Mercury+ MP1 SoC Module

Reference Design for Mercury+ PE1 Base Board
User Manual

Purpose

The purpose of this document is to present to the user the overall view of the Mercury+ MP1 SoC module reference design and to provide the user with a step-by-step guide to the complete Microchip® SoC design flow used for the Mercury+ MP1 SoC module.

Summary

This document first gives an overview of the Mercury+ MP1 SoC module reference design and then guides through the complete Microchip SoC design flow for the Mercury+ MP1 SoC module in the getting started section. In addition, the internals and the boot options of the Mercury+ MP1 SoC module reference design are described.

Product Information	Code	Name
Product	ME-MP1	Mercury+ MP1 SoC Module

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-491-001	2022.2_v1.0.0	30.09.2022

Approval Information	Name	Position	Date
Written by	ARUD	Design Engineer	31.08.2022
Verified by	GKOE	Design Expert	15.09.2022
Approved by	IJOS	Manager, BU SP	30.09.2022

License

Table of Contents

1   Overview

1.1   Introduction

The Mercury+ MP1 SoC module reference design demonstrates a system using the Mercury+ MP1 SoC module in combination with the Mercury+ PE1 base board. It presents the basic configuration of the device and contains a guided getting started tutorial.

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 Microchip tools is provided by the documents below:

- Libero SoC Design Suite v2022.2 Design Flow User Guide for All FPGA Families [1]
- SmartDesign v2022.2 User Guide for all FPGA Families [2]
- SoftConsole documentation v2022.2 [3]
- PolarFire® SoC MSS Configurator User Guide v2022.2 [4]

More information on the Mercury+ MP1 SoC module and the Mercury+ PE1 base board can be retrieved from their respective user manuals [8] [9].

The following directory structure applies to the MP1 Reference Design:

- `components` — Libero SmartDesign components and hierarchical SmartDesigns

- `constraints` — Libero IO constraints

- `doc` — Reference Design documentation

- `scripts` — Scripts directory required for Libero project creation

- `src` — VHDL source code and MSS configuration directory

- `sw` — SoftConsole example workspace directory

Pre-generated binaries for any MP1 variant are released on the MP1 Reference Design Github page.

1.2 Prerequisites

- IT
  - A computer running Windows 10 64-bit (or later) or a supported Linux OS (the supported Linux OS can be found in [6] under supported platforms).
- Software
  - Microchip Libero 2022.2 (check the Mercury+ MP1 SoC Module User Manual [8] for details on device support in Microchip tools)
  - Microchip SoftConsole v2022.2
  - A terminal emulation program (e.g. Tera Term)
- Hardware
  - An Enclustra Mercury+ MP1 SoC module
  - An Enclustra Mercury+ PE1 base board
- Accessories
  - A 12 V DC power supply
  - A standard micro USB cable
  - A Microchip FlashPro 6 JTAG programmer

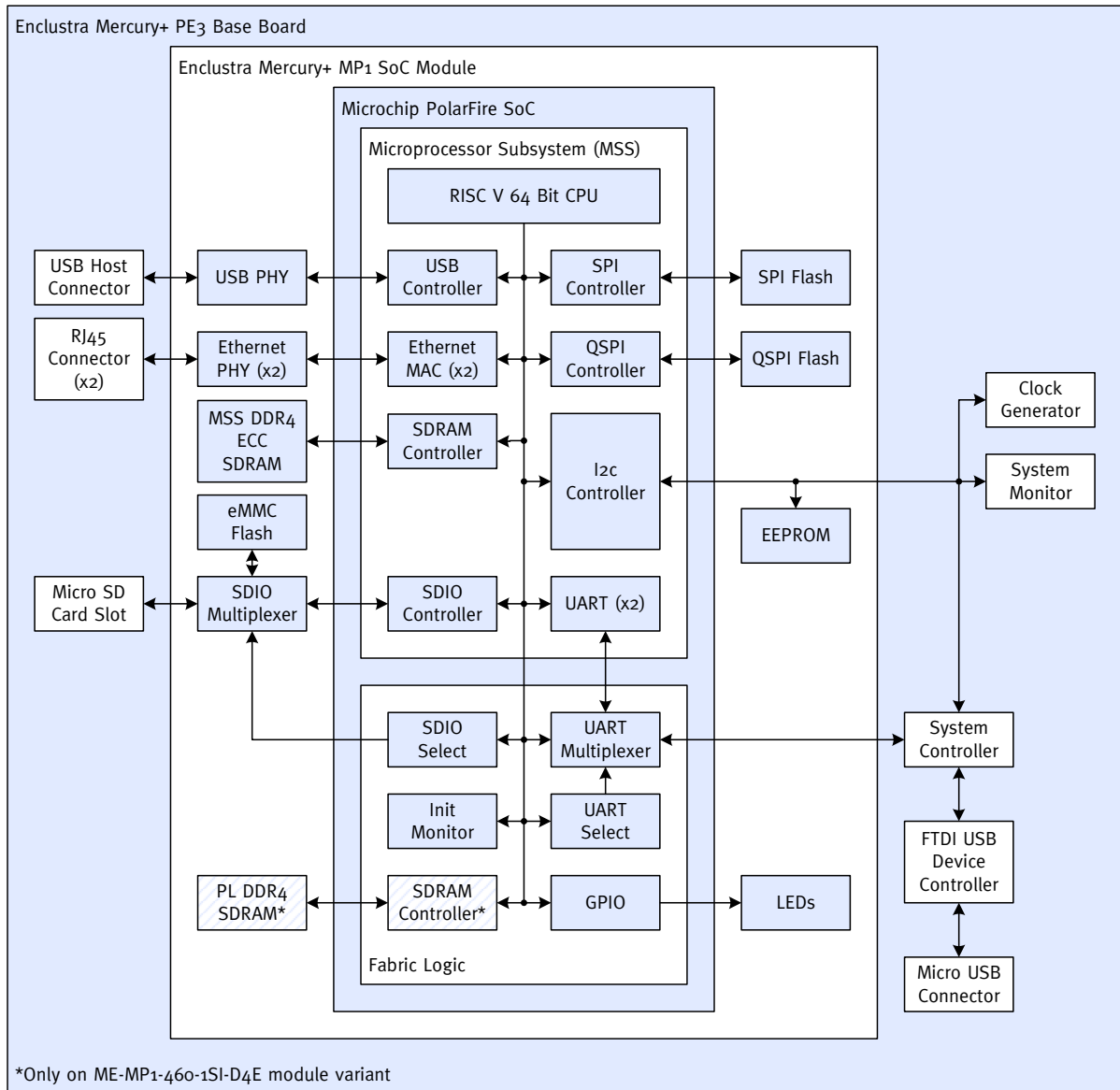# 2   Reference Design Description

Figure 1: Hardware Block Diagram

2.1   Microprocessor Subsystem (MSS)

2.1.1   Clocks

The MSS input clock frequency is configured to 125 MHz. The MSS CPU clock, MSS AXI clock and MSS AHB/APB are configured to their corresponding maximum clock frequency, as specified in the Microchip PolarFire SoC Advanced Datasheet DS0147 [5].

2.1.2   MSS DDR4 SDRAM

The DDR4 SDRAM memory runs at its corresponding maximum MSS DDR frequency. ECC RAM is enabled. The DDR clock frequency can be modified in the PolarFire SoC MSS configurator under the DDR4 section and must be configured according to the Mercury+ MP1 SoC Module User Manual [8].

2.1.3   SD Card

The SD card is configured to MSS IO bank 4 pins 0..6.

2.1.4   eMMC

The eMMC interface is configured to MSS IO bank 4 pins 0..11. This enables eMMC device access, as well as booting from the eMMC device. Please refer to section 2.1.5 for the necessary prerequisites.

2.1.5   MSS SD Card And eMMC Muxing

The SD card and eMMC interface share MSS IO pins on bank 4. Therefore, embedded software driver and fabric design support is required to allow eMMC and SD muxing. The reference design already includes the necessary fabric design support and the Enclustra HSS binary implements the embedded software driver support. The HSS binary file for the eNVM memory is included in the respective github reference design release section. Instructions for building the HSS binary manually are provided in [13].

2.1.6   I2C

The I2C controller I2C_0 is configured to use fabric IO pins in the MSS. I2C_1 is configured to use MSS IO bank 2 B and 26..27 pins. For available devices on the I2C bus refer to the Mercury+ MP1 SoC Module and Mercury+ PE1 Base Board User Manual [8] [9].

2.1.7   Quad SPI Flash Controller

The quad SPI flash controller is connected to MSS IO bank 2 and uses 30..35 pins.

2.1.8   UART

MMUART_0 and MMUART_1 are mapped to fabric IO pins and connected to the FTDI controller on the Mercury+ PE1 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

2.1.9   Ethernet

The Gigabit Ethernet MAC_0 and MAC_1 are mapped to the SGMII IOs bank 5. Both are connected to a Texas Instruments (TI) DP83867IS Ethernet PHY on the Mercury+ MP1 SoC module using an SGMII interfaces. The PHYs can be configured via the MDIO management interface on address 0 and 3.

2.1.10   USB

The USB controller on MSS IO 14..25 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 settings in the system controller and the DIP switches on the Mercury+ PE1 base board must be configured correctly. Please refer to the Mercury+ PE1 Base Board User Manual [9] for details.

2.1.11   GPIOs

The unused MSS IO pins from the MSS are available as GPIOs. GPIO pins 12..13 on GPIO_0 bank 4 are available. In addition pins 22..23 on GPIO_1 bank 2 are available as outputs to control user LEDs 0 and 1.

2.2 PolarFire Fabric Logic

2.2.1 Fabric Logic DDR4 SDRAM

The fabric logic dedicated DDR4 SDRAM memory runs at its corresponding Maximum Physical Interface (PHY) rate at a voltage of 1.2 V. The data width of this interface is 64 bits.

Note that the maximum DDR performance depends on the device speedgrade as specified in the PolarFire® SoC Advanced Datasheet DS0147 [5]. The DDR clock frequency can be modified in the settings of PF DDR4 IP core in Libero and must be configured according to the Mercury+ MP1 SoC Module User Manual [8]. Please note, that not all variants of the MP1 are equipped with fabric logic DDR4 SDRAM memory. Refer to the Mercury+ MP1 SoC Module User Manual [8] for details.

2.2.2 GPIOs

A CoreGPIO controller in the fabric logic is connected to the MSS via APB bus. Some fabric GPIOs are connected to LEDs in the top level, as described in Table 3.

The Fabric Logic firmware contains a 24-bit counter freely running at 50 MHz. The MSB of this counter is used to blink FPGA_LED0# with a frequency of approximately 3 Hz.

Fabric Logic Pin	Signal	Function
E23	FPGA_LED0#	GPIO 0, Blinking LED counter MSB, shared with MSS_LED0#
D23	FPGA_LED1#	GPIO 1, controlled by the CoreGPIO controller, shared with MSS_LED1#
A25	FPGA_LED2#	GPIO 2, controlled by the CoreGPIO controller
B25	FPGA_LED3#	GPIO 3, controlled by the CoreGPIO controller

Table 3: Fabric Logic Firmware I/O Configuration

A second CoreGPIO controller in the fabric logic is connected to the MSS via APB bus and allows controlling the UART_SEL signal. This signal is used to switch between the MMUARTs enabled in the MSS.

2.2.3 Clocks and Resets

A SmartDesgin block containing a CoreReset, ClockConditioningCircuitry and InitMonitor IP core is instantiated and used to provide the necessary reset, clock and startup signals.

2.2.4 SD card and eMMC muxing

The necessary fabric logic support for SD card and eMMC muxing is implemented using an HDL core. This IP core is connected to the MSS via APB and allows controlling the SDIO_SEL signal.

3   Getting Started

This section describes the steps required to configure the Mercury+ MP1 SoC module and Mercury+ PE1 base board in order to run a simple HelloWorld example application. The section includes information on how to:

- Mount the module and configure the Mercury+ PE1 base board
- Generate the Fabric Logic bitstream
- Prepare the software workspace
- Run a software application

3.1   Essential Information

Warning!

Always check that the mounting holes on the Mercury+ PE1 base board are aligned with the mounting holes of the Mercury+ MP1 SoC module. The base board and module may be damaged if the module is mounted the wrong way round and powered up.

If the module cannot be mounted correctly due to the mechanical collision, please contact Enclustra support.

Warning!

Never mount or remove the Mercury+ MP1 SoC module to or from the Mercury+ PE1 base board while the Mercury+ PE1 base board is powered. Always remove or turn off the power supply before mounting or removing the Mercury+ MP1 SoC module.

Warning!

Please read carefully the Mercury+ MP1 SoC module and Mercury+ PE1 base board user manuals before proceeding.

Warning!

Depending on the user application, the Mercury+ MP1 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 make sure that a single JTAG adapter is connected to the base board and enabled at a given moment, otherwise the development tools may report errors during JTAG connecting attempts.

Note that when Enclustra MCT [10] is used for SoC configuration or flash programming, all other tools that may be connected to the FTDI device (e.g. FlashPro Express, SoftConsole, UART terminal) must be closed.
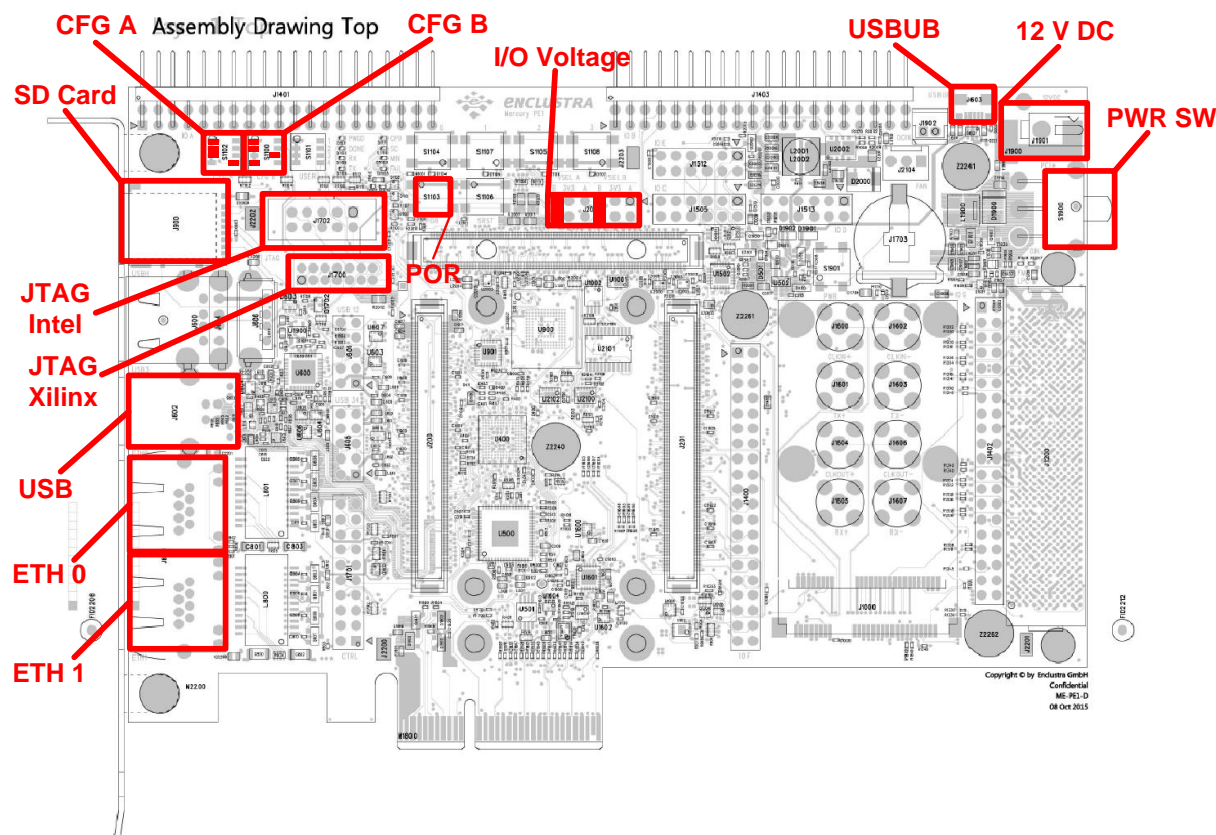
3.2 Hardware Setup

Figure 2: Mercury+ PE1 Base Board Assembly Drawing (Top View)

Step	Description
1	Set the I/O voltage jumpers on the Mercury+ PE1 base board according to label **I/O Voltage** in Figure 2 (the jumpers are marked with red rectangles): • VSEL A = 1.8 V (position B) • VSEL B = 1.8 V (position B)
2	Set the configuration DIP switches on the Mercury+ PE1 base board as follows (see labels **CFG A** and **CFG B** in Figure 2): • CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON] • CFG B = [1: OFF, 2: OFF, 3: ON, 4: OFF]
3	Mount the Mercury+ MP1 SoC module to the Mercury+ PE1 base board. Make sure that the mounting holes of the Mercury+ MP1 SoC module are aligned with the mounting holes of the Mercury+ PE1 base board before proceeding.
4	Connect the micro USB cable between your computer and the Mercury+ PE1 base board. Use the micro USB port labeled **USBUB** in Figure 2.

Continued on next page...

Step	Description
5	Connect the 12 V DC power supply plug to the power connector of the Mercury+ PE1 base board (see label **12 V** in Figure 2).
6	Connect the FlashPro6 Microchip JTAG signals to the JTAG connector of the Mercury+ PE1 base board (see label **JTAG** in Figure 2. Please ensure you connect the JTAG signals according to Figure 3.
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 higher number from the two newly detected ports. For issues related to COM ports detection, refer to Section 4.2. Configure the UART parameters according to Section 2.1.8.

Table 4: Hardware Setup Step-By-Step Guide

Warning!

The JTAG pins on the Mercury+ PE1 base board connector and the default FlashPro6 connector do not match. The correct connection is shown in Figure 3.
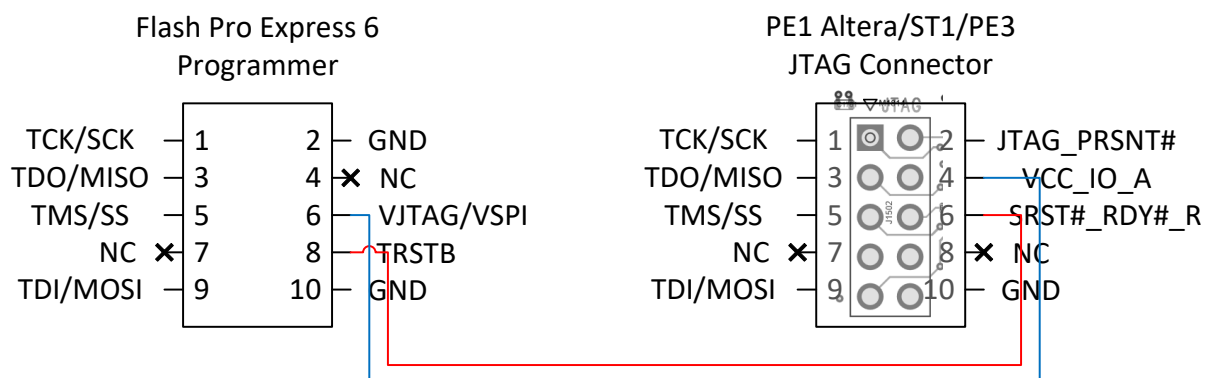

Figure 3: Mercury+ PE1 Base Board Connection between FlashPro 6 JTAG connector and Mercury+ PE1 base board JTAG connector

3.3   FPGA Bitstream Generation

For a fast test of the HelloWorld example application, the pre-generated bitstream may alternatively be used, therefore the steps described in this section may be skipped.

A pre-generated bitstream for any MP1 variant is released on the MP1 Reference Design Github page.

Step	Description

Continued on next page...

Step	Description
1	Configure the settings file: 1. Edit the `module_name` variable in `<base_dir>/scripts/settings.tcl` file, according to your modules name. This file includes module name and board information required for the project creation script. All settings, except for `module_name` should be left on default. The list of options for `module_name` is given in the comments within the Tcl file. 2. Save the file after editing.
2	Start Microchip Libero SoC 2022.2 and create the Mercury+ MP1 SoC module reference design project: 1. Click on "Project" in the top left corner 2. Select "Execute Script..." 3. Navigate to the "create_reference_design.tcl" script that comes with the reference design 4. Click on "Run" 5. Wait for completion (it will take some time until everything is set up) 6. If Libero SoC crashes during project creation and no report screen is shown after running the script, please refer to the Libero troubleshooting section 4.1.
3	Derive timing constraints in Libero SoC 2022.2: 1. In "Design Flow" open the "Manage Constraints" window 2. Navigate to "Timing" 3. Click on "Derive Constraints" and wait for completion 4. When prompted whether to associate the derived constraint for 'Synthesis', 'Place and Route' and 'Timing Verification', →Yes
4	Generate the bitstream: 1. Run "Synthesize" and wait for completion. The warnings regarding not connected top level ports can be ignored. The reason for these warnings is that not all top level ports are used in the reference design. 2. Run "Place and Route" and wait for completion 3. Click on "Generate FPGA Array Data" and wait for completion. 4. (Optional) Add HSS binary to eNVM section. This step is only necessary if eMMC/SD muxing is needed, e.g. for Linux deployment. Follow the steps described in [14] 5. Click on "Generate Design Initialization Data" and wait for completion 6. Cick on "Generate Bitstream"
5	Export FlashPro Express Job for easier deployment: 1. Click on "Export FlashPro Express Job" 2. Leave all settings as default →OK 3. The generated FlashPro Express Job can be used to create a FlashPro Express project with the FPExpress tool

Table 5: FPGA Bitstream Generation Step-By-Step Guide

3.4 FlashPro Express Deployment

This section describes how to create the FlashPro Express project from the previously generated FlashPro Express Job file.
Furthermore, the generated bitstream is programmed onto the device to prepare the hardware for the software deployment. A pre-generated job file of the reference design is released on the MP1 Reference Design Github page.

| Step	Description
1	Create the FlashPro Express project 1. Create an empty folder (e.g. `<base_dir>/build/FlashProExpressProject`) 2. Open the FPExpress application 3. Project →New Job Project 4. The dialog shown in figure 4 opens 5. Browse to the FlashPro Express Job file generated in 3.3 (typically `<base_dir>/build/Libero/<module_name>/designer/<reference_design>/export/*.job`) →Open 6. For project location choose the previously generated empty folder →Open Folder 7. Click on "OK" and wait for completion 8. The generated project should look similar to figure 5
2	Program the bitstream 1. Choose "PROGRAM" from the available options 2. Click on "RUN" and wait for completion

Table 6: FlashPro Express Step-By-Step Guide
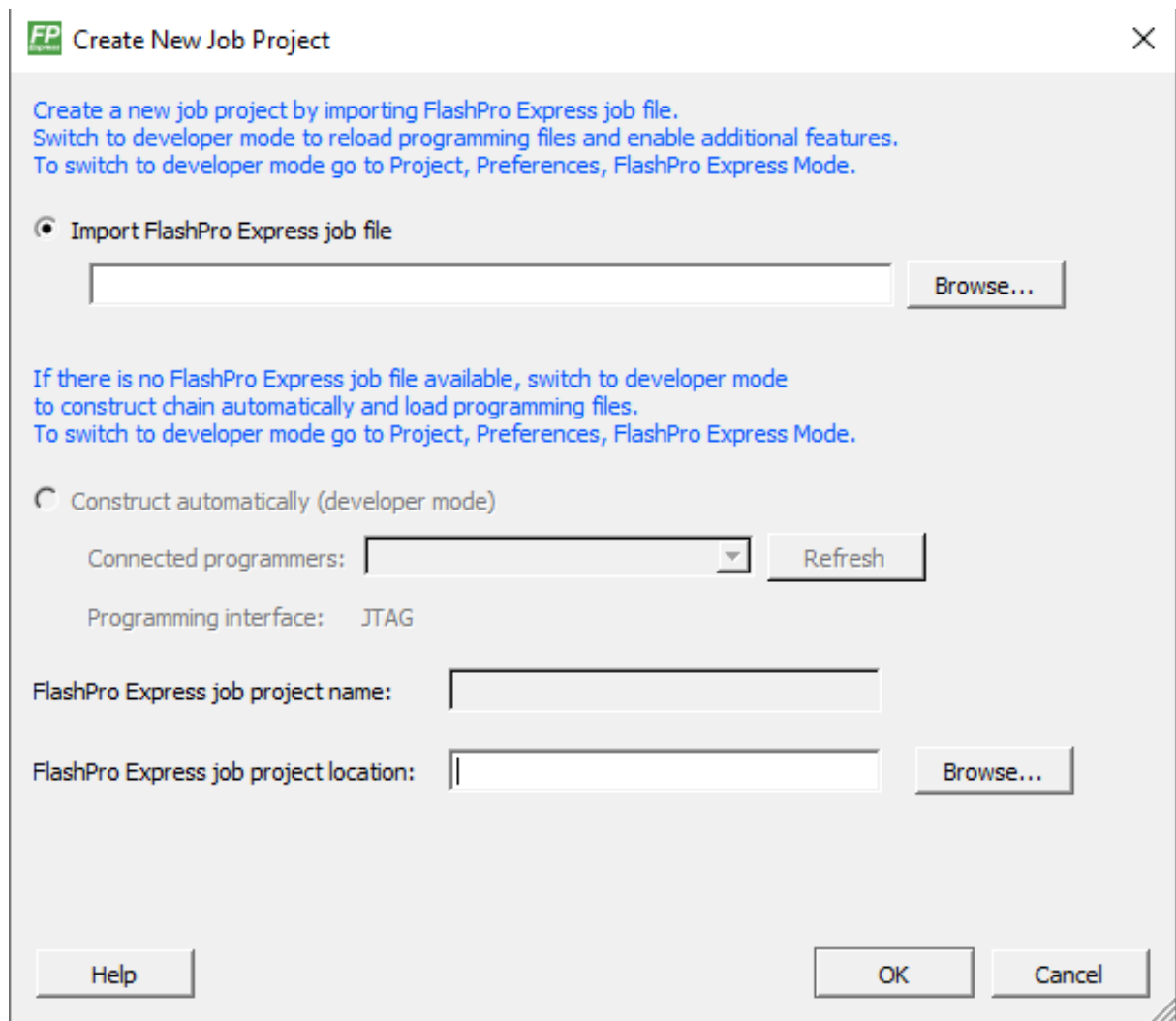
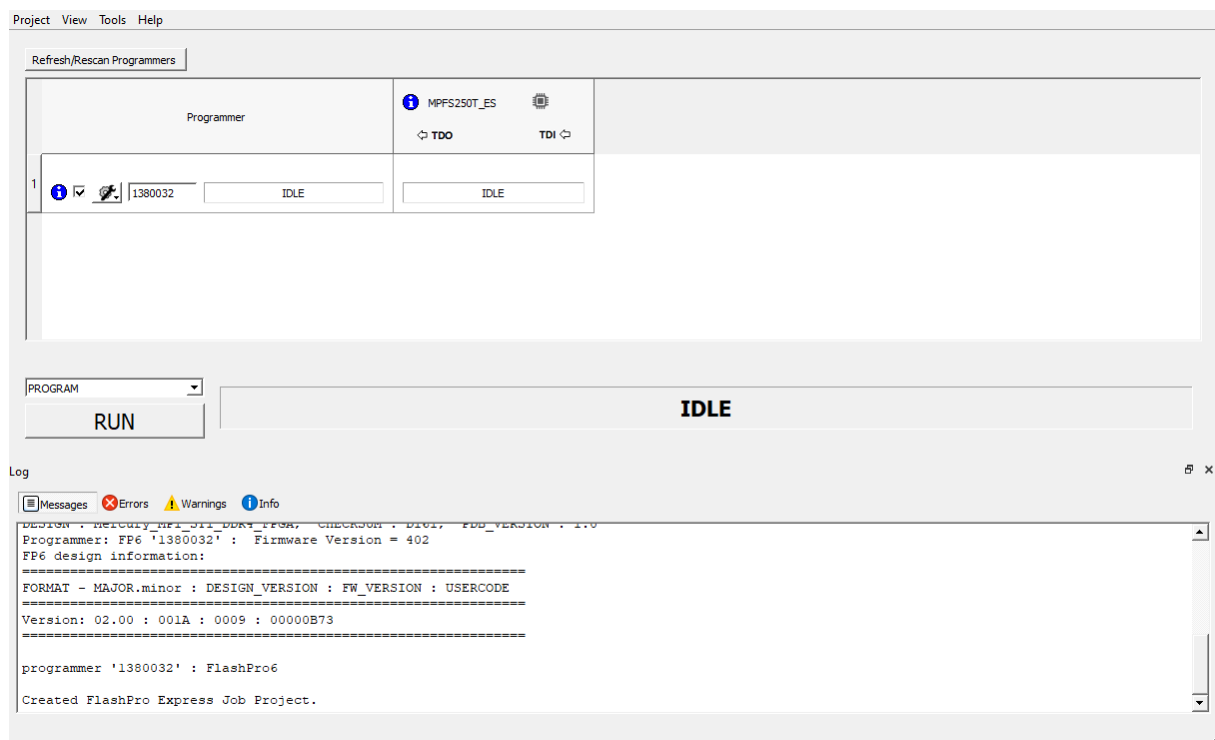*Figure 4: FlashPro Express Create New Job Project dialog*

Figure 5: Generated FlashPro Express project

3.5 SoftConsole Workspace Preparation

This section describes how to create and run software example applications. The HelloWorld example project is a modified version of the Microchip PolarFire Bare Metal MMUART example [7].

A pre-generated binary file of the HelloWorld example application is released on the MP1 Reference Design Github page.

Step	Description
1	Use the empty workspace provided by the SoftConsole installation: 1. Create an empty folder (e.g. `<base_dir>/build/SoftConsole`) 2. Extract the contents of `<path-to-SoftConsole>/extras/workspace.empty.zip` the folder created in the previous step 3. Open SoftConsole 2022.2 4. Choose the created workspace directory (the one containing .metadata folder) as the SoftConsole workspace →Launch
2	Import existing project into workspace: 1. →File →Import… →Existing Projects into Workspace →Next 2. Select the prepared project for your module variant as the root directory for the import (`<base_dir>/sw/<module_name>/\AppName`) 3. Click "Finish" 4. The prepared workspace should look like figure 6

Continued on next page…

Step	Description
3	Build and deploy the application: 1. The default build configuration is LIM-Debug. An overview over the different build configurations is given in [7]. 2. Press Ctrl+B and wait for completion

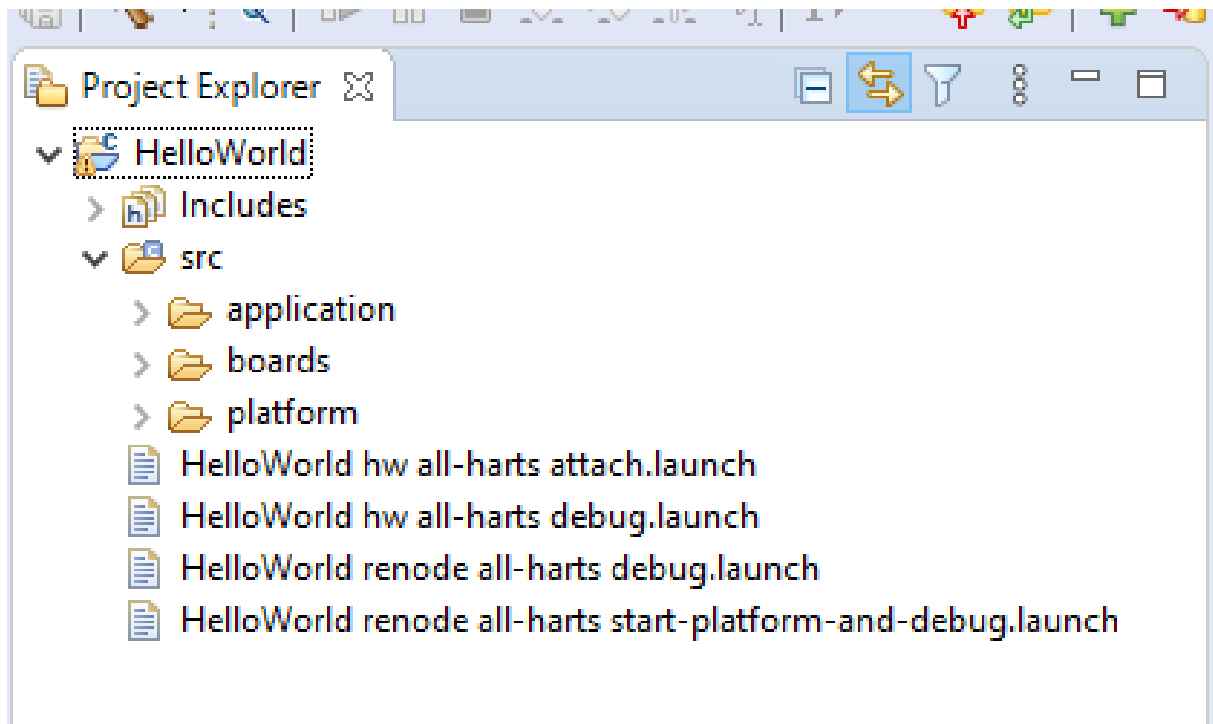*Table 7: SoftConsole Workspace Preparation Step-By-Step Guide*

Figure 6: HelloWorldSoftConsole workspace

3.6   Running Software Applications

This section describes how to run software applications on the Mercury+ MP1 SoC module. The steps are generic for any software application.

Step	Description
1	Use the provided debug configuration: 1. To run the application right click the project →Debug as →Debug Configurations… 2. From the available templates choose "HelloWorldhw all-harts debug" and click "Debug" 3. Check if the debugger pauses during Debug and click on "Resume" (F8) 4. The output is shown in figure 7. 5. You can try the different command line options to transfer messages over UART via the different methods shown in the serial terminal

Table 8: Running an Application Step-By-Step Guide

Figure 7: HelloWorldserial temrinal output

4 Troubleshooting

4.1 Libero Issues

- Recreating the reference design in Libero SoC GUI sometimes crashes. As an alternative the create_reference_design.tcl script can be executed in shell mode:
    1. Open a terminal in the directory containing the reference design
    2. Execute the command `<path-to-Libero>/libero script:create_reference_design.tcl`
    3. Wait for completion (depending on the terminal there will be output messages during creation or not)
    4. Open Libero SoC 2022.2 and click on "Open Project" (CTRL + O)
    5. Navigate to the generated reference design project file in `<base\_dir>/build/Libero/<module_name>/`

4.2 UART Connection Issues

- If the computer is not able to recognize the USB UART on the Mercury+ PE1 base board:
    1. Check that the USB cable is connected properly
    2. Check that the FTDI VCP drivers are installed
        (a) Disconnect all JTAG connections
        (b) Open Device Manager
        (c) Universal Serial Bus controllers → USB Serial Converter A/B → Properties → Advanced tab → enable Load VCP checkbox
        (d) 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. 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.

List of Figures

List of Tables

References

[1] Libero SoC Design Suite v2022.2 Design Flow User Guide for All FPGA Families
[2] SmartDesign v2022.2 User Guide for all FPGA Families
[3] SoftConsole documentation v2022.2
[4] PolarFire® SoC MSS Configurator User Guide v2022.2
[5] PolarFire® SoC Advanced Datasheet DS0147
[6] Libero tool page
[7] Github Microchip Bare Metal Examples
[8] Mercury+ MP1 SoC Module User Manual
      → Ask Enclustra for details
[9] Mercury+ PE1 Base Board User Manual
      → Ask Enclustra for details
[10] Enclustra Module Configuration Tool (MCT)

[11] Enclustra Modules Heat Sink Application Note
      → Ask Enclustra for details
[12] Enclustra Application Notes
      → I2C Application Note
      → Gigabit Ethernet Application Note

[13] Enclustra Github Hart Software Services

[14] eNVM HSS binary deployment