Mercury+ MP1 SoC Module

Reference Design for Mercury+ PE3 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 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
Module	ME-MP1	Mercury+ MP1 SoC Module
Baseboard	ME-PE3	Mercury+ PE3 Base Board

Document Information	Reference	Version	Date
Reference / Version / Date	D-0000-491-002	2024.2_v1.1.0	07.03.2025

Approval Information	Name	Position	Date
Written by	ARUD	FPGA/SoC Senior Embedded Software Engineer	07.03.2025
Verified by	ABUE	FPGA/SoC Senior Embedded Software Engineer	07.03.2025
Approved by	IJOS	Manager, BU SP	07.03.2025

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+ PE3 base board. It presents the basic configuration of the device and contains a getting started tutorial.

A troubleshooting section is included at the end of the document to help users resolve potential issues related to board connectivity and system functionality.

An introduction to the Microchip tools is available online:

- Libero SoC Design Suite v2024.2 Design Flow User Guide [9]
- SmartDesign v2024.2 User Guide [10]
- SoftConsole Documentation v2022.2 [11]
- PolarFire® SoC MSS Configurator User Guide v2024.2 [12]

More information on the module and the base board can be found in the Mercury+ MP1 SoC Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3]. The following directory structure applies to the reference design:

Directory	Description
components	Libero SmartDesign components and hierarchical SmartDesigns
constraints	Libero IO constraints
doc	Reference design documentation
scripts	Scripts directory required for project creation and settings
src	VHDL source code and MSS configuration directory
sw	SoftConsole example workspace directory

Table 1: Directory structure

1.2 Prerequisites

- IT infrastructure
  - Computer
  - Supported OS[1]
- Software
  - Microchip Libero 2024.2
  - Microchip SoftConsole v2022.2
  - Terminal emulation program, for example, Tera Term
- Hardware
  - Enclustra Mercury+ MP1 SoC module
  - Enclustra Mercury+ PE3 base board
- Accessories
  - 12 V DC power supply
  - Micro-USB cable
  - Microchip FlashPro 6 JTAG programmer

---

[1]A comprehensive list of supported operating systems is given in the Libero SoC Software and License Installation Guide [14].
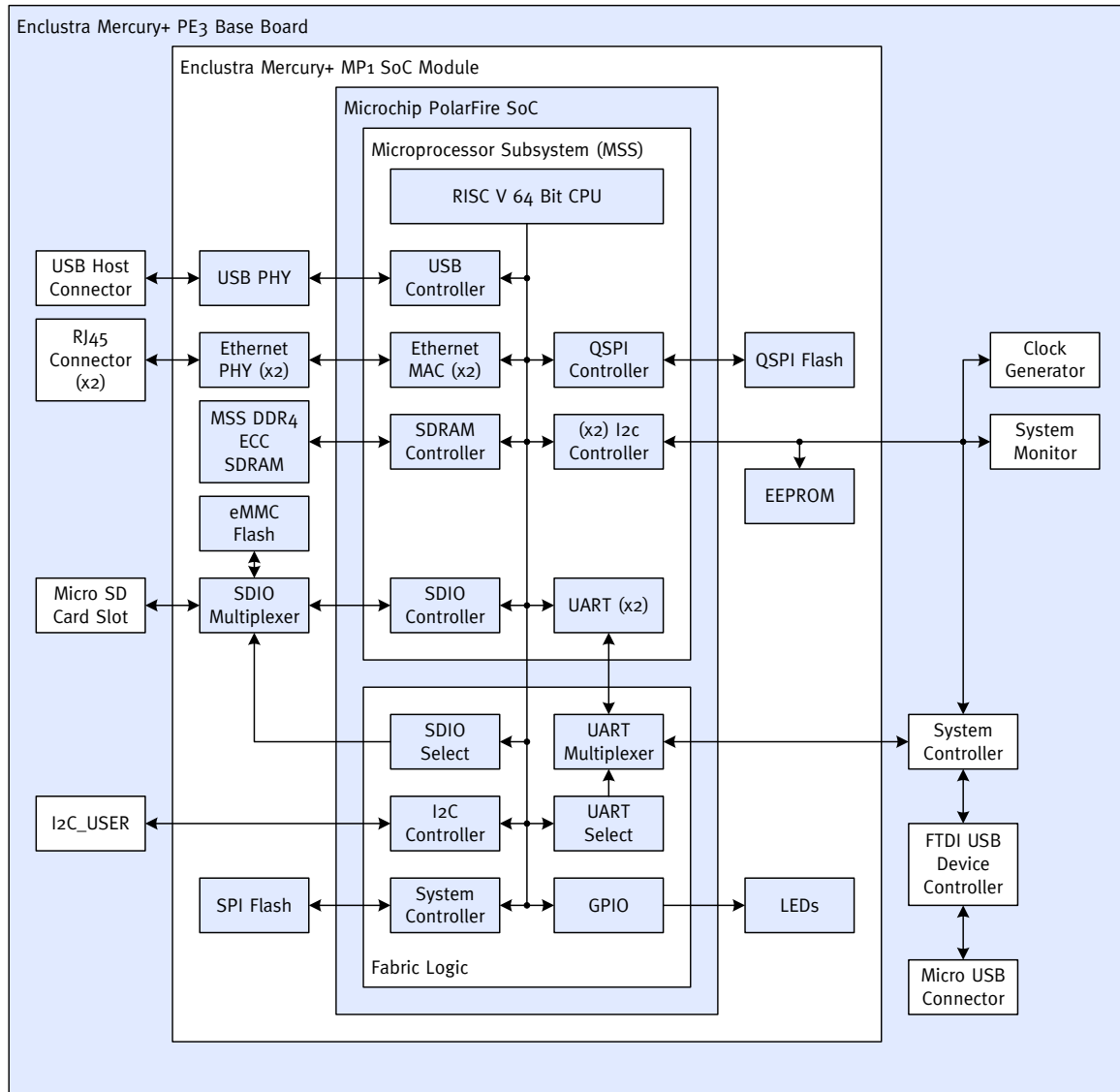
2 Reference Design Description

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 frequencies, as specified in the Microchip PolarFire® SoC Datasheet DS4248 [13].

2.1.2 MSS DDR4 SDRAM

The DDR4 SDRAM memory is configured as specified in the Mercury+ MP1 SoC Module User Manual [2]. ECC is enabled. The DDR clock frequency can be modified in the PolarFire SoC MSS Configurator([12]) under the **DDR Memory** section.

2.1.3 SD Card

The SD card is mapped to MSS I/O bank 4, pins MSSIO[0:6].

2.1.4 eMMC

The eMMC interface is mapped to MSS I/O bank 4, pins MSSIO[0:11].

2.1.5 MSS SD Card and eMMC Muxing

The SD card and eMMC interface share MSS IO pins. Software and FPGA fabric design support is required to allow switching between eMMC and SD card. The reference design includes the necessary logic in the FPGA fabric and the Enclustra HSS binary implements the software support.

2.1.6 I2C

The I2C_0 controller is mapped to FPGA fabric IO pins. The I2C_1 controller is mapped to I/O bank 2, pins MSSIO[26:27]. The available I2C devices on each bus are listed in the Mercury+ MP1 SoC Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3].

2.1.7 Quad SPI Flash Controller

The quad SPI flash controller is mapped to MSS I/O bank 2, pins MSSIO[30:35].

2.1.8 UART

MMUART_0 and MMUART_1 controller are mapped to FPGA fabric and are connected via multiplexer to I/O pins. These pins are connected to the FTDI USB device controller on the base board. The UART is configured as shown in Table 2.

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

Table 2: UART configuration

2.1.9 Ethernet

Both Ethernet controllers (MAC_0 and MAC_1) are mapped to MSS SGMII I/O bank 5. Each interface is connected to a Texas Instruments (TI) DP83867IS Ethernet PHY on the module. The two PHYs have a shared MDIO interface. The MDIO signals are mapped to MSS I/O bank 2. The PHYs can be configured via the MDIO interface on address 0 (MAC_0) and address 3 (MAC_1).

2.1.10 USB

The USB controller is mapped to I/O bank 2, pins MSSIO[14:25] and connected to a USB3320C USB 2.0 PHY. This interface can be configured for USB host, USB device and USB On-The-Go (OTG) operation. The settings in the system controller and the DIP switches on the base board must be configured according to the desired USB mode. Refer to the Mercury+ PE3 Base Board User Manual [3] for details.

2.1.11 GPIOs

Unused MSS IO pins are available as GPIOs. GPIO pins MSSIO[12:13] on bank 4 are available. In addition, pins MSSIO[22:23] on bank 2 are available as outputs to control user LEDs 0 and 1.

2.2 PolarFire FPGA Fabric

2.2.1 LEDs and GPIOs

The logic in the FPGA fabric 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.

Pin	Signal	Function
E23	FPGA_LED0#	Blinking LED counter MSB

Table 3: FPGA fabric blinking LED configuration

In addition, the block design instantiates a CoreGPIO IP block to control the LEDs described in Table 4.

Pin	Signal	Function
D23	FPGA_LED1#	GPIO 1, shared with MSS_LED1#
A25	FPGA_LED2#	GPIO 2
B25	FPGA_LED3#	GPIO 3

Table 4: FPGA fabric CoreGPIO configuration

A second CoreGPIO controller is connected to the MSS on the APB bus and allows controlling the UART_SEL signal. This signal is used to switch between the MMUART controllers enabled in the MSS.

2.2.2 Clocks and Resets

A SmartDesign block containing a CoreReset, ClockConditioningCircuitry and InitMonitor IP core is instantiated and provides the necessary reset, clock and startup signals.

2.2.3 SD Card and eMMC muxing

The necessary FPGA fabric logic to support SD card and eMMC multiplexing is implemented by a GPIO IP core. This IP core is connected to the MSS on the APB bus and allows controlling the SDIO_SEL signal.

3 Getting Started

This section describes the steps required to configure the Mercury+ MP1 SoC module and the Mercury+ PE3 base board in order to run a simple HelloWorld application example:

1. Install the module and configure the base board.
2. Generate the bitstream.
3. Prepare the software workspace.
4. Run a software application.

Read the Mercury+ MP1 SoC Module User Manual [2] and the Mercury+ PE3 Base Board User Manual [3] carefully before proceeding.

3.1 Essential Information

Pre-generated binaries may be used instead of building them manually as described in the following sections. The binaries for any supported MP1 product model and boot mode are released on the reference design Github release page [5].

> **Tip**
>
> Workarounds and fixes for potential issues can be found in Section 4.

NOTICE

Damage to the device due to overheating

Depending on the user application, the Mercury+ MP1 SoC module may consume more power than can be dissipated without additional cooling measures.

- Ensure that the SoC is always adequately cooled by installing a heat sink and/or providing air flow.

3.2 Setting up the Hardware

NOTICE

Damage to the device when mounting or removing the module

Mounting or removing the module while the base board is powered can lead to damage to the module or the base board.

- Ensure that the base board is not powered before mounting or removing the module.

The assembly drawing of the Mercury+ PE3 base board is shown in Figure 2. The relevant interfaces are marked in red in the figure as well as in the instructions below.
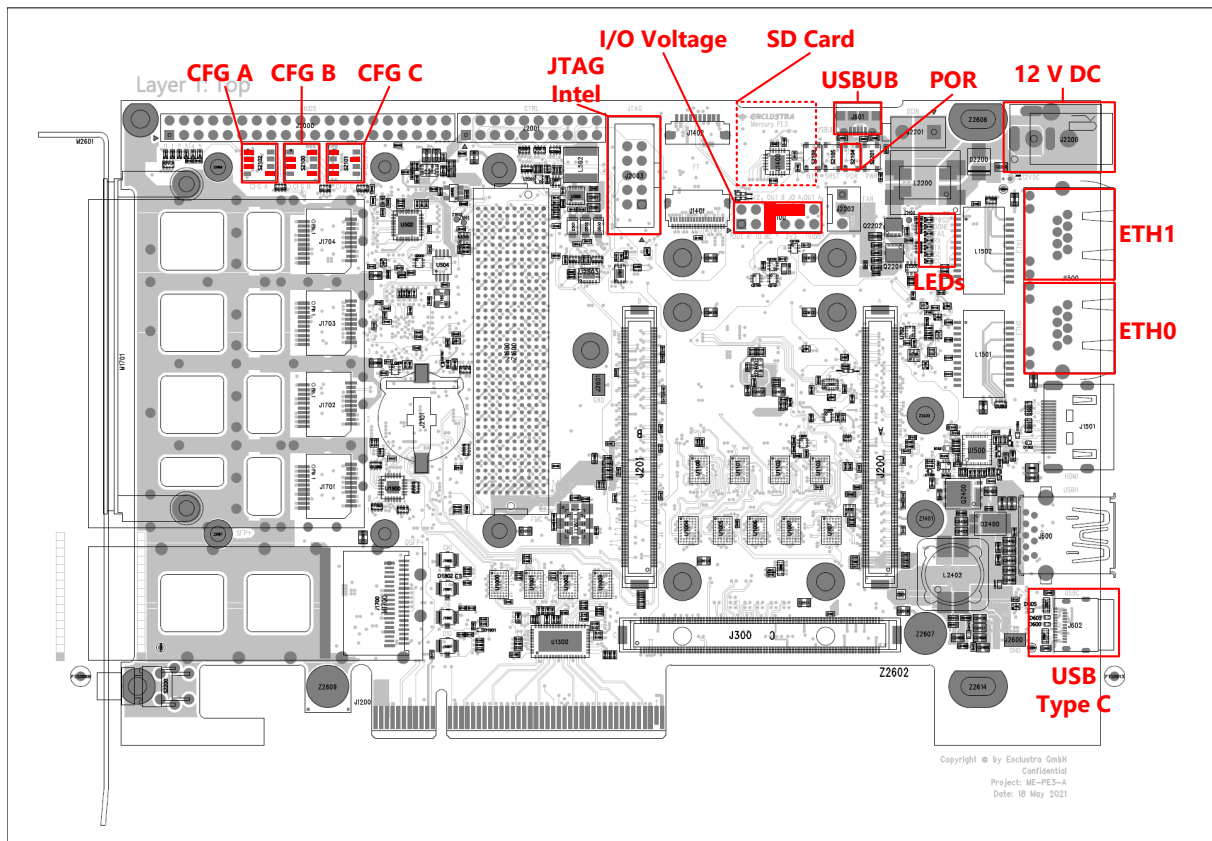
Figure 2: Mercury+ PE3 base board assembly drawing (top view)

1. Ensure that no power is connected to the power connector of the base board (label **12 V DC**).
2. Set the I/O voltage jumpers on the base board according to label **I/O Voltage** (the jumpers are marked in red):
   - VSEL A = 1.8 V (position 8-10)
   - VSEL BC = 1.8 V (position 5-6)
3. Set the configuration DIP switches on the base board as follows (labels **CFG A**, **CFG B** and **CFG C**):
   - CFG A = [1: OFF, 2: OFF, 3: OFF, 4: ON]
   - CFG B = [1: ON, 2: OFF, 3: OFF, 4: ON]
   - CFG C = [1: OFF, 2: ON, 3: OFF, 4: OFF]
4. Install the Mercury+ MP1 SoC module on the base board.

> **Tip**
>
> A small golden square on the bottom left corner of Enclustra modules is provided as landmark. The same landmark is provided on the Enclustra base boards to help orient the module in the right direction when connecting it.

5. Connect the Micro-USB cable between the computer and the base board using the Micro-USB port labeled **USBUB**. The SC LED starts blinking (label **LEDs**).
6. Two new serial ports should be detected by the OS.
7. Open the serial port with the highest number of the two detected ports and configure it using the parameters specified in Section 2.1.8. Any suitable utility for establishing a serial connection can be used. The output of the HelloWorld application will be printed in this terminal after running a software application, as presented in Section 3.6.
8. Connect the Microchip FlashPro 6 JTAG signals to the JTAG connector of the Mercury+ PE3 base board (label **JTAG**).

> **Tip**
>
> On Enclustra base boards, the pin assignment of the JTAG connector is different than the Microchip FlashPro Programmer. The appropriate connection is shown in Figure 3.

9. **NOTICE: Damage to the device when applying power. Ensure that the mounting holes on the base board are aligned with the mounting holes of the module before applying power.**
   Connect a 12 V DC power supply plug to the power connector of the base board (label **12 V DC**). The PWGD LED lights up (label **LEDs**).
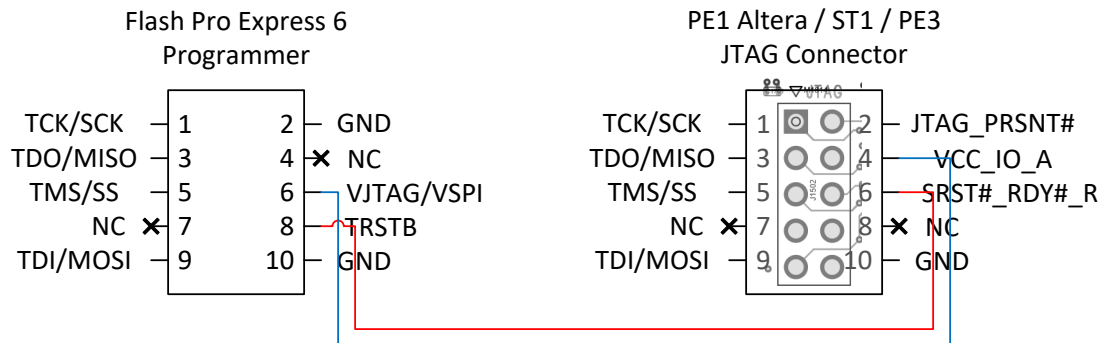

Figure 3: Mercury+ PE3 base board connection between FlashPro 6 JTAG connector and Mercury+ PE3 base board JTAG connector

3.3   Generating the FPGA Bitstream

Follow the steps described in this section to generate the reference design bitstream manually. Refer to the troubelshooting section 4.1 in case of problems.

> **Tip**
>
> Pre-generated bitstreams for all supported MP1 product models are available on the MP1 reference design Github release page [5]. These can be used directly instead of generating the bitstream manually. If using those files directly, the bitstream does not need to be generated manually and this section can be skipped.

1. Configure the `<base_dir>/scripts/settings.tcl` file:
   (a) Set the `module_name` variable to the desired product model[2].
   (b) Save the file after editing.
2. Start Microchip Libero SoC 2024.2.
3. Create the Mercury+ MP1 SoC module reference design project:
   (a) Click **Project** in the top left corner of the window.
   (b) Select **Execute Script...**.
   (c) Select the **create_reference_design.tcl** script that is part of the reference design.
   (d) Click **Run**.
   (e) Wait for completion (it will take some time until everything is set up)[3].
4. Derive the timing constraints:
   (a) Under **Design Flow**, open the **Manage Constraints** window.
   (b) Navigate to **Timing**.
   (c) Click **Derive Constraints** and wait for completion.

(d) When prompted whether to associate the derived constraints for **Synthesis**, **Place and Route** and **Timing Verification**, select **Yes**.

5. Generate the bitstream:

   (a) Run **Synthesize**[4].
   (b) Run **Place and Route**.
   (c) Click **Generate FPGA Array Data**.
   (d) (Optional) This step is only necessary if the MSS is needed, for example, for Linux deployment. Follow the steps described in [17] to add the HSS binary to the eNVM section.
   (e) Click **Generate Design Initialization Data**.
   (f) Click **Generate Bitstream**.

6. Export the FlashPro Express job for easier deployment:

   (a) Click **Export FlashPro Express Job**.
   (b) Leave all settings as default and click **OK**.
   (c) The generated FlashPro Express job can be used to create a FlashPro Express project with the FPExpress tool.

3.4 FlashPro Express Deployment

Follow the steps described in this section to create the FlashPro Express project using the previously generated FlashPro Express job file.

Tip

Alternatively, FlashPro Express job files are available on the reference design Github release page [5]. These can be used directly instead of generating the job files manually.

1. Create the FlashPro Express project:

   (a) Create an empty folder, for example, `<base_dir>/build/FlashProExpressProject`.
   (b) Open the FPExpress application.
   (c) Select **Project** > **New Job Project**.
   (d) The dialog shown in Figure 4 opens.
   (e) Browse to the FlashPro Express job file generated in Section 3.3 (typically `<base_dir>/build/Libero/<module_name>/designer/<reference_design>/export/*.job`) and click **Open**.
   (f) Set the project location to the previously generated empty folder and click **Select Folder**.
   (g) Click **OK**.
   (h) The generated project should look similar to Figure 5.

2. Program the bitstream:

   (a) Select **PROGRAM** from the available options.
   (b) Click **RUN** and wait for completion.

---

[2]Valid values for the variables are listed at the beginning of the file.

[3]If Libero SoC crashes during project creation and no report screen is shown after running the script, refer to the troubleshooting section 4.1.

[4]The warnings regarding unconnected top level ports can be ignored. The reason for these warnings is that not all top level ports are used in the reference design.
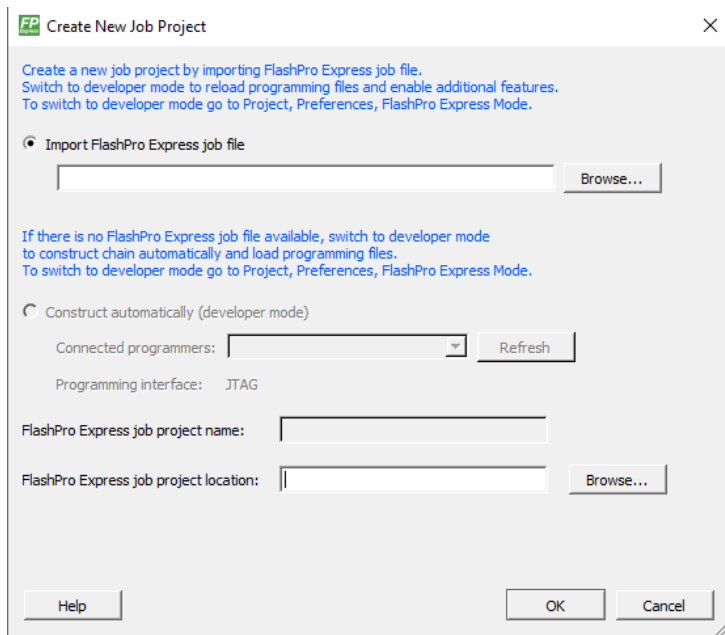

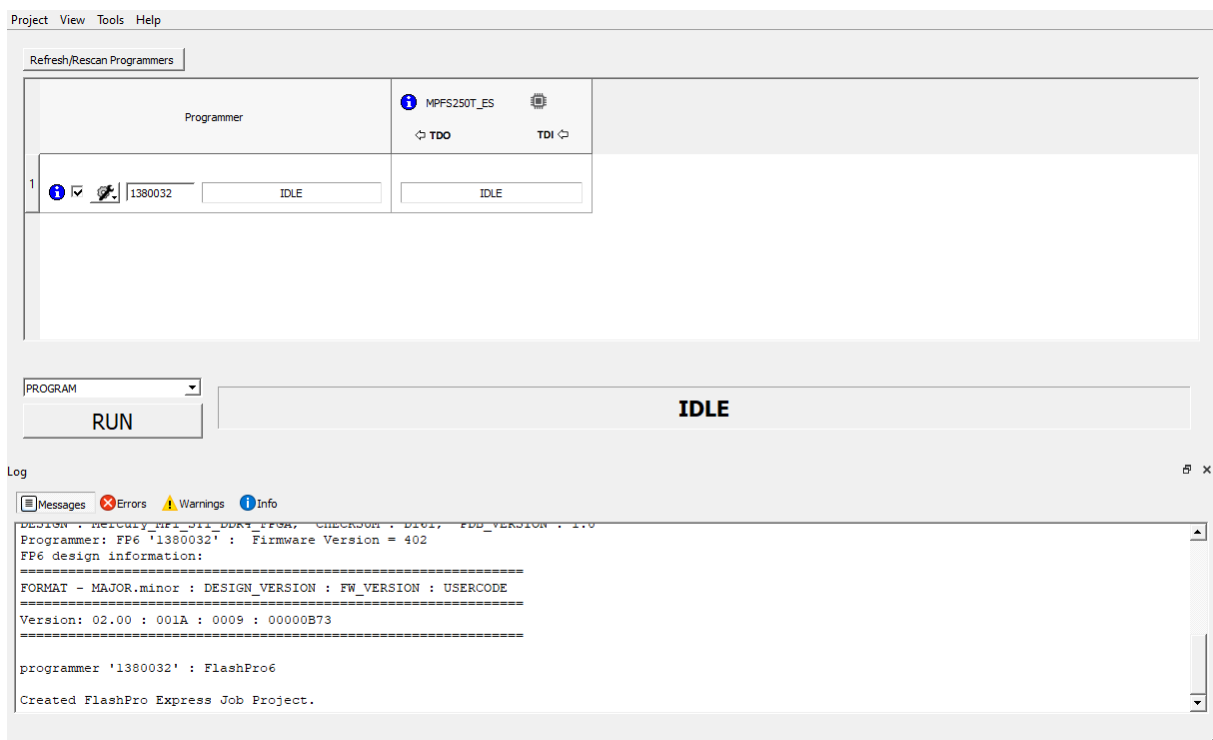*Figure 4: FlashPro Express create new job project dialog*

Figure 5: Generated FlashPro Express project

3.5 Preparing the SoftConsole Workspace

Follow these steps to create the SoftConsole workspace and import the example project. The HelloWorld example project is a modified version of the Microchip PolarFire Bare Metal MMUART example [15].

> **Tip**
>
> Pre-generated binaries for all supported MP1 product models are available on the reference design Github release page [5]. These can be used directly instead of generating the binaries manually. In this case, the following section can be skipped.

1. Create a workspace:

   (a) Create an empty folder, for example, `<base_dir>/build/SoftConsoleWorkspace`.
   (b) Open SoftConsole 2022.2, choose the created directory as the SoftConsole workspace and click **Launch**.
   (c) Close the welcome page.

2. Import the existing project into the workspace:

   (a) Select **File** > **Import...** > **Existing Projects into Workspace** > **Next**.
   (b) Select the prepared project for your product model as the root directory for the import (`<base_dir>/sw/Hello World`)
   (c) Click **Finish**.
   (d) The prepared workspace should look like Figure 6.

3. Build the application:

   (a) The default build configuration is **LIM-Debug**. An overview over the different build configurations is given in [15].
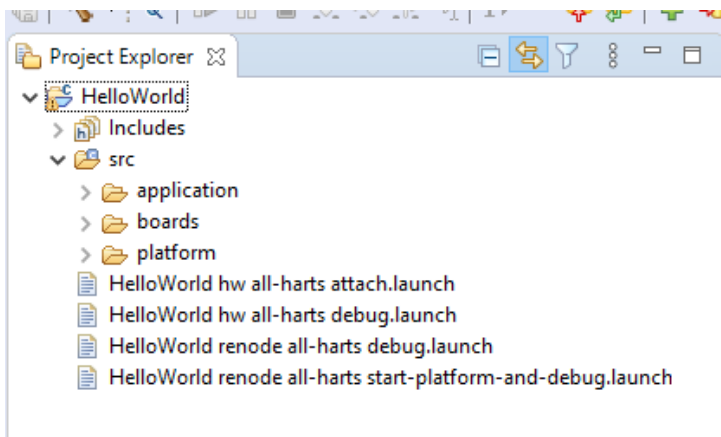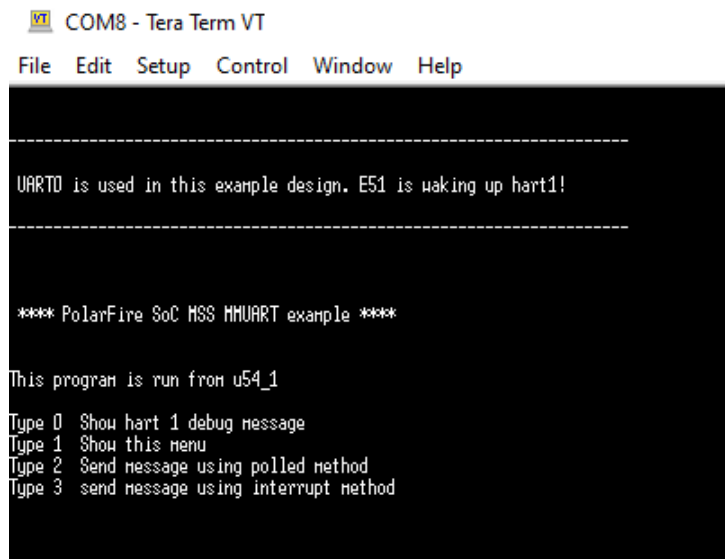   (b) Press **Ctrl+B** and wait for completion.

Figure 6: HelloWorld SoftConsole workspace

3.6   Running Software Applications

This section describes how to run software applications.

1. Right-click the HelloWorld project and select **Debug as** > **Debug Configurations...**.
2. Choose the **HelloWorld hw all-harts debug** template and click **Debug**.
3. If the debugger pauses during debugging, click **Resume**.
4. The output is shown in Figure 7.
5. The CLI offers various options to transfer messages over UART.

Figure 7: HelloWorld serial terminal output

4 Troubleshooting

4.1 Libero Issues

Libero SoC GUI Crashes.

- Execute the `create_reference_design.tcl` script in shell mode:
    1. Open a terminal in the directory containing the reference design.
    2. Execute following command:
       `<path-to-Libero>/libero script:create_reference_design.tcl .`
    3. Wait for the process to complete. Depending on the terminal, you may see output messages during the creation process.
    4. Open Libero SoC 2024.2 and click **Open Project**.
    5. Navigate to the generated reference design project file in `<base_dir>/build/Libero/<module_name>/`.

4.2 UART Connection Issues

The COM Ports Are Not Recognized.

- Ensure that the USB cable is properly connected to the Micro-USB connector on the Mercury+ PE3 base board
- Ensure that the FTDI VCP drivers are installed properly.
- (Windows OS only) Ensure that VCP is enabled:
    1. Open the Windows **Device Manager**.
    2. Under **Universal Serial Bus Controllers**, right-click **USB Serial Converter A** and select **Properties**.
    3. In the **Advanced** tab, activate the **Load VCP** checkbox.
    4. Disconnect and reconnect the Micro-USB cable.
    5. After a refresh of the **Device Manager**, two new COM ports should appear in the **Ports (COM & LPT)** section.

There Is No Output in the Serial Terminal.

- (Windows OS only) Ensure that no instance of the Enclustra MCT [4] is running. If MCT is open:
    1. Close MCT.
    2. Disconnect and reconnect the Micro-USB cable.
    3. Reopen the serial terminal.

There Are Unexpected Characters in the Serial Terminal Output.

Ensure that the serial terminal settings comply with Table 2.

List of Figures

List of Tables

References

[1]   Enclustra Support Channel
      support@enclustra.com
[2]   Mercury+ MP1 SoC Module User Manual
      https://www.enclustra.com/me-mp1-user-manual
[3]   Mercury+ PE3 Base Board User Manual
      https://www.enclustra.com/me-pe3-user-manual
[4]   Enclustra Module Configuration Tool (MCT)
      https://www.enclustra.com/module-configuration-tool
[5]   Mercury+ MP1 PE3 Reference Design Releases
      https://github.com/enclustra/Mercury_MP1_PE3_Reference_Design/releases
[6]   Enclustra Application Notes
      https://github.com/enclustra/I2CAppNote
      https://github.com/enclustra/GigabitEthernetAppNote
[7]   Mercury+ MP1 SoC Module Known Issues and Changes
      https://www.enclustra.com/me-mp1-known-issues-and-changes
[8]   Mercury+ PE3 Base Board Known Issues and Changes
      https://www.enclustra.com/me-pe3-known-issues-and-changes
[9]   Libero SoC Design Suite v2024.2 Design Flow User Guide
      https://coredocs.s3.amazonaws.com/Libero/2024_2/Tool/libero_ecf_ug.pdf
[10] SmartDesign v2024.2 User Guide
      https://coredocs.s3.amazonaws.com/Libero/2024_2/Tool/smartdesign_ug.pdf
[11] SoftConsole Documentation v2022.2
      https://mi-v-ecosystem.github.io/SoftConsole-Documentation/SoftConsole-v2022.
      2-RISC-V/
[12] PolarFire® SoC MSS Configurator User Guide v2024.2
      https://coredocs.s3.amazonaws.com/Libero/2024_2/pfsoc_mss_configurator_ug.pdf
[13] PolarFire® SoC Datasheet DS4248
      https://ww1.microchip.com/downloads/aemDocuments/documents/FPGA/ProductDocuments/
      DataSheets/PolarFire-SoC-Datasheet-DS00004248.pdf
[14] Libero SoC Software and License Installation Guide
      https://ww1.microchip.com/downloads/aemDocuments/documents/FPGA/swdocs/libero/
      Libero_Installation_Licensing_Setup_User_Guide_2024_2.pdf
[15] Github Microchip Bare Metal Examples
      https://github.com/polarfire-soc/polarfire-soc-bare-metal-examples
[16] Enclustra Github Hart Software Services
      https://github.com/enclustra/hart-software-services
[17] eNVM HSS binary deployment
      https://github.com/enclustra/hart-software-services#embed-hss-binary-into-bitstream-in-libero