## Contents

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

2 PolarFire FPGA System Services ............................................... 2
   2.1 PF_SYSTEM_SERVICES Core Overview ................................. 2
   2.2 Design Requirements ..................................................... 4
   2.3 Prerequisites ................................................................ 5
   2.4 Demo Design ................................................................ 5
     2.4.1 Design Implementation .............................................. 6
   2.5 Clocking Structure .......................................................... 11

3 Libero Design Flow ................................................................. 12
   3.1 Synthesize .................................................................. 12
   3.2 Place and Route ........................................................... 13
     3.2.1 Resource Utilization .................................................. 13
   3.3 Verify Timing ............................................................... 13
   3.4 Generate FPGA Array Data ............................................. 14
   3.5 Configure Design Initialization Data and Memories ............. 14
   3.6 Configure Programming Options ..................................... 17
   3.7 Generate Bitstream ....................................................... 17
   3.8 Run PROGRAM Action .................................................. 18
   3.9 Export FlashPro Express Job .......................................... 20

4 Setting up the Serial Terminal Program - PuTTY .......................... 21

5 Running the Demo ................................................................. 23

6 Appendix 1: Programming the Device Using FlashPro Express .... 29

7 Appendix 2: Device Certificate Information .................................. 32

8 Appendix 3: Query Security .................................................... 34

9 Appendix 4: Debug Information .............................................. 35

10 Appendix 5: Digest Information .............................................. 36

11 Appendix 6: Running the TCL Script ....................................... 37

12 Appendix 7: References ......................................................... 38
Tables

Table 1 System Services in the Demo ................................................................. 2
Table 2 System Services Descriptor ................................................................. 3
Table 3 Design Requirements ............................................................................ 4
Table 4 I/O Signals ............................................................................................ 6
Table 5 Resource Utilization—Evaluation Kit .................................................. 13
Table 6 Resource Utilization—Splash Kit ......................................................... 13
Table 7 Jumper Settings for PolarFire Device Programming—Evaluation Kit .... 18
Table 8 Jumper Settings for PolarFire Device Programming—Splash Kit .......... 18
Table 9 Device Certificate Fields (1024 bytes) .................................................. 32
Table 10 Security Locks Fields ........................................................................ 34
Table 11 Debug Info Fields .............................................................................. 35
Table 12 Digest Information Bit Fields ............................................................. 36
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 8.0
The following is a summary of the changes made in this revision.
• Replaced text CoreSysServices_PF with PF_SYSTEM_SERVICES.
• Updated Figure 1, page 3, Figure 3, page 5, and Figure 11, page 11.
• Replaced Figure 4, page 6, Figure 10, page 10, and Figure 25, page 23 through Figure 36, page 28.

1.2 Revision 7.0
Added Appendix 6: Running the TCL Script, page 37.

1.3 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.

1.4 Revision 5.0
Updated the document for Libero SoC v12.0.

1.5 Revision 4.0
The following is a summary of the changes made in this revision.
• Updated for Libero® SoC PolarFire v2.3.
• Merged SPLASH kit related content.

1.6 Revision 3.0
Updated the document for Libero SoC PolarFire v2.2.

1.7 Revision 2.0
Updated the document for Libero SoC PolarFire v2.1.

1.8 Revision 1.0
The first publication of this document.
2 PolarFire FPGA System Services

System services are System Controller actions initiated from the FPGA design using the PF_SYSTEM_SERVICES core. The System Controller hard block in PolarFire® FPGAs provide various system services. The PF_SYSTEM_SERVICES core issues service requests to the System Controller and fetches the relevant data.

This document describes how to run the system services listed in the following table using the demo design.

<table>
<thead>
<tr>
<th>Service Category</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device and Data Services</td>
<td>Read Device Serial Number</td>
</tr>
<tr>
<td></td>
<td>Read Device User-code</td>
</tr>
<tr>
<td></td>
<td>Read Device Design-info</td>
</tr>
<tr>
<td>Design and Data Security Services</td>
<td>Read Device Certificate</td>
</tr>
<tr>
<td></td>
<td>Read Digest</td>
</tr>
<tr>
<td></td>
<td>Query security</td>
</tr>
<tr>
<td></td>
<td>Read Debug Information</td>
</tr>
<tr>
<td></td>
<td>Digital signature</td>
</tr>
<tr>
<td></td>
<td>Secure NVM services</td>
</tr>
<tr>
<td></td>
<td>PUF Emulation</td>
</tr>
<tr>
<td></td>
<td>Nonce service</td>
</tr>
</tbody>
</table>

The demo design includes the Mi-V soft processor, which initiates the system service requests and enables the PF_SYSTEM_SERVICES core to access the System Controller. For more information about the system services design implementation, and the necessary blocks and IP cores instantiated in Libero SoC, see, Demo Design, page 5.

The demo 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 demo design, see Appendix 1: Programming the Device Using FlashPro Express, page 29.
- **Using Libero SoC**: To program the device using Libero SoC, see Libero Design Flow, page 12.

The demo design can be used as a reference to build a fabric design with the system services feature.

2.1 PF_SYSTEM_SERVICES Core 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 core 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 2 • System Services Descriptor**

<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.

**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 demo, the sample SoftConsole project is migrated to SoftConsole and the application file `main.c` is modified to provide the user options.
2.2 Design Requirements

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

<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</td>
<td>Rev D or later</td>
</tr>
<tr>
<td>MPF300T-IFCG1152E</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>PolarFire Splash Kit</td>
<td>Rev 2 or later</td>
</tr>
<tr>
<td>MPF300T-1FCG484E</td>
<td></td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>FlashPro Express</td>
<td></td>
</tr>
<tr>
<td>Libero SoC Design Suite</td>
<td>Note: Refer to the readme.txt file provided in the design files for the software versions used with this reference design.</td>
</tr>
<tr>
<td>SoftConsole</td>
<td></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 demo.

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. 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

2. For demo design files download link:
   • For Evaluation Kit
     http://soc.microsemi.com/download/rsc/?f=mpf_dg0798_eval_df
   • For Splash Kit
     http://soc.microsemi.com/download/rsc/?f=mpf_dg0798_splash_df

   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 demo 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 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 system services design.

Figure 3 • System Services Design Block Diagram
2.4.1 Design Implementation

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

Figure 4 • Top Level Libero Design

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

Table 4 • 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 onboard 50 MHz oscillator</td>
</tr>
<tr>
<td>resetn</td>
<td>Onboard 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>Onboard LED outputs</td>
</tr>
</tbody>
</table>
2.4.1.1 PF_CCC_0 Configuration

The PolarFire Clock Conditioning Circuitry (CCC) block takes an input clock of 50 MHz from the onboard 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 5 • PF_CCC_0 Input Clock Configuration

![PF_CCC_0 Input Clock Configuration](image1)

The following figure shows the PF_CCC_0 output clock configuration. The Mi-V processor supports up to 120 MHz. This design uses a 100 MHz system clock for configuring the APB peripherals.

Figure 6 • PF_CCC_0 Output Clock Configuration

![PF_CCC_0 Output Clock Configuration](image2)
2.4.1.2 Mi-V Soft Processor Configuration

The Mi-V soft processor default Reset Vector Address 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 Figure 7, page 8.

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

In the Mi-V processor memory map, the 0x8000_0000 to 0x8000_FFFF range is defined for TCM memory interface and the 0x6000_0000 to 0x6FFF_FFFF range is defined for APB3 I/O interface.

Figure 7 • Mi-V RV32 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 less than 50 KB so this can fit into either sNVM or μPROM. In this demo, μPROM is selected as a data storage client as shown in the following figure.
2.4.1.3 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 a 32-bit data width for APB read and write data.
- **Number of I/Os** is set to 4
  The design controls 2 onboard 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, the first two I/Os are configured as an output and the last two I/Os are configured as an input.

*Note*: The first two I/Os configured as output are used by the design and the last two I/Os are not used. The I/Os are interfaced with on-board LEDs to control the LED states.

- **Interrupt Type**: Disabled
  When I/O states change, no interrupt is required for the application as these are used for only LEDs.
The following figure shows the CoreGPIO_0 configuration.

**Figure 9 • CoreGPIO_0 Configuration**

![CoreGPIO Configurator](image)

### 2.4.1.4 Design Memory Map

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

**Figure 10 • Memory Map**

![Memory Map](image)

### 2.4.1.5 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 a 32-bit data width for APB read and write data.
- **Number of address bits driven by the 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.
2.5 Clocking Structure

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

*Figure 11 • Clocking Structure*
3 Libero Design Flow

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

- **Synthesize**, page 12
- **Place and Route**, page 13
- **Verify Timing**, page 13
- **Generate FPGA Array Data**, page 14
- **Configure Design Initialization Data and Memories**, page 14
- **Configure Programming Options**, page 17
- **Generate Bitstream**, page 17
- **Run PROGRAM Action**, page 18
- **Export FlashPro Express Job**, page 20

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

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

**Figure 12** Libero Design Flow Options

### 3.1 Synthesize

To synthesize the design, perform the following steps:

   
   When the synthesis is successful, a green tick mark appears as shown in Figure 12, page 12.

2. Right-click Synthesize and select View Report to view the synthesis report and log files in the Reports tab.
3.2 Place and Route

The Place and Route process requires the I/O, timing, and floor planner constraints. The demo 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_constant.sdc file for timing constraints

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, 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. The following table lists the resource utilization of the evaluation kit.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14207</td>
<td>299544</td>
<td>4.74</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>1536</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Element</td>
<td>15187</td>
<td>299544</td>
<td>5.07</td>
</tr>
</tbody>
</table>

The following table lists the resource utilization splash kit.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14597</td>
<td>299544</td>
<td>4.87</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>732</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Element</td>
<td>15556</td>
<td>299544</td>
<td>5.19</td>
</tr>
</tbody>
</table>

3.3 Verify Timing

To verify timing, perform the following steps:

   When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 12, page 12.

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 12, page 12.

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 the demo, the TCM initialization client is stored in the μPROM.

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

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 13 • Design and Memory Initialization

2. Select the Fabric RAMs tab, select the tcm_ram client from the list, and click Edit as shown in the following figure.
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 folder and Click OK as shown in the following figure.

Figure 15 • Edit Fabric RAM Initialization Client
4. Click **Apply** as shown in the following figure.

*Figure 16 • Apply Fabric RAM Content*

5. Select **sNVM** tab -> Select **Add** from the list -> click **Add PlainText NonAuthenticated Client** as shown in the following figure:

*Figure 17 • Add PlainText NonAuthenticated Client Option*

6. In the preceding step select a client and click **Edit**.
7. In the **Edit PlainText NonAuthenticated client** dialog box, select **Content filled with 0's** option and provide the **Number of bytes**. Click **OK** as shown in the following figure.

*Figure 18 • Edit PlainText NonAuthenticated Client*

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

9. On the **Design Flow** window, click **Generate 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 12*, page 12.

### 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.

*Figure 19 • Configure Programming Options*

### 3.7 Generate Bitstream

To generate the bitstream, perform the following steps:

1. On the **Design Flow window**, double-click **Generate Bitstream**. When the bitstream is successfully generated, a green tick mark appears as shown in *Figure 12*, page 12

2. Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.
3.8 Run PROGRAM Action

After generating the bitstream, the PolarFire device must be programmed with the system services design.

To program the PolarFire device, perform the following steps:
1. Ensure that the following jumper settings are set on the evaluation board.

| Table 7 • Jumper Settings for PolarFire Device Programming—Evaluation Kit |
|-----------------------------|--------------------------------------------------|
| Jumper                      | Description                                      |
| J18, J19, J20, J21, and J22 | Close pin 2 and 3 for programming the PolarFire FPGA through FTDI |
| J28                         | Close pin 1 and 2 for programming through the on-boardFlashPro5 |
| J4                          | Close pin 1 and 2 for manual power switching using SW3 |
| J12                         | Close pin 3 and 4 for 2.5 V                       |

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

| Table 8 • Jumper Settings for PolarFire Device Programming—Splash Kit |
|-----------------------------|--------------------------------------------------|
| Jumper                      | Description                                      |
| J5, J6, J7, J8, and J9     | Close pin 2 and 3 for programming the PolarFire FPGA through FTDI |
| J11                         | Close pin 1 and 2 for programming through FTDI chip |
| J10                         | Close pin 1 and 2 for programming through FTDI SPI |
| J4                          | Close pin 1 and 2 for manual power switching using SW1 |
| J3                          | Open pin 1 and 2 for 1.0 V                        |

2. Connect the power supply cable to the connector J9 on Evaluation board or J2 on Splash board.
3. Connect the USB cable from the host PC to the J5 on Evaluation board or J1 on Splash board (FTDI port).
4. Power on the board using the slide switch SW3 on Evaluation board or SW1 on Splash board.
The following figure shows the board setup of evaluation kit.

*Figure 20 • Board Setup—Evaluation Kit*

The device is successfully programmed and the onboard LEDs 7, 8, 9, 10, and 11 glow. A green tick mark appears next to Run PROGRAM Action as shown in *Figure 12*, page 12.

*Figure 21*, page 20 shows the board setup of splash kit.
3.9 Export FlashPro Express Job

On the Design Flow window, double-click Export FlashPro Express Job. When the job file is successfully generated, a green tick mark appears as shown in Figure 12, page 12.

Figure 21 • Board Setup—Splash Kit

5. On the Design Flow window, double-click Run PROGRAM Action. The device is successfully programmed and the onboard LEDs 4, 5, 6, 7, and 8 glow. A green tick mark appears next to Run PROGRAM Action as shown in Figure 12, page 12.

To program the device using the .job file provided along with the demo design, see Appendix 1: Programming the Device Using FlashPro Express, page 29.
4  Setting up the Serial Terminal Program - PuTTY

The user application receives system service 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 port on the Splash board.
2. Connect the power supply cable to the J9 connector on the Evaluation board or J2 on Splash board.
3. Power on the board using the SW3 slide switch for Evaluation board or SW1 slide switch for the Splash board.
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 5 (COM5) is selected as shown in the following figure. COM Port-numbers may vary.

Figure 22 • Finding the COM Port

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

Figure 23 • 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.

*Figure 24 • PuTTY Configuration*

9. Set the **Flow control** to **None** as shown in the preceding figure and click **Open**.

PuTTY opens successfully, and this completes the serial terminal emulation program setup. See *Running the Demo*, page 23.
5 Running the Demo

This chapter describes how to run the system services demo using the serial terminal program (PuTTY). The prerequisite for the following procedure is to program the device and to set up the serial terminal. For more information on setting up the serial terminal, see Setting up the Serial Terminal Program - PuTTY, page 21.

To run the demo, perform the following steps:

1. Power on the board using the SW3 slide switch.
   System services options are displayed on the PuTTY as shown in the following figure.

2. Enter 1 to select **Read Device Serial number**.
   The 128-bit Device Serial Number (DSN) is displayed as shown in the following figure.

3. Enter 2 to select **Read Device User-code**.
   The 32-bit device USERCODE/Silicon signature is displayed as shown in the following figure.

---

**Figure 25 • System Services Options**

<table>
<thead>
<tr>
<th>Select Service:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Read Device Serial number</td>
</tr>
<tr>
<td>2. Read Device User-code</td>
</tr>
<tr>
<td>3. Read Device Design-info</td>
</tr>
<tr>
<td>4. Read Device certificate</td>
</tr>
<tr>
<td>5. Read Digest</td>
</tr>
<tr>
<td>6. Query security</td>
</tr>
<tr>
<td>7. Read debug information</td>
</tr>
<tr>
<td>8. Digital signature</td>
</tr>
<tr>
<td>9. Secure NUM services</td>
</tr>
<tr>
<td>a. PUF Emulation</td>
</tr>
<tr>
<td>b. None</td>
</tr>
</tbody>
</table>

---

**Figure 26 • Device Serial Number**

Device serial number: 2783F8138C7F3CDBCB1A4AB3059C8AB5

Each PolarFire FPGA device has a unique, publicly readable, 128-bit DSN. The DSN can be used in cryptographic protocols to uniquely identify the device. The DSN comprises two 64-bit fields: FSN and SNM.

Factory Serial Number (FSN)—The first (most significant) field is the FSN. It is a pseudo-random per-device unique value assigned during Microsemi’s manufacturing test and persists for the lifetime of the device.

Serial Number Modifier (SNM)—The second component is the SNM. It is initialized during the factory test and is destroyed during the recoverable zeroization action. If the device is subsequently recovered, a new SNM is assigned such that each SNM generated for a given FSN, is unique.

**Note:** The system services main menu is displayed after the execution of any of the options.

---

**Figure 27 • Device User-code**

32-bit USERCODE/Silicon signature (MSB first): 89ABCD

This can be configured from Design flow->Program and Debug Design->Configure Programming Options.
4. Enter 3 to select **Read Device Design-info**.

As shown in Figure 28, page 24, the device design information consists of:

- **256-bit user-defined Design ID**
- **16-bit design version**
  - This can be configured from Design flow->Program and Debug Design->Configure Programming Options. In auto update programming, the current design version is compared with the available images in external SPI flash to initiate the auto update on power up.
- **16-bit design back-level**
  - This can be configured from Design flow->Program and Debug Design->Configure Security. When back level protection is enabled, the device can only be programmed if the target design version is more than the back level value.

---

**Figure 28** • Device Design Information

```
Design ID: 6F44746F700000000000000000000000
00000000000000000000000000000000
Design Version: 0100
Design Back-Level: 0000
```

---
Running the Demo

5. Enter 4 to select **Read Device certificate**.

The device supply chain assurance certificate is displayed as shown in **Figure 29**, page 25.

For more information about decoding the device certificate, see **Appendix 2: Device Certificate Information**, page 32.

**Figure 29 • Device Certificate**
6. Enter 5 to select **Read Digest**.
The 416 byte Digest contains the fabric digest, sNVM digest, and user key digests. The Digest protects data integrity. The following figure shows the 416 byte digest displayed. For more information about decoding the digest output, see Appendix 5: Digest Information, page 36.

**Figure 30 • Digest**

```
Read Digest:
4AF13BCD5DE76E8BC0BE63F93CD0FD40B
290308AC4B3F9184499D023E57B5C2
DF42EB1347CC2F71415E777D5B0F7984
56F42F65B26A2F61DDFE70B28F74F90
CCDE013695FAB6B66F6999D3A5BBFB8
53B226C3BEBE828EB40CE67F5EFBF063
22316EF913839429F186AC19F222D712
2212B305AC0E072596B9F919F34BC
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
2E9A7B9198D1638007400CD2C3BEF1CC
745BB64B761108EBC521800C6452D4
F5A5FD42016A20003279EF6B630977B
43003D232009F0E8A98310A92759FB4D
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
2E9A7B9198D1638007400CD2C3BEF1CC
745BB64B761108EBC521800C6452D4
F5A5FD42016A20003279EF6B630977B
43003D232009F0E8A98310A92759FB4D
93BB6A72F64DE6EBC5E8EB8F991CA
9D7B5DDFED02F2DF6195F4F8C818549
68F6EAE10355BBD76F735971A0803CF
2598CD4EEBB80833049D3DE4C0F32
```

7. Enter 6 to select **Query Security**.
The non-volatile states of user security locks are displayed as shown in the following figure.

**Figure 31 • Security Locks Information**

```
Security locks: 00000000000000000000000000000000
```

The design does not include any security settings for device safety. Security locks can be configured from Design flow->Program and Debug Design->Configure Security. For more information about security locks, see Appendix 3: Query Security, page 34.

**Note:** If security locks are not handled properly, the device can go into the locked state.

8. Enter 7 to select **Read debug information**.
The debug information is displayed as shown in the following figure.

**Figure 32 • Debug Information**

```
Debug info:
0000000228080220020300C809BC093C09
1507402CC190000000000000000000000
0102D420194FC1140E77F000000000000
000128F50000000000000000000000000
00DEB1369FB6BB000000000000
```

In Figure 32, page 26, the highlighted 4 bytes E7000000 (LSB first) indicates the number of times (in this example, 0x00000000), the device was programmed (programming cycles). For more information about the Debug Info fields, see Appendix 4: Debug Information, page 35.
9. Enter 8 to select **Digital Signature**.

The digital signature in both Raw and DER formats are displayed in the following figure.

**Figure 33 • Digital Signature**

Digital Signature service:
48 byte hash value:
911601143DB01D7E2676B0434FP45D7BB
562415CBF08B56F685E582D7B038882F9
911601143DB01D7E2676B0434FP45D7BB

Raw format:
Digital Signature service successful.
Output Digital Signature - Raw format:
2B04CA5762DE3F6F5E10EBBE4BBD03
895DB43244AC2BCC51A4475ED95B007
73F5BB94B28864B51B76F1E08A383E31
AC3D1985F185FEC0322C068552F2197E
8F74C008B198ECA22378635F4F458
51FP5A39083752CD1F6123DE4699AD

DER format:
Digital Signature service successful.
Output Digital Signature - DER format:
366623109B0B49180B10B05299054D
EB43C5D3BE29EF631BD7421D57C728A
F7E66A9F292E1F939D6C61BCE69AFCB
B9F31A1D92310299F669907511A4A4E
B3C62C1E1B1E3F002F7EEFC31827E32
7A786F38F3B2301945A9D8027AD90926C
21AB0B6859F1D978

The digital signature service takes a user-supplied SHA384 hash and signs it with the device's private key. The application randomly generates the SHA384 hash value. The Digital Signature service sends the hash value to the System Controller. The Athena core runs the Elliptic Curve Digital Signature Algorithm (ECDSA) using the hash and the device's private key to generate the signature.
10. Enter 9 to select Secure NVM.
When the sNVM page/module address is entered (in this case, 0), the randomly generated 60 byte data is written to the specified sNVM page and read back, as shown in Figure 34, page 28.

Figure 34 • Secure NVM Services

```
<table>
<thead>
<tr>
<th>Secure NUM (sNUM) Functions</th>
<th>Non-authenticated plain-text</th>
</tr>
</thead>
</table>
```

Enter the sNUM page/module address (1 HEX byte. Page value range is from 0x01 to 0x83) and Press Enter key!

```
1
```

Input Data (60 Byte):
554D9936446F48C97E9A95C1124B9747
E55699564D9036446F48C97E9A95C1
124B9747E55E699A554D9036446F48C9
7E9A95C1124B9747E55E699A

Data read from sNUM region:
554D9936446F48C97E9A95C1124B9747
E55699564D9036446F48C97E9A95C1
124B9747E55E699A554D9036446F48C9
7E9A95C1124B9747E55E699A

Secure NUM write successful.

11. Enter ‘a’ (without quotes) to select PUF Emulation service

The PUF emulation service provides a mechanism for authenticating a device, or for generating a pseudo random bit strings that can be used for different purposes. When this service is selected, the service by default accepts a 128-bit challenge and an 8-bit optype, and returns a 256-bit response unique to the challenge and the optype as shown in the following figure.

Figure 35 • PUF Emulation Service

```
The challenge OPTYPE range (0x0 to 0xFF): 85
```

16 byte challenge:
85A8BC1E77A041F7F009C31B6F1CD10
PUF emulation service successful. Generated Response:
7120E8B7136CCE385D82E800742E4A
D0EE06174D1A99D457B73499662DA1

12. Enter ‘b’ to select Nonce service.

The 32 byte nonce value is displayed as shown in the following figure. The nonce service provides the ability to strengthen the Deterministic Random Bit Generator (DRBG) of the Athena by providing an alternate entropy source to use as additional seed data in its DRBG functions.

Figure 36 • Generated Nonce

```
```

Generated Nonce:
FF5249B99411CFB36AF3BE3097E3C37
E166EF46E82888916CE964BA96C43

```
6 Appendix 1: Programming the Device Using FlashPro Express

This chapter describes how to program the PolarFire device with the Job programming file using a FlashPro programmer. The default location of the Job file are located at following location:

```
mpf_dg0798_eval_df\Programming_Job\top.job
```

and

```
mpf_dg0798_splash_df\Programming_Job\top.job
```

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

1. Ensure that the jumper settings on the board are the same as listed in Table 8, page 18 and Table 7, page 18.
   
   **Note:** The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to the J5 (FTDI port) on the board.
4. Power on the board using the SW3 slide switch.
5. On the host PC, launch the FlashPro Express software.
6. To create a new job project, click **New** or

   In the **Project** menu, select **New Job Project from FlashPro Express Job**, as shown in the following figure.

   **Figure 37 • 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_dg0798_eval_df\Programming_Job`
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.
Figure 38 • New Job Project from FlashPro Express Job

8. Click OK. The required programming file is selected and ready to be programmed in the device.
9. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click Refresh/Rescan Programmers.

Figure 39 • Programming the Device
10. Click **RUN**. When the device is programmed successfully, a RUN PASSED status is displayed as shown in the following figure.

*Figure 40* • **FlashPro Express—RUN PASSED**

![FlashPro Express—RUN PASSED](image)

11. Close **FlashPro Express** or in the Project tab, click **Exit**.
Appendix 2: Device Certificate Information

The Device Certificate is a 1024 byte Microsemi-signed X-509 certificate programmed during manufacturing. The certificate is used to guarantee the authenticity of a device and its characteristics.

The following table lists the main fields of the device certificate.

<table>
<thead>
<tr>
<th>Offset (Byte)</th>
<th>Length (Bytes)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>854</td>
<td>Signed region of certificate</td>
</tr>
<tr>
<td>234</td>
<td>120</td>
<td>Device Public Key</td>
</tr>
<tr>
<td>368</td>
<td>16</td>
<td>DSN</td>
</tr>
</tbody>
</table>

The device certificate is encoded in the ASN.1 format. To view the content, the certificate must be decoded to a user readable format using the online JAVA tool: [http://lapo.it/asn1js/#](http://lapo.it/asn1js/#).

For decoding a certificate, perform the following steps:

1. Right click PuTTY, select **Copy All to Clipboard**, and paste the same to notepad as shown in the following figure.

   **Figure 41 • Copy Device Certificate**

2. Copy the 1024 bytes of device certificate from notepad to ASN.1 decoder as shown in the following figure and click **decode** button.

   **Note:** The sample device certificate (1024 bytes) is provided at:
   - For Evaluation kit `mpf_dg0798_eval_df\Device_Certificate\sample.txt`
   - For Splash kit `mpf_dg0798_splash_df\Device_Certificate\sample.txt`
Appendix 2: Device Certificate Information

**Figure 42 • Certificate Decoding Using Java Script**

3. The web page displays all the fields in certificate as shown in the following figure.

**Figure 43 • Decoded Certificate**
The following table lists each security lock bit and its features.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit</th>
<th>Lock</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>UL_DEBUG</td>
<td>Debug instructions disabled</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>UL_SNVM_DEBUG</td>
<td>sNVM debug disabled</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>UL_LIVEPROBE</td>
<td>Live probes disabled</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>UL_JTAG_DISABLE</td>
<td>User JTAG interface disabled</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>JTAG_BS_DISABLE</td>
<td>JTAG boundary scan disabled</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>UL_TVSD_MONITOR</td>
<td>External access to system Temperature and Voltage Sensor (TVS) disabled</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>JTAG_MONITOR</td>
<td>JTAG fabric monitor enabled</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>JTAG_TAP</td>
<td>JTAG TAP disabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>UL_PLAINTEXT</td>
<td>Plain text passcode unlock disabled</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>UL_FAB_PROTECT</td>
<td>Fabric erase/write disabled</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>UL_EXT_DIGEST</td>
<td>External digest check disable</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>UL_VERSION</td>
<td>Replay protection enabled</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>UL_FACT_UNLOCK</td>
<td>Factory test disabled</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>UL_IAP</td>
<td>IAP disabled</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>UL_EXT_ZEROIZE</td>
<td>External zeroization disabled</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>SPI_SLAVE_DISABLE</td>
<td>SPI port disabled</td>
</tr>
<tr>
<td>2-8</td>
<td></td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
## Appendix 4: Debug Information

The following table lists the debug information bit fields.

**Table 11 • Debug Info Fields**

<table>
<thead>
<tr>
<th>Byte Offset</th>
<th>Size (Bytes)</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>TOOL_INFO</td>
<td>• Reflects the TOOL_INFO passed in during ISC_ENABLE prior to programming</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IAP sets this to 0</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>TOOL_TYPE</td>
<td>Tool type used to program device 1 = JTAG, 2 = IAP, and 3 = SPI_SLAVE</td>
</tr>
<tr>
<td>37</td>
<td>4</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>7</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>UIC_STATUS</td>
<td>Design initialization status</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>UIC_SOURCE_TYPE</td>
<td>Design initialization Data source type when execution finished or halted</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>4</td>
<td>UIC_START_ADDRESS</td>
<td>Design initialization Data source address when execution finished or halted</td>
</tr>
<tr>
<td>56</td>
<td>4</td>
<td>UIC_INSTR_ADDRESS</td>
<td>Design initialization Data instruction count from the start of Design initialization execution</td>
</tr>
<tr>
<td>60</td>
<td>4</td>
<td>CYCLECOUNT</td>
<td>Programming cycle count</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>IAP_ERROR_CODE</td>
<td>IAP error information</td>
</tr>
<tr>
<td>65</td>
<td>7</td>
<td>RESERVED</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>4</td>
<td>IAP_LOCATION</td>
<td>External SPI flash memory address that was used during IAP</td>
</tr>
</tbody>
</table>
The following table lists the digest information bit fields.

<table>
<thead>
<tr>
<th>Offset (byte)</th>
<th>Size (bytes)</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>CFD</td>
<td>Fabric digest</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>CCDIGEST</td>
<td>Fabric Configuration segment digest</td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>SNVMDIGEST</td>
<td>sNVM Digest</td>
</tr>
<tr>
<td>96</td>
<td>32</td>
<td>ULDIGEST</td>
<td>User lock segment</td>
</tr>
<tr>
<td>128</td>
<td>32</td>
<td>UKDIGEST0</td>
<td>User Key Digest 0 in User Key segment (includes SRAM PUF activation code and device Integrity bit)</td>
</tr>
<tr>
<td>160</td>
<td>32</td>
<td>UKDIGEST1</td>
<td>User Key Digest 1 in User Key segment</td>
</tr>
<tr>
<td>192</td>
<td>32</td>
<td>UKDIGEST2</td>
<td>User Key Digest 2 in User Key segment (UPK1)</td>
</tr>
<tr>
<td>224</td>
<td>32</td>
<td>UKDIGEST3</td>
<td>User Key Digest 3 in User Key segment (UEK1)</td>
</tr>
<tr>
<td>256</td>
<td>32</td>
<td>UKDIGEST4</td>
<td>User Key Digest 4 in User Key segment (DPK)</td>
</tr>
<tr>
<td>288</td>
<td>32</td>
<td>UKDIGEST5</td>
<td>User Key Digest 5 in User Key segment (UPK2)</td>
</tr>
<tr>
<td>320</td>
<td>32</td>
<td>UKDIGEST6</td>
<td>User Key Digest 6 in User Key segment (UEK2)</td>
</tr>
<tr>
<td>352</td>
<td>32</td>
<td>UPDIGEST</td>
<td>User Permanent lock (UPERM) segment</td>
</tr>
<tr>
<td>384</td>
<td>32</td>
<td>FDIGEST</td>
<td>Digest for Factory Key Segments</td>
</tr>
<tr>
<td>Total</td>
<td>416</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 6: Running the TCL Script

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_dg0798_eval_df/TCL_Scripts/readme.txt or mpf_dg0798_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 7: References

This section lists documents that provide more information about system services and other IP cores used to build the system services.

- For more information on Design and Data Security Services, see UG0753: PolarFire FPGA Security User Guide.
- For more information about the CoreJTAGDEBUG IP core, see CoreJTAGDebug_HB.pdf from Libero->Catalog.
- 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 webpage.