# Contents

1 Revision History ................................................. 1  
  1.1 Revision 4.0 ......................................................... 1  
  1.2 Revision 3.0 ......................................................... 1  
  1.3 Revision 2.0 ......................................................... 1  
  1.4 Revision 1.0 ......................................................... 1  
2 PolarFire FPGA Auto Update and In-Application Programming .......... 2  
  2.1 CoreSysServices_PF IP 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 Synthesize ........................................................ 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 .................................... 23  
  3.7 Generate Bitstream .................................................. 24  
  3.8 Programming the Device ............................................. 24  
    3.8.1 Programming the Device on the Evaluation Board ............. 25  
    3.8.2 Programming the Device on the Splash Board ............... 26  
4 Programming the Device Using FlashPro Software ....................... 27  
5 Serial Terminal Emulation Program Setup ................................ 28  
6 Running the Demo .................................................... 30  
  6.1 Programming On-board SPI Flash Using Libero .................... 30  
  6.2 Running Auto Update ................................................ 32  
  6.3 Running Authentication ............................................. 32  
  6.4 Running Auto Programming .......................................... 33  
  6.5 Running IAP ......................................................... 34  
7 Appendix: Programming On-board SPI Flash Using the Fabric Logic ... 35  
  7.1 Programming the SPI Flash Using Fabric Logic .................... 35  
8 Appendix: References ................................................ 37
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Core System Services IP Interfacing with Fabric User Logic</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Firmware catalog</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>PolarFire Programming Design Block Diagram</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Accessing On-board SPI Flash Using Fabric</td>
<td>7</td>
</tr>
<tr>
<td>Figure 5</td>
<td>SPI Flash Memory</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Top Level Libero Design</td>
<td>8</td>
</tr>
<tr>
<td>Figure 7</td>
<td>PF_INIT_MONITOR Configuration</td>
<td>9</td>
</tr>
<tr>
<td>Figure 8</td>
<td>PF_CCC_0 Input Clock Configuration</td>
<td>10</td>
</tr>
<tr>
<td>Figure 9</td>
<td>PF_CCC_0 Output Clock Configuration</td>
<td>10</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Mi-V Configuration</td>
<td>11</td>
</tr>
<tr>
<td>Figure 11</td>
<td>PF_SRAM_AHBL_AXI Configuration</td>
<td>11</td>
</tr>
<tr>
<td>Figure 12</td>
<td>CoreGPIO_0 Configuration</td>
<td>12</td>
</tr>
<tr>
<td>Figure 13</td>
<td>CoreSPI Configuration</td>
<td>13</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Memory Map</td>
<td>14</td>
</tr>
<tr>
<td>Figure 15</td>
<td>CoreAHBLite_0 Configuration</td>
<td>15</td>
</tr>
<tr>
<td>Figure 16</td>
<td>CoreAHBLite_1 Configuration</td>
<td>16</td>
</tr>
<tr>
<td>Figure 17</td>
<td>CoreAPB3_0 Configuration</td>
<td>17</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Clocking Structure</td>
<td>18</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Libero Design Flow Options</td>
<td>19</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Design and Memory Initialization</td>
<td>21</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Fabric RAMs Tab</td>
<td>22</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Edit Fabric RAM Initialization Client</td>
<td>22</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Apply Fabric RAM Content</td>
<td>23</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Configure Programming Options</td>
<td>23</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Generate Bitstream—Configure Bitstream Options</td>
<td>24</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Board Setup—Evaluation Kit</td>
<td>25</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Board Setup—Splash Kit</td>
<td>26</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Programming the Device using FlashPro Software</td>
<td>27</td>
</tr>
<tr>
<td>Figure 29</td>
<td>COM Port Number</td>
<td>28</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Select Serial as the Connection Type</td>
<td>28</td>
</tr>
<tr>
<td>Figure 31</td>
<td>PuTTY Configuration</td>
<td>29</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Configure Programming Settings</td>
<td>30</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Configure Design Initialization Data and Memories Option</td>
<td>30</td>
</tr>
<tr>
<td>Figure 34</td>
<td>SPI Flash Tab</td>
<td>31</td>
</tr>
<tr>
<td>Figure 35</td>
<td>SPI Flash Programming</td>
<td>31</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Authentication and Programming Options</td>
<td>31</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Auto Update</td>
<td>32</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Successful Bitstream Authentication</td>
<td>32</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Successful IAP Image Authentication</td>
<td>32</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Notifying ERASE Action</td>
<td>33</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Successful Auto Programming</td>
<td>33</td>
</tr>
<tr>
<td>Figure 42</td>
<td>Successful IAP at Index 2</td>
<td>34</td>
</tr>
<tr>
<td>Figure 43</td>
<td>Successful IAP by Address</td>
<td>34</td>
</tr>
<tr>
<td>Figure 44</td>
<td>Erasing SPI Flash</td>
<td>36</td>
</tr>
<tr>
<td>Figure 45</td>
<td>Command Prompt Status</td>
<td>36</td>
</tr>
</tbody>
</table>
Tables

<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>Jumper Settings—Evaluation Board</td>
<td>25</td>
</tr>
<tr>
<td>Table 7</td>
<td>Jumper Settings—Splash Board</td>
<td>26</td>
</tr>
<tr>
<td>Table 8</td>
<td>Programming Images</td>
<td>35</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 4.0
Merged Splash kit related content and updated the document for Libero SoC PolarFire v2.3 release.

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

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

1.4 Revision 1.0
The first publication of this document.
2 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 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 PolarFire software.

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

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

- **Using the pre-generated .stp file:** To program the device using the .stp file provided along with the design, see Programming the Device Using FlashPro Software, page 27.
- **Using Libero SoC PolarFire:** To program the device using Libero SoC PolarFire, see Libero Design Flow, page 19.

This design can be used as a reference to build a fabric design with programming features.
2.1 CoreSysServices_PF IP 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 CoreSysServices_PF IP for initiating the system services. A service request interrupt 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 specify 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 CoreSysServices_PF IP Interfaces with the fabric logic.

Figure 1 • Core System Services IP Interfacing with Fabric User Logic

The system services driver and the sample SoftConsole project are generated from Firmware Catalog as shown Figure 2, page 4.
In this design, the sample SoftConsole project is migrated to SoftConsole v5.3. 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 2 • Design Requirements

<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><strong>Hardware</strong></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-1FCG484EES)</td>
<td>Rev 2 or later</td>
</tr>
<tr>
<td>Host PC</td>
<td></td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>FlashPro</td>
<td>12.200.35.9</td>
</tr>
<tr>
<td>Libero SoC PolarFire Design Suite</td>
<td>2.3</td>
</tr>
<tr>
<td>SoftConsole</td>
<td>v5.3</td>
</tr>
<tr>
<td>Serial Terminal Emulation Program</td>
<td>PuTTY or HyperTerminal <a href="http://www.putty.org">www.putty.org</a></td>
</tr>
</tbody>
</table>

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

2.3 Prerequisites

Before you start:

1. Download the design files from the following location:
   For Evaluation kit:  
   http://soc.microsemi.com/download/rsc/?f=mpf_ac466_eval_liberosocpolarfirev2p3_df
   For Splash kit:  
   http://soc.microsemi.com/download/rsc/?f=mpf_ac466_splash_liberosocpolarfirev2p3_df
2. Download and install Libero SoC PolarFire v2.3 on the host PC from the following location.  
   https://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc-polarfire#downloads
   The latest versions of ModelSim and Synplify Pro are included in the Libero SoC PolarFire installation package.
2.4 Demon 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 CoreSysServices_PF IP 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 CoreSysServices_PF IP 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 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 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 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.

**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 including the LSRAM initialization. The following figure shows 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**

**Figure 9 • PF_CCC_0 Output Clock Configuration**
2.4.1.3 **Mi-V Soft Processor Configuration**

The Mi-V soft processor Reset Vector Address is set to 0x8000_0000 from default value 0x6000_0000. After device reset, the processor executes the application from LSRAM, which is mapped to 0x80000000. Hence, the Reset Vector Address is set to 0x80000000 as shown in the following figure.

In the Mi-V processor memory map, the 0x8000_0000 to 0x8FFF_FFFC range is defined for AHB memory interface and the 0x6000_0000 to 0x7FFF_FFFF range is defined for AHB I/O interface.

*Figure 10 • Mi-V Configuration*

2.4.1.4 **PF_SRAM_AHBL_AXI Configuration**

The PF_SRAM_AHBL_AXI IP is the main memory of the Mi-V processor, and it gets initialized with the user application from µPROM. It is connected to Mi-V soft processor as an AHB slave. LSRAM is configured for the following settings:

- **Optimize for**: By default, Low power is selected. It optimizes the LSRAM macro for low power. If design demands high speed memory access, High Speed can be selected.
- **Fabric Interface type**: By default, AHBLite is selected. The Mi-V soft processor is AHB based, so the SRAM is interfaced to the processor using AHB bus for code execution.
- **Memory depth**: This field is set to 65536 words to accommodate an application of up to 256 KB into LSRAM. 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. The following figure shows the PF_SRAM_AHBL_AXI (LSRAM_0) IP configuration.

*Figure 11 • PF_SRAM_AHBL_AXI Configuration*
2.4.1.5 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** is set to 32
  The design uses 32-bit data width for APB read and write data.

- **Number of I/Os** is 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*
2.4.1.6 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 is 32-bit. The default value is 8.
- **Mode**: select Motorola Mode (default) as the target SPI slave (VSC Phy) supports Motorola mode.
- **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*
2.4.1.7 Design Memory Map

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

*Figure 14 • Memory Map*
2.4.1.7.1  CoreAHBLite Configuration

Two instances of CoreAHBLite are used in this design. The following figures show the configurations of CoreAHBLite_0 and CoreAHBLite_1 IP cores. The CoreAHBLite_0 interfaces with the APB peripherals to the Mi-V processor at 0x6000_0000.

**Figure 15** • CoreAHBLite_0 Configuration
The CoreAHBLite_1 interfaces PF_SRAM with Mi-V soft processor for accessing the LSRAM at memory address 0x8000_0000. This configuration is required as the Mi-V processor executes the code from 0x8000_0000.

Figure 16 • CoreAHBLite_1 Configuration
2.4.1.7.2 CoreAPB3 Configuration

The CoreAPB3 IP connects the peripherals, CoreSysServices_PF, 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 CoreSysServices_PF, and Slot 3 for CoreSPI.

The following figure shows the CoreAPB3 configuration.

*Figure 17: CoreAPB3 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 an 100 MHz system clock for configuring the APB peripherals.

Figure 18 • Clocking Structure
The Libero design flow involves running the following processes in the Libero SoC PolarFire:

- Synthesize, 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 23
- Generate Bitstream, page 24
- Programming the Device, page 24

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

Figure 19 • Libero Design Flow Options
3.1 Synthesize

To synthesize the design:

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

Note: PROC_SUBSYSTEM.srr and the PROC_SUBSYSTEM_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 following constraint files in the Constraint Manager window:

- The io.pdc and the user.pdc file for the I/O assignments
- The PROC_SUBSYSTEMderived_constraints.sdc file for timing constraints
- JTAG_constraint.sdc file for creating the JTAG clock with 30 MHz frequency.
- The Async_Clock_groups.sdc file defines that the CCC_0 output clock and the JTAG clock as asynchronous clocks.

To Place and Route, double-click Place and Route from the Design Flow window.

When place and route is successful, a green tick mark appears next to Place and Route.

Note: The file, PROC_SUBSYSTEM_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 PROC_SUBSYSTEM_layout_log.log file in the Reports tab -> PROC_SUBSYSTEM 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 4 • Resource Utilization—Evaluation Board

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14462</td>
<td>299544</td>
<td>4.83</td>
</tr>
<tr>
<td>DFF</td>
<td>7488</td>
<td>299544</td>
<td>2.50</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>15215</td>
<td>299544</td>
<td>5.08</td>
</tr>
</tbody>
</table>

Table 5 • Resource Utilization—Splash Board

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14696</td>
<td>299544</td>
<td>4.91</td>
</tr>
<tr>
<td>DFF</td>
<td>7482</td>
<td>299544</td>
<td>2.50</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>15423</td>
<td>299544</td>
<td>5.15</td>
</tr>
</tbody>
</table>
3.3 Verify Timing

To verify timing:

1. Double-click Verify Timing from the Design Flow tab. When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 19, 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:

1. Double-click Generate FPGA Array Data from the Design Flow window.
2. A green tick mark is displayed after the successful generation of the FPGA array data as shown in Figure 19, page 19.

3.5 Configure Design Initialization Data and Memories

The Configure Design Initialization Data and Memories step generates the LSRAM 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 LSRAM initialization client is stored in the sNVM.

This process requires the user application executable file (hex file) to initialize the LSRAM blocks 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 LSRAM blocks.

Follow these steps:

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

*Figure 20* • Design and Memory Initialization
2. Select the Fabric RAMs tab and select the PF_SRAM client from the list and click Edit as shown in the following figure.

**Figure 21 • Fabric RAMs Tab**

![Fabric RAMs Tab](image)

3. In the Edit Fabric RAM Initialization Client dialog box, select the 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 22 • Edit Fabric RAM Initialization Client**

![Edit Fabric RAM Initialization Client](image)
4. Click **Apply** as shown in the following figure.

**Figure 23 • Apply Fabric RAM Content**

5. Click **Apply** in the Design Initialization tab.
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 19, page 19.

### 3.6 Configure Programming Options

The Design version and user code (Silicon signature) are configured in this step. Double click Design flow->Program and Debug Design->Configure Programming Options to give values as shown in the following figure.

**Figure 24 • Configure Programming Options**
3.7 **Generate Bitstream**

To generate the bitstream:

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

*Figure 25 • Generate Bitstream—Configure Bitstream Options*

2. Double-click **Generate Bitstream** from the **Design Flow** tab. When the bitstream is successfully generated, a green tick mark appears as shown in *Figure 19*, page 19.

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

3.8 **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
3.8.1 Programming the Device on the Evaluation Board

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

Follow these steps to program the PolarFire device:

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

   Table 6 • Jumper Settings—Evaluation Board

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J18, J19, J20, J21, 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 26 • Board Setup—Evaluation Kit


   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 19, page 19.
3.8.2 Programming the Device on the Splash Board

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

Follow these steps to program the PolarFire device:

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

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J5, J6, J7, J8, 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>
</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.


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 19, page 19.
This section describes how to program the PolarFire device with the .stp programming file using FlashPro. The .stp file is available at the following design files folder location:

`/mpf_ac466_eval/splash_liberosocpolarfirev2p3_df/Programming_File`

To program the PolarFire device using FlashPro, complete the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 6, page 25 (for Evaluation board) and Table 7, page 26 (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 SW1 slide switch.

5. On the host PC, launch the FlashPro software.

6. Click **New Project** to create a new project.

   In the New Project window, enter a project name.

7. Click **Browse** and navigate to the location where you want to save the project.

8. Select **Single device** as the programming mode and click **OK** to save the project.

9. Click **Configure Device**.

10. Click **Browse**, and select the `programming_appnote_v1.stp` file from the following folder:

    `/<design file directory>/mpf_ac466_eval/splash_liberosocpolarfirev2p3_df/Programming_File`

**Figure 28 • Programming the Device using FlashPro Software**

11. Click **Open**. The required programming file is selected and ready to be programmed in the device.

12. Click **PROGRAM** to program the device.

When the device is programmed successfully, a **Run PASSED** status is displayed.
5 Serial Terminal Emulation Program Setup

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 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](image)

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](image)

7. Set the Serial line to connect to COM port number noted in Step 3.
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 30.
6 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 28.

6.1 Programming On-board SPI Flash Using Libero

Libero SoC PolarFire 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.

To optimize the SPI flash programming time, change the TCK frequency value under Configure Programmer as shown in the following figure. The default value is 4 MHz.

Figure 32 • Configure Programming Settings

To program the SPI flash using JTAG:

1. Ensure that the jumper settings on the board are the same as those listed in Table 6, page 25 (for Evaluation board) and Table 7, page 26 (for Splash board).
2. In the Design Flow window, select Program and Debug Design and then double-click Configure Design Initialization Data and Memories.

Figure 33 • Configure Design Initialization Data and Memories Option

3. In the Design and Memory Initialization page, select the SPI Flash tab, as shown in Figure 34, page 31.
4. In SPI Flash Clients pane, add the required programming images (.spi images), and click Apply. These images are provided at mpf_ac466_eval/splash_liberosocpolarfirev2p3_dflibero_Project\hw_project\designer\PROC_SUB\SYSTEM\export.
5. Connect the power supply cable to the J9 (Evaluation board) or J2 (Splash board) connector on the board.

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

7. 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 35 • SPI Flash Programming**

8. Power-cycle the board once you program the device.

   **Note:** If you program the external SPI flash using Libero, set the on-board SW11 (Evaluation board) or SW8 (Splash board) DIP 1 to **ON** because the fabric design is not required to program the SPI flash.

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

**Figure 36 • Authentication and Programming Options**

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

This concludes the on-board SPI Flash Programming.
6.2 Running Auto Update

To run auto update:

1. Set the on-board SW11 (Evaluation board) or SW8 (Splash board) DIP 1 to OFF.
2. 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.

![Auto Update](image)

6.3 Running Authentication

To run bitstream authentication:

1. Press 1 to initiate the bitstream authentication.

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

![Successful Bitstream Authentication](image)

2. Press 2 to initiate the IAP image authentication.

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

![Successful IAP Image Authentication](image)

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

To run Auto programming:

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 40 • Notifying ERASE Action**

   ![Notifying ERASE Action](image)

   2. Using FlashPro, erase the device and power-cycle the board. All the LEDs stop glowing for few seconds, which indicates that the auto programming is in progress. The highest programming image version is selected from 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 41 • Successful Auto Programming**

   ![Successful Auto Programming](image)

   This concludes running the Auto programming feature.
6.5 Running IAP

To run IAP:

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 42 • Successful IAP at Index 2

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

Figure 43 • 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: Programming On-board SPI Flash Using the Fabric Logic, page 35.
Appendix: Programming On-board SPI Flash Using the Fabric Logic

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 PolarFire software.

Before you start:
1. Ensure that the device is programmed with the `programming_appnote_v1.stp` file.
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 J5 (FTDI port) on the Evaluation board or J1 (FTDI port) on the Splash 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 (for Evaluation board).
6. Power ON the Evaluation board using the SW3 or the Splash board using SW1 slide switch.

### 7.1 Programming the SPI Flash Using Fabric Logic

To program the SPI flash:

1. Power OFF the Evaluation board using the SW3 slide switch or the Splash board using 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 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.

#### Table 8 • 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>0x1234567</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>0xA000000</td>
</tr>
<tr>
<td>iap_image_v5.spi</td>
<td>5</td>
<td>0x56789ABC</td>
<td>2</td>
<td>0x1400000</td>
</tr>
</tbody>
</table>

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 44 • 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 45 • Command Prompt Status**

```
BEGIN transaction
Ack 'b' is received from the target
Requested address from the target = 0527296
Requested return bytes from the target = 1196
bytes read from the file = 1296
remaining bytes = 0
Sending the data to the target
End of one transaction
```

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

DONE press ctrl-c to terminate the application.

8. Close the application.
8 Appendix: References

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

- For more information about PolarFire FPGA programming, see the UG0714: PolarFire FPGA Programming User Guide.
- For more information about the CoreJTAGDEBUG IP core, see CoreJTAGDebug_HB.pdf from Libero->Catalog.
- For more information about the CoreAHBtoAPB3 IP core, see CoreAHBtoAPB3_HB.pdf.
- For more information about the CoreUARTapb IP core, see CoreUARTapb_HB.pdf.
- For more information about the CoreAHBLite IP core, see CoreAHBLite_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 the SRAM buffer, see UG0680: PolarFire FPGA Fabric User Guide.
- For more information about Libero, ModelSim, and Synplify, see the Microsemi Libero SoC PolarFire web page.