Contents

1 Revision History ................................................................. 1
  1.1 Revision 2.0 ...................................................................... 1
  1.2 Revision 1.0 ...................................................................... 1

2 PolarFire FPGA System Services ............................................. 2
  2.1 CoreSysServices_PF IP Overview ...................................... 2
  2.2 Design Requirements ....................................................... 4
  2.3 Prerequisites .................................................................... 5
  2.4 Demo Design ................................................................. 7
    2.4.1 Design Implementation ........................................... 7
    2.4.2 IP Configuration ..................................................... 8
  2.5 Clocking Structure .......................................................... 19

3 Libero Design Flow ............................................................... 20
  3.1 Synthesize ....................................................................... 21
  3.2 Place and Route ............................................................. 21
    3.2.1 Resource Utilization ................................................. 21
  3.3 Verify Timing .................................................................. 21
  3.4 Generate FPGA Array Data ............................................ 21
  3.5 Configure Design Initialization Data and Memories .......... 22
  3.6 Configure Programming Options ...................................... 25
  3.7 Generate Bitstream ........................................................ 26
  3.8 Run PROGRAM Action ..................................................... 26

4 Programming the Device Using FlashPro ................................. 27

5 Setting up the Serial Terminal Program - PuTTY ......................... 28

6 Running the Demo ............................................................... 30

7 Appendix: Device Certificate Information ............................... 37

8 Appendix: Query Security ...................................................... 39

9 Appendix: Debug Information ................................................ 40

10 Appendix: Digest Information ............................................... 41

11 Appendix: References ........................................................ 42
## Figures

<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>System Services Design Block Diagram</td>
<td>6</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Top Level Libero Design</td>
<td>7</td>
</tr>
<tr>
<td>Figure 5</td>
<td>PF_INIT_MONITOR Configuration</td>
<td>8</td>
</tr>
<tr>
<td>Figure 6</td>
<td>PF_CCC_0 Input Clock Configuration</td>
<td>9</td>
</tr>
<tr>
<td>Figure 7</td>
<td>PF_CCC_0 Output Clock Configuration</td>
<td>9</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Mi-V Configuration</td>
<td>10</td>
</tr>
<tr>
<td>Figure 9</td>
<td>CoreUARTapb Configuration</td>
<td>11</td>
</tr>
<tr>
<td>Figure 10</td>
<td>CoreJTAGDebug Configuration</td>
<td>11</td>
</tr>
<tr>
<td>Figure 11</td>
<td>PF_SRAM_AHBL_AXI Configuration</td>
<td>12</td>
</tr>
<tr>
<td>Figure 12</td>
<td>CoreGPIO_0 Configuration</td>
<td>13</td>
</tr>
<tr>
<td>Figure 13</td>
<td>CoreSysServices_PF Configuration</td>
<td>14</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Memory Map</td>
<td>15</td>
</tr>
<tr>
<td>Figure 15</td>
<td>CoreAHBLite_0 Configuration</td>
<td>16</td>
</tr>
<tr>
<td>Figure 16</td>
<td>CoreAHBLite_1 Configuration</td>
<td>17</td>
</tr>
<tr>
<td>Figure 17</td>
<td>COREAHBTOAPB3 Configuration</td>
<td>17</td>
</tr>
<tr>
<td>Figure 18</td>
<td>CoreAPB3_0 Configuration</td>
<td>18</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Clocking Structure</td>
<td>19</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Libero Design Flow Options</td>
<td>20</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Design and Memory Initialization</td>
<td>22</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Fabric RAMs Tab</td>
<td>23</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Edit Fabric RAM Initialization Client</td>
<td>23</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Apply Fabric RAM Content</td>
<td>24</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Add PlainText NonAuthenticated Option</td>
<td>24</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Edit PlainText NonAuthenticated Client</td>
<td>25</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Configure Programming Options</td>
<td>25</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Board Setup</td>
<td>26</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Run Passed</td>
<td>27</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Finding the COM Port</td>
<td>28</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Select Serial as the Connection Type</td>
<td>28</td>
</tr>
<tr>
<td>Figure 32</td>
<td>PuTTY Configuration</td>
<td>29</td>
</tr>
<tr>
<td>Figure 33</td>
<td>System Services Options</td>
<td>30</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Device Serial Number</td>
<td>30</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Device User-code</td>
<td>30</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Device Design Information</td>
<td>31</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Device Certificate</td>
<td>32</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Digest</td>
<td>33</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Security Locks Information</td>
<td>33</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Debug Information</td>
<td>33</td>
</tr>
<tr>
<td>Figure 41</td>
<td>Digital Signature</td>
<td>34</td>
</tr>
<tr>
<td>Figure 42</td>
<td>Secure NVM Services</td>
<td>35</td>
</tr>
<tr>
<td>Figure 43</td>
<td>PUF Emulation Service</td>
<td>35</td>
</tr>
<tr>
<td>Figure 44</td>
<td>Generated Nonce</td>
<td>35</td>
</tr>
<tr>
<td>Figure 45</td>
<td>Flash Freeze</td>
<td>36</td>
</tr>
<tr>
<td>Figure 46</td>
<td>I/O State in Flash*Freeze</td>
<td>36</td>
</tr>
<tr>
<td>Figure 47</td>
<td>Copy Device Certificate</td>
<td>37</td>
</tr>
<tr>
<td>Figure 48</td>
<td>Certificate Decoding Using Java Script</td>
<td>38</td>
</tr>
<tr>
<td>Figure 49</td>
<td>Decoded Certificate</td>
<td>38</td>
</tr>
</tbody>
</table>
## Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>System Services in the Demo</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>System Services Descriptor</td>
<td>3</td>
</tr>
<tr>
<td>Table 3</td>
<td>Design Requirements</td>
<td>4</td>
</tr>
<tr>
<td>Table 4</td>
<td>I/O Signals</td>
<td>7</td>
</tr>
<tr>
<td>Table 5</td>
<td>Resource Utilization</td>
<td>21</td>
</tr>
<tr>
<td>Table 6</td>
<td>Jumper Settings for PolarFire Device Programming</td>
<td>26</td>
</tr>
<tr>
<td>Table 7</td>
<td>Device Certificate Fields (1024 bytes)</td>
<td>37</td>
</tr>
<tr>
<td>Table 8</td>
<td>Security Locks Fields</td>
<td>39</td>
</tr>
<tr>
<td>Table 9</td>
<td>Debug Info Fields</td>
<td>40</td>
</tr>
<tr>
<td>Table 10</td>
<td>Digest Information Bit Fields</td>
<td>41</td>
</tr>
</tbody>
</table>
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 2.0
Updated the document for Libero SoC PolarFire v2.2.

### 1.2 Revision 1.0
The first publication of this document.
System services are the system controller actions initiated from the FPGA design using the CoreSysServices_PF IP core. The system controller hard block in PolarFire® FPGAs provides various system services. The CoreSysServices_PF IP 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.

Note: In the demo, the Flash*Freeze service is demonstrated with timed entry and timed exit.

The demo design includes the Mi-V soft processor, which initiates the system service requests and enables the CoreSysService_PF IP 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 PolarFire, see Demo Design, page 5.

The demo 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 demo design, see Programming the Device Using FlashPro, page 27.
- **Using Libero SoC PolarFire**: To program the device using Libero SoC PolarFire, see Libero Design Flow, page 20.

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

### 2.1 CoreSysServices_PF IP Overview

The 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 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 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 demo, the sample SoftConsole project is migrated to SoftConsole v5.2 and the application file `main.c` is modified to provide the user options.

**Figure 2 • Firmware catalog**

[Image of Firmware catalog]

### 2.2 Design Requirements

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

**Table 3 • 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>Hardware</td>
<td></td>
</tr>
<tr>
<td>PolarFire Splash Kit (MPF300TS-1FCG484EES)</td>
<td>Rev 2 or later</td>
</tr>
<tr>
<td>• PolarFire splash board</td>
<td></td>
</tr>
<tr>
<td>• 12 V, 5 A AC power adapter and cord</td>
<td></td>
</tr>
<tr>
<td>• USB 2.0 A to mini-B cable for universal asynchronous receiver-transmitter (UART) and programming</td>
<td></td>
</tr>
<tr>
<td>Host PC</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>FlashPro</td>
<td>12.200.30.10</td>
</tr>
<tr>
<td>Libero SoC PolarFire Design Suite</td>
<td>2.2</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>
<tr>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>PF_INIT_MONITOR</td>
<td>2.0.103</td>
</tr>
</tbody>
</table>
### 2.3 Prerequisites

Before you start:

- Download the demo design files from the following location:
  
  ![Image](http://soc.microsemi.com/download/rsc/?f=mpf dg0824_liberosocpolarfirev2p2_df)

- 1. Download and install Libero SoC PolarFire v2.2 on the host PC from the following location:
  
  ![Image](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 Demo Design

The following steps describe the data flow in the demo design:

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 CoreSysServices_PF IP 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 CoreSysServices_PF IP over the mailbox interface.
4. The Mi-V processor receives the service response and forwards the data to the UART interface.

---

### Table 3 • Design Requirements

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF_CCC</td>
<td>1.0.113</td>
</tr>
<tr>
<td>CoreJTAGDEBUG</td>
<td>2.0.100</td>
</tr>
<tr>
<td>CORESET_PF</td>
<td>2.1.100</td>
</tr>
<tr>
<td>Mi-V soft processor (MIV_RV32IMA_L1_AHB)</td>
<td>2.0.100</td>
</tr>
<tr>
<td>COREAHBLite</td>
<td>5.3.101</td>
</tr>
<tr>
<td>COREAHBTOAPB3</td>
<td>3.1.100</td>
</tr>
<tr>
<td>CoreAPB3</td>
<td>4.1.100</td>
</tr>
<tr>
<td>CoreUARTapb</td>
<td>5.6.102</td>
</tr>
<tr>
<td>CoreGPIO</td>
<td>3.2.102</td>
</tr>
<tr>
<td>CoreSysServices</td>
<td>2.3.116</td>
</tr>
<tr>
<td>PF_SRAM_AHBL_AXI</td>
<td>1.1.125</td>
</tr>
<tr>
<td>Core_SPI</td>
<td>5.1.104</td>
</tr>
</tbody>
</table>

Note: Any serial terminal emulation program can be used. PuTTY is used in this demo.
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>
2.4.2  **IP Configuration**

The following sections describe the IP cores used in the design and their configurations. The other IP cores retain the default configuration.

2.4.2.1  **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 5 • PF_INIT_MONITOR Configuration*
2.4.2.2 PF_CCC_0 Configuration

The PolarFire Clock Conditioning Circuitry (CCC) block takes an input clock of 50 MHz from the onboard oscillator through CLKINT 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 6 • PF_CCC_0 Input Clock Configuration**

**Figure 7 • PF_CCC_0 Output Clock Configuration**
2.4.2.3 Mi-V Soft Processor Configuration

The Reset Vector Address of the Mi-V soft processor is set to 0x8000_0000 from 0x6000_0000. After the 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 Figure 8, page 10.

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 8 • Mi-V Configuration

2.4.2.4 CoreUARTapb

The CoreUARTapb IP is connected to the Mi-V soft processor as an APB slave. It interfaces with the host PC for UART communication. Figure 9, page 11 shows the configuration settings of the CoreUARTapb IP:

- **TX FIFO**: Disabled by default.
  The UART transmit state machine immediately begins to transmit data and continues transmission until the data buffer is empty in normal mode. If TX FIFO is enabled, it continues to transmit until TX FIFO is empty. In this design, normal mode (without FIFO) is selected.
- **RX FIFO**: Disabled by default.
  The UART receive state machine stores the data in receive data buffer if FIFO is not enabled.
- **Configuration**: Set to **Programmable** by default.
The user application programs the baud rate, character size, and the parity configuration using the UART driver. If the **Fixed** option is selected, the user application can not overwrite these parameters.

*Figure 9 • CoreUARTapb Configuration*

![CoreUARTapb Configurator](image1)

### 2.4.2.5 CoreJTAGDebug

The CoreJTAGDebug IP connects the Mi-V soft processor to the JTAG header for debugging. The following figure shows the configuration of the CoreJTAGDebug IP core.

*Figure 10 • CoreJTAGDebug Configuration*

![CoreJTAGDebug Configurator](image2)
2.4.2.6 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 16384 words to accommodate an application of up to 65 KB into LSRAM. The present application is below 50 KB so this can fit into either sNVM or μPROM. In this demo, μPROM 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*

2.4.2.7 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 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, 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 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 12 • CoreGPIO_0 Configuration

2.4.2.8 CoreSysServices_PF Configuration

CoreSysServices IP provides access to the system controller. It is connected to Mi-V soft processor as an APB slave. The configuration settings of CoreSysServices_PF IP are as follows:

- By default, all the service check boxes are selected. The application can initiate these selected services.
- **FlashFreeze time out**: The default value is 0x20000000 ms. This value can be overwritten by the application before initiating the Flash*Freeze service. When device enters the Flash*Freeze state, the device automatically exits the Flash*Freeze state after 0x20000000 ms.
- **FlashFreeze Mailbox Address**: It is the mailbox address location where time out value is passed as input in the service request. The default value is 0x100. The allowed value is between 0 to 511.
CoreSysServices IP is configured as shown in the following figure.

*Figure 13 • CoreSysServices_PF Configuration*
2.4.2.9 **Design Memory Map**

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

*Figure 14 • Memory Map*
2.4.2.9.1 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**

![CoreAHBLite Configurator](image)

### 2.4.2.9.2 COREAHBTOAPB3

The CoreAHBtoAPB3 works as a bridge in between the AHB and the APB domains. CoreAHBtoAPB3 interfaces with CoreAHBLite through its AHB interface and with CoreAPB3 through its APB interface. The following figure shows configuration of COREAHBTOAPB3 IP core.

**Figure 17 • COREAHBTOAPB3 Configuration**

![CoreAHBtoAPB3 Configurator](image)
2.4.2.9.3 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 18 • CoreAPB3 Configuration*
2.5 Clocking Structure

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

*Figure 19* • Clocking Structure
The Libero design flow involves running the following processes in the Libero SoC PolarFire:

- Synthesize, page 21
- Place and Route, page 21
- Verify Timing, page 21
- Generate FPGA Array Data, page 21
- Configure Design Initialization Data and Memories, page 22
- Configure Programming Options, page 25
- Generate Bitstream, page 26
- Run PROGRAM Action, page 26

The following figure shows these options in the Design Flow tab.
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 20, page 20.
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. 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_SUBSYSTEM Derived_Constraints.sdc file for timing constraints
- Th pll_placement.pdc file for PLL placement

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>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>14471</td>
<td>299544</td>
<td>4.83</td>
</tr>
<tr>
<td>DFF</td>
<td>7457</td>
<td>299544</td>
<td>2.49</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>15153</td>
<td>299544</td>
<td>5.06</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 20, page 20.
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 20, page 20.
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 the demo, the LSRAM initialization client is stored in the μPROM.

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.

   ![Design and Memory Initialization](image)

   **Figure 21 • 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 22 • Fabric RAMs Tab**

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

**Figure 24 • Apply Fabric RAM Content**

5. Select the **sNVM** tab and select the **Add** from the list. Click **Add PlainText NonAuthenticated Client** as shown in the following figure.

**Figure 25 • Add PlainText NonAuthenticated Option**
6. In the **Edit PlainText NonAuthenticated** client dialog box, select the **Content filled with 0's** option, and provide the **Number of bytes** and Click **OK** as shown in the following figure.

   **Figure 26 • Edit PlainText NonAuthenticated Client**

   ![Edit PlainText NonAuthenticated Client](image)

7. Click **Apply** in the **Design Initialization** tab.

8. In Libero Design Flow, 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 20**, page 20.

### 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 27 • Configure Programming Options**

   ![Configure Programming Options](image)
3.7 Generate Bitstream

To generate the bitstream:

1. Double-click **Generate Bitstream** from the **Design Flow** tab. When the bitstream is successfully generated, a green tick mark appears as shown in Figure 20, page 20.

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.

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.

![Board Setup](image)

5. Double-click **Run PROGRAM Action** from the **Libero Design Flow**.

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 20, page 20.
4 Programming the Device Using FlashPro

This chapter describes how to program the PolarFire device with the .stp programming file using FlashPro software without opening Libero SoC PolarFire. The system_services.stp file is available at the following design files folder location:

mpf_dg0824_liberosocpolarfirev2p2_df\Programming_file

Follow these steps to program the device:

1. Connect the jumpers and set up the PolarFire Splash Kit board as described in steps 1 to 4 of Run PROGRAM Action, page 26
2. Figure 28, page 26 shows the board setup after these connections are made.
3. On the host PC, start the FlashPro software.
4. Click New Project to create a new project.
5. In the New Project window, do the following, and click OK:
   • Enter a project name
   • Select Single device as the programming mode
6. Click Configure Device.
7. Click Browse, and locate the system_services.stp file from the following location:
   mpf_dg0824_liberosocpolarfirev2p2_df\Programming_file
8. Click Program to program the PolarFire Device.

The device is successfully programmed and the onboard LEDs 4, 5, 6, 7, and 8 glow. The RUN PASSED message is displayed as shown in the following figure.

Figure 29 • Run Passed

After the device is programmed, PuTTY must be launched to run the system services demo. See, Setting up the Serial Terminal Program - PuTTY, page 28.
5 Setting up the Serial Terminal Program - PuTTY

The user application receives the 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 J1 (USB) port on the board.
2. Connect the power supply cable to the J2 connector on the board.
3. Power on the 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 5 (COM5) is selected as shown in the following figure. COM Port-numbers may vary.

   Figure 30 • Finding the COM Port

   ![Finding the COM Port](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 31 • Select Serial as the Connection Type

   ![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.

*Figure 32 • PuTTY Configuration*

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

PuTTY opens successfully, and this completes the serial terminal emulation program setup. See *Running the Demo*, page 30.
This chapter describes how to run the system services demo using the serial terminal program (PuTTY). Before running the demo, ensure the device is programmed and serial terminal is set up. For more information on setting up the serial terminal, see Setting up the Serial Terminal Program - PuTTY, page 28.

To run the demo, perform the following steps:

1. Power on the board using the SW1 slide switch.

System services options are displayed on the PuTTY as shown in the following figure.

**Figure 33 • System Services Options**

```
**** PolarFire Device and Design system services Example ****
Notes: Return data from System controller is displayed byte-wise with LSB first.
Input data is provided LSB first. Each ASCII character is one Mibble of data.

Select Service:
1. Read Device Serial number
2. Read Device User-code
3. Read Device Design-info
4. Read Device Certificate
5. Read Digest
6. Query security
7. Read debug information
8. Digital signature
9. Secure XMM services
a. PUF Emulation
b. None
c. Flash Freeze
```

2. Enter 1 to select **Read Device Serial number**.

The 128-bit device serial number (DSN) is displayed as shown in the following figure.

**Figure 34 • Device Serial Number**

```
Device serial number: 42ED5BFF9B99F4B5DFDE8989F7B
```

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

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

SNM—The second component is the serial number modifier (SNM). SNM is initialized during 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.

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 35 • Device User-code**

```
32bit USERCODE/Silicon signature (MSB first): 89ABCDEF
```
Running the Demo

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 36**, page 31, 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 36 • Device Design Information**

```
Design ID: 2D8150524F635F5355425353545454D
0000000000000000000000000000000000000000
Design Version: 0100
Design Back-Level: 0000
```

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

The device supply chain assurance certificate is displayed as shown in **Figure 37**, page 32 and **Figure**, page 32.
Running the Demo

For more information about decoding the device certificate see, Appendix: Device Certificate Information, page 37.

**Figure 37 • Device Certificate**

```
Device Certificate:
308203C082B0052A00302010021240
D1A3A3CA58A88B07C6E960280C7CD3
01300A06082A8E6BCE3AB4033003931
000300263101042205330102000A
00035504081305334F34310C0000
0500410A13040D343931103D0180033F
0403131D28263462233328353310
3113193533073765302017031393130
311343303303373457D03003933339331
3223310323339339A034814000008
060035041C1304303003003103063155
040000050406C112446652E020020
302800350502A0C014D0649393054
2020202020202020202020202020
2020202020202020202020202020
286483C0020106020B1090202362
000000014C0D8494F4F6D5C84D8C8C84
355057394935397C1343363B13C932
337C7790F99120D300F19633521D
C060403719A626A002E55532F80B
9688A0655040690311387313D108P0A
610629142C3830D7002F1D16C0704
77F91850403D464D29151G08321100
42ED28F9019F481558DF48ED6889F7B
A3B201D309201D201060A02B69104
01822D6401010400000003018P06
0A2B026101020B0E1D0004810F0FFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF
```
6. Enter 5 to select **Read Digest**.

The 416 byte Digest contains the fabric digest, sNVM digest and user key digests. The Digest protects the data integrity. The following figure shows the 416 byte digest displayed. For more information about decoding the digest output see, **Appendix: Digest Information**, page 41

**Figure 38 • Digest**

```
Read Digest: 01067C0FA01C52CAAB4F85358511F2E53 5A35A03121DB0E550B32376A7C4ACE
70B0FF5452C52CA6831BC1B4A5748B592A 3B070F7372A0B60FE2CBE3B974284C6D
E3B0C44299F1C1C494EF64C8996FB924 27A42E4E49B934CA495991D7552B055 223186F31R639A92B1869AC19F222D712
2212B3055CA0A725D96B9FB188F348C 00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
108CF3C9D97E2E8E1A651B3CD24C108E
CD346ED9Y3CD6525656E272BF596A6
```

7. Enter 6 to select **Query Security**.

The non-volatile states of user security locks are displayed as shown in the following figure.

**Figure 39 • Security Locks Information**

```
Security lock: 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: Query Security**, page 39.

**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 40 • Debug Information**

```
Debug info: 000000320027002700BB69AC09B409
FE064302A211000000000000000000000
0102DC40166ZFA83001BBA50000000
000107300000004024000042000000
000BC4290FC1C14000000
```

In **Figure 40**, page 33, the highlighted 4 bytes 42000000 (LSB first) indicate the number of times (in this example, 00000042), the device was programmed (programming cycles). For more information about the Debug Info fields, see **Appendix: Debug Information**, page 40.
9. Enter 8 to select **Digital Signature**.

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

**Figure 41 • Digital Signature**

```
Digital Signature service:
48 byte hash value:
9C3A68D6D604199740984213C5586E6
C71F7F9EBC8A42800FB72D43899167F8
9C3A68D6D604199740984213C5586E6
Raw format:
Digital Signature service successful.
Output Digital Signature - Raw format:
92189A45879E2BAEDF4DCA996E6833F
CB659278CA550CF7C016F7D2666E653
8757032485D6C887F6F6DCACDF45A5C1
C31F028BB154049739C9D45368E91A4AD
9D2883D8B94FB516F81F0DC5F3E58464
BE617946589F248569897AE21CB92CB
DER format:

Digital Signature service successful.
Output Digital Signature - DER format:
30066023100D0909760FC8E5AB496D6E12
67030BEE9F8F3D692994A0A66B6E272
7697D582D01E538614892600294300F
F6A65C53A02310009AE507B47CDEA42
273F60CA77F61403D45958E4D413A88
C4B3A33F9E2555256F3A32E529B9EA2D
C0D4F02822A40A8
```

The digital signature service takes a user-supplied SHA384 hash and signs it with the device 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 elliptic curve digital signature algorithm (ECDSA) using the hash and the device 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 42, page 35.

**Figure 42 • Secure NVM Services**

![Secure NVM Services](image)

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 43 • PUF Emulation Service**

![PUF Emulation Service](image)

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 44 • Generated Nonce**

![Generated Nonce](image)
13. Enter ‘c’ to select **Flash**-*Freeze service*. The device enters Flash-*Freeze* state for 5 seconds and exits automatically. The Flash-*Freeze* duration is shown in **Figure 45**, page 36.

**Figure 45 • Flash Freeze**

Device will exit Flash Freeze after the 5 seconds.

In the Flash-*Freeze* state LED7 glows. Because, the last value is selected for I/O state in Flash-*Freeze* mode in the Libero I/O Editor as shown in the following figure

**Figure 46 • I/O State in Flash-*Freeze***

With this step the system services demo is successfully completed. Power cycle the board to run the system services options again.
Appendix: Device Certificate Information

The Device Certificate is a 1024-byte Microsemi-signed X-509 certificate programmed during manufacturing. This 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/.

Follow these steps for decoding the certificate:

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

Figure 47 • 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 mpf_dg0824_liberosocpolarfirev2p2_df\Device_certificate\sample.txt.
3. The web page displays all the fields in certificate as shown in the following figure.

Figure 49 • Decoded Certificate
The following table lists each security lock bit and its features.

<table>
<thead>
<tr>
<th>Table 8 • Security Locks Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Byte</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>2-8</td>
</tr>
</tbody>
</table>
The following table lists the bit fields of debug information.

<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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1=JTAG, 2=IAP, 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 bit fields of digest information.

**Table 10 • 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></td>
<td></td>
<td></td>
<td>Total 416 bytes</td>
</tr>
</tbody>
</table>
This section lists the documents that provide more information about system services and other IP cores used to build the demo design.

- For more information on Design and Data Security Services, see the *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 CoreAHBtoAPB3 IP core, see CoreAHBtoAPB3_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 webpage.
- For more information about SoftConsole migration from v5.1 to v5.2, see *Migrating a SoftConsole v5.1 Project to SoftConsole v5.2 Application Note*. 