to the PS via an AXI bus. The PL GPIOs are connected to LEDs in the top level, as described in Table 4.

The FPGA firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink LED3# on FPGA pin AB15 with a frequency of approximately 3 Hz.

The least three significant bits of the GPIO bus from the PL are connected to the LED0#, LED1# and LED2# outputs.

FPGA Pin	Signal	Function
H18	LED0#	GPIO 0, controlled by the PL GPIO controller
AA14	LED1#	GPIO 1, controlled by the PL GPIO controller
AA13	LED2#	GPIO 2, controlled by the PL GPIO controller
AB15	LED3#	Blinking LED counter MSB

Table 4: FPGA Firmware I/O Configuration

XADC

A Xilinx XADC IP core instance is connected to the PS via an AXI bus, in order to monitor the temperature of the device. The temperature threshold for this module is configured to the industrial applications temperature, 85° Celsius.

The constraints provided in the reference design enable FPGA bitstream power-down, when the temperature increases above the threshold. In this case, the PL will be reset, while the ARM processor will still be running.

Depending on the user application, the Mars ZX3 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the SoC is adequately cooled by installing a heat sink and/or providing air flow. Temperature control and monitoring is very important in a complex design.

Information that may assist in selecting a suitable heat sink for the Mars ZX3 SoC module can be found in the Enclustra Modules Heat Sink Application Note [8].

For Enclustra Mars modules an Enclustra heat sink is available for purchase along with the product. It represents an optimal solution to cool the Mars ZX3 SoC module - it is low profile (less than 7 mm tall) and covers the whole module surface. It comes with a gap pad for the SoC device and four screws to attach it to the module PCB. For details, please refer to the Enclustra website.

Getting Started

This section describes the steps required to configure the Mars ZX3 SoC module and Mars PM3 base board in order to run the example applications. The section includes information on how to:

- Mount the module and configure the base board
- Generate the FPGA bitstream
- Prepare the software workspace
- Run the software applications

The example applications, including the expected results of running the software, are described in detail in Section 3.7.

Essential Information

Warning!

Never mount or remove the Mars ZX3 SoC module to or from the Mars PM3 base board while the Mars PM3 base board is powered. Always remove or turn off the power supply before mounting or removing the Mars ZX3 SoC module.

Warning!

Depending on the user application, the Mars ZX3 SoC module may consume more power than can be dissipated without additional cooling measures; always make sure the SoC is adequately cooled by installing a heat sink and/or providing air flow.

Warning!

Please read carefully the Mars ZX3 SoC module and Mars PM3 base board user manuals before proceeding.

Note that when Enclustra MCT [7] is used for SoC configuration or flash programming, all other tools that may be connected to the FTDI device (e.g. Vivado Hardware Manager, SDK, UART terminal) must be closed.

Hardware Setup

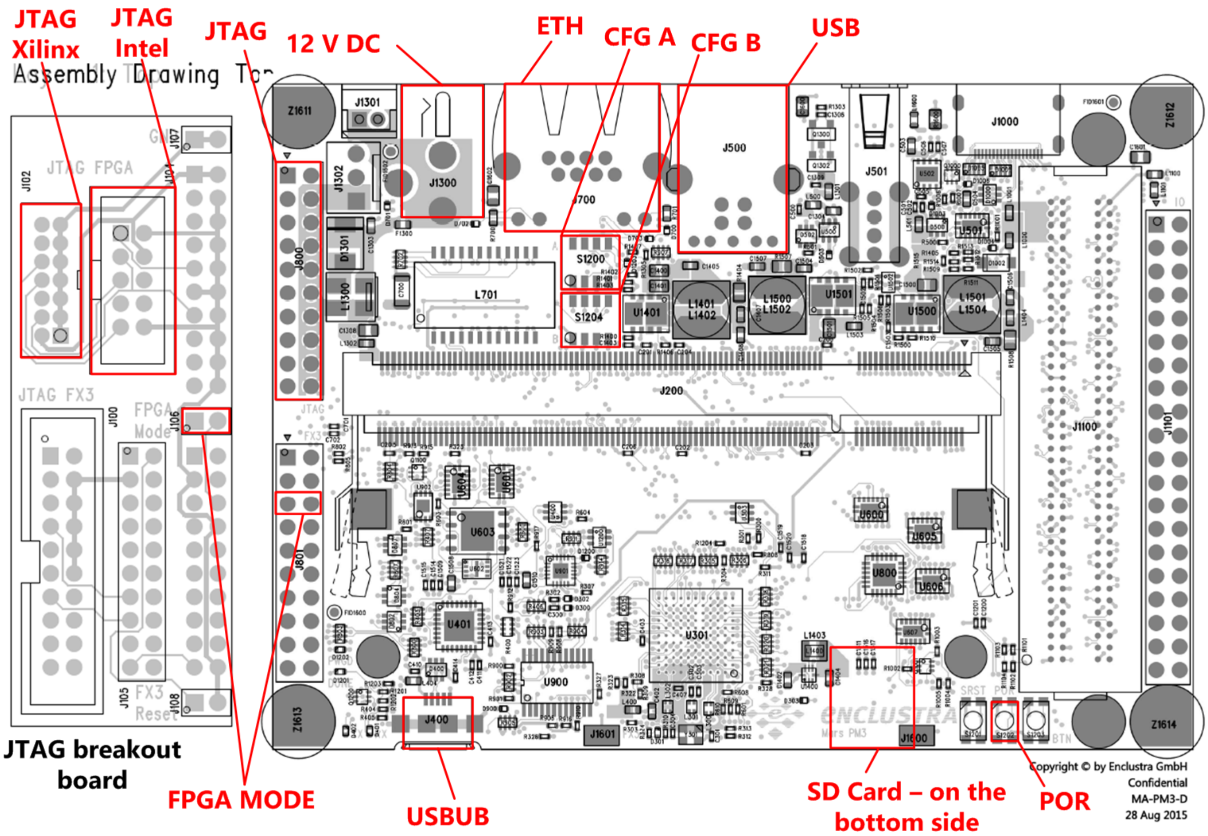


Figure 2: Mars PM3 Base Board Assembly Drawing (Top View)

Step	Description
1	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: OFF, 3: OFF, 4: OFF] • CFG B = [1: ON, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, set the configuration DIP switches as follows: • CFG = [1: OFF, 2: OFF, 3: OFF, 4: OFF]
2	Make sure the SD card slot of the Mars PM3 base board is empty (see label **SD Card** in Figure 2).
3	Mount the Mars ZX3 SoC module and the JTAG breakout board to the Mars PM3 base board.
4	Connect the micro USB cable between your computer and the Mars PM3 base board. Use the micro USB port labeled **USBUB** in Figure 2.

Continued on next page...

Step	Description
5	Connect the Xilinx Platform USB download cable to the JTAG connector of the JTAG PM3 breakout board (see label **JTAG Xilinx** in Figure 2). If a Xilinx JTAG breakout cable is used, use the pins labeled **JTAG** in Figure 2. Please refer to the Mars PM3 Base Board User Manual [6] for JTAG pins mapping.
6	Connect the 12 V DC power supply plug to the power connector of the Mars PM3 base board (see label **12 V DC** in Figure 2).
7	Open a terminal program on your computer (e.g. Tera Term) and open a serial port connection using the COM port labeled with the lower number from the two newly detected ports. For issues related to COM ports detection, refer to Section 5.4. Configure the UART parameters according to Section 2.2.6.

Table 5: Hardware Setup Step-By-Step Guide

FPGA Bitstream Generation

For a fast test of the provided software applications, the pre-generated bitstream included in the `binaries` directory may alternatively be used, therefore the steps described in this section may be skipped.

The `<base_dir>\binaries` directory includes bitstream files for any SoC device that may be equipped on the module.

Step	Description
1	Edit the `fpga_part` variable in `scripts\settings.tcl` file, according to your SoC device. This file includes module name and board information required for the project creation script. All settings, except for `fpga_part` should be left on default. The list of options for `fpga_part` is given in the comments within the Tcl file. Save the file after editing.
2	Start Xilinx Vivado 2017.4 and create the Mars ZX3 SoC module reference design project: 1. Click on the Tcl console at the bottom of the page and type: (a) `cd <base_dir>` (`<base_dir>` is the directory in which you extracted the archive contents). Note that you must use / for hierarchy separator, instead of \. (b) `source scripts/create_project.tcl` 2. Wait for completion
3	Run Synthesis, Implementation & Bitstream Generation in Vivado 2017.4: 1. Click on Generate Bitstream from the Flow Navigator bar 2. In the Launch Runs window click OK - this will start automatically the entire implementation process 3. Wait for completion → select View Reports → OK

Continued on next page...

Step	Description
4	Export the hardware system information (required for the SDK projects): 1. File → Export → Export Hardware 2. Enable Include bitstream checkbox If you want to save the .hdf file to another path than the default location: 3. Click on Choose Location → Select 4. Hit OK

Table 6: FPGA Bitstream Generation Step-By-Step Guide

SDK Workspace Preparation

This section describes how to create and import the software applications. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

The <base_dir>\SdkExport directory includes pre-generated hardware description files for any SoC device that may be equipped on the module.

Step	Description
1	Start Xilinx SDK 2017.4 1. Select any workspace (e.g. <base_dir>\workspace)
2	Create a new board support package (BSP) 1. File → New → Board Support Package → Specify 2. In the New Hardware Project window: (a) For Project Name type hw_platform_0 (b) For Target Hardware Specification select the .hdf file you exported from Vivado, as described in Section 3.3. The default location used by Vivado is <base_dir>\<vivado_proj_dir>\<project_name>.sdk\system_top.hdf Alternatively, the pre-compiled hardware description file from <base_dir>\SdkExport may be used. (c) Hit Finish 3. In the New Board Support Package Project window: (a) For Project Name type standalone_bsp_0 (b) For Hardware Platform select hw_platform_0 (c) For CPU select ps7_cortexa9_0 (d) Hit Finish 4. In the Board Support Package Settings window (see Figure 3): (a) Enable the checkboxes corresponding to the drivers and libraries required in the BSP: lwip141, xilffs, xilmfs, xilrsa (b) Hit OK

Continued on next page...

Step	Description

3	Create new First Stage Boot Loader application
	
	1. File → New → Application Project
	2. In the Application Project window:
	    (a) For Project Name type FSBL
	    (b) For Board Support Package select Use existing, and select standalone_bsp_0
	    (c) Leave the other settings on default values
	    (d) Hit Next
	3. In the Templates window:
	    (a) Select Zynq FSBL
	    (b) Hit Finish
	4. Copy the files provided in <base_dir>\software\FSBL\src directory from the reference design release archive to <workspace>\FSBL\src directory (and replace the fsbl_hooks.c file). This step is required as per explanation in Section 3.7.8.

4	Import the example projects into the workspace (see Figure 4): 1. File → Import... 2. Select General → Existing Projects into Workspace and hit Next 3. For Select root directory choose <base_dir>\software and hit OK 4. Enable checkbox Copy projects into workspace 5. Hit Finish
5	Build all projects 1. Hit Ctrl-B and wait for completion

Table 7: SDK Workspace Preparation Step-By-Step Guide

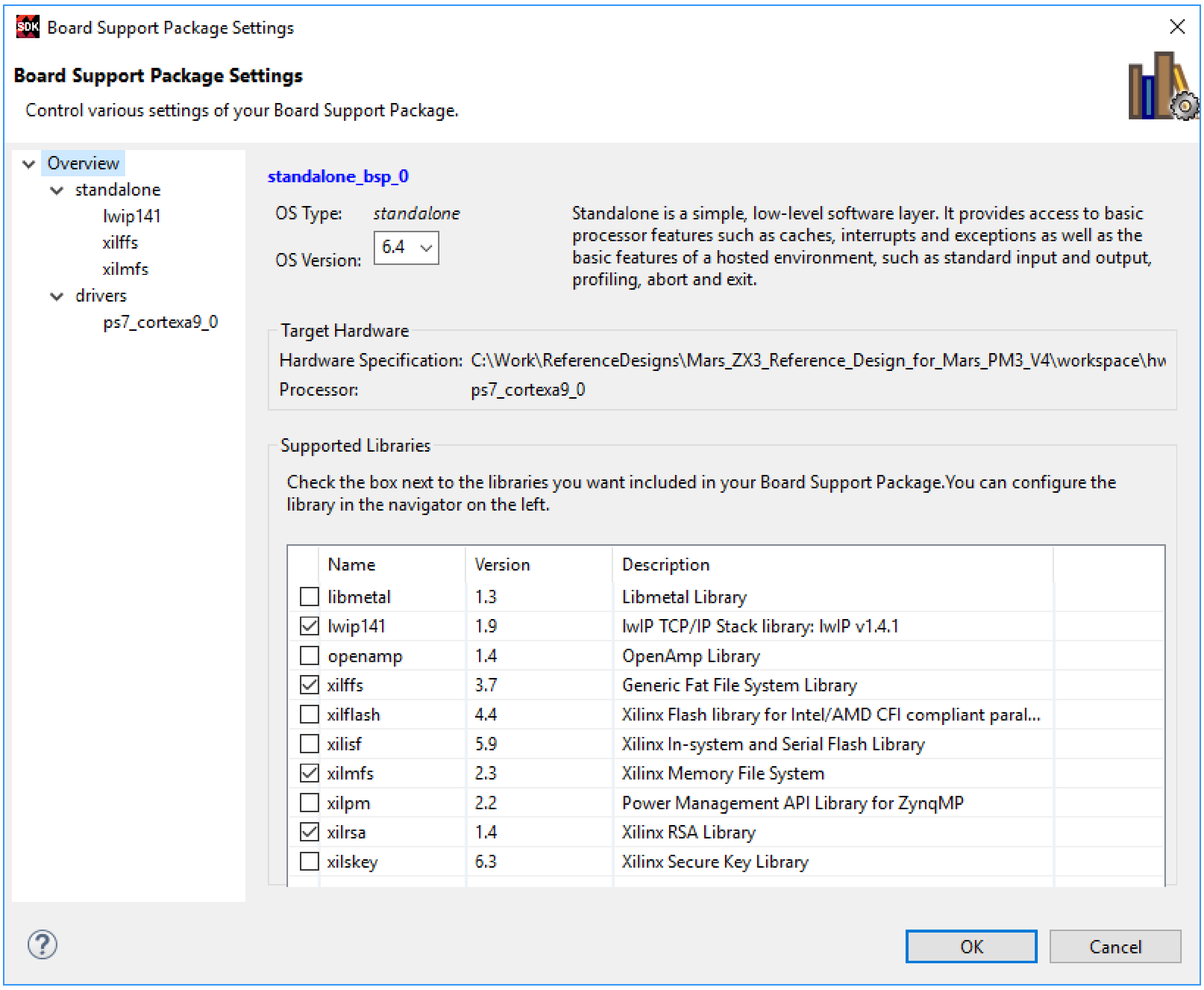


Figure 3: Board Support Package Settings

Warning!

For a successful build of the software examples the hardware project must be named "hw_platform_0" and that the BSP must be named "standalone_bsp_0".

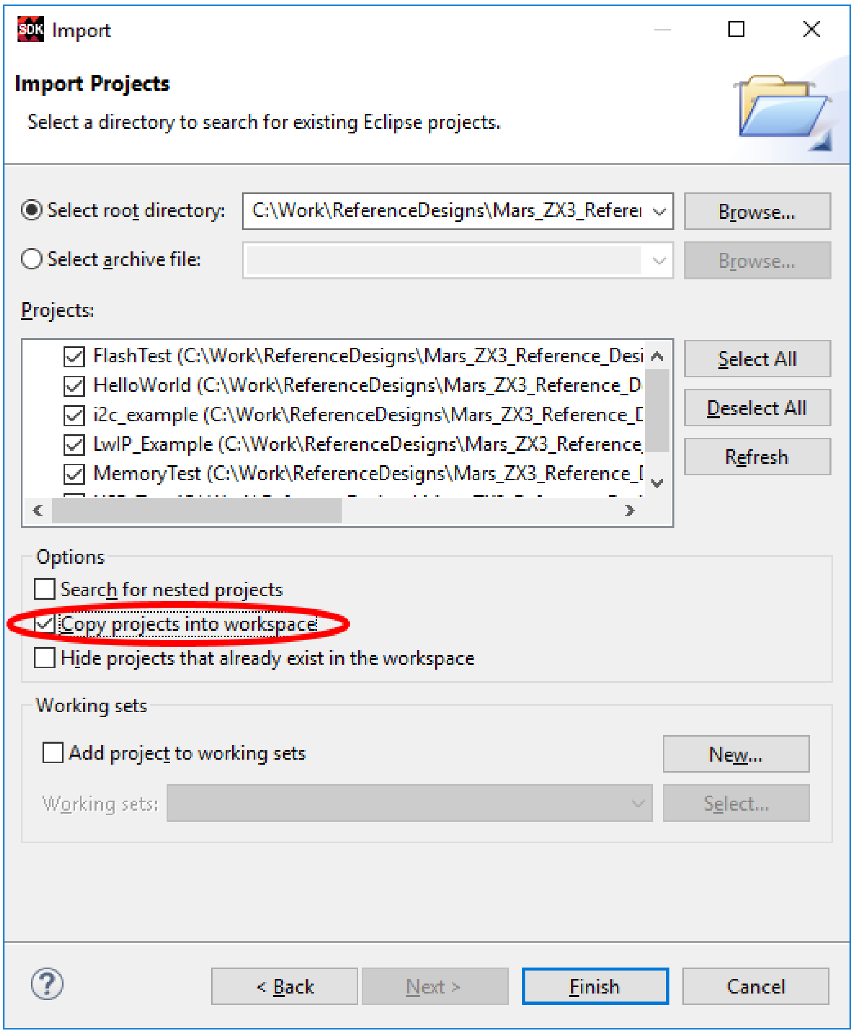


Figure 4: Importing Software Projects

Warning!

Please note that the software applications may not work properly if they are imported in another software version than the one specified in this document. A solution for such cases is creating a fresh software project with the sources provided in the reference design.

Warning!

Please make sure that during import process the checkbox for copying the projects into workspace has been enabled, otherwise the build step will fail due to incorrect file paths.

FPGA Programming

Step	Description
1	Open Xilinx SDK 2017.4: 1. Click on Xilinx Tools → Program FPGA 2. For Hardware Platform select hw_platform_0 3. For Bitstream field hit Search → select system_top.bit 4. Hit Program The configuration is shown in Figure 5. For issues related to JTAG connection, refer to Section 5.3.
2	After the FPGA is successfully configured, the **DONE** LED should be lit. For Mars PM3 base board revision 4 or older, the **RDY** LED should be lit.

Table 8: FPGA Programming Step-By-Step Guide

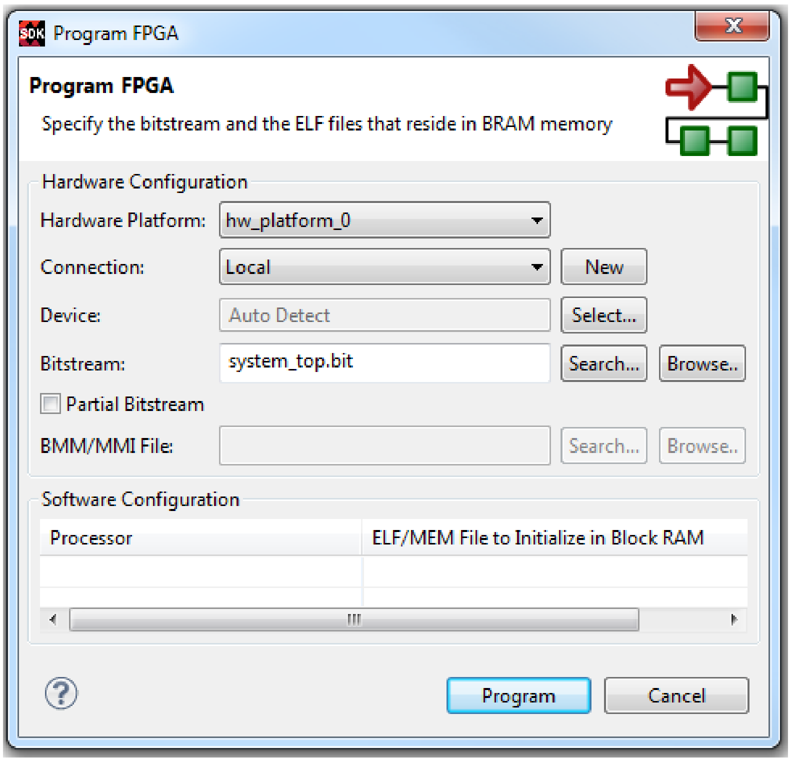


Figure 5: FPGA Programming Settings

Running Software Applications

This section describes how to run software applications on the Mars ZX3 SoC module. The steps are generic, and apply to all software examples provided in the release archive, along with this document.

In order to execute the applications, the hardware needs to be configured as described in Section 3.2.

Note that the FPGA must be programmed before running the software applications. Refer to Section 3.5 for details on FPGA programming.

<table>
<tr><th>Step</th><th>Description</th></tr>
<tr><td>1</td><td>Create a run configuration for the application in SDK 2017.4:
1. Run → Run Configurations...
2. Right-click Xilinx C/C++ application (GDB) and hit New or double-click on Xilinx C/C++ application (GDB)
3. Enter a run configuration name in the Name field (e.g. HelloWorld)
4. Target Setup tab (see Figure 6):
(a) For Hardware Platform select hw_platform_0
(b) For CPU select ps7_cortexa9_0
(c) In the Bitstream file field, hit Search...
(d) Select system_top.bit and hit OK
(e) In the Initialization file field, hit Search...
(f) Select ps7_init.tcl and hit OK
(g) Enable checkboxes Run ps7_init and Run ps7_post_config
5. Application tab:
(a) In the Project Name field click browse and select an application (e.g. MemoryTest)
(b) In the Application field click search and select an .elf file
(c) Hit Apply</td></tr>
<tr><td>2*</td><td>Optional - for the lwIP example application, the following extra steps are required:
1. Select the Application tab (see Figure 7)
2. In the Data Files to download before launch section:
(a) Click Add
(b) Select the memory file system image from the lwIP example: <code><base_dir>\software\lwip_example\image.mfs</code>
(c) In Address field type 0x8000000 and hit Open
(d) Hit Apply</td></tr>
<tr><td>3</td><td>Start the application by clicking the Run button.
In some test setup cases it was observed that the SDK tool was not able to start a second run session without a hardware reset. If required, power off and on the base board and restart the run configuration.</td></tr>
</table>

Table 9: Running an Application Step-By-Step Guide

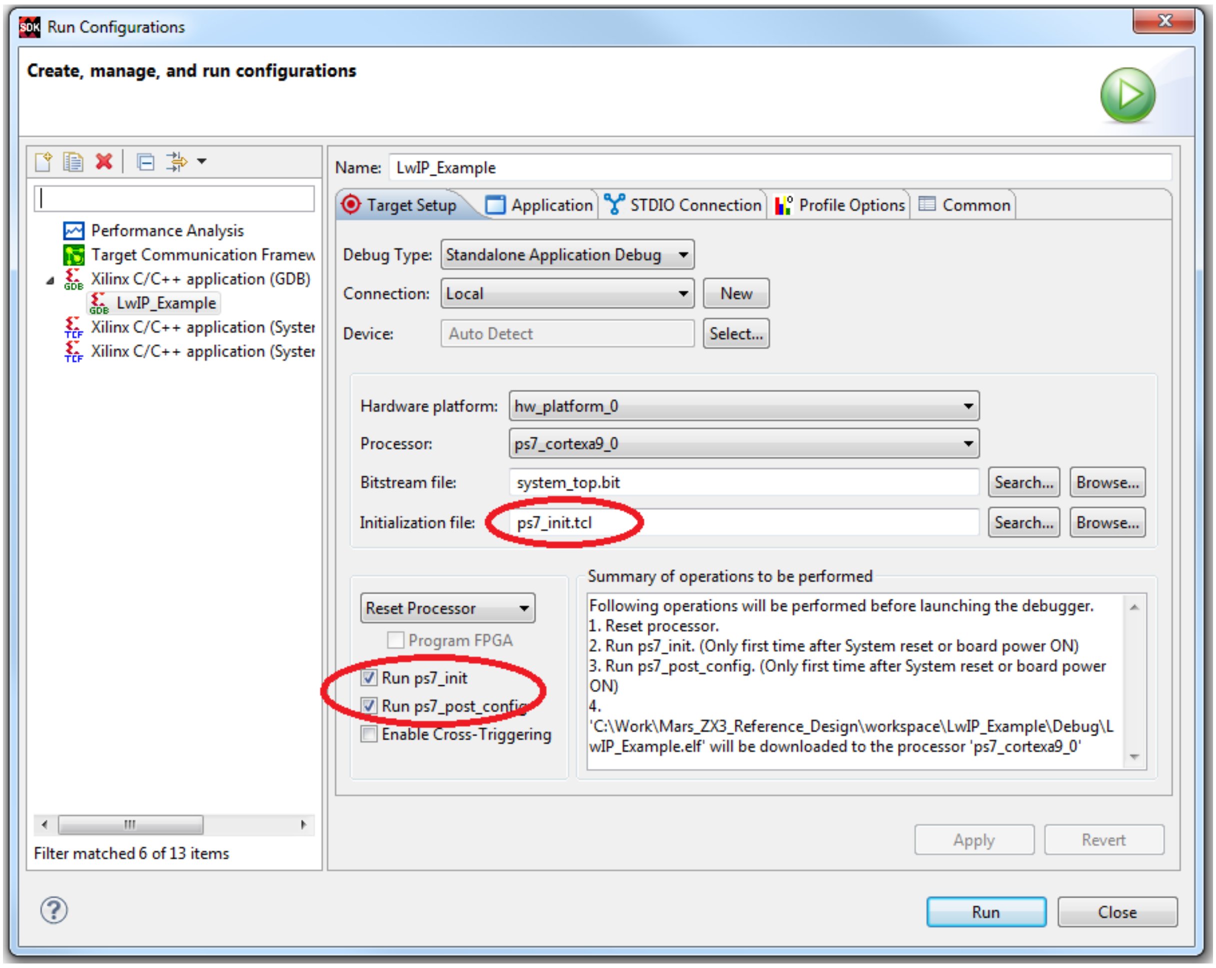


Figure 6: Run Configurations Settings

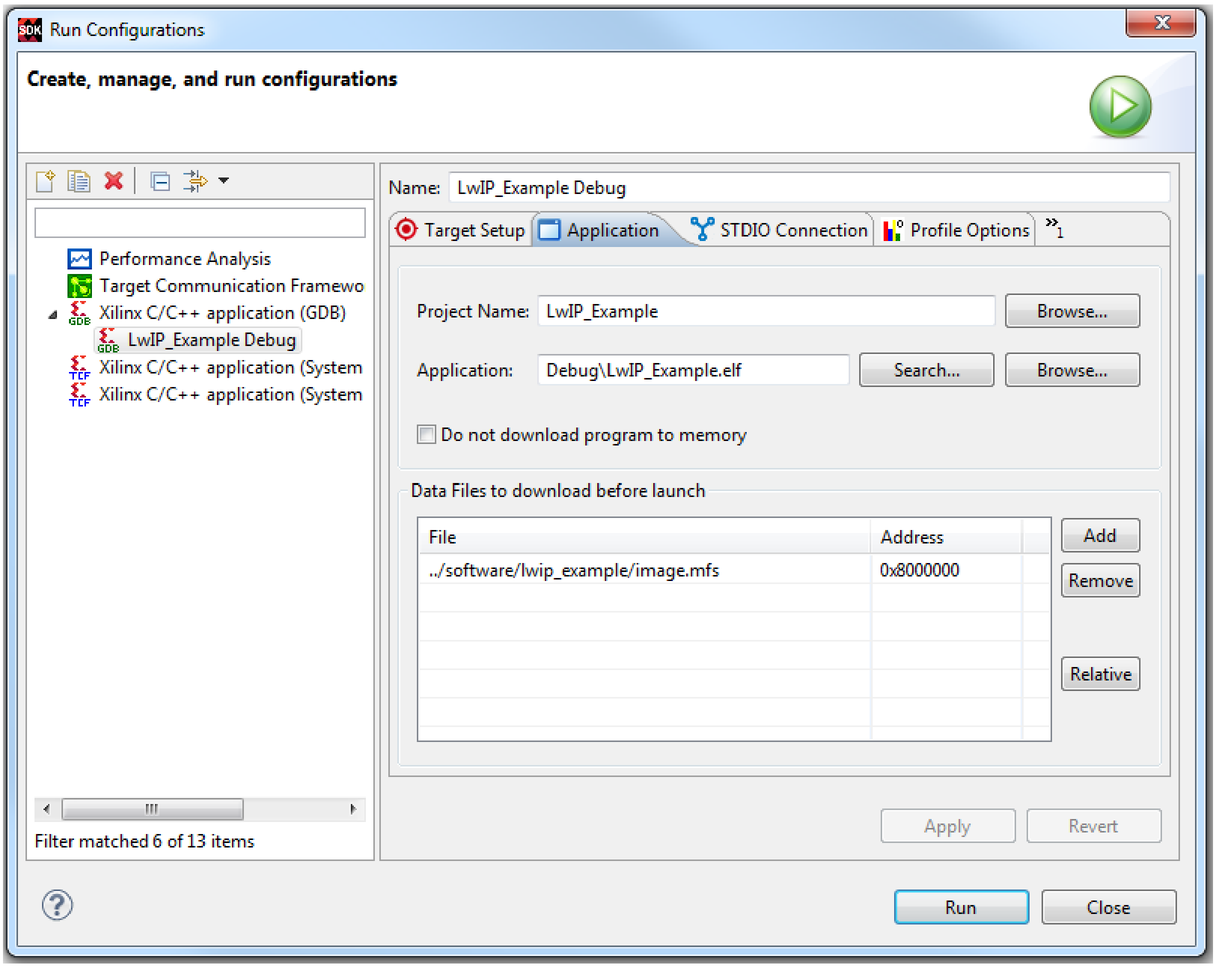


Figure 7: Run Configurations Settings for LwIP Application

Embedded Software

This section describes the software examples and the expected UART output while running these applications.

General

The Mars ZX3 SoC module reference design comes with a number of example applications, which show how to initialize the peripheral controllers and how to access the external devices. All of them are bare-metal applications that are executed in the DDR SDRAM memory.

The applications included are a Hello World example, an I2C test that reads and prints out the module and board configuration, a flash test which checks the QSPI flash memory available on the Mars ZX3 SoC module, a web server implementation example using lwIP networking stack, a USB storage emulation example, and a memory test that checks the DDR SDRAM memory available on the module.

An additional section is included to describe how to create and modify the First Stage Boot Loader (FSBL) application. This application is required during the process of creating a boot image for the module.

The procedure of importing, creating and building the example applications is explained in Sections 3.4, 3.5 and 3.6.

Hello World Application

The Hello World application is a very simple application, which is used to demonstrate all the required steps for getting a bare-metal application running on the Mars ZX3 SoC module.

The Hello World application prints „Hello World x" for twenty times, while x is incremented by one at every iteration. Figure 8 shows the UART output of the Hello World application.

```
== Enclustra Hello World Example ==

Hello World 0
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
Hello World 6
Hello World 7
Hello World 8
Hello World 9
Hello World 10
Hello World 11
Hello World 12
Hello World 13
Hello World 14
Hello World 15
Hello World 16
Hello World 17
Hello World 18
Hello World 19

Goodbye...
```

Figure 8: Hello World Application UART Output

I2C Example Application

The I2C example application demonstrates the configuration and use of the I2C controller on the Mars ZX3 SoC module.

The I2C example application reads the module configuration data from the secure EEPROM on the Mars ZX3 SoC module, date and time values from the RTC on the Mars ZX3 SoC module, as well as voltage information from the system monitor present on the Mars PM3 base board.

The module configuration includes: module type, serial number, DDR SDRAM and flash memory sizes, and first MAC address. The MAC addresses can be used by the user to configure the Ethernet MAC.

Figure 9 shows the UART output of the I2C application.

```
== Enclustra I2C test ==

EEPROM:
   Module type                  Mars ZX3
   Revision                     4
   Serial number                110269
   MAC Address 0                20:B0:F7:03:5D:7A
   SoC type                     Xilinx Zynq-7020 XC7Z020
   SoC speed grade              1
   Temperature grade            Commercial
   Power grade                  Normal
   Ethernet port count          1
   Ethernet speed               Gigabit
   Real-time clock equipped     Yes
   USB device port count        1
   DDR3 RAM size (MB)           512
   SPI flash size (MB)          16
   NAND flash size (MB)         512

Real Time Clock:
   Time: 11:22:33
   Date: 22.11.10
   Temperature: 41 Celsius

System Monitor:
   VCC_MAIN       Voltage = 12027 mV
   VCC_5V         Voltage = 5064 mV
   VCC_3V3        Voltage = 3298 mV
   VCC_IO         Voltage = 2512 mV
   VCC_1V2        Voltage = 1215 mV
   VMON_P41       Voltage = 1350 mV
   VCC_OUT        Voltage = 1770 mV


== End of test ==
```

Figure 9: I2C Example Application UART Output

Memory Test Application

The memory test application performs several tests on the DDR memory present on the Mars ZX3 SoC module. A quick simple test and a detailed full test are executed.

The simple test is run on a bigger part of the memory and checks that the sequential incrementing of the address and writing values to the memory works as expected. The full test uses several writing patterns on two different parts of the memory, then reads the values and compares the results. This test is performed on a smaller part of the memory.

Figure 10 shows the UART output of the memory test application. Note that the starting address and the memory size configured for the test depend on the module and on the application settings.

Please note that depending on the DDR memory controller speed, memory cache enable and on the test size, the memory test may take several minutes to complete.

```
== Enclustra Memory Test ==

  Testing 4MB @ Address 0xC0100000 (full test)

  Loop 1/1:
    Stuck Address      : ........ok
    Random Value       : ok
    Compare XOR        : ok
    Compare SUB        : ok
    Compare MUL        : ok
    Compare DIV        : ok
    Compare OR         : ok
    Compare AND        : ok
    Sequential Increment: ....................ok
    Solid Bits         : ........ok
    Block Sequential   : ........ok
    Checkerboard       : ........ok
    Bit Spread         : ........ok
    Bit Flip           : ........ok
    Walking Ones       : ........ok
    Walking Zeroes     : ........ok
    8-bit Writes       : ok
    16-bit Writes      : ok

  Testing 255MB @ Address 0xC0100000 (quick test)

  Loop 1/2:
    Sequential Increment: ....................ok

  Loop 2/2:
    Sequential Increment: ....................ok

== Test finished, no errors occurred ==
```

Figure 10: Memory Test Application - UART Output Example (addresses and test sizes may vary)

Flash Test Application

The flash test application is based on the Xilinx SPI controller driver example and performs a write/read-back test of one sector starting at a specific address of the QSPI flash memory. The application initializes the SPI and the interrupt controller and includes read, write and erase functions that show how to access the flash memory on the Mars ZX3 SoC module using the Xilinx SPI controller.

Figure 11 shows the UART output of the flash test application.

```
-- QSPI FLASH Interrupt Example Test --


-- Successfully ran QSPI FLASH Interrupt Example Test --
```

Figure 11: Flash Test Application UART Output

Ethernet LwIP Application

The Ethernet lwIP application represents a basic example of running a web server on the Mars ZX3 SoC module using the open-source TCP/IP networking stack lightweight IP (lwIP).
The example is based on the Xilinx lwIP example (see Xilinx Application Note [4]), modified for compatibility with the Ethernet PHY used on the Mars ZX3 SoC module.

In order to do this change, one of the files generated automatically in the board support package (xemacpsif_physpeed.c) has been modified and copied to the project sources.

Before launching the Ethernet lwIP application, the network adapter of the computer needs to be configured as described below.
It is recommended to have a second network adapter for local connections and keep the current network setup on the default adapter, in order to avoid reconfiguring the connection parameters.

Computer Setup

Step	Description
1	Connect a network cable (RJ45) between your computer and the Mars PM3 base board. Use the connector marked with label **ETH** in Figure 2.
2	Open Control Panel → Network and Internet → Network and Sharing Center
3	Click on the Local Area Connection → Properties (see Figure 12) 1. Disable all connections except Internet Protocol Version 4 (TCP/IPv4) 2. Click on Internet Protocol Version 4 (TCP/IPv4) → Properties (a) Set the IP address to 192.168.1.1 (b) Set the subnet mask to 255.255.255.0 (c) Hit OK

Table 10: Computer Network Setup Step-By-Step Guide

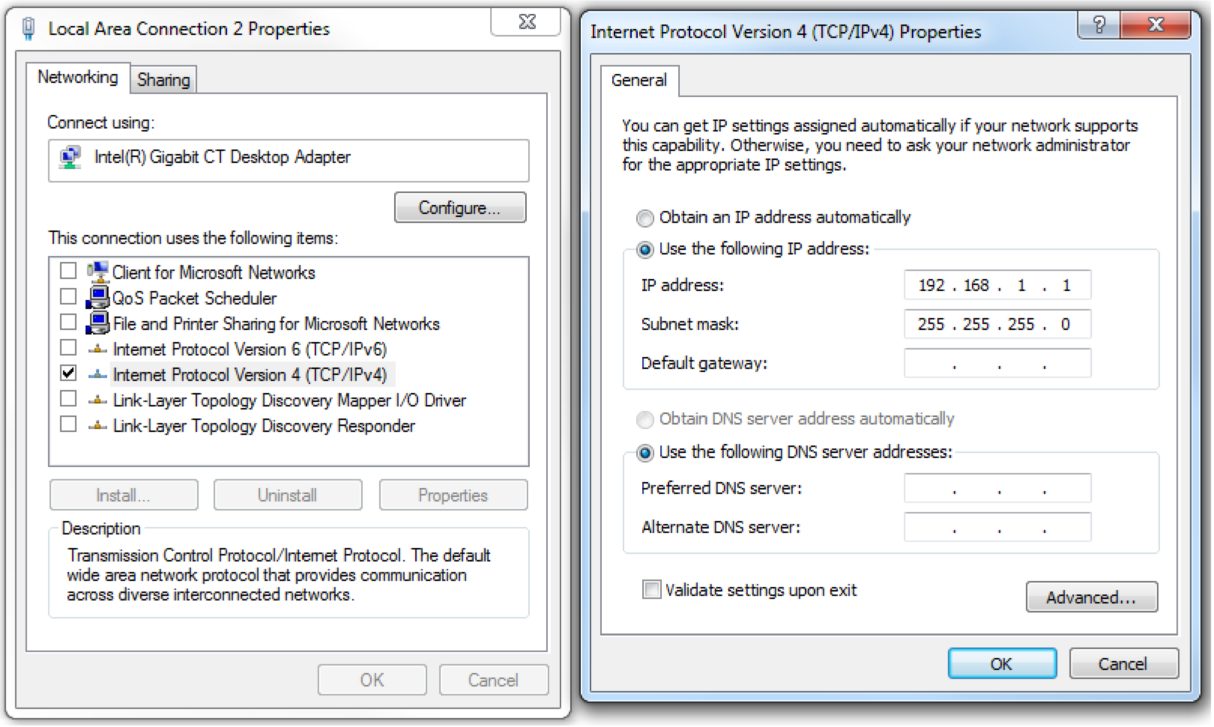


Figure 12: Network Adapter Configuration for LwIP Application

Please note that a memory file system (MFS) image must be downloaded to the RAM for the application to run. Section 3.6 describes the required steps to run the lwIP Ethernet application.

After having the computer setup, as well as the run configurations done, two tests can be performed using the lwIP application.

The testing procedure and the expected UART output are further presented.

Testing the LwIP Web Server

Step	Description
1	Open a web browser
2	Type http://192.168.1.10/ in the address bar A basic web page indicating the current state of the LEDs on the Mars ZX3 SoC module will be loaded in the browser. The user can toggle the LEDs using a button in the web page. Figure 13 illustrates the lwIP application web page.
3	Click the Toggle LEDs button, in order to change the state of the LEDs.

Table 11: Testing the LwIP Web Server Step-By-Step Guide

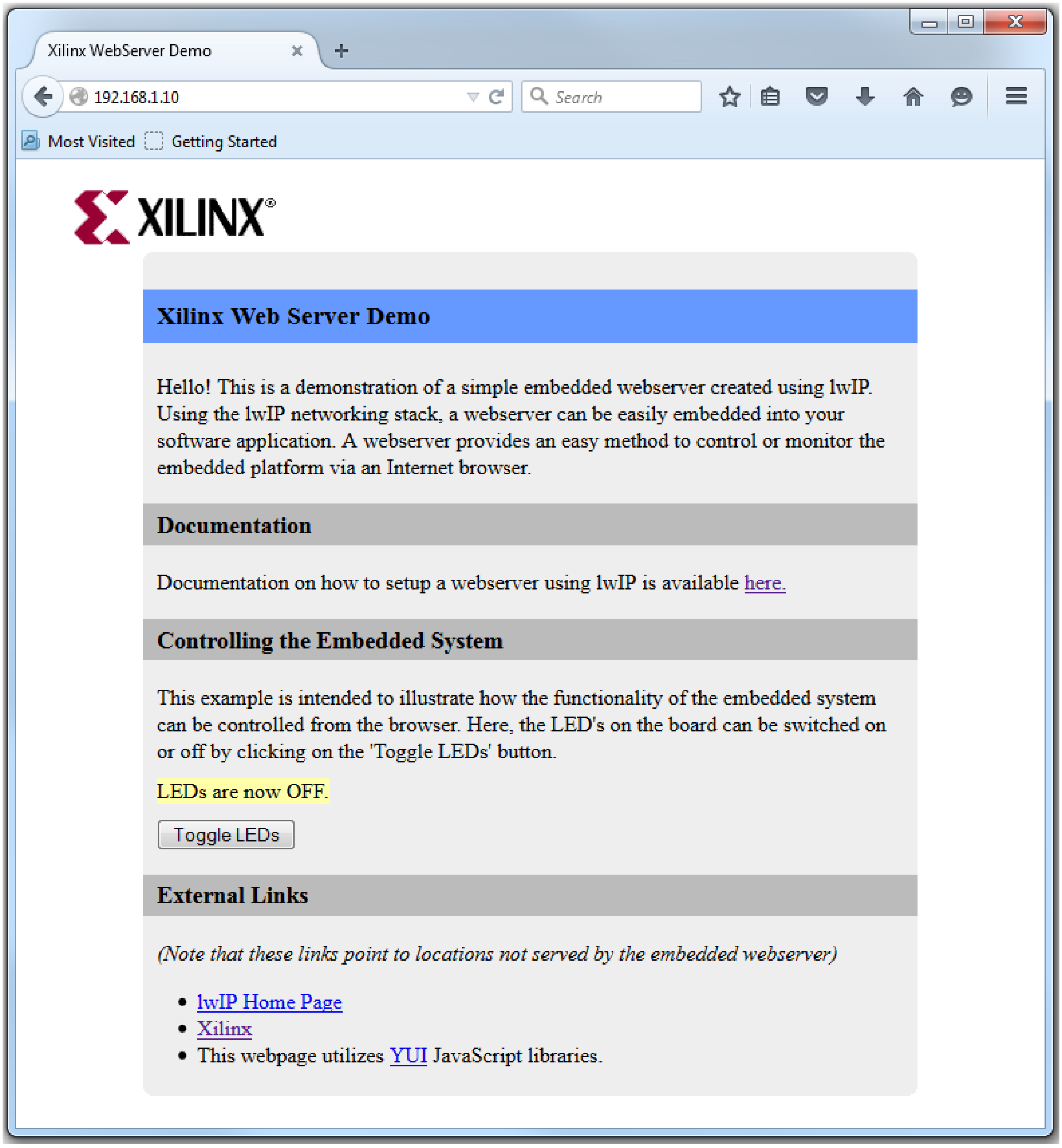


Figure 13: LwIP Application Web Page

Testing the LwIP Telnet Echo Server

Step	Description
1	Open a Telnet program (e.g. PuTTY) and configure the following parameters (see Figure 14): 1. For Host Name type 192.168.1.10 2. For Protocol select Telnet 3. For Port type 7 4. Click Open
2	Each character sent to the Mars ZX3 SoC module is immediately sent back (each character appears twice in the Telnet terminal).

Table 12: Testing the LwIP Telnet Echo Server Step-By-Step Guide

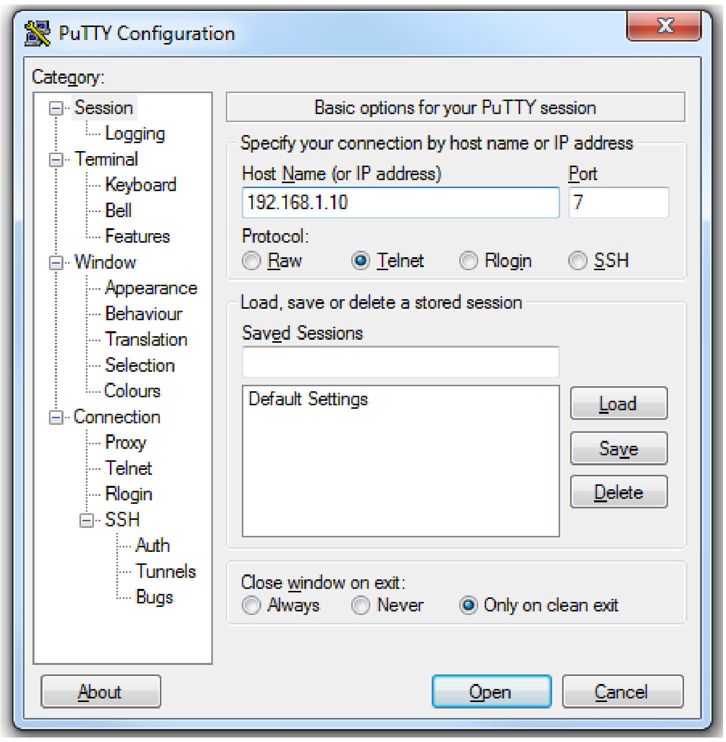
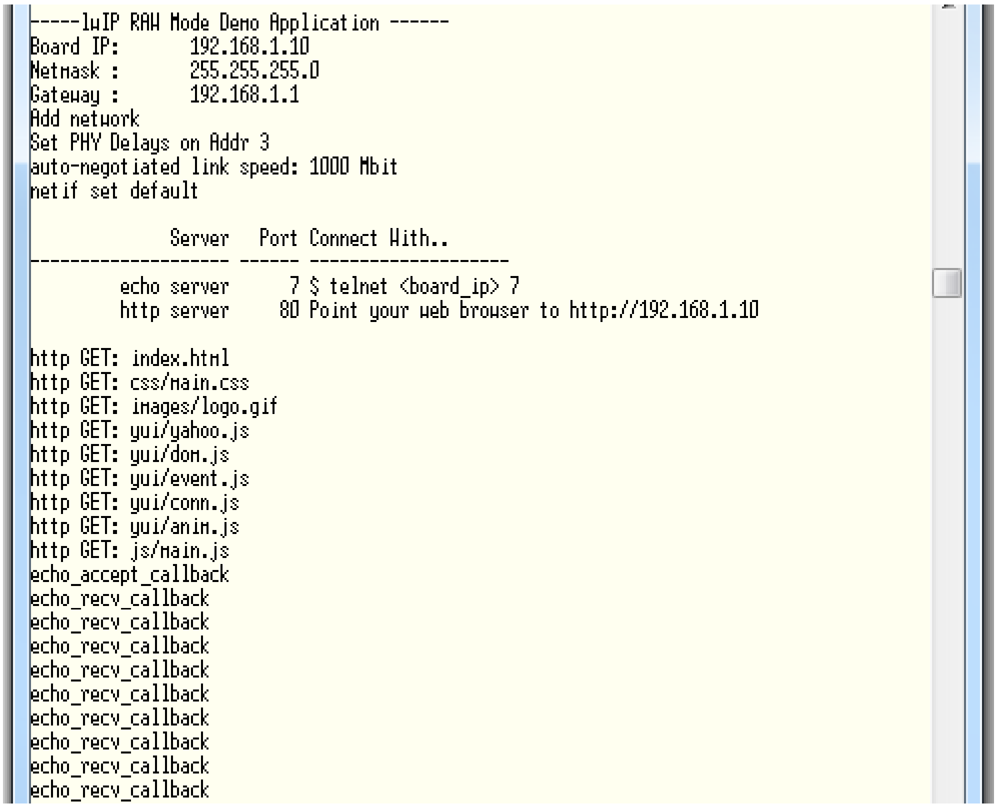


Figure 14: PuTTY Configuration for Telnet Communication

Figure 15 displays the UART output of the Ethernet lwIP application, corresponding to web page loading action and to Telnet echo calls.



```
-----lwIP RAW Mode Demo Application ------
Board IP:       192.168.1.10
Netmask :       255.255.255.0
Gateway :       192.168.1.1
Add network
Set PHY Delays on Addr 3
auto-negotiated link speed: 1000 Mbit
netif set default

              Server   Port Connect With..
-------------------- ------ --------------------
         echo server      7 $ telnet <board_ip> 7
         http server     80 Point your web browser to http://192.168.1.10

http GET: index.html
http GET: css/main.css
http GET: images/logo.gif
http GET: yui/yahoo.js
http GET: yui/dom.js
http GET: yui/event.js
http GET: yui/conn.js
http GET: yui/anim.js
http GET: js/main.js
echo_accept_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
echo_recv_callback
```

Figure 15: Ethernet LwIP Application UART Output

USB Storage Application

The USB storage application represents a basic example of a USB storage emulation.
The example is based on the Xilinx USB example (xusbps_intr_example.c), including additional printing messages.

In order to run the USB storage emulation example, two USB cables are required: one for the UART output and one for the USB storage emulation. While running the application, the USB cable for the USB storage needs to be connected to the base board. After starting the example the cable, Windows will issue a message asking the user to format the disk (see Figure 16).

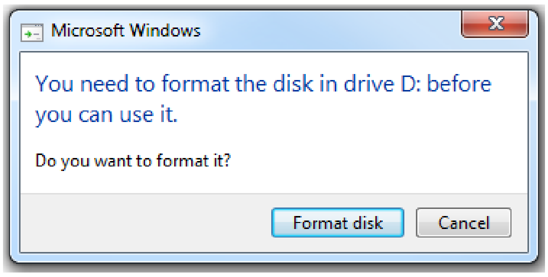


Figure 16: Windows Message after Connecting a USB Cable

After formatting the disk (capacity 1 MB, FAT), a USB storage is visible and usable as a standard removable disk in Windows.

In order to run the USB example, the following steps are required:

Step	Description
1	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: OFF, 3: ON, 4: OFF] • CFG B = [1: ON, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, set the configuration DIP switches as follows: • CFG = [1: OFF, 2: OFF, 3: ON, 4: OFF]
2	For UART output, connect a USB cable between your computer and the Mars PM3 base board. Use the micro USB port labeled **USBUB** in Figure 2.
3	For the USB storage, connect a USB cable between your computer and the Mars PM3 base board. Use the USB type B port labeled **USB** in Figure 2.
4	Start the application as described in Section 3.6
5	Format the disk and use it as standard removable disk storage.

Table 13: Running the USB Storage Example Step-By-Step Guide

Figure 17 shows the UART output of the USB storage application.

```
== Enclustra USB Test ==

Make sure the USB cable is connected to the USB connector
Windows will ask you to format the disk
```

Figure 17: USB Storage Application UART Output

First Stage Boot Loader (FSBL) Application

The First Stage Boot Loader application is used in the boot image creation process described in Section 4.1.1. It is not used as an independent application.

The FSBL application is based on the Xilinx FSBL, extended by functions that enable the RGMII delays in the Ethernet PHY and initialize the real-time clock (RTC).

In addition, the FSBL reads the Ethernet MAC address via I2C from the secure EEPROM and configures the Ethernet MAC accordingly. If the secure EEPROM cannot be accessed, a default MAC address is assigned to the MAC.

The changes mentioned above are done in fsbl_hooks.c file.

The source files provided in the ZIP archive along with this document must be copied into the software workspace, after generating the FSBL using Xilinx tools. Section 3.4 describes the required steps for FSBL project generation.

Boot Configurations

Once a software application has been developed and tested, this can be used to build a boot image for the module.

The boot image contains the FSBL, the bitstream for programming the PL, and the software bare-metal application.

In order to use a software application for the boot image, the code must be mapped for execution from the external DDR memory. If the program is mapped to the on-chip memory, it will overwrite the boot loader during execution. The example applications in the reference design are mapped for execution from the DDR memory by default.

For a fast test of the boot configurations, the pre-generated .bin images included in the <base_dir>\binaries directory may be used for boot, instead of rebuilding the image. You need to select the file corresponding to the SoC device that is equipped on the module.

QSPI Flash Boot

Generating the Image Files

Step	Description
1	Create the boot image from Xilinx SDK 2017.4 (see Figure 18): 1. Right click on the application in the Project Explorer 2. Select Create Boot Image
2	For the lwIP Application, the following extra steps are required (see Figure 19): In Create Zynq Boot Image Window hit Add ... 1. In File path field type the path to the memory file system image from the lwIP example: <base_dir>\software\lwip_example\image.mfs 2. In Load field type 0x8000000 3. Hit OK
3	In Create Zynq Boot Image Window hit Create Image. An image will be created in <workspace>\<app_name>\bootimage\BOOT.bin.

Table 14: Generating the Image Files for QSPI Flash Boot Mode Step-by-Step Guide

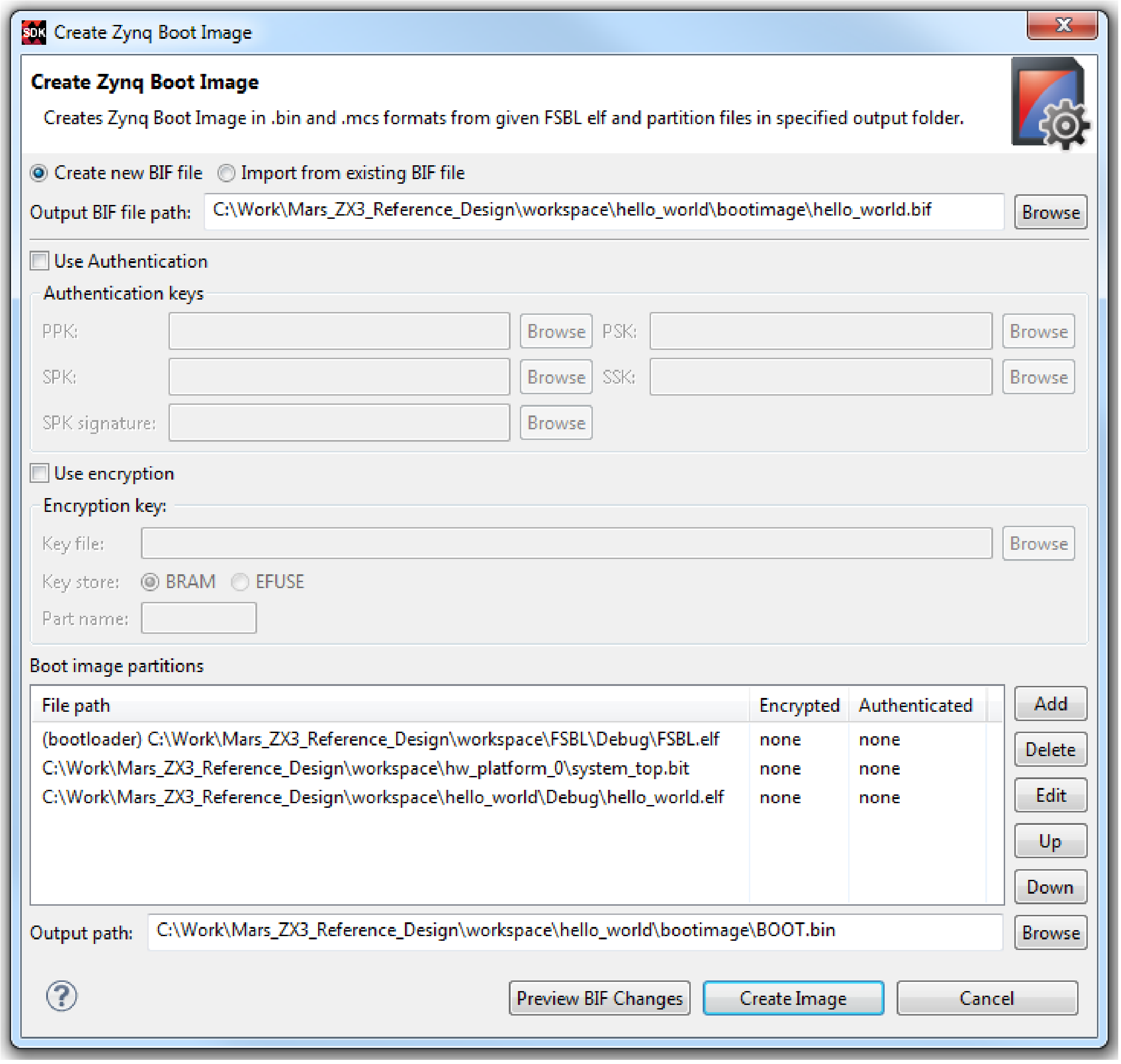


Figure 18: Create Zynq Boot Image Settings

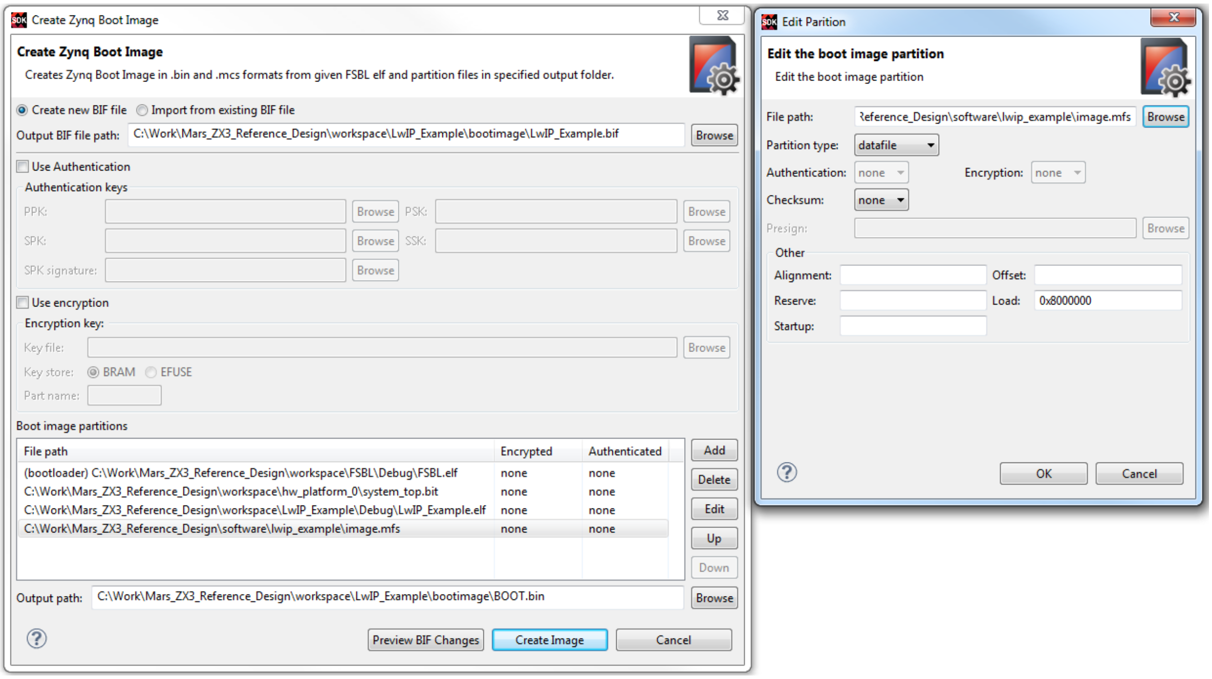


Figure 19: Create Zynq Boot Image Settings for LwIP Application

Preparing the Hardware

Step	Description
1	Remove the power supply from the Mars PM3 base board (see label **12 V DC** in Figure 2).
2	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: OFF, 3: OFF, 4: OFF] • CFG B = [1: OFF, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, enable the QSPI flash boot mode by removing the jumper between pins 5 and 6 of J801 of Mars PM3 base board or the FPGA MODE jumper located on the JTAG breakout board (see label **FPGA MODE** in Figure 2).
3	Connect the 12 V DC power supply plug to the power connector of the Mars PM3 base board (see label **12 V DC** in Figure 2).

Table 15: Preparing the Hardware for QSPI Flash Boot Mode Step-by-Step Guide

Programming the QSPI Flash

<table>
<tr><th>Step</th><th>Description</th></tr>
<tr><td>1</td><td>Open Xilinx SDK 2017.4:
1. Xilinx Tools → Program Flash
2. In Program Flash Memory window (see Figure 20):
(a) For Image File select the boot image generated as described in Section 4.1.1
(b) For FSBL File select the FSBL binary generated as described in Section 3.4
(c) For Flash Type select qspi_single
(d) Hit Program and wait for completion

The settings in the pictures are for reference only. Note that the configuration file must be selected according to your application.
Some Vivado tool versions support QSPI flash programming only in JTAG boot mode. Please check the Mars ZX3 SoC module and the Mars PM3 base board user manuals for details on how to configure this boot mode. If this is not available, please use one of the alternative methods.</td></tr>
<tr><td>2*</td><td>Optional - if SDK returns errors during flash programming or if the system does not boot properly, another option is to use Vivado 2017.4 to program the QSPI flash.
1. Flow → Open Hardware Manager
2. Click on Open target → Auto Connect
3. Right click on the corresponding FPGA device in the left bar → Add Configuration Memory Device (see Figure 21)
(a) For Select Configuration Memory Part choose the memory part according to the Mars ZX3 SoC Module User Manual [5], part type single.
This is in most cases s25fl512s-qspi-x4-single.
(b) Hit OK
4. In Program Configuration Memory Device window (see Figure 22):
(a) For Configuration file select the boot image generated as described in Section 4.1.1
(b) For Zynq FSBL select the FSBL binary generated as described in Section 3.4
(c) In Program Operations section:
• For Address Range select Entire Configuration Memory Device
• Enable checkboxes Erase, Blank Check, Program and Verify
• Hit OK and wait for completion

The settings in the pictures are for reference only. Note that the memory part and the configuration file must be selected according to your application.
Some Vivado tool versions support QSPI flash programming only in JTAG boot mode. Please check the Mars ZX3 SoC module and the Mars PM3 base board user manuals for details on how to configure this boot mode. If this is not available, please use one of the alternative methods.</td></tr>
<tr><td>3*</td><td>Optional - alternatively, Enclustra Module Configuration Tool (MCT) [7] can be used to program the QSPI flash.

Before programming the QSPI flash from MCT, make sure the hardware configuration on the Mars PM3 base board is done according to Section 4.1.2.</td></tr>
</table>

Continued on next page...

Step	Description
	Use USB 3.0 type B port (see label **USB** in Figure 2) to program the QSPI flash using MCT.

Table 16: Programming the QSPI Flash for QSPI Flash Boot Mode Step-by-Step Guide

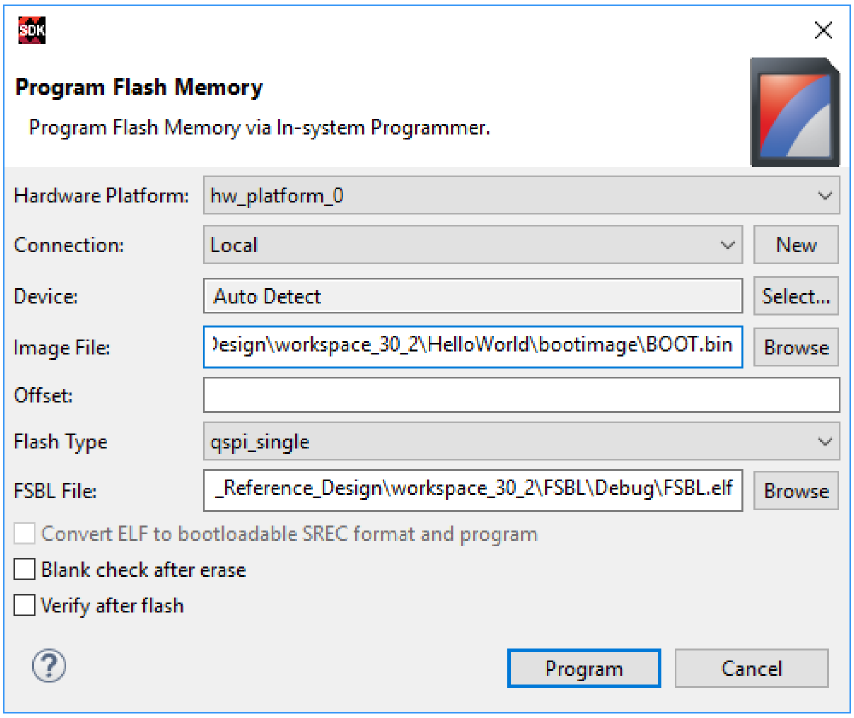


Figure 20: QSPI Flash Programming Settings in SDK

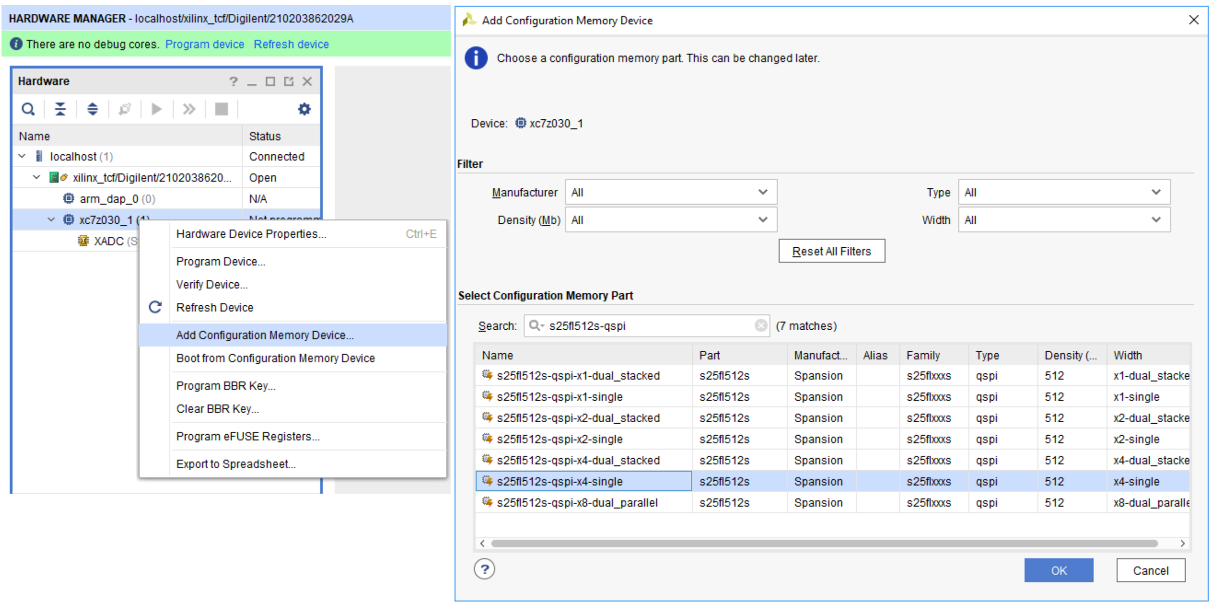


Figure 21: QSPI Flash Programming Settings in Vivado - Adding the Memory Device

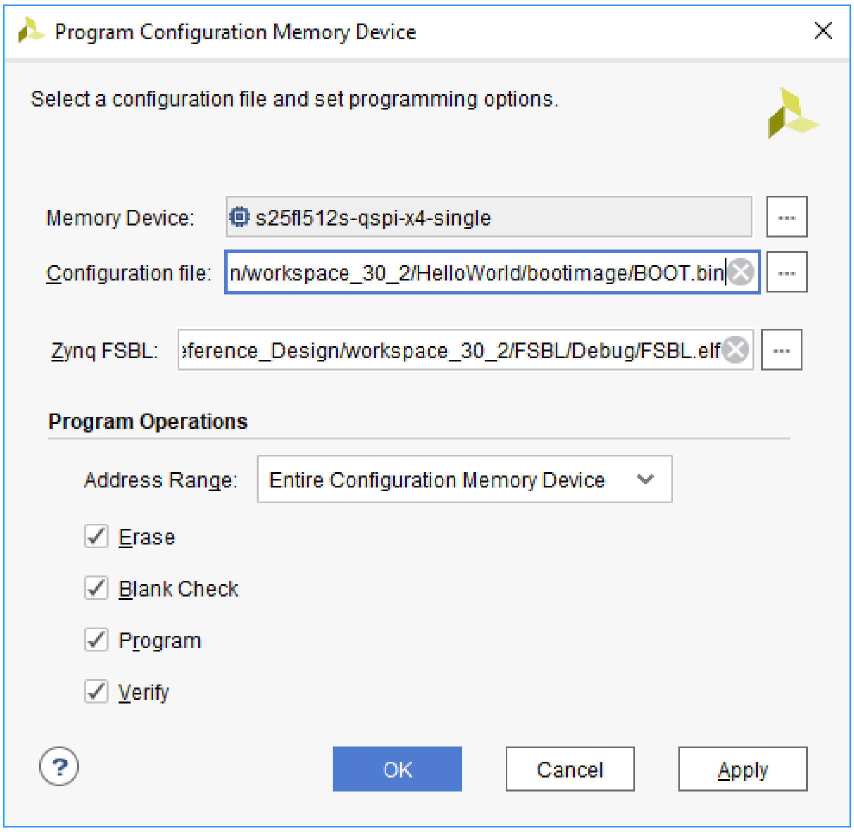


Figure 22: QSPI Flash Programming Settings in Vivado

Booting from the QSPI Flash

Step	Description
1	Check that the hardware configuration is done according to Section 4.1.2.
2	Press the power-on reset button (see label **POR** in Figure 2) and release it after a second. For Mars PM3 base board revision 4 or older, press the power-on reset button labeled **LD**.

Table 17: Booting from the QSPI Flash Step-by-Step Guide

SD Card Boot

Generating the Image Files

Please refer to Section 4.1.1 describing the steps required to generate a boot image.

Preparing the Hardware

Step	Description
1	Remove the power supply from the Mars PM3 base board (see label **12 V DC** in Figure 2).
2	Set the configuration DIP switches on the Mars PM3 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: ON, 2: OFF, 3: OFF, 4: OFF] • CFG B = [1: ON, 2: OFF, 3: OFF, 4: OFF] For Mars PM3 base board revision 4 or older, enable the SD card boot mode by mounting a jumper between pins 5 and 6 of J801 of Mars PM3 base board or the FPGA MODE jumper located on the JTAG breakout board (see label **FPGA MODE** in Figure 2).

Table 18: Preparing the Hardware for SD Card Boot Mode Step-by-Step Guide

Programming the SD Card

Step	Description
1	Write the Xilinx SD card boot image to the SD card 1. Insert the SD card into the SD card slot of your computer 2. Copy the boot image generated for your application from `<workspace>\<app_name>\bootimage\BOOT.bin` to your SD card (directly in the root directory). Note that the name of the image must be preserved.

Table 19: Programming the SD Card for SD Card Boot Mode Step-by-Step Guide

Booting from the SD Card

Step	Description
1	Insert the SD card into the SD card slot of the Mars PM3 base board (see label **SD Card** in Figure 2).
2	Connect the power supply to the Mars PM3 base board (see label **12 V DC** in Figure 2).

Table 20: Booting from the SD Card Step-by-Step Guide

Troubleshooting

Vivado Issues

- If the changes in the block design (including licenses for special IPs) are not propagated into implementation, open the Hierarchy tab in Vivado and regenerate the block design files:
  1. Right click on the block design file (.bd)
  2. Click on Reset Output Products → Reset
  3. Click on Generate Output Products → Generate → OK

SDK Runtime Exceptions

- In order to avoid runtime exceptions issued by SDK, always stop running processes with the red Terminate button. In debug mode, hit this button before resetting or disconnecting the Mars ZX3 SoC module.
- If the SDK reports runtime exceptions while downloading a program to memory, the following steps should be followed:
  1. Close SDK
  2. Shutdown any javaw, eclipse or xmd processes in Windows Task Manager
  3. Power off the Mars ZX3 SoC module
  4. Restart SDK and power on the Mars ZX3 SoC module
- After using SD card or flash boot on Zynq, the debugger will not connect to the target device anymore. In order to re-enable the debugger connection, the following steps should be followed:
  1. Remove the SD card and select the SD card boot mode, or alternatively, delete the QSPI flash and select the flash boot mode
  2. Restart the Mars ZX3 SoC module

JTAG Connection Issues

- If the JTAG cable is not detected, the following steps should be followed:
  1. Make sure that the hardware configuration is made according to Section 3.2
  2. Remove the USB connection and power supply from the Mars PM3 base board and close SDK
  3. Reconnect the USB and power supply and start SDK again
  4. Reboot the computer if the problem persists

UART Connection Issues

- If the computer is not able to recognize the USB UART on the Mars PM3 base board:
  1. Check that the USB cable is connected properly
  2. Check that the FTDI VCP drivers are installed
     (a) Open Device Manager
     (b) Universal Serial Bus controllers → USB Serial Converter A/B → Properties → Advanced tab → enable Load VCP checkbox
     (c) Reboot the computer if the COM port is still not detected
  3. Reinstall the FTDI drivers if the problem persists
- If the computer does not output any character in the terminal program:
  1. Check that the FTDI device is set to UART mode:
     (a) Download and open FT_Prog utility (this is a third party tool offered by the FTDI company to configure FTDI devices)
     (b) DEVICES → Scan and Parse

   (c) Check that for Port A and B the RS232 UART property is true
2. Check that the baud rate for the UART in the block design matches the baud rate set in the terminal program
3. If the UART used is mapped to the EMIO pins in the PS, resetting the ARM core will not suffice. Reprogramming the PL is necessary, as the UART lines go through PL.
4. Make sure that Enclustra MCT software is not open. After closing it, unplug and plug in again the USB cable corresponding to the UART communication.

QSPI Boot Issues

- If the Mars ZX3 SoC module is not able to boot from the QSPI flash:
  1. Use Vivado to program the flash
     (a) Make sure that the Memory Device part type is correctly selected
     (b) Make sure Erase and Program options are enabled
     (c) Select Entire Configuration Memory Device for Address Range
  2. If the problem persists, a possible solution is to first erase the flash, and then program it either from Vivado or SDK

  Please refer to Section 4.1.3 for details on QSPI flash programming.

List of Figures

List of Tables

References

[1] Zynq-7000 All Programmable SoC Embedded Design Tutorial, UG1165, Xilinx, 2015
[2] Vivado Design Suite User Guide, Embedded Processor Hardware Design, UG898, Xilinx, 2016
[3] Vivado Design Suite Tutorial, Embedded Processor Hardware Design, UG940, Xilinx, 2016
[4] Xilinx XAPP1026, LightWeight IP Application Examples, Xilinx, 2014
[5] Mars ZX3 SoC Module User Manual
→ Ask Enclustra for details

[6] Mars PM3 Base Board User Manual
→ Ask Enclustra for details
[7] Enclustra Module Configuration Tool (MCT)
→ Ask Enclustra for details
[8] Enclustra Modules Heat Sink Application Note
→ Ask Enclustra for details