Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.
## Contents

1 Revision History .................................................. 1  
1.1 Revision 7.0 ................................................... 1  
1.2 Revision 6.0 ................................................... 1  
1.3 Revision 5.0 ................................................... 1  
1.4 Revision 4.0 ................................................... 1  
1.5 Revision 3.0 ................................................... 1  
1.6 Revision 2.0 ................................................... 1  
1.7 Revision 1.0 ................................................... 1  

2 PolarFire FPGA Auto Update and  
In-Application Programming ........................................... 2  
2.1 PF_SYSTEM_SERVICES Overview ................................... 3  
2.2 Design Requirements ........................................... 5  
2.3 Prerequisites .................................................. 5  
2.4 Demo Design ................................................ 6  
2.4.1 Design Implementation ...................................... 8  
2.5 Clocking Structure ........................................... 18  

3 Libero Design Flow .................................................. 19  
3.1 Synthesis .................................................. 20  
3.2 Place and Route ............................................. 20  
3.2.1 Resource Utilization ........................................ 20  
3.3 Verify Timing .............................................. 21  
3.4 Generate FPGA Array Data ..................................... 21  
3.5 Configure Design Initialization Data and Memories .............. 21  
3.6 Configure Programming Options ................................ 24  
3.7 Generate Bitstream .......................................... 25  
3.8 Generating the SPI programming Images ......................... 25  
3.9 Export FlashPro Express Job ..................................... 26  
3.10 Programming the Device ....................................... 27  
3.10.1 Programming the Device on the Evaluation Board .......... 27  
3.10.2 Programming the Device on the Splash Board .......... 28  

4 Serial Terminal Emulation Program Setup .......................... 30  

5 Running the Demo .................................................. 32  
5.1 Programming On-board SPI Flash Using Libero ................. 33  
5.2 Running Auto Update .......................................... 34  
5.3 Running Authentication ......................................... 34  
5.4 Running Auto Programming ..................................... 35  
5.5 Running IAP ................................................ 36  

6 Appendix 1: Programming On-board SPI Flash Using the Fabric Logic Through  
the Host Loader .................................................. 37  

7 Appendix 2: Programming the Device and External SPI Flash Using  
FlashPro Express .................................................. 39
Figures

Figure 1 PF_SYSTEM_SERVICES IP Interfacing with Fabric User Logic .................................................. 3
Figure 2 Firmware catalog .................................................................................................................. 4
Figure 3 PolarFire Programming Design Block Diagram .................................................................. 6
Figure 4 Accessing On-board SPI Flash Using Fabric ....................................................................... 7
Figure 5 SPI Flash Memory ................................................................................................................. 7
Figure 6 Top Level Libero Design ...................................................................................................... 8
Figure 7 PF_INIT_MONITOR Configuration ...................................................................................... 9
Figure 8 PF_CCC_0 Input Clock Configuration ................................................................................. 10
Figure 9 PF_CCC_0 Output Clock Configuration .............................................................................. 11
Figure 10 Mi-V Configuration ........................................................................................................... 12
Figure 11 Mi-V RV32 Configuration ................................................................................................ 13
Figure 12 CoreGPIO_0 Configuration ................................................................................................ 14
Figure 13 CoreSPI Configuration ...................................................................................................... 15
Figure 14 Memory Map ....................................................................................................................... 16
Figure 15 CoreAPB3_0 Configuration ................................................................................................ 17
Figure 16 Clocking Structure ................................................................................................................ 18
Figure 17 Libero Design Flow Options ............................................................................................... 19
Figure 18 Design and Memory Initialization ....................................................................................... 21
Figure 19 Fabric RAMs Tab ................................................................................................................ 22
Figure 20 Edit Fabric RAM Initialization Client .................................................................................. 22
Figure 21 Apply Fabric RAM Content ................................................................................................. 23
Figure 22 Client in the sNVM Option .................................................................................................. 24
Figure 23 Configure Programming Options ....................................................................................... 24
Figure 24 Generate Bitstream—Configure Bitstream Options ............................................................... 25
Figure 25 Generating the .SPI Programming Images .......................................................................... 26
Figure 26 Export FlashPro Express Job .............................................................................................. 27
Figure 27 Board Setup—Evaluation Kit ............................................................................................. 28
Figure 28 Board Setup—Splash Kit ..................................................................................................... 29
Figure 29 COM Port Number ............................................................................................................... 30
Figure 30 Select Serial as the Connection Type .................................................................................. 30
Figure 31 PuTTY Configuration .......................................................................................................... 31
Figure 32 Authentication and Programming Options .......................................................................... 32
Figure 33 Authentication Error ............................................................................................................ 32
Figure 34 Configure Programming Settings ...................................................................................... 33
Figure 35 Configure Design Initialization Data and Memories Option .................................................. 33
Figure 36 SPI Flash Tab ....................................................................................................................... 33
Figure 37 SPI Flash Programming ....................................................................................................... 34
Figure 38 Auto Update ......................................................................................................................... 34
Figure 39 Successful Bitstream Authentication .................................................................................... 34
Figure 40 Successful IAP Image Authentication ................................................................................... 34
Figure 41 Notifying ERASE Action ..................................................................................................... 35
Figure 42 Successful Auto Programming ............................................................................................. 35
Figure 43 Successful IAP at Index 2 ..................................................................................................... 36
Figure 44 Successful IAP by Address .................................................................................................. 36
Figure 45 Erasing SPI Flash ................................................................................................................ 37
Figure 46 Command Prompt Status .................................................................................................... 38
Figure 47 FlashPro Express Job Project ............................................................................................. 39
Figure 48 New Job Project from FlashPro Express Job ....................................................................... 40
Figure 49 Programming the Device ..................................................................................................... 40
Figure 50 FlashPro Express—RUN PASSED ..................................................................................... 41
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>System Services Descriptor</td>
<td>3</td>
</tr>
<tr>
<td>Table 2</td>
<td>Design Requirements</td>
<td>5</td>
</tr>
<tr>
<td>Table 3</td>
<td>I/O Signals</td>
<td>8</td>
</tr>
<tr>
<td>Table 4</td>
<td>Resource Utilization—Evaluation Board</td>
<td>20</td>
</tr>
<tr>
<td>Table 5</td>
<td>Resource Utilization—Splash Board</td>
<td>20</td>
</tr>
<tr>
<td>Table 6</td>
<td>Programming Images</td>
<td>25</td>
</tr>
<tr>
<td>Table 7</td>
<td>Jumper Settings—Evaluation Board</td>
<td>27</td>
</tr>
<tr>
<td>Table 8</td>
<td>Jumper Settings—Splash Board</td>
<td>28</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 7.0
Added Appendix 3: Running the TCL Script, page 42.

1.2 Revision 6.0
- The following is a summary of the changes made in this revision.
- Updated the document for Libero SoC v12.2.
- Removed the references to Libero version numbers.

1.3 Revision 5.0
The document was updated for Libero SoC v12.0.

1.4 Revision 4.0
Merged Splash kit related content and updated the document for Libero SoC PolarFire v2.3 release.

1.5 Revision 3.0
The document was updated for Libero SoC PolarFire v2.2 release.

1.6 Revision 2.0
The document was updated for Libero SoC PolarFire v2.1 release.

1.7 Revision 1.0
The first publication of this document.
PolarFire FPGA Auto Update and In-Application Programming

PolarFire® FPGAs support the SPI master programming mode for auto update and in-application programming (IAP). In this programming mode, the programming images are stored in an external SPI flash memory.

Auto update—on power-up, if the version of the update image is found to be different from the current programmed version, the System Controller reads the update image bitstream from the external SPI flash memory and programs the device.

IAP—the user application initiates the program action and the System Controller reads the bitstream from the external SPI flash memory to program the device.

The System Controller supports fetching programming images from the SPI Flash device based on the Index value or direct addressing. The SPI directory contains the start addresses of the programming images.

The following components of PolarFire devices are programmable:

• FPGA fabric
• Secure non-volatile memory (sNVM)
• User security settings (keys, passcodes, and locks)

This document explains how to use the accompanying design to demonstrate the auto update and IAP features on the PolarFire Evaluation/Splash board.

The on-board 1 GB Micron SPI flash device is connected to System Controller SPI and can be programmed using the fabric logic or Libero® SoC software.

This application note includes the Mi-V soft processor, which initiates the system service requests for the device programming and enables the PF_SYSTEM_SERVICES core to access the System Controller. For more information about the design implementation, and the necessary blocks and IP cores instantiated in Libero SoC, see Demo Design, page 6.

This design can be programmed using any of the following options:

• Using the pre-generated .job file: To program the device using the .job file provided along with the design, see Appendix 2: Programming the Device and External SPI Flash Using FlashPro Express, page 39.
• Using Libero SoC: To program the device using Libero SoC, see Libero Design Flow, page 19.

This design can be used as reference to build a fabric design with programming features.
2.1 PF_SYSTEM_SERVICES Overview

System Controller actions are initiated by the fabric logic through the System Service Interface (SSI) of the System Controller. The fabric logic requires the PF_SYSTEM_SERVICES for initiating the system services. A service request interrupts to the System Controller is triggered when the fabric user logic writes a 16-bit system service descriptor to the SSI. The lower seven bits of the descriptor specifies the service to be performed. The upper nine bits specify the address offset (0–511) in the 2 KB mailbox RAM. The mailbox address specifies the service-specific data structure used for any additional inputs or outputs for the service. The fabric logic must write additional parameters to the mailbox before requesting a system service. The following table lists the system service descriptor bits.

<table>
<thead>
<tr>
<th>Descriptor Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:7</td>
<td>MBOXADDR</td>
</tr>
<tr>
<td>6:0</td>
<td>SERVICEID</td>
</tr>
</tbody>
</table>

SSI consists of an asynchronous command-response interface that transfers a system service command from the fabric master to the System Controller and the status from the System Controller to the fabric master. The following figure shows how the PF_SYSTEM_SERVICES Interfaces with the fabric logic.

The system services driver and the sample SoftConsole project are generated from Firmware Catalog as shown in Figure 2, page 4.
The Mi-V soft processor is compatible with only SoftConsole v5.2 or later. The application files `main.c` and `hw_platform.h` are modified to provide the programming user options, system clock frequency, and APB peripheral addresses.

**Figure 2** • Firmware catalog
2.2 Design Requirements

The following table lists the resources required to run the design.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows 7, 8.1, or 10</td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
</tr>
<tr>
<td>PolarFire Evaluation Kit (MPF300-EVAL-KIT)</td>
<td>Rev D or later</td>
</tr>
<tr>
<td>PolarFire Splash Kit (MPF300TS-1FCG484E)</td>
<td>Rev 2 or later</td>
</tr>
<tr>
<td>Host PC</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>FlashPro Express</td>
<td></td>
</tr>
<tr>
<td>Libero SoC</td>
<td></td>
</tr>
<tr>
<td>SoftConsole</td>
<td></td>
</tr>
<tr>
<td>Serial Terminal Emulation Program</td>
<td>PuTTY or HyperTerminal</td>
</tr>
</tbody>
</table>

**Note:** Any serial terminal emulation program can be used. PuTTY is used in this application note.

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

2.3 Prerequisites

Before you begin:

1. For demo design files download link:
   
   For Evaluation kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_ac466_eval_df
   
   For Splash kit:
   http://soc.microsemi.com/download/rsc/?f=mpf_ac466_splash_df

2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location.
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads
   
   The latest versions of ModelSim and Synplify Pro are included in the Libero SoC installation package.
2.4 Demo Design

The following steps describe the data flow in the design:

1. The host PC sends the system service requests to CoreUARTapb block through the UART Interface.
2. The Mi-V soft processor initializes the System Controller using the PF_SYSTEM_SERVICES and sends the requested system service command to the System Controller.
3. The System Controller executes the system service command by reading the bitstream images from the external SPI flash and sends the relevant response to the PF_SYSTEM_SERVICES over the mailbox interface.
4. The Mi-V processor receives the service response and forwards the data to the UART interface.

The following figure shows the block diagram of the PolarFire programming design.

Figure 3 • PolarFire Programming Design Block Diagram
To initiate an auto update or IAP system service request, the on-board SPI flash must be programmed with programming images. The fabric logic interfaces to the on-board SPI flash using the SPI controller and PF_SPI macro. When the System Controller’s SPI is enabled and configured as master, the System Controller hands over the control of the SPI to the fabric on device power-up. The fabric logic programs the on-board SPI flash with flash directory and programming images using the UART interface. The programming images are transferred from the host PC using SPI flash loader (spi_loader.exe).

The on-board SPI flash can be programmed using fabric logic as shown in the following figure. For more information, see Appendix 1: Programming On-board SPI Flash Using the Fabric Logic Through the Host Loader, page 37.

**Figure 4 • Accessing On-board SPI Flash Using Fabric**

The following figure shows the SPI flash memory with directory and programming images.

**Figure 5 • SPI Flash Memory**
When System Controller receives programming or authentication system service from fabric user logic, the System Controller fetches the programming images from the on-board SPI flash to execute the service request. In this application note, the following system services are initiated on user request.

- Bitstream authentication
- IAP image authentication
- Auto update
- IAP

For more information about the preceding services, see the UG0714: PolarFire FPGA Programming User Guide.

### 2.4.1 Design Implementation

The following figure shows the top-level Libero design of the PolarFire system services design.

**Figure 6 • Top Level Libero Design**

The following table lists the important I/O signals of the design.

**Table 3 • I/O Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF_CLK_0</td>
<td>Input 50 MHz clock from the on-board 50 MHz oscillator</td>
</tr>
<tr>
<td>resetn</td>
<td>On-board reset push-button for the PolarFire device</td>
</tr>
<tr>
<td>RX</td>
<td>Input signals received from the serial UART terminal</td>
</tr>
<tr>
<td>TX</td>
<td>Output signals transmitted to the serial UART terminal</td>
</tr>
<tr>
<td>GPIO_OUT[3:0]</td>
<td>On-board LED outputs</td>
</tr>
<tr>
<td>GPIO_IN[3:0]</td>
<td>To interface on-board DIP switches</td>
</tr>
</tbody>
</table>
2.4.1.1 **PF_INIT_MONITOR**

The PolarFire Initialization Monitor gets the status of device initialization. The following figure shows the PF_INIT_MONITOR configuration.

*Figure 7 • PF_INIT_MONITOR Configuration*
2.4.1.2 PF_CCC_0 Configuration

The PolarFire Clock Conditioning Circuitry (CCC) block takes an input clock of 50 MHz from the on-board oscillator and generates a 100 MHz fabric clock to the Mi-V processor subsystem and other peripherals. The following figures show the input and output clock configurations.

Figure 8 • PF_CCC_0 Input Clock Configuration
2.4.1.3 Mi-V Soft Processor Configuration

The Mi-V soft processor default Reset Vector Address value is 0x8000_0000. After the device reset, the processor executes the application from TCM, which is mapped to 0x80000000, hence the Reset Vector Address is set to 0x80000000 as shown in the following figure.

TCM is the main memory of the Mi-V processor. It gets initialized with the user application from sNVM.

In the Mi-V processor memory map, the 0x8000_0000 to 8000_FFFF range is defined for TCM memory interface and the 0x6000_0000 to 0x6FFF_FFFF range is defined for APB interface.
Figure 10 • Mi-V Configuration

- **Memory depth**: This field is set to 16384 words to accommodate an application of up to 64 KB into TCM. The present application is below 50 KB so this can fit into either sNVM or µPROM. In this design, sNVM is selected as data storage client as shown in the following figure.
2.4.1.4 CoreGPIO_0 Configuration

The CoreGPIO IP controls the on-board LEDs using GPIOs. It is connected to Mi-V soft processor as an APB slave. The configuration settings of the COREGPIO_0 IP are as follows:

In the Global Configurations pane:

- **APB Data width**: Set to 32
  The design uses 32-bit data width for APB read and write data.
- **Number of I/Os**: Set to 4
  The design controls 2 on-board LEDs for output and 2 DIP Switches for input.
- **I/O Bit**: The following list shows the sub-options under I/O Bit option.
  - **Output on reset**: Set to 0
  - **Fixed Config**: Yes
  - **I/O type**: As shown in the following figure, first two I/Os are configured as output and the last two I/Os are configured as input.

**Note:** The first two I/Os configured as output are used by the design and last two I/Os are not used. The I/Os are interfaced to on-board LEDS and DIP switches.

- **Interrupt Type**: Disabled
  When I/O states change, no interrupt is required for the application.

The following figure shows the CoreGPIO_0 configuration.
Figure 12 • CoreGPIO_0 Configuration
2.4.1.5 CoreSPI Configuration

The CoreSPI is used to program the external SPI flash using Mi-V processor. PF_SPI macro interfaces the fabric logic to the external SPI flash, which is connected to System Controller.

- **APB Data Width**: select 32 as APB data width in the design. The default value is 8.
- **Mode**: select Motorola Mode (default) as the target SPI slave supports Motorola mode. **Mode 3** is selected under **Motorola Configuration**.
- **Frame Size**: enter 8. The default value is 4.
- **FIFO Depth**: enter 32 to store maximum frames (Tx and Rx) in FIFO. The default value is 4.
- **Clock Rate**: enter 16. The default value is 8.
  The SPI clock becomes system clock/ 2*(16+1).
- **Keep SSEL active**: enabled to keep the slave peripheral active between back to back data transfers.

The following figure shows the CoreSPI configurator.

*Figure 13 • CoreSPI Configuration*
### 2.4.1.6 Design Memory Map

The Mi-V processor bus interface memory map is shown in the following figure.

**Figure 14 • Memory Map**

![Memory Map](image)

<table>
<thead>
<tr>
<th>Master/Bus/Peripheral</th>
<th>Start Address</th>
<th>Range</th>
<th>DRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>tscp/MIV_FV32_C0_0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoreAPB0_0 &amp; APB3immediate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoreUART0_0 &amp; APB_bif</td>
<td>0x6000_0000</td>
<td>0x0000_1000</td>
<td></td>
</tr>
<tr>
<td>CoreGPIO_OUT &amp; APB_bif</td>
<td>0x6000_1000</td>
<td>0x0000_1000</td>
<td></td>
</tr>
<tr>
<td>CoreSysServices_PF_0 &amp; APBSlave</td>
<td>0x6000_2000</td>
<td>0x0000_1000</td>
<td></td>
</tr>
<tr>
<td>Core_SPI &amp; APB_bif</td>
<td>0x6000_3000</td>
<td>0x0000_1000</td>
<td></td>
</tr>
</tbody>
</table>
2.4.1.6.1 CoreAPB3 Configuration

The CoreAPB3 IP connects the peripherals, PF_SYSTEM_SERVICES, CoreSPI, CoreGPIO, and CoreUARTapb as slaves. The configuration settings of COREAPB3 are as follows:

- **APB Master Data bus width:** 32-bit
  The design uses 32-bit data width for APB read and write data.

- **Number of address bits driven by master:** 16
  The Mi-V processor accesses the slaves using the 16-bit. The final addresses for these slaves are translated into 0x6000_0000, 0x6000_1000, 0x6000_2000 and 0x6000_3000.

- **Enabled APB slave slots:** Slot 0 for CoreUARTapb, Slot 1 for CoreGPIO, Slot 2 for PF_SYSTEM_SERVICES, and Slot 3 for CoreSPI.

The following figure shows the CoreAPB3 configuration.

*Figure 15 • CoreAPB3_0 Configuration*
2.5 Clocking Structure

The following figure shows the clocking structure of this design. The Mi-V processor supports up to 120 MHz. This design uses a 100 MHz system clock for configuring the APB peripherals.

Figure 16 • Clocking Structure
3 Libero Design Flow

The Libero design flow involves running the following processes in the Libero SoC:

• Synthesis, page 20
• Place and Route, page 20
• Verify Timing, page 21
• Generate FPGA Array Data, page 21
• Configure Design Initialization Data and Memories, page 21
• Configure Programming Options, page 24
• Generate Bitstream, page 25
• Export FlashPro Express Job, page 26
• Programming the Device, page 27

Note: To initialize the TCM in PolarFire using the system controller, a local parameter `lcfg_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*.

The following figure shows these options in the Design Flow tab.

*Figure 17 • Libero Design Flow Options*
3.1 **Synthesis**

To synthesize the design, perform the following steps:

1. On the **Design Flow** window, double-click **Synthesis**. When the synthesis is successful, a green tick mark appears as shown in Figure 17, page 19.
2. Right-click **Synthesis** and select **View Report** to view the synthesis report and log files in the **Reports** tab.

   **Note:** Set the correct tool profile before you start Synthesis.

   **Note:** The files, `top.srr` and the `top_compile_netlist.log` files are recommended to be viewed for debugging synthesis and compile errors.

3.2 **Place and Route**

The Place and Route process requires the I/O, timing, and floor planner constraints. This design includes the following constraint files in the **Constraint Manager** window:

- The `io_constraints.pdc` file for the I/O assignments
- The `top_derived_constraints.sdc` file for timing constraints
- The `timing_user_constraints.sdc` file for creating the JTAG clock with 30 MHz frequency.

To Place and Route, on the **Design Flow** window, double-click **Place and Route**. When place and route is successful, a green tick mark appears next to Place and Route.

   **Note:** The file, `top_place_and_route_constraint_coverage.xml` is recommended to be viewed for place and route constraint coverage.

3.2.1 **Resource Utilization**

The resource utilization report is written to the `top_layout_log.log` file in the **Reports** tab -> top reports -> Place and Route. It lists the resource utilization of the design after place and route. These values may vary slightly for different Libero runs, settings, and seed values.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14339</td>
<td>299544</td>
<td>4.79</td>
</tr>
<tr>
<td>DFF</td>
<td>8066</td>
<td>299544</td>
<td>2.69</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>510</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Element</td>
<td>15187</td>
<td>299544</td>
<td>5.07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14693</td>
<td>299544</td>
<td>4.91</td>
</tr>
<tr>
<td>DFF</td>
<td>8069</td>
<td>299544</td>
<td>2.69</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>242</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Element</td>
<td>15511</td>
<td>299544</td>
<td>5.18</td>
</tr>
</tbody>
</table>
3.3 Verify Timing

To verify timing, perform the following steps:

1. On the Design Flow window, double-click Verify Timing. When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 17, page 19.
2. Right-click Verify Timing and select View Report, to view the verify timing report and log files in the Reports tab.

3.4 Generate FPGA Array Data

To generate the FPGA array data, perform the following steps:

2. A green tick mark is displayed after the successful generation of the FPGA array data as shown in Figure 17, page 19.

3.5 Configure Design Initialization Data and Memories

The Configure Design Initialization Data and Memories step generates the TCM initialization client and adds it to sNVM, μPROM, or an external SPI flash, based on the type of non-volatile memory selected. In this design, the TCM initialization client is stored in the sNVM.

This process requires the user application executable file (hex file) to initialize the TCM options on device power-up. The hex file (application.hex) is available in the DesignFiles Directory\Libero_Project\hw_project folder. When the hex file is imported, a memory initialization client is generated for TCM options.

Follow these steps:

1. On the Design Flow window, double-click Configure Design Initialization Data and Memories. The Design and Memory Initialization window opens as shown in the following figure.

Figure 18 • Design and Memory Initialization
2. Select Fabric RAMs tab and select tcm_ram client from the list and click Edit as shown in the following figure.

Figure 19 • Fabric RAMs Tab

3. In the Edit Fabric RAM Initialization Client dialog box, select Content from file option, and locate the application.hex file from DesignFiles_directory\Libero_Project\hw_project folder and Click OK as shown in the following figure.

Figure 20 • Edit Fabric RAM Initialization Client
4. Click **Apply** as shown in the following figure.

**Figure 21 • Apply Fabric RAM Content**

5. In the **Design Initialization** tab, click **Apply**.

6. From Libero Design Flow, double-click **Generate Design Initialization Data** to generate design initialization data.

   After successful generation of the Initialization data, a green tick mark appears next to **Generate Initialization Data** option as shown in the **Figure 17**, page 19.

   The following **Figure 22**, page 24 shows the client in the sNVM after **Generate Design Initialization Data**.
3.6 Configure Programming Options

The Design version and user code (Silicon signature) are configured in this step. Double-click Configure Programming Options to give values as shown in the following figure.
### 3.7 Generate Bitstream

To generate the bitstream, perform the following steps:

1. Right-click **Generate Bitstream** and select **Configure Options**... to select the bitstream components—Custom security, Fabric, and sNVM.

*Figure 24 • Generate Bitstream—Configure Bitstream Options*

2. On the **Design Flow** window, double-click **Generate Bitstream**. When the bitstream is successfully generated, a green tick mark appears as shown in *Figure 17*, page 19.

3. Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.

### 3.8 Generating the SPI programming Images

The following programming images are generated and copied to external SPI flash memory.

*Table 6 • Programming Images*

<table>
<thead>
<tr>
<th>Image Name</th>
<th>Version</th>
<th>Silicon Signature/User Code</th>
<th>Image Index in SPI Flash Directory</th>
<th>Image Address in SPI Flash Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>golden_image_v0.spi</td>
<td>0</td>
<td>N/A</td>
<td>0</td>
<td>0x00000400</td>
</tr>
<tr>
<td>update_image_v2.spi</td>
<td>2</td>
<td>0x23456789</td>
<td>1</td>
<td>0x00A0000</td>
</tr>
<tr>
<td>iap_image_v5.spi</td>
<td>5</td>
<td>0x56789ABC</td>
<td>2</td>
<td>0x01400000</td>
</tr>
</tbody>
</table>

**Note:** The golden image cannot contain any security or Silicon signature information.
Figure 25 • Generating the .SPI Programming Images

Note: The golden image cannot contain any security or Silicon signature information.

The .SPI programming images are generated using Export Bitstream option as shown in Figure 25, page 26. Before generating the .SPI images modify the Design version and Silicon Signature as shown in Configure Programming Options, page 24.

3.9 Export FlashPro Express Job

To generate .job file, perform the following steps:

On the Design Flow tab, double-click Export FlashPro Express Job and select Design and SPI Flash as shown in figure. The exported job file contains the data contents to be programmed into PolarFire FPGA and external SPI flash. This Job file is utilized in FlashPro Express software to program both Device and external SPI flash as shown in Appendix 2: Programming the Device and External SPI Flash Using FlashPro Express, page 39.
3.10 Programming the Device

To program the device, see any of the following sections based on the board used.

- Programming the Device on the Evaluation board
- Programming the Device on the Splash board

3.10.1 Programming the Device on the Evaluation Board

After generating the bitstream, the PolarFire device must be programmed with the Auto Update and IAP design.

To program the PolarFire device, perform the following steps:

1. Ensure that the following jumper settings are set on the board.

Table 7 • Jumper Settings—Evaluation Board

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

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.
The following figure shows the board setup after these connections are made.

**Figure 27 • Board Setup—Evaluation Kit**

5. On the **Libero Design** Flow, double-click **Run PROGRAM Action**.
   The device is successfully programmed and the on-board LEDs glow. A green tick mark appears next to **Run PROGRAM Action** as shown in Figure 17, page 19.

### 3.10.2 Programming the Device on the Splash Board

After generating the bitstream, the PolarFire device must be programmed with the Auto Update and IAP design.

To program the PolarFire device, perform the following steps:

1. Ensure that the following jumper settings are set on the board.

   **Table 8 • Jumper Settings—Splash Board**

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J5, J6, J7, J8, and J9</td>
<td>Close pin 2 and 3 for programming the PolarFire FPGA through FTDI</td>
</tr>
<tr>
<td>J11</td>
<td>Close pin 1 and 2 for programming through FTDI chip</td>
</tr>
<tr>
<td>J10</td>
<td>Close pin 1 and 2 for programming through FTDI SPI</td>
</tr>
<tr>
<td>J4</td>
<td>Close pin 1 and 2 for manual power switching using SW1</td>
</tr>
<tr>
<td>J3</td>
<td>Open pin 1 and 2 for 1.0 V</td>
</tr>
<tr>
<td>J35</td>
<td>Open pin 1 and 2 for SPI master mode programming</td>
</tr>
</tbody>
</table>

2. Connect the power supply cable to the J2 connector on the board.
3. Connect the USB cable from the host PC to the J1 (FTDI port) on the board.
4. Power ON the board using the SW1 slide switch.
The following figure shows the board setup after these connections are made.

*Figure 28 • Board Setup—Splash Kit*

5. On the **Design Flow** window, double-click **Run PROGRAM Action**.

The device is successfully programmed and the on-board LEDs glow. A green tick mark appears next to **Run PROGRAM Action** as shown in *Figure 17*, page 19.
The user application receives programming commands on the serial terminal through the UART interface. This chapter describes how to set up the serial terminal program.

To setup PuTTY, perform the following steps:

1. Connect the USB cable from the host PC to the J5 (USB) port on the Evaluation board or J1 (USB) port on the Splash board.
2. Connect the power supply cable to the J9 connector on the Evaluation board or J2 connector on the Splash board.
3. Power on the Evaluation board using the SW3 or Splash board using the SW1 slide switch.
4. From the host PC, click Start and open Device Manager to note the second highest COM Port number and use that in the PuTTY configuration. In this example, COM Port 9 (COM9) is selected as shown in the following figure. COM Port-numbers may vary.

   Figure 29 • COM Port Number

5. From the host PC, click Start, and then find and select the PuTTY program.
6. Select Serial as the Connection type as shown in the following figure.

   Figure 30 • Select Serial as the Connection Type

7. Set the Serial line to connect to COM port number noted in Step 4.
8. Set the Speed (baud) to 115200 as shown in the following figure.
9. Set the **Flow control** to **None** as shown in the following figure and click **Open**.

*Figure 31 • PuTTY Configuration*

PuTTY opens successfully, and this completes the serial terminal emulation program setup. See *Running the Demo, page 32.*
5 Running the Demo

This section describes how to run the authentication, auto update, and IAP. The following procedure assumes that the serial terminal is setup, for more information about setting up the serial terminal, see Serial Terminal Emulation Program Setup, page 30.

The on-board 1 GB Micron SPI flash device is connected to System Controller SPI and can be programmed using the Libero SoC PolarFire software or fabric logic. For more information about programming the on-board SPI flash using Fabric Logic, see Appendix 1: Programming On-board SPI Flash Using the Fabric Logic Through the Host Loader, page 37.

Before you begin:

1. Ensure that the device is programmed with the `progamming_appnote_only_FPGA_v1.job` file. See Appendix 2: Programming the Device and External SPI Flash Using FlashPro Express, page 39.
2. Connect the power supply cable to the J9 (Evaluation board) or J2 (Splash board) connector on the board.
3. Connect the USB cable from the host PC to FTDI port J5 (Evaluation board) or J1 (Splash board) on the board.
4. Ensure that on-board SW11 (Evaluation board) or SW8 (Splash board) DIP 1 is set to OFF.
5. Open pin 1 and 2 of the J23 jumper.
6. Power ON the Evaluation board using the SW3 or the Splash board using the SW1 slide switch.

After power-up, PuTTY displays the options as shown in the following figure. Observe the design version 01 loaded onto the device.

![Authentication and Programming Options](image)

At this point, the on-board SPI Flash device is empty. Hence, selecting Option 1 or 2 returns unsuccessful status codes as shown in the following figure.

![Authentication Error](image)

Selecting option 4, 5, or 6 does not initiate any program operation as the on-board SPI flash is empty.
5.1 Programming On-board SPI Flash Using Libero

Libero SoC Design Suite supports the on-board SPI Flash programming using JTAG. For more information about the SPI Flash programming modes, see UG0714: PolarFire FPGA Programming User Guide.

The external SPI flash can also be programmed through FlashPro Express software. See Appendix 2: Programming the Device and External SPI Flash Using FlashPro Express, page 39 for more information.

To optimize the SPI flash programming time, change the TCK frequency value under Configure Programmer as shown in the following figure.

**Figure 34 • Configure Programming Settings**

To program the SPI flash using JTAG, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 7, page 27 (for Evaluation board) and Table 8, page 28 (for Splash board).
2. On the Design Flow window, select Program Design and then double-click Configure Design Initialization Data and Memories.

**Figure 35 • Configure Design Initialization Data and Memories Option**

3. In the Design and Memory Initialization page, select SPI Flash tab, as shown in Figure 36, page 33.
4. In SPI Flash Clients pane, select Add SPI Bitstream Client to add the required programming images (.spi images), and click Apply. These images are provided at mpf_ac466_eval/splash_df/Libero_Project/hw_project/designer/top/export.

**Figure 36 • SPI Flash Tab**
5. Double-click **Generate SPI Flash Image** and double-click **Run PROGRAM_SPI_IMAGE Action** to get the SPI flash programmed with the programming images as shown in the following figure.

*Figure 37 • SPI Flash Programming*

At this point, the on-board SPI Flash device is programmed with the images.

### 5.2 Running Auto Update

To run auto update, perform the following steps:

1. Start the PuTTY and power-cycle the board. The auto update is initiated and update image (update_image_v2.spi) gets programmed into the device.

Observe the design version **02** as shown in the following figure.

*Figure 38 • Auto Update*

### 5.3 Running Authentication

To run bitstream authentication, perform the following steps:

1. Press 1 to initiate the bitstream authentication.

After successful authentication, PuTTY displays the status code as shown in the following figure.

*Figure 39 • Successful Bitstream Authentication*

2. Press 2 to initiate the IAP image authentication.

After successful authentication, PuTTY displays the status code, as shown in the following figure.

*Figure 40 • Successful IAP Image Authentication*

This concludes the bitstream and IAP image authentication.
5.4 Running Auto Programming

To run Auto programming, perform the following steps:

1. Press 3 in PuTTY. The PuTTY notifies to erase the device using FlashPro and power-cycle the board as shown in the following figure.

   ![Figure 41 • Notifying ERASE Action](image1)

   - Erase the PolarFire device using the FlashPro ERASE action.
   - Next Power cycle the board to Initiate AutoProgramming on Blank device.
   - Select option:
     1. **AUTHENTICATION**
     2. Bit-stream authentication
     3. IAP image authentication
     4. **DEVICE PROGRAMMING**
     5. Auto programming
     6. IAP Program by address
     7. Initiate Auto-Update

2. Using FlashPro, erase the device and power-cycle the board. All the LEDs stop glowing for a few seconds, which indicates that auto programming is in progress. The highest programming image version is selected from the first two available images in external SPI Flash for auto programming. In this case, it is version 2 (update_image_v2.spi). PuTTY displays the updated design version, as shown in the following figure.

   ![Figure 42 • Successful Auto Programming](image2)

   - Design Version (USB first): 00.02
   - USB USERCODE/Signature (USB first): 23606789
   - Select option:
     1. **AUTHENTICATION**
     2. Bit-stream authentication
     3. IAP image authentication
     4. **DEVICE PROGRAMMING**
     5. Auto programming
     6. IAP Program by address
     7. Initiate Auto-Update

This concludes running the Auto programming feature.
5.5 Running IAP

To run IAP, perform the following steps:

1. Press 4, IAP program by Index. After around 28 seconds, the IAP with image at index 2 is executed successfully and the design version 05 is displayed as shown in the following figure.

   **Figure 43** - Successful IAP at Index 2

   ![Successful IAP at Index 2]

2. Press 5, IAP program by address. After around 28 seconds, the IAP with the image at address 0x1400000 is executed successfully and the design version 05 is displayed as shown in the following figure.

   **Figure 44** - Successful IAP by Address

   ![Successful IAP by Address]

This concludes running the IAP feature.

For information about programming the on-board SPI flash using the fabric logic, see Appendix 1: Programming On-board SPI Flash Using the Fabric Logic Through the Host Loader, page 37.
Appendix 1: Programming On-board SPI Flash Using the Fabric Logic Through the Host Loader

To program the SPI flash, perform the following steps:

1. Power OFF the Evaluation board using the SW3 slide switch or the Splash board using the SW1 slide switch. Close the PuTTY and set the on-board SW11 (Evaluation board) or SW8 (Splash board) DIP 1 to ON.
2. Disconnect and connect the USB cable from the host PC to FTDI port J5 on the Evaluation board and J1 on the Splash board. This ensures clearing off UART buffers.
3. Power ON the Evaluation board using the SW3 or the Splash board using the SW1 slide switch.
4. Locate the load_spi_flash.bat batch file from the $DesignFiles_Folder\host_pc_tool_pf folder.
5. Right-click load_spi_flash.bat batch file and edit it as follows to match the COM port number. For example, COM Port 9 in this instance.
   
   spi_loader.exe 9 golden_image_v0.spi  update_image_v2.spi  iap_image_v5.spi

6. Double-click the load_spi_flash.bat file to load the programming images—listed in the following table—into external SPI flash. The application firmware writes the flash directory contents into the external SPI flash along with programming images.

The command window prompts to press enter to erase and program the SPI Flash with programming images.

The LED 4 blinks to indicate that the SPI Flash Erase operation is in progress. The command prompt displays the status as shown in the following figure.

Figure 45 • Erasing SPI Flash
7. The SPI Flash programming operation starts and takes 20-30 minutes to complete. LED 5 blinks to indicate that the SPI Flash programming operation is in progress. When the SPI Flash programming operation completes successfully, LED 5 starts to glow. The Command prompt shows the status and the time taken as shown in the following figure.

**Figure 46 • Command Prompt Status**

```
BEGIN transaction Ack 'b' is received from the target
Requested address from the target =9527296
Requested returns from the target =1296
bytes read from the file=1296
Remaining bytes =0
Sending the data to the target..................................................  
End of one transaction: Ack 'a' received from target for the data from the host

start time 22:54:23
end time 23:24:28

DONE press ctrl+c to terminate the application.
```

8. Close the application.
9. Set the on-board SW11 (Evaluation board) or SW8 (Splash board) DIP1 to OFF and open the PuTTY terminal. Power cycle the board to select the programming options.
Appendix 2: Programming the Device and External SPI Flash Using FlashPro Express

This section describes how to program the PolarFire device with the .job programming file using FlashPro Express. The .job files are available at the following design files folder location:

mpf_ac466_eval_df\Programming_Job or mpf_ac466_splash_df\Programming_Job

programming_appnote_FPGA_SPI_images_v1 contains both PolarFire device contents and SPI images. When the file is selected for programming, the FlashPro express software programs the PolarFire FPGA device and external SPI flash memory with programming images. The programming takes nearly 30 minutes to complete.

programming_appnote_only_FPGA_v1 contains only PolarFire device contents. When the file selected for programming, the FlashPro express software programs the PolarFire FPGA device only.

To program the device and external SPI flash, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 7, page 27 (for Evaluation board) and Table 8, page 28 (for Splash board).

   **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 Evaluation board or J2 connector on the Splash board.

3. Connect the USB cable from the host PC to the J5 (FTDI port) on the Evaluation board or J1 (FTDI port) on the Splash board.

4. Power ON the Evaluation board using the SW3 slide switch or the Splash board using the SW1 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 47** • 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_ac466_eval_df/Programming_Job`
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

   ![New Job Project from FlashPro Express Job](image)

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**.

   ![Programming the Device](image)
10. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure. See Running the Demo, page 32 to run the demo.

**Figure 50 • FlashPro Express—RUN PASSED**

11. Close **FlashPro Express** or in the **Project** tab, click **Exit**.
TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select Project > Execute Script....
3. Click Browse and select script.tcl from the downloaded TCL_Scripts directory.
4. Click Run.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to mpf_ac466_eval_df/TCL_Scripts/readme.txt and mpf_ac466_splash_df/TCL_Scripts/readme.txt.

Refer to Libero® SoC TCL Command Reference Guide for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.
Appendix 4: References

This section lists the documents that provide more information about programming and other IP cores used.

- For more information about PolarFire FPGA programming, see UG0714: PolarFire FPGA Programming User Guide.
- For more information about the CoreJTAGDEBUG IP core, see CoreJTAGDebug_HB.pdf.
- For more information about the MIV_RV32 IP core, see MIV_RV32 Handbook from the Libero SoC Catalog.
- For more information about the CoreUARTapb IP core, see CoreUARTapb_HB.pdf.
- For more information about the CoreAPB3 IP core, see CoreAPB3_HB.pdf.
- For more information about the CoreGPIO IP core, see CoreGPIO_HB.pdf.
- For more information about the PolarFire initialization monitor, see UG0725: PolarFire FPGA Device Power-Up and Resets User Guide.
- For more information about how to build a Mi-V processor subsystem for PolarFire devices, see TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial.
- For more information about the PF_CCC IP core, see UG0684: PolarFire FPGA Clocking Resources User Guide.
- For more information about Libero, ModelSim, and Synplify, see Microsemi Libero SoC PolarFire web page.