

IGLOO2 Flash*Freeze Entry and Exit - Libero SoC v11.4

Table of Contents

Purpose	1
Introduction	1
References	2
Design Requirements	2
Design Description	2
Entering into F*F Mode	4
Exiting from F*F Mode	4
Hardware Implementation	5
Running the Design	10
Steps to Run the Design	10
Conclusion	15
Appendix A - Design Files	16
List of Changes	17

Purpose

This application note describes the methods and steps of how to enter and exit Flash*Freeze (F*F) mode. The application note further shows how to set different user-defined settings that define the behavior of static random-access memory (SRAM) blocks during F*F entry and exit modes using Libero® System-on-Chip (SoC) software. It also describes how to use System Services provided by the CoreSystemServices soft IP to enter into F*F mode.

Introduction

Microsemi® IGLOO®2 field programmable gate array (FPGA) devices provide an ultra-low static power solution through F*F technology. Entry into F*F mode retains all the SRAM and registers information and F*F exit mode achieves rapid recovery to Active mode.

One of the functions of the System Controller in the IGLOO2 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 [IGLOO2 FPGA System Controller User Guide](#) for more details. The IGLOO2 device enters into F*F mode by using the F*F services request that the System Controller provides. You can set some of the F*F hardware settings options during the design time, such as the clock source to be used as the standby clock source for the high performance memory subsystem (HPMS) during F*F or defining the state of the fabric SRAM during F*F mode.

The HPMS stand by clock source and the state of the SRAMs are configured in the F*F hardware settings in the Libero SoC software. 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. That is, when the device exits F*F mode, the contents of the SRAMs are retained. In **Sleep** mode, the SRAMs contents are not retained. Exiting from F*F is achieved through user configurable mechanism through external I/O events (either transitions or pattern matching on I/Os). The state and the role that I/Os play during F*F must be specified during the design time using 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 these options may not be available. Refer to the [IGLOO2 FPGA Low Power Design User 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 the CoreSysServices soft IP, which provides access to the System Services. The CoreSysServices soft IP communicates with the COMM_BLK through one of the fabric interfaces controllers (FICs). Each System Service has a service request phase and a response phase. For more details, refer to the **CoreSysServices IP Handbook** which can be accessed through Libero SoC software. Managing the MDDR, FDDR, or SERDES before and after F*F mode, power measurements, are not discussed in this document.

References

The following list of references is used in this document. The references complement and help in understanding the relevant Microsemi IGLOO2 FPGA device flows and features that are demonstrated in this document.

- [IGLOO2 FPGA System Controller User Guide](#)
- [IGLOO2 FPGA Low Power Design User Guide](#)
- [IGLOO2 Evaluation Kit](#)

Design Requirements

Table 1 shows the design requirements.

Table 1 • Design Requirements

Design Requirements	Description
Hardware Requirements	
IGLOO2 Evaluation Kit	Rev C, Rev D, or later
Host PC	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.4
FlashPro programming software	v11.4
Host PC Drivers	USB to UART drivers
Set the following SmartDebug flag before launching the Libero SoC v11.4 software: data SMART_DEBUG_DISABLE_JTAG_RESET 1	SmartDebug: http://soc.microsemi.com/kb/article.aspx?id=KI8956

Design Description

The design example consists of the HPMS configured using System Builder, a counter, SRAM wrapper logic, IP cores (CoreSysServices, CoreAHBLite, CoreAHBToAPB3, and CoreAPB3), FLASH_FREEZE macro, fabric AHB master, on-chip 1 MHz RC oscillator, fabric CCC (FCCC), and F*F request and command generator logic (FF_BLKs). The fabric AHB master along with the SRAM wrapper (AHBMASTER_FIC_RAM) 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 and slave interfaces using the AHB master in the fabric. 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.

In Active mode (non F*F), the HPMS_CCC is configured to provide a 100 MHz clock that is sourced from the FPGA fabric through the CLK_BASE port. The FCCC is configured to provide the 50 MHz CLK_BASE reference. The on-chip 1 MHz oscillator is the reference clock source for the FCCC.

The CoreSysServices IP is configured to use only the F*F service option. It sends the F*F command to the System Controller whenever it receives the F*F request enable and command from the FF_BLKs logic. The FF_BLKs logic generates the F*F request and command based on the F*F entry input signal (ff_trig). The FF_BLKs logic also monitors the busy signal from the CoreSysServices IP and the FF_TO_START signal from the FLASH_FREEZE macro.

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. Table 2 shows the LEDs ports assignments.

Table 2 • LED to Pins Assignments (IGLOO2 Evaluation Kit Board)

Counter Output	Package Pin
LED_1	F4
LED_2	F3
LED_3	G7
LED_4	H7

Figure 1 shows the top-level block diagram with the main blocks used in the design.

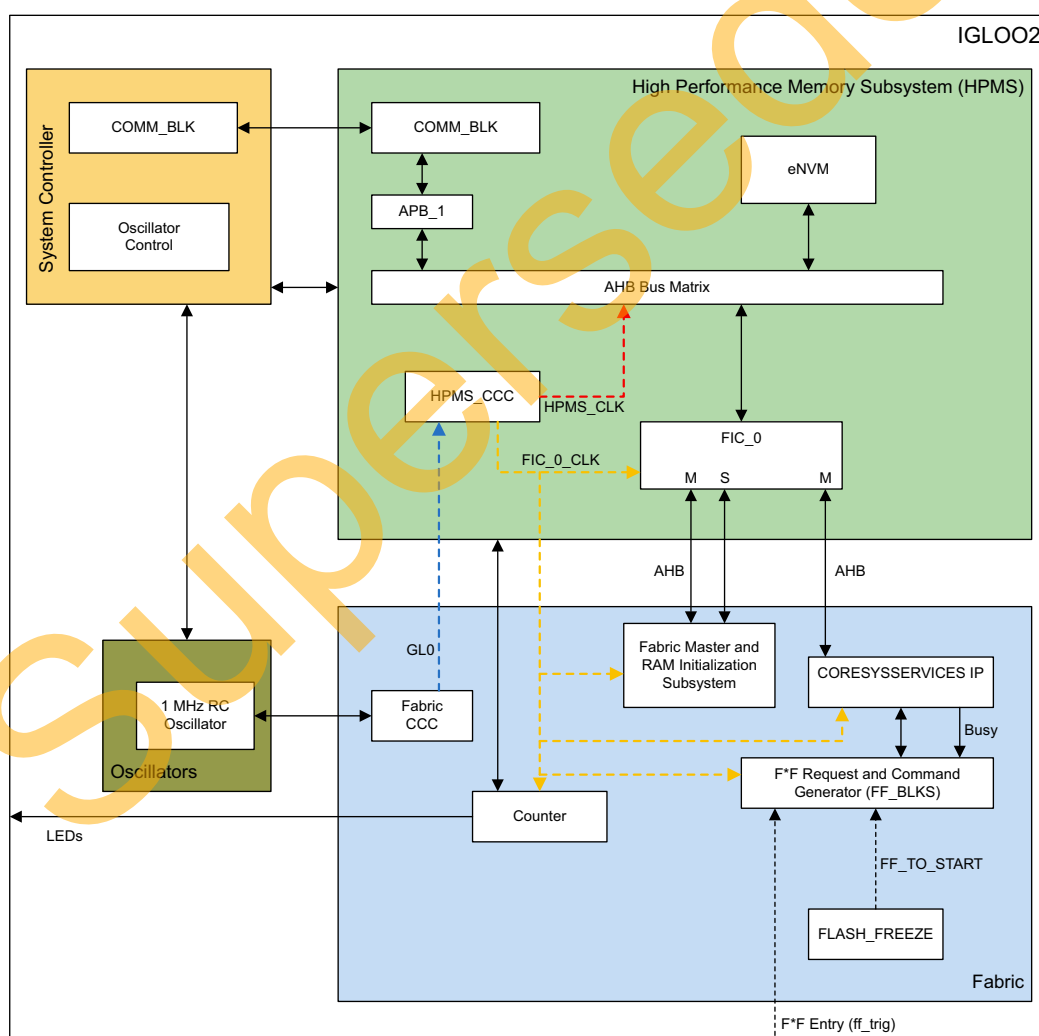


Figure 1 • Top-Level Block Diagram of the Design

Entering into F*F Mode

Entering into F*F mode is done through System Services using CoreSysServices IP core. The F*F request and command service is generated by initiating F*F entry request through the port ff_trig to the FF_BLKs. Upon the trigger of the ff_trig port, the FF_BLKs sends a service enable request along with a service command byte describing the function to be performed. 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 HPMS 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 *IGLOO2 FPGA System Controller User 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 based on the options specified. All these options are available as System Services through CoreSysServices IP core by defining the SERV_OPTION_MODE [2:0] input. This defines the mode options for F*F. Refer to the **CoreSysServices IP Handbook** for more details.

Exiting from F*F Mode

In IGLOO2, exiting from F*F mode can be initiated by external I/Os events. User I/Os (MSIO, MSIOD, or DDRIO) that are single-ended inputs can participate in the F*F exit in the following two ways:

- **I/O Activity:** Force F*F exit upon an activity (Wake_On_Change)
- **I/O Signature:** Force F*F exit upon 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 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 7 on page 9](#).

SW5 (four different dual in-line package (DIP) switches) on the IGLOO2 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 2 on page 5](#).

SW4 on the Evaluation Kit board is used to demonstrate the transition (Wake_On_Change) wake-up event mechanism, as shown in [Figure 2](#).

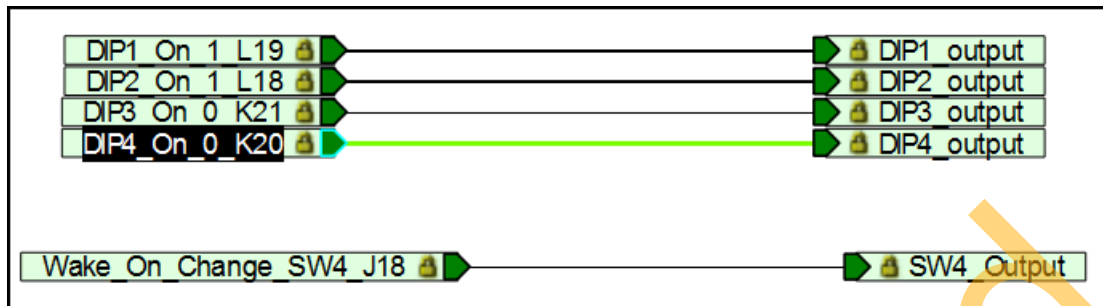


Figure 2 • DIP Switches and the SW4 Connectivity in Top Level Design

Hardware Implementation

The hardware implementation involves configuring the HPMS and the necessary F*F settings. The HPMS configuration is done by using the System Builder. The design example consists of the HPMS, a counter, SRAM wrapper logic, IP cores (CoreSysServices, CoreAHBLite, CoreAHBToAPB3, and CoreAPB3), FLASH_FREEZE macro, fabric AHB master, on-chip 1 MHz RC oscillator, FCCC, and FF_BLKs as shown in [Figure 3 on page 6](#). The IP cores along with the SRAM wrapper are used to initialize the fabric SRAM (AHBMASTER_FIC_RAM) by moving data from the eNVM to the fabric SRAM through FIC_0 AHB master and slave interfaces. 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.

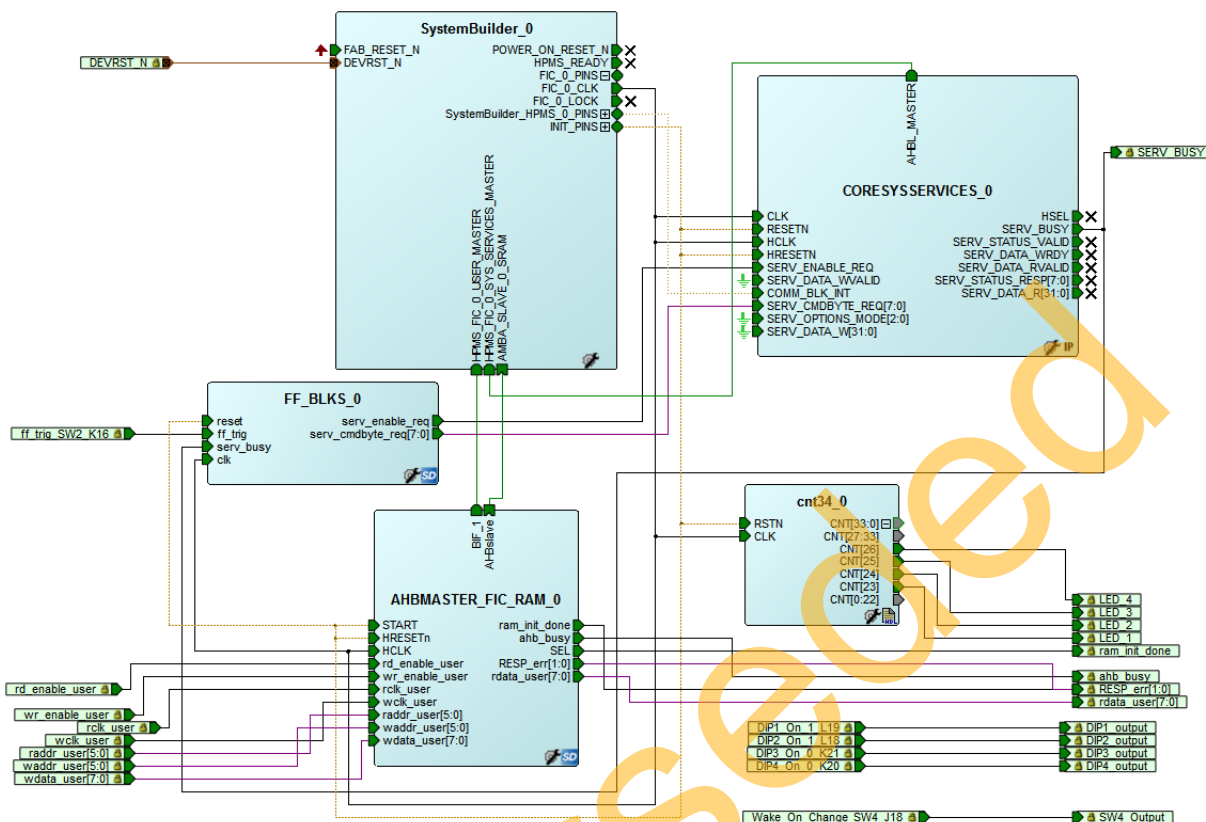


Figure 3 • Top-Level Hardware Design

The HPMS is configured, using the **Device Features** page in the System Builder, to use HPMS System Services and HPMS on-chip Flash Memory (eNVM) as shown in [Figure 4](#). The HPMS is also configured to provide the clock and reset signals to all the blocks including the CoreSysServices IP and FF_BLKs.

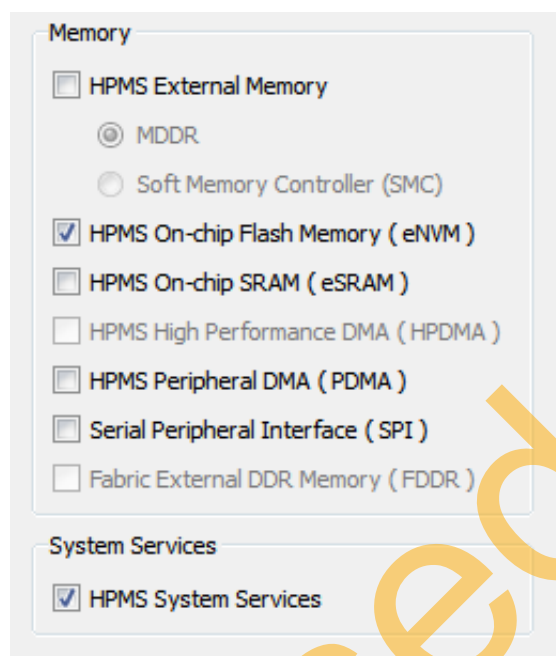


Figure 4 • System Builder Configurations for HPMS System Services and eNVM

The eNVM data storage client is defined using the **Configure Flash Memory** option under the **Memories** page in the **System Builder** configurator. The .mem file used to define the data storage client is located at <project location>\VGL002_FlashFreeze\constraint\ folder.

The HPMS_CCC clock source is sourced from the FPGA Fabric Input through the CLK_BASE port where an FCCC is used. The FCCC is configured to provide the 50 MHz CLK_BASE clock using GL0 output. The reference clock for the FCCC is the on-chip 50 MHz RC oscillator. [Figure 5](#) shows the system clocks configurations for the HPMS_CLK and FIC_0_CLK clock settings. System Builder automatically instantiates FCCC and RCOSC and configures them accordingly.

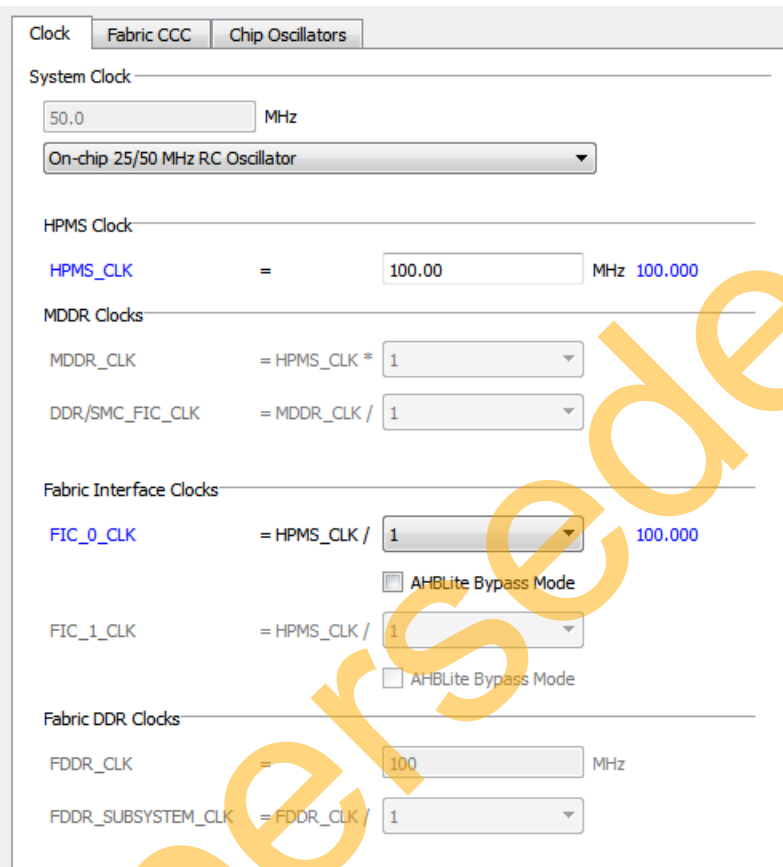


Figure 5 • HPMS System Clocks Configurations

The standby clock source for the HPMS 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 6 on page 9](#). Following are the HPMS 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

Following are the uRAM/LSRAM states options that are available to be used during F*F mode:

- Suspend
- Sleep

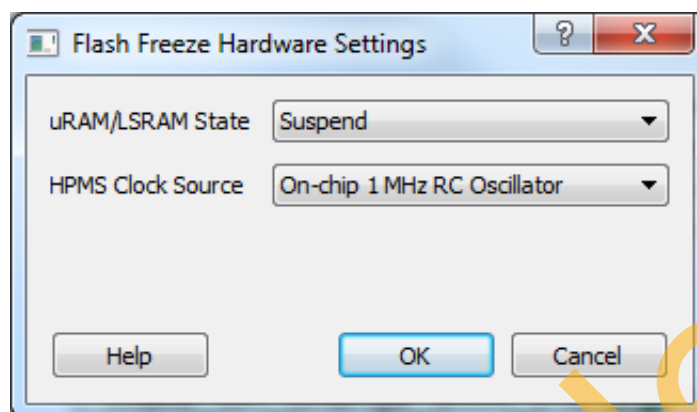


Figure 6 • Flash*Freeze Hardware Settings Dialog

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 7.

Note:

- The I/O available in F*F option applies only to I/Os allocated to the HPMS 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	Resistor Pull	I/O available in Flash*Freeze mode	Low Power Exit
1	Wake_On_Change_SW4_J18	J18	TRISTATE	None	No	Wake_On_Change
2	DIP2_On_1_L18	L18	TRISTATE	Down	No	Wake_On_1
3	DIP1_On_1_L19	L19	TRISTATE	Down	No	Wake_On_1
4	DIP4_On_0_K20	K20	TRISTATE	Up	No	Wake_On_0
5	DIP3_On_0_K21	K21	TRISTATE	Up	No	Wake_On_0

Figure 7 • Specifying I/O State and Functionality Options Using I/O Editor

The F*F exit behavior of input I/Os (DIP1-4) and SW5 are configured using the I/O Editor in the Libero SoC, as shown in Figure 7. The DIP switches-to-package pin assignments for the IGLOO2 Evaluation Kit are shown in Table 3.

Table 3 • DIP Switches to Package Pins Assignments

Input DIP Switch	Package Pin
DIP1	L19
DIP2	L18
DIP3	K21
DIP4	K20
SW4	J18
SW2 (ff_trig)	K16

Running the Design

The design example demonstrates the following options:

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

The design example is designed to run on the IGLOO2 Evaluation Kit board. Refer to www.microsemi.com/index.php?option=com_content&id=2067&lang=en&view=article&tab=documentation for more detailed board information.

Steps to Run the Design

Programming

This step will run FlashPro in batch mode to program the IGLOO2 M2GL010 on the IGLOO2 Evaluation Kit board.

1. Before programming and powering up the IGLOO2 board, confirm that the jumpers are positioned as shown in Table 4.

Table 4 • Board Jumper Settings

Jumper	Setting
J3	1-2 installed
J8	1-2 installed

2. Plug the FlashPro4 ribbon cable into connector J5 (JTAG Programming Header) on the IGLOO2 Evaluation Kit board.
3. Connect the power supply to the J6 connector and FlashPro Programmer.
4. Change the power supply SW7 switch to **ON**.
5. Expand **Program Design** in the **Design Flow** window. Double-click on **Run PROGRAM Action** to begin programming as shown in Figure 8. A green check mark will appear next to the **Program Design** in the **Design Flow** window to indicate programming is completed successfully.

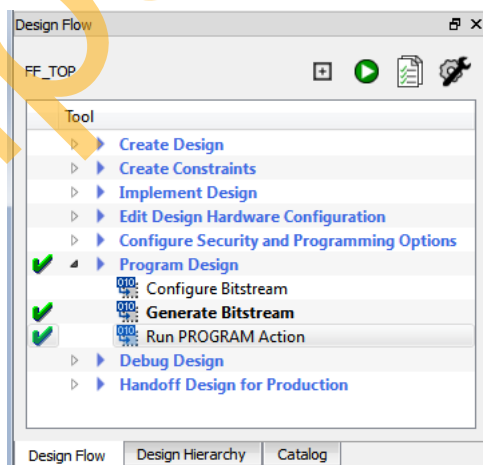


Figure 8 • Program the Design

Note: The IGLOO2 Evaluation Kit board can be programmed using the FlashPro standalone with the provided *.stp file. Refer to "Appendix A - Design Files" on page 16 for more information.

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

1. To enter into F*F mode, press and release the F*F entry (ff_trig) push button (**SW2**). This puts the device state into F*F mode. Observe that the LEDs stop toggling indicating that the fabric entered into the F*F mode.
2. To exit from F*F, press and release the push button switch 4 (**SW4**). SW4 is configured to wake the device from F*F upon an 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 design, SW4 (package pin J18) is used. Observe that the LEDs start to toggle again indicating that the device exited from the F*F mode.

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. If the device is in F*F mode, wake up the device as indicated in the previous step.
2. To enter into F*F mode, press and release the F*F entry push button (**SW2**). This puts the device state into F*F mode. Observe that the LEDs stop toggling, indicating that the fabric entered into the F*F mode.
3. To wake-up the device from F*F mode, toggle DIP switches 1 and 2 to 1 position (OFF) **AND** toggle DIP switches 2 and 3 to 0 position (ON) as shown in Figure 9. Upon this setting, the device exits from F*F mode. Observe that the LEDs start to toggle again indicating that the device exited from the F*F mode.

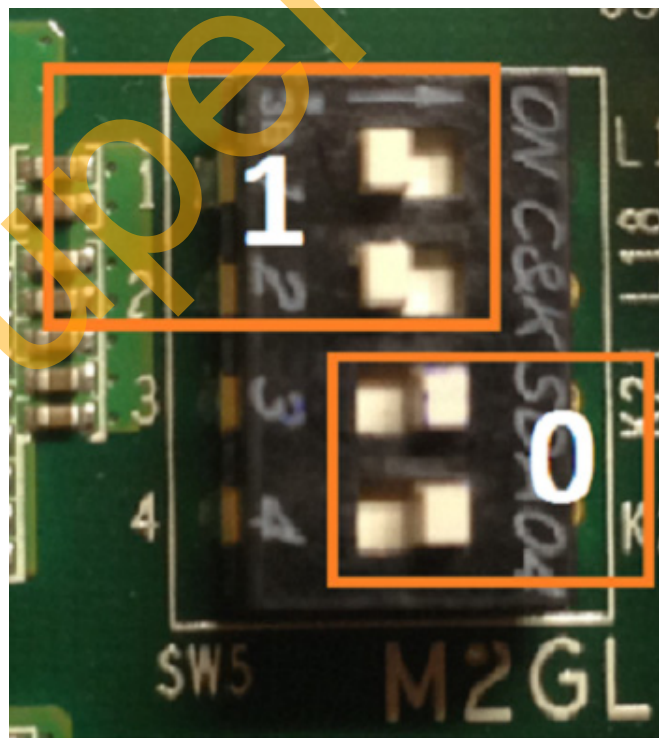


Figure 9 • Toggling DIP Switches

Note: The DIP switches combination shown in [Figure 9 on page 11](#) 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 shown in [Figure 9 on page 11](#).

SRAM Content After Entering and Exiting from 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 is set to be in Suspend mode during F*F. Refer to "[Hardware Implementation](#)" on [page 5](#) for more information. To check the content of the SRAM after entering and exiting from F*F, SmartDebug is used to read back the content of the SRAM from the device after exiting from the F*F mode.

1. Check the content of the SRAM before entering into the F*F mode. While the device is in Active Mode (non F*F), double-click the SmartDebug Design entry from the Design Flow window as shown in [Figure 10](#).

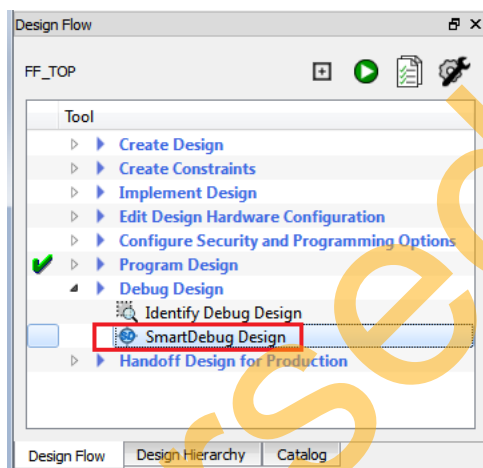


Figure 10 • Launching SmartDebug Design Tools

The SmartDebug window opens.

2. Click on Debug FPGA Array as shown in Figure 11.

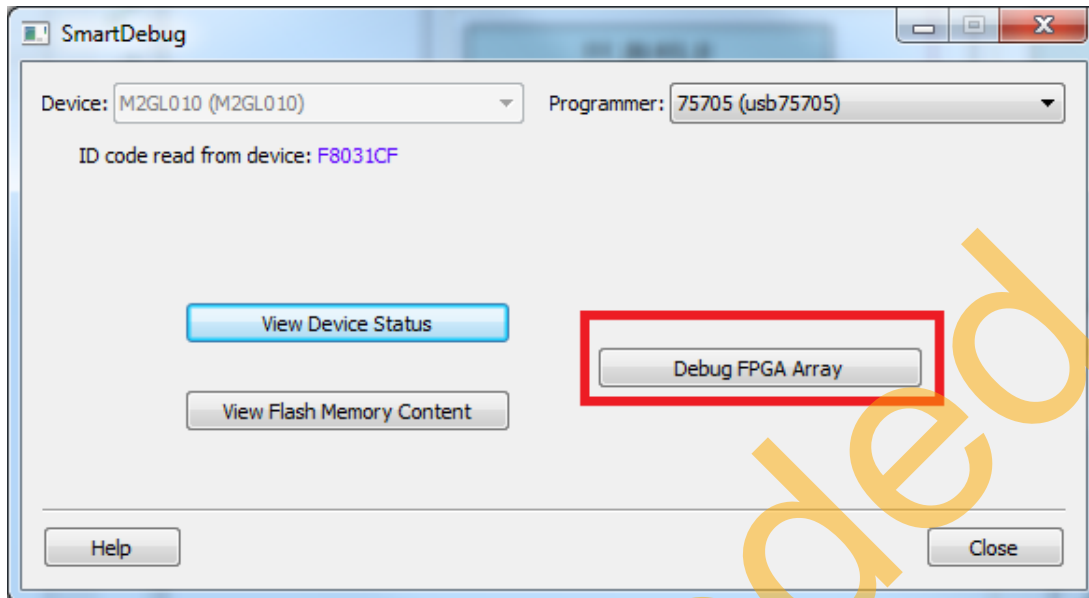


Figure 11 • SmartDebug Window - Debug FPGA Array

- a. Point to the debug file. The debug file is automatically generated into the Libero SoC project. Click **Browse** and navigate to **<Libero SoC project path>/designer/<top level design name>/<design_name>_debug.txt** as shown in Figure 12 and click **Open**.

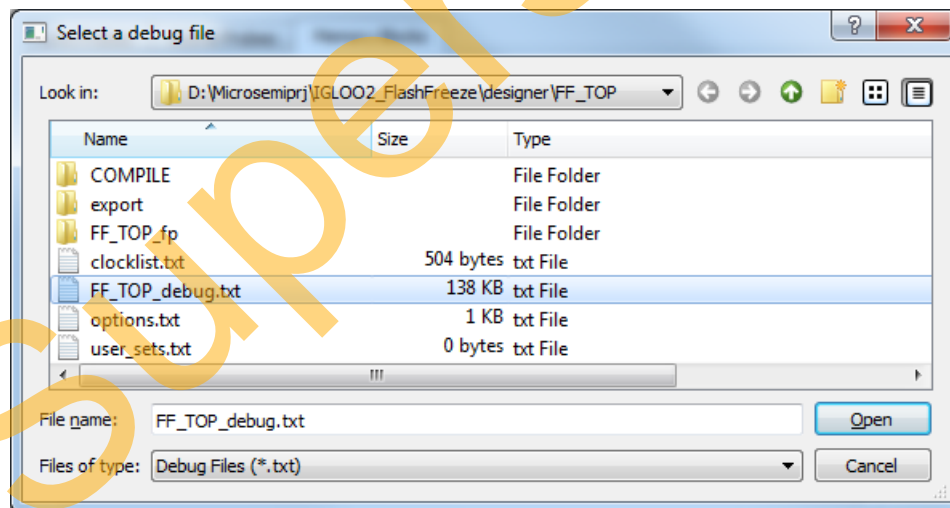


Figure 12 • Selecting the Debug File

- b. Select the **Memory Block** tab in the **Debug FPGA Array** window and select **Read Block** as shown in Figure 13. The SmartDebug tool reads the SRAM content from the device and shows it in the **Memory Block Data** section as shown in Figure 13.

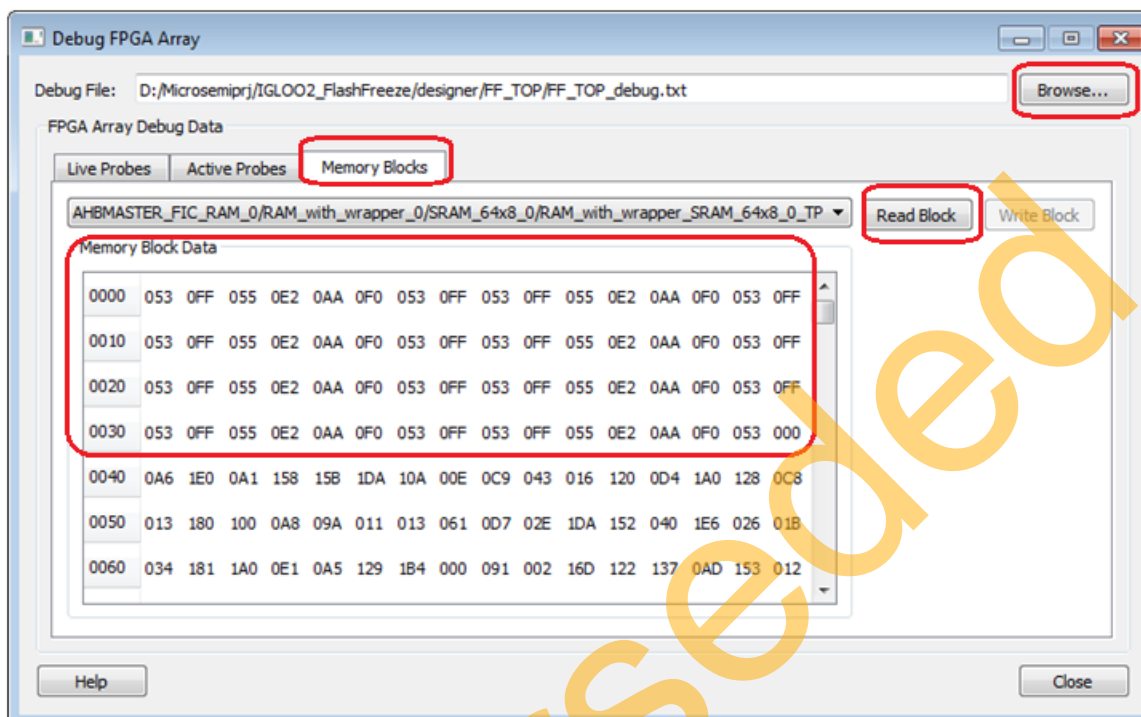


Figure 13 • SRAM Read-back Content before F*F entry

In the previous steps, the data shown is the content of the SRAM while the device is in Active mode. The next steps demonstrate putting the device into F*F, exiting from it, and finally checking the content of the SRAM after exiting from the F*F mode.

3. Enter into F*F mode. Press and release the F*F entry push button (**SW2**). This puts the device state into F*F mode. Observe that the LEDs stop toggling, indicating that the fabric entered into the F*F mode.
4. Exit from F*F. Press and release **SW4**.

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 through SmartDebug, the SRAM content after F*F exit is the same data that is stored into the SRAM before entering into F*F mode, as shown in [Figure 14](#).

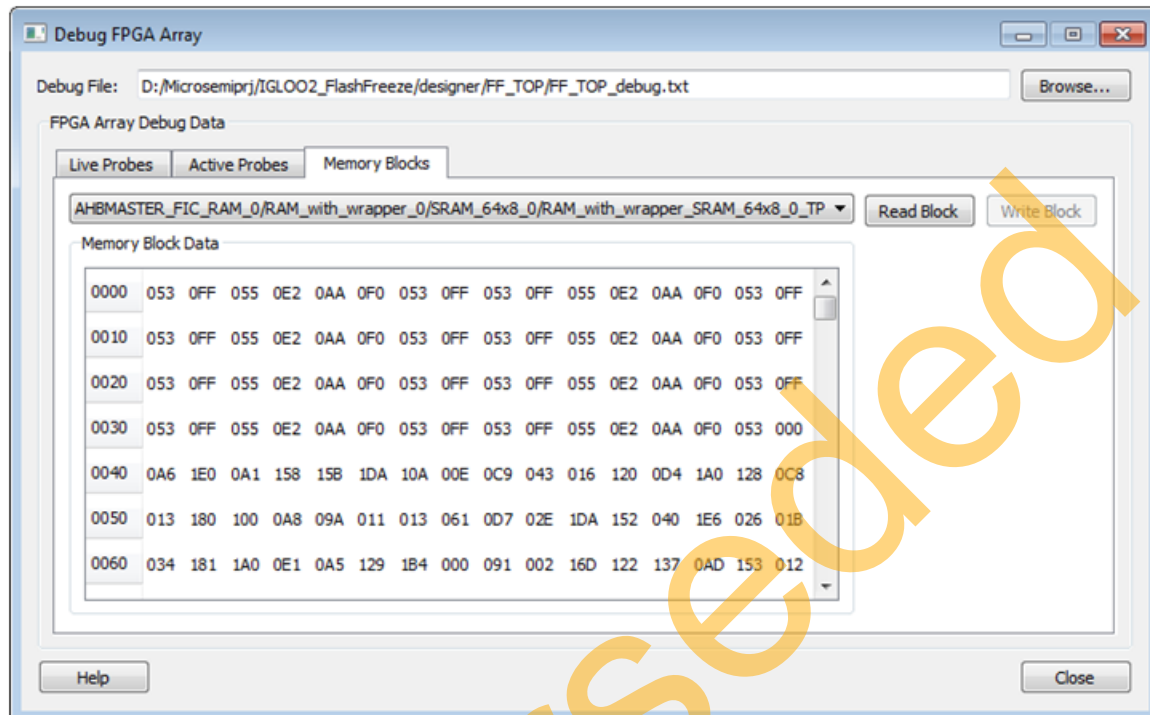


Figure 14 • Reading SRAM Content After F*F Exit

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

Conclusion

This application note describes how to put the IGLOO2 device into F*F mode using System Services and demonstrates the different options that can be used to wake-up the IGLOO2 device from F*F mode. In addition, it 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:

http://soc.microsemi.com/download/rsc/?f=M2GL_FlashFreeze_11p4_DF

The design file has Libero SoC Verilog project, the .mem file for the eNVM data storage client, and programming files (*.stp) for IGLOO2 Evaluation Kit board. Refer to the Readme.txt file included in the design file for the directory structure and description.

Superseded

List of Changes

The following table lists critical changes that were made in each revision of the chapter in the demo guide.

Date	Changes	Page
Revision 2 (August 2014)	Updated the document for Libero v11.4 software release (SAR 59065).	NA
Revision 1 (January 2014)	Initial release.	NA

Superseded

Superseded



Microsemi[®]

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

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and security, aerospace, 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; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at www.microsemi.com.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.