DG0471
Demo Guide
SmartFusion2 SoC FPGA In-System Programming Using USB OTG Controller Interface - Libero SoC v11.8

Microsemi®

Power Matters.™

Power Matters.™

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

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

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. 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 most current publication.

1.1 Revision 8.0

Updated the document for Libero v11.8 software release.

1.2 Revision 7.0

Updated the design files in the document for Libero v11.6 software release (SAR 72882).

1.3 Revision 6.0

Updated the design files in the document for Libero v11.5 software release (SAR 68427).

1.4 Revision 5.0

Updated the document for Libero v11.5 software release changes (SAR 65133).

1.5 Revision 4.0

Updated the document for Libero v11.4 software release (SAR 59740).

1.6 Revision 3.0

Updated the document for Libero v11.3 software release (SAR 56662).

1.7 Revision 2.0

Updated section Description, page 4 (SAR 53451).

1.8 Revision 1.0

Updated the document for Libero v11.2 software release (SAR 52963).

1.9 Revision 0

Initial release.

2 In-System Programming Using USB OTG Controller Interface

In-system programming (ISP) allows to reprogram the design iterations and field upgrades. The SmartFusion®2 devices support ISP using universal serial bus (USB) on-the-go (OTG) controller interface. This document describes how to program the following using ISP through the USB OTG controller interface:

- Embedded nonvolatile memory (eNVM)
- FPGA fabric
- Both the eNVM and the FPGA fabric

For information on different programming modes supported by SmartFusion2 SoC FPGAs, refer to the *UG0451: IGLOO2 and SmartFusion2 Programming User Guide*. For information on USB OTG controller, refer to the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

2.1 Design Requirements

The following table lists the hardware and software design requirements.

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware	
SmartFusion2 Advanced Development Kit: — 12 V adapter — FlashPro5 programmer — USB A to Mini-B cable	Rev A or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software	
Libero® System-on-Chip (SoC)	v11.8
SoftConsle	v4.0
FlashPro Programming Software	v11.8
Host PC Drivers	USB to UART

2.2 Demo Design

The demo design files are available for download at:
http://soc.microsemi.com/download/rsc/?f=m2s_dg0471_liberov11p8_df

The demo design files include:

- Sample programming files
- Libero SoC project
- STAPL programming file
- Source files
- readme.txt

The following figure shows the top-level structure of the design files. For further details, refer to the `readme.txt` file.

Figure 1 • **Demo Design Files Top-Level Structure**

```
<download_folder>
    |
    └── sf2_isp_using_usb_interface_demo_df
            ├── sample_programming_files
            ├── libero
            ├── stapl_programming_file
            ├── Sourcefiles
            └── readme.txt
```

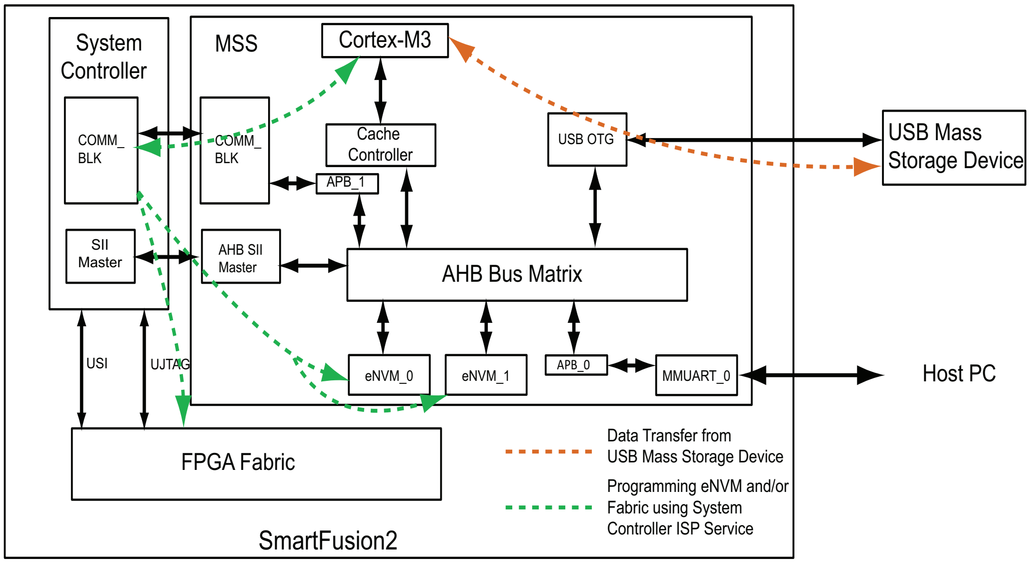
The following figure describes the top-level demo. The SmartFusion2 device application configures the following:

- The MMUART_0 peripheral for serial communication with a host PC.
- The USB OTG as mass storage class host, which can read or write files from the mass storage device connected to the SmartFusion2 device through USB cable with Micro A end. Refer to the Appendix: Hardware Project Implementation Settings, page 26.

The SmartFusion2 device also initializes the system controller to run the ISP service. The SmartFusion2 device detects the connected USB mass storage device and accesses the programming files. The Cortex-M3 processor reads 512 byte blocks of the programming file data from the USB mass storage device and sends the received blocks of data to the system controller ISP service.

The system controller ISP service executes the ISP operation in the requested mode and reports the status to the Cortex-M3 processor. Refer to the Description, page 4 for information on the various modes of operation.

Figure 2 • **Top Level Demo Diagram**



Connect the host PC to the SmartFusion2 device using the USB to UART (FTDI) interface.

2.2.1 Features

This demo design performs three types of programming based on the input provided by the programming file.

- **eNVM programming**: The ISP programming service programs only eNVM. In this case, the input programming file has only eNVM content.
- **FPGA fabric programming**: The ISP programming service programs only the FPGA fabric. In this case, the input programming file has only the FPGA fabric content.
- **eNVM and FPGA fabric programming**: The ISP programming service programs both the FPGA fabric and eNVM. In this case, the input programming file has both the FPGA fabric and eNVM content.

2.2.2 Description

The ISP in SmartFusion2 devices is performed by the Cortex-M3 processor and the system controller. The system controller manages the SmartFusion2 device programming and handles the system service requests. The SmartFusion2 device allows the Cortex-M3 processor to directly provide a bitstream to the system controller for programming. The Cortex-M3 processor initializes the system controller and receives the programming bitstream from the USB mass storage device through the USB OTG controller interface. The received bitstream is sent to the system controller to execute the ISP service in one of the following modes of operations:

- **Authenticate**: The system controller ISP service validates the integrity of the input data bitstream and reports the status information to the Cortex-M3 processor.
  - For security and reliability reasons, Microsemi recommends that the bitstream is authenticated before the program is executed using the Authenticate Operation mode. The SmartFusion2 device application must commit only the bitstream for programming after successful authentication and the integrity of the bitstream is validated.
- **Program**: The system controller ISP service programs the following depending on the input data bitstream:
  - eNVM
  - FPGA fabric
  - Both the eNVM and the FPGA fabric
- **Verify**: The system controller ISP service verifies the contents of the SmartFusion2 device against the input data bitstream and reports the status information to the Cortex-M3 processor.

The system controller ISP service utilizes the COMM_BLK interface to receive the entire programming data bitstream as a continuous stream of bytes. Refer to the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide* for more information on communication block (COMM_BLK).

The Cortex-M3 processor in the SmartFusion2 device can execute an application image from embedded SRAM (eSRAM), eNVM or DDR/SDR memories. Refer to the *AC390: SmartFusion2 SoC FPGA Remapping eNVM, eSRAM, and DDR/SDR SDRAM Memories Application Notes* for more information on remapping techniques. In this demo design, the Cortex-M3 processor executes the ISP application Appendix: Hardware Project Implementation Settings, page 26ming taking place, that is during Program operation mode. In order to execute the application image from eSRAM, the Cortex-M3 processor copies the ISP application image (resides in eNVM data client) to the eSRAM and remaps the eSRAM to the Cortex-M3 processor code region. For Verify and Authenticate operation modes, the application image can be executed from either eNVM or eSRAM since the eNVM programming is not taking place. Refer to the Appendix: Hardware Project Implementation Settings, page 26.

2.2.2.1 Programming Files

Sample programming files with the file extension `.spi` are provided to program:

- eNVM
- FPGA fabric
- Both the eNVM and the FPGA fabric

The folder *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files* contains the following sample programming files along with Libero design files.

- `envmonly.spi`: Programs only eNVM. The eNVM client has a simple message display program.
- `fabriconly.spi`: Programs only the FPGA fabric. The FPGA fabric has a light-emitting diode (LED) blinking logic.
- `fabenvm.spi`: Programs both the FPGA fabric and eNVM. The eNVM client has a message display program and the FPGA fabric has an LED blinking logic. The folder *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files\fabric_and_envm* contains the Libero design to generate this sample programming file.
- `isp_demo.spi`: This is the `.spi` file format version of `isp_demo.stp` file provided in *<download_folder>\sf2_isp_using_usb_interface_demo_df\stapl_programming_file*.

Note: For more information on generating `.spi` programming files refer to the Appendix: Generating .spi Programming File using Libero, page 23.

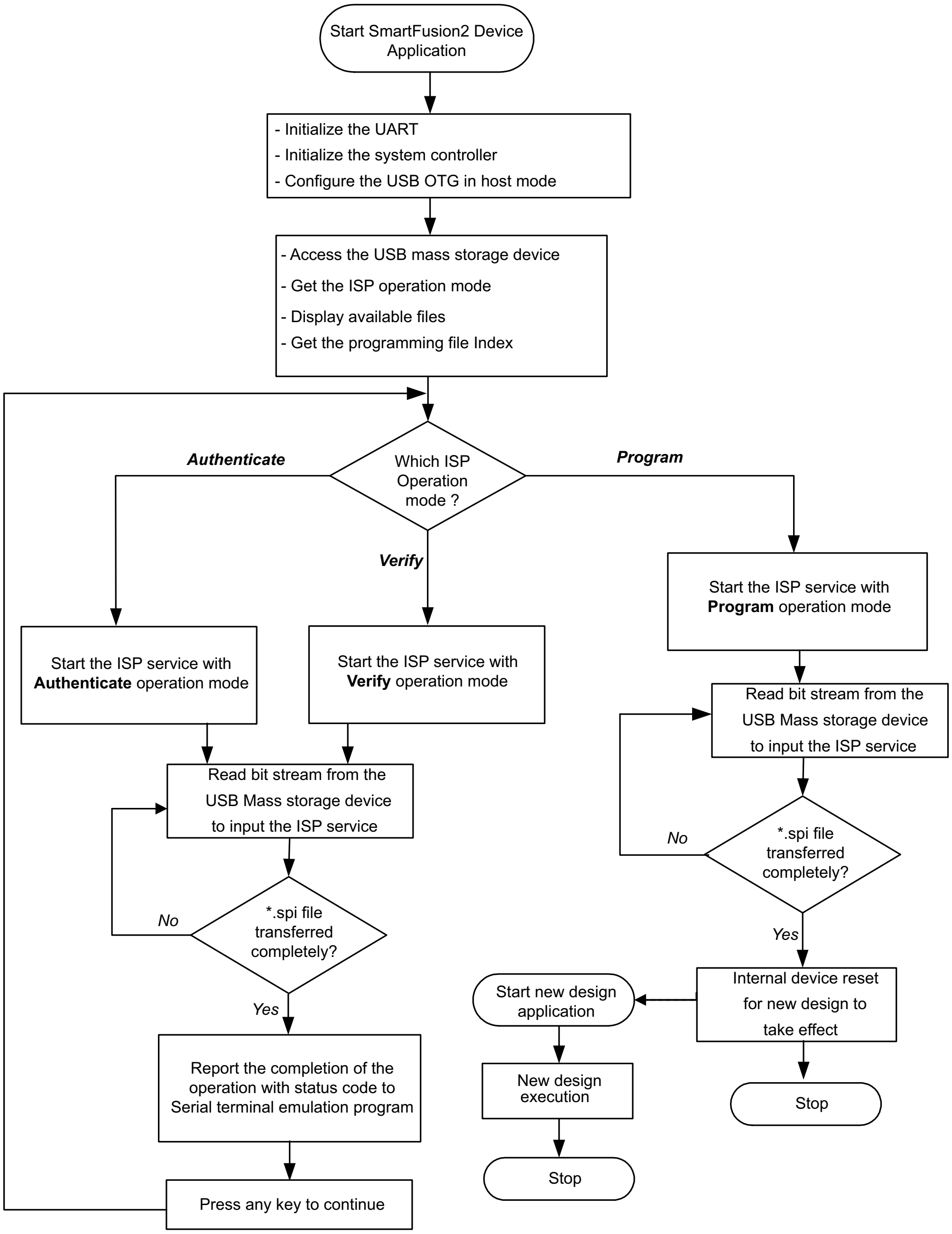
2.2.2.2 ISP Execution Flow

The SmartFusion2 device application initially configures the USB OTG controller in host mode and MMUART_0 for serial communication. It also initializes the system controller to start the ISP service in the selected operation mode.

On receiving the ISP operation mode and the programming file index, the application starts reading the input source programming file in a 512 byte blocks. The application stores the received data in a temporary buffer and sends the same data to the ISP service. The application requests the next block of 512 byte data until the entire file gets transferred from the USB mass storage device. The SmartFusion2 device application is notified with a status code when the ISP service completes the authentication or the verification process. When the operation mode is Program, an internal device reset is generated for the new design to take effect.

The following figure shows the ISP execution flow.

Figure 3 • **Execution Flow of ISP Operation Mode**

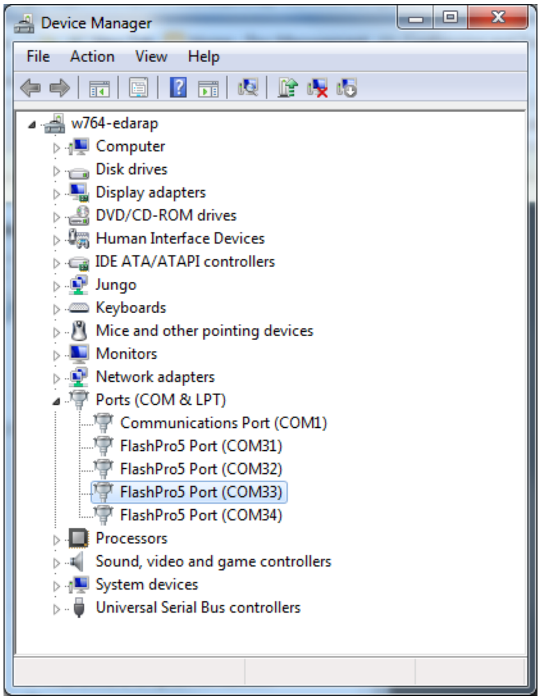

Start SmartFusion2 Device Application
- Initialize the UART
- Initialize the system controller
- Configure the USB OTG in host mode
- Access the USB mass storage device
- Get the ISP operation mode
- Display available files
- Get the programming file Index
Which ISP Operation mode ?
Authenticate
Verify
Program
Start the ISP service with Authenticate operation mode
Start the ISP service with Verify operation mode
Start the ISP service with Program operation mode
Read bit stream from the USB Mass storage device to input the ISP service
*.spi file transferred completely?
No
Yes
Report the completion of the operation with status code to Serial terminal emulation program
Press any key to continue
Read bit stream from the USB Mass storage device to input the ISP service
*.spi file transferred completely?
No
Yes
Internal device reset for new design to take effect
Start new design application
New design execution
Stop
Stop

2.3 Setting Up the Demo Design

The following steps describe how to setup the demo for SmartFusion2 Advanced Development Kit board:

1. Connect the host PC to the J18 Connector using the USB A to mini-B cable. The USB to UART bridge drivers are automatically detected. Download and install the drivers from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip* if the drivers are not installed or detected automatically.
   Verify if the detection is made in the device manager, as shown in the following figure.

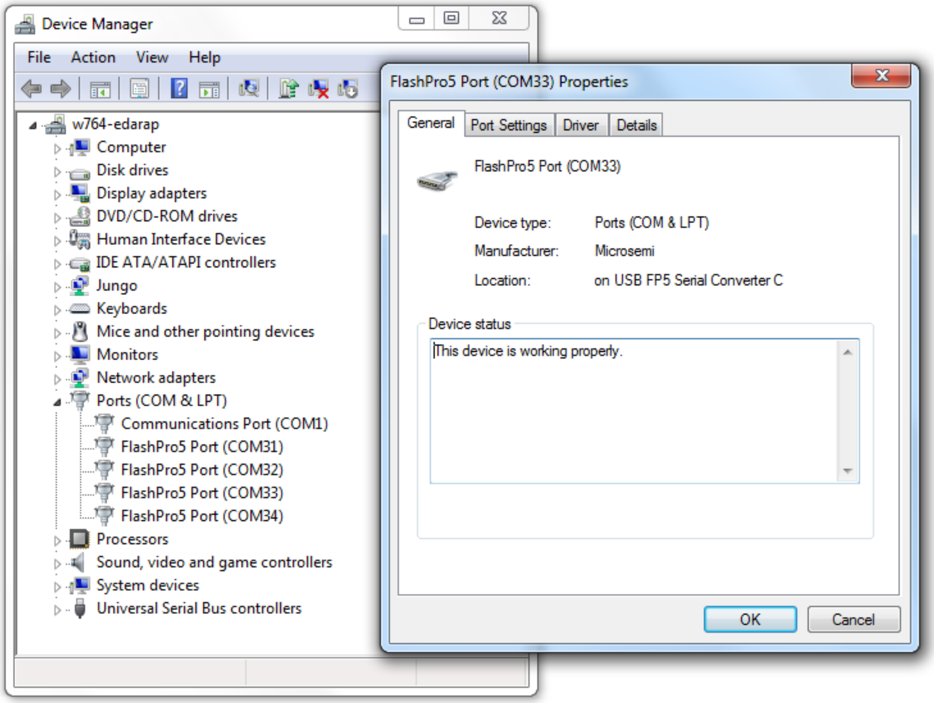
Figure 4 • **Device Manager**



w764-edarap
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Jungo
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Communications Port (COM1)
  - FlashPro5 Port (COM31)
  - FlashPro5 Port (COM32)
  - FlashPro5 Port (COM33)
  - FlashPro5 Port (COM34)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

2. Select one of the four COM ports with **Location** as on **USB FP5 Serial Converter C**. The following figure shows the **Device Manager** window and its properties that display the **USB Serial Port details**.The COM port number is required to run the demo design.

Figure 5 • **Device Manager - FlashPro5 (COM33) Properties**



FlashPro5 Port (COM33)

Device type: Ports (COM & LPT)

Manufacturer: Microsemi

Location: on USB FP5 Serial Converter C

Device status

This device is working properly.

3. Connect the jumpers on the SmartFusion2 Advanced Development Kit board as listed in the following table.

CAUTION: Switch **OFF** the power supply switch, **SW7** while connecting the jumpers.

Table 2 • **SmartFusion2 Advanced Development Kit Jumper Settings**

Jumper Number	Pin From	Pin To	Comments
J116, J353, J354, J54	1	2	These are the default jumper settings of the SmartFusion2 Advanced Development Kit board. Ensure these jumpers are set accordingly.
J123	2	3	
J124, J121, J32	1	2	JTAG programming via FTDI

4. Connect the power supply to the J6 connector on the SmartFusion2 Advanced Development Kit board.

2.4 Running the Demo Design

1. Download the demo design from:
   http://soc.microsemi.com/download/rsc/?f=m2s_dg0471_liberov11p8_df
2. Switch **ON** the SW7 power supply switch.
3. Start any serial terminal emulation program such as:
   - HyperTerminal
   - PuTTY
   - TeraTerm

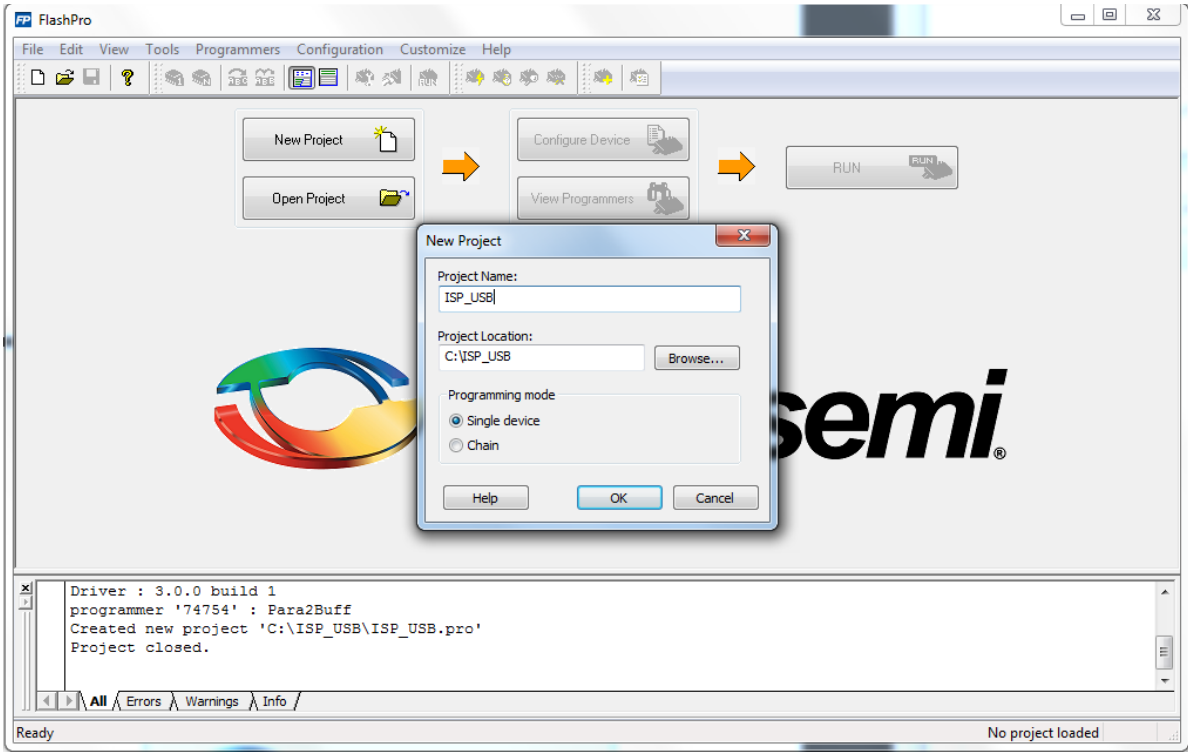
   The configuration for the program is:
   - Baud Rate: 57600
   - 8 Data bits
   - 1 Stop bit

  - No Parity
  - No Flow Control

  For information on configuring the serial terminal emulation programs, refer to the *Configuring Serial Terminal Emulation Programs Tutorial*.

4. Connect the USB cable Micro A end to the P1 connector of SmartFusion2 Advanced Development Kit board and other end to the USB mass storage device.
   Ensure to connect preformatted USB Flash drive to the SmartFusion2 device with the sample programming files provided in *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files*.
5. Launch the **FlashPro** software.
6. Click **New Project**.
7. In the **New Project** window, type the project name.

Figure 6 • **FlashPro New Project**



Project Name: ISP_USB

Project Location: C:\ISP_USB

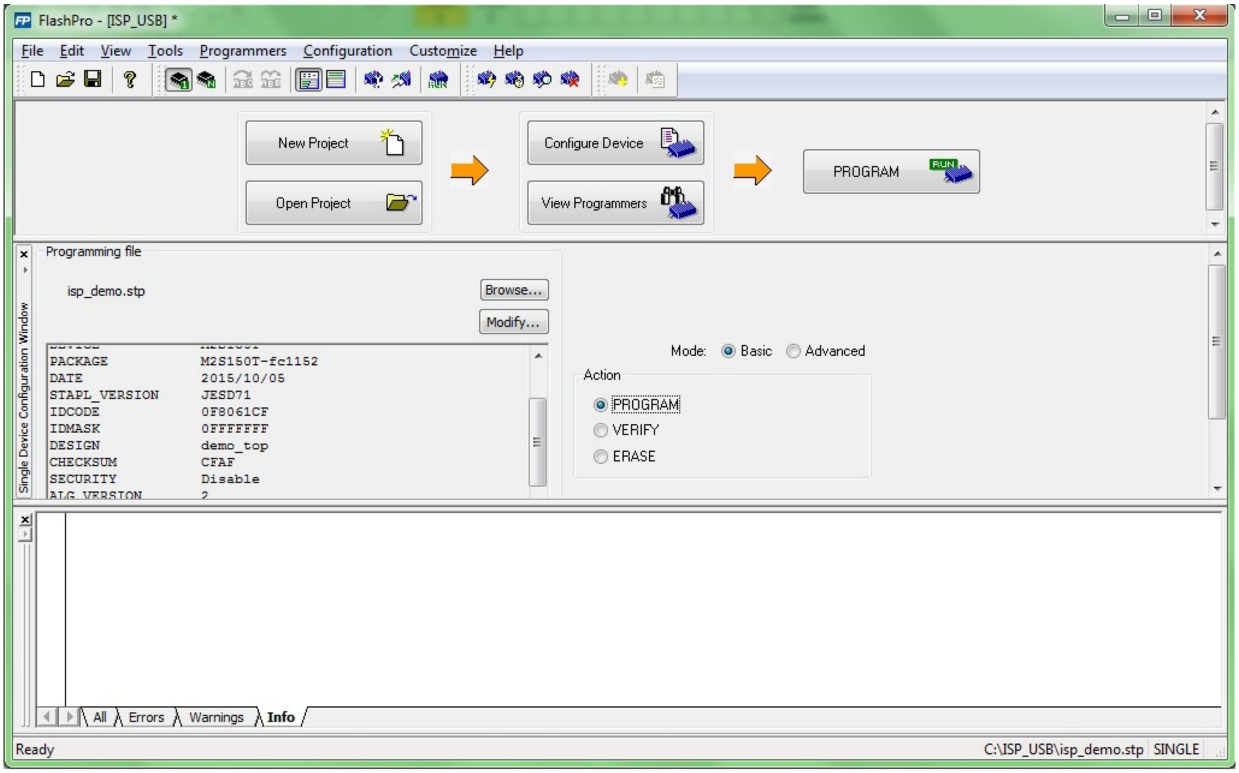
Programming mode
● Single device
○ Chain

```
Driver : 3.0.0 build 1
programmer '74754' : Para2Buff
Created new project 'C:\ISP_USB\ISP_USB.pro'
Project closed.
```

8. Click **Browse** and navigate to the location where you want to save the project.
9. Select **Single device** as the **Programming mode**.
10. Click **OK** to save the project.
11. Click **Configure Device** on the FlashPro GUI.

12. Click **Browse** and navigate to the location where the `isp_demo.stp` file is located and select the file. The default location is: *<download_folder>\sf2_isp_using_usb_interface_demo_df\stapl_programming_file.* The required programming file is selected and is ready to be programmed in the device.

Figure 7 • **FlashPro Project Configured**



Programming file

isp_demo.stp

```
PACKAGE             M2S150T-fc1152
DATE                2015/10/05
STAPL_VERSION       JESD71
IDCODE              0F8061CF
IDMASK              0FFFFFFF
DESIGN              demo_top
CHECKSUM            CFAF
SECURITY            Disable
ALG_VERSION         2
```

Mode: ● Basic ○ Advanced

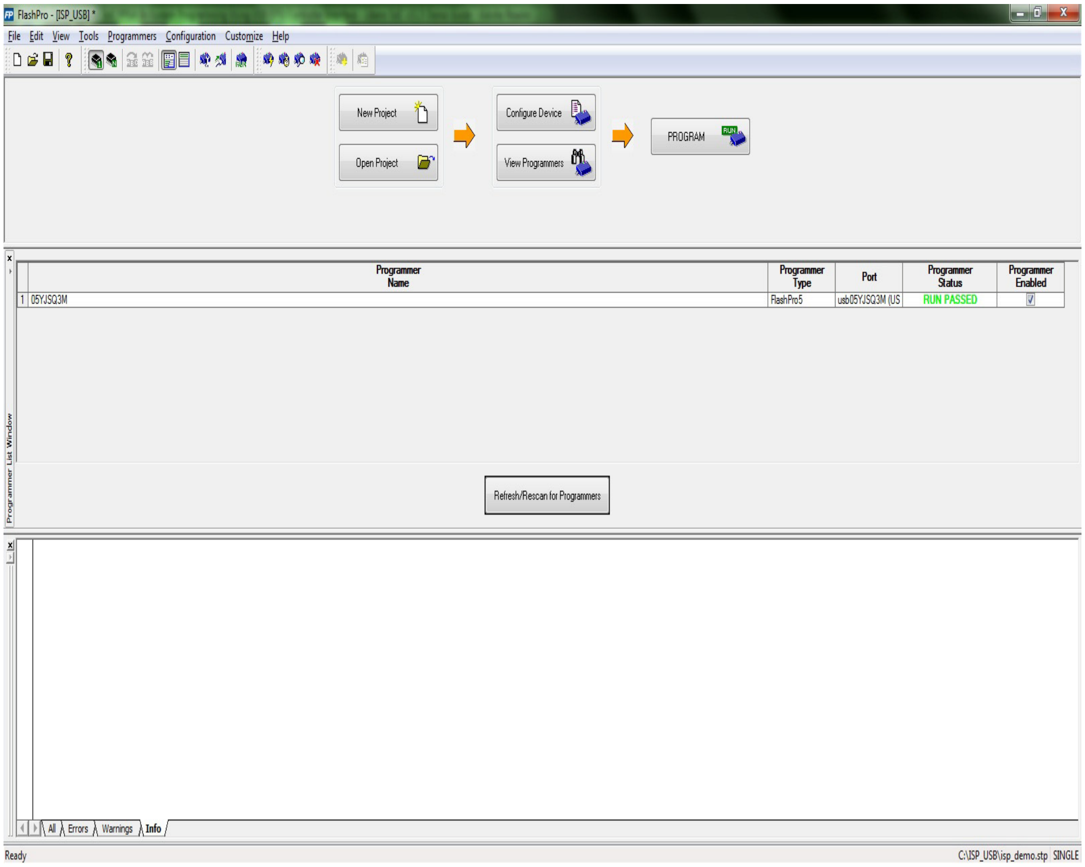
Action

● PROGRAM
○ VERIFY
○ ERASE

Ready   C:\ISP_USB\isp_demo.stp   SINGLE

13. Click **PROGRAM** to start programming the device. Wait until you get a message indicating that the program passed. ISP requires the SmartFusion2 device to be preprogrammed with the application code to activate the ISP service. Therefore, the SmartFusion2 device is preprogrammed with the `isp_demo.stp` using the FlashPro software.

Figure 8 • **FlashPro Program Passed**



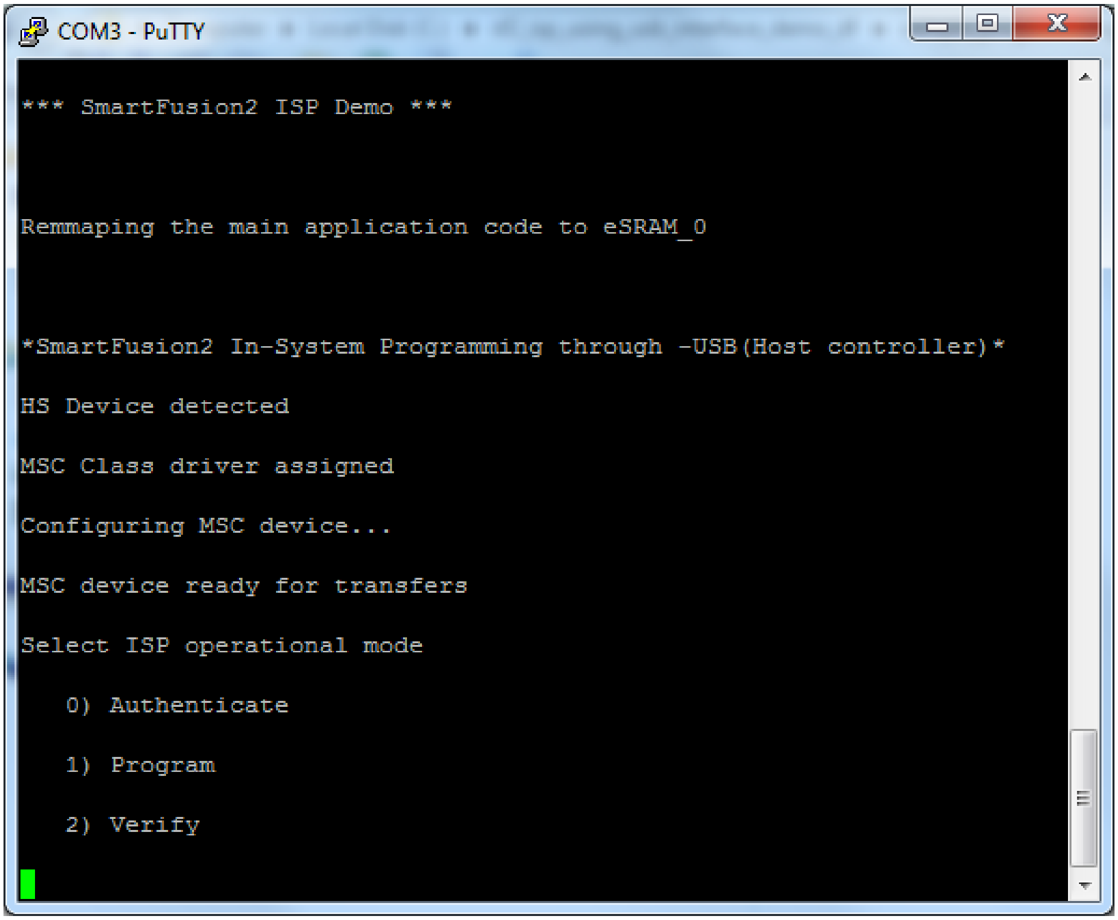
	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled
1	05YJSQ3M	FlashPro5	usb05YJSQ3M (US	RUN PASSED	☑

Ready C:\ISP_USB\isp_demo.stp SINGLE

- LEDs 4 to 7 (C26, A27, F26, D26) blinking in the board indicate that the SmartFusion2 device fabric is preprogrammed successfully.

On programming the SmartFusion2 device successfully using FlashPro, the serial terminal emulation program shows the initialization messages and ISP operation modes, as shown in the following figure.

Figure 9 • ISP Operation Modes Selection

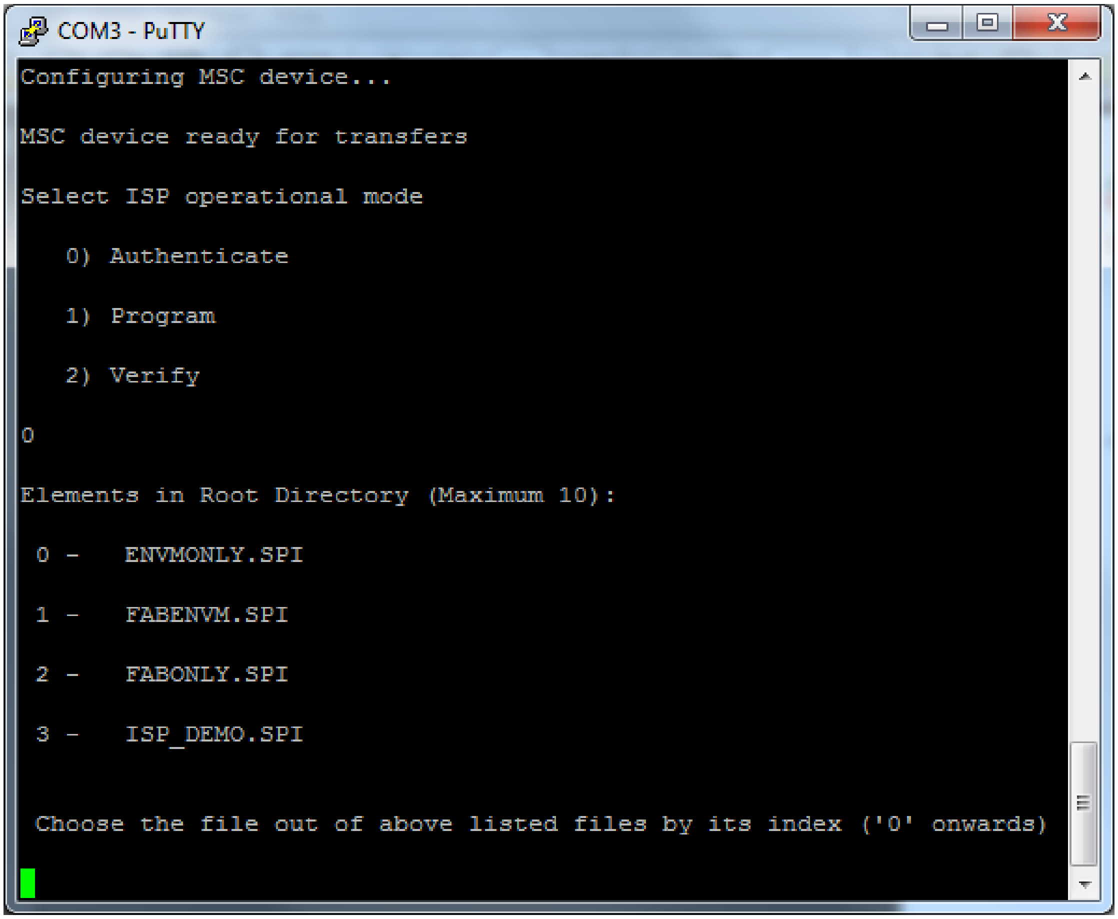


```
*** SmartFusion2 ISP Demo ***


Remmaping the main application code to eSRAM_0


*SmartFusion2 In-System Programming through -USB(Host controller)*

HS Device detected

MSC Class driver assigned

Configuring MSC device...

MSC device ready for transfers

Select ISP operational mode

   0) Authenticate

   1) Program

   2) Verify
```

On selecting the operation mode, the files in the USB storage device are displayed, as shown in the following figure.

Note: Maximum of 10 files can be shown from the USB storage device.

Figure 10 • **Available Files in USB Mass Storage Device**



```
Configuring MSC device...

MSC device ready for transfers

Select ISP operational mode

   0)  Authenticate

   1)  Program

   2)  Verify

0

Elements in Root Directory (Maximum 10):

 0 -    ENVMONLY.SPI

 1 -    FABENVM.SPI

 2 -    FABONLY.SPI

 3 -    ISP_DEMO.SPI


 Choose the file out of above listed files by its index ('0' onwards)
```

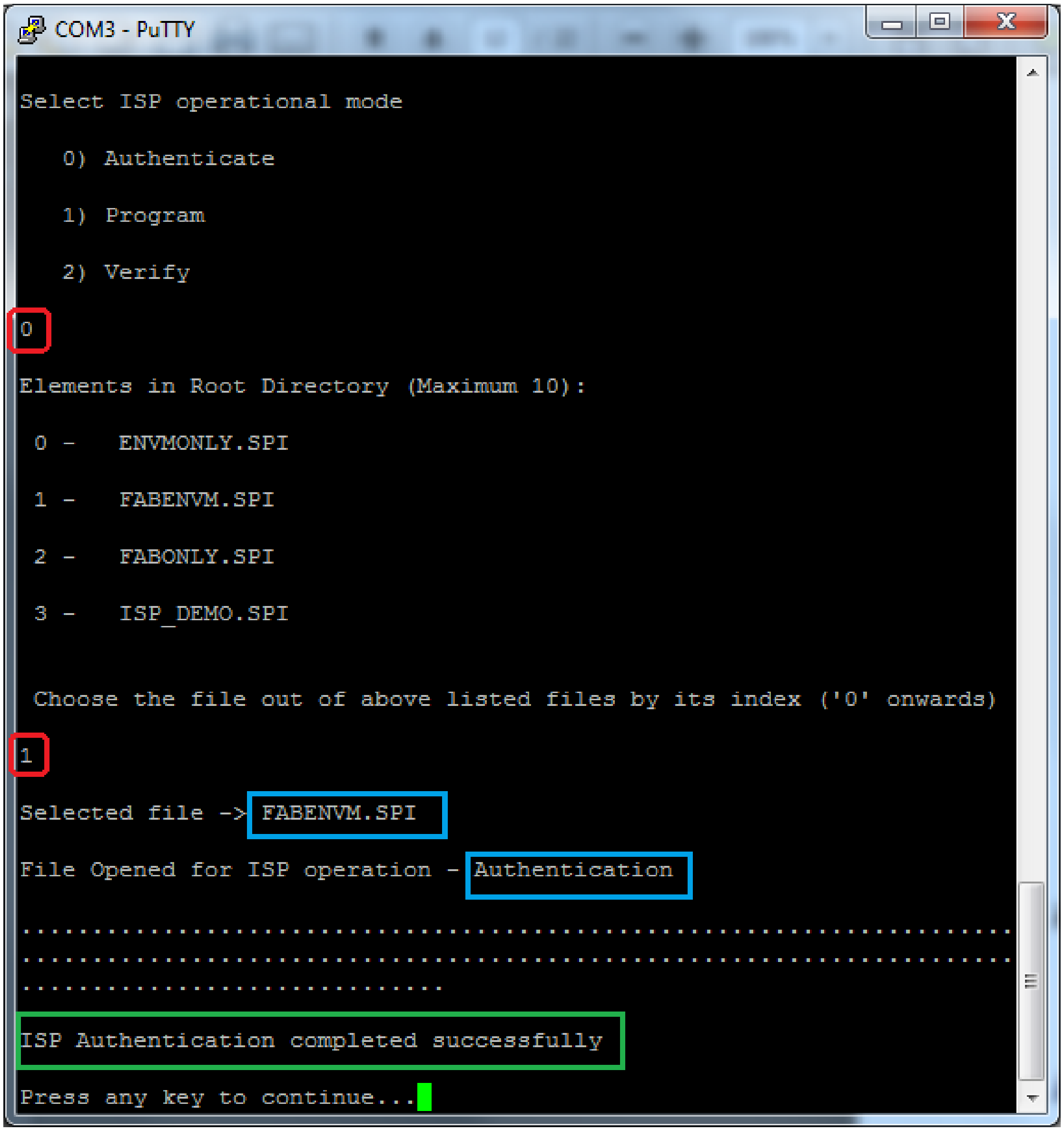
14. Select the programming file from the listed files by its index to perform the selected ISP operation mode.

2.5 Authenticate Operation Mode

To authenticate the data from `fabenvm.spi`, enter:

1. **0** to select **Authenticate** operation mode under **Select ISP operational mode**.
2. The corresponding index number to select `fabenvm.spi` programming file.
   On selecting the programming file, the application starts reading the programming file from USB mass storage device to execute the ISP operation mode. On completion of the ISP authentication, the serial terminal emulation program displays an operation success message. The following figure shows the operation success message.

Figure 11 • **ISP Authentication Results**



```
Select ISP operational mode

   0) Authenticate

   1) Program

   2) Verify

0

Elements in Root Directory (Maximum 10):

 0 -    ENVMONLY.SPI

 1 -    FABENVM.SPI

 2 -    FABONLY.SPI

 3 -    ISP_DEMO.SPI


 Choose the file out of above listed files by its index ('0' onwards)

1

Selected file -> FABENVM.SPI

File Opened for ISP operation - Authentication

......................................................................
......................................................................
..............................

ISP Authentication completed successfully

Press any key to continue...
```

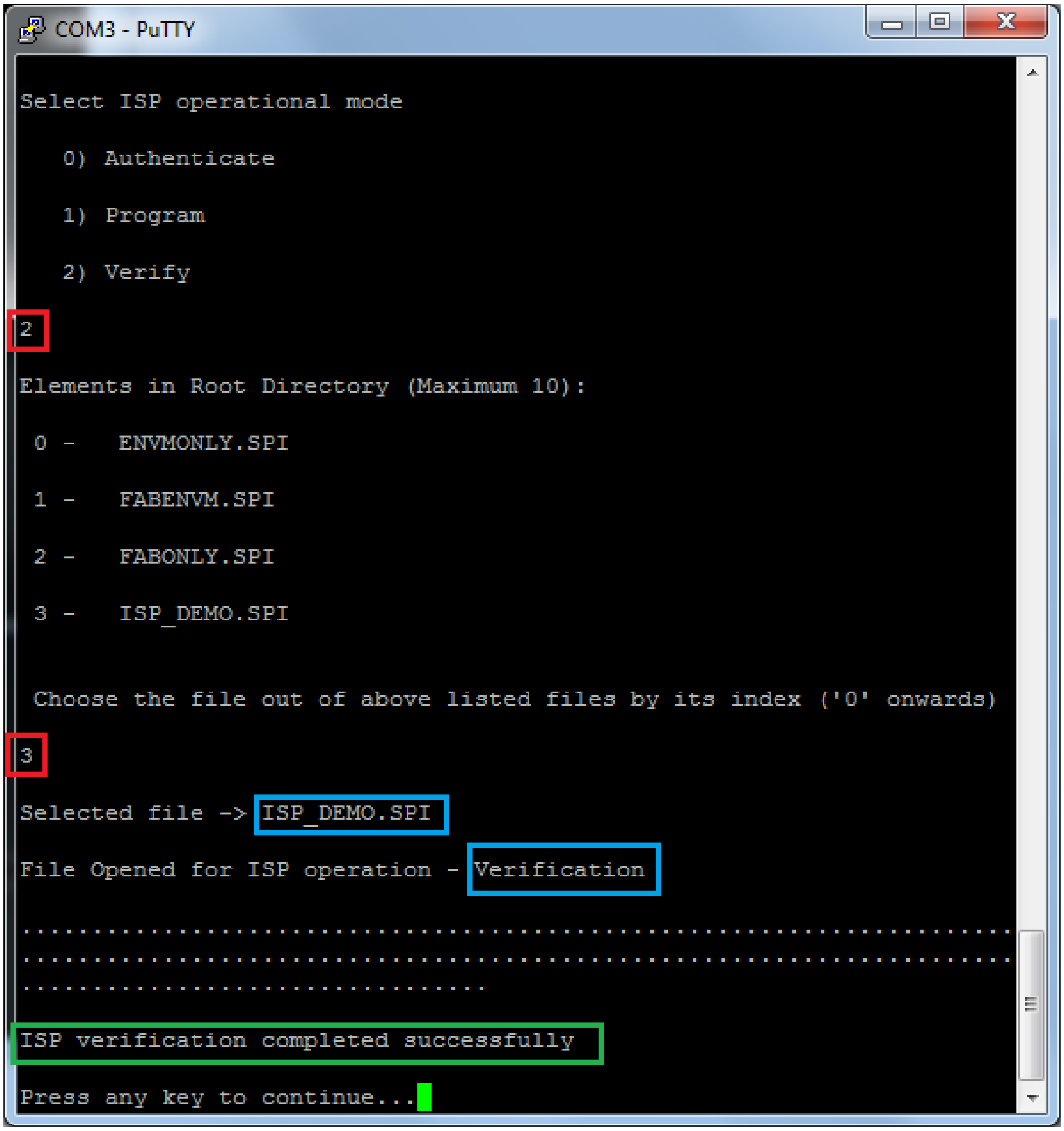
3. Press any key to continue for other ISP operation mode. If the USB storage device files are not displayed on serial terminal emulation program, press **SW6** to reset the board.

2.6 Verify Operation Mode

To verify the device FPGA fabric and eNVM contents, enter:

1. **2** to select **Verify** operation mode under **Select ISP operation mode**.
2. The corresponding index number to select `isp_demo.spi` programming file.
   On selecting the programming file, the application starts reading the programming file from USB mass storage device to execute the ISP operation mode. On completion of the ISP verification, the serial terminal emulation program displays an operation success message. The following figure shows the operation success message.

Figure 12 • **ISP Verification Results**



```
Select ISP operational mode

   0) Authenticate

   1) Program

   2) Verify

2

Elements in Root Directory (Maximum 10):

 0 -    ENVMONLY.SPI

 1 -    FABENVM.SPI

 2 -    FABONLY.SPI

 3 -    ISP_DEMO.SPI


 Choose the file out of above listed files by its index ('0' onwards)

3

Selected file -> ISP_DEMO.SPI

File Opened for ISP operation - Verification

......................................................................
......................................................................
.................................

ISP verification completed successfully

Press any key to continue...
```

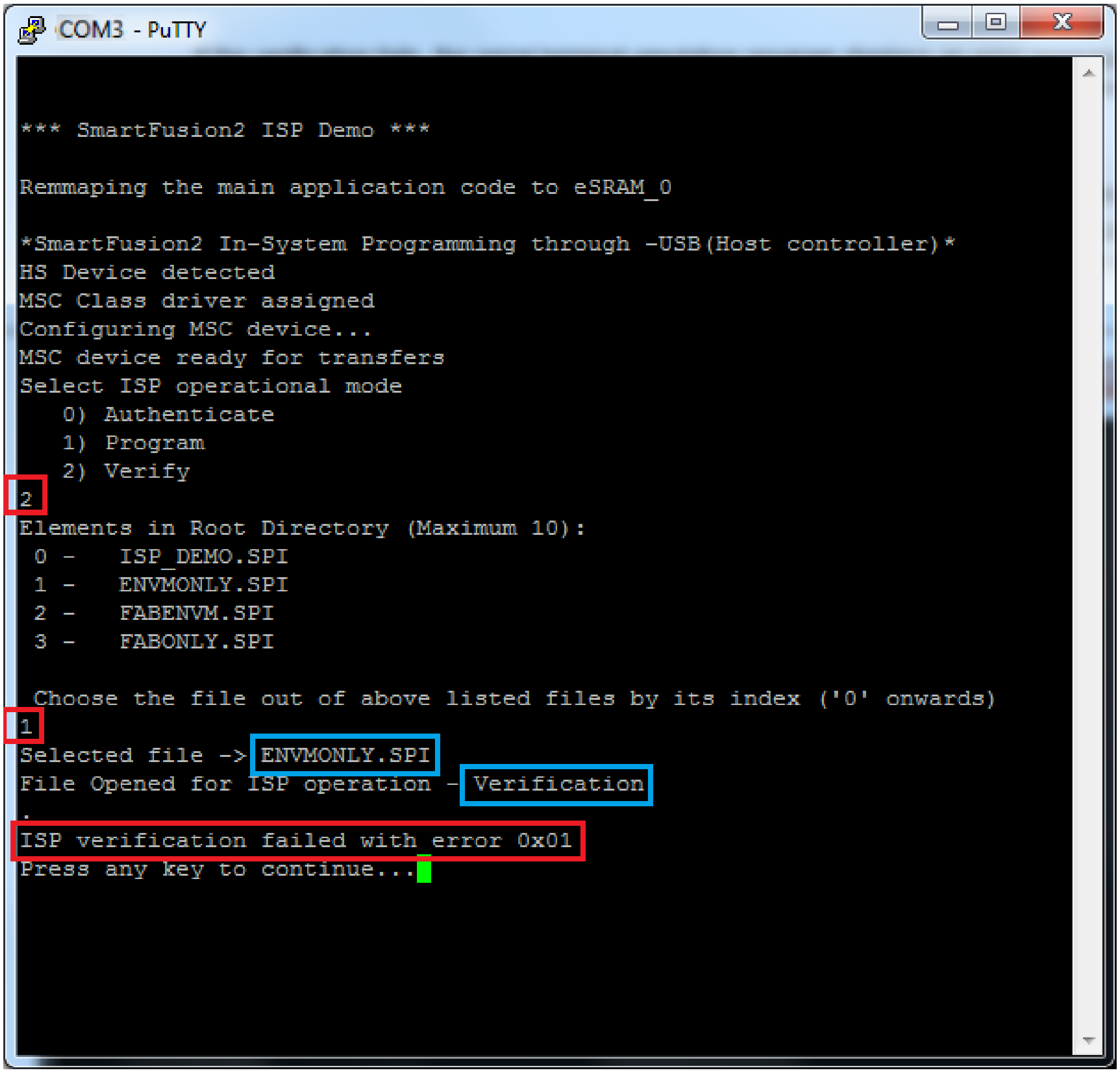
The verification operation demonstrated is for the `isp_demo.stp` file that is already running in the SmartFusion2 device. If any other `.spi` file is verified while the `isp_demo.stp` file is still running, that verification operation fails.

If the verification fails, the serial terminal emulation program displays an error message with an error code. The following figure shows an example error message. For more information on error codes, refer to the Appendix: Error Codes, page 22.

The programming files are at:
<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files.

All of them do not pass the verification. Only the `isp_demo.spi` file passes the verification operation as it matches with the SmartFusion2 device contents (`isp_demo.stp`). The other programming files fail verification.

Figure 13 • **ISP Verification Failure Error Message**



```
*** SmartFusion2 ISP Demo ***

Remmaping the main application code to eSRAM_0

*SmartFusion2 In-System Programming through -USB(Host controller)*
HS Device detected
MSC Class driver assigned
Configuring MSC device...
MSC device ready for transfers
Select ISP operational mode
   0) Authenticate
   1) Program
   2) Verify
2
Elements in Root Directory (Maximum 10):
 0 -   ISP_DEMO.SPI
 1 -   ENVMONLY.SPI
 2 -   FABENVM.SPI
 3 -   FABONLY.SPI

 Choose the file out of above listed files by its index ('0' onwards)
1
Selected file -> ENVMONLY.SPI
File Opened for ISP operation - Verification
.
ISP verification failed with error 0x01
Press any key to continue...
```

3. Press any key to continue for other ISP operation mode. If the USB storage device files are not displayed on the serial terminal emulation program, press **SW6** to reset the board.

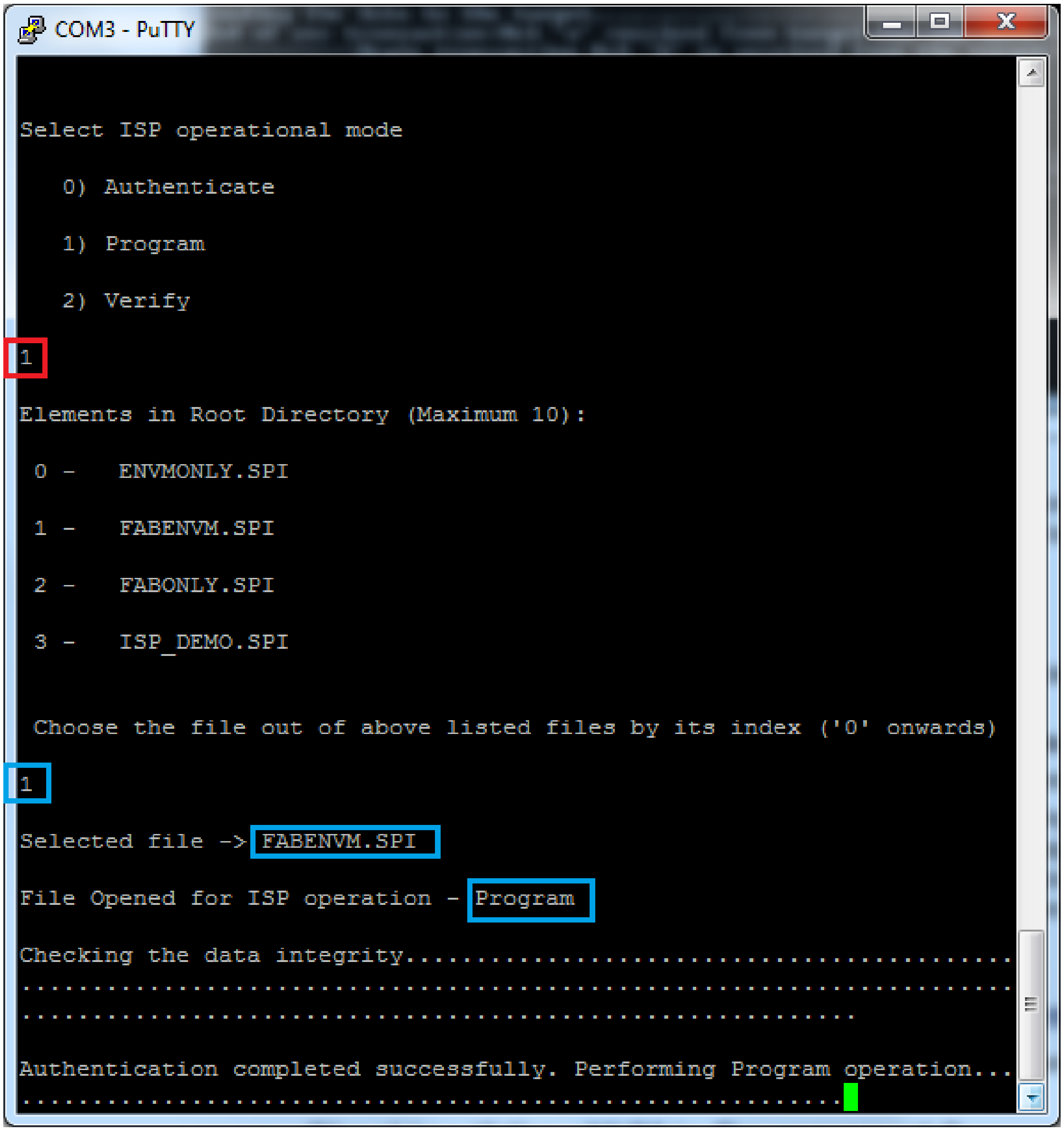
2.7 Program Operation Mode

To program the FPGA fabric and the eNVM of the SmartFusion2 device using the `fabenvm.spi` file, enter:

1. **1** to select **Program** operation mode under **Select ISP operation mode**.
2. The corresponding index number to select `fabenvm.spi` programming file.
   On selecting the programming file, the application starts reading the programming file from the USB mass storage device to execute the ISP operation mode. The application checks the data integrity of the selected programming file prior to perform the ISP program operation. After completing the programming operation, an internal reset is generated for the new design to take effect.

The following figure shows the selection of program operation mode for the `fabenvm.spi` programming file.

Figure 14 • **ISP Program Operation Mode**



```

Select ISP operational mode

   0) Authenticate

   1) Program

   2) Verify

1

Elements in Root Directory (Maximum 10):

 0 -    ENVMONLY.SPI

 1 -    FABENVM.SPI

 2 -    FABONLY.SPI

 3 -    ISP_DEMO.SPI


 Choose the file out of above listed files by its index ('0' onwards)

1

Selected file -> FABENVM.SPI

File Opened for ISP operation - Program

Checking the data integrity......................................................
.....................................................................
...........................................................

Authentication completed successfully. Performing Program operation...
..........................................................
```

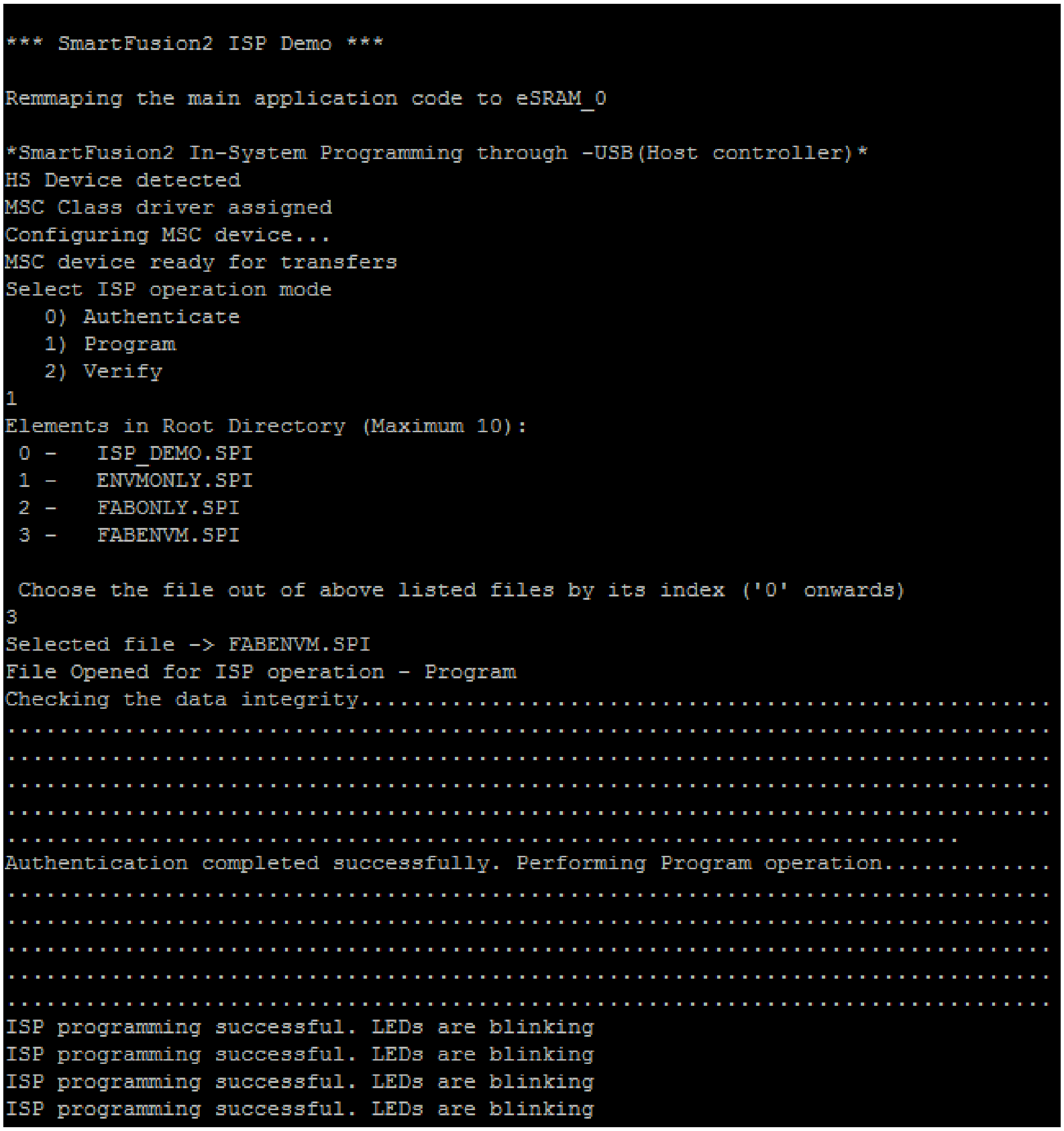
2.7.1 Checking if the Fabric is Programmed Successfully

LEDs 0 to 3 (C28, B27, C27, E26) blinking in the board indicate that the fabric is programmed successfully.

2.7.2 Checking if the eNVM is Programmed Successfully

The serial terminal emulation program displays the success message as shown in the following figure if the eNVM is programmed successfully.

Figure 15 • **ISP Program Results**



```
*** SmartFusion2 ISP Demo ***

Remmaping the main application code to eSRAM_0

*SmartFusion2 In-System Programming through -USB(Host controller)*
HS Device detected
MSC Class driver assigned
Configuring MSC device...
MSC device ready for transfers
Select ISP operation mode
   0) Authenticate
   1) Program
   2) Verify
1
Elements in Root Directory (Maximum 10):
 0 -   ISP_DEMO.SPI
 1 -   ENVMONLY.SPI
 2 -   FABONLY.SPI
 3 -   FABENVM.SPI

 Choose the file out of above listed files by its index ('0' onwards)
3
Selected file -> FABENVM.SPI
File Opened for ISP operation - Program
Checking the data integrity.....................................................
................................................................................
................................................................................
................................................................................
................................................................................
.........................................................................
Authentication completed successfully. Performing Program operation.............
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
ISP programming successful. LEDs are blinking
ISP programming successful. LEDs are blinking
ISP programming successful. LEDs are blinking
ISP programming successful. LEDs are blinking
```

2.7.3 Programming Results

The result shown in Figure 15, page 18 is for the fabenvm.spi file. The following table lists the possible results for ISP Program operation mode for sample programming files provided in folder *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files*. Not all .spi files listed in the table are demonstrated.

Table 3 • **ISP Programming Results**

*.spi Programming File Name	eNVM Programming Result	FPGA fabric Programming Result
envmonly.spi	The serial terminal emulation program shows successful eNVM program message	NA
fabonly.spi	NA	SmartFusion2 LEDs 0 to 3 (C28, B27, C27, E26) blinks
fabenvm.spi	The serial terminal emulation program shows successful eNVM program message	SmartFusion2 LEDs 0 to 3 (C28, B27, C27, E26) blinking

After successful ISP Program operation, the SmartFusion2 Advanced Development Kit board must be reprogrammed with the original isp_demo.stp file to try the ISP operation modes again.

2.8 Known Issue

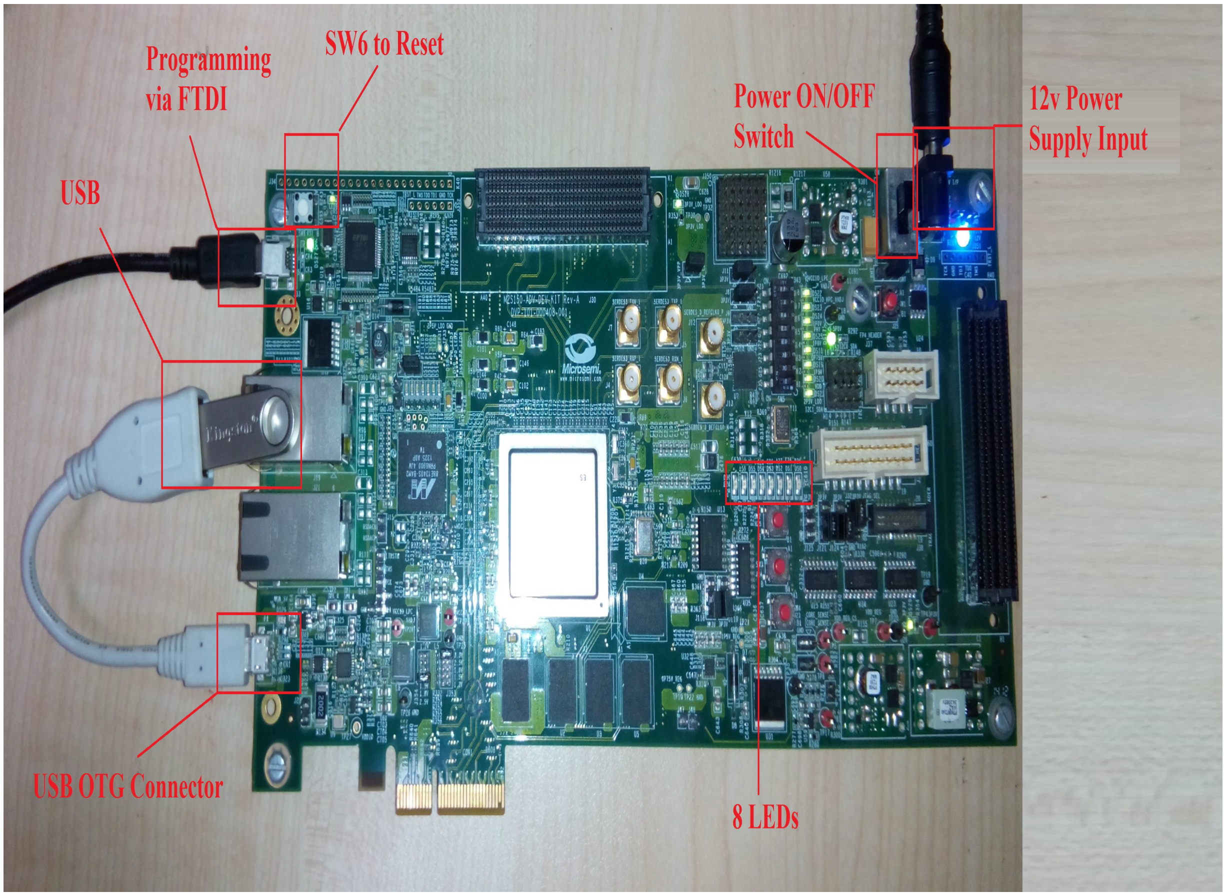
After successful completion of the two-step IAP or ISP, LSRAM read and write access fails from the fabric path. This is a known silicon issue, which is documented in the *ER0196: SmartFusion2 Device, Errata*. The workaround for this problem is to reset the system after the IAP or ISP program operation. Microsemi recommends that this workaround can be implemented for any design, which accesses LSRAM after IAP or ISP. For more information about how to implement this workaround, refer to the Appendix: Implementing Workaround to Access Fabric LSRAM after IAP or ISP Program Operation, page 29.

The design example provided in this demonstration implements the workaround for accessing LSRAM after implementing the IAP or ISP program operation in the Libero software. The design files are available in the following location: *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files\LSRAM_Workaround.*

3 Appendix: Board Setup for Running the Demo

The following figure shows the board setup through the USB to UART (FTDI) Interface using the USB A to Mini - B cable on the SmartFusion2 Advanced Development Kit board.

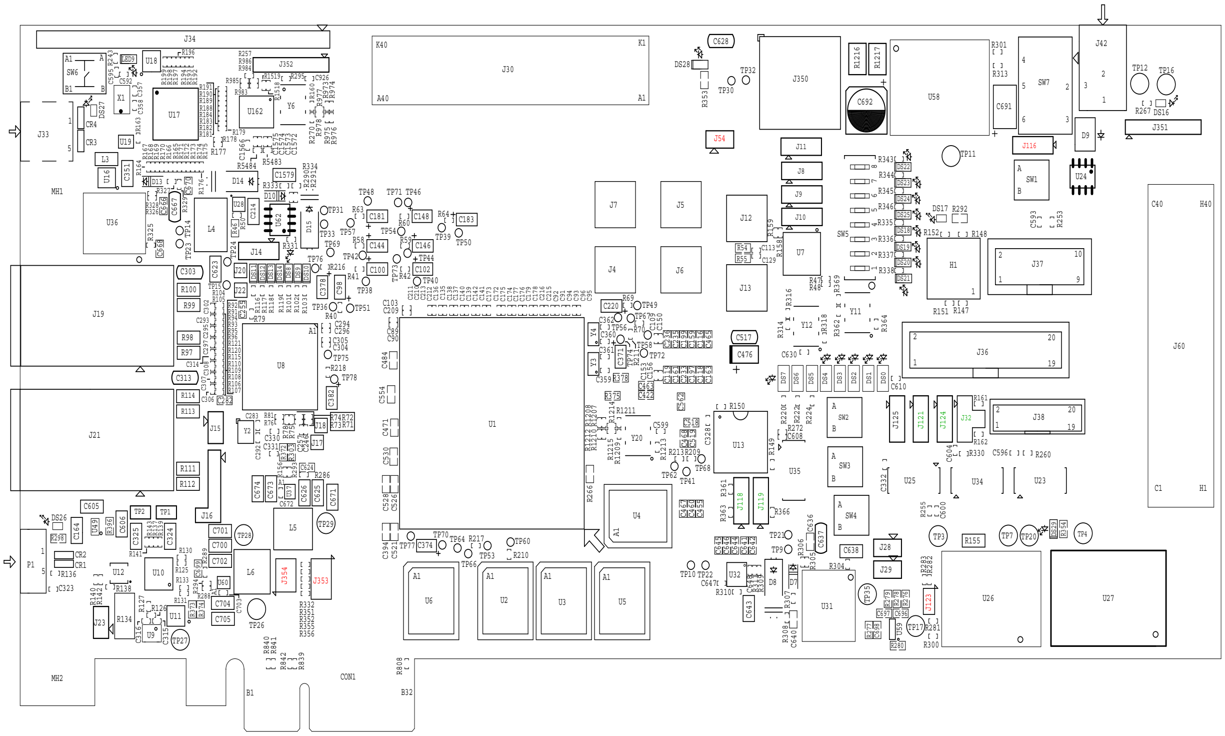
Figure 16 • Board Setup for Running the Demo



4 Appendix: Jumper Locations

The following figure shows the jumper locations in SmartFusion2 Advanced Development Kit board.

Figure 17 • **SmartFusion2 Advanced Development Kit Silkscreen Top View**



Note: Jumpers highlighted in red are set by default.

Note: The location of the jumpers in Figure 17, page 21 is searchable.

5 Appendix: Error Codes

The following table lists the error codes in SmartFusion2 Advanced Development Kit board.

Table 4 • **Error Codes**

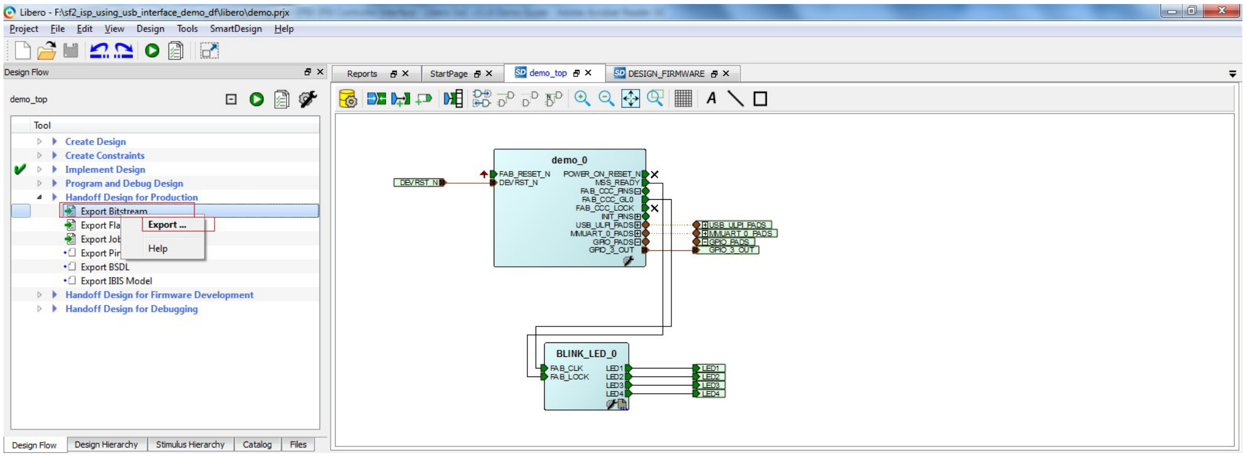
Define	Error Code	Description
#define MSS_SYS_CHAINING_MISMATCH	1u	Device contents mismatch
#define MSS_SYS_UNEXPECTED_DATA_RECEIVED	2u	Data is not supported
#define MSS_SYS_INVALID_ENCRYPTION_KEY	3u	Invalid encryption key
#define MSS_SYS_INVALID_COMPONENT_HEADER	4u	Invalid file header
#define MSS_SYS_BACK_LEVEL_NOT_SATISFIED	5u	corrupted /invalid bitstream
#define MSS_SYS_DSN_BINDING_MISMATCH	7u	corrupted /invalid bitstream
#define MSS_SYS_ILLEGAL_COMPONENT_SEQUENCE	8u	corrupted /invalid bitstream
#define MSS_SYS_INSUFFICIENT_DEV_CAPABILITIES	9u	Invalid Device capabilities
#define MSS_SYS_INCORRECT_DEVICE_ID	10u	Invalid Device id
#define MSS_SYS_UNSUPPORTED_BITSTREAM_PROT_VER	11u	bitstream is not supported
#define MSS_SYS_VERIFY_NOT_PERMITTED_ON_BITSTR	12u	Verification is not allowed for input bitstream
#define MSS_SYS_ABORT	127u	Operation aborted
#define MSS_SYS_NVM_VERIFY_FAILED	129u	eNVM verification failed
#define MSS_SYS_DEVICE_SECURITY_PROTECTED	130u	Device is secured
#define MSS_SYS_PROGRAMMING_MODE_NOT_ENABLED	131u	Programming mode is not enabled.

6 Appendix: Generating .spi Programming File using Libero

The following steps describe how to generate a `.spi` programming file using Libero:

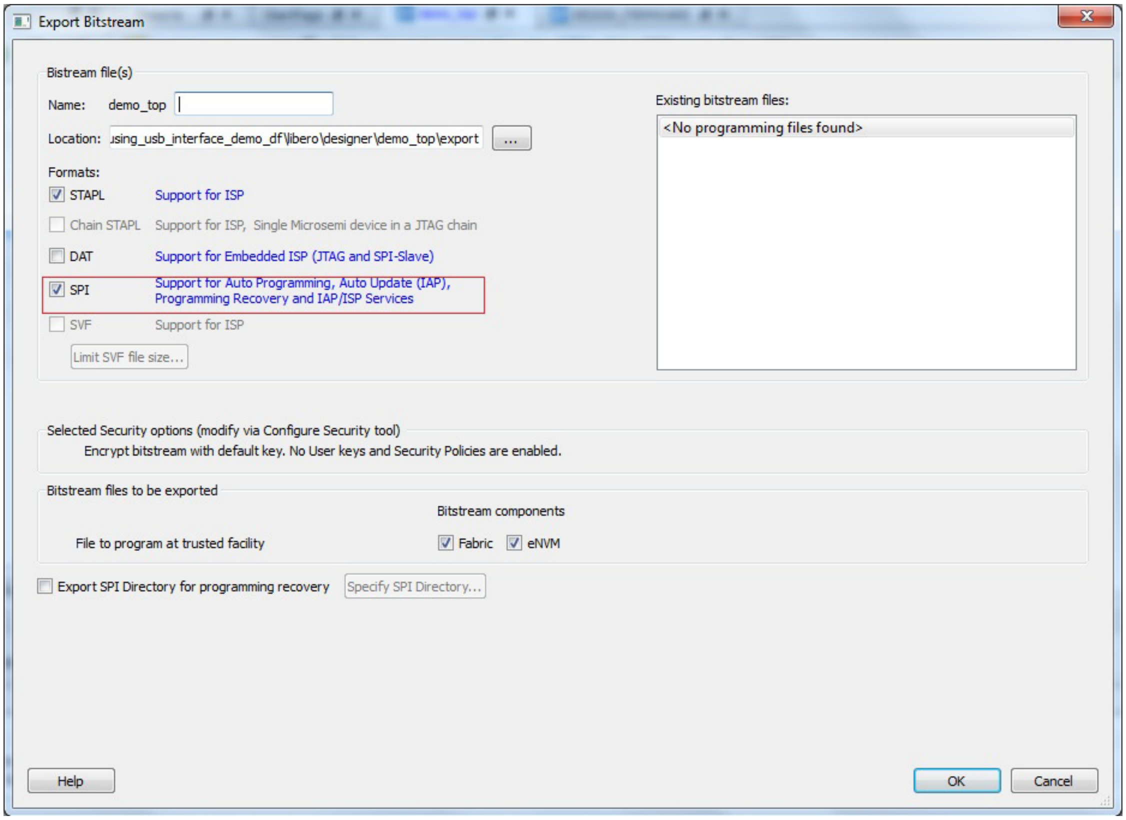
1. Launch the Libero SoC software to open a Libero project for `fabenvm.spi` programming file. The Libero design file is provided in *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_file\fabric_and_envm.*
2. Right-click **Bitstream** under **Handoff Design for Production** in the **Design Flow** tab, and click **Export...** from the context menu.

Figure 18 • **Configuring Export Bitstream**



3. On the **Export Bitstream** window, select the **SPI file** check box.

Figure 19 • **Export Programming File Options window**



Export Bitstream

Bistream file(s)

Name: demo_top

Location: ısing_usb_interface_demo_df\libero\designer\demo_top\export

Formats:

- ☑ STAPL Support for ISP
- □ Chain STAPL Support for ISP, Single Microsemi device in a JTAG chain
- □ DAT Support for Embedded ISP (JTAG and SPI-Slave)
- ☑ SPI Support for Auto Programming, Auto Update (IAP), Programming Recovery and IAP/ISP Services
- □ SVF Support for ISP

Existing bitstream files:

<No programming files found>

Selected Security options (modify via Configure Security tool)

Encrypt bitstream with default key. No User keys and Security Policies are enabled.

Bitstream files to be exported

Bitstream components

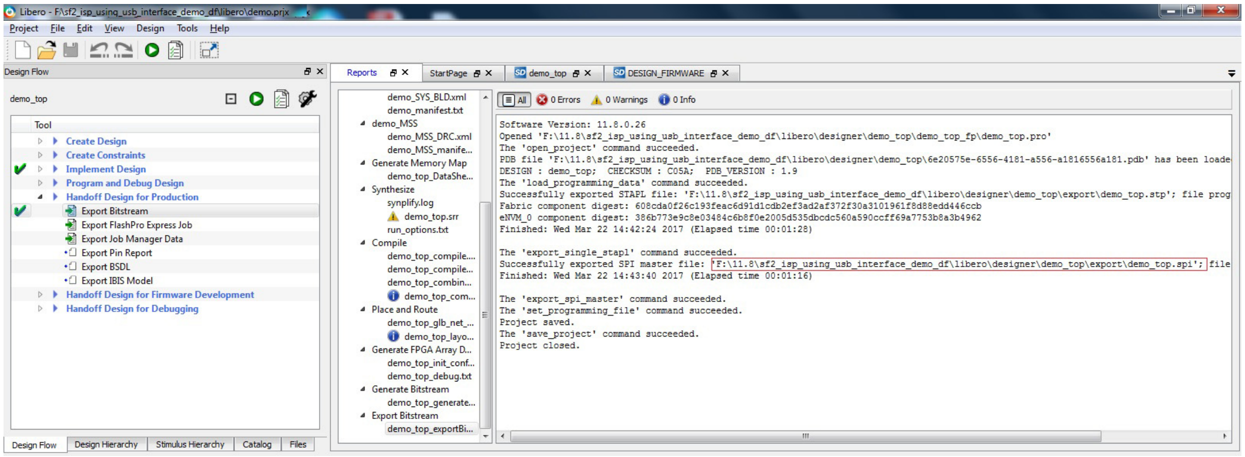
File to program at trusted facility ☑ Fabric ☑ eNVM

□ Export SPI Directory for programming recovery

4. Click **OK**.

5. Double-click **Export Bitstream** under **Handoff Design for Production** in the **Design Flow** tab to generate the `.spi` file (Figure 18, page 23). The following figure shows the `.spi` file location in **Reports** tab.

Figure 20 • **.SPI File Location**



```
Software Version: 11.8.0.26
Opened 'F:\11.8\sf2_isp_using_usb_interface_demo_df\libero\designer\demo_top\demo_top_fp\demo_top.pro'
The 'open_project' command succeeded.
PDB file 'F:\11.8\sf2_isp_using_usb_interface_demo_df\libero\designer\demo_top\6e20575e-6556-4181-a556-a1816556a181.pdb' has been loade
DESIGN : demo_top;  CHECKSUM : C05A;  PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
Successfully exported STAPL file: 'F:\11.8\sf2_isp_using_usb_interface_demo_df\libero\designer\demo_top\export\demo_top.stp'; file prog
Fabric component digest: 608cda0f26c193feac6d91d1cdb2ef3ad2af372f30a3101961f8d88edd446ccb
eNVM_0 component digest: 386b773e9c8e03484c6b8f0e2005d535dbcdc560a590ccff69a7753b8a3b4962
Finished: Wed Mar 22 14:42:24 2017 (Elapsed time 00:01:28)

The 'export_single_stapl' command succeeded.
Successfully exported SPI master file: 'F:\11.8\sf2_isp_using_usb_interface_demo_df\libero\designer\demo_top\export\demo_top.spi'; file
Finished: Wed Mar 22 14:43:40 2017 (Elapsed time 00:01:16)

The 'export_spi_master' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
```

7 Appendix: Hardware Project Implementation Settings

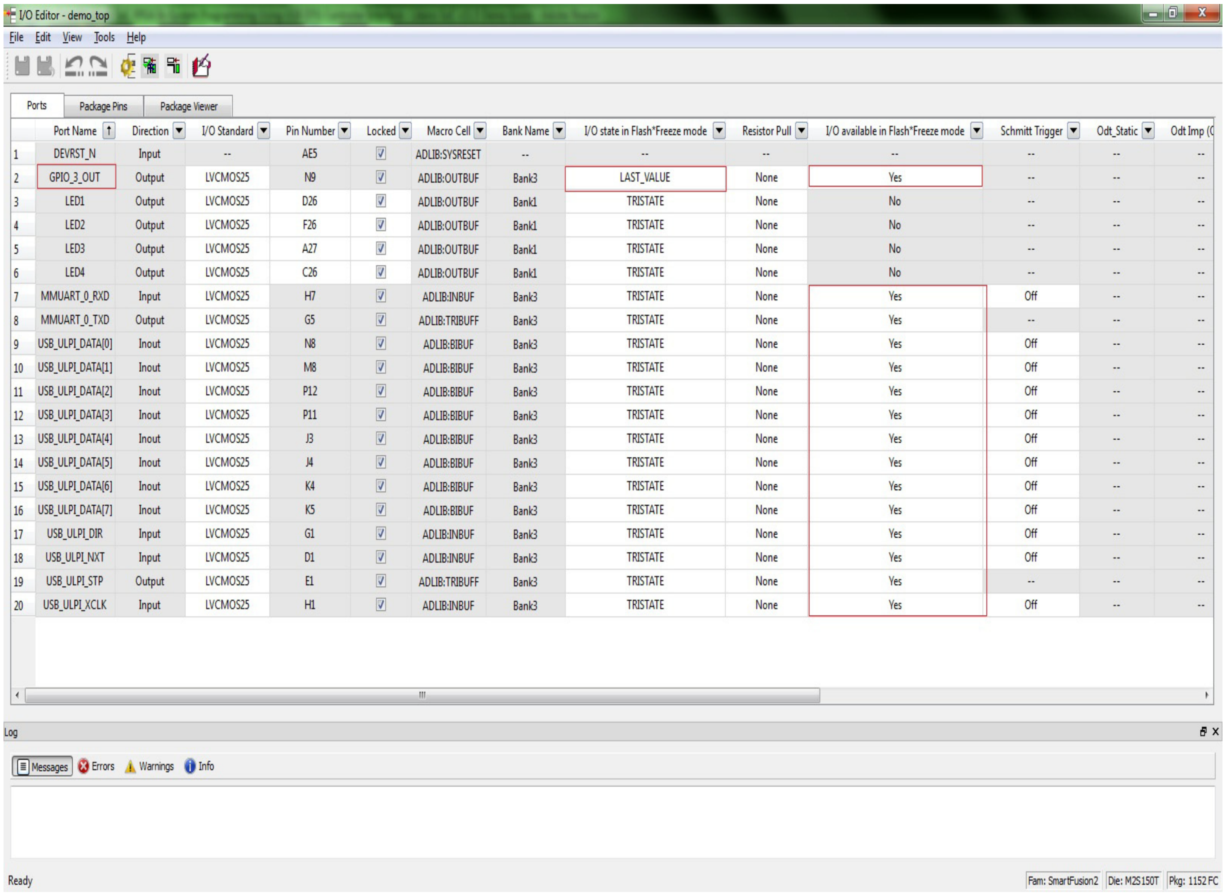
The following hardware project settings are required to build the demo design:

- Configuring the I/Os for Flash*Freeze Mode
- Standby Clock Source Configuration
- SoftConsole Project Generation

7.1 Configuring the I/Os for Flash*Freeze Mode

The Libero demo design configures M3_CLK to operate at 50 MHz, one UART interface (MMUART_0) for serial communication and USB OTG as mass storage class host. The FPGA fabric is not operational during Program or Verify operation as the device enters into Flash*Freeze (F*F). During F*F mode, the fabric and I/Os are not available. Therefore, the MMUART_0's RXD and TXD ports are configured using the I/O Editor to be available during F*F mode, as shown in the following figure. The reset of OTG controller reset is driven by GPIO_3_OUT through fabric. This I/O is configured to hold the last value during F*F mode.The user has to **Commit and Check** the settings from the File menu after configuring the ports.

Figure 21 • Configuring MMUART_0 Ports to be Available During F*F



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Schmitt Trigger	Odt_Static	Odt Imp (
1	DEVRST_N	Input	--	AE5	☑	ADLIB:SYSRESET	--	--	--	--	--	--	--
2	GPIO_3_OUT	Output	LVCMOS25	N9	☑	ADLIB:OUTBUF	Bank3	LAST_VALUE	None	Yes	--	--	--
3	LED1	Output	LVCMOS25	D26	☑	ADLIB:OUTBUF	Bank1	TRISTATE	None	No	--	--	--
4	LED2	Output	LVCMOS25	F26	☑	ADLIB:OUTBUF	Bank1	TRISTATE	None	No	--	--	--
5	LED3	Output	LVCMOS25	A27	☑	ADLIB:OUTBUF	Bank1	TRISTATE	None	No	--	--	--
6	LED4	Output	LVCMOS25	C26	☑	ADLIB:OUTBUF	Bank1	TRISTATE	None	No	--	--	--
7	MMUART_0_RXD	Input	LVCMOS25	H7	☑	ADLIB:INBUF	Bank3	TRISTATE	None	Yes	Off	--	--
8	MMUART_0_TXD	Output	LVCMOS25	G5	☑	ADLIB:TRIBUFF	Bank3	TRISTATE	None	Yes	--	--	--
9	USB_ULPI_DATA[0]	Inout	LVCMOS25	N8	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
10	USB_ULPI_DATA[1]	Inout	LVCMOS25	M8	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
11	USB_ULPI_DATA[2]	Inout	LVCMOS25	P12	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
12	USB_ULPI_DATA[3]	Inout	LVCMOS25	P11	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
13	USB_ULPI_DATA[4]	Inout	LVCMOS25	J3	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
14	USB_ULPI_DATA[5]	Inout	LVCMOS25	J4	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
15	USB_ULPI_DATA[6]	Inout	LVCMOS25	K4	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
16	USB_ULPI_DATA[7]	Inout	LVCMOS25	K5	☑	ADLIB:BIBUF	Bank3	TRISTATE	None	Yes	Off	--	--
17	USB_ULPI_DIR	Input	LVCMOS25	G1	☑	ADLIB:INBUF	Bank3	TRISTATE	None	Yes	Off	--	--
18	USB_ULPI_NXT	Input	LVCMOS25	D1	☑	ADLIB:INBUF	Bank3	TRISTATE	None	Yes	Off	--	--
19	USB_ULPI_STP	Output	LVCMOS25	E1	☑	ADLIB:TRIBUFF	Bank3	TRISTATE	None	Yes	--	--	--
20	USB_ULPI_XCLK	Input	LVCMOS25	H1	☑	ADLIB:INBUF	Bank3	TRISTATE	None	Yes	Off	--	--

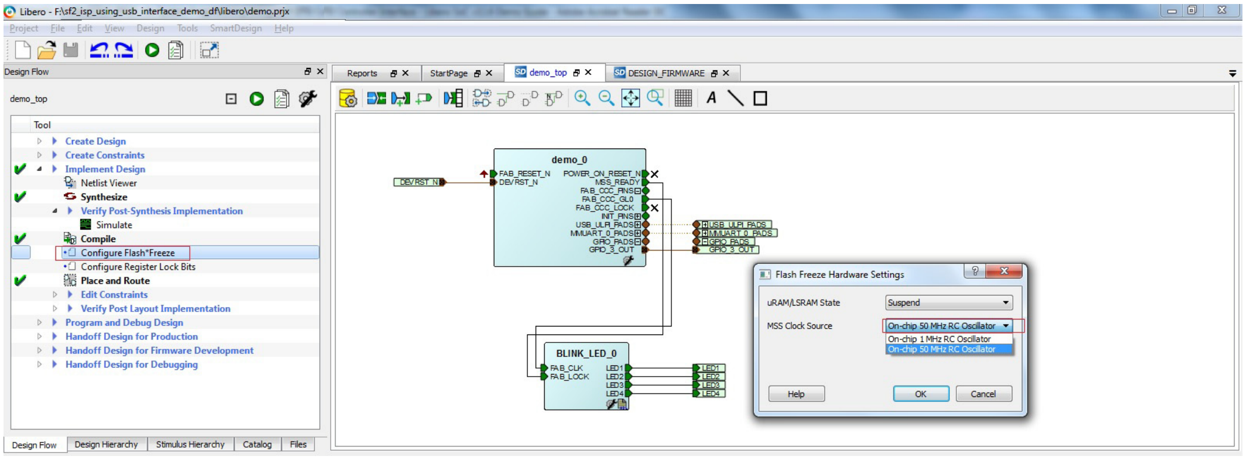
Ready

Fam: SmartFusion2 | Die: M2S150T | Pkg: 1152 FC

7.2 Standby Clock Source Configuration

The standby clock source for the MSS in F*F mode is configured to **On-chip 50 MHz RC Oscillator** using the **Flash*Freeze Hardware Settings** dialog box in the Libero SoC software, as shown in the following figure. A higher MSS clock frequency is required in F*F mode to meet the MMUART baud rate requirements.

Figure 22 • **Flash*Freeze Hardware Settings Dialog Box**



Libero - F:\sf2_isp_using_usb_interface_demo_df\libero\demo.prjx

Flash Freeze Hardware Settings

uRAM/LSRAM State: Suspend

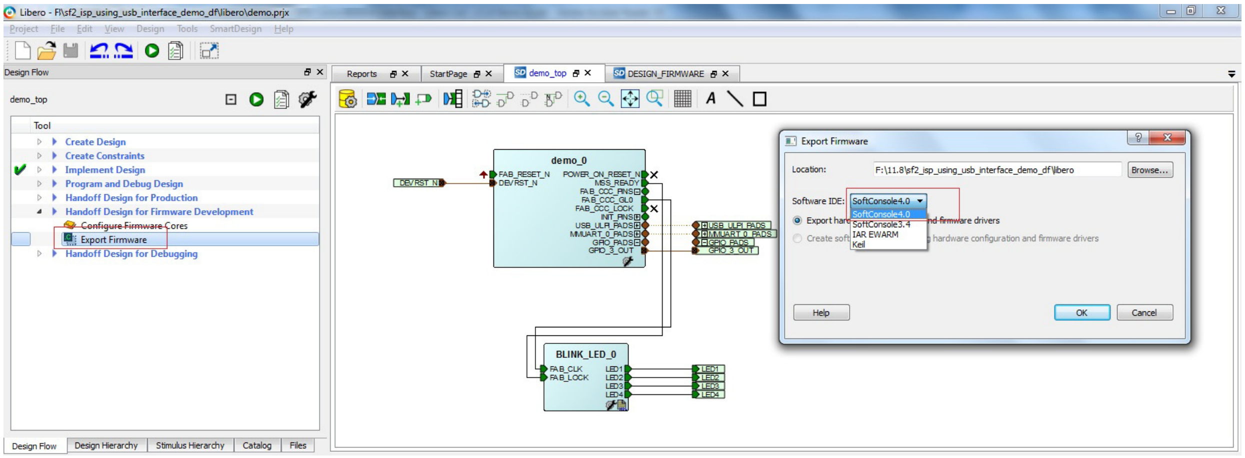
MSS Clock Source: On-chip 50 MHz RC Oscillator

On-chip 1 MHz RC Oscillator
On-chip 50 MHz RC Oscillator

7.3 SoftConsole Project Generation

The firmware can be generated by checking the Create Project and selecting a Software IDE option in Libero project, as shown in the following figure.

Figure 23 • **Export Firmware Options**



Libero - F:\sf2_isp_using_usb_interface_demo_df\libero\demo.prjx

Export Firmware

Location: F:\11.8\sf2_isp_using_usb_interface_demo_df\libero

Software IDE: SoftConsole4.0

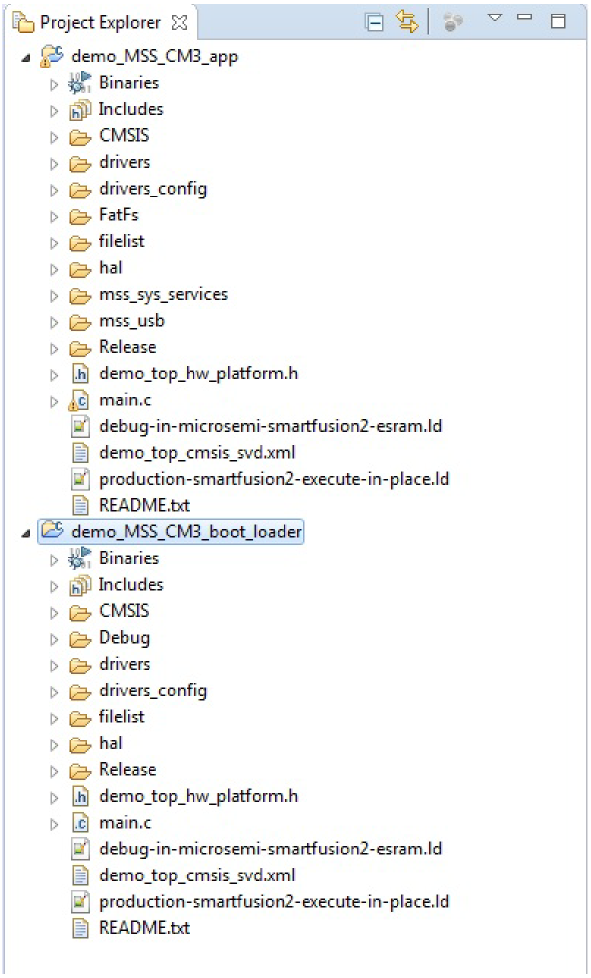
SoftConsole4.0
SoftConsole3.4
IAR EWARM
Keil

● Export hard... nd firmware drivers

○ Create soft... g hardware configuration and firmware drivers

On successful firmware generation, the firmware and SoftConsole folders are generated at *<download_folder>\sf2_isp_using_usb_interface_demo_df\libero* as specified in Location field of **Export Firmware** dialog box, as shown in Figure 23, page 27.

Figure 24 • **SoftConsole Project Workspace**



Project Explorer

- demo_MSS_CM3_app
  - Binaries
  - Includes
  - CMSIS
  - drivers
  - drivers_config
  - FatFs
  - filelist
  - hal
  - mss_sys_services
  - mss_usb
  - Release
  - demo_top_hw_platform.h
  - main.c
  - debug-in-microsemi-smartfusion2-esram.ld
  - demo_top_cmsis_svd.xml
  - production-smartfusion2-execute-in-place.ld
  - README.txt
- demo_MSS_CM3_boot_loader
  - Binaries
  - Includes
  - CMSIS
  - Debug
  - drivers
  - drivers_config
  - filelist
  - hal
  - Release
  - demo_top_hw_platform.h
  - main.c
  - debug-in-microsemi-smartfusion2-esram.ld
  - demo_top_cmsis_svd.xml
  - production-smartfusion2-execute-in-place.ld
  - README.txt

The SoftConsole workspace consists three projects.

- demo_MSS_CM3_app
  This project displays the available programming files from USB mass storage device. This project receives the bitstream from the host PC through UART interface and invokes the system controller programming services.
- demo_MSS_CM3_boot_loader
  This project implements the remapping of the eSRAM to Cortex-M3 processor code space after copying the ISP code to eSARM from eNVM.
- demo_MSS_CM3_hw_platform
  This project contains all the firmware and hardware abstraction layers that correspond to the hardware design. This project is configured as a library and is referenced by `demo_MSS_CM3_app` and `demo_MSS_CM3_boot_loader` application projects. The contents of this folder get over-written every time the firmware is exported, as shown in Figure 23, page 27.

8 Appendix: Implementing Workaround to Access Fabric LSRAM after IAP or ISP Program Operation

The LSRAM write and read accesses are denied after implementing IAP or ISP program operation. The workaround for this problem is to apply System Reset after IAP or ISP program operation. This workaround can be implemented by one of the following ways.

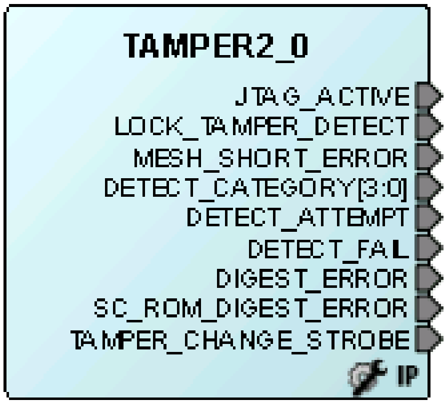
- Using SmartDesign
- Importing the `.cxf` file

8.1 Using SmartDesign

The following steps describe how to apply System Reset:

1. Choose **File** > **New** > **SmartDesign**.
2. Enter **Name** as **Dev_Restart_after_ISP_blk** in the **Create New SmartDesign** window.
3. Navigate to **Libero Catalog** to open **Tamper Macro**.
   a. Drag-and-drop the **Tamper Macro** available in **Libero Catalog** to the **Dev_Restart_after_ISP_blk SmartDesign** canvas, as shown in the following figure.

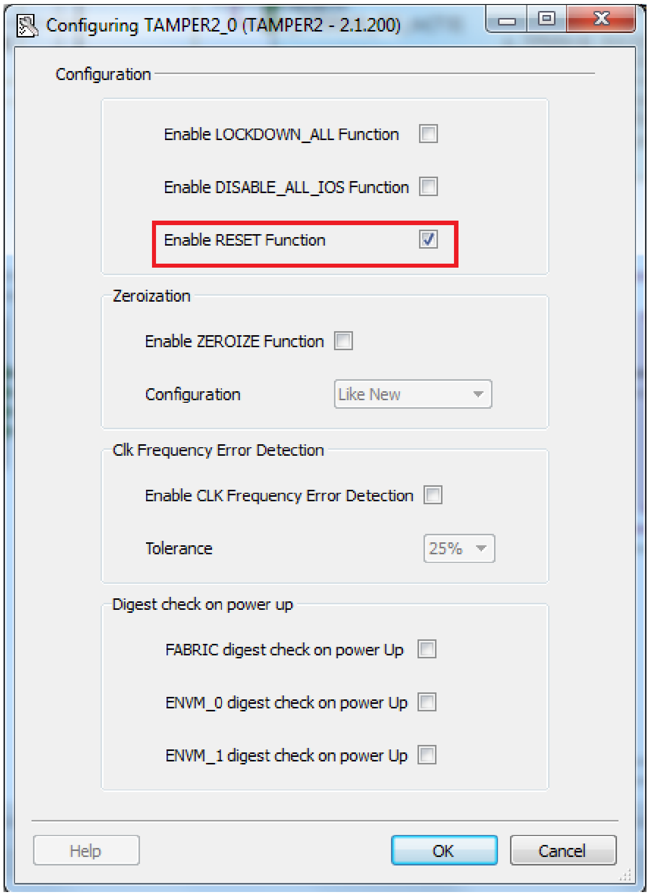
Figure 25 • **Tamper Macro**



   b. Select the **Enable RESET Function** check box in the **Configuring Tamper 2_0** window.

   c. Click **OK**. The **System Reset** is enabled.

Figure 26 • **Tamper Macro Configuration Window**



Configuring TAMPER2_0 (TAMPER2 - 2.1.200)

Configuration

Enable LOCKDOWN_ALL Function □

Enable DISABLE_ALL_IOS Function □

Enable RESET Function ☑

Zeroization

Enable ZEROIZE Function □

Configuration: Like New

Clk Frequency Error Detection

Enable CLK Frequency Error Detection □

Tolerance: 25%

Digest check on power up

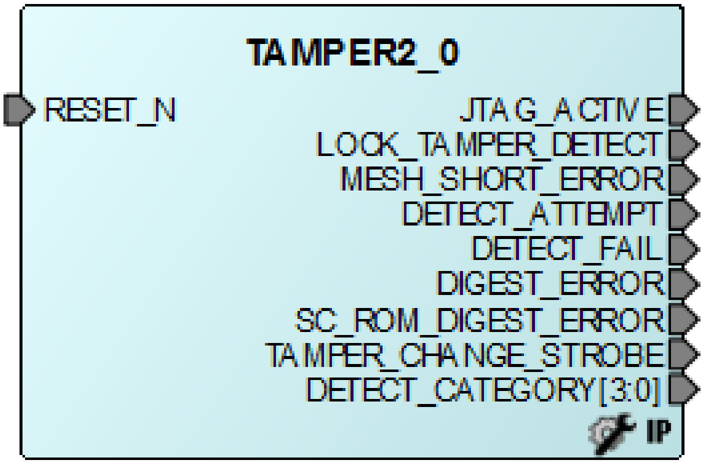
FABRIC digest check on power Up □

ENVM_0 digest check on power Up □

ENVM_1 digest check on power Up □

The following figure shows the TAMPER2_0 macro after configuration.

Figure 27 • **Tamper Macro**

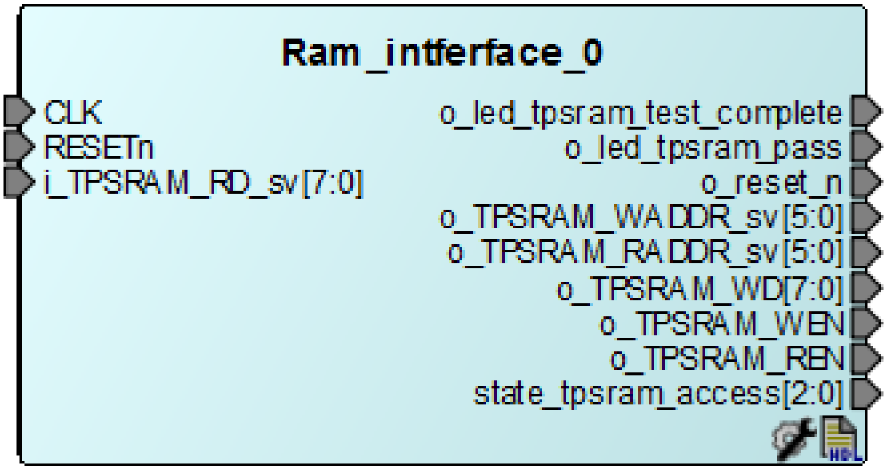


4. Instantiate the **FSM Module** provided in the design files. This FSM Logic performs 3 consecutive address writes to the Two-Port Large SRAM with the known data pattern and then reads back data from those 3 consecutive address locations to compare. If the read back data pattern does NOT match with the written data pattern, then the FSM asserts the RESET_N input to Tamper Macro, which in turn causes a System Reset. If the read back data pattern matches with the written data pattern, then the FSM does not do anything. Follow the steps to add the FSM logic to the PCIe IAP design,
   a. Choose **File** > **Import** > **HDL Source Files**.

b. Browse to the following **Ram_interface.v file** location in the design files folder. *<download_folder>\sf2_isp_using_usb_interface_demo_df\Sourcefiles*

c. Click the **Dev_Restart_after_ISP_blk** tab and drag-and-drop the **Ram_interface** component from the **Design Hierarchy** to the **Dev_Restart_after_ISP_blk SmartDesign** canvas. Figure 28, page 31 shows the **Ram_interface** component.

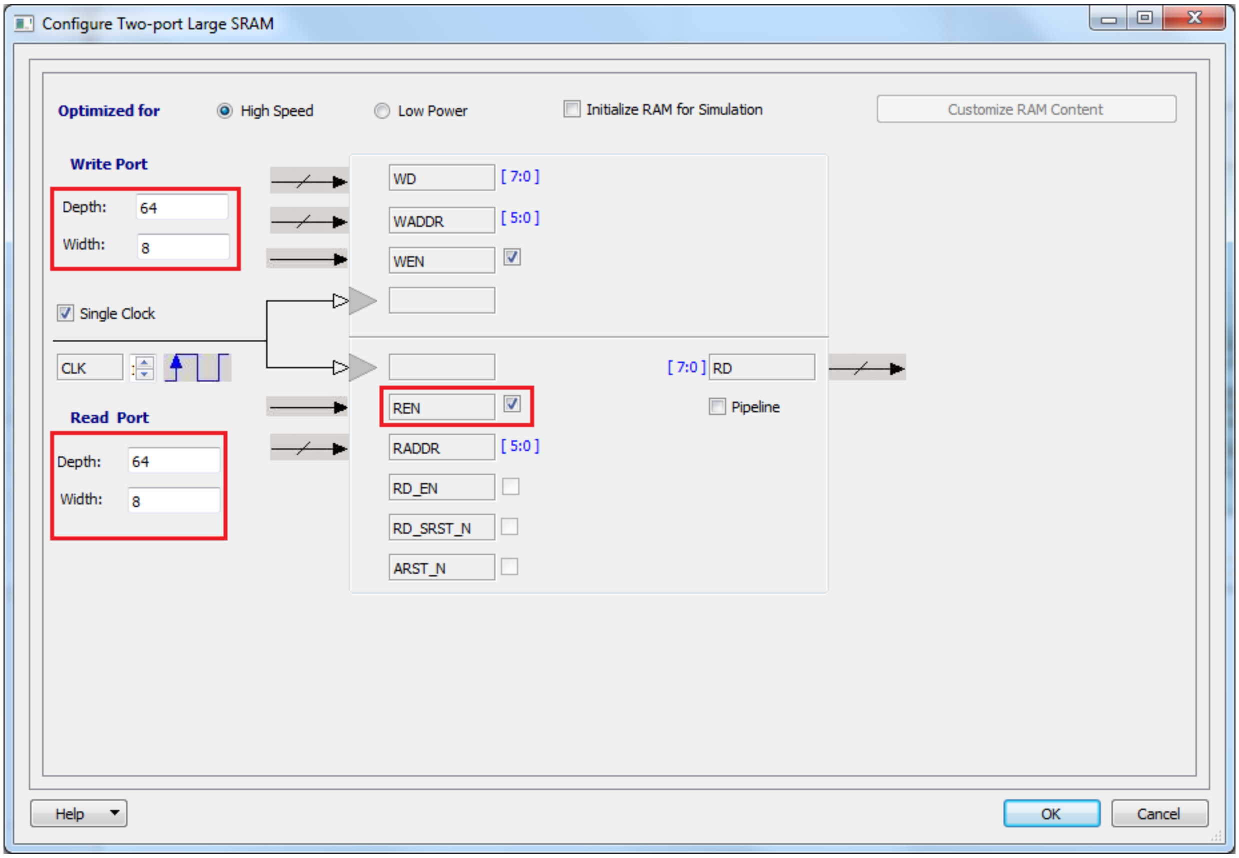
Figure 28 • **Ram_interafce FSM Component**



After completing the IAP programming, the System Controller asserts POWER_ON_RESET_n to FPGA fabric. This triggers the RESETn signal and initiates the state machine in the FSM module.

5. Drag-and-drop the **Two-Port Large SRAM (TPSRAM)** available in the **Libero Catalog** to the **Dev_Restart_after_ISP_blk SmartDesign** canvas. Configure the **TPSRAM** with the following settings:
   - Write Port
     - Depth: 64
     - Width: 8
   - Read Port
     - Depth: 64
     - Width: 8
   - Select **Check REN** check box

Figure 29 • **Two-Port SRAM Configurator Window**



Configure Two-port Large SRAM

Optimized for ● High Speed ○ Low Power □ Initialize RAM for Simulation

Customize RAM Content

Write Port

Depth: 64

Width: 8

☑ Single Clock

CLK

Read Port

Depth: 64

Width: 8

WD [7:0]

WADDR [5:0]

WEN ☑

REN ☑

RADDR [5:0]

RD_EN □

RD_SRST_N □

ARST_N □

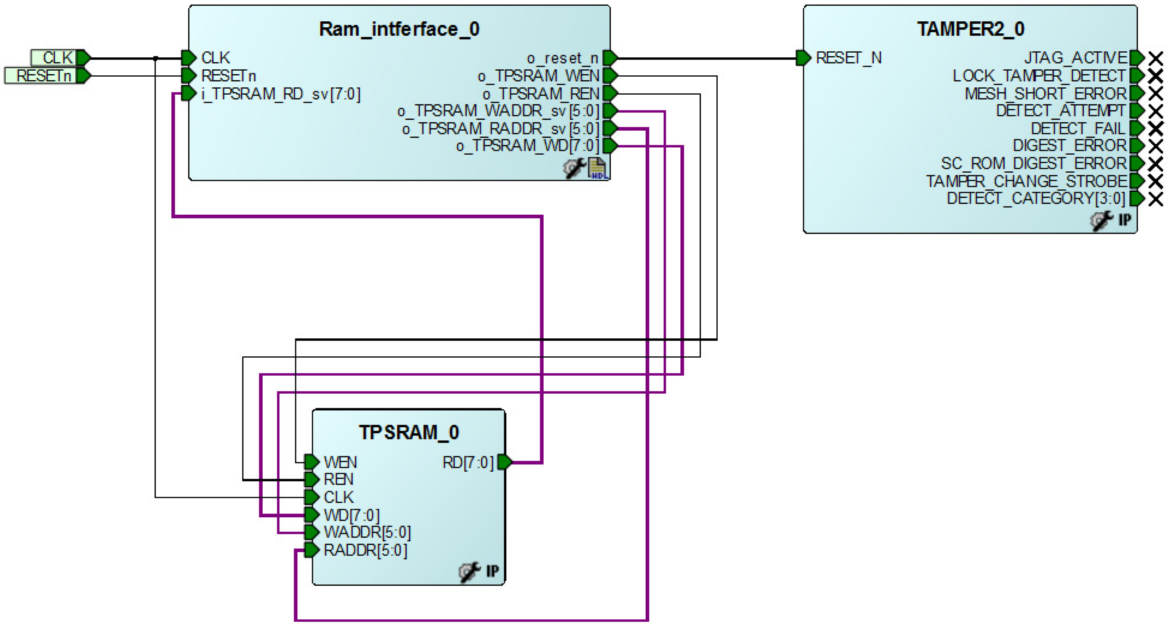
[7:0] RD

□ Pipeline

Help OK Cancel

6. Make the connections for **Tamper Macro**, **FSM**, and **TPSRAM**, as shown in Figure 30, page 32.

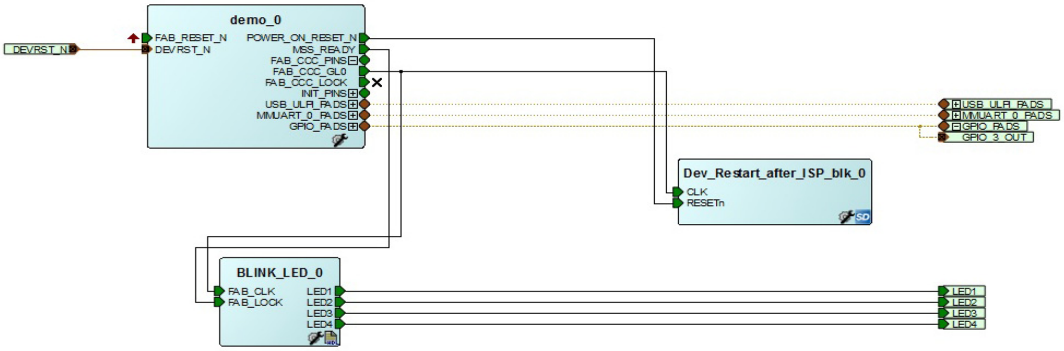
Figure 30 • **Dev_Restart_after_ISP_blk SmartDesign**



7. Click the **demo_top** tab and drag-and-drop the **Dev_Restart_after_ISP_blk** component from the **Design Hierarchy** to the **demo_top SmartDesign** canvas.

8. Make the connection as shown in the following figure and generate **demo_top SmartDesign**. This completes the implementation of the workaround.

Figure 31 • **demo_top SmartDesign**



Note: This workaround is applicable for v11.5 software release or later, and must be implemented in the Libero design, which is used to generate the `.spi` programming file. Older versions of Libero might prune Tamper Macro during Synthesis. To avoid pruning, one of the recommended options is to promote the DETECT_ATTEMPT signal of Tamper Macro to the top-level.

8.2 Importing the .cxf File

Import the `.cxf` file for SmartDesign Dev_Restart_after_ISP_blk. This `.cxf` file is provided with the design files and it has all the component instantiations and connections mentioned in Using SmartDesign, page 29 from step 1 to 6.

The following steps describe how to import `.cxf` file to the Libero project:

1. Choose **File** > **Import** > **Others**.
2. Browse to the following **Dev_Restart_after_ISP_blk.cxf** file location in the design files folder. *<download_folder>\sf2_isp_using_usb_interface_demo_df\sample_programming_files\LSRAM_Workaround\component\work\Dev_Restart_after_ISP_blk*
3. Browse to the following ***Ram_interface.v** file* location in the design files folder. *<download_folder>\sf2_isp_using_usb_interface_demo_df\Sourcefiles*
4. Repeat **Step 7** and **Step 8** to instantiate **Dev_Restart_after_ISP_blk** in **demo_top SmartDesign**.