AC482
Application Note
PolarFire FPGA: How to Perform On-Demand Digest Check

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

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

Contents

Figures

Tables

1 Revision History

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

1.1 Revision 4.0

Information about CoreSysServices_PF IP core was replaced with PF_SYSTEM_SERVICES IP core.

1.2 Revision 3.0

Added Appendix: Running the TCL Script, page 13.

1.3 Revision 2.0

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 1.0

The first publication of this document.

2 How to Perform On-Demand Digest Check

PolarFire devices have a built-in self-test mechanism that can be used (optionally) to check the design integrity and security of the device automatically upon power-up, or on-demand. The contents of all the nonvolatile configuration memory segments, including security keys, security settings, and the FPGA fabric configuration, plus any sNVM memory pages declared as ROM by the user (all the write-protected pages) are tested using the digest check feature. This test provides assurance against both natural and maliciously induced failures.

In the factory and user security segment, each logical page contains an automatically generated digest, calculated dynamically at the time of programming the data to be written. For the FPGA fabric, the digest includes an overall value covering the data to be programmed. Also, the digests are calculated and stored for the sNVM pages marked as ROM. The digests can be verified on-demand by the user, either internally using a system service, or externally using a programming instruction. In addition, the user can automatically run digest checks on each power-up.

An endurance limit specifies how many times a digest of the FPGA fabric can be run. For more information about the FPGA configuration memory endurance limits, see *DS0141: PolarFire FPGA Datasheet*. Therefore, depending upon how the system is deployed and used (for example, how often it is powered-up), the on-demand digest check may be more appropriate for testing the integrity of the FPGA fabric.

2.1 On-Demand Digest Check

The on-demand digest check recalculates and compares digests of selected non-volatile memories (FPGA fabric, sNVM, and security segments) with the stored digests. A failure of any digest results in the tamper event being triggered for a user action. The on-demand digest check is invoked by calling digest check design system service.

If the digest check is performed on the FPGA fabric, then the FPGA fabric is automatically placed in suspend state before commencing the digest check operation. Except for LSRAMs, all the FPGA fabric logic elements and uSRAMs hold their current state during the suspend state. Upon completion of the fabric digest, the suspend state is automatically exited. Since the LSRAMs does not retain the user data after performing digest check on FPGA fabric. The status of the fabric digest check must be monitored by a state machine implemented in the fabric.

After checking the status of the fabric digest check, the state machine needs to issue a design reset or device reset depending on the design requirements. The device reset reinitializes the device and design memories according to the user configuration. The user design reset does not reinitialize the device or LSRAMs. The user design reset must be implemented to reset the user design. If the design uses LSRAMs initialization at power-up then the user design needs to issue a device reset otherwise design reset is sufficient to restart the design. Use RESET_DEVICE tamper response signal for device reset.

The FPGA fabric design including LSRAMs content is not affected when the digest check is performed on sNVM or security segments. see *UG0753: PolarFire FPGA Security User Guide* for more information on Digest Check Service.

This application note demonstrates how to perform an on-demand digest check with a design example.

2.2 Design Requirements

The following table lists the hardware and software required to perform fabric digest check operation:

Table 1 • **Resource Requirements**

Requirement	Version
Host PC Operating system	Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit with MPF300TS-1FCG1152 device	Rev D
Software	
Libero SoC	**Note:** Refer to the readme.txt file provided in the design files for the software versions used with this reference design.
Tera Term	

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

Download the design files from the following location:
http://soc.microsemi.com/download/rsc/?f=mpf_ac482_df

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

Note: The provided design example is created for MPF300TS-1FCG1152 device (Non-ES device).

2.4 Design Description

The design example provided with the application note performs the following operations:

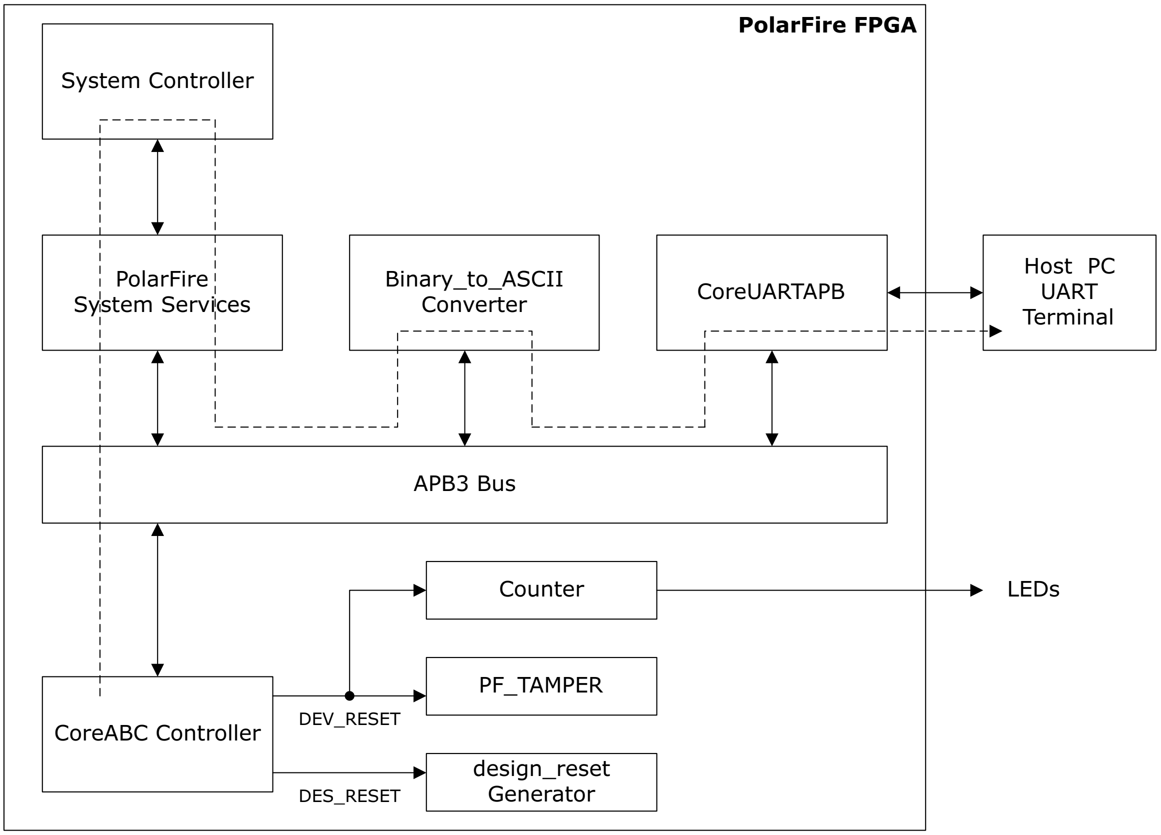
1. Initiates system service request to perform the following system services:
    - Fetch Device Serial Number (DSN)
    - Fetch USERCODE
    - Digest Check on FPGA Fabric
2. Demonstrates how to perform design reset or device reset after fabric digest check
3. Provides user interface through UART terminal

System services are system controller actions initiated from the user design. PF_SYSTEM_SERVICES IP must be used to initiate system service requests. PF_SYSTEM_SERVICES IP provides a set of registers, which are accessible through the APB interface for executing the system services. On the other hand, the PF_SYSTEM_SERVICES IP interfaces to the PolarFire System Controller through a system service interface. The System Controller has a 2 KB mailbox RAM for any additional input parameters and any outputs from the service.

A CoreABC processor is used as a controller to perform the preceding operations. CoreABC (APB Bus Controller) is a simple, configurable, low gate count, programmable state machine/controller primarily targeted towards the implementation of AMBA APB-based designs. For information about the instructions supported by the CoreABC, see *CoreABC handbook*.

The following figure shows the block diagram of the example design. The design is implemented without consuming LSRAMs since the LSRAMs content is not retained after performing the fabric digest check.

Figure 1 • **Design Block Diagram**



The dotted line in the Figure 1, page 4 shows the data flow. The data flow in Figure 1, page 4 is explained as follows:

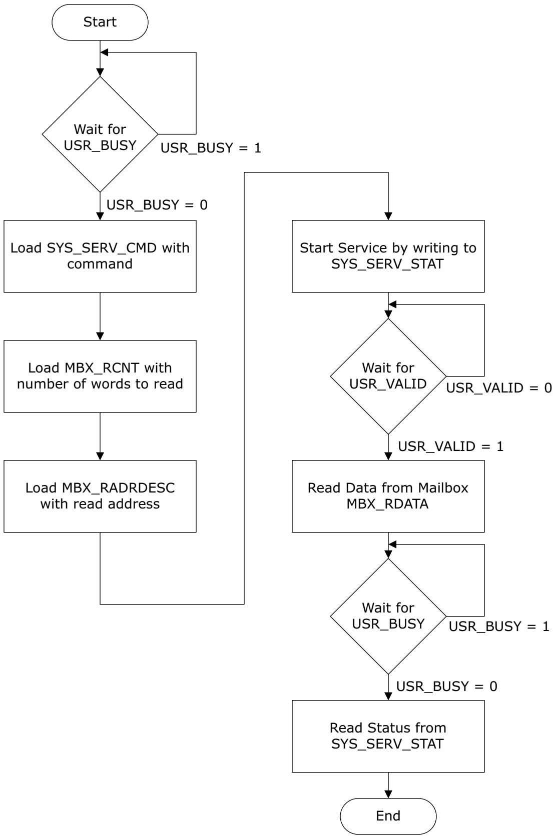
1. CoreABC processor initiates a system service through PF_SYSTEM_SERVICES IP.
2. System Controller performs the requested action and writes the result to the System Controller mailbox.
3. CoreABC processor reads the result and converts it into ASCII format using the Binary_to_ASCII converter.
4. CoreABC processor prints the result on a host PC UART terminal through CoreUARTAPB controller.
5. After digest check, the CoreABC processor issues a device reset or a design reset based on user input. The PF_TAMPER macro is used to issue a device reset.

The System Services are invoked by writing to appropriate PF_SYSTEM_SERVICES IP registers with the system service command and any additional information required to perform the system service. Then the PF_SYSTEM_SERVICES IP initiates necessary transactions on the system service interface. The PF_SYSTEM_SERVICES IP asserts the USR_BUSY signal when the IP is busy in executing the service. Upon completion of a system service, a status code is written to the status register and the USR_BUSY signal gets de-asserted. For services requiring the mailbox reads, the signal is de-asserted only when the required number of reads for the service is completed. If a new service request is initiated when the USR_BUSY is in progress, the new request is ignored until the current service completes. For more information about PF_SYSTEM_SERVICES IP, see *UG0848: PolarFire System Services User Guide*.

The following flowchart shows the sequence of instructions coded in the CoreABC program to read DSN or USERCODE using system services. The command value and number of words (32 bits) to read (MBX_RCNT) are service dependent:

- For read DSN service, the command is 0x00 and MBX_RCNT is 4
- For read USERCODE service, the command is 0x01 and MBX_RCNT is 1

Figure 2 • **Sequence of Instructions to read DSN or USERCODE**


Start
Wait for USR_BUSY
USR_BUSY = 1
USR_BUSY = 0
Load SYS_SERV_CMD with command
Load MBX_RCNT with number of words to read
Load MBX_RADRDESC with read address
Start Service by writing to SYS_SERV_STAT
Wait for USR_VALID
USR_VALID = 0
USR_VALID = 1
Read Data from Mailbox MBX_RDATA
Wait for USR_BUSY
USR_BUSY = 1
USR_BUSY = 0
Read Status from SYS_SERV_STAT
End

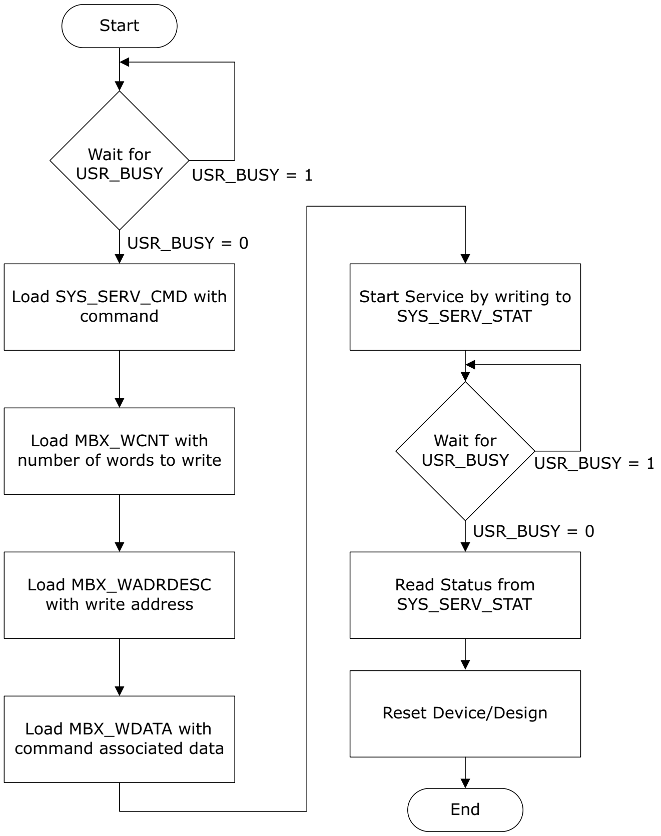
The command value to perform digest check service is 0x47 and this service request requires additional information (OPTIONS [15:0], as listed in the following table) to specify the area for which the digest check must be performed. The additional information is provided through mailbox writes. In this reference design, 0x1 is written to the mailbox (MBX_WDATA) to perform digest check on FPGA fabric. Users can change this parameter to perform digest on other NVM components. The digest check service does not expect any return data other than the status of the service.

Table 2 • **OPTIONS[15:0]**

OPTIONS [i]	Name	Description
0	CHECK FABRIC	Enables fabric digest
1	CC	Enables digest of fabric configuration data
2	SNVM	Enables digest of sNVM pages marked as ROM
3	UL	Enables digest of user security segment
4	UKDIGEST0	Enables digest of user key segment containing SRAMPUF data
5	UKDIGEST1	Enables digest of user key segment containing UEK (User EC key)
6	UKDIGEST2	Enables digest of user key segment containing UPK1
7	UKDIGEST3	Enables digest of user key segment containing UEK1

The following flowchart shows the sequence of instructions coded in the CoreABC program to perform digest check on the FPGA fabric using system service.

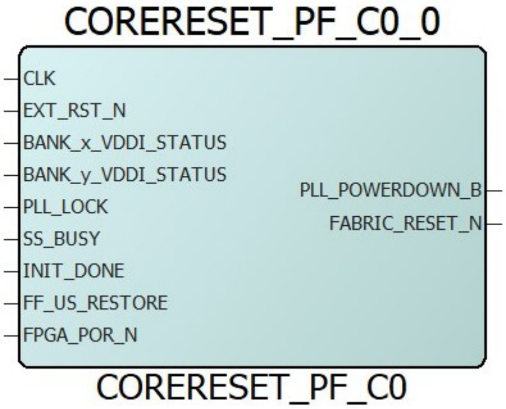
Figure 3 • **Sequence of Instructions to perform Digest Check on FPGA Fabric**


Start
Wait for USR_BUSY
USR_BUSY = 1
USR_BUSY = 0
Load SYS_SERV_CMD with command
Load MBX_WCNT with number of words to write
Load MBX_WADRDESC with write address
Load MBX_WDATA with command associated data
Start Service by writing to SYS_SERV_STAT
Wait for USR_BUSY
USR_BUSY = 1
USR_BUSY = 0
Read Status from SYS_SERV_STAT
Reset Device/Design
End

The FPGA fabric is put into suspend state during fabric digest check calculation as explained in On-Demand Digest Check, page 2. During the suspend state, the clocks are gated off to low, the PLLs are powered down, and all the I/O input buffers are disabled. After digest check, the FPGA fabric automatically comes out of suspend state, and the PLLs and clocks are re-enabled. The PLLs take time to re-acquire lock. Since the PLL lock is used in design reset generation, the PLL lock signal needs to be bypassed until the lock is re-acquired after exiting from suspend state. If not, the design gets reset because of the loss of PLL lock.

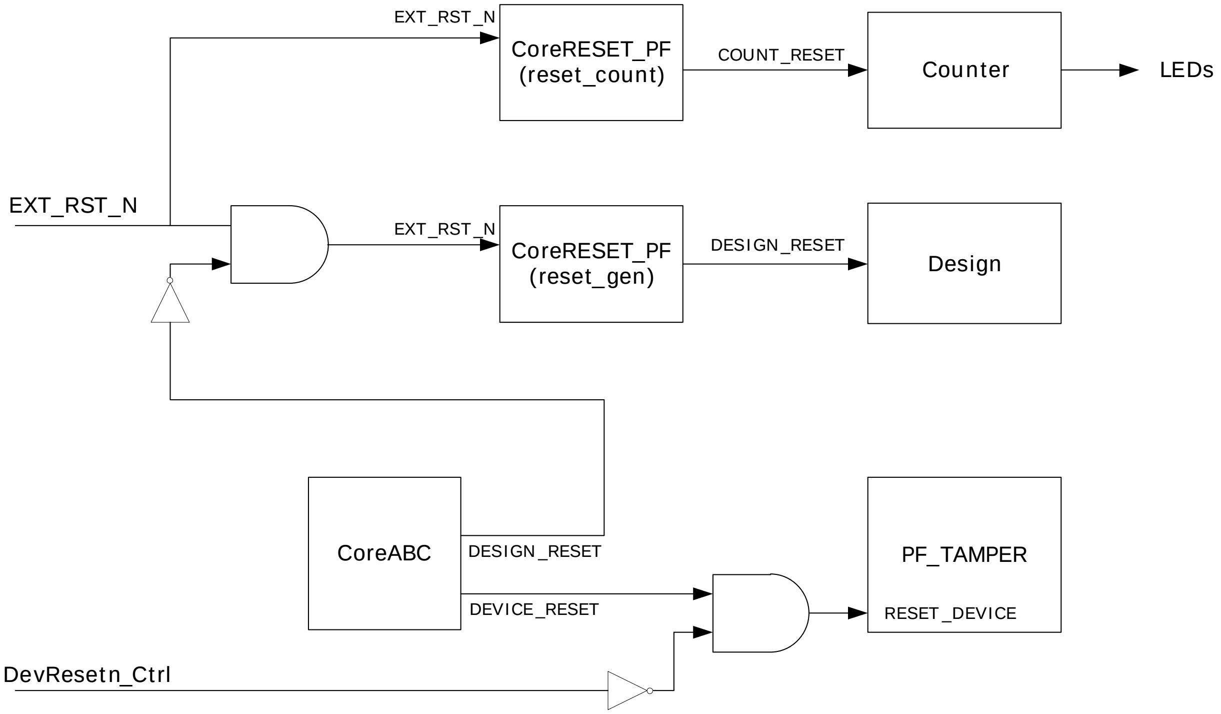
The example design uses CoreReset_PF IP core (as shown in the following figure) to serve the preceding purpose. The CoreReset_PF IP generates a reset, which is asserted asynchronously by one of the multiple potential sources and de-asserted synchronously with a specified clock. The potential sources of reset can be external reset through a GPIO, PLL lock, and INIT_DONE from PolarFire Initialization Monitor (PF_INIT_MONITOR). The CoreReset_PF IP uses USR_BUSY signals from PF_SYSTEM_SERVICES IP to bypass PLL_LOCK and EXT_RESET_N at the time of suspend state exit. This ensures that no reset occurs while the PLL is re-acquiring lock after exiting suspend state. FF_US_RESTORE must be tie low.

Figure 4 • **CoreReset_PF Ports**



If LSRAMs are used in the design, then the user design needs to issue a device reset or design reset after completing the digest check on the FPGA fabric. Based on the user input, the CoreABC processor asserts reset_device input of PF_TAMPER macro for device reset or asserts EXT_RST_N input of CoreReset_PF IP for design reset. To show differentiation between device reset and design reset, the example design uses two CoreReset_PF IPs: one is used to generate reset to complete design except a counter, which blinks LEDs on board and the other IP is used to reset the counter. The counter does not get reset when the design reset is initiated from the CoreABC processor. The counter gets reset only when the device reset is issued.

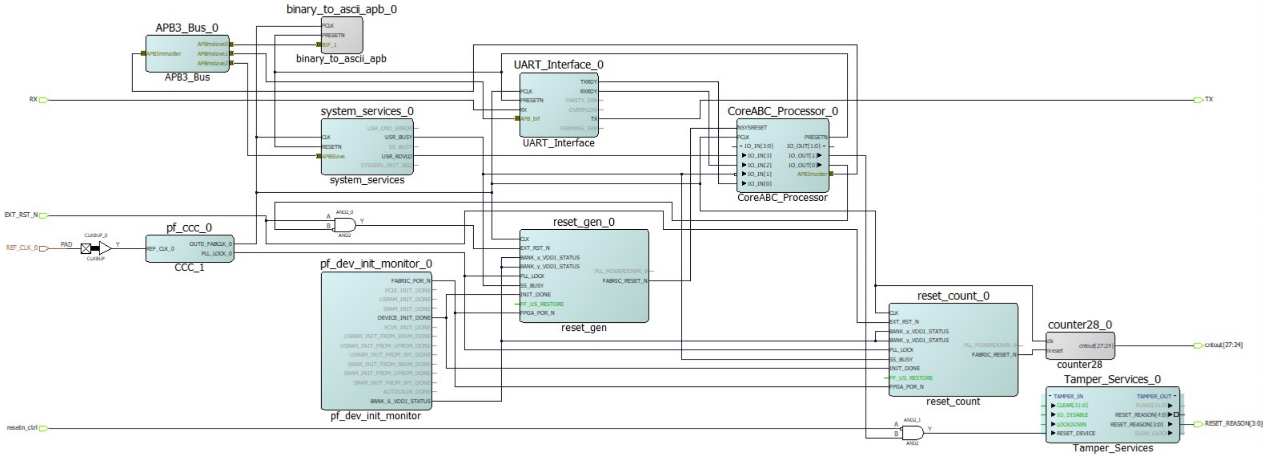
Figure 5 • **Reset Structure**



Note: RESET_DEVICE input of PF_TAMPER macro should not be promoted to top-level I/Os directly.The design example provides device reset control from an external DIP switch (SW11-DIP1) through DevResetn_Ctrl port. See Figure 5, page 8 for DevResetn_Ctrl port connection. SW11-DIP1 switch position should be set to ON to control the device reset from the user design. Logic '0' is driven on DevResetn_Ctrl port when the DIP1 switch position is set to ON.

The following figure shows the hardware implementation of the design example using Libero.

Figure 6 • **Hardware Implementation of the Design**

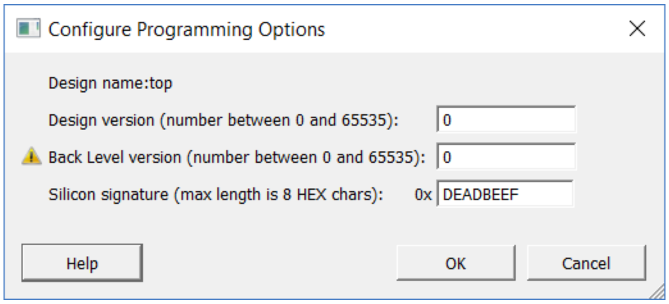


The design example also demonstrates the following system services:

- Fetch Device Serial Number system service
- Fetch JTAG USERCODE system service

The device serial number (DSN) is a 128-bit unique device ID set in the factory. JTAG USERCODE is a 32-bit user configurable silicon signature to be programmed into the device. It can be set using **Program Design** > **Configure Programming Options** available in the Libero Design Flow as shown in the following figure.

Figure 7 • **JTAG USERCODE or Silicon Signature Configuration**



Configure Programming Options

Design name:top

Design version (number between 0 and 65535): 0

Back Level version (number between 0 and 65535): 0

Silicon signature (max length is 8 HEX chars): 0x DEADBEEF

2.5 How to Run the Demo

This section provides instructions to run the demo of the design example.

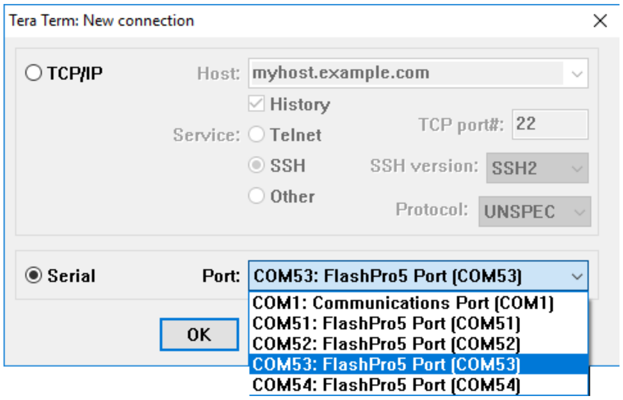
2.5.1 Tera Term Setup

The design example provides a user interface on the Tera Term terminal through the UART interface.

To set up the Tera Term program, perform the following steps:

1. Ensure that the USB cable connects the host PC to the J5 (USB) port on the PolarFire Evaluation board.
2. Start Tera Term.
3. Select **File** > **New connection…** from the Tera Term menu.
4. Select **Serial** as the Connection type.
5. Set the Serial **Port** to the second highest COM port number from the drop-down list as shown in the following figure. For example, **COM53: FlashPro5 Port [COM53]** in this case.

Figure 8 • **Select Serial Port**



Tera Term: New connection

○ TCP/IP Host: myhost.example.com

☑ History

Service: ○ Telnet TCP port#: 22

● SSH SSH version: SSH2

○ Other Protocol: UNSPEC

● Serial Port: COM53: FlashPro5 Port [COM53]

COM1: Communications Port (COM1)
COM51: FlashPro5 Port (COM51)
COM52: FlashPro5 Port (COM52)
COM53: FlashPro5 Port (COM53)
COM54: FlashPro5 Port (COM54)
OK

6. In the Tera Term window, go to Setup > **Serial port**..., set Baud rate as 115200.

This completes the Tera Term program setup.

2.5.2 Board Setup

To program the PolarFire device, perform the following steps:

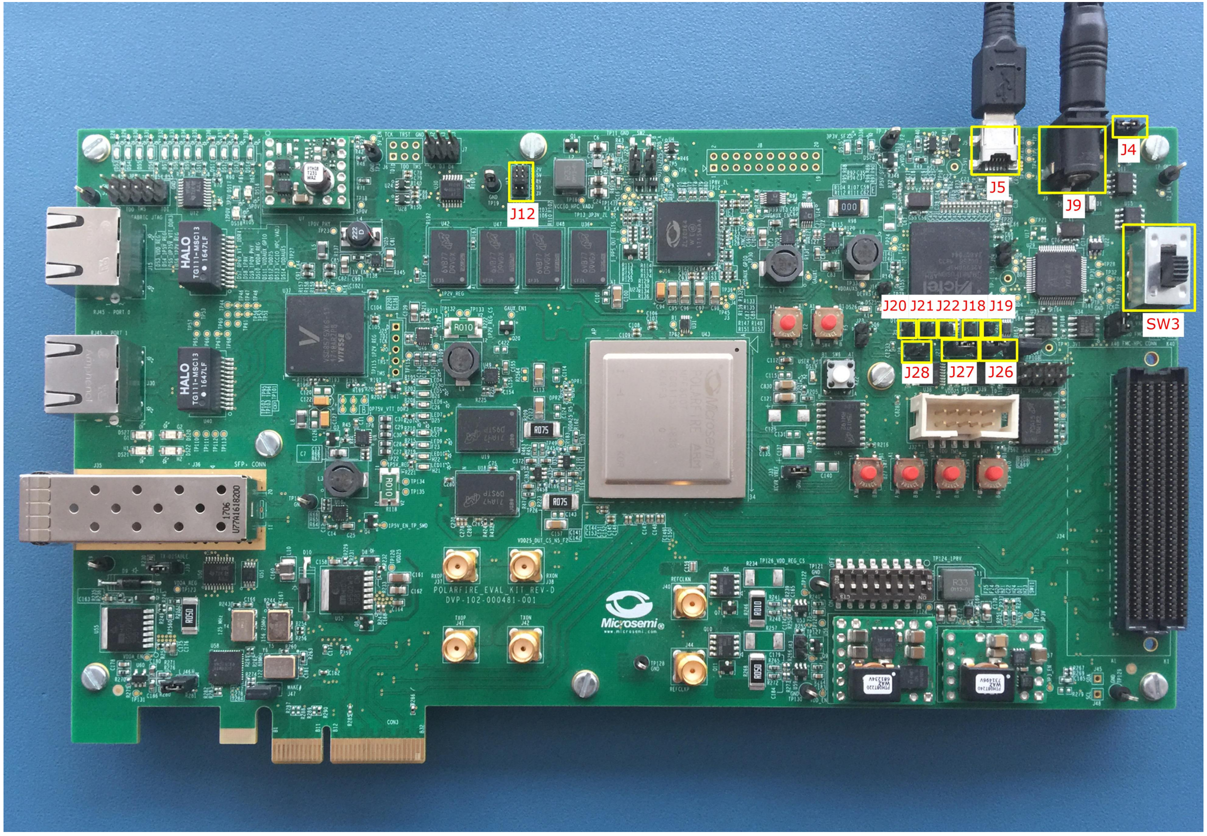
1. Ensure that the jumpers on the evaluation board are set as specified in the following table.

Table 3 • **Jumper Settings**

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-board FlashPro5
J26	Close pin 1 and 2 for programming through the FTDI SPI
J27	Close pin 1 and 2 for programming through the FTDI SPI
J23	Open pin 1 and 2 for programming SPI Flash
J4	Close pin 1 and 2 for manual power switching using SW3
J12	Close pin 3 and 4 for 2.5 V
SW11	DIP1 switch position should be set to ON.

2. Connect the power supply cable to the **J9** connector on the evaluation board.
3. Connect the USB cable from the host PC to **J5** (FTDI port) on the evaluation board.
4. Power on the evaluation board using the **SW3** slide switch.

Figure 9 • **Board Setup**



2.5.3 Program the Device

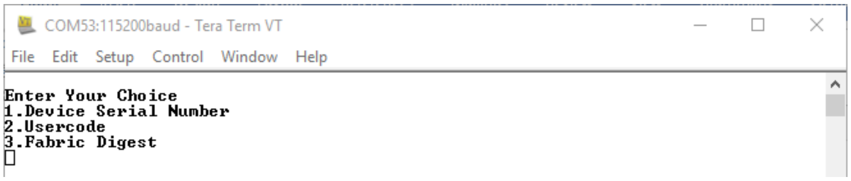
1. Open the provided Libero project.
2. Click **Run PROGRAM Action** to program the device.

Note: Users can use FlashPro Express to program the device using a job file without the need to open Libero. The programming job (.job) file for the demo design is available as part of the design files.

2.5.4 Running the Demo

After the device is programmed with the provided design. The design prints the menu on the Tera Term program through the UART interface, as shown in following figure. The state of a 4-bit free-running counter is shown using four user LEDs on the board.

Figure 10 • **User Interface Menu**

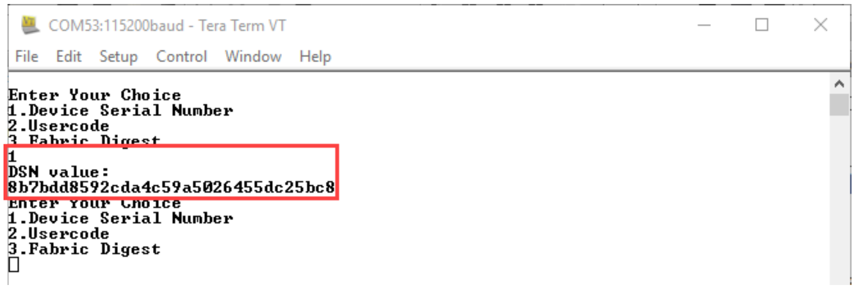


```
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
```

To run the demo, perform the following steps:

1. Press '1' to run device serial number (DSN) system service. As a result, the DSN of the device present on the evaluation board is printed on the terminal, as shown in the following figure. The DSN returned from the service is same as the DSN exported during device programming.

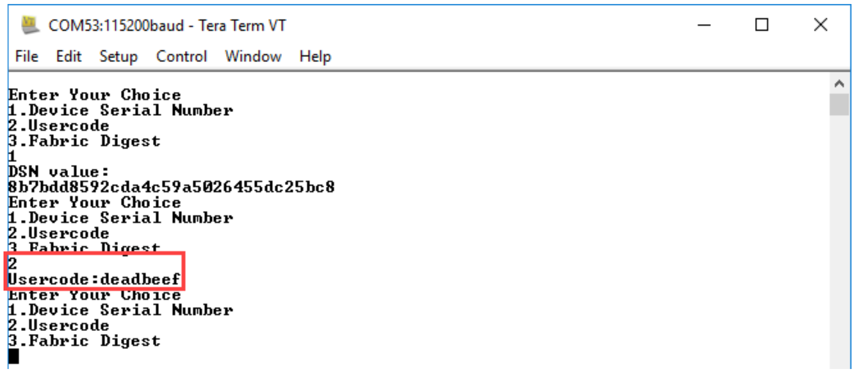
Figure 11 • **Fetch DSN System Service**



```
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
1
DSN value:
8b7bdd8592cda4c59a5026455dc25bc8
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
```

2. Press '2' to execute Fetch JTAG USERCODE system service. As a result, the USERCODE or silicon signature programmed in the device is printed on the terminal, as shown in the following figure.

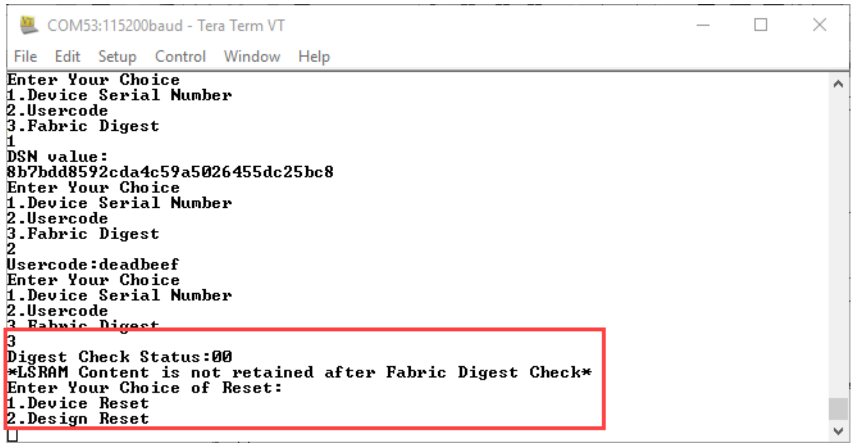
Figure 12 • **Fetch JTAG USERCODE System Service**



```
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
1
DSN value:
8b7bdd8592cda4c59a5026455dc25bc8
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
2
Usercode:deadbeef
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
```

3. Press '3' to perform digest check on FPGA fabric. Notice that the user LEDs on the board gets off while fabric digest check is being performed and then resume counting from previous state after exiting from suspend state.
   The status of the fabric digest check is returned by the service and it is printed on the terminal. Status code of 0x00 means no error in the fabric digest check.

Figure 13 • **Fabric Digest Check System Service**


COM53:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
1
DSN value:
8b7bdd8592cda4c59a5026455dc25bc8
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
2
Usercode:deadbeef
Enter Your Choice
1.Device Serial Number
2.Usercode
3.Fabric Digest
3
Digest Check Status:00
*LSRAM Content is not retained after Fabric Digest Check*
Enter Your Choice of Reset:
1.Device Reset
2.Design Reset
```

4. Press '1' to issue a device reset. Notice that the LEDs start counting from 0x0 since the counter gets reset. If you press '2' instead of '1' then design gets reset but the counter continues to count without reset. The rest of the design gets restarted in both the cases and terminal shows the start menu.

This concludes the demo of Fabric Digest Check system service.

3 Appendix: 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_ac482_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.