AC400
Application Note
SmartFusion2 SoC FPGA Flash*Freeze Entry and Exit -
Libero SoC v11.7 SP3

Microsemi

Power Matters.™

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 7.0

The following is a summary of the changes in revision 7.0 of this document.

- The Libero SoC and FlashPro versions were updated in the Design Requirements, page 2.
- The design files and the document was updated for Libero SoC v11.7 SP3.
- Added a new section, Prerequisites, page 2.
- The block diagram was updated to include the FLASH_FREEZE macro.
- The significance of CoreResetP IP core v8.0.103 was elaborated. For more information, see Hardware Implementation, page 9.

1.2 Revision 6.0

In revision 6.0 of this document, updated the document for Libero SoC v11.7 software release (SAR 75559).

1.3 Revision 5.0

In revision 5.0 of this document, updated the document for Libero SoC v11.6 software release (SAR 68370).

1.4 Revision 4.0

In revision 4.0 of this document, updated the document for Libero SoC v11.5 software release (SAR 62938).

1.5 Revision 3.0

In revision 3.0 of this document, updated the document for Libero SoC v11.4 software release and targeted the SmartFusion2 Evaluation Board (SAR 59063).

1.6 Revision 2.0

In revision 2.0 of this document, updated the document for Libero SoC v 11.2 software release (SAR 53247).

1.7 Revision 1.0

Revision 1.0 was the first publication of this document.

.

2 SmartFusion2 SoC FPGA Flash*Freeze Entry and Exit

SmartFusion®2 System-on-Chip (SoC) field programmable gate array devices provide an ultra-low static power solution through Flash*Freeze technology. Flash*Freeze mode entry retains all the static random-access memory (SRAM) and registers information. Flash*Freeze mode exit achieves rapid recovery to active mode.

This application note specifies how to enter and exit Flash*Freeze mode on the SmartFusion2 Security Evaluation Board using the ".stp" programming file. The SRAM content retention capability during Flash*Freeze is also shown in this application note.

The ".stp" file is present at the following location of the design files folder:

`m2s_ac400_flashfreeze_liberov11p7sp3_df\Programming_File`

For more information on the Flash*Freeze entry and exit implementation, Flash*Freeze Libero design project, and all the necessary blocks and IP cores instantiated in Libero SoC, see Design Details, page 6.

2.1 Design Requirements

Table 1 • **Design Requirements**

Hardware Requirements	Description
SmartFusion2 Security Evaluation Kit	M2S090TS-EVAL-KIT
Host PC	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.7 SP3
FlashPro Programming Software	v11.7 SP3
SoftConsole	v4.0
Host PC Drivers	USB to UART drivers
Serial Console	Any serial console like PuTTY, HyperTerminal, and TeraTerm.

Note: You can use any serial console, we have used PuTTY.

2.2 Prerequisites

Before you start:

1. Download and extract the design files from the following link:
   http://soc.microsemi.com/download/rsc/?f=m2s_ac400_flashfreeze_liberov11p7sp3_df

   The design file consists of Libero SoC Verilog project, SoftConsole software project, CPZ file of CoreResetP v8.0.103, and programming files (*.stp) for SmartFusion2 Security Evaluation Kit board. Refer to the `Readme.txt` file included in the design file for the directory structure and description.
2. Connect the power supply cable to the J6 connector on the board.
3. Connect the FlashPro4 programmer to the PROG HEADER J5 connector on the board.

4. Connect the jumpers to the SmartFusion2 Security Evaluation Kit board as shown in the following table.

Table 2 • **SmartFusion2 Security Evaluation Kit Jumper Settings**

Jumper	Pin (from)	Pin (to)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

5. Download any free serial console program like PuTTY, TeraTerm, or HyperTerminal.

2.3 Enter and Exit Flash*Freeze

All the necessary blocks of the device are programmed using the ".stp" file. Then, PuTTY is used as an interface between the Host PC and the Flash*Freeze system services running on the device. You initiate Flash*Freeze entry and exit service requests from the Host PC, these service requests are executed by the Flash*Freeze system services.

Follow these steps to program the device:

1. Turn on the board using the SW7 slide switch.
2. Start FlashPro, and click **New Project** to create a new FlashPro project.
3. Create a new project folder and select the **Single device** option in the **New Project** Dialog box.
4. Click the **Configure Device** and browse the existing ".stp" programming file to load it.
5. Click the **Program** button to program the device.
   The **Programmer List Window** in the FlashPro, shows the Programmer Name, Programmer Type, Port, Programmer Status, and the Programmer Enabled information.

   When the device is programmed, the Programmer Status column displays the "RUN PASSED" status. And, the H5, H6, J6, and H7 LEDs start blinking.

Follow these steps to enter and exit Flash*Freeze:

1. Connect the USB cable from Host PC to FTDI port on the board, and start PuTTY.
2. Create and load a session with the following properties:
   Baud Rate = 57600, 8 data bits, 1 stop bit, no parity, and no flow control. For more information on configuring the serial console, see the *Configuring Serial Terminal Emulation Programs*.
3. When the PuTTY opens, press the Device Reset (SW6) push button.
4. On your serial console, you see the Flash*Freeze system services prompt you to enter 1 (Flash*Freeze), 2 (Write to SRAM), 3 (Read from SRAM), or to enter 4 (RTC Wake-up).
5. Press "1" to first enter Flash*Freeze.
   The H5, H6, J6, and H7 LEDs stop blinking, and the serial console displays the "**Flash*Freeze system service request success**" message as shown in the following figure.

Figure 1 • **Flash*Freeze Shutdown**


```
1. Flash*Freeze
2. Write to SRAM
3. Read from SRAM
4. RTC Wake-Up
1

Requesting Flash*Freeze shutdown.


-----------------------------------------------------------------------
!!!!!!!!!!!!!!!!!!!!!!!!! Flash*Freeze shutdown. !!!!!!!!!!!!!!!!!!!!!!!!!
-----------------------------------------------------------------------

Flash*Freeze system service request success.
--------------------
```
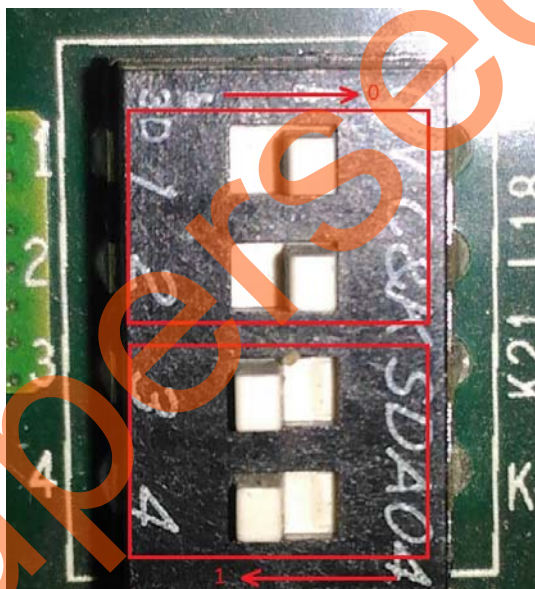
      6.   Press "4" to exit Flash*Freeze.

Note: You can also use the dip slide switches, or press the SW1 push button on the board to exit Flash*Freeze. Set the dip slide switches as shown in the following figure to exit Flash*Freeze. For more information, see Figure 7, page 9.

Figure 2 • **Dip Slide Switches Setting for FlashFreeze Exit**

The device exits Flash*Freeze and the H5, H6, J6, and H7 LEDs start blinking. The serial console shows the "**Flash*Freeze Exited**" message as shown in Figure 3, page 5.

Figure 3 • **Flash*Freeze Exited**

7. Press "2" (Write to SRAM).
    The content stored in eNVM is written to SRAM as shown in the following figure.

Figure 4 • **Write to SRAM**

8. Press "3" to read from SRAM to verify that the SRAM content is written successfully.

Figure 5 • **Read Data from SRAM**

9. Press "1" to enter Flash*Freeze, and press "4" to exit Flash*Freeze.
10. Press "3" to read from SRAM.
    You see the contents shown in the previous figure, which shows that SRAM content was retained during Flash*Freeze.

2.4 Design Details

One of the functions of the System Controller in the SmartFusion2 device is to handle the System Services requests through the communication block (COMM_BLK). Flash*Freeze service is one the system service provided by the System Controller. The SmartFusion2 device enters Flash*Freeze by using the Flash*Freeze services request that the System Controller provides.

Exit from Flash*Freeze can be initiated by internal timed events, such as a real-time counter (RTC) event or external I/O events (either transitions or pattern matching on I/Os).

2.4.1 Design Description

The design consists of the MSS, a counter, SRAM wrapper logic, IP cores (CoreAHBLite, CoreAHBToAPB3, CoreResetP, and CoreAPB3), FLASH_FREEZE macro, and fabric CCC (FCCC). The IP cores along with the SRAM wrapper are used to initialize the fabric SRAM by moving data from the embedded nonvolatile memory (eNVM) to the fabric SRAM through FIC_0 AHB master interface. A Data Storage client is defined in the eNVM with the data to be written to the SRAM. This is Data Storage client demonstrate the state of the fabric SRAM content after exiting from Flash*Freeze mode. The CoreResetP handles the sequencing of reset signals in the device. For more information on this core, see the CoreResetP Handbook.

Using the System Builder, the MSS is configured to use one UART interface (MMUART_1), MSS clock condition circuit (MSS_CCC), the RTC to generate the RTC interrupt event to wake up the device, and one instance of the fabric interface (FIC_0). The FIC_0 interface is configured to use the master interface with AHB-Lite (AHBL) interface type. The MMUART_1 is used as an interface for reading and writing to the HyperTerminal and is clocked by PCLK1 on the APB bus1 (APB_1). PCLK1 is derived from the Cortex-M3 processor and MSS main clock (M3_CLK). The M3_CLK, FIC_0_CLK, and APB_1_CLK are configured as 100 MHz clocks generated from the MSS_CCC. The top-level block diagram of the design is shown in the following figure.

Figure 6 • **Top-Level Block Diagram of the Design**
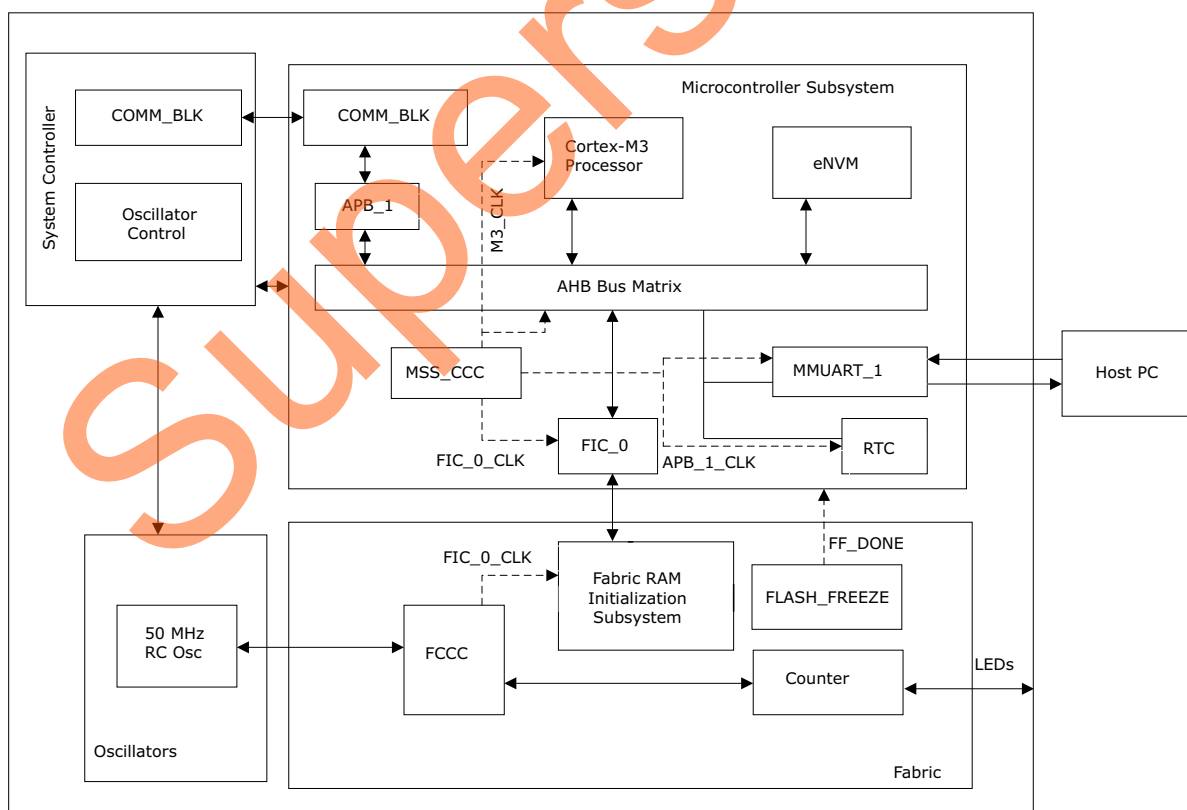
In Active mode (non Flash*Freeze), the MSS_CCC is configured to be sourced from the FPGA fabric through the CLK_BASE port. The FCCC is configured to provide the 100 MHz CLK_BASE reference. The on-chip 50 MHz oscillator is the reference clock source for the FCCC. The output of a counter is connected to a set of light-emitting diodes (LEDs) to monitor the state of the fabric while entering and exiting Flash*Freeze mode. The LEDs ports assignments are shown in the following table.

Table 3 • **LED to Pins Assignments (SmartFusion2 Security Evaluation Kit Board)**

Counter Output	Package Pin
LED_1	H5
LED_2	H6
LED_3	J6
LED_4	H7

2.4.2 Entering Flash*Freeze Mode

Entering into Flash*Freeze mode is done through the System Services using software drivers. System Services are requested, through firmware drivers, by sending a command byte describing the function to be performed followed by command specific sub-commands and/or data. The Flash*Freeze service requests the System Controller to execute the Flash*Freeze entry sequence. When the Flash*Freeze service begins execution, the System Controller informs the MSS by sending a command byte E0H that Flash*Freeze shutdown is imminent. The service is stalled until this command byte can be accepted by the COMM_BLK FIFO. If a new service request is received while servicing another request, the new service request is immediately aborted.

As the Flash*Freeze system service command is initiated, the System Controller disables the fabric, each eNVM block, or the MSS PLL circuit. All these options are available as part of the firmware System Services driver function MSS_SYS_flash_freeze(), which is part of the mss_sys_services driver. For more information, see Software Implementation, page 14.

2.4.3 Exiting Flash*Freeze Mode

Exiting from Flash*Freeze mode can be initiated by external I/Os events or by an RTC event. User I/Os (MSIO, MSIOD, or DDRIO) that are single-ended inputs can participate in the Flash*Freeze exit in the following ways.

- I/O Activity: Force Flash*Freeze exit up on an activity (Wake_On_Change)
- I/O Signature: Force Flash*Freeze exit up on a signature (Wake_On_1/Wake_On_0) match in which the I/O participates with other I/Os to trigger Flash*Freeze exit. This is a logical AND behavior where all I/Os must meet the Low Power Exit settings.

The external I/O events are specified during the design time using the I/O Editor in the Libero SoC software. Only input I/Os participate in the Flash*Freeze exit event.

Note: The Wake_On_Change is a logical **OR** behavior with I/Os that are set as Wake_ON_1/Wake_ON_0. This means that to wake from Flash*Freeze, it must be {(All Wake-on-0 **ANDed**) **ANDed** with (All Wake-on-1 **ANDed**)} **ORed** with (All Wake-on-Change ORed).

2.4.3.1 I/O Activity

In I/O Activity mode, an input I/O can be selected to be part of a transition. The value at the pin of the activity I/O is latched before going to Low Power mode. When a change happens on the configured I/O, the device wakes up from Flash*Freeze mode. The change can either be 1-to-0 or 0-to-1. This option is equivalent to the "Wake_On_Change" option in the I/O Editor. This can be set on more than one I/O. The Wake_On_Change is a logical OR behavior with other I/Os that are set as Wake_On_Change.

2.4.3.2 I/O Signature

Any input I/O can be selected to be a part of a signature match value that is used to wake-up the device from Flash*Freeze mode. All the selected I/Os have to match a static predetermined value at the same time. If the configured signature values match the values at I/Os, then the device exits from Flash*Freeze mode. I/Os can be a mixture of different signature settings. An I/O can be configured to participate in the Flash*Freeze exit upon a 0-to-1 or it can be configured to participate in the Flash*Freeze exit upon a 1-to-0 transition. These options are equivalent to Wake_On_1 (transition from 0-to-1) and Wake_On_0 (transition from 1-to-0) settings in the I/O Editor in the Libero SoC software.

All other I/Os that are not participating in the Flash*Freeze exit mechanism are tristated or held to the previous state (LAST_VALUE) before entering Flash*Freeze mode. The selection is set using **I/O state in Flash*Freeze mode** column options in the I/O Editor using Libero SoC, as shown in Figure 14, page 12.

SW5 (four different dual in-line package (DIP) switches) on the Evaluation Kit board is used to demonstrate the pattern matching wake-up mechanism. Four different inputs are created in the top-level design where each input is assigned to a DIP switch, as shown in Figure 7, page 9. SW1 on the Evaluation Kit board is used to demonstrate the transition (Wake_On_Change) wake-up event mechanism, as shown in Figure 7, page 9.

*Figure 7 •* **DIP Switches and the SW1 Connectivity in SmartDesign**
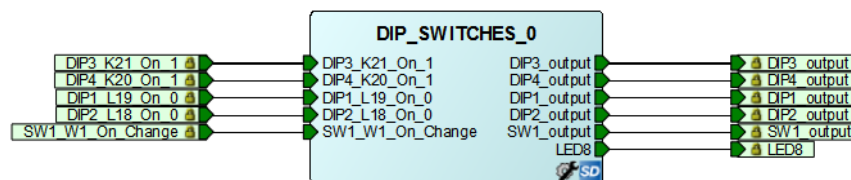
To demonstrate the RTC wake-up event mechanism, the RTC is configured in Binary mode. For more information, see Software Implementation, page 14. The timeout value should be set per the application needs and should also ensure that one of the on-chip clock resources is driving the RTC. Exit from Flash*Freeze mode can also be achieved by the Cortex-M3 processor by setting the "Wakeup_set" bit in the RTC control register that results in assertion of the RTC wakeup interrupt. The RTC wakeup interrupt is routed to the System Controller, fabric, and Cortex-M3 processor nested vectored interrupt controller (NVIC). For more information, see Hardware Implementation, page 9.

2.4.4 Hardware Implementation

The hardware implementation involves configuring the MSS and the necessary Flash*Freeze settings. The FIC_0, MMUART_1, and RTC are enabled using the MSS configurator. The design example consists of MSS, a counter, SRAM wrapper logic, IP cores (CoreAHBLite, CoreAHBToAPB3, and CoreAPB3), FLASH_FREEZE_0 macro, and FCCC, as shown in the following figure. The IP cores along with the SRAM wrapper are used to initialize the fabric SRAM by moving data from the eNVM to the fabric SRAM through FIC_0 AHB master interface. The following two Data storage clients are defined in the eNVM:

- A ".mem" file is defined in the eNVM with the data to be written to the SRAM. This is used to demonstrate the state of the fabric SRAM content after exiting from Flash*Freeze.
- A ".hex" file is defined in the eNVM. This is the Flash*Freeze firmware executable used to enter and exit Flash*Freeze.

The following figure shows how these two Data Storage clients are defined in the eNVM.

*Figure 8 •* **Defining Data Storage Clients in the eNVM**
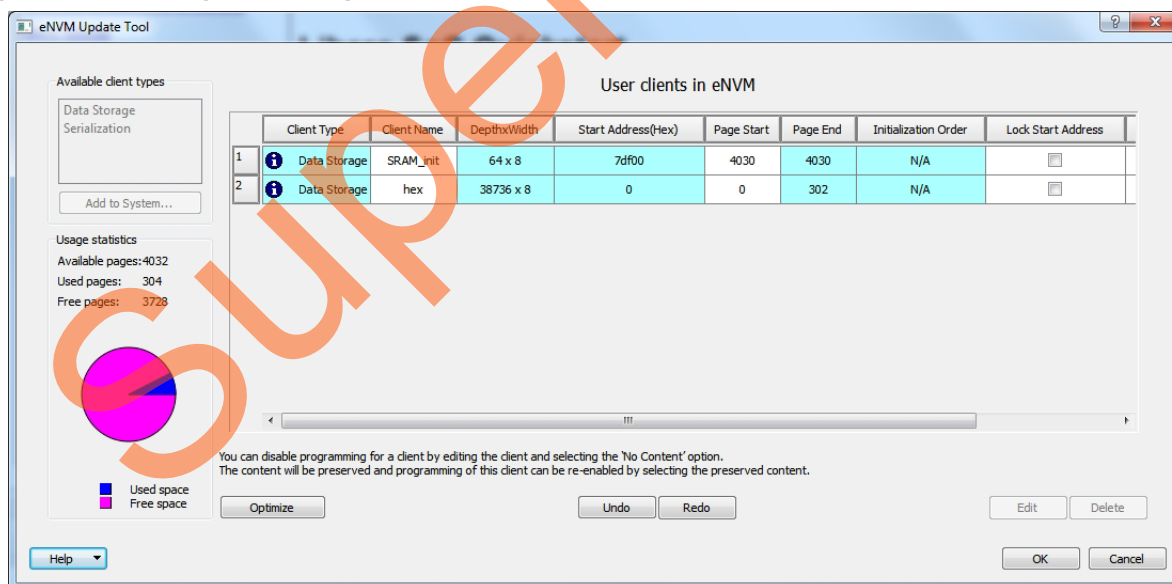
The instantiated FLASH_FREEZE_0 macro is used as an interface between the FPGA fabric and the system controller. The FLASH_FREEZE_0 macro provides two active output signals: FF_TO START and FF_DONE to the FPGA fabric. For more information on FLASH_FREEZE_0 macro and its signal details, see the *UG0444: SmartFusion2 and IGLOO2 Low Power Design User Guide*.

Figure 9 • **Top-Level Hardware Design**
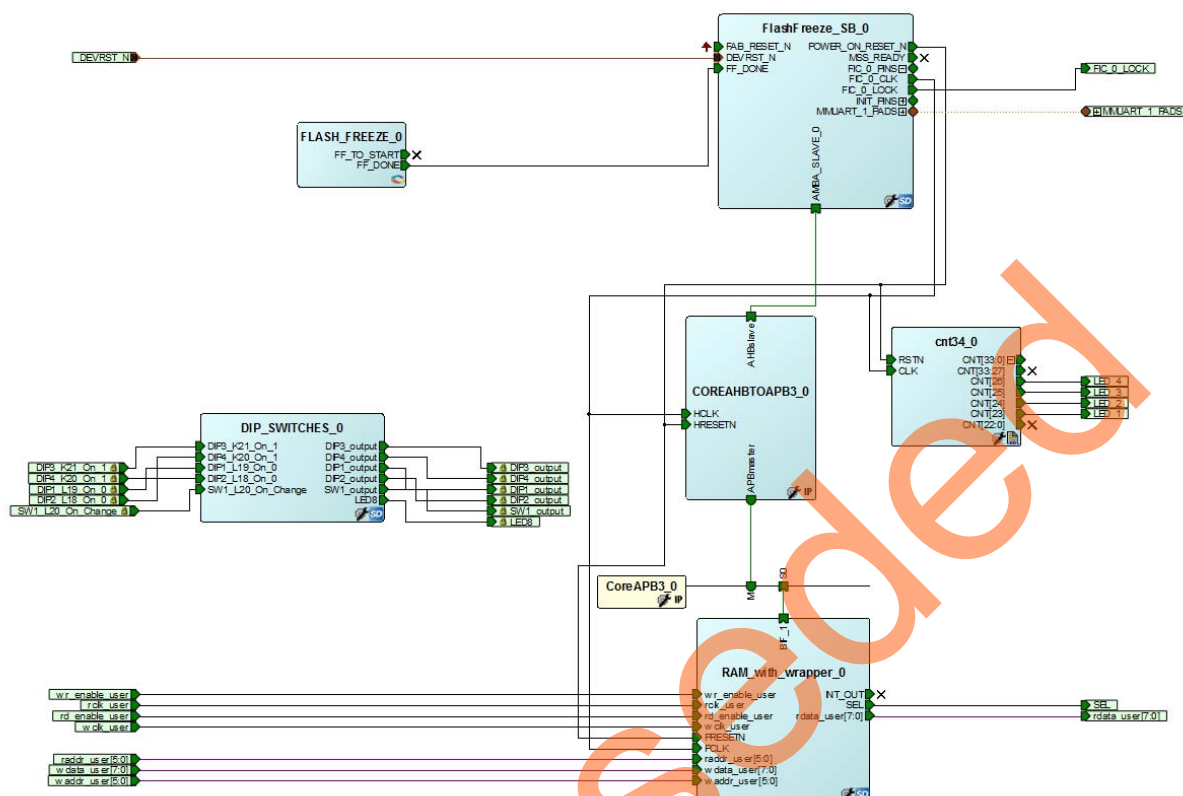
The FIC_0 interface is configured part of the System Builder as AHBL master interface, as shown in the following figure.

Figure 10 • **FIC_0 AHBL Master Interface Configuration**
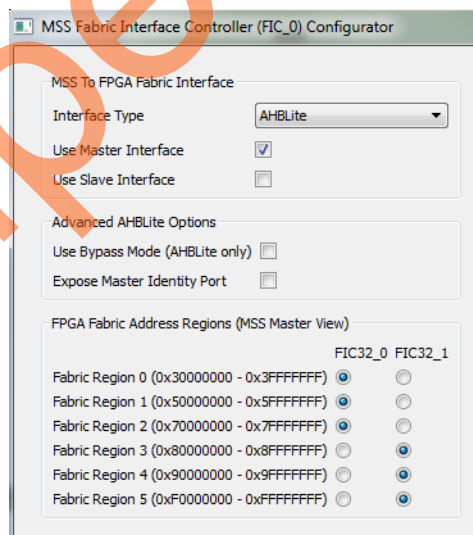
The RTC block is enabled and is clocked from the internal 1 MHz RC oscillator. This option is selected in the Libero SoC during the hardware design flow. **Enable WakeUp interrupt to Cortex-M3** is selected, as shown in Figure 11, page 11.

Figure 11 • **RTC Configuration**
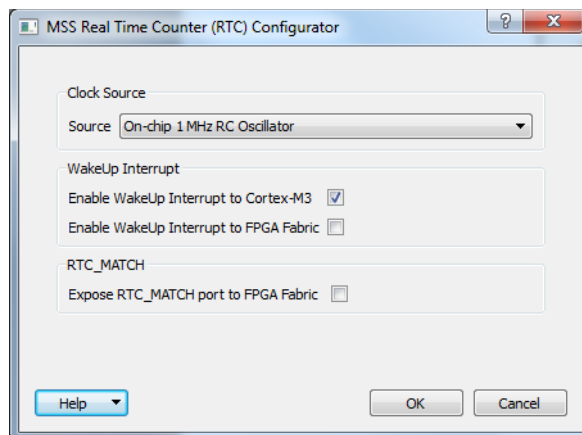
The MSS_CCC clock source is sourced from the FCCC through the CLK_BASE port. The FCCC is configured to provide the 100 MHz clock using GL0. The FCCC reference clock is sourced from the On-chip 25/50 MHz RC Oscillator. The following figure shows the system clocks configurations for the M3_CLK, APB_1_CLK, and FIC_0_CLK clock settings.

Figure 12 • **MSS CCC System Builder System Clocks Configurations**
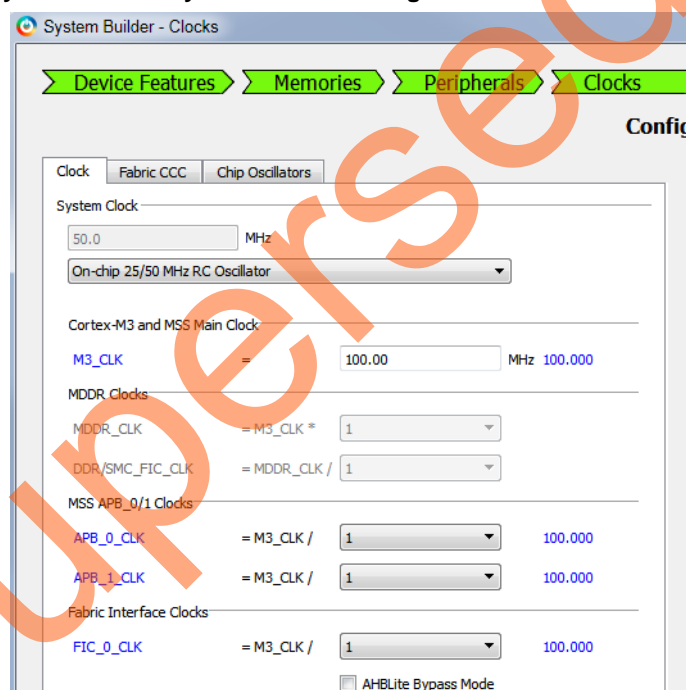

Note: Connect the inverted FF_Done signal to all fabric CCC reset inputs (PLL_ARST_N) for resetting CCC during flash freeze exit.

The standby clock source for the MSS in Flash*Freeze mode and the state of the SRAMs (uSRAM and LSRAM) during Flash*Freeze mode are configured using the Flash*Freeze Hardware Settings dialog in the Libero SoC software, as shown in the following figure. For some peripherals that can remain active (such as SPI or MMUART), a higher MSS clock frequency (for example, MMUART to meet the baud rate) might be required. Following are the MSS clock source options that are available to be used during Flash*Freeze mode:

- On-chip 1 MHz RC oscillator
- On-chip 50 MHz RC oscillator

The fabric SRAM state during F*F can either be "Sleep" or "Suspend". In the "Suspend" mode, the large SRAM (LSRAM) and micro SRAM (uSRAM) contents are retained. It means, when the device exits F*F mode, the content of the SRAM is not lost. In Sleep mode, the LSRAM and µSRAM contents are not retained. In this design, the fabric SRAM state and the standby clock source for the MSS during F*F are configured in the **Flash*Freeze Hardware Settings** dialog box of Libero SoC as shown in the following figure.

Figure 13 • **Flash*Freeze Hardware Settings Dialog Box**
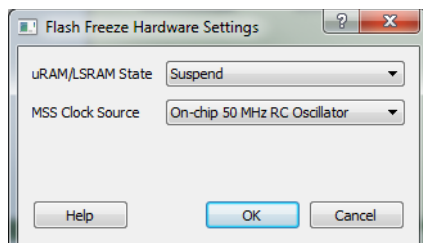
We recommend using CoreResetP IP core v8.0.103 included in the design files to ensure that FF_DONE signal is used to gate any signal that is used as asynchronous resets or presets in fabric and signals that are intended for use as inputs to ASIC blocks on the device (MDDR, FDDR and SERDES). This is to avoid any spurious resets as we exit Flash*Freeze.

You can implement Flash*Freeze in your existing design by importing the CoreResetP IP core. For more information on importing this IP core, see Appendix: Importing IP Core to User Vault, page 17.

The following figure shows how the Flash*Freeze support is enabled using CORERESETP configurator window.
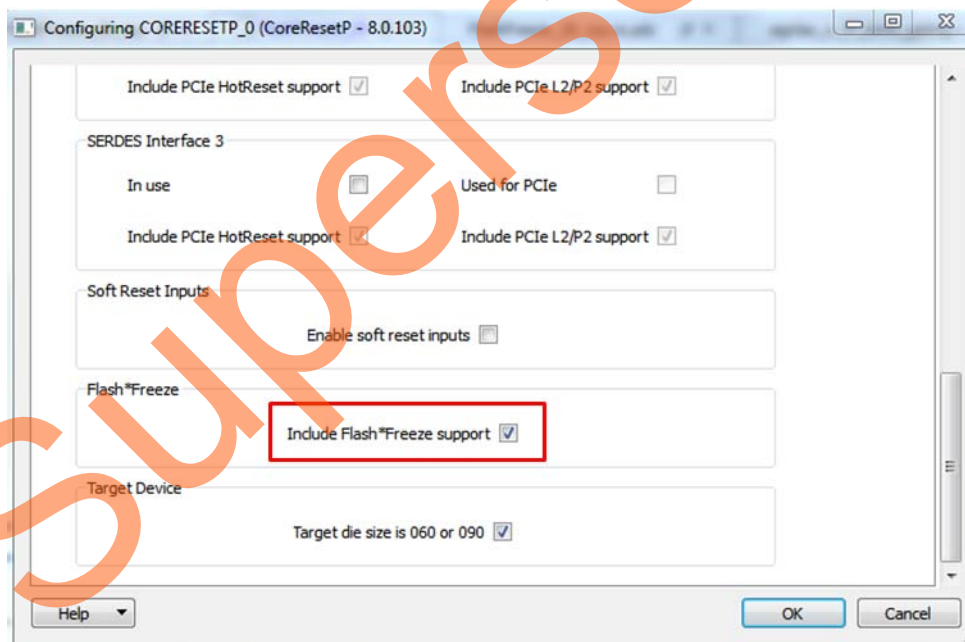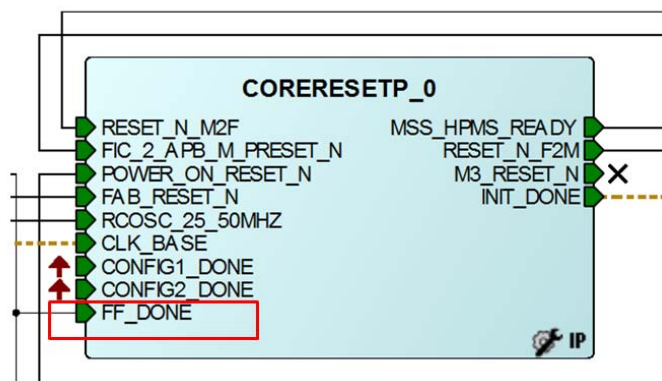
Figure 14 • **Configuring CORERESETP**

Figure 15, page 13 shows the SmartDesign component of CORERESETP with flash freeze support enabled.

Figure 15 • **SmartDesign of CORERESETP**

The I/Os Flash*Freeze exit mechanism is specified using the Low Power Exit setting in the I/O Constraints Editor in the Libero SoC, as shown in the following figure.

Note the following points:

- The I/O available in Flash*Freeze option applies only to I/Os allocated to the MSS peripherals.
- When I/Os are set to be available during Flash*Freeze mode, the I/O state in Flash*Freeze option does not apply.
- Only inputs or bidirectional I/Os participate in signature/activity Flash*Freeze exit. This means that the Low Power Exit options are available to be set on inputs and/or bidirectional I/Os only.

Figure 16 • **Specifying I/O State and Functionality Options Using I/O Editor**

Port Name	Pin Number	Resistor Pull	I/O available in Flash*Freeze mode	Low Power Exit
DIP1_L19_On_0	L19	Up	No	Wake_On_0
DIP2_L18_On_0	L18	Up	No	Wake_On_0
DIP3_K21_On_1	K21	Down	No	Wake_On_1
DIP4_K20_On_1	K20	Down	No	Wake_On_1
SW1_L20_On_Change	L20	None	No	Wake_On_Change
MMUART_1_TXD	H19	None	Yes	--
MMUART_1_RXD	G18	None	Yes	Off

The Flash*Freeze exit behavior of input I/Os (DIP1 - 4) and SW1 are configured using the I/O Editor in the Libero SoC, as shown in the previous figure. The DIP switches to package pin assignments are shown in the following table.

Table 4 • **DIP Switches to Package Pins Assignments**

Input DIP Switch and SW1	Package Pin
DIP1	L19
DIP1	L18
DIP1	K21
DIP1	K20
SW1	L20

The MMUART_1 is used to read and write to the HyperTerminal window and the RXD and TXD ports are configured using the I/O Constraints Editor to be available during Flash*Freeze mode, as shown in the previous figure.

Note: The "I/O available in Flash*Freeze mode" is available only on the I/Os allocated to the MSS peripherals.

Figure 17 • **Configuring MMUART_1 Ports to be Available During Flash*Freeze**

2.4.5 Software Implementation

The SmartFusion2 MSS System Services software driver provides a set of functions to access different System Services that the System Controller performs in conjunction with the communication block (COMM_BLK) that is part of the MSS. One of these services is to request the SmartFusion2 device to enter Flash*Freeze mode. The following figure shows the System Services driver. For more information, right-click the **SmartFusion2_MSS_System_Services_Driver_UG** as shown in the following figure.

Figure 18 • **System Services Firmware Driver**
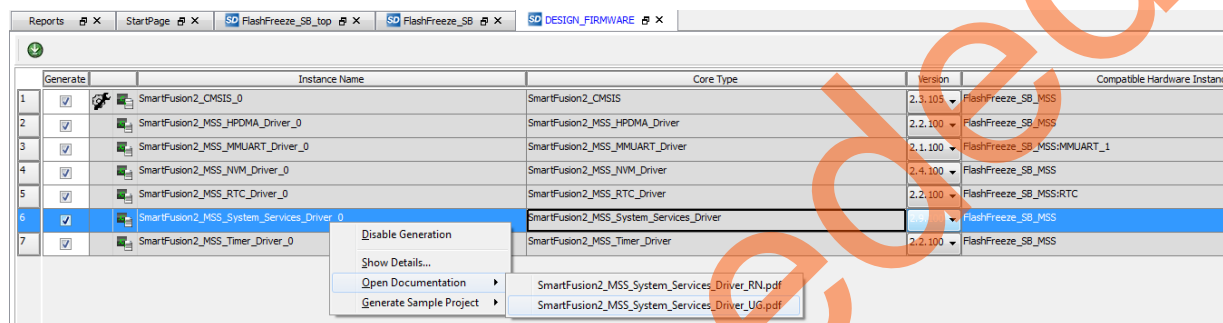
The following drivers and APIs are used in the example design to configure different aspects of the design.

- `MSS_SYS_init(sys_services_event_handler);`
  The System Services driver is initialized through a call to the MSS_SYS_init() function. The MSS_SYS_init() function must be called before any other System Service driver functions are called.

- `MSS_SYS_flash_freeze(options);`
  The function requests the SmartFusion2 device to enter Flash*Freeze mode. The options parameter can be used to power-down different parts of SmartFusion2, as shown in the following table.

Table 5 • **Flash*Freeze Request Function Options Descriptions**

Options	Description
MSS_SYS_FPGA_POWER_DOWN	MSS_SYS_flash_freeze() function should request the FPGA fabric to enter Flash*Freeze mode.
MSS_SYS_ENVM0_POWER_DOWN	MSS_SYS_flash_freeze() function should request eNVM0 to enter Flash*Freeze mode.
MSS_SYS_ENVM1_POWER_DOWN	MSS_SYS_flash_freeze() function should request eNVM1 to enter Flash*Freeze mode.
MSS_SYS_MPLL_POWER_DOWN	MSS_SYS_flash_freeze() function should request the MSS PLL to enter Flash*Freeze mode.

- `MSS_RTC_init(MSS_RTC_BINARY_MODE, RTC_PRESCALER);`
- `MSS_RTC_set_binary_count_alarm(FLASH_FREEZE_TIMEOUT,`
  `MSS_RTC_SINGLE_SHOT_ALARM);`
  Using firmware drivers, the RTC is configured as Binary Counter mode. The RTC prescaler value that is passed to the RTC driver initialization function needs to be modified to match the RTC clock source selected in the Libero SoC flow. This is done by modifying the value of the RTC_PRES-CALER defined at the top of "main.c".

```
/* RTC_PRESCALER value for 1 MHz clock.
* In this demo, the RTC clock source is set to be 1 MHz. For different clock
source settings, adjust the RTC_PRESCALER accordingly */
#define RTC_PRESCALER (1000000µ - 1µ)
```

- nvm_access ();
  The fabric SRAM is initialized through a call to the nvm_access() function. Before entering Flash*Freeze mode, the nvm_access() function is called to initialize the fabric SRAM based on data client that is specified into the eNVM.

- SRAM_read ();
  Checking the fabric SRAM content after exiting from Flash*Freeze is done through a call to the SRAM_read() function.

2.5 Conclusion

This application note specified how to enter and exit Flash*Freeze on the SmartFusion2 device using the ".stp" programming file. The SRAM content retention capability during Flash*Freeze was also shown in this application note.

3 Appendix: References

The following references complement and help in understanding the relevant Microsemi SmartFusion2 SoC FPGA device features and flows that are demonstrated in this document.

- For information on the Flash*Freeze services provided by the System Controller, see the *UG0450:SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide.*
- For information on the Flash*Freeze technology supported by SmartFusion2 and IGLOO2 devices, see the *UG0444: SmartFusion2 and IGLOO2 Low Power Design User Guide.*
- For information on initializing SRAM using eNVM, see the *AC392: SmartFusion2 SoC FPGA SRAM Initialization from eNVM Application Note.*
- SoftConsole v4.0 is used in this application note. For more information on using SoftConsole, see the *TU0546: SoftConsole v4.0 and Libero SoC v11.7 Tutorial.*
- For complete information on the SmartFusion2 Security Evaluation Kit, see the *UG0594: M2S090TS-EVAL-KIT SmartFusion2 Security Evaluation Kit User Guide.*
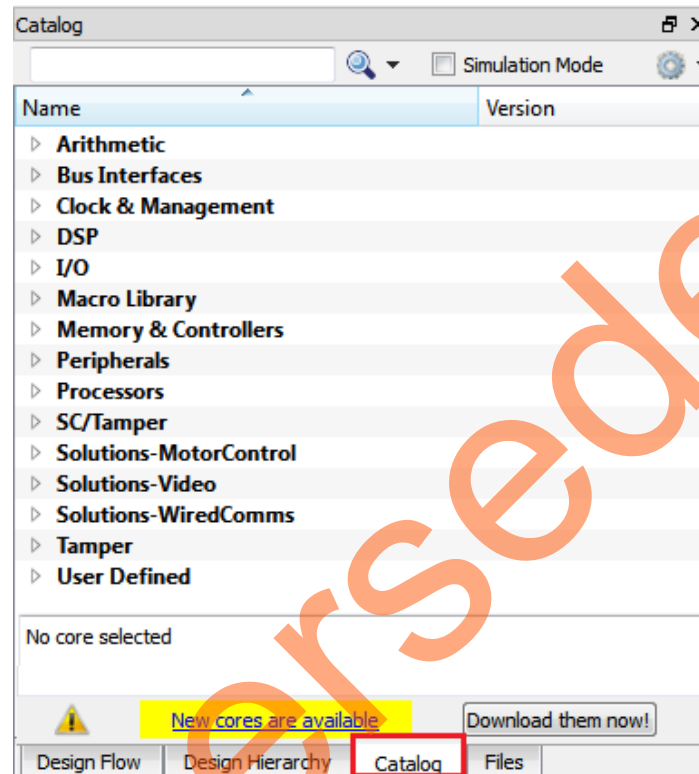
4   Appendix: Importing IP Core to User Vault

The following steps describe how to import the CoreResetP IP core to User Vault in Libero SoC.

1. Select the **Catalog** tab in Libero SOC as shown in the following figure.

Figure 19 • **Catalog Tab**

2. Click the **Settings** drop-down and select the **Add Core to Vault** option as shown in the Figure 20, page 18.

![Microsemi Power Matters.]

Figure 20 • **Selecting the Add Core to Vault Option**
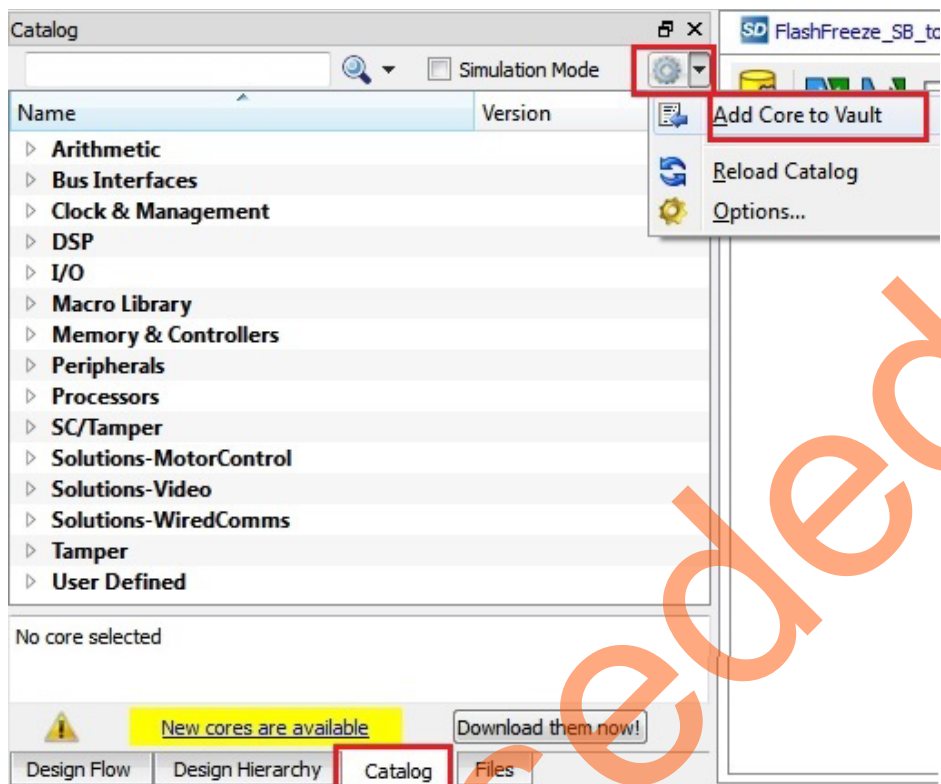
The **Add Core to Vault** dialog box opens. Change file type to core files (.ccz,.cpz) from the drop down list and navigate the IP core location as shown in the following figure.

Figure 21 • **Add Core to Vault Dialog Box**
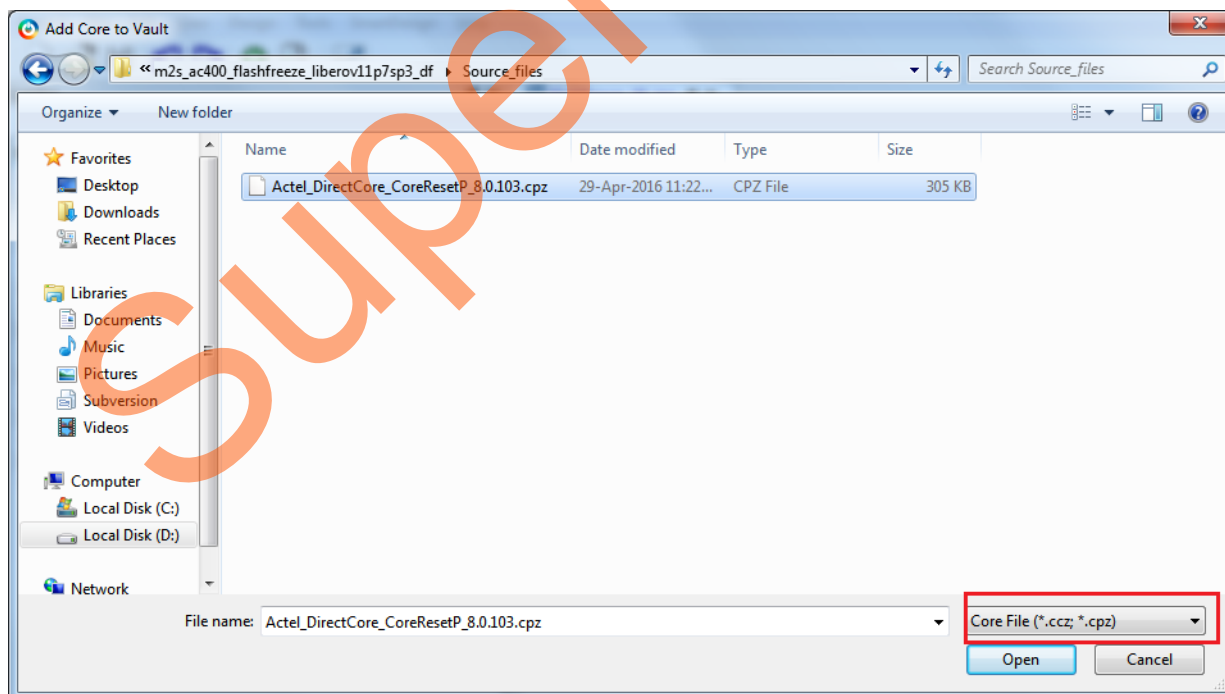
You have successfully imported the CoreResetP IP core to the User Vault.