PolarFire FPGA: Building a Mi-V Processor Subsystem
## Contents

1 Revision History .......................................................... 1
   1.1 Revision 9.0 ............................................................ 1
   1.2 Revision 8.0 ............................................................ 1
   1.3 Revision 7.0 ............................................................ 1
   1.4 Revision 6.0 ............................................................ 1
   1.5 Revision 5.0 ............................................................ 1
   1.6 Revision 4.0 ............................................................ 1
   1.7 Revision 3.0 ............................................................ 1
   1.8 Revision 2.0 ............................................................ 2
   1.9 Revision 1.0 ............................................................ 2

2 Building a Mi-V Processor Subsystem ................................. 3
   2.1 Requirements .......................................................... 3
   2.2 Prerequisites ........................................................... 4
   2.3 Design Description .................................................... 4
      2.3.1 Fabric RAMs Initialization ....................................... 5
   2.4 Creating a Mi-V Processor Subsystem ............................... 5
      2.4.1 Creating a Libero Project ....................................... 5
      2.4.2 Creating a New SmartDesign Component ......................... 6
      2.4.3 Instantiating IP Cores in SmartDesign ......................... 6
      2.4.4 Connecting IP Instances in SmartDesign ....................... 18
      2.4.5 Generating SmartDesign Component ............................ 22
      2.4.6 Managing Timing Constraints .................................. 22
      2.4.7 Running the Libero Design Flow ............................... 23

3 Building the User Application Using SoftConsole .................. 33
   3.1 Creating a Mi-V SoftConsole Project .............................. 33
   3.2 Downloading the Firmware Drivers ................................ 36
   3.3 Importing the Firmware Drivers ................................... 37
   3.4 Creating the main.c File ............................................ 39
   3.5 Mapping Firmware Drivers and the Linker Script ................ 40
   3.6 Mapping Memory and Peripheral Addresses ....................... 46
   3.7 Setting the UART Baud Rate ....................................... 48
   3.8 Building the Mi-V Project ......................................... 49
   3.9 Debugging the User Application Using SoftConsole ............. 50
   3.10 Debugging the User Application from DDR3 Memory ............. 56

4 Appendix 1: Programming the Device Using FlashPro Express .... 57

5 Appendix 2 - References .................................................. 60

6 Appendix 3 - DDR3 Configuration ....................................... 61
Figures

Figure 1  Download New Cores Option .................................................. 4
Figure 2  Block Diagram ........................................................................ 4
Figure 3  New Project Details ................................................................. 5
Figure 4  Device Selection ..................................................................... 6
Figure 5  Create New SmartDesign ......................................................... 6
Figure 6  Mi-V Configuration ................................................................. 7
Figure 7  TCM and DDR3 Memory Map .................................................. 8
Figure 8  CoreAXI4Interconnect Configurator – Bus Configuration Section ................................................................. 9
Figure 9  CoreAXI4Interconnect - Master0 Configuration ......................... 9
Figure 10 CoreAXI4Interconnect Configurator – Slave0 Configuration .... 10
Figure 11 Crossbar Configuration and Enabling Master Write Access Settings ................................................................. 10
Figure 12 Apply Option for MPF300T ....................................................... 11
Figure 13 DDR3 General Configuration .................................................. 11
Figure 14 DDR3 Controller Configuration ............................................... 12
Figure 15 CoreAPB3 Configuration ......................................................... 13
Figure 16 CoreGPIO Configuration ......................................................... 14
Figure 17 CoreSPI Configuration ............................................................ 15
Figure 18 CCC Configurator Clock Options PLL Tab ............................... 16
Figure 19 CCC Configurator Output Clocks Tab ...................................... 16
Figure 20 INIT_MONITOR Configuration ................................................ 17
Figure 21 Top SmartDesign with All Components Instantiated ................. 18
Figure 22 Connection Method ............................................................... 18
Figure 23 Mi-V Subsystem Connected ................................................... 19
Figure 24 Edit Slices Window ............................................................... 21
Figure 25 Generate Component Icon ...................................................... 22
Figure 26 Derive Constraints Button ..................................................... 22
Figure 27 Derived Constraints .............................................................. 22
Figure 28 I/O Attributes ..................................................................... 24
Figure 29 Edit with I/O Editor Option .................................................... 24
Figure 30 Port View ............................................................................. 24
Figure 31 I/O Editor Design View – DDR3 Selection ................................. 24
Figure 32 Memory View [active] Tab with DDR3 Subsystem Placement ........ 24
Figure 33 DDR3_0 Placed .................................................................... 25
Figure 34 Fabric RAMs Tab .................................................................. 26
Figure 35 Edit Fabric RAM Initialization Client Dialog Box ....................... 26
Figure 36 Import Memory File Dialog Box .............................................. 27
Figure 37 Fabric RAMs Tab - Apply Button ............................................ 27
Figure 38 Design Initialization Data ....................................................... 28
Figure 39 SPI Flash Tab ....................................................................... 28
Figure 40 COM Port Number .................................................................. 30
Figure 41 Connection Type Selection ..................................................... 30
Figure 42 PutTY Configuration ............................................................. 31
Figure 43 Hello World String ............................................................... 32
Figure 44 Hello World String After the Board is Power Cycled ............... 32
Figure 45 Workspace Launcher ............................................................. 33
Figure 46 New C Project Creation ......................................................... 33
Figure 47 C Project Dialog Box ............................................................. 34
Figure 48 Select Configurations Dialog Box .......................................... 34
Figure 49 GNU RISC-V Cross Toolchain .............................................. 35
Figure 50 Empty Mi-V Project ............................................................... 35
Figure 51 Firmware Catalog Window .................................................... 36
Figure 52 RISCV HAL Files Report ...................................................... 36
Figure 53 CoreUARTapb Files Report .................................................... 36
Figure 54 CoreGPIO Files Report ......................................................... 37
Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Tutorial Requirements</td>
<td>3</td>
</tr>
<tr>
<td>Table 2</td>
<td>CCC_0_0 Pin Connections</td>
<td>20</td>
</tr>
<tr>
<td>Table 3</td>
<td>DEBUG_TARGET Pin Connections</td>
<td>20</td>
</tr>
<tr>
<td>Table 4</td>
<td>AXI4_Interconnect_0 Pin Connections</td>
<td>21</td>
</tr>
<tr>
<td>Table 5</td>
<td>APB3_0 Pin Connections</td>
<td>21</td>
</tr>
<tr>
<td>Table 6</td>
<td>Jumper Settings</td>
<td>29</td>
</tr>
</tbody>
</table>
1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 9.0
The following is a summary of the changes made in this revision.

- LSRAM was renamed to TCM throughout the document.
- Added Figure 6, page 7.
- Updated the reason for using the CoreAXI4Interconnect IP in Instantiating AXI Interconnect Bus IP, page 8.
- Updated the start and end addresses of AXI Slave0 port in Instantiating AXI Interconnect Bus IP, page 8.
- Updated Figure 10, page 10.
- Updated the SYS_CLK_FREQ macro definition from 111110000UL to 83333000UL, see Mapping Memory and Peripheral Addresses, page 46.

1.2 Revision 8.0
The following is a summary of the changes made in this revision.

- Updated for Libero SoC v2021.1.
- Updated Table 1, page 3.
- Added Appendix 3 - DDR3 Configuration, page 61.

1.3 Revision 7.0
The following is a summary of the changes made in this revision.

- Updated Figure 2, page 4.
- Replaced Figure 6, page 7, and Figure 23, page 19.
- Removed sections Instantiating On-chip SRAM, page 10, Instantiating the AXI3 to AHB-Lite Bridge, page 14, Instantiating the AHB-Lite Bus, page 14, and Instantiating the AHB-Lite to APB3 Bridge, page 14.
- Updated section Connecting IP Instances in SmartDesign, page 18.

1.4 Revision 6.0
Updated for Libero SoC v12.5.

1.5 Revision 5.0
The following is a summary of the changes made in this revision.

- Updated for Libero SoC v12.2.
- Updated the design for AXI-based Mi-V Soft Processor for an enhanced performance with DDR memories.
- Removed Libero SoC and SoftConsole version numbers.

1.6 Revision 4.0
The following is a summary of the changes made in this revision.

- Added Fabric RAMs Initialization, page 5.
- The document was updated for Libero SoC v12.0.

1.7 Revision 3.0
The following is a summary of the changes made in this revision.
Revision History

• Added Design Description, page 4.
• The document was updated for Libero SoC PolarFire v2.1.

1.8 Revision 2.0

The following is a summary of the changes made in this revision.

• The document was updated for the Mi-V processor upgrade.
• The document was updated for Libero SoC PolarFire v2.0 and SoftConsole v5.2. For more information, see Building the User Application Using SoftConsole, page 33.
• Information about TCM initialization from external SPI flash was added. For more information, see Configure Design Initialization Data and Memories, page 25.

1.9 Revision 1.0

The first publication of this document.
Microchip offers the Mi-V processor IP and software toolchain free of cost to develop RISC-V processor-based designs. RISC-V, a standard open instruction set architecture (ISA) under the governance of the RISC-V foundation, offers numerous benefits, which include enabling the open source community to test and improve cores at a faster pace than closed ISAs.

PolarFire® FPGAs support Mi-V soft processors to run user applications. The objective of the tutorial is to build a Mi-V processor subsystem that can execute an application from the designated fabric RAMs initialized from the sNVM/SPI Flash. The tutorial also describes how to build a RISC-V application using SoftConsole and run it on a PolarFire Evaluation Board.

2.1 Requirements

The following table lists the tutorial requirements for building a Mi-V processor subsystem.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td></td>
</tr>
<tr>
<td>Host PC</td>
<td>Windows 7, 8.1, or 10</td>
</tr>
<tr>
<td>POLARFIRE-EVAL-KIT (MPF300TS-FCG1152I)</td>
<td>Rev D or Rev E¹</td>
</tr>
<tr>
<td>– 12 V/5 A AC power adapter and cord</td>
<td></td>
</tr>
<tr>
<td>– USB 2.0 A to mini-B cable</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>Libero SoC Design Suite</td>
<td>See the readme.txt file provided in the design files for all software versions needed to create this reference design.</td>
</tr>
<tr>
<td>Firmware Catalog²</td>
<td></td>
</tr>
<tr>
<td>SoftConsole</td>
<td>See the readme.txt file provided in the design files for all software versions needed to create this reference design.</td>
</tr>
<tr>
<td>PuTTTY (serial terminal emulation program)</td>
<td></td>
</tr>
</tbody>
</table>

1. Rev E Kit has a different on-board DDR part. For more information, refer to *PolarFire Evaluation Kit Quick Start Guide*.
2. Firmware catalog is included in the installation package of Libero SoC.
2.2 Prerequisites

1. Download the design files from:
   http://soc.microsemi.com/download/rsc/?f=mpf_tu0775_df
   The design files folder contains the following folders:
   - **Programming Job**: Two programming files (.job) one each for Rev D (top_RevD.job) and
     Rev E (top_RevE.job) Kit are provided.
   - **Solution**: Contains the final Libero and SoftConsole projects for reference
   - **Source**: Contains the source files required to complete this tutorial

2. Download and install Libero SoC from:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads

3. Download and install SoftConsole from:
   https://www.microsemi.com/products/fpga-soc/design-resources/design-software/softconsole#downloads

4. From the Libero Catalog, download the latest versions of the IP cores from the warning pop-up as shown in the following figure.

   **Figure 1** • Download New Cores Option

2.3 Design Description

The tutorial describes how to create a Mi-V subsystem for executing user applications. The user application can be stored in µPROM, sNVM, or an external SPI flash. At device power-up, the PolarFire System Controller initializes the designated TCM with the user application and releases the system reset. If the user application is stored in SPI Flash, the System Controller uses the SC_SPI interface for reading the user application from SPI Flash. The given user application prints the UART message “Hello World!” and blinks user LEDs on the board.

The following figure shows the top-level block diagram of the design.

   **Figure 2** • Block Diagram
2.3.1  **Fabric RAMs Initialization**

Each logical RAM instance in the design can be initialized from a different source—sNVM, µPROM, or SPI-Flash. The initialization client storage location is configurable. Generate the initialization data to add the initialization clients to the chosen non-volatile memories and program the device. Program SPI-Flash, if chosen as storage location for initialization data. For more information, see Configure Design Initialization Data and Memories.

*Note:*  Libero SmartDesign and configuration screen shots shown in this tutorial are for illustration purpose only. Open the Libero project to see the latest updates and IP versions.

2.4  **Creating a Mi-V Processor Subsystem**

Creating a Mi-V processor subsystem involves:

- Creating a Libero Project
- Creating a New SmartDesign Component
- Instantiating IP Cores in SmartDesign
- Connecting IP Instances in SmartDesign
- Generating SmartDesign Component
- Managing Timing Constraints
- Running the Libero Design Flow

This section describes all the steps required to create a Mi-V processor subsystem on a new SmartDesign canvas.

2.4.1  **Creating a Libero Project**

Follow these steps to create a Libero project:

1. On the Libero Menu bar, click **Project > New Project**.
2. Enter the following details, and click **Next**.
   - Project name: PF_Mi_V_Tut
   - Project location: For example, F:/Libero_Projects
   - Preferred HDL type: Verilog

*Figure 3* • **New Project Details**

![New Project Details](image)
3. To choose the PolarFire device present on the PolarFire Evaluation Board, select the following settings in the Device Selection window, and click Next.
   • Family: PolarFire
   • Die: MPF300TS
   • Package: FCG1152
   • Speed: -1
   • Range: IND
   • Part Number: MPF300TS-1FCG1152I

4. In the Device Settings window, click Next to retain the default core voltage and I/O settings.
5. In the Add HDL Sources window, click Next to retain the default settings.
6. In the Add constraints window, click Import file to import the I/O constraint file.
7. In the Import files window, locate the io_constraints.pdc file in the DesignFiles_directory\Source\io folder, and double-click it.
8. Click Finish.
   The Log pane displays a message indicating that the PF_Mi_V_Tut project was created.

### 2.4.2 Creating a New SmartDesign Component

To create a new SmartDesign component:

1. In Libero, select File > New > SmartDesign.
2. In the Create New SmartDesign dialog box, enter top as the name of the new SmartDesign project, as shown in the following figure.

3. Click OK.
   The top SmartDesign component is created.

### 2.4.3 Instantiating IP Cores in SmartDesign

When an IP core is dragged from the Catalog to SmartDesign, Libero prompts you to name the component, and if applicable, to configure the IP core. After the core is configured, Libero generates the component for that core and instantiates it in SmartDesign.
2.4.3.1 Instantiating Mi-V Processor IP

1. From the Catalog, drag the MIV_RV32 to SmartDesign.
2. In the Create Component dialog box, enter MiV_RV32_C0 as the component name, and click OK.
3. In the Configurator, set the following configuration:
   • Set Reset Vector Address -> Upper 16 bits (Hex) to 0x8000 and retain the default setting for Lower 16 Bits (Hex) as shown in Figure 6, page 7. This is the address the processor will start executing from after a reset.

Figure 7, page 8 shows the memory map of TCM and DDR3 memory.

Figure 6 • Mi-V Configuration
2.4.3.2 Instantiating AXI Interconnect Bus IP

The AXI interconnect bus must be configured to connect the Mi-V core with memory. Also, the
AXI4Interconnect is needed for converting the Mi-V processor’s AXI4 32-bit data to the DDR3 AXI4 64-
bit data, and also for bridging the Mi-V processor’s AXI4 clock rate of 83.3 MHz to the DDR3 AXI4 clock
rate of 166.66 MHz.

1. From the Catalog, drag the CoreAXI4Interconnect IP core to SmartDesign.
2. In the Create Component dialog box, enter AXI4_Interconnect as the component name, and click
   OK.
   The Configurator opens.
3. In the Bus Configuration section, configure the AXI4_Interconnect IP to have one slave with an ID
   width of 1, as shown in the following figure. Leave the rest as defaults.
4. In the **Master Configuration** section, retain the following Master0 default settings:
   - **M0 Type**: AXI4
   - **M0 Data Width**: 32 bits
   - **M0 DWC Data FIFO Depth**: 16
   - **M0 Register Slice**: Selected
   
   The following figure shows the Master0 configuration.

5. In the **Slave Configuration** section, configure the Slave0 port as follows:
   - **S0 SLAVE Start Address (Lower 32 bits)**: 0x80010000
   - **S0 SLAVE End Address (Lower 32 bits)**: 0x8FFFFFFF
   - **S0 Clock Domain Crossing**: Enabled
   - **Leave the rest as defaults**
6. In the Crossbar Configuration section, ensure that the following options are set:
   - Under Enable Master Write Access, enable M0 access S0.
   - Under Enable Master Read Access, enable M0 access S0.
   - Leave the rest as defaults.

2.4.3.3 Instantiating DDR3 Memory Controller

This tutorial demonstrates how to build and debug an application from DDR3 memory. Executing an application from DDR3 memory in the release mode requires a bootloader. The bootloader use case is not in the scope of this tutorial.

If you are using the Rev D Kit, configure DDR IP as shown below. (If you are using Rev E Kit, see Appendix 3 - DDR3 Configuration.)

1. From the Catalog, drag the PolarFire DDR3 IP core to SmartDesign.
2. In the Create Component dialog box, enter DDR3_0 as the component name, and click OK.
3. In the left pane of the Configurator, expand Microsemi PolarFire Evaluation Kits > PolarFire Evaluation Kit > MPF300T.
4. Left-click MT41K1G8SN-125, and click Apply, as shown in the following figure. This configures the DDR3 controller with the initialization and timing parameters of the DDR3 memory (MT41K1G8SN-125) present on the PolarFire Evaluation Kit.
5. On the **General** tab, set the **CCC PLL Clock Multiplier** to 8, and the **DQ Width** to 16, as shown in **Figure 13**, page 11. The clock multiplier value of 8 sets the CCC PLL reference clock frequency to 83.333 MHz. A reference clock of this frequency is required for the PLL present inside the DDR3 subsystem. The PLL generates a 666.666 MHz DDR3 memory clock frequency and a 166.666 MHz DDR3 AXI clock frequency. The DQ width is set to 16 to match the width of the DDR3 memory present on the board.

**Figure 13** • DDR3 General Configuration
6. On the **Controller** tab, ensure that the settings are as follows:
   - Instance Number: 0
   - Fabric Interface: AXI4
   - AXI ID Width: 4
   - Enable Rank0 - ODT0 check box: Selected

   ![Figure 14 • DDR3 Controller Configuration](image)

7. Retain the default settings for others tabs and click **OK**.

### 2.4.3.4 Instantiating APB3 Bus

1. From the Catalog, drag the **CoreAPB3** IP core to SmartDesign.
2. In the **Create Component** dialog box, enter **APB3** as the component name, and click **OK**.
3. In the CoreAPB3 Configurator, select the following data width and address configuration settings, as shown in the following figure:
   - **APB Master Data Bus Width**: 32-bit
   - **Number of address bits driven by master**: 16
   - **Position in slave address of upper 4 bits of master address**: [27:24] (Ignored if master address width >= 32 bits)
   - **Enabled ABP Slave Slots**: **Slot 0**, **Slot 1**, and **Slot 2**.

   This configuration sets the slave address map as follows:
   - **Slot0**: 0x0000 - 0x0FFF
   - **Slot1**: 0x1000 - 0x1FFF
   - **Slot2**: 0x2000 - 0x2FFF
2.4.3.5 Instantiating UART Controller
1. From the Catalog, drag the CoreUARTapb IP core to SmartDesign.
2. In the Create Component dialog box, enter UART_apb as the component name, and click OK.
3. In the CoreUARTapb Configurator, retain the default configuration, and click OK.

2.4.3.6 Instantiating the GPIO Controller
1. From the Catalog, drag the CoreGPIO IP core to SmartDesign.
2. In the Create Component dialog box, enter CoreGPIO_0 as the component name, and click OK.
3. In the CoreGPIO Configurator, select the following Global Configuration settings, as shown in the following figure:
   • APB Data Width: 32
   • Number of I/Os: 4
   • Single-bit interrupt port: Disabled
   • Output enable: Internal
4. Under I/O bit 0, I/O bit 1, I/O bit 2, and I/O bit 3, do the following, as shown in the following figure:
   • Select Fixed Config.
   • Set the I/O type as Output.
   • Select the interrupt type as Disabled.
Four GPIO outputs are configured.
Figure 16 • CoreGPIO Configuration

5. Click OK to close the CoreGPIO Configurator.

2.4.3.7 Instantiating CoreSPI

The PolarFire Evaluation board contains two SPI Flash memories. One SPI Flash is connected to the System Controller SPI interface (SC_SPI) for design initialization. The CoreSPI IP is used to interface with the other SPI Flash, which is connected to the fabric I/Os. To instantiate CoreSPI:

1. From the Catalog, drag the CoreSPI IP core to SmartDesign.
2. In the Create Component dialog box, enter SPI_Controller as the component name, and click OK.
3. In the CoreSPI Configurator, do the following:
   • Set the APB Data Width to 32
   • In the SPI Configuration section, set the mode to Motorola, frame size to 8, FIFO depth to 32, and clock rate to 16.
   • In the Motorola Configuration section, set the mode to Mode 0, and select the Keep SSEL active check box.
2.4.3.8 Instantiating PolarFire Clock Conditioning Circuitry (CCC)

The PolarFire Clock Conditioning Circuitry (CCC) block generates a 83.333 MHz clock to the processor subsystem, which is used as a reference clock to the DDR3_0_0 PLL. To instantiate the CCC block:

1. From the Catalog, drag the Clock Conditioning Circuitry (CCC) core to SmartDesign.
2. In the Create Component dialog box, enter CCC_0 as the component name, and click OK.
3. In the Configurator, set the configuration to PLL-Single.
4. In the Clock Options PLL tab, do the following:
   - Set the input frequency to 50 MHz.
   - Set the feedback mode to Post-VCO.
   - Set the Bandwidth to High.
5. In the **Output Clocks** tab, under the **Output Clock 0** section, do the following:
   - Select the **Enabled** check box to enable PLL output 0.
   - Set the requested frequency to 83.333 MHz.
   - Select the **Global Clock** check box.

6. Click **OK** and acknowledge the pop-up.
2.4.3.9 Instantiating PolarFire Initialization Monitor

The PolarFire Initialization Monitor is used to get the status of device initialization including the TCM initialization. To instantiate the PolarFire Initialization Monitor:

1. From the Catalog, drag the PolarFire Initialization Monitor core to SmartDesign.
2. In the Create Component dialog box, enter INIT_Monitor as the component name, and click OK.
3. In the INIT_MONITOR Configurator > Bank Monitor tab, clear all the check boxes under Calibration Monitor except for BANK1_CALIB_STATUS, and click OK.
4. In the INIT_MONITOR Configurator > VDDI Monitor tab, clear all the check boxes under VDDI Monitor except for BANK_6_VDDI_STATUS, and click OK.

Figure 20 • INIT_MONITOR Configuration

2.4.3.10 Instantiating CORERESET_PF

Two instances of the CORERESET_PF IP are required in this design.

1. From the Catalog, drag the CORERESET_PF IP.
2. In the Component Name dialog box, enter reset_syn_0 as the name of this component, and click OK.
3. Retain the default configuration for this IP and click OK.
4. Similarly, instantiate another instance with reset_syn_1 as its name.

2.4.3.11 Instantiating CoreJTAGDebug

The CoreJTAGDebug IP connects the Mi-V soft processor to the JTAG header for debugging. To instantiate CoreJTAGDebug:

1. From the Catalog, drag the CoreJTAGDebug IP core to SmartDesign.
2. In the Create Component window, enter COREJTAGDebug_0 as the component name, and click OK.
3. In the Configurator, retain the default configuration, and click OK.

The following figure shows the top in SmartDesign after all the components are instantiated.
2.4.4 Connecting IP Instances in SmartDesign

Connect the IP blocks in SmartDesign using any of the following methods:

- **Using the Connection Mode icon**: You can initiate the connection mode in SmartDesign by clicking the **Connection Mode** icon in the SmartDesign toolbar, as shown in the following figure. The cursor changes from a normal arrow to the shape of the connection mode icon. To make a connection in this mode, click the first pin and drag it to the second pin that you want to connect.

  ![Connection Mode Icon](image1)

- **Using the Connect option in the Context menu**: You can also connect pins by selecting the pins, and then selecting **Connect** from the context menu. To connect multiple pins, hold down the Ctrl key while selecting the pins. Right-click the input source signal, and select **Connect**. To disconnect signals, right-click the input source signal, and select **Disconnect**.

- **Right-clicking on a pin provides a list of options like Mark Unused, Edit Slice, Tie Low, Promote to Top-Level, and Tie High. Use these options for individual pins settings.**

  ![Connection Method](image2)

*Figure 21* • Top SmartDesign with All Components Instantiated

*Figure 22* • Connection Method

*Figure 23*, page 19 shows the Mi-V subsystem in SmartDesign with all IP blocks connected and top-level I/Os.
Figure 23 • Mi-V Subsystem Connected

Note: Grayed out pins are marked unused, green pins are tied Low, and red pins are tied High. Ensure that unused, tied-low, and tied-high pins are strictly set as per Figure 23, page 19.

Follow these steps to connect the IP blocks as per Figure 23, page 19:

1. Set the pins as follows on INIT_MONITOR_0:
   - Select PCIE_INIT_DONE, USRAM_INIT_DONE, SRAM_INIT_DONE, XCVR_INIT_DONE, USRAM_INIT_FROM_SNVM_DONE, USRAM_INIT_FROM_UPROM_DONE, USRAM_INIT_FROM_SPI_DONE, SRAM_INIT_FROM_SNVM_DONE, SRAM_INIT_FROM_UPROM_DONE, SRAM_INIT_FROM_SPI_DONE, and AUTOCALIB_DONE pins.
   - Right-click the pins, and select Mark Unused.
   - Connect the FABRIC_POR_N pin to FPGA_POR_N pin of reset_syn_0 and reset_syn_1.
   - Connect the DEVICE_INIT_DONE pin to reset_syn_0:INIT_DONE.
   - Connect the BANK_1_CALIB_STATUS pin to reset_syn_1:INIT_DONE.
   - Connect the BANK_6_VDDI_STATUS pin to reset_syn_0:BANK_x_VDDI_STATUS, reset_syn_0:BANK_y_VDDI_STATUS, reset_syn_1:BANK_x_VDDI_STATUS, and reset_syn_1:BANK_y_VDDI_STATUS.

2. Set the pins as follows on CCC_0_0:
   - Right-click the REF_CLK_0 pin, and select Promote to Top Level.
   - Connect the other pins as specified in the following table:
3. Set the pins of reset_syn_0 as follows:
   • Connect EXT_RST_N pin to DDR3_0_0:CTRLR_READY.
   • Right-click SS_BUSY and FF_US_RESTORE pins and tie them low.
4. Connect the reset_syn_0:FABRIC_RESET_N to the following pins:
   • MIV_RV32_C0 : RESETN
   • AXI4_Interconnect_0:ARESETN
   • UART_apb_0:PRESETN
   • COREGPIO_0:PRESETN
   • SPI_Controller_0:PCLK
   • COREGPIO_0:PCLK
   • AXI4_Interconnect_0:ACLK

   **Note:** As DDR3_0_0:CTRL READY pin is connected to reset_syn_0:EXT_RST_N, the Mi-V processor is held in reset until the DDR3 controller is ready. The rest of the system is out of reset as soon as device initialization is done.

5. Set the pins of reset_syn_1 as follows:
   • Right-click SS_BUSY and FF_US_RESTORE pins and tie them low using the Tie Low option.
   • Select the EXT_RST_N pin and promote it to top level and rename it to resetn.
   • Connect the FABRIC_RESET_N pin to DDR3_0_0:SYS_RESET_N.
   • Right-click the PLL_POWERDOWN_B pin and mark it unused.
6. Set the pins as follows on COREJTAGDebug_0_0:
   • Expand JTAG HEADER.
   • Right-click the TDI, TCK, TMS, and TRSTB pins, and select Promote to Top Level.
   • Expand JTAG HEADER.
   • Right-click the TDO pin, and select Promote to Top Level.
   • Connect the other pins as specified in the following table.

   **Table 2 • CCC_0_0 Pin Connections**

<table>
<thead>
<tr>
<th>Connect From</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL_LOCK_0</td>
<td>reset_syn_0:PLL_LOCK and</td>
</tr>
<tr>
<td></td>
<td>reset_syn_1:PLL_LOCK</td>
</tr>
<tr>
<td></td>
<td>reset_syn_0:CLK and</td>
</tr>
<tr>
<td></td>
<td>reset_syn_1:CLK</td>
</tr>
<tr>
<td>OUT0_FABCLK_0</td>
<td>MIV_RV32:CLK</td>
</tr>
<tr>
<td></td>
<td>DDR3_0_0:PLL_REF_CLK</td>
</tr>
<tr>
<td></td>
<td>SPI_Controller_0:PCLK</td>
</tr>
<tr>
<td></td>
<td>UART_apb_0:PCLK</td>
</tr>
<tr>
<td></td>
<td>COREGPIO_0:PCLK</td>
</tr>
<tr>
<td></td>
<td>AXI4_Interconnect_0:ACLK</td>
</tr>
<tr>
<td>PLLPOWERDOWN_N_0</td>
<td>reset_syn_0:PLL_POWERDOWN_B</td>
</tr>
</tbody>
</table>

   **Table 3 • DEBUG_TARGET Pin Connections**

<table>
<thead>
<tr>
<th>Connect From</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TCK_0</td>
<td>MIV_RV32_C0:JTAG_TCK</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TRSTB_0</td>
<td>MIV_RV32_C0:JTAG_TRSTN</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TMS_0</td>
<td>MIV_RV32_C0:JTAG_TMS</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TDI_0</td>
<td>MIV_RV32_C0:JTAG_TDI</td>
</tr>
<tr>
<td>CoreJTAGDebug_0_0:TGT_TDO_0</td>
<td>MIV_RV32_C0:JTAG_TDO</td>
</tr>
</tbody>
</table>

7. Set the pins as follows on MIV_RV32_C0:
• Right-click the JTAG_TDO_DR pin, and select **Mark Unused**.
• Right-click the EXT_RESETN pin, and select **Mark Unused**.
• Connect APB_MSTR to APB3_0:APB3mmaster.
• Connect AXI4_MSTR to AXI4_Interconnect_0:AXI4mmaster0.

8. Connect the AXI4_Interconnect_0 pins as specified in the following table.

**Table 4**: **AXI4_Interconnect_0 Pin Connections**

<table>
<thead>
<tr>
<th>AXI4_Interconnect_0 Pin Name</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_CLK0</td>
<td>DDR3_0_0:SYS_CLK</td>
</tr>
<tr>
<td>AXI4mslave0</td>
<td>DDR3_0_0:AXI4slave0</td>
</tr>
</tbody>
</table>

9. Connect the APB3_0 pins as specified in the following table.

**Table 5**: **APB3_0 Pin Connections**

<table>
<thead>
<tr>
<th>Connect From</th>
<th>Connect To</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB3_0:APBmslave0</td>
<td>UARTapb_0:APB_bif</td>
</tr>
<tr>
<td>APB3_0:APBmslave1</td>
<td>COREGPIO_0:APB_bif</td>
</tr>
<tr>
<td>APB3_0:APBmslave2</td>
<td>SPI_Controller_0:APB_bif</td>
</tr>
</tbody>
</table>

10. Set the pins as follows on DDR3_0_0:
• Right-click the PLL_LOCK output pin, and select **Mark Unused**.
• Right-click the CTRLR_READY pin, and select **Promote to Top Level** for debug purpose. The 
CTRLR READY signal is used to monitor the status of the DDR controller.
• Ensure that the other pins are promoted to top level.

11. Set the pins as follows on SPI_Controller_0:
• Right-click the SPISSI pin, and select **Tie High**.
• Right-click the SPICLKi pin, and select **Tie Low**.
• Right-click the SPIINT, SPIRXAVAIL, SPITXRFM, SPIOEN, and SPIMODE pins, and select 
**Mark Unused**.
• Right-click the SPISDI, SPISCLKO and SPISDO pins, and select **Promote to Top Level**.

**Note**: In this tutorial, a single SPI Flash is used. Hence, while setting the pins of the SPI_Controller_0 block, we need only 0th bit of the SPISS. Bits 1:7 need to be sliced and marked as unused.

**Figure 24**: **Edit Slices Window**

![Edit Slices Window](image)
13. Set the pins as follows on UARTapb_0:
   - Right-click the RX and TX pins, and select Promote to Top Level.
   - Right-click the TXRDY, RXRDY, PARITY_ERR, OVERFLOW, FRAMING_ERR pins, and select Mark Unused.

14. Set the pins as follows on GPIO_0:
   - Right-click the GPIO_IN[3:0] pin, and select Tie Low.
   - Right-click the INT[3:0] pin, and select Mark Unused.
   - Right-click the GPIO_OUT[3:0] pin, and select Promote to Top Level.

15. Right-click the top SmartDesign canvas, and select Auto Arrange Layout.

The IP blocks are successfully connected. Figure 23, page 19 shows all the IP blocks of the Mi-subsystem connected.

### 2.4.5 Generating SmartDesign Component

To generate the SmartDesign component:

1. In Design Hierarchy, right-click top, and select Set As Root.
2. Save the project.
3. Click the Generate Component icon (shown in the following figure) on the SmartDesign toolbar.

![Generate Component Icon](image)

When the Mi-V component is generated, the Message window displays the message, “The top was generated successfully.”

4. Select the Build Hierarchy option and save the project.

### 2.4.6 Managing Timing Constraints

Before running the Libero design flow, you must derive the timing constraints and import the JTAG and asynchronous clocking constraints.

#### 2.4.6.1 Deriving Constraints

To derive constraints:

2. In the Manage Constraints window, select the Timing tab, and click Derive Constraints, as shown in the following figure.

![Derive Constraints Button](image)

The design hierarchy is built, and the top_derived_contraints.sdc file is generated in the project folder.

In the dialog box that appears, click Yes to associate the SDC file to the Synthesis, Place and Route, and Timing Verification tools, as shown in the following figure.

![Derived Constraints](image)

3. Save the project.
2.4.6.2 Importing Other Constraint Files

The JTAG clock constraint and the asynchronous clocks constraint must be imported. These constraints (.sdc) files are available in the DesignFiles_directory\Source folder.

To import and map the constraint files:
1. On the Timing tab, click Import.
2. Navigate to the DesignFiles_directory\Source folder, and select the timing_user_constraints.sdc file.
3. Select the Synthesis, Place and Route, and Timing Verification check boxes next to the timing_user_constraints.sdc file.
   This constraint file defines that the CCC_0_0 output clock and DDR3_0_0 AXI clock are asynchronous clocks.
4. Save the project.

2.4.7 Running the Libero Design Flow

This section describes the Libero design flow, which involves the following steps:

- Synthesis
- Place and Route
- Verify Timing
- Generate FPGA Array Data
- Configure Design Initialization Data and Memories
- Generate Design Initialization Data
- Generate Bitstream
- Run PROGRAM Action
- Generate SPI Flash Image
- Run PROGRAM_SPI_IMAGE Action

After each step is completed, a green tick mark appears next to the step on the Design Flow tab.

**Note:** To initialize the TCM in PolarFire using the system controller, a local parameter `l_cfg_hard_tcm0_en`, in the miv_rv32_opsrv_cfg_pkg.v file should be changed to `1'b1` prior to synthesis. See the 2.7 TCM section in the MIV_RV32 Handbook.

2.4.7.1 Synthesis

To synthesize the design:
1. Right-click Synthesis, select Configure Options and disable the Enable automatic compile point checkbox.
   When the synthesis is complete, a green tick mark appears next to Synthesize.
3. Right-click Synthesize and select View Report to view the synthesis report in the Reports tab.

2.4.7.2 Place and Route

The place and route process requires the following steps to be completed:

- Selecting the already imported io_constraints.pdc file
- Placing the DDR3_0_0 block using the I/O Editor
- Ensuring all the I/Os are locked

To complete these steps and to place and route the design:
2. On the I/O Attributes tab, select the check box next to the io_constraints.pdc file, as shown in the following figure. The io_constraints.pdc file contains the I/O assignment for reference clock, UART, GPIO, and SPI interfaces, and other top-level I/Os.
From the **Edit** drop-down list, select **Edit with I/O Editor**, as shown in the following figure.

In the I/O Editor, click the **Port View [active]** tab, and lock the CTRLR_READY port to pin C27, as shown in the following figure. This ensures that the CTRLR_READY port is assigned to pin C27, which is connected to an user LED for debug purposes.

To place the DDR3 I/O lanes, in the I/O Editor Design View, click the **Port** tab in the left pane, and select **DDR3**, as shown in the following figure.

Drag and place the DDR3 subsystem on the **NORTH_NE** side, as shown in the following figure. The DDR3 memory on the board is connected to DDR I/Os present on the north-east side.

The DDR3 subsystem is placed on the **NORTH_NE** side, as shown in the following figure.
7. From I/O Editor Port View tab, check if there are any unlocked I/Os, and lock them as mapped in the io_constraints.pdc file available in the Design_Files_Directory\Source\io folder.
8. Click Save.
9. Close the I/O Editor.
   A user.pdc file is created for DDR3_0_0 block in the Constraint Manager > I/O Attributes and Floor Planner tabs.

**Note:** DDR3_0_0 can also be placed using the fp_constraints.pdc. Import the fp_constraints.pdc from Constraint Manager > Floor Planner tab and select the place and route option after synthesis. This constraint file is available in the Design_Files_Directory\Source\fp folder.

10. Double-click Place and Route from the Design Flow tab.
    When place and route is successful, a green tick mark appears next to Place and Route.

### 2.4.7.3 Verify Timing

   When the design successfully meets the timing requirements, a green tick mark appears next to Verify Timing.
2. Right-click Verify Timing and select View Report to view the verify timing report in the Reports tab.

### 2.4.7.4 Generate FPGA Array Data

Double-click Generate FPGA Array Data on the Design Flow tab.

When the FPGA array data is generated, a green tick mark appears next to Generate FPGA Array Data.

### 2.4.7.5 Configure Design Initialization Data and Memories

The Configure Design Initialization Data and Memories step in the Libero design flow is used to configure the TCM initialization data and storage location. User can use µPROM, sNVM, or SPI Flash as storage location based on the size of the initialization data and design requirements. In this tutorial, the SPI Flash memory is used to store the TCM initialization data.

This process requires the user application executable file (HEX file) as input to initialize the TCM blocks after device power-up. The hex file is provided with the design files. For more information about building the user application, see Building the User Application Using SoftConsole.

**Note:** The HEX file available in the DesignFiles_Directory\Source folder is already modified to be compatible.

To generate an TCM initialization client and add it to an external SPI flash device:

2. On the Fabric RAMs tab, select top/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/gen_tcm0.u_opsrv_TCM_0/tcm_ram_macro.u_ram_0 from the list of logical instances, and click Edit, as shown in the following figure. The top/MIV_RV32_C0_0/MIV_RV32_C0_0/u_opsrv_0/gen_tcm0.u_opsrv_TCM_0/tcm_ram_macro.u_ram_0 instance is the MIV_RV32 processor's main memory. The System Controller initializes this instance with the imported client at power-up.
3. In the **Edit Fabric RAM Initialization Client** dialog box, set **Storage type** to **SPI-Flash** and click the Import button next to **Content from file**, as shown in the following figure.

4. In the **Import Memory File** dialog box, locate the **MiV_uart_blinky.hex** file from **DesignFiles_directory\Source** folder. Select the “Use relative path from project directory” option.
5. In the **Edit Fabric RAM Initialization Client** window, click **OK**.

6. On the **Fabric RAMs** tab, click **Apply**, as shown in the following figure.

7. In the **Design Initialization** tab, under Third stage (uPROM/sNVM/SPI-Flash), select the **SPI-Flash - No-binding Plaintext** option is selected and ensure that the SPI Clock divider value is set to 6, as shown in the following figure. This means that the imported user application will be written to SPI-Flash without encryption and authentication.

**Note:** The SPI Clock divider value specifies the required SPI SCK frequency to read the initialization data from SPI Flash. The SPI Clock divider value must be selected based on the external SPI Flash operating frequency range.

8. Click **Apply**.
This concludes the configuring of the storage type and application file for the fabric RAMs initialization.

2.4.7.6 Generate Design Initialization Data
1. Double-click Generate Design Initialization Data on the Design Flow tab. When the design initialization data is generated successfully, a green tick mark appears next to Generate Design Initialization Data in the Libero Design flow, and the following messages appear in the Log window:
   Info: 'Generate design initialization data' has completed successfully.
   Info: Stage 1 initialization client has been added to sNVM.
   Info: Stage 2_3 initialization client has been added to sNVM.
   Info: Stage 3 initialization client has been added to SPI.

2. Click the SPI Flash tab to verify that the bin file has been added, as shown in Figure 39, page 28.

Note: In order to streamline the SPI-Flash Programming support with FlashPro6, effective from Libero SoC v12.4, the vendor information is replaced with the density of the target memory.

Figure 39 • SPI Flash Tab

Note: For more information about design initialization, see UG0725: PolarFire FPGA Device Power-Up and Resets User Guide.
2.4.7.7 Generate Bitstream
To generate the programming bitstream:
- Double-click Generate Bitstream on the Design Flow tab.
When the bitstream is generated, a green tick mark appears next to Generate Bitstream.

2.4.7.8 Run PROGRAM Action
After generating the bitstream, the PolarFire Evaluation Board must be set up so the device is ready to be programmed. Also, the serial terminal emulation program (PuTTY) must be set up to view the output of the user application. This step involves the following:
1. Board Setup
2. Serial Terminal Emulation Program (PuTTY) Setup
3. Programming the PolarFire Device

2.4.7.8.1 Board Setup
To set up the board:
1. Ensure that the jumper settings on the board are as listed in the following table.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J18, J19, J20, J21, J22</td>
<td>Short pins 2 and 3 for programming the PolarFire FPGA through FTDI.</td>
</tr>
<tr>
<td>J28</td>
<td>Short pins 1 and 2 for programming through the on-board FlashPro5.</td>
</tr>
<tr>
<td>J26</td>
<td>Short pins 1 and 2 for programming through the FTDI SPI.</td>
</tr>
<tr>
<td>J27</td>
<td>Short pins 1 and 2 for programming through the FTDI SPI.</td>
</tr>
<tr>
<td>J23</td>
<td>Open pins 1 and 2 for programming SPI flash.</td>
</tr>
<tr>
<td>J4</td>
<td>Short pins 1 and 2 for manual power switching using SW3</td>
</tr>
<tr>
<td>J12</td>
<td>Short pins 3 and 4 for 2.5 V.</td>
</tr>
</tbody>
</table>

Note: For more information about the Jumper locations on the board, see the silkscreen provided in UG0747: PolarFire FPGA Evaluation Kit User Guide.

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the host PC to the J5 (USB) port on the PolarFire Evaluation Board using the USB cable.
4. Power on the board using the SW3 slide switch.

2.4.7.8.2 Serial Terminal Emulation Program (PuTTY) Setup
The user application (MiV_uart_blinky.hex file) prints the string Hello World! on the serial terminal through the UART interface.

Follow these steps to set up the serial terminal:
1. Start the PuTTY program.
2. Start Device Manager, note the second-highest COM port number, and use that in the PuTTY configuration. For example, in the list of ports shown in the following figure, COM93 is the port with the second highest number assigned to it.
3. Select **Serial** as the **Connection type**, as shown in the following figure.

![Figure 40 • COM Port Number](image)

4. Set the serial line to connect to the COM port number noted in Step 3.
5. Set the **Speed (baud)** to 115200 and **Flow Control** to **None**, as shown in the following figure.

![Figure 41 • Connection Type Selection](image)
6. Click **Open**.
PuTTY opens successfully, and the serial terminal emulation program is set up.

**2.4.7.8.3 Programming the PolarFire Device**

To program the PolarFire device:

- Double-click **Run PROGRAM Action** on the **Design Flow** tab.
  When the device is programmed, a green tick mark appears next to Run PROGRAM action.

**2.4.7.9 Generate SPI Flash Image**

To generate the SPI flash image:

- Double-click **Generate SPI Flash Image** on the **Design Flow** tab.
  When the SPI file image is successfully generated, a green tick mark appears next to **Generate SPI Flash Image**.
2.4.7.10 Run PROGRAM_SPI_IMAGE Action

To program the SPI image:

2. In the dialog box that appears, click Yes. When the SPI image is successfully programmed on to the device, a green tick mark appears next to Run PROGRAM_SPI_IMAGE.

After SPI flash programming is completed, the device needs to be reset to execute the application. The following sequence of operations occurs after device reset or power-cycling the board:

1. The PolarFire System Controller initializes the TCM with the user application code from the external SPI flash and releases the system reset.
2. The Mi-V processor exits reset after DDR3 controller is ready and executes the user application from the TCM. As a result, LEDs 4, 5, 6, and 7 blink, and the string **Hello World!** is printed on the serial terminal, as shown in the following figure.

*Figure 43 • Hello World String*

Hello World!

3. When the board is power cycled, the device performs the same sequence of operations. As a result, LEDs 4, 5, 6, and 7 blink, and **Hello World!** is printed again on the serial terminal, as shown in the following figure.

*Figure 44 • Hello World String After the Board is Power Cycled*

Hello World!

Hello World!
3 Building the User Application Using SoftConsole

This section describes how to build a RISC-V user application executable (.hex) file and debug it using SoftConsole.

Building the user application involves the following steps:

- Creating a Mi-V SoftConsole Project, page 33
- Downloading the Firmware Drivers, page 36
- Importing the Firmware Drivers, page 37
- Creating the main.c File, page 39
- Mapping Firmware Drivers and the Linker Script, page 40
- Mapping Memory and Peripheral Addresses, page 46
- Setting the UART Baud Rate, page 48
- Building the Mi-V Project, page 49

3.1 Creating a Mi-V SoftConsole Project

To create a Mi-V SoftConsole project:

1. Create a SoftConsole workspace folder on the host PC for storing SoftConsole projects. For example, D:\Tutorial\MiV_Workspace.
2. Start SoftConsole.
3. In the Workspace Launcher dialog box, paste D:\Tutorial\MiV_Workspace as the workspace location, and click Launch, as shown in the following figure.

*Figure 45 • Workspace Launcher*

When the workspace is successfully created, the SoftConsole main window opens.

4. Select File > New > Project, as shown in the following figure.

*Figure 46 • New C Project Creation*

5. Expand C/C++ and select C Project in the New Project dialog box.
6. In the **C Project** dialog box, do the following:
   - Enter a name for the project in the **Project name** field. For example, MiV_uart_blinky.
   - In the **Project type** pane, expand **Executable**, and select **Empty Project** and the Toolchain as **RISC-V Cross GCC**, as shown in the following figure. Then, click **Next**.

   **Figure 47 • C Project Dialog Box**

7. After selecting the platforms and configurations you want to deploy, click **Next**.

   **Figure 48 • Select Configurations Dialog Box**

8. Ensure that the Toolchain name and Toolchain path are as shown in **Figure 49**.
9. Click **Finish** in the **GNU RISC-V Cross Toolchain** wizard. An empty Mi-V project (MiV_uart_blinky) is created, as shown in the following figure.
3.2 Downloading the Firmware Drivers

The empty Mi-V project requires the MIV_RV32 Hardware Abstraction Layer (HAL) files and the following peripheral drivers:

- CoreGPIO
- CoreUARTapb
- CoreSPI Driver

Download the MIV_RV32 HAL files and drivers using the Firmware Catalog application.

To download the drivers:

1. Create a folder named firmware in the Mi-V project workspace.
2. Open Firmware Catalog. The following figure shows the Firmware Catalog window.

![Firmware Catalog Window](image)

3. If new cores are available, click Download them now!
4. Right-click MIV_RV32 Hardware Abstraction Layer (HAL), and select Generate.
5. In the Generate Options window, enter D:\Tutorial\MiV_Workspace\firmware as the project folder, and click OK.
   
   When the files are generated, the Reports window lists the files generated, as shown in the following figure.

![RISCV HAL Files Report](image)

6. Right-click CoreUARTapb Driver, and select Generate.
7. In the Generate Options window, enter D:\Tutorial\MiV_Workspace\firmware as the project folder, and click OK.
   
   When the files are generated, the Reports window lists the files, as shown in the following figure.

![CoreUARTapb Files Report](image)

8. Right-click CoreGPIO Driver, and select Generate.
9. In the Generate Options dialog box, enter D:\Tutorial\MiV_Workspace\firmware as the project folder, and click OK.
When the files are generated, the Reports window lists the files, as shown in the following figure.

Figure 54 • CoreGPIO Files Report

10. Right-click CoreSPI Driver, and select Generate.
11. In the Generate Options window, enter D:\tutorial\MiV_Workspace\firmware as the project folder, and click OK.
   When the files are generated, the Reports window lists the files, as shown in the following figure.

Figure 55 • CoreSPI Driver Files Report

The RISC-V HAL and firmware drivers are generated.

3.3 Importing the Firmware Drivers

After the driver files are downloaded, they must be imported into the empty project.

To import the drivers:

1. In SoftConsole, right-click the MiV_uart_blinky project, and select Import, as shown in the following figure.

Figure 56 • Import Option
2. In the **Import** dialog box, expand the **General** folder, and double-click **File System**, as shown in the following figure.

*Figure 57 • Import Dialog Box*

3. On the next page of the **Import** dialog box, do the following (see *Figure 58*):
   - Click **Browse**, and locate the `D:\Tutorial\MiV_Workspace\firmware` folder.
   - Select the **firmware** folder, and click **OK**.
   - Expand the **firmware** folder, and select the **drivers**, **hal**, and **riscv_hal** folders.
   - Click **Finish**.

*Figure 58 • Import Dialog Box - Page 2*

The `miv_rv32_hal`, **hal**, and driver files are imported into the **MiV_uart_blinky** project.
3.4 Creating the main.c File

To update the main.c file:

2. In the New Source File dialog box, enter main.c in the Source file field, and click Finish, as shown in the following figure.

*Figure 59 • main.c File Creation*

![Source File Creation Image]

The main.c file is created inside the project, as shown in the following figure.

*Figure 60 • The main.c file*

3. Copy all of the content of the DesignFiles_directory\Source\main.c file, and paste it in the main.c file of the SoftConsole project.
5. Similarly, create another file named hw_platform.h.
6. Copy all of the content of the DesignFiles_directory\Source\hw_platform.h file, and paste it in the newly created hw_platform.h file.

**Note:** The hw_platform.h file includes the system clock frequency, baud rate, and base addresses of peripherals. The hw_platform.h file appears as shown in Figure 61.
3.5 Mapping Firmware Drivers and the Linker Script

At this stage, the drivers and the MIV_RV32 HAL files are not mapped. Therefore, the corresponding header files in the `main.c` file are unresolved, as shown in the following figure.

To map the drivers and HAL files:

1. Right-click the MiV_uart_blinky project, and select **Properties**.
2. Expand **C/C++ Build**, and select **Settings**.
3. Set the configuration to **All Configurations**, as shown in the following figure. This setting applies the upcoming tool settings to both release and debug modes.

4. In the **Tool Settings** tab, expand **Target Processor**, and select the following settings:
   - Architecture: RV32I(-march=rv32i*)
   - Integer ABI: ILP32(-mabi=il32*)
   - Multiply extension: Enabled
5. Expand **GNU RISC-V Cross C Compiler**, and select **Includes**.
6. Click **Add** to add the driver and MIV_RV32 HAL directories, as shown in the following figure.

**Figure 65 • GNU RISC-V Cross C Compiler Tool Settings**

**Note:** This application does not require including system paths and other files.

7. In the **Add directory path** dialog box, click **Workspace**, as shown in the following figure.
8. In the **Folder Selection** dialog box, expand **MiV_uart_blinky project > drivers**, select the CoreGPIO folder, and click **OK**, as shown in the following figure.

**Figure 67 • CoreGPIO Folder Selection**

9. In the **Add directory path** dialog box, click **OK**.
   The CoreGPIO folder path is added, as shown in the following figure.

**Figure 68 • Tool Settings Tab with CoreGPIO Path Added**

10. Repeat the preceding steps to add the CoreUARTapb, CoreSPI, hal, MiV_RV32_HAL, and MiV_uart_blinky (ProjName) folder paths.
    The drivers and MiV_RV32_HAL files are successfully mapped, as shown in **Figure 69**.
11. Select the **GNU RISC-V Cross C Linker > General** to map the linker script.
12. Click **Add** as shown in Figure 65, and in the Add file path dialog, click **Workspace** as shown in Figure 66.
13. In the File Selection dialog box, expand MiV_uart_blinky and select the linker script as shown in the following figure.
14. The linker script is mapped as shown in Figure 71.
15. Select the Do not use standard start files (-nostartfiles) option as shown in Figure 71.
16. Select the GNU RISC-V Cross Create Flash Image > General and set Other Flags to "--change-section-lma "-0x80000000" as shown in Figure 72. This excludes the extended linear record in the first line of the hex file.
17. Click **Apply** and when prompted to rebuild, choose **Yes**.
18. Then click **Apply and Close**.

The firmware drivers and linker script are successfully mapped. Notice that the header files are now resolved in the `main.c` file.

### 3.6 Mapping Memory and Peripheral Addresses

In the Libero design flow, the Mi-V processor execution memory address is mapped to 0x80000000, and its size is set to 64 KB. This information must be checked in the linker script before building the application.

To map the memory address:

1. Open the linker script (`miv-rv32-ram.ld`) available in the MIV_RV32_HAL folder.
2. Ensure that the ram ORIGIN address is mapped to 0x80000000.
3. Ensure that the LENGTH of the ram is 64 KB.
4. Ensure that the RAM_START_ADDRESS is mapped to 0x80000000.
5. Ensure that the RAM_SIZE is 64 KB.
6. Ensure that the STACK_SIZE is 2 KB.
7. Ensure that the HEAP_SIZE is 2 KB.
8. Save the file.

**Note:** The MTVEC_OFFSET macro places trap vectors appropriately. This macro is already defined in the `miv-rv32-ram.ld` file.
The following figure shows the linker script.

**Figure 73**  Linker Script

```c
/* SVN $Revision: 12759 $ */
/* SVN $Date: 2020-05-14 19:43:19 +0530 (Thu, 14 May 2020) $ */

28 OUTPUT_ARCH( "riscv" )
29 ENTRY(_start)
30

ram (rwx) : ORIGIN = 0x80000000, LENGTH = 64k

28 RAM_START_ADDRESS = 0x80000000;
30 MTVEC_OFFSET = 0x100;
31 STACK_SIZE = 64k;
32 MEAP_SIZE = 2k;

---

In the Libero design flow, the UART, GPIO, and SPI peripheral addresses are mapped to 0x60000000, 0x60001000, and 0x60002000 respectively. This information needs to be provided in the `hw_platform.h` file.

To map the peripheral address:

1. Open the hardware platform header file (`hw_platform.h`).
2. Ensure that the SYS_CLK_FREQ macro is defined as 83333000UL.
3. Ensure that the COREUARTAPB0_BASE_ADDR macro is defined as 0x60000000UL.
4. Ensure that the COREGPIO_OUT_BASE_ADDR macro is defined as 0x60001000UL.
5. Ensure that the FLASH_CORE_SPI_BASE macro is defined as 0x60002000UL.
6. Save the file.

The following figure shows the `hw_platform.h` after these updates.

**Figure 74**  Updated hw_platform.h File

The memory and peripheral addresses are successfully mapped.
3.7 Setting the UART Baud Rate

The value of the `BAUD_VALUE_115200` macro in the `hw_platform.h` file must be defined according to the system clock frequency to achieve the UART baud rate of 115200. The baud value is calculated using the following formula.

\[
BAUD_VALUE = \frac{CLOCK}{16 \times BAUD_RATE} - 1
\]

To define the system clock frequency:

1. Look for `#define SYS_CLK_FREQ` statement in the `hw_platform.h` file.
2. Define it as:
   
   ```
   #define SYS_CLK_FREQ 83333000UL
   ```

The SYS_CLK_FREQ value must be same as that of the clock generated in the design.

The following figure shows the system clock frequency definition.

**Figure 75 • System Clock Frequency Definition**

```c
#ifndef HW_PLATFORM_H
#define HW_PLATFORM_H

/* Soft-processor clock definition
   * This is the only clock brought over from the Mi-V Libero design.
   */

#define SYS_CLK_FREQ 83333000UL
#endif
```
3.8 Building the Mi-V Project

To build the Mi-V project, right-click the MiV_uart_blinky project in SoftConsole, and select Build Project.

The project is built successfully, and the hex file is generated in the Debug folder, as shown in the following figure.

Figure 76 • Hex File

![Project Explorer](image)

The HEX file can be used for Design and Memory Initialization. For more information, see Configure Design Initialization Data and Memories, page 25.
3.9 Debugging the User Application Using SoftConsole

Before debugging, the board and the serial terminal must be set up. For more information about the board and serial terminal setup, see Board Setup, page 29 and Serial Terminal Emulation Program (PuTTY) Setup, page 29.

To debug the application:

1. From the Project Explorer, select the MiV_uart_blinky project, and then click the Debug icon from the SoftConsole toolbar, as shown in the following figure.

   ![Debug Icon](image)

2. In the Create, manage and run configurations window, double-click GDB OpenOCD Debugging to generate the debug configuration for the MiV_uart_blinky project.

3. Select the generated MiV_uart_blinky Debug configuration, and click Search Project (if by default not available), as shown in the following figure.

   ![Create, manage, and run configurations Window - Main Tab](image)
4. Select the **MiV_uart_blinky.elf** binary, and click **OK**, as shown in the following figure.

*Figure 79 • MiV_uart_blinky.elf Selection*

5. Go to the **Debugger** tab, and replace the Config Options, Executable, and Commands as follows:
   - Config Options: --file board/microsemi-riscv.cfg
   - Executable: `${cross_prefix}gdb${cross_suffix}
   - Commands:
     
     ```
     set mem inaccessible-by-default off
     set arch riscv:rv32
     set $target_riscv = 1
     set remotetimeout 7
     ```
Figure 80 • Create, manage, and run configurations Window – Debugger Tab
6. In **Debug Configurations** -> **Startup** tab, clear the **Pre-run/Restart reset** check box to halt the program at the main () function and clear the **Enable ARM semihosting** check box.

**Figure 81 • Debug Settings- Startup Tab**

7. Click **Apply**, and then click **Debug**, as shown in the preceding figure. The **Confirm Perspective Switch** dialog opens, as shown in **Figure 82**.

**Figure 82 • Confirm Perspective Switch Dialog Box**

8. Click **Yes**. The debugger halts the execution at the first instruction in the main.c file, as shown in the following figure.
9. On the SoftConsole toolbar, click **Resume** to resume the application execution, as shown in the following figure.

**Figure 84 • Resume Application Execution**

10. The string **Hello World!** is printed on the serial terminal, as shown in the following figure. Also, LEDs 4, 5, 6, 7 on the PolarFire Evaluation Board blink.

**Figure 85 • Hello World in Debug Mode**

11. On the SoftConsole menu, click **Run > Suspend** to suspend the execution of the application.
12. Click the **Registers** tab to view the values of the Mi-V internal registers, as shown in the following figure.
13. Click the **Variables** tab to view the values of variables in the source code, as shown in the following figure.

*Figure 87 • Variable Values*

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpio_pattern</td>
<td>uint32_t</td>
<td>5</td>
</tr>
<tr>
<td>delay_count</td>
<td>volatile int32_t</td>
<td>485870</td>
</tr>
</tbody>
</table>

14. From the SoftConsole toolbar, use the **Step Over** option to view the application execution line by line, or use the **Step Into** option to execute the instructions inside a function. Use the **Step Return** option to come out the function. You can also add breakpoints in the application source code.

15. On the SoftConsole toolbar, click **Terminate** to terminate the debugging of the application.

### 3.10 Debugging the User Application from DDR3 Memory

The SoftConsole debugger loads the application to the memory-mapped RAM based on the RAM start address specified in the `miv-rv32-ram.ld` linker file. The following figure shows the RAM Start Address parameters in the linker file.

**Figure 88 • RAM Start Address Parameters**

The SoftConsole reference project specifies the TCM start address, which is 0x80000000 (highlighted in Figure 88). To perform application debugging from DDR3 memory, modify this value to the DDR3 memory starting address, 0x80010000. After modifying the value, clean and build the project.

When the application is debugged from DDR3, the stack pointer and locations in the disassembly must point to DDR3 address, as shown in the following figure.

**Figure 89 • Debugging from DDR3**
Appendix 1: Programming the Device Using FlashPro Express

This chapter describes how to program the PolarFire device with the Job programming file using a FlashPro programmer. The default location of the .job file is: \mpf_tu0775_df\Programming_Job

To program the PolarFire device using FlashPro Express, perform the following steps:

1. Ensure that the jumper settings on the board are the same as listed in Table 6, page 29.
   **Note:** The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to the J5 (FTDI port) on the board.
4. Power on the board using the SW3 slide switch.
5. On the host PC, launch the FlashPro Express software.
6. Click **New** or select **New Job Project** from FlashPro Express Job from Project menu to create a new job project, as shown in the following figure.

   **Figure 90 • FlashPro Express Job Project**

7. Enter the following in the New Job Project from FlashPro Express Job dialog box:
   - **Programming job file**: Click Browse, and navigate to the location where the .job file is located and select the file. The default location is: <download_folder>\mpf_tu0775_df\Programming_Job.
     - \mpf_tu0775_df\Programming_Job\top_RevD
     - \mpf_tu0775_df\Programming_Job\top_RevE
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.
8. Click OK. The required programming file is selected and ready to be programmed in the device.
9. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click Refresh/Rescan Programmers.

**Figure 92 • Programming the Device**
10. Click **RUN**. When the device is programmed successfully, a RUN PASSED status is displayed as shown in the following figure.

*Figure 93 • FlashPro Express—RUN PASSED*

11. Close **FlashPro Express** or in the **Project** tab, click **Exit**.
This section lists documents that provide more information about RISC-V and other IP cores used to build the RISC-V subsystem.

- For more information about MIV_RV32, see *MIV_RV32 Handbook* from the Libero SoC Catalog.
- For more information about CoreJTAGDebug, see *CoreJTAGDebug_HB.pdf*.
- For more information about CoreAHBtoAPB3, see *CoreAHBtoAPB3_HB.pdf*.
- For more information about CoreAXItoAHBL, see *CoreAXItoAHBL_HB.pdf*.
- For more information about CoreGPIO, see *CoreGPIO_HB.pdf*.
- For more information about CoreUARTapb, see *CoreUARTapb_HB.pdf*.
- For more information about CoreAHBLite, see *CoreAHBLite_HB.pdf*.
- For more information about CoreAPB3, see *CoreAPB3_HB.pdf*.
- See the following documents on PolarFire FPGAs Documentation web page:
  - For more information about PolarFire Initialization Monitor, see *PolarFire FPGA and PolarFire SoC FPGA Device Power-Up and Resets User Guide*.
  - For more information about PolarFire Clock Conditioning Circuitry (CCC), see *PolarFire FPGA and PolarFire SoC FPGA Clocking Resources User Guide*.
  - For more information about PolarFire SRAM, see *PolarFire FPGA and PolarFire SoC FPGA Fabric User Guide*.
- For more information about Libero, ModelSim, and Synplify, see the *Libero SoC PolarFire webpage*.
- For more information about SoftConsole, see the *SoftConsole webpage*.
- For more information about loading a Job file using FlashPro Express, see the User Guide from *FlashPro Express - > Help - > User Guide*. 
If you are using Rev E kit the following are the configurations for DDR3 controller with the initialization and timing parameters for MT41K512M8DA-107: P part present on the Rev E PolarFire Evaluation Kit.

1. On General tab, set CCC PLL Clock Multiplier to 8, and DQ Width as 16, as shown in below figure. The clock multiplier value of 8 sets the CCC PLL reference clock frequency to 83.333 MHz. A reference clock of this frequency is required for the PLL present inside the DDR3 subsystem. The PLL generates a 666.666 MHz DDR3 memory clock frequency and a 166.666 MHz DDR3 AXI clock frequency. The DQ width is set to 16 to match the width of the DDR3 memory present on the board.

![General Tab](image-url)
2. The following figure shows initialization configuration settings for the DDR3 memory.

*Figure 95 • Memory Initialization*
3. The following figure shows timing configuration settings for the DDR3 memory.

*Figure 96 • Memory Timing*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tRAS (ns)</td>
<td>36</td>
</tr>
<tr>
<td>tRCD (ns)</td>
<td>13.91</td>
</tr>
<tr>
<td>tRP (ns)</td>
<td>13.91</td>
</tr>
<tr>
<td>tRC (ns)</td>
<td>49.5</td>
</tr>
<tr>
<td>tWR (ns)</td>
<td>15</td>
</tr>
<tr>
<td>tFAW (ns)</td>
<td>30</td>
</tr>
<tr>
<td>tWTR (cycles)</td>
<td>5</td>
</tr>
<tr>
<td>tRRD (ns)</td>
<td>6</td>
</tr>
<tr>
<td>tRTP (ns)</td>
<td>7.5</td>
</tr>
<tr>
<td>tREFI (us)</td>
<td>7.8</td>
</tr>
<tr>
<td>tRFC (ns)</td>
<td>260</td>
</tr>
<tr>
<td>tZQinit (cycles)</td>
<td>512</td>
</tr>
<tr>
<td>ZQ Calibration Type</td>
<td>Short</td>
</tr>
</tbody>
</table>
4. The following figure shows controller configuration settings for the DDR3 memory.

*Figure 97 • Controller*

5. The following figure shows miscellaneous configuration settings for the DDR3 memory.

*Figure 98 • Misc*

**Note:** Return to section *Instantiating APB3 Bus*, page 12 for completing the design implementation.