

IGLOO2 Flash*Freeze Entry and Exit

- Libero SoC v11.6

Table of Contents

Purpose	1
Introduction	1
References	2
Design Requirements	2
Design Description	2
Entering into F*F Mode	5
Exiting from F*F Mode	6
Hardware Implementation	8
Running the Design	12
Steps to Run the Design	12
Conclusion	17
Appendix: Design Files	18
List of Changes	19

Purpose

This application note describes the methods and steps of how to enter and exit the Flash*Freeze (F*F) mode. It shows how to set different user-defined settings that define the behavior of static random-access memory (SRAM) blocks during the F*F entry and exit modes using the Libero® System-on-Chip (SoC) software. It also describes how to use System Services provided by the CoreSysServices soft IP to enter into the 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 the F*F mode retains all the SRAM and registers information and the F*F exit mode achieves rapid recovery to the 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 [UG0450: IGLOO2 FPGA System Controller User Guide](#) for more details. The IGLOO2 device enters into the F*F mode by using the F*F services request that the System Controller provides. Some of the F*F hardware settings options can be set 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 the F*F mode.

The HPMS standby 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 the **Suspend** mode, the large SRAM (LSRAM) and micro SRAM (μSRAM) contents are retained. That is, when the device exits the F*F mode, the contents of the SRAMs are retained. In the **Sleep** mode, the SRAMs contents are not retained. Exiting from F*F is achieved by 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 the F*F mode, I/O availability in the F*F mode, and I/O role in exiting from the F*F mode.

Depending on the type of the I/O, some or all of these options may not be available. Refer to the [UG0444: 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 how to enter the 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 interface 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 the Libero SoC software. Managing the MDDR, FDDR, or SERDES before and after the 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.

- [UG0450: IGLOO2 FPGA System Controller User Guide](#)
- [UG0444: 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	v11.6
FlashPro programming software	v11.6
Host PC Drivers	USB to UART drivers
CoreSysServices	v3.1.101

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), F*F request, command generator logic (FF_BLKs), and a synchronizer counter (CLK_Sync_CNTR_Dly) to synchronize the clocks between the fabric and the HPMS system clock after exiting from Flash*Freeze. The fabric AHB master along with the SRAM wrapper (AHBMASTER_FIC_RAM) is 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 the F*F mode.

In the 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 FF_TO_START signal is asserted by the System Controller to indicate that the Flash*Freeze service is about to start. Only 10 μ s are available to do housekeeping before the core is powered off. Microsemi recommends the user to use this signal as part of the clock gating process to ensure that any glitches do not cause a sequential element in the design to transition to an unwanted state when entering Flash*Freeze. For more information about the FLASH_FREEZE macro, refer to the [UG0450: IGLOO2 FPGA System Controller User Guide](#).

When the system enters into Flash*Freeze, the main clock is switched to a standby clock that is defined by the user, where the user sets the Flash*Freeze hardware settings in the Libero design flow as shown in [Figure 7](#).

When the system controller comes out of the Flash*Freeze mode, MSS_CCC still runs off the standby clock. The system controller then waits for the MCCC_MPLL_LOCK assertion and then switches the clock to user system clock. After the lock assertion and before the MSS_CCC clock is switched to user clock, the system controller is ready to communicate with the fabric. The following are the steps that happen during the Flash*Freeze exit process:

1. Switch from the standby clock to the system clock (user clock) and wait for the MPLL lock
2. Wait for the clocks required by the HPMS sub-blocks and the FPGA fabric interface clocks to be aligned

Within the MSS or HPMS CCC, the fabric alignment clock controller (FACC) interfaces with the MPLL, generating the aligned clocks required by the MSS or HPMS sub-blocks, and controls the alignment of the FPGA fabric interface clocks. MCCC_GLMUX_SEL is the register that contains the select line for the four non-glitch multiplexers within FACC, which are related to the aligned clocks. All the four multiplexers are switched by one signal as follows:

- 1: M3_CLK, APB_0_CLK, APB_1_CLK, DDR_SMC_FIC_CLK all driven from CLK_STANDBY
- 0: M3_CLK, APB_0_CLK, APB_1_CLK, DDR_SMC_FIC_CLK all driven from stage B dividers

For more information on the description of the FACC, refer to the [UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide](#).

The sync-up counter logic (CLK_Sync_CNTR_Dly) achieves the following:

- Waits for MCCC_MPLL_LOCK to assert
- Waits for MCCC_GLMUX_SEL to switch to the user clock
- Accounts for the time required for the HPMS clock to switch from the standby clock to the operating clock after PLL achieves the lock and the system controller is ready to communicate with the fabric

When MCCC_MPLL_LOCK achieves lock, MCCC_GLMUX_SEL selects the user clock instead of the standby clock and the required time passes. GL0_EN is asserted to enable the GL0 clock that clocks the fabric logic.

To expose the MCCC_MPLL_LOCK and MCCC_GLMUX_SEL signals to the fabric, if you are using System Builder, convert the System Builder block into SmartDesign block. In the HPMS block, enable the options in the **Advanced Options** tab, as shown in [Figure 1](#).

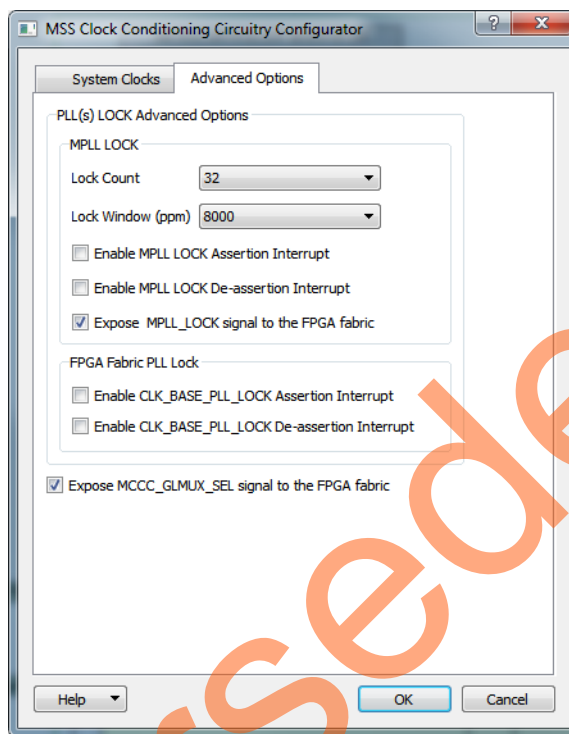


Figure 1 • MSS Clock CCC - Advanced Options

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 the F*F mode. [Table 2](#) shows the LED to pin assignment.

Table 2 • LED to Pin Assignment (IGLOO2 Evaluation Kit Board)

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

Figure 2 shows the top-level block diagram with the main blocks used in the design.

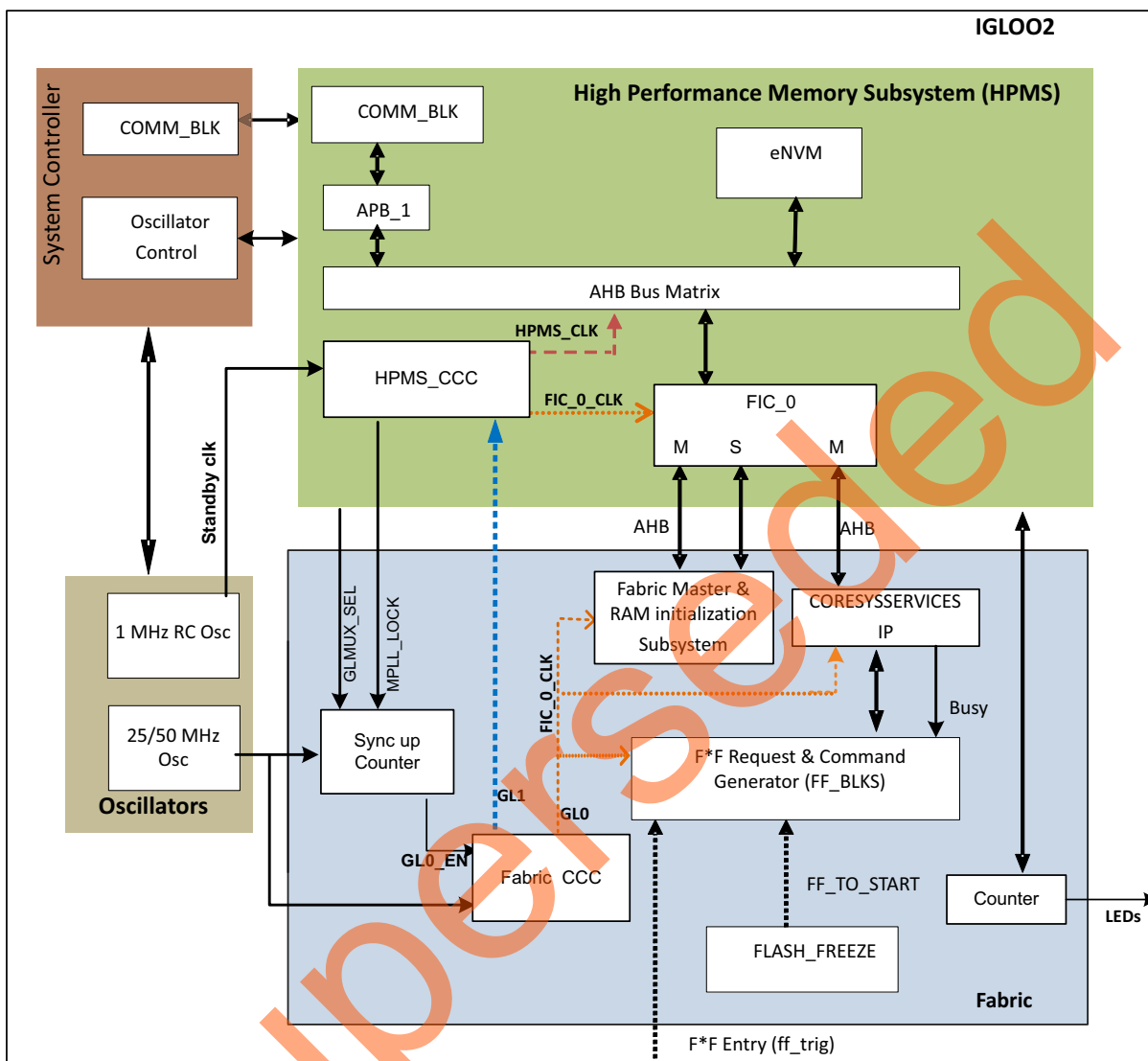


Figure 2 • Top-Level Block Diagram of the Design

Entering into F*F Mode

Entering into the F*F mode is done through the System Services using CoreSysServices IP core. The F*F request and command service is generated by initiating the F*F entry request through the port `ff_trig` to the FF_BKLS. Upon the trigger of the `ff_trig` port, the FF_BKLS 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 is accepted by the COMM_BLK FIFO. If a new service request is received while servicing another request, the new service request is immediately aborted. For more information, refer to the "Flash*Freeze Service" section in the [UG0450: IGLOO2 FPGA System Controller User Guide](#).

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. For more information, refer to the **CoreSysServices IP Handbook**.

Exiting from F*F Mode

In IGLOO2, exiting from the 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 **AND**ed) **AND**ed with (All Wake-on-1 **AND**ed)} **OR**ed with (All Wake-on-Change **OR**ed).

I/O Activity

In the 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 the Low-power mode. When a change happens on the configured I/O, the device wakes up from the 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 the 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 the 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 the 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 11.

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 3 on page 7.

SW4 on the Evaluation Kit board is used to demonstrate the transition (Wake_On_Change) wake-up event mechanism, as shown in Figure 3.

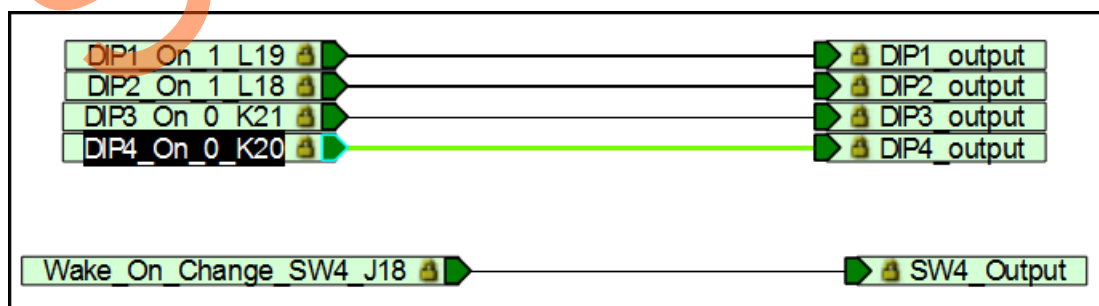


Figure 3 • DIP Switches and the SW4 Connectivity in Top-Level Design

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 5](#). The HPMS is also configured to provide the clock and reset signals to all the blocks including the CoreSysServices IP and FF_BLKs.

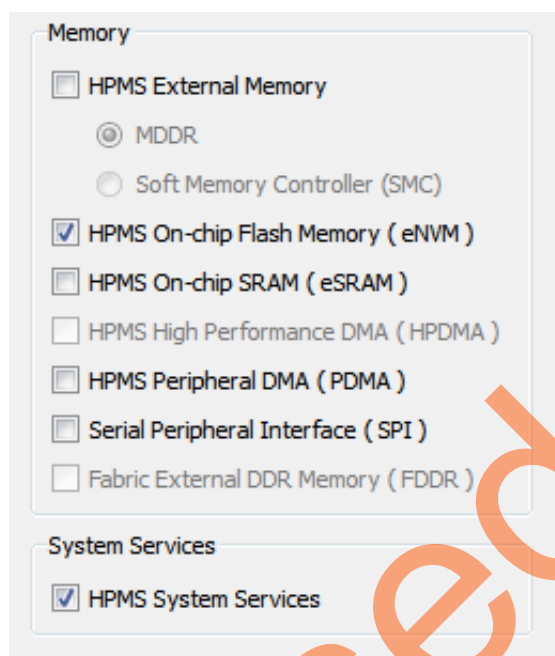


Figure 5 • System Builder Configurations for HPMS System Services and eNVM

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>\IGLOO2_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 6 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.

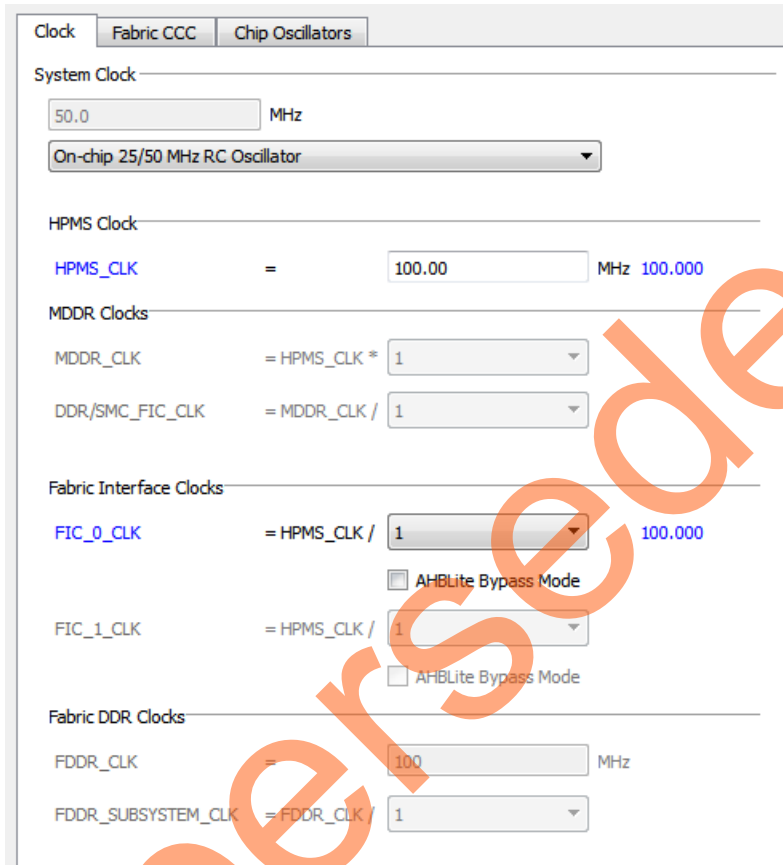


Figure 6 • HPMS System Clocks Configurations

The standby clock source for the HPMS in the F*F mode and the state of the SRAMs (μ RAM and LSRAM) during the F*F mode are configured using the **Flash*Freeze Hardware Settings** dialog in the Libero SoC software, as shown in Figure 7 on page 11. The following are the HPMS clock source options that are available to be used during the F*F mode:

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

The following are the μ RAM/LSRAM state options that are available to be used during the F*F mode:

- Suspend
- Sleep

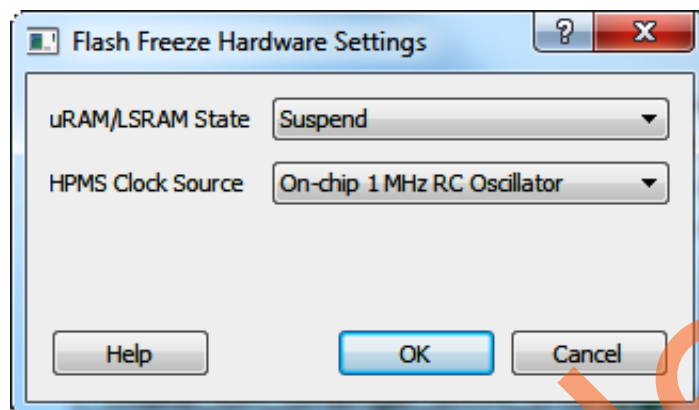


Figure 7 • Flash*Freeze Hardware Settings Dialog

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

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 the 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 ▼
Wake_On_Change_SW4_	J18	TRISTATE	None	No	Wake_On_Change
DIP1_On_1_L19	L19	TRISTATE	Down	No	Wake_On_1
DIP2_On_1_L18	L18	TRISTATE	Down	No	Wake_On_1
DIP3_On_0_K21	K21	TRISTATE	Up	No	Wake_On_0
DIP4_On_0_K20	K20	TRISTATE	Up	No	Wake_On_0
ff_trig_SW2_K16	K16	TRISTATE	None	No	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 SW5 are configured using the I/O Editor in the Libero SoC, as shown in Figure 8. The DIP switch-to-package pin assignment for the IGLOO2 Evaluation Kit is shown in Table 3.

Table 3 • DIP Switch to Package Pin Assignment

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

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 runs FlashPro in the 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. Open the IGLOO2_FlashFreeze Libero project (refer to "Appendix: Design Files" section on page 18).
6. Update the eNVM client memory file path. For more information about how to correct the errors detected during eNVM programming data generation, refer to <http://soc.microsemi.com/kb/article.aspx?id=SL5657>.

- Expand **Program Design** in the **Design Flow** window. Double-click on **Run PROGRAM Action** to begin programming as shown in Figure 9. A green check mark appears next to the **Program Design** in the **Design Flow** window to indicate programming is completed successfully.

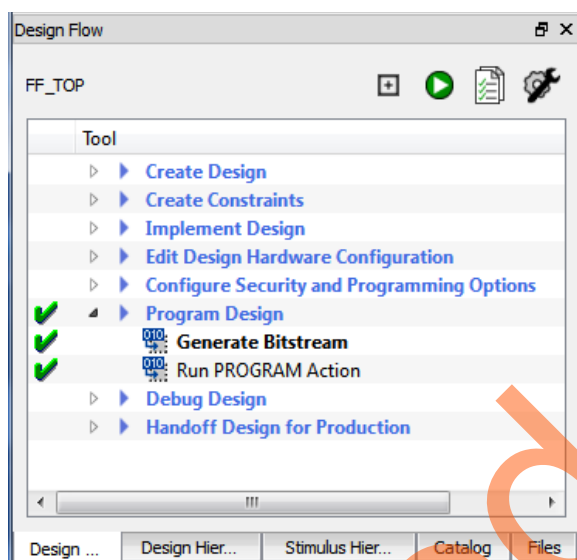


Figure 9 • Program the Design

Note: The IGLOO2 Evaluation Kit board can be programmed using the FlashPro standalone with the provided *.stp file. For more information, refer to "Appendix: Design Files" on page 18.

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

- To enter into the F*F mode, press and release the F*F entry (ff_trig) push button (**SW2**). This puts the device state into the F*F mode. Observe that the LEDs stop toggling indicating that the fabric entered into the F*F mode.
- 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 the F*F mode.

- If the device is in the F*F mode, wake up the device as indicated in the previous step.
- To enter into the F*F mode, press and release the F*F entry push button (**SW2**). This puts the device state into the 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 the 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 10](#). Upon this setting, the device exits from the F*F mode. Observe that the LEDs start to toggle again indicating that the device exited from the F*F mode.

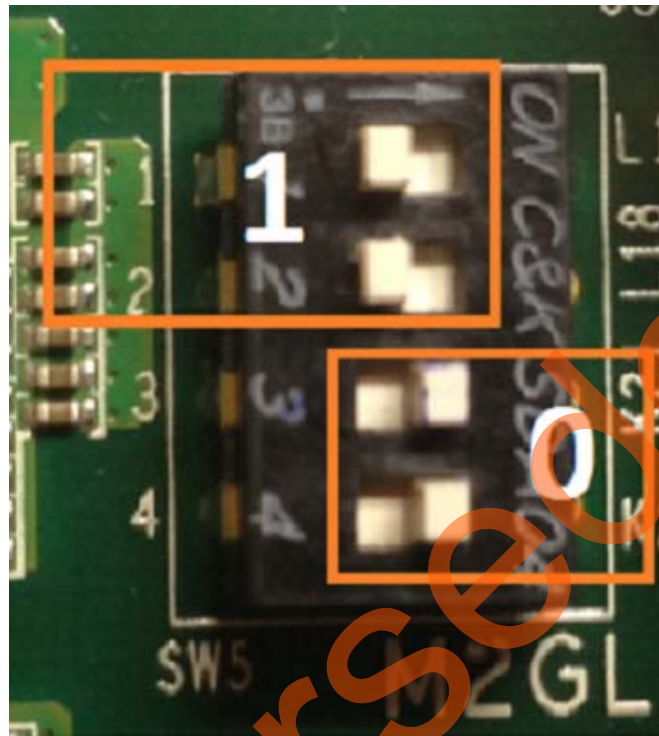


Figure 10 • Toggling DIP Switches

Note: The DIP switches combination shown in [Figure 10](#) constantly keeps the device in the 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 10](#) on page 14.

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 the F*F mode. The SRAM is set to be in the Suspend mode during F*F. For more information, refer to "[Hardware Implementation](#)" on page 8. 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 the Active mode (non F*F), double-click **SmartDebug Design** entry from the **Design Flow** window as shown in Figure 11.

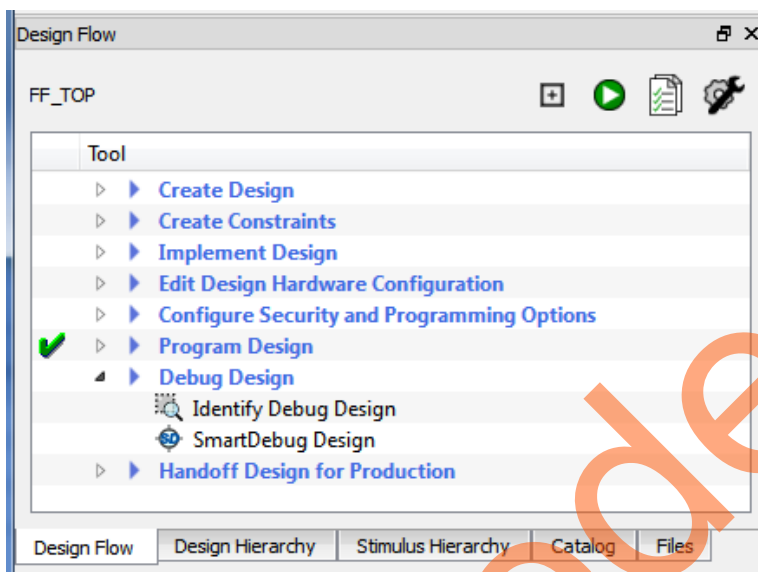


Figure 11 • Launching SmartDebug Design Tools

The SmartDebug window opens.

2. Click **Debug FPGA Array** as shown in Figure 12.

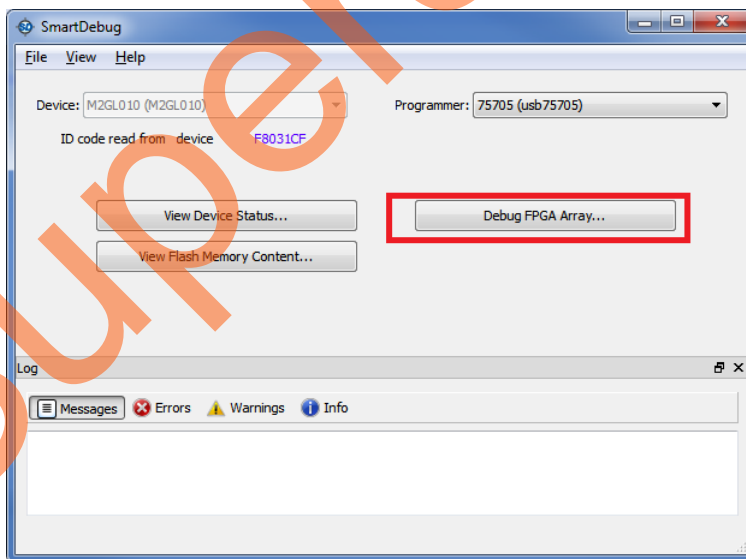


Figure 12 • SmartDebug Window - Debug FPGA Array

The debug file is automatically generated into the Libero SoC project (<Libero SoC project path>/designer/<top level design name>/<design_name>_debug.txt). The debug file is automatically loaded into the **SmartDebug** window.

3. Select the **Memory Blocks** 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.

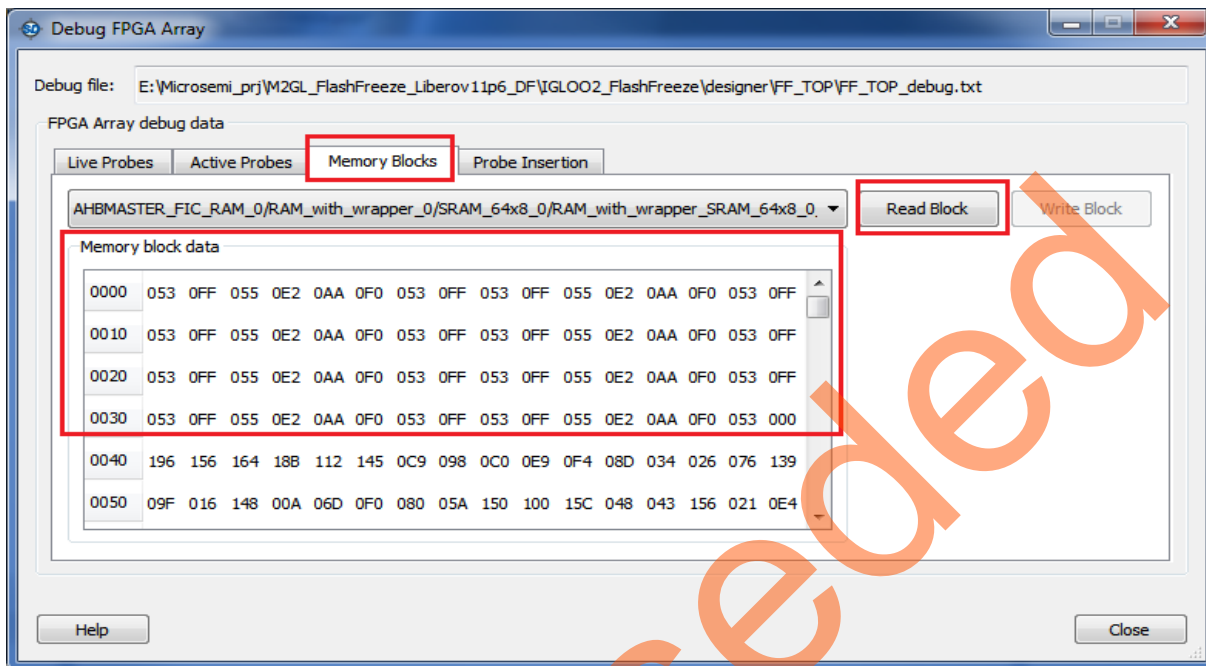


Figure 13 • SRAM Read-back Content before F*F entry

- In the previous steps, the data shown is the content of the SRAM while the device is in the Active mode. The next steps demonstrate putting the device into F*F, exiting from it, and checking the content of the SRAM after exiting from the F*F mode.
4. Enter into the F*F mode. Press and release the F*F entry push button (**SW2**). This puts the device state into the F*F mode. Observe that the LEDs stop toggling, indicating that the fabric entered into the F*F mode.
 5. Exit from F*F. Press and release **SW4**.

In this design, the SRAM is set for the Suspend mode during the F*F mode, therefore, the content of the SRAM is retained. Therefore, when reading through SmartDebug, the SRAM content after F*F exit is the same data that is stored into the SRAM before entering into the F*F mode, as shown in Figure 14.

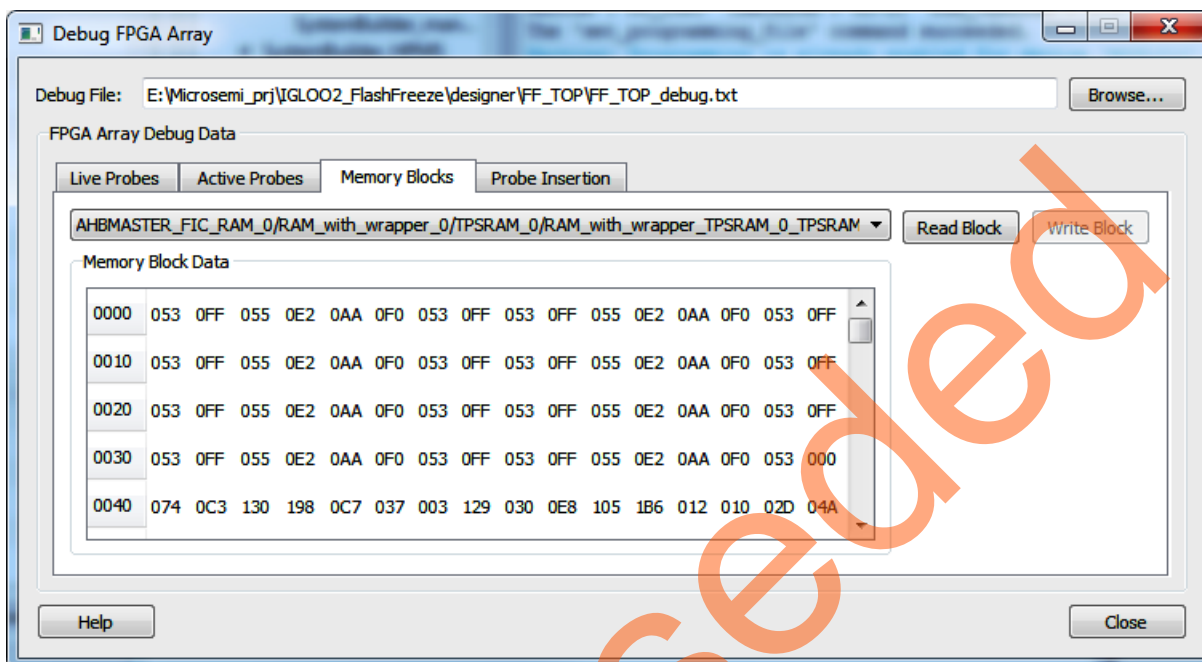


Figure 14 • Reading SRAM Content After F*F Exit

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

Conclusion

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

Appendix: Design Files

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

http://soc.microsemi.com/download/rsc/?f=m2gl_ac412_liberov11p6_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 shows the important changes made in this document for each revision.

Date	Changes	Page
Revision 4 (December 2015)	Updated the document for Libero v11.6 software release (SAR 68372).	NA
Revision 3 (January 2015)	Updated the document for Libero v11.5 software release (SAR 62939).	NA
Revision 2 (August 2014)	Updated the document for Libero v11.4 software release (SAR 59065).	NA
Revision 1 (January 2014)	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.		

Superseded



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

© 2015 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.

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & 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; Ethernet Solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at www.microsemi.com.

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