SmartFusion2 Flash*Freeze Entry and Exit - With SoftConsole

Table of Contents

References

The following list of references was used in this document. The references complement and help in understanding the relevant Microsemi® SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device features and flows that are demonstrated in this document.

- SmartFusion2 System Controller User Guide
- SmartFusion2 Low Power Design User Guide
- SmartFusion2 eNVM Initialization Application Note
- SmartFusion2 Development Kit Board

Tools Required

The application design provided with this application note uses the following tools:

- Libero® System-on-Chip (SoC) v11.2
- SoftConsole v3.4
- Terminal emulator program (HyperTerminal or equivalent)

Introduction

SmartFusion2 devices provide an ultra-low static power solution through Flash*Freeze (F*F) technology. Entry into F*F mode retains all the static random-access memory (SRAM) and registers information and F*F exit mode achieves rapid recovery to Active mode.

One of the System Controller’s functions in the SmartFusion2 device is to handle the System Services requests through the communication block (COMM_BLK). The System Services are grouped into different services. Refer to the *SmartFusion2 System Controller User Guide* for more details. The SmartFusion2 device enters into F*F mode by using the F*F services request that the System Controller provides. Some of these options need to be set by the user during the design time, such as the clock source to be used as the standby clock source for the microcontroller subsystem (MSS) during F*F mode. Furthermore, the fabric SRAM state can also be defined during F*F mode.

The fabric SRAM state during F*F can either be "Sleep" or "Suspend". In 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 SRAMs is not lost. In Sleep mode, the LSRAM and uSRAM contents are not retained. The standby clock source for the MSS during F*F and the state are configured in the F*F hardware settings in Libero SoC.

There are different ways to exit from F*F mode. Exit from F*F mode 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). The state and the role that I/Os play during F*F mode must be specified during the design time using the Libero SoC. There are three different settings available. These settings are categorized as the I/O state in F*F mode, I/O availability in F*F mode, and I/O role in exiting from F*F mode. Depending on the type of the I/O, some or all of those options may not be available. Refer to the *SmartFusion2 Low Power Design User's Guide* for more details.

This application note describes how to set the different user defined settings during the design time using the Libero SoC software. It also describes in detail how to enter F*F mode using the System Services through ARM$^{®}$ Cortex™-M3 firmware and exiting from F*F mode using different mechanisms, such as external I/Os events and/or RTC time-out event.

Managing the MDDR, FDDR, or SERDES before and after F*F mode, power measurements, or using fabric master option to enter into F*F mode are not discussed in this document.

Design Description

The design example consists of the MSS, a counter, SRAM wrapper logic, IP cores (CoreAHBLite, CoreAHBToAPB3, CoreResetP, and CoreAPB3), 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 used to demonstrate the state of the fabric SRAM content after exiting from F*F mode. Refer to the *SmartFusion2 eNVM Initialization Application Note* for more details on how to use eNVM to initialize fabric SRAMs. The CoreResetP handles the sequencing of reset signals in the device. Refer to the CoreResetP Handbook for more details on this core.

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). Refer to the top-level block diagram in Figure 1 on page 3. The M3_CLK, FIC_0_CLK, and APB_1_CLK are configured as 100 MHz clocks generated from the MSS_CCC.

In Active mode (non F*F), 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 F*F mode. The LEDs ports assignments are shown in Table 1.

Table 1 • **LED to Pins Assignments (Development Kit Board)**

Counter Output	Package Pin
LED_1	A18
LED_2	B18
LED_3	D18
LED_4	E18

The top-level block diagram shows the main blocks used in the design, as shown in Figure 1.

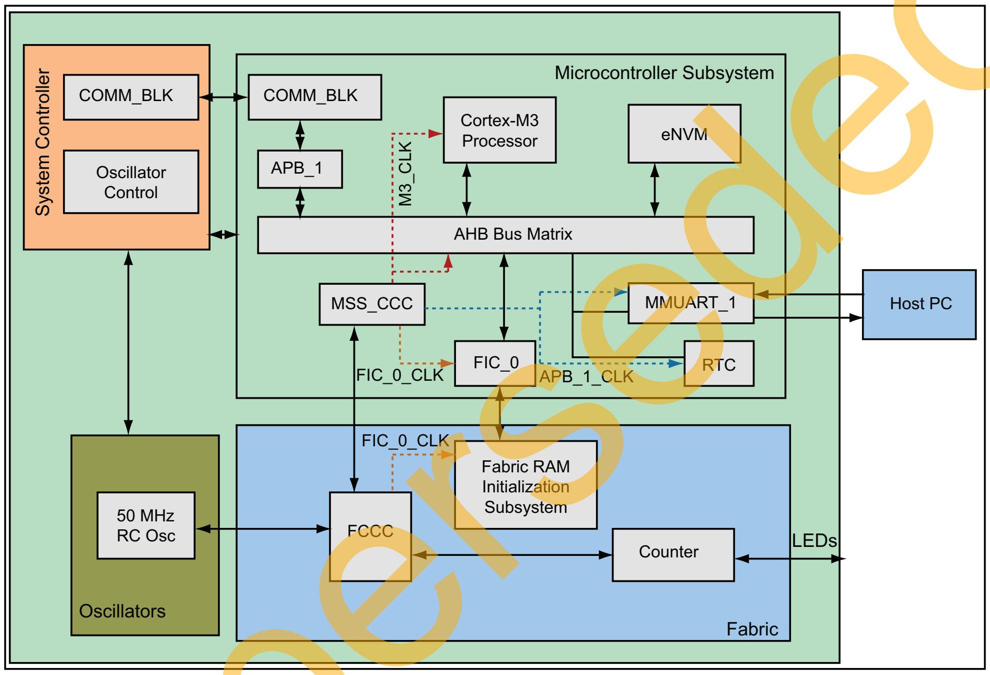


Figure 1 • **Top-Level Block Diagram of the Design**

Entering into F*F Mode

Entering into F*F 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 F*F service requests the System Controller to execute the F*F entry sequence. When the F*F service begins execution, the System Controller informs the MSS by sending a command byte E0H that F*F 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. Refer to the "Flash*Freeze Service" section in the *SmartFusion2 System Controller User's Guide* for more details.

As the F*F 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. Refer to the "Software Implementation" section on page 9 for more details.

Exiting from F*F Mode

Exiting from F*F 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 F*F exit in two ways.

- I/O Activity: Force F*F exit up on an activity (Wake_On_Change)
- I/O Signature: Force F*F exit up on a signature (Wake_On_1/Wake_On_0) match in which the I/O participates with other I/Os to trigger F*F 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 F*F 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 F*F, it must be {(All Wake-on-0 **ANDed**) **ANDed** with (All Wake-on-1 **ANDed**)} **ORed** with (All Wake-on-Change ORed).

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 F*F 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.

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 F*F 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 F*F mode. I/Os can be a mixture of different signature settings. An I/O can be configured to participate in the F*F exit upon a 0-to-1 or it can be configured to participate in the F*F 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 F*F exit mechanism are tristated or held to the previous state (LAST_VALUE) before entering F*F mode. The selection is set using **I/O state in Flash*Freeze mode** column options in the I/O Editor using the Libero SoC, as shown in Figure 8 on page 8.

SW10 (four different dual in-line package (DIP) switches) on the Development 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 2. SW1 on the Development Kit board is used to demonstrate the transition (Wake_On_Change) wake-up event mechanism, as shown in Figure 2.

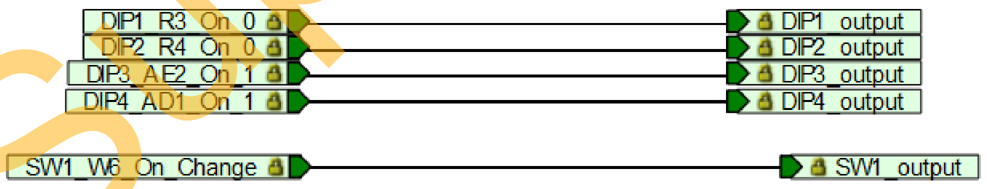


***Figure 2* • DIP Switches and the SW1 Connectivity in SmartDesign**

To demonstrate the RTC wake-up event mechanism, the RTC is configured in Binary mode. Refer to the "Software Implementation" section on page 9 for more information. 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 F*F 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). Refer to the "Hardware Implementation" section on page 5 for more information.

Hardware Implementation

The hardware implementation involves configuring the MSS and the necessary F*F 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, CoreAPB3), and FCCC, as shown in Figure 3. 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. A Data Storage client 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 F*F. Refer to the *SmartFusion2 eNVM Initialization Application Note* for more information on how to use the eNVM to initialize fabric SRAMs.

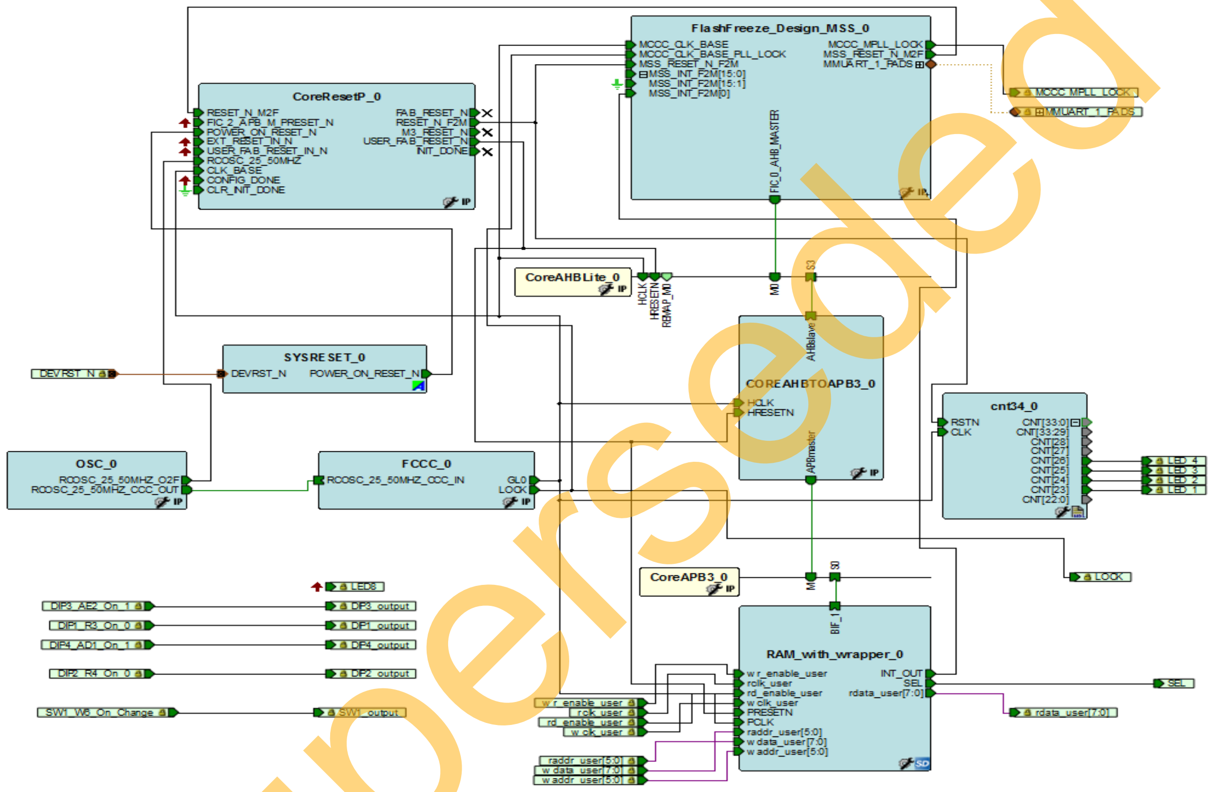
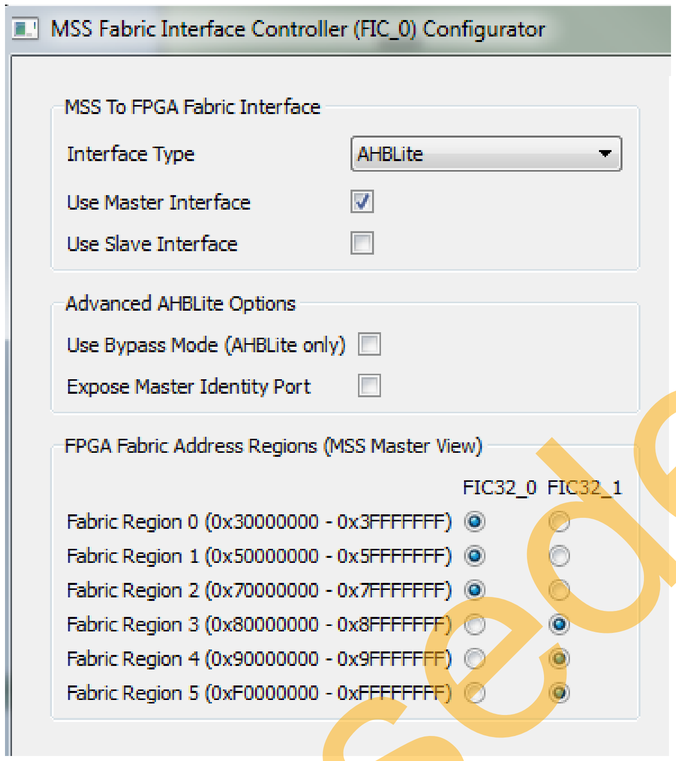


Figure 3 • **Top-Level Hardware Design**

The FIC_0 interface is configured as AHBL master interface, as shown in Figure 4.


MSS Fabric Interface Controller (FIC_0) Configurator

MSS To FPGA Fabric Interface

Interface Type	AHBLite
Use Master Interface	✓
Use Slave Interface	

Advanced AHBLite Options

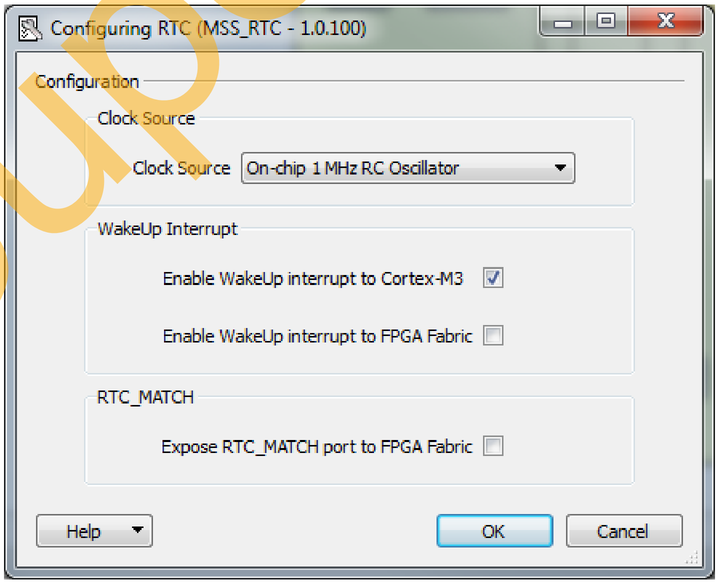
Use Bypass Mode (AHBLite only)	
Expose Master Identity Port	

FPGA Fabric Address Regions (MSS Master View)

	FIC32_0	FIC32_1
Fabric Region 0 (0x30000000 - 0x3FFFFFFF)	●	
Fabric Region 1 (0x50000000 - 0x5FFFFFFF)	●	
Fabric Region 2 (0x70000000 - 0x7FFFFFFF)	●	
Fabric Region 3 (0x80000000 - 0x8FFFFFFF)		●
Fabric Region 4 (0x90000000 - 0x9FFFFFFF)		●
Fabric Region 5 (0xF0000000 - 0xFFFFFFFF)		●

***Figure 4* • 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 Coretex-M3** is selected, as shown in Figure 5.


Configuring RTC (MSS_RTC - 1.0.100)

Configuration

Clock Source

Clock Source: On-chip 1 MHz RC Oscillator

WakeUp Interrupt

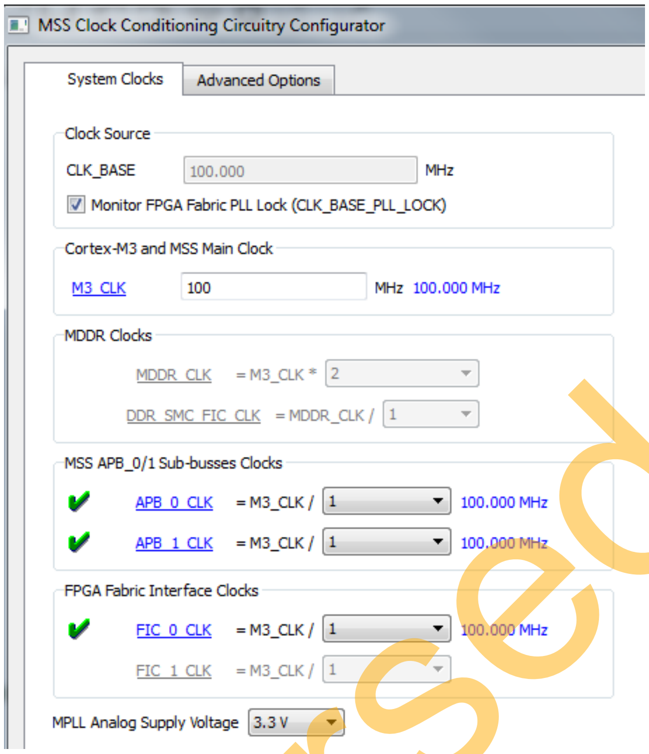
Enable WakeUp interrupt to Cortex-M3	✓
Enable WakeUp interrupt to FPGA Fabric	

RTC_MATCH

Expose RTC_MATCH port to FPGA Fabric	

***Figure 5* • 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. Figure 6 shows the system clocks configurations for the M3_CLK, APB_1_CLK, and FIC_0_CLK clock settings.



Clock Source
CLK_BASE 100.000 MHz
[✓] Monitor FPGA Fabric PLL Lock (CLK_BASE_PLL_LOCK)

Cortex-M3 and MSS Main Clock
M3_CLK 100 MHz 100.000 MHz

MDDR Clocks
MDDR_CLK = M3_CLK * 2
DDR_SMC_FIC_CLK = MDDR_CLK / 1

MSS APB_0/1 Sub-busses Clocks
✓ APB_0_CLK = M3_CLK / 1 100.000 MHz
✓ APB_1_CLK = M3_CLK / 1 100.000 MHz

FPGA Fabric Interface Clocks
✓ FIC_0_CLK = M3_CLK / 1 100.000 MHz
FIC_1_CLK = M3_CLK / 1

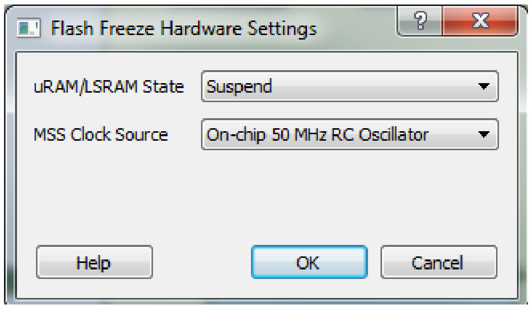
MPLL Analog Supply Voltage 3.3 V

Figure 6 **• MSS CCC System Clocks Configurations**

The standby clock source for the MSS in F*F mode and the state of the SRAMs (uRAM and LSRAM) during F*F mode are configured using the Flash*Freeze Hardware Settings dialog in the Libero SoC software, as shown in Figure 7. 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 F*F mode:

- On-chip 1 MHz RC oscillator
- On-chip 50 MHz RC oscillator
- External 32 KHz crystal oscillator

Note: During F*F mode, the external 32 KHz crystal oscillator option as the MSS clock source for the M2S050 devices is not supported.



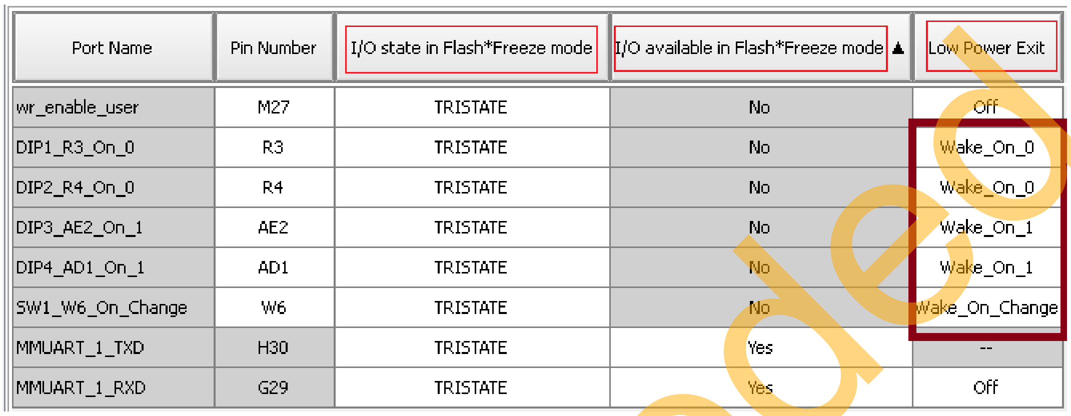
uRAM/LSRAM State: Suspend
MSS Clock Source: On-chip 50 MHz RC Oscillator

Figure 7 **• Flash*Freeze Hardware Settings Dialog**

The I/Os F*F exit mechanism is specified using the Low Power Exit setting in the I/O Editor in the Libero SoC, as shown in Figure 8.

Note:

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



Port Name	Pin Number	I/O state in Flash*Freeze mode	I/O available in Flash*Freeze mode	Low Power Exit
wr_enable_user	M27	TRISTATE	No	Off
DIP1_R3_On_0	R3	TRISTATE	No	Wake_On_0
DIP2_R4_On_0	R4	TRISTATE	No	Wake_On_0
DIP3_AE2_On_1	AE2	TRISTATE	No	Wake_On_1
DIP4_AD1_On_1	AD1	TRISTATE	No	Wake_On_1
SW1_W6_On_Change	W6	TRISTATE	No	Wake_On_Change
MMUART_1_TXD	H30	TRISTATE	Yes	--
MMUART_1_RXD	G29	TRISTATE	Yes	Off

***Figure 8* • Specifying I/O State and Functionality Options Using I/O Editor**

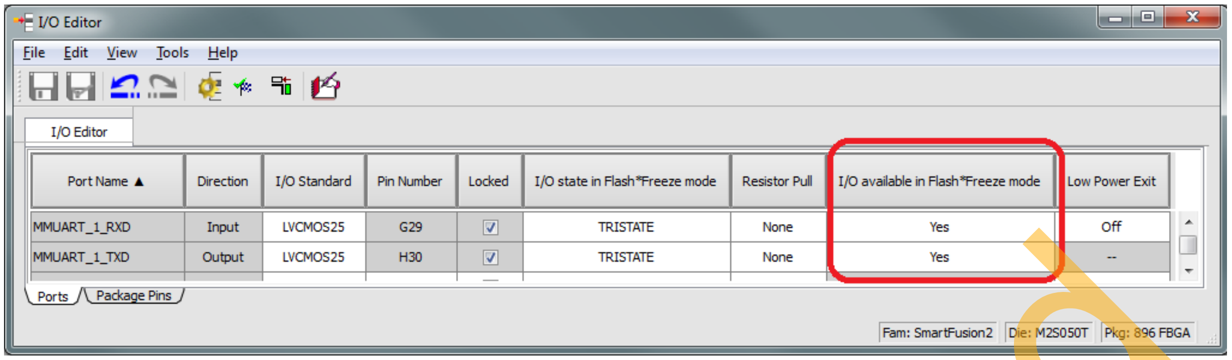
The F*F exit behavior of input I/Os (DIP1-4) and SW1 are configured using the I/O Editor in the Libero SoC, as shown in Figure 8. The DIP switches to package pin assignments are shown in Table 2.

***Table 2* • DIP Switches to Package Pins Assignments**

Input DIP Switch and SW1	Package Pin
DIP1	R3
DIP2	R4
DIP3	AE2
DIP4	AD1
SW1	W6

The MMUART_1 is used to read and write to the HyperTerminal window. On the Development Kit board, the MMUART_0 TX and RX are connected to the mini-B USB through the fabric and fabric I/Os. During F*F mode, the fabric and I/Os are not available. The MMUART_0 cannot be used as the interface to read and write to HyperTerminal during F*F mode. As such, MMUART_1 is used, and the RXD and TXD ports are configured using the I/O Editor to be available during F*F mode, as shown in Figure 9 on page 9.

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



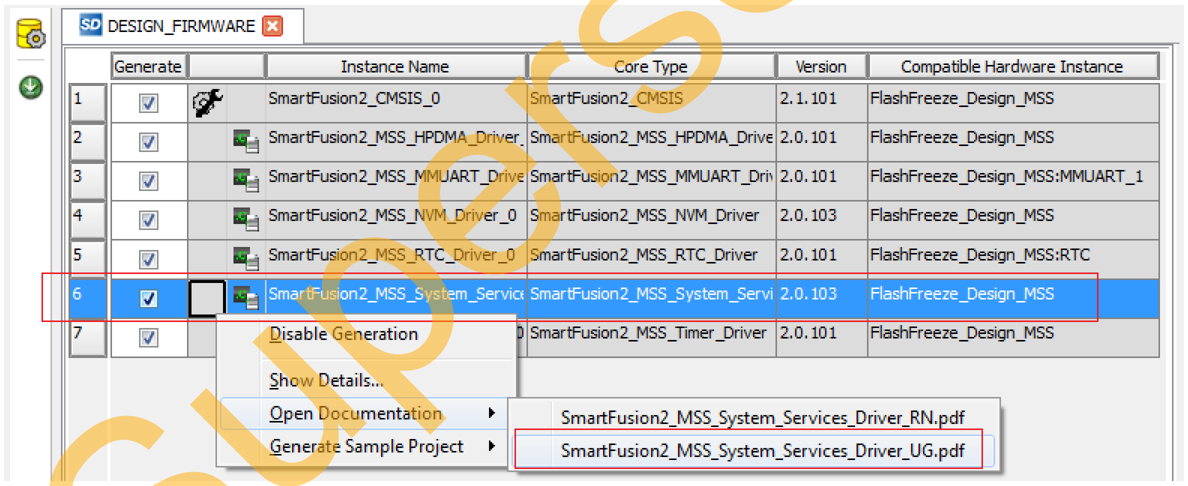
Port Name ▲	Direction	I/O Standard	Pin Number	Locked	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Low Power Exit
MMUART_1_RXD	Input	LVCMOS25	G29	✓	TRISTATE	None	Yes	Off
MMUART_1_TXD	Output	LVCMOS25	H30	✓	TRISTATE	None	Yes	--

Fam: SmartFusion2 Die: M2S050T Pkg: 896 FBGA

Figure 9 **• Configuring MMUART_1 Ports to be Available During F*F**

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 F*F mode. Figure 10 shows the System Services driver. Refer to the **SmartFusion2 MSS System Services Driver User Guide** for more information. Right-click **SmartFusion2_MSS_System_Services_Driver** to access the user guide, as shown in Figure 10.


DESIGN_FIRMWARE
Disable Generation
Show Details...
Open Documentation
Generate Sample Project
SmartFusion2_MSS_System_Services_Driver_RN.pdf
SmartFusion2_MSS_System_Services_Driver_UG.pdf

	Generate		Instance Name	Core Type	Version	Compatible Hardware Instance
1	✓		SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.1.101	FlashFreeze_Design_MSS
2	✓		SmartFusion2_MSS_HPDMA_Driver_	SmartFusion2_MSS_HPDMA_Drive	2.0.101	FlashFreeze_Design_MSS
3	✓		SmartFusion2_MSS_MMUART_Drive	SmartFusion2_MSS_MMUART_Driv	2.0.101	FlashFreeze_Design_MSS:MMUART_1
4	✓		SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.0.103	FlashFreeze_Design_MSS
5	✓		SmartFusion2_MSS_RTC_Driver_0	SmartFusion2_MSS_RTC_Driver	2.0.101	FlashFreeze_Design_MSS:RTC
6	✓		SmartFusion2_MSS_System_Servic	SmartFusion2_MSS_System_Servi	2.0.103	FlashFreeze_Design_MSS
7	✓			SmartFusion2_MSS_Timer_Driver	2.0.101	FlashFreeze_Design_MSS

Figure 10 **• 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 F*F mode. The options parameter can be used to power-down different parts of SmartFusion2, as shown in Table 3.

***Table 3* • F*F 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_PRESCALER 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 (1000000u - 1u)

```
nvm_access ();
```

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

```
SRAM_read ();
```

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

Running the Design

The design example demonstrates the following options:

- Entering into F*F mode
- Initializing the SRAM from eNVM
- Checking the content of the SRAM post F*F based on whether the SRAM was put into Sleep or Suspend modes
- Exiting from F*F by the means of RTC, I/O activity, or I/Os signature.

The design example is designed to run on the SmartFusion2 Development Kit board with pre-production (PP) devices or later. Refer to www.microsemi.com/index.php?option=com_content&id=1645&lang=en&view=article for more detailed board information.

Board Jumper Settings

Connect the jumpers on the SmartFusion2 Development Kit as described in Table 4. While making the jumper connections, the power supply switch (SW7) on the board should be in OFF position.

Table 4 • **SmartFusion2 SoC FPGA Development Kit Jumper Connections**

From		To	
Jumper	**Pin**	**Jumper**	**Pin**
J129	3	J197	2
J133	3	J188	2

These jumper connections are used to connect the MMUART_1 RX and TX port to the Future Technology Devices International (FTDI) UART ports.

Host PC to Board Connections

1. Connect the FlashPro4 programmer to the FP4 HEADER J59 connector of the SmartFusion2 Development Kit board.
2. Connect one end of the USB mini-B (FTDI interface) cable to the J24 connector provided on the SmartFusion2 Development Kit board. Connect the other end of the USB cable to the host PC.

USB Driver Installation

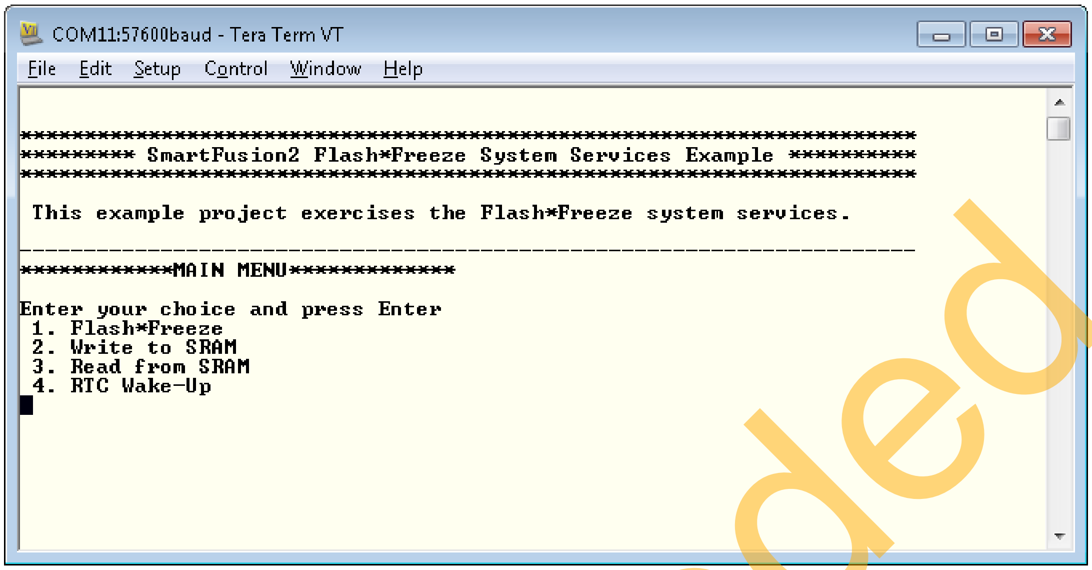
For serial terminal communication through FTDI mini USB cable, install the FTDI D2XX driver. The drivers and installation guide can be downloaded from www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.

Make sure that the USB to UART bridge drivers are detected (can be verified in Device Manager in the system).

Run the Design Steps

1. Connect the power supply to the J18 connector and FlashPro Programmer.
2. Change the power supply SW7 switch to **ON**.
3. Program the SmartFusion2 Development Kit Board with the generated or provided *.stp file (refer to "Appendix A – Design Files" on page 17) using FlashPro.
4. Invoke the SoftConsole3.4 Integrated Design Environment (IDE) by clicking Write Application Code under Develop Firmware in the Libero SoC tool and launch the debugger.
5. Start HyperTerminal program with the baud rate set to 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If the PC does not have HyperTerminal, use any free serial terminal emulation program, such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring HyperTerminal, Tera Term, and PuTTY.

When the debugger is run in SoftConsole, HyperTerminal window displays a message followed by a menu to enter a choice, as shown in Figure 11.


COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
*********************************************************************
********* SmartFusion2 Flash*Freeze System Services Example **********
*********************************************************************

 This example project exercises the Flash*Freeze system services.

---------------------------------------------------------------------
************MAIN MENU**************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

Figure 11 **• Message with Menu Options**

Table 5 shows a description summary of the results of each choice.

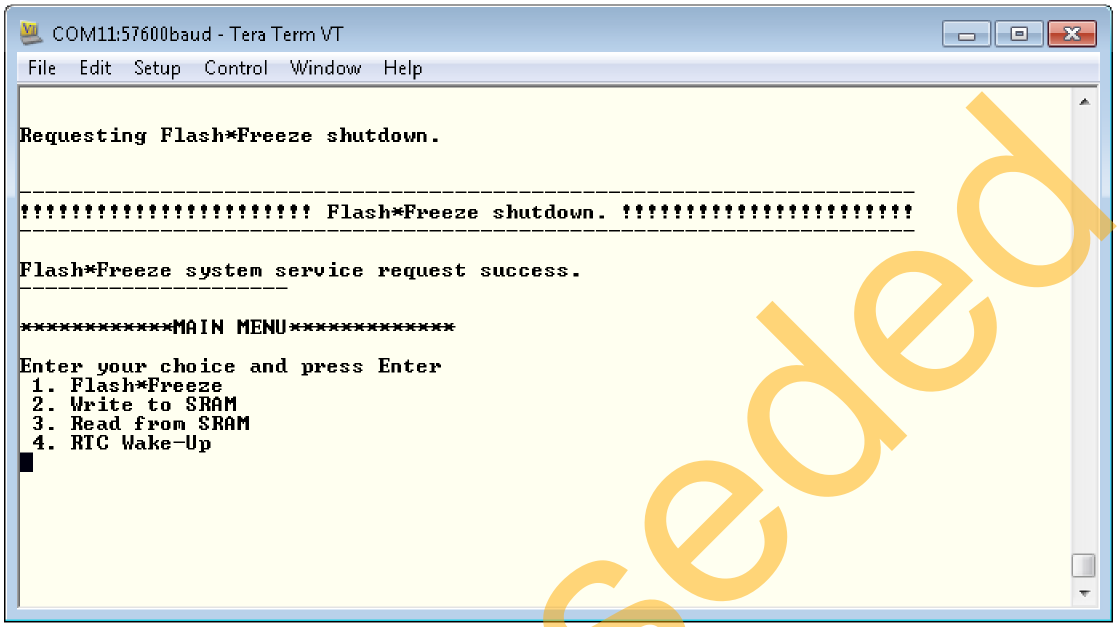
Table 5 **• Menu Options Descriptions**

Options	Description
1	When selecting this option, the SmartFusion2 device is put into F*F mode by powering down the FPGA fabric and MPLL.
2	This option is to demonstrate the state of the SRAM after exiting from F*F mode depending on whether the "Sleep" or "Suspend" option was selected as the SRAM state during F*F. When selecting this option, the fabric SRAM is initialized from the eNVM.
3	This option reads back from the fabric SRAM after the SmartFusion2 device exits from F*F mode. If the SRAM state was selected as "Suspend" during F*F mode, then the content before entering into F*F persists. If the state of the SRAM was selected as "Sleep", then the content of the SRAM is not retained during F*F.
4	Use this option to exit from F*F by generating an RTC interrupt.

Entering F*F Mode and Using RTC to exit F*F

1. Select **1** (Flash*Freeze). This will put the device state into F*F mode, as shown in Figure 12 on page 13.

   Note that the LEDs on the board stop toggling, which indicates that the SmartFusion2 has entered into F*F mode.

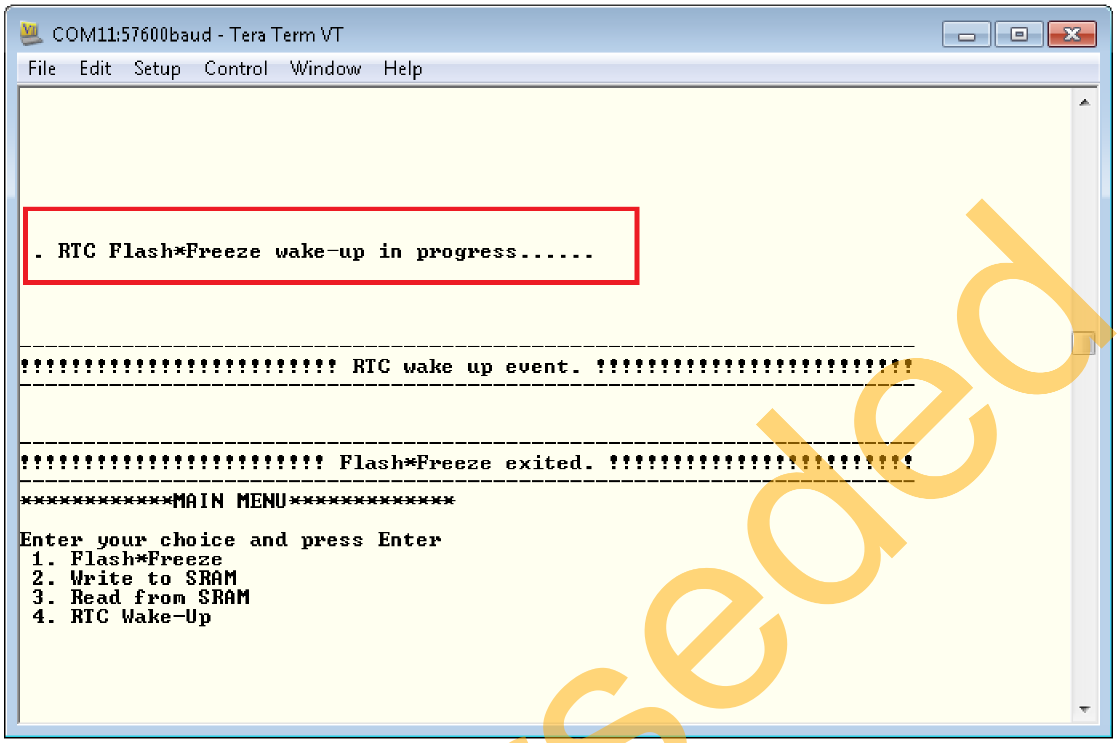

COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
Requesting Flash*Freeze shutdown.

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

Flash*Freeze system service request success.
---------------------

************MAIN MENU**************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

Figure 12 • **Flash*Freeze Shutdown**

2. Select **4** (RTC Wake-Up). The RTC is initialized with an RTC_PRESCALER value. The RTC counter is reset then the RTC is configured as Single Shot Alarm mode. The counter is then started. When the counter reaches its set value, an interrupt is triggered and the device wakes up from F*F mode, as shown in Figure 13 on page 14.

Note that the LEDs on the board start toggling, which indicates that the SmartFusion2 exited from F*F mode.


COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
. RTC Flash*Freeze wake-up in progress......

------------------------------------------------------------------------
!!!!!!!!!!!!!!!!!!!!!!!!! RTC wake up event. !!!!!!!!!!!!!!!!!!!!!!!!!!
------------------------------------------------------------------------

------------------------------------------------------------------------
!!!!!!!!!!!!!!!!!!!!!!!! Flash*Freeze exited. !!!!!!!!!!!!!!!!!!!!!!!!!
------------------------------------------------------------------------
************MAIN MENU*************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

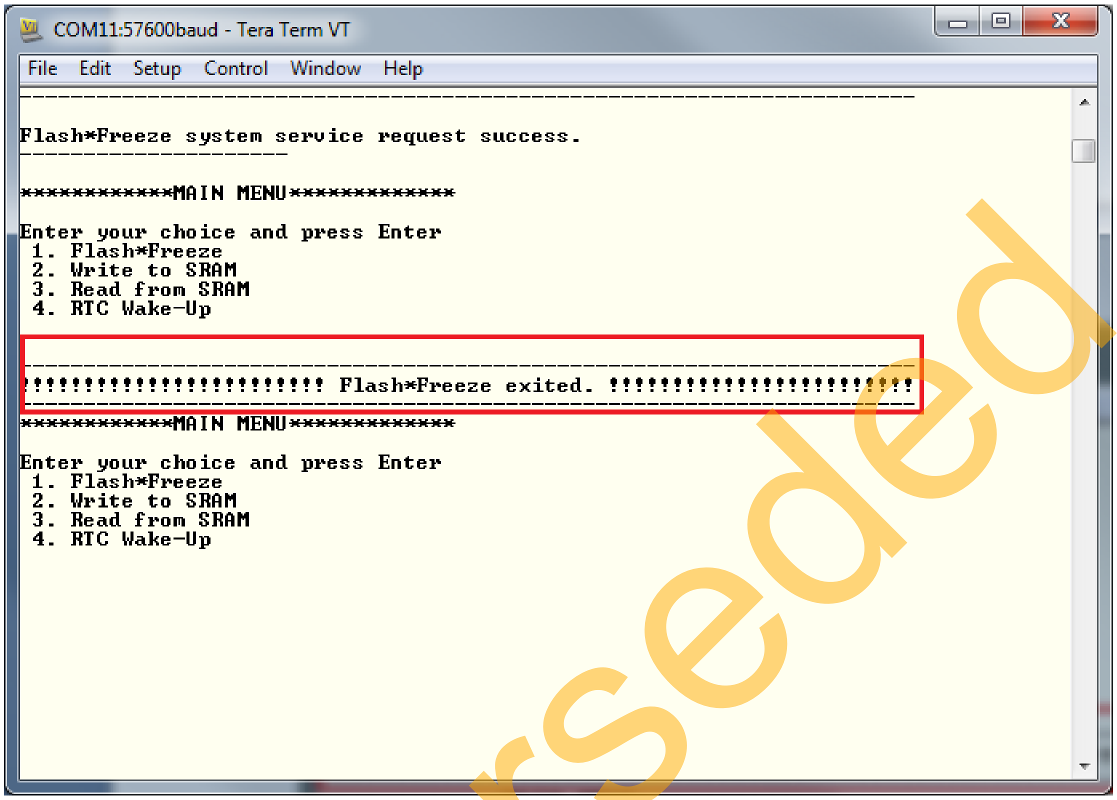
Figure 13 • **RTC F*F Exit Event**

*Entering F*F Mode and Using External I/O Activity (Wake_On_Change) to Exit F*F Mode*

The following steps demonstrate how to exit from F*F using external I/O activity. The activity could be a change from 1-to-0 or a 0-to-1. This is set on per I/O basis in the I/O Editor by setting the Wake_On_Change attribute. For the purpose of this demo, SW1 (package pin W6) is used.

1. Select **1** (Flash*Freeze). This puts the device into F*F mode, as shown in Figure 12 on page 13. Note that the LEDs on the board stop toggling, which indicates that the SmartFusion2 device entered into F*F mode.

2. To wake up the device from F*F mode, press SW1 switch on the board. This indicates a change on W6 package pin I/O and wakes up the device from F*F, as shown in Figure 14.


COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
--------------------------------------------------------------------------

Flash*Freeze system service request success.
---------------------

************MAIN MENU*************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up

--------------------------------------------------------------------------
!!!!!!!!!!!!!!!!!!!!!!! Flash*Freeze exited. !!!!!!!!!!!!!!!!!!!!!!!
--------------------------------------------------------------------------
************MAIN MENU*************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

Figure 14 • **I/O Activity F*F Exit Event**

*Entering F*F Mode and Using External I/O Signature (Wake_On_1/Wake_On_0) to Exit F*F Mode*

The following steps demonstrate how to exit from F*F using signature I/O matching. One or more I/Os can be configured to wake-up the device based on a change from 0- to-1 or 1- to-0 or a combination of both.

When more than one I/O is configured to participate in the signature wake-up, it is a logical **AND** of all I/Os. For the purpose of this demo, a set of DIP switches are used. Two DIP switches are configured as Wake_On_1 and two are configured as Wake_On_0. All four switches must meet the criteria for the device to exit F*F mode.

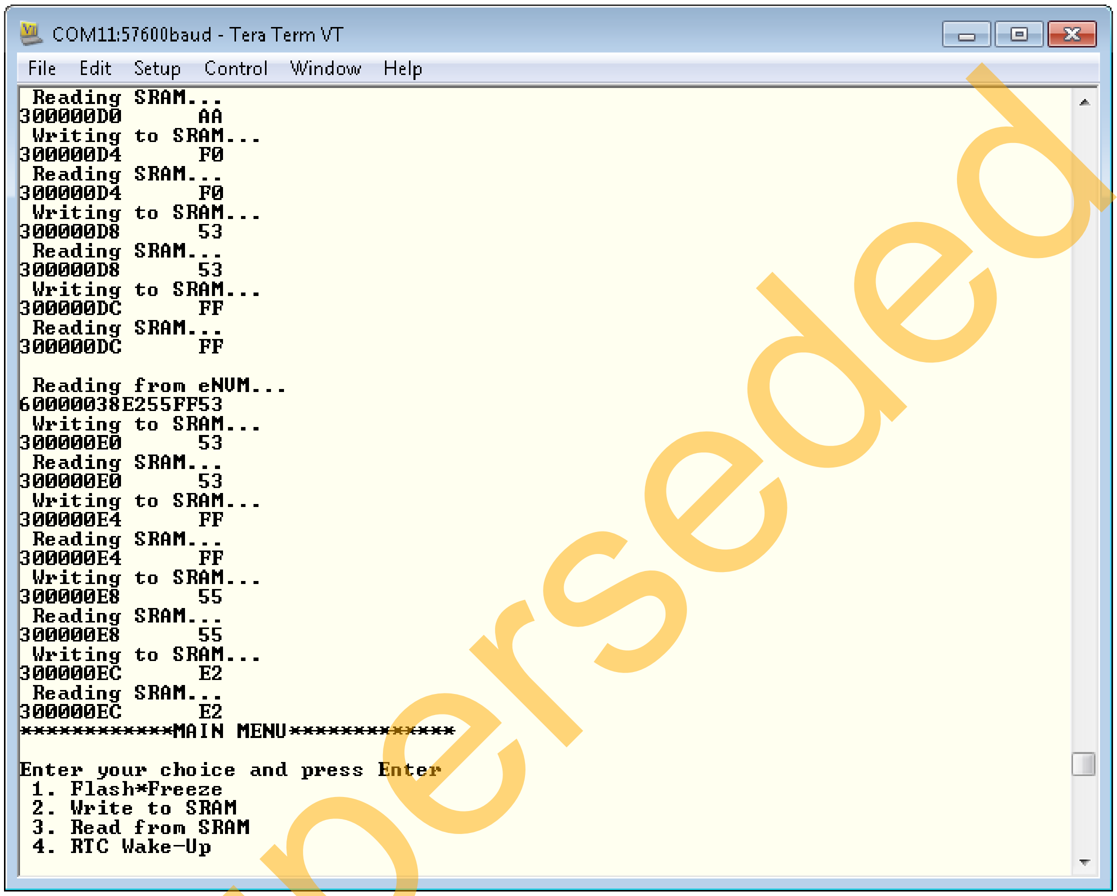
1. Select **1** (Flash*Freeze). This puts the device into F*F mode, as shown in Figure 12 on page 13. Note that the LEDs on the board stop toggling, which indicates that the SmartFusion2 entered into F*F mode.
2. To wake-up the device from F*F mode, toggle DIP switches 1 and 2 to 0 position (ON) **AND** toggle DIP switches 2 and 3 to 1 position (OFF). Up on this setting, the device exits from F*F mode.

Note: The DIP switches combination setting in step 2 constantly keeps the device in active mode, since that combination is configured to wake-up the device. Before proceeding to the next step, ensure that the combination setting of the DIP switches is different than what is described in step 2.

SRAM Content After Entering into F*F Mode

This step demonstrates that the SRAM content is retained and not lost while the device is in F*F mode. The SRAM was set to be in "Suspend" mode during F*F. Refer to "Hardware Implementation" section on page 5 for more information.

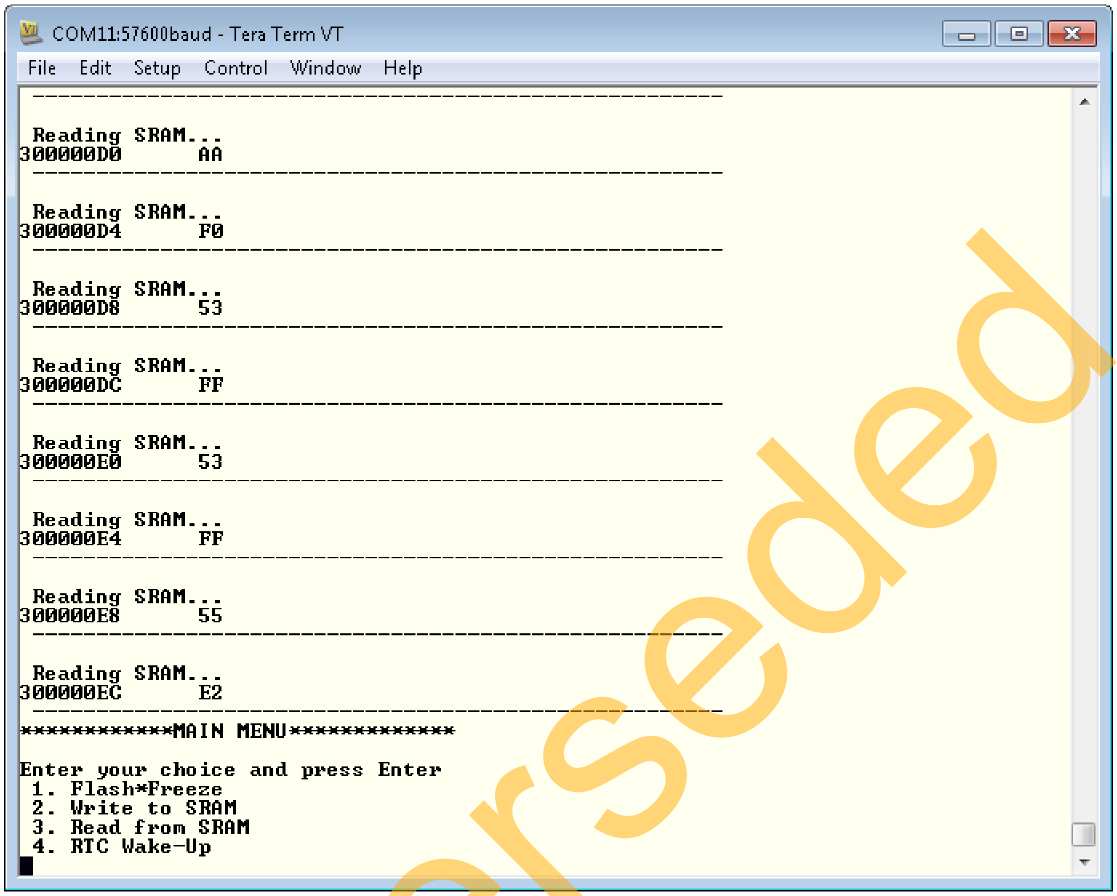
1. Select **2** (Write to SRAM). This step reads from the eNVM and writes to the SRAM, as shown in Figure 15.


COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
 Reading SRAM...
300000D0       AA
 Writing to SRAM...
300000D4       F0
 Reading SRAM...
300000D4       F0
 Writing to SRAM...
300000D8       53
 Reading SRAM...
300000D8       53
 Writing to SRAM...
300000DC       FF
 Reading SRAM...
300000DC       FF

 Reading from eNVM...
60000038E255FF53
 Writing to SRAM...
300000E0       53
 Reading SRAM...
300000E0       53
 Writing to SRAM...
300000E4       FF
 Reading SRAM...
300000E4       FF
 Writing to SRAM...
300000E8       55
 Reading SRAM...
300000E8       55
 Writing to SRAM...
300000EC       E2
 Reading SRAM...
300000EC       E2
*************MAIN MENU*************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

Figure 15 **• Reading from eNVM and Writing to SRAM**

2. Select **1** (Flash*Freeze). This puts the device into F*F mode, as shown in Figure 12 on page 13. Note that the LEDs on the board stop toggling, which indicates that the SmartFusion2 entered into F*F mode.
3. Select **4** (RTC Wake-Up) to exit from F*F mode.
4. Select **3** (Read from SRAM) to read the SRAM content after the device exits from F*F mode. In this design, the SRAM is set for "Suspend" mode during F*F mode so the content of the SRAM is retained. Thus when reading the SRAM content after F*F exit, it is the same data that was stored into the SRAM before entering into F*F mode, as shown in Figure 16 on page 17.


COM11:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
 ------------------------------------------------------------

 Reading SRAM...
300000D0      AA
 ------------------------------------------------------------

 Reading SRAM...
300000D4      F0
 ------------------------------------------------------------

 Reading SRAM...
300000D8      53
 ------------------------------------------------------------

 Reading SRAM...
300000DC      FF
 ------------------------------------------------------------

 Reading SRAM...
300000E0      53
 ------------------------------------------------------------

 Reading SRAM...
300000E4      FF
 ------------------------------------------------------------

 Reading SRAM...
300000E8      55
 ------------------------------------------------------------

 Reading SRAM...
300000EC      E2
 ------------------------------------------------------------
************MAIN MENU*************

Enter your choice and press Enter
 1. Flash*Freeze
 2. Write to SRAM
 3. Read from SRAM
 4. RTC Wake-Up
```

Figure 16 • **Reading SRAM Content After F*F**

The data read from the SRAM at a particular address is the same data that was written into the SRAM before entering into F*F mode.

Conclusion

This application note describes how to put the SmartFusion2 device into F*F mode using System Services and demonstrates the different options that can be used to wake-up the SmartFusion2 device from F*F mode. In addition, the application note also shows how to set different hardware behavior during F*F at design time, and demonstrates the effect of the F*F on the fabric SRAM content depending on the user defined F*F hardware settings in the Libero SoC.

Appendix A – Design Files

The design files can be downloaded from the Microsemi SoC Products Group website:

www.microsemi.com/soc/download/rsc/?f=M2S_AC400_DF.

The design file consists of Libero SoC Verilog project, SoftConsole software project, and programming files (*.stp) for SmartFusion2 Development Kit board. Refer to the **Readme.txt** file included in the design file for the directory structure and description.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 1 (January 2014)	Updated the document for Libero SoC v 11.2 software release (SAR 53247).	NA
Revision 0 (May 2013)	Initial Release.	NA

Note: **The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

51900268-1/01.14