

UG0444
User Guide
SmartFusion2 SoC and IGLOO2 FPGA Low-Power
Design



a  **MICROCHIP** company



a  MICROCHIP company

Microsemi 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

Email: sales.support@microsemi.com

www.microsemi.com

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

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

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

1	Revision History	1
1.1	Revision 6.0	1
1.2	Revision 5.0	1
1.3	Revision 4.0	1
1.4	Revision 3.0	1
1.5	Revision 2.0	1
1.6	Revision 1.0	1
2	Flash*Freeze	2
2.1	Features	2
2.2	Functional Description	2
2.2.1	Flash*Freeze Entry Phase	2
2.2.2	During Flash*Freeze Phase	5
2.2.3	Flash*Freeze Exit Phase	7
2.3	Using Flash*Freeze in SmartFusion2	10
2.3.1	Design Flow	11
2.3.2	Flash*Freeze Use Model	13
2.4	Using Flash*Freeze in IGLOO2	14
2.4.1	Enabling Flash*Freeze	15
2.4.2	HPMS Subsystem	16
2.4.3	Entering and Exiting Flash*Freeze Mode	17
2.5	Flash*Freeze Design	19
2.5.1	Clocks	19
2.5.2	I/Os	21
2.5.3	FLASH_FREEZE Macro	21
2.5.4	Flash*Freeze Guidelines	21
2.6	SYSREG Control Registers for Flash*Freeze	26
2.7	Acronyms	27
2.8	Terminology	27

Figures

Figure 1	Flash*Freeze Entry	4
Figure 2	Flash*Freeze Modes	5
Figure 3	Bus Keeper Configuration in I/O Editor	6
Figure 4	Resistor Pull Configuration	6
Figure 5	I/O Activity Configuration	7
Figure 6	I/O Signature Configuration	7
Figure 7	Fabric Master initiated Flash*Freeze Exit	9
Figure 8	MSS_CCC and RTC Options for Flash*Freeze	11
Figure 9	RTC Configurator	12
Figure 10	Flash*Freeze Example Projects	12
Figure 11	Opening SmartFusion2 MSS System Services Driver User Guide	13
Figure 12	System Builder - Device Features Window	14
Figure 13	System Builder - Device Features Tab	15
Figure 14	System Builder - Memory Map Tab	16
Figure 15	HPMS Subsystem	16
Figure 16	CoreSysServices Configuration	17
Figure 17	HPMS Subsystem Connections with the CoreSysServices AHB Bus Master	18
Figure 18	Configure Flash*Freeze in Design Flow Tab	18
Figure 19	Flash*Freeze Hardware Settings	18
Figure 20	Wake_on_Change Configuration	19
Figure 21	Global Gated Clock Macros	19
Figure 22	Gated External Clocks Using Single GCLKBUF Macro	20
Figure 23	Gated External Clock Using Multiple GCLKBUF Macros	20
Figure 24	FLASH_FREEZE Macro	21

Tables

Table 1	Flash*Freeze Service Message Request Command	3
Table 2	System Service API for Flash*Freeze	13
Table 3	Gating AMBA-related Signals from Fabric to MSSDDR	22
Table 4	Gating AMBA-related Signals from Fabric to FDDR	22
Table 5	Gating of AMBA-related Signals from Fabric to SERDESIF	23
Table 6	Gating MSS Interrupt Signal	23
Table 7	Gating MSS Non-Interrupt Signals	23
Table 8	Gating Fabric PLL Lock Signals	23
Table 9	Gating MSS Peripheral Signals	24
Table 10	Gating System Controller Response Signals	26
Table 11	SYSREG Control Register	26

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 6.0

A note about RTC timeout was added. For more information, see [RTC Timeout](#), page 8.

1.2 Revision 5.0

A note about Cortex-M3 processor sleep modes was added to the introduction. For more information, see [Flash*Freeze](#), page 2.

1.3 Revision 4.0

The following is a summary of the changes in revision 4.0 of this document.

- The document was updated for Libero v11.7 SP1 software updates.
- Information about flash freeze guidelines was added. For more information, see [Flash*Freeze Guidelines](#), page 21.

1.4 Revision 3.0

The following is a summary of the changes in revision 3.0 of this document.

- Merged SmartFusion2 and IGLOO2 user guides.
- Updated SARs: 54450, 55852, 55855, 55856, 56532, 57027, 57140, 59868, 64019, 65635, 68477, and 69512.

1.5 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Updated [I/O Activity](#), page 7 (SAR 50532).
- Added [Using Flash*Freeze in IGLOO2](#), page 14 (SAR 50362).
- Updated [Flash*Freeze Entry Phase](#), page 2 and [Flash*Freeze Exit Phase](#), page 7 (SAR 50581, 50582, 50585).
- Updated [Flash*Freeze Entry](#), page 3 and Fabric Master Initiated Flash*Freeze Exit (SAR 50583, 50584).
- Updated Flash*Freeze Entry Phase section (SAR 54986)
- Updated Flash*Freeze Entry Time section (SAR 54986)
- Updated Fabric State During Flash*Freeze Mode section (SAR 54986)
- Updated Enabling Flash*Freeze Mode section (SAR 54986)

1.6 Revision 1.0

Restructured the user guide (SAR 41389, 44796).

2 Flash*Freeze

SmartFusion[®]2 SoC FPGAs and IGLOO[®]2 FPGAs offer the Flash*Freeze technology for implementing low-power solutions. Flash*Freeze mode is an ultra-low power static mode with lowest standby power of 1.92 mW. Flash*Freeze technology allows easy entry and exit from ultra-low power static mode while retaining SRAM content, I/O state, and register data, thereby dramatically reducing power. Flash*Freeze mode can be used in a wide variety of applications, such as patient monitoring systems, industrial applications, automotive applications, mobile applications, power distribution systems, and applications that require the lowest possible static power to function.

Note: SmartFusion2 devices also support Cortex-M3 processor sleep modes, which reduce power consumption through clock gating. For more information about the sleep modes, see the Power Management section in *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

2.1 Features

Flash*Freeze has the following features:

- User configurable Flash*Freeze entry mechanism through:
 - Cortex-M3 processor firmware
 - AHB-Lite/APB fabric master (if Cortex-M3 processor is not enabled)
- User configurable Flash*Freeze exit mechanism through:
 - I/O activity
 - I/O signature match
 - RTC wakeup interrupt
- External clock sources are not needed
- MSS or HPMS stays powered-up to retain the SRAM data
- The state information for fabric registers, fabric SRAM content, and I/O is retained, enabling fast recovery to active mode

Notes:

- Ethernet MAC is not supported during Flash*Freeze mode.
- No new tamper events are registered during Flash*Freeze mode. However, any active events before Flash*Freeze entry are still active after the Flash*Freeze exit.

2.2 Functional Description

Flash*Freeze implementation consists of three phases:

- Flash*Freeze entry phase
- During Flash*Freeze phase
- Flash*Freeze exit phase

2.2.1 Flash*Freeze Entry Phase

SmartFusion2 and IGLOO2 FPGA devices are designed and optimized to enter Flash*Freeze mode only when the power supply is stable.

Flash*Freeze entry can be initiated by ARM Cortex-M3 processor firmware or AHB-Lite/APB Fabric master (if ARM Cortex-M3 processor is not enabled). On initiation, the system master (the ARM Cortex-M3 processor or the fabric master) must check for the high-speed peripherals (FDDR, MDDR, and SerDes) to power them down if required and also halt the existing user defined fabric logic. The system master has to send a Flash*Freeze service request command as shown in [Table 1](#), page 3 to enter into Flash*Freeze. This service request command is sent to the system controller over the COMM_BLK interface. When Flash*Freeze service request commands are sent from the system master to the system controller, the device is put into Flash*Freeze.

For more information about Flash*Freeze command, see the Flash*Freeze Services section of [UG0450: SmartFusion 2 and IGLOO2 System Controller User Guide](#).

Table 1 • Flash*Freeze Service Message Request Command

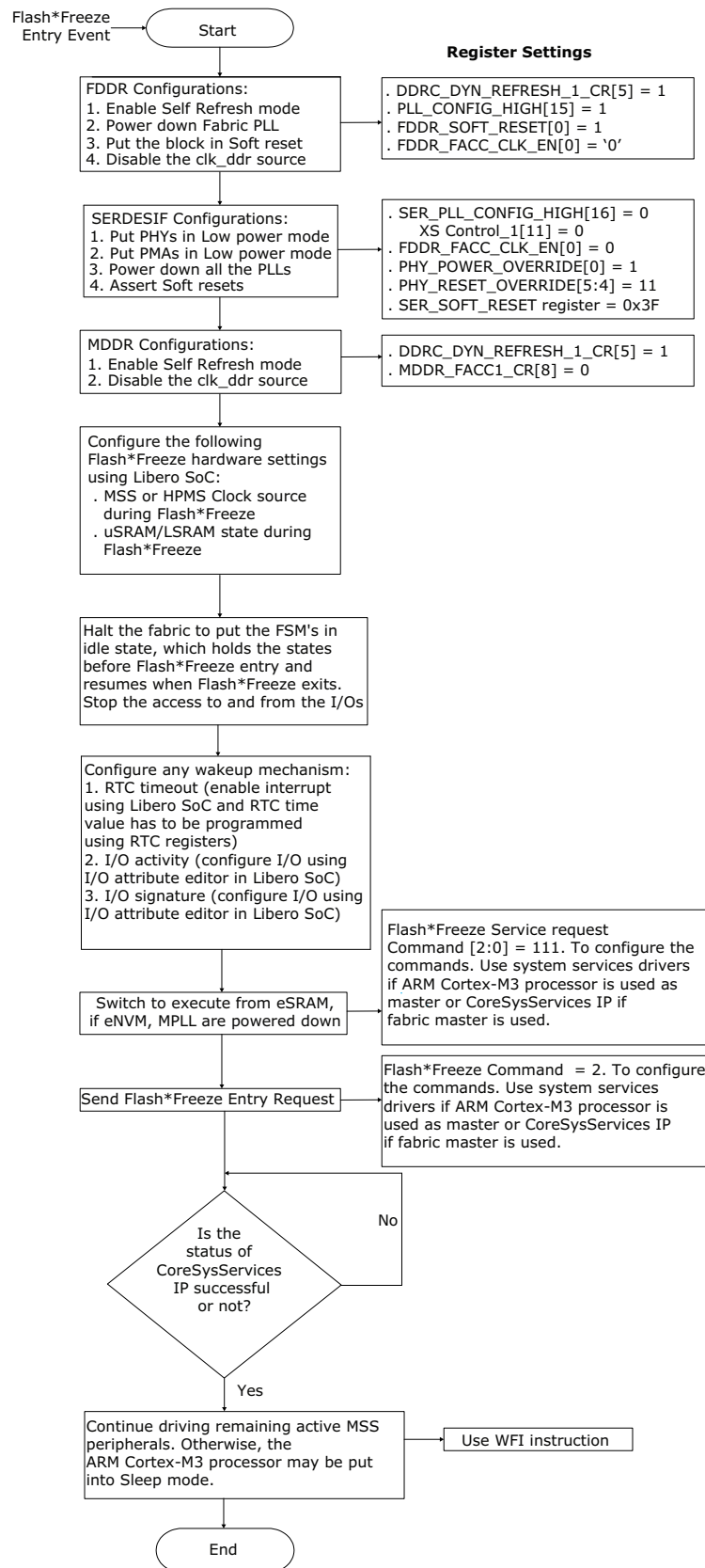
Bit Number	Command	Description
[7:0]	0x02	Flash*Freeze service request message command

2.2.1.1 Flash*Freeze Entry

The user fabric logic must follow the sequence of steps for Flash*Freeze entry phase as shown in [Figure 1](#), page 4. The user fabric logic must be interfaced with the CoreSysServices IP, which is an AHB bus master that puts the device to the lowest possible power state. The steps can be customized to skip the power saving steps, if not required.

For more information, see the Flash*Freeze Services section in Table 1 in the [CoreSysServices v3.1 Handbook](#).

Figure 1 • Flash*Freeze Entry



2.2.1.1.1 Flash*Freeze Entry Time

After receiving the Flash*Freeze system service commands through the COMM_BLK, the system controller disables the PLL output clocks if they are used in the design. The clock is disabled by asserting the PLL power down (MPLLPD) bit in the FFOPTIONS register. For more information, see the FFOPTIONS table in the *UG0450: SmartFusion 2 and IGLOO2 System Controller User Guide*. The time taken from the initiation of Flash*Freeze trigger to the disabling of the clock is known as Flash*Freeze entry time.

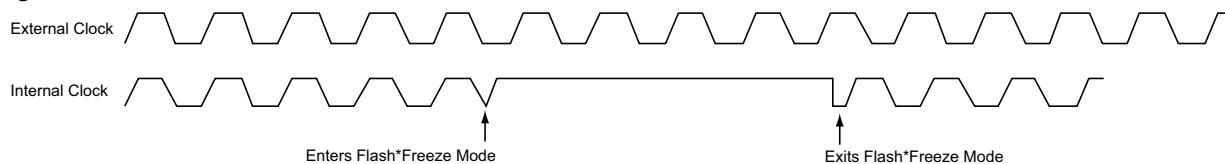
2.2.1.2 PLL/CCC

If a fabric or embedded PLL/CCC is used, entering Flash*Freeze mode automatically powers down the PLL/CCC. The PLL/CCC configuration retains its state during Flash*Freeze operation and recovers the previous state, that is, when Flash*Freeze exits PLL/CCC returns to the normal operation.

2.2.1.3 I/Os and Globals

While entering Flash*Freeze mode, inputs, globals, and PLL/CCC enter the Flash*Freeze state asynchronously to each other. As a result, clock, data glitches, and narrow pulses can be generated while entering.

Figure 2 • Flash*Freeze Modes



I/O banks are not deactivated simultaneously while entering Flash*Freeze mode. This can cause clocks and inputs to become disabled at different times, resulting in unexpected data being captured.

The clocks must be gated properly to prevent glitches. For more information about handling clock gating, see *Flash*Freeze Design*, page 19.

2.2.2 During Flash*Freeze Phase

The operational behavior of the MSS or HPMS, FPGA fabric, FDDR, high-speed serial blocks, PLLs/CCCs, and I/Os after entering into Flash*Freeze mode are discussed in subsequent sections.

2.2.2.1 MSS or HPMS Operation in Flash*Freeze Mode

During Flash*Freeze mode:

- The MSS or HPMS is always powered.
- The MSS or HPMS is clocked by the 50 MHz RC oscillator, the 1 MHz RC oscillator.
- You can specify which I/Os remain active during Flash*Freeze mode in I/O attribute editor.
- The MDDR is not active. The MDDR must be put into self-refresh mode before entering Flash*Freeze mode.

2.2.2.2 Fabric State During Flash*Freeze Mode

During Flash*Freeze mode:

- The FPGA fabric is fully powered down, but the register contents are held in the suspend latches. The suspend latches remain powered up, and upon exit from the Flash*Freeze mode, they restore the register with the contents present before Flash*Freeze entry.
- The fabric SRAMs can be configured in Libero® to enter into suspend mode or sleep mode during Flash*Freeze. This applies to both the large SRAM (LSRAM) instances of RAM1xK18 and the micro SRAM (μSRAM) instances of RAM64x18. These SRAM instances can independently be setup to enter into one of the two modes defined below:
 - Suspend mode: LSRAM and μSRAM contents are retained.
 - Sleep mode: LSRAM and μSRAM contents are not retained.

- CCCs are turned off during Flash*Freeze mode. The CCC configuration retains its state during Flash*Freeze operation and recovers the previous state upon Flash*Freeze exit.
- Mathblocks are not operational. Mathblocks are turned off during Flash*Freeze mode. The mathblock configuration retains its state during the Flash*Freeze operation and recovers the previous state upon Flash*Freeze exit. If the Flash*Freeze is configured in Sleep mode, the mathblock configuration is lost.

2.2.2.3 FDDR Operation in Flash*Freeze Mode

The FDDR controller is not operational during Flash*Freeze mode.

2.2.2.4 High-Speed Serial Block Operation in Flash*Freeze Mode

The high-speed serial block is not operational (powered down) during Flash*Freeze mode.

2.2.2.5 I/O State in Flash*Freeze Mode

The I/O pads are placed in a low power mode, except for MSS or HPMS related I/Os, which are not affected by Flash*Freeze mode. The I/Os can be individually configured using the following options:

- Hold the previous state: This feature holds the last valid state of the input or output pad before the device enters into the Flash*Freeze mode. Weak pull up or weak pull down of I/O pads can be configured along with the hold feature without affecting the hold state. The I/O pad can be configured to hold the previous state in the **I/O Editor** of Libero. The following figure shows the I/O Editor with the I/O state in Flash*Freeze mode column and options highlighted. There are two options available:
 - **TRISTATE**: Tristates the I/O.
 - **LAST_VALUE**: Holds the previous state of the I/O.

Note: Setting this option will activate a feature called Bus Keeper, which is only available in Flash*Freeze mode (not during normal operation).

Figure 3 • Bus Keeper Configuration in I/O Editor

I/O Editor								
Port Name ▲	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode
CLK0_PAD	Input	LVC MOS25	Unassigned	--	--	TRISTATE	None	No
FF_0_O	Output	LVC MOS25	Unassigned	--	--	TRISTATE	None	No
FF_1_O	Output	LVC MOS25	Unassigned	--	--	LAST_VALUE	None	No

- Set I/O pad to weak pull up or weak pull down: If this feature is used without the output hold feature, the input and output pads maintain the configured weak pull up or pull down status during the Flash*Freeze mode and performs normal operation. The I/O pad can be configured to weak pull up or pull down in the I/O Editor of Libero. The following figure shows the I/O Editor with the Resistor Pull column and options highlighted. There are three options available:
 - **None**: No weak pull up or pull down
 - **Down**: Weak pull down
 - **Up**: Weak pull up

Figure 4 • Resistor Pull Configuration

I/O Editor								
Port Name ▲	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode
CLK0_PAD	Input	LVC MOS25	Unassigned	--	--	TRISTATE	None	No
FF_0_O	Output	LVC MOS25	Unassigned	--	--	TRISTATE	None	No
FF_1_O	Output	LVC MOS25	Unassigned	--	--	TRISTATE	Down	No

The I/Os that do not use the hold state or I/O pad weak pull-up or pull-down features are tristated during Flash*Freeze mode.

2.2.2.6 Fabric PLL/CCC

The PLL/CCC is powered down during Flash*Freeze mode while retaining the configuration state.

Note: The input pads and input clocks to the FPGA fabric can toggle without any impact on the static power consumption if weak pull up or pull down is not selected. The input pads must never be allowed to float near the mid-rail, as it can cause totem-pole current in the input buffer.

2.2.3 Flash*Freeze Exit Phase

Flash*Freeze exit wakes up the device from Flash*Freeze mode and returns the device or design to normal operation. Exit from the Flash*Freeze state can be initiated by any one of the following:

- I/O activity
- I/O signature
- RTC timeout

2.2.3.1 I/O Activity

In I/O activity, an I/O can be selected to be part of the Flash*Freeze exit trigger. The value at the pin of the selected I/O is latched before going to low-power mode. When a change takes place on the configured I/O, the device wakes up from Flash*Freeze mode.

The fabric I/Os, MSS, or HPMS peripheral I/Os are available to select the Flash*Freeze wake-up pin. The following figure shows the I/O configuration in the I/O attribute editor.

Figure 5 • I/O Activity Configuration

I/O Editor												
Port Name ▲	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Schmitt Trigger	Odt_Static	Odt Imp (Ohm)	Low Power Ext
CLK0_PAD	Input	LVCMS25	Unassigned	--	--	TRISTATE	None	No	Off	--	--	Off
FF_Wakeonchange	Input	LVCMS25	AB4	<input checked="" type="checkbox"/>	Bank7	TRISTATE	None	No	Off	--	--	Wake_On_Change ▼
SERV_BUSY	Output	LVCMS25	Unassigned	--	--	TRISTATE	None	No	--	--	--	Off
SERV_ENABLE_REQ	Input	LVCMS25	Unassigned	--	--	TRISTATE	None	No	Off	--	--	Wake_On_Change
Wakeon_0	Output	LVCMS25	Unassigned	--	--	TRISTATE	None	No	--	--	--	Wake_On_0
												Wake_On_1

2.2.3.2 I/O Signature

Any I/O can be selected to be part of a signature match value while in Flash*Freeze mode. All other I/Os are tristated or held to the previous state before entering Flash*Freeze mode or weakly pulled up/pulled down. The selected I/Os need to match a static predetermined value at the same time.

If the configured signature values match the values at the pins, then the device exits low-power mode. The following options are available in Libero:

- Low power signature look for 0 (Wake_on_0)
- Low power signature look for 1 (Wake_on_1)

Flash*Freeze exit is executed in a sequence by the system controller after triggering any of the configured events (I/O activity, I/O signature, or RTC timeout). Exit from Flash*Freeze mode can be achieved by the following:






- ARM Cortex-M3 firmware
- Fabric master (if ARM Cortex-M3 processor is not enabled)

The following figure shows the I/O signature configuration in the I/O attribute editor.

Figure 6 • I/O Signature Configuration

I/O Editor - top_top*

File Edit View Tools Help



I/O Editor

Port Name ▼	Direction	I/O Standard	Pin Number	Locked	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Schmitt Trigger	Odt_Static	Odt Imp (Ohm)	Low Power Exit
SERV_ENABLE_REQ	Input	LVCMS25	Unassigned	--	--	TRISTATE	None	No	Off	--	--	Off
SERV_BUSY	Output	LVCMS25	Unassigned	--	--	TRISTATE	None	No	--	--	--	--
FF_IOPaterrn_4_i	Input	LVCMS25	AB1	<input checked="" type="checkbox"/>	Bank7	TRISTATE	None	No	Off	--	--	Wake_On_0
FF_IOPaterrn_3_i	Input	LVCMS25	AA29	<input checked="" type="checkbox"/>	Bank3	TRISTATE	None	No	Off	--	--	Wake_On_1
FF_IOPaterrn_2_i	Input	LVCMS25	AA28	<input checked="" type="checkbox"/>	Bank3	TRISTATE	None	No	Off	--	--	Wake_On_0
FF_IOPaterrn_1_i	Input	LVCMS25	AA27	<input checked="" type="checkbox"/>	Bank3	TRISTATE	None	No	Off	--	--	Wake_On_1
FF_IOPaterrn_0_i	Input	LVCMS25	AA26	<input checked="" type="checkbox"/>	Bank3	TRISTATE	None	No	Off	--	--	Wake_On_0

2.2.3.3 RTC Timeout

Prior to entering Flash*Freeze mode, user logic must configure the RTC module. The timeout value must be set appropriately for the application needs. One of the on-chip clock resources must be driving the RTC.

Exit from Flash*Freeze mode can also be achieved via the Cortex-M3 processor. Setting the WAKEUP_SET bit in the RTC control register results in assertion of the RTC wakeup interrupt. The RTC wakeup interrupt is routed to the system controller, fabric, and the Cortex-M3 processor nested vectored interrupt controller (NVIC). RTC_WAKEUP_CONFIG in the SYSREG block provides masking for the RTC_WAKEUP interrupt to the fabric, Cortex-M3 processor, and the system controller. For accessing the RTC_WAKEUP_CONFIG register, see [Table 1](#), page 3.

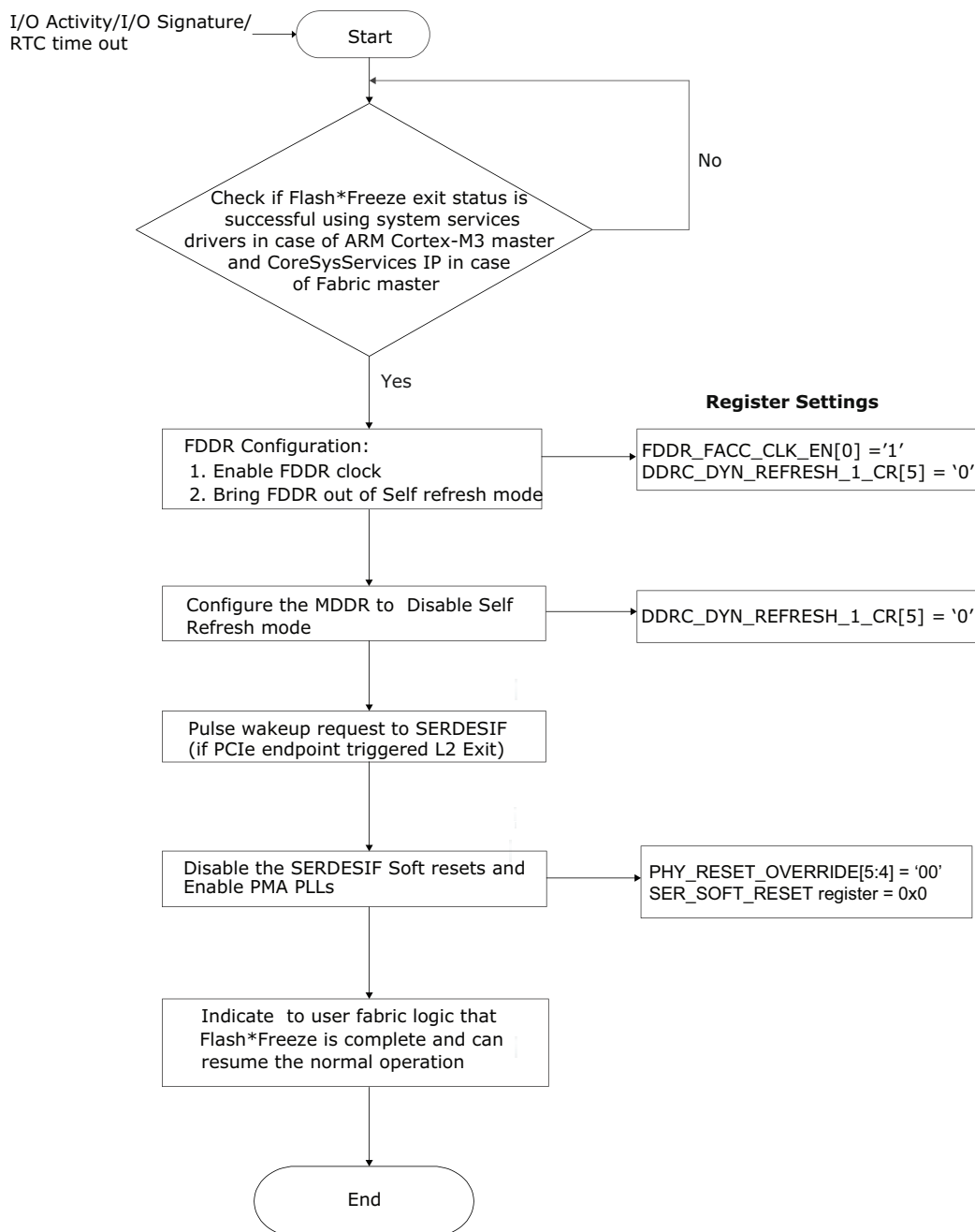
Note: RTC timeout is not available for the IGLOO2 devices.

Flash*Freeze exit is executed in a sequence by the system controller after triggering any of the configured events (I/O activity, I/O signature, or RTC timeout). Exit from Flash*Freeze mode can be achieved by the following:

- Cortex-M3 firmware
- Fabric master (if Cortex-M3 processor is not enabled).

2.2.3.4 Flash*Freeze Exit

To resume normal operation, the ARM Cortex-M3 firmware or user fabric logic has to follow the sequence of steps listed in the following flow chart based on the master used for triggering Flash*Freeze exit. If fabric master is used, then user fabric logic needs to interface with the AHB bus master CoreSysServices IP on initiating the Flash*Freeze exit, as shown in [Figure 7](#), page 9. For more information, see the [CoreSysServices v3.1 Handbook](#).

Figure 7 • Fabric Master initiated Flash*Freeze Exit

2.2.3.4.1 Flash*Freeze Exit Time

The system controller wakes the device from Flash*Freeze mode on Flash*Freeze exit trigger. The time taken from the initiation of the Flash*Freeze exit trigger to the enabling of the clock is known as Flash*Freeze exit time. If MPLL is used, the MPLL clock is enabled, or if a fabric PLL is used, the fabric PLL clock and MPLL clock are enabled.

If the MSS_CLK_BASE is being generated from the fabric PLL, then there would be a sequence of fabric PLL lock first and then MSS PLL lock.

For more information about PLL lock feature, see the PLLs Lock Monitoring section under MSS Clock Conditioning Circuitry chapter in [UG0449: SmartFusion2 and IGLOO2 Clocking Resources User Guide](#).

2.2.3.5 I/Os and Globals

The following describes how the I/O and globals behave during Flash*Freeze exit:

- While exiting Flash*Freeze mode, the inputs and globals exit their Flash*Freeze state asynchronously to each other. The order in which individual I/Os and globals exit the Flash*Freeze state is a function of routing and logic delays in the I/O ring. Clock, data glitches, and narrow pulses can be generated while exiting Flash*Freeze mode, unless clock gating schemes are used. For more information about how to handle the clock gating, see [Flash*Freeze Design](#), page 19.
- All I/O banks are not activated simultaneously when exiting the Flash*Freeze mode. This can cause the clocks and inputs to enable themselves at different times, resulting in unexpected data being captured.
- The output hold state is asynchronously controlled by the signal driving the output buffer (output signal). This ensures a clean, glitch-free transition from the hold state to output drive. However, any glitches on the output signal during exit from Flash*Freeze mode can result in glitches on the output pad.
- The preceding situations can cause glitches or invalid data to be clocked into and preserved in the device.

2.2.3.6 PLL/CCC

If the embedded PLL/CCC is used, the design must allow maximum acquisition time (per device datasheet) for the PLL/CCC to acquire the lock signal. The lock signal can be used to gate the clock coming out of a PLL/CCC.

2.3 Using Flash*Freeze in SmartFusion2

The following sub sections describe how to use Flash*Freeze in an application:

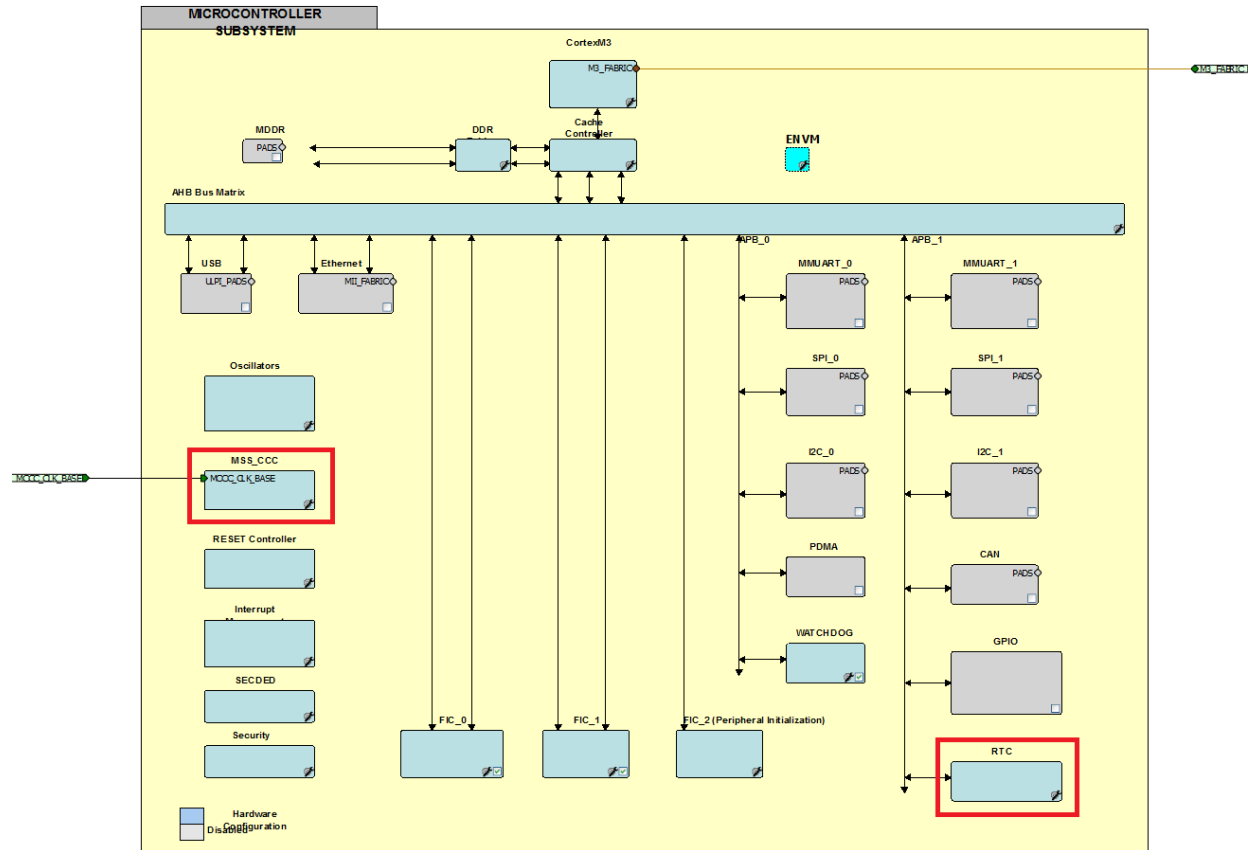
- Design flow
- Flash*Freeze use models
- Flash*Freeze design

2.3.1 Design Flow

The following steps describe the Libero configuration settings for using Flash*Freeze in an application:

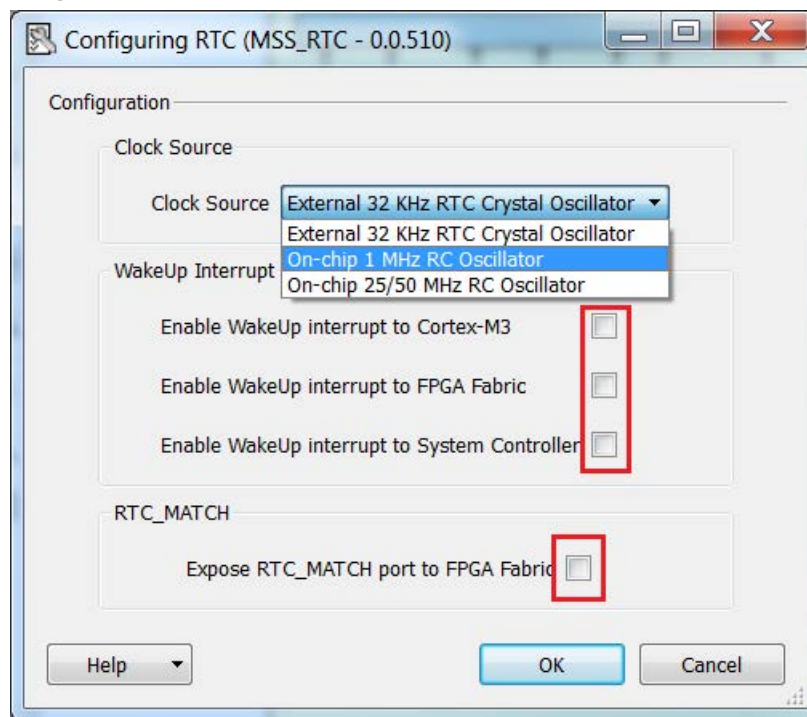
1. Configure the highlighted blocks (MSS_CCC and RTC) as shown in the following figure; these blocks are enabled by default.

Figure 8 • MSS_CCC and RTC Options for Flash*Freeze



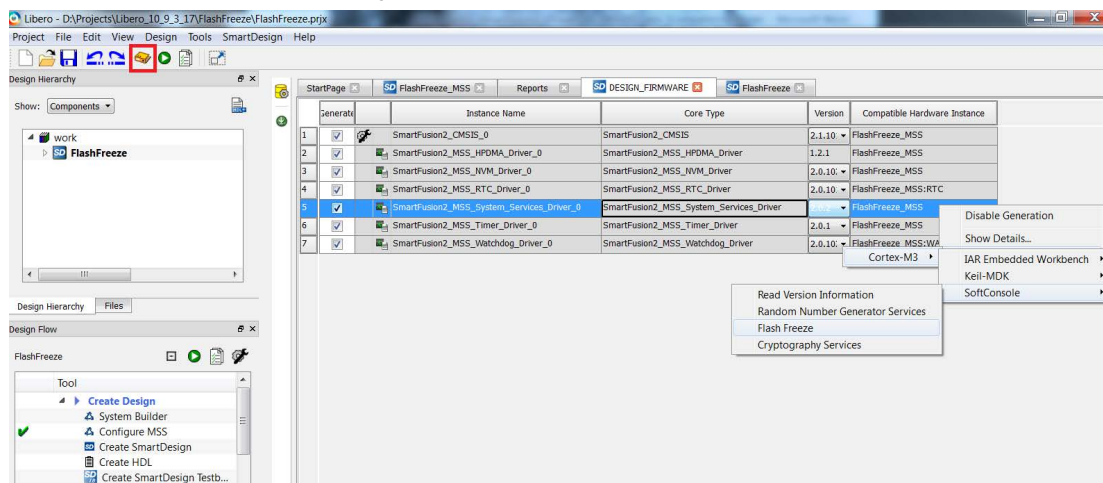
- If RTC timeout interrupt is used for triggering Flash*Freeze exit, configure the **Clock Source** and **WakeUp Interrupt** as shown in the following figure. See **Help** for options available on RTC configurator.

Figure 9 • RTC Configurator



- To generate the component, click **Generate Component** or select **SmartDesign > Generate Component**. For more information about component generation, see the [Libero SoC User's Guide](#). The firmware driver folder and Soft Console workspace is created in the project.
- Click **Configure firmware** as shown in the following figure to find the system service drivers for Flash*Freeze.

Figure 10 • Flash*Freeze Example Projects



- Click **Generate Programming Data** to complete *.fdb file generation. Double-click **Write Application Code** under the Libero design flow window to invoke the Soft Console IDE. The Soft Console folder contains the mss_system_services drivers. The firmware driver, mss_system_services (mss_sys_services.c and mss_sys_services.h) provides functions for Flash*Freeze.

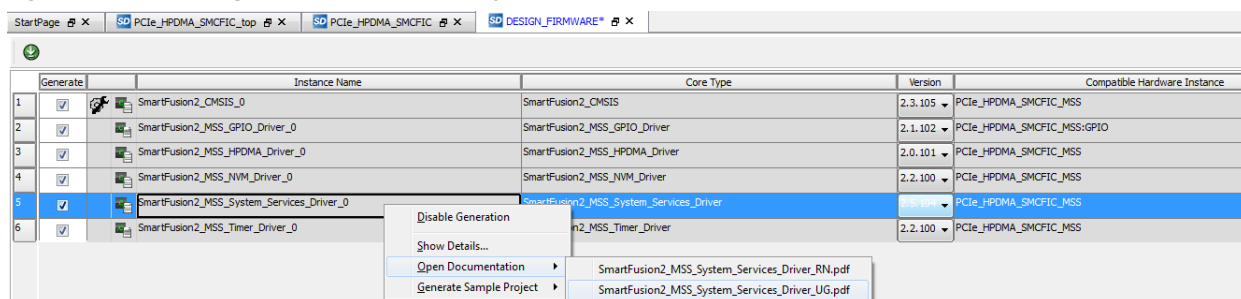
The following table lists the APIs for Flash*Freeze.

Table 2 • System Service API for Flash*Freeze

Category	API	Description and Usage
Initialization	MSS_SYS_init()	Initializes the system services communication with system controller.
Control and status	MSS_SYS_flash_freeze()	Requests the FPGA to enter Flash*Freeze mode. The function returns following status codes: - Success - Memory access error - Unexpected error

For more information on the APIs, see the **SmartFusion2 MSS System Services Driver User Guide** by following the path shown in the following figure.

Figure 11 • Opening SmartFusion2 MSS System Services Driver User Guide



For more information about MSS system services support for Flash*Freeze usage, a sample project is available, which can be generated as shown in [Figure 10](#), page 12.

2.3.2 Flash*Freeze Use Model

For Cortex-M3 firmware Flash*Freeze entry on GPIO input and Flash*Freeze exit on RTC timeout condition (20 μ s), use the following steps for Flash*Freeze entry and exit:

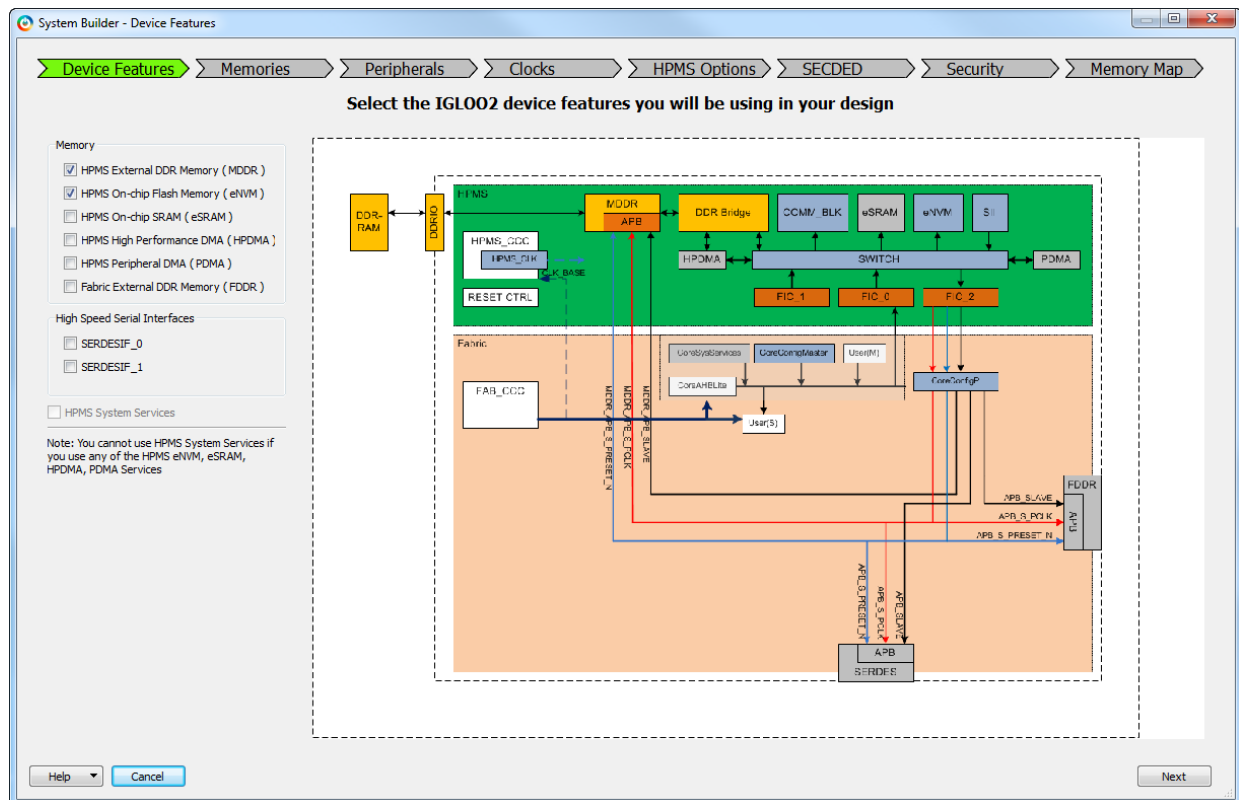
1. Select the standby source clock using Libero.
2. Initialize the communication with system controller using MSS_SYS_init().
3. If RTC is used for Flash*Freeze exit trigger, initialize the RTC using RTC_init().
4. Initialize the peripheral GPIOs using MSS_GPIO_init().
5. Wait for Flash*Freeze entry trigger when there is a transition on GPIO input.
6. Reset the RTC counter using MSS_RTC_reset_counter().
7. Set the RTC timeout value to 20 μ s and single shot alarm to exit Flash*Freeze using MSS_RTC_set_binary_count_alarm().
8. Enable the RTC wakeup interrupt using MSS_RTC_enable_irq() or configure using Libero.
9. Start the RTC counter using MSS_RTC_start().
10. Request Flash*Freeze shutdown using MSS_SYS_flash_freeze().
11. Check the status of the command by reading the return value of the MSS_SYS_flash_freeze().
12. Check for Flash*Freeze entry with system services event handler opcode.
13. After the timeout, RTC interrupt is generated. Check for Flash*Freeze exit with system services event handler opcode.
14. Clear the RTC interrupt using MSS_RTC_clear_irq().
15. The device comes out of Flash*Freeze mode.

2.4 Using Flash*Freeze in IGL002

This section describes how to use Flash*Freeze mode in IGLOO2 devices. To configure the IGLOO2 device features and then build a complete system, use the System Builder graphical design wizard in Libero.

The following figure shows the initial System Builder window where the desired device features can be selected. For information about using the System Builder wizard, see the *IGLOO2 System Builder User Guide*.

Figure 12 • System Builder - Device Features Window

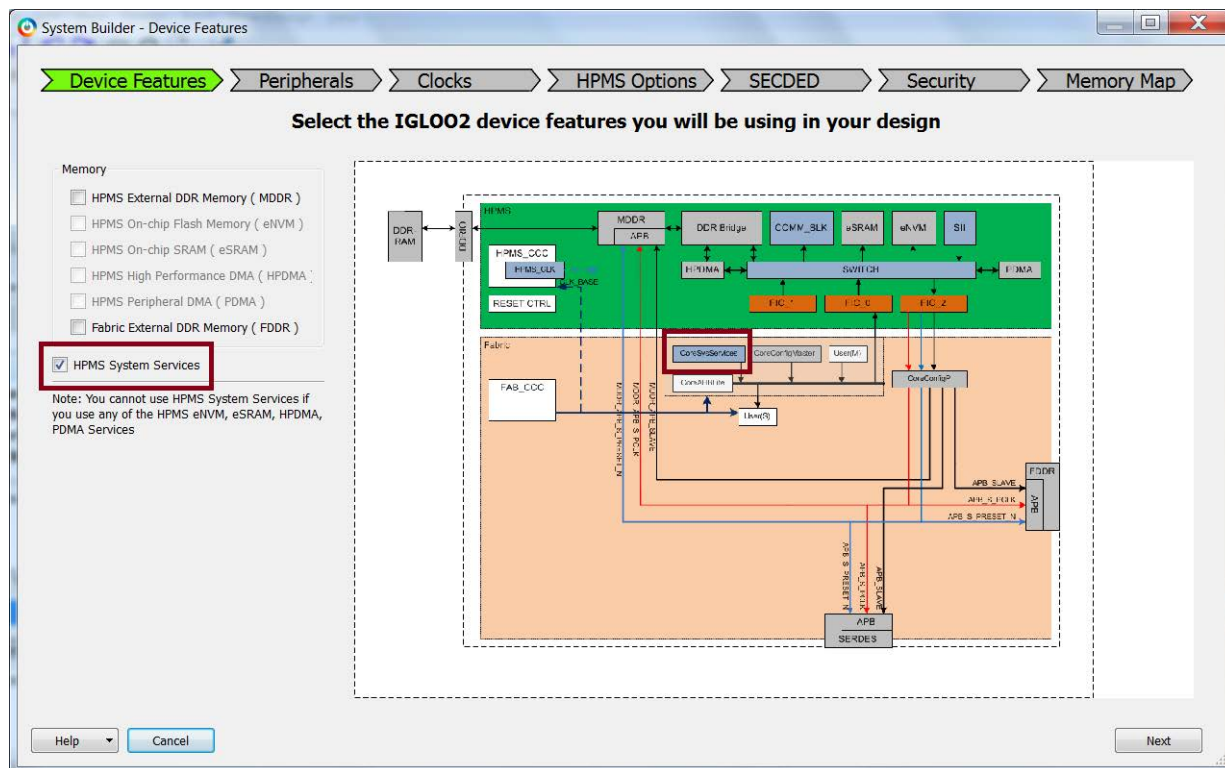


2.4.1 Enabling Flash*Freeze

The following steps describe how to enable Flash*Freeze mode in IGLOO2 devices:

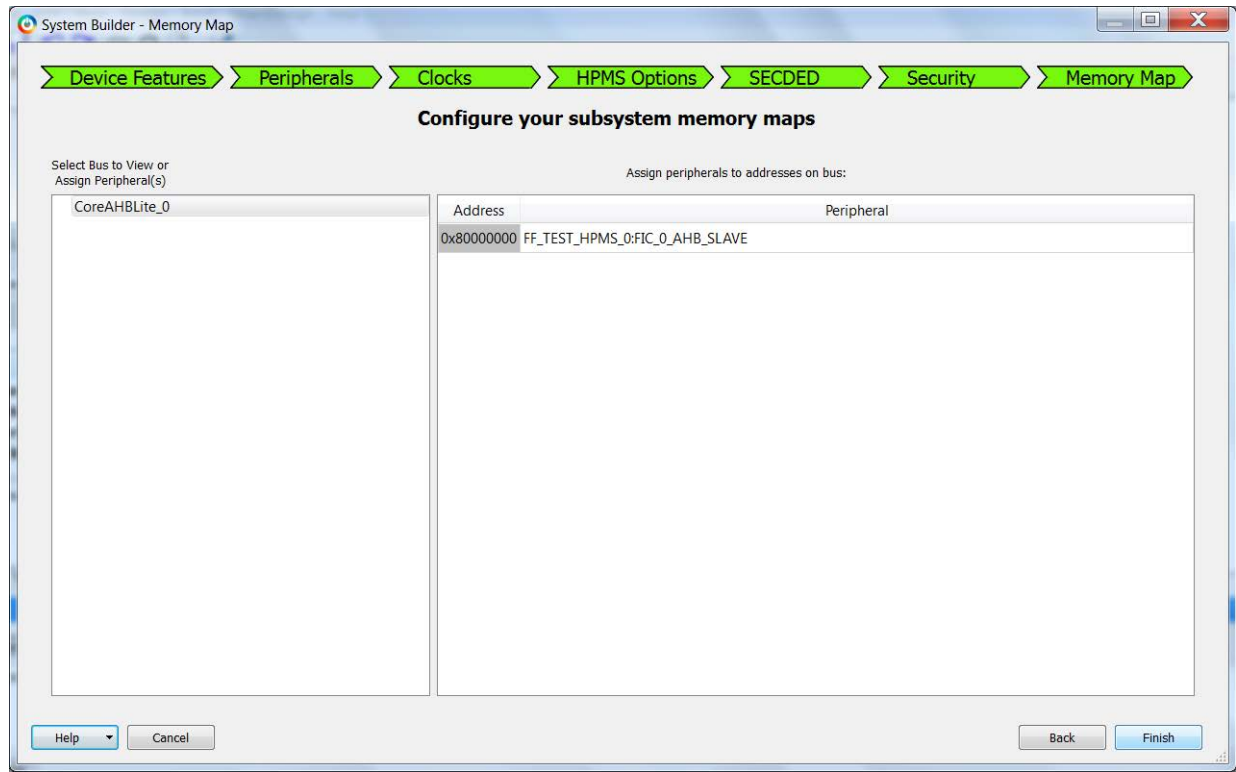
1. Select **HPMS System Services** under the **Device Features** tab to enable Flash*Freeze. This establishes a path for connecting the CoreSysServices soft IP to the COMM_BLK through the FIC_0 interface. This is indicated by a change of color in the CoreSysServices block in the System Builder. If the CoreSysServices IP is connected as an AHB master, the other interfaces (eNVM, eSRAM, HPDMA, and PDMA) cannot be used to access the COMM_BLK through the FIC_0 interface. The following figure shows the CoreSysServices block in the System Builder wizard.

Figure 13 • System Builder - Device Features Tab



- Go to the **Memory Map** tab, as the remaining System Builder tabs need not be configured. The following figure shows the **System Builder - Memory Map** tab.

Figure 14 • System Builder - Memory Map Tab

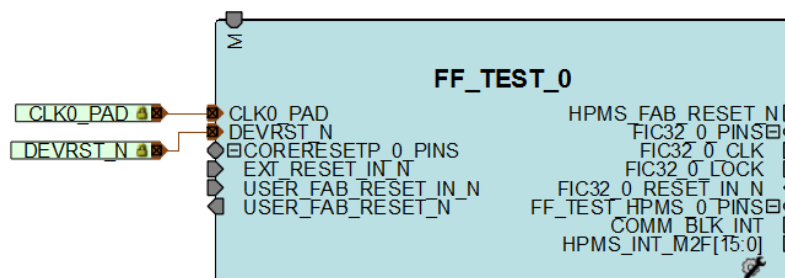


- Click **Finish** to proceed with creating the HPMS System.

2.4.2 HPMS Subsystem

The following figure shows an example HPMS subsystem that can be used to connect the CoreSysServices fabric master to place the device in Flash*Freeze.

Figure 15 • HPMS Subsystem



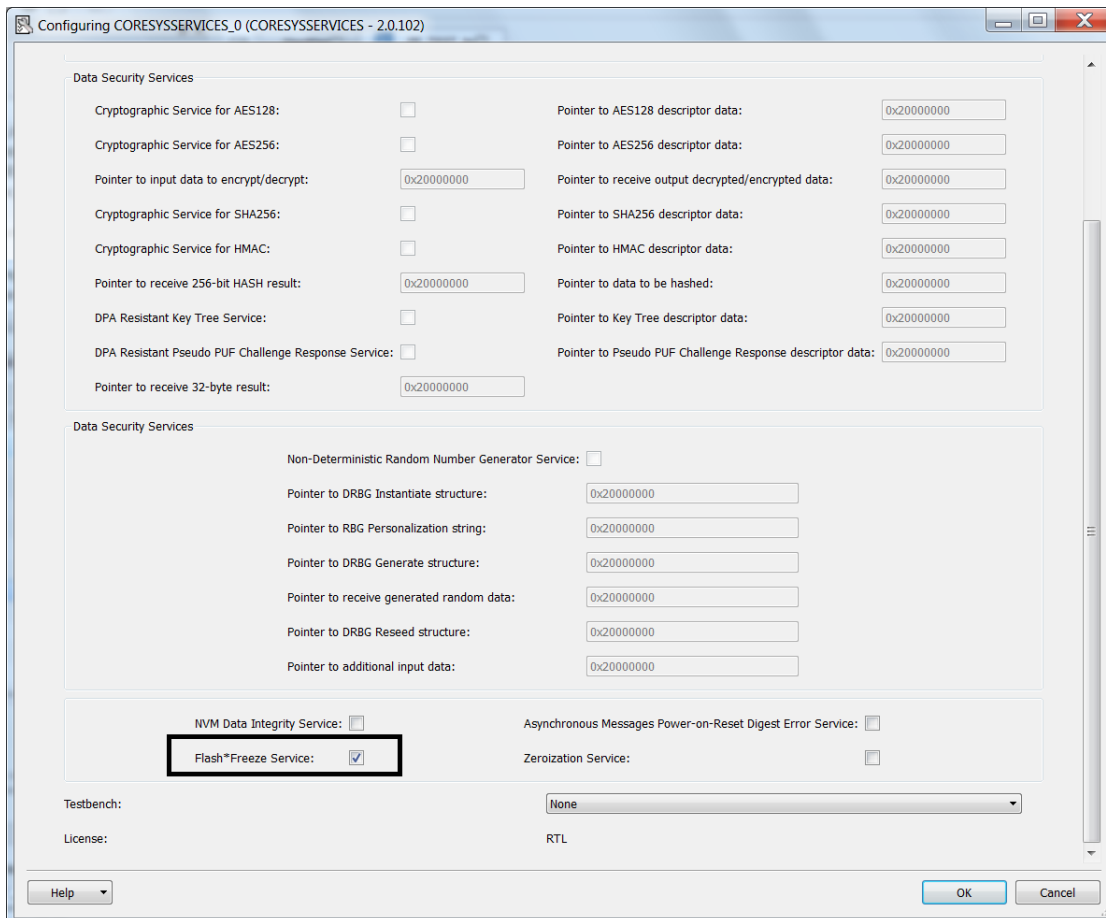
2.4.3 Entering and Exiting Flash*Freeze Mode

This section describes how to enter the Flash*Freeze entry phase using the CoreSysServices IP fabric master and go to the Flash*Freeze exit phase using the *Wake_On_Change* input option.

Flash*Freeze Entry Phase

1. Enable Flash*Freeze mode in Libero using the System Builder wizard as described in [Enabling Flash*Freeze](#), page 15.
2. Select the CoreSysServices IP from the Libero Catalog and instantiate in the SmartDesign canvas.
3. Select **Flash*Freeze Service**. Enter the other parameters required and click **OK** to configure the CoreSystemServices IP and enable Flash*Freeze service.

Figure 16 • CoreSysServices Configuration



Configuring CORESYS SERVICES_0 (CORESYS SERVICES - 2.0.102)

Data Security Services

Cryptographic Service for AES128:	<input type="checkbox"/>	Pointer to AES128 descriptor data:	<input type="text" value="0x20000000"/>
Cryptographic Service for AES256:	<input type="checkbox"/>	Pointer to AES256 descriptor data:	<input type="text" value="0x20000000"/>
Pointer to input data to encrypt/decrypt:	<input type="text" value="0x20000000"/>	Pointer to receive output decrypted/encrypted data:	<input type="text" value="0x20000000"/>
Cryptographic Service for SHA256:	<input type="checkbox"/>	Pointer to SHA256 descriptor data:	<input type="text" value="0x20000000"/>
Cryptographic Service for HMAC:	<input type="checkbox"/>	Pointer to HMAC descriptor data:	<input type="text" value="0x20000000"/>
Pointer to receive 256-bit HASH result:	<input type="text" value="0x20000000"/>	Pointer to data to be hashed:	<input type="text" value="0x20000000"/>
DPA Resistant Key Tree Service:	<input type="checkbox"/>	Pointer to Key Tree descriptor data:	<input type="text" value="0x20000000"/>
DPA Resistant Pseudo PUF Challenge Response Service:	<input type="checkbox"/>	Pointer to Pseudo PUF Challenge Response descriptor data:	<input type="text" value="0x20000000"/>
Pointer to receive 32-byte result:	<input type="text" value="0x20000000"/>		

Data Security Services

Non-Deterministic Random Number Generator Service: ☐

Pointer to DRBG Instantiate structure:

Pointer to RBG Personalization string:

Pointer to DRBG Generate structure:

Pointer to receive generated random data:

Pointer to DRBG Reseed structure:

Pointer to additional input data:

NVM Data Integrity Service: ☐

Flash*Freeze Service: ☒

Asynchronous Messages Power-on-Reset Digest Error Service: ☐

Zeroization Service: ☐

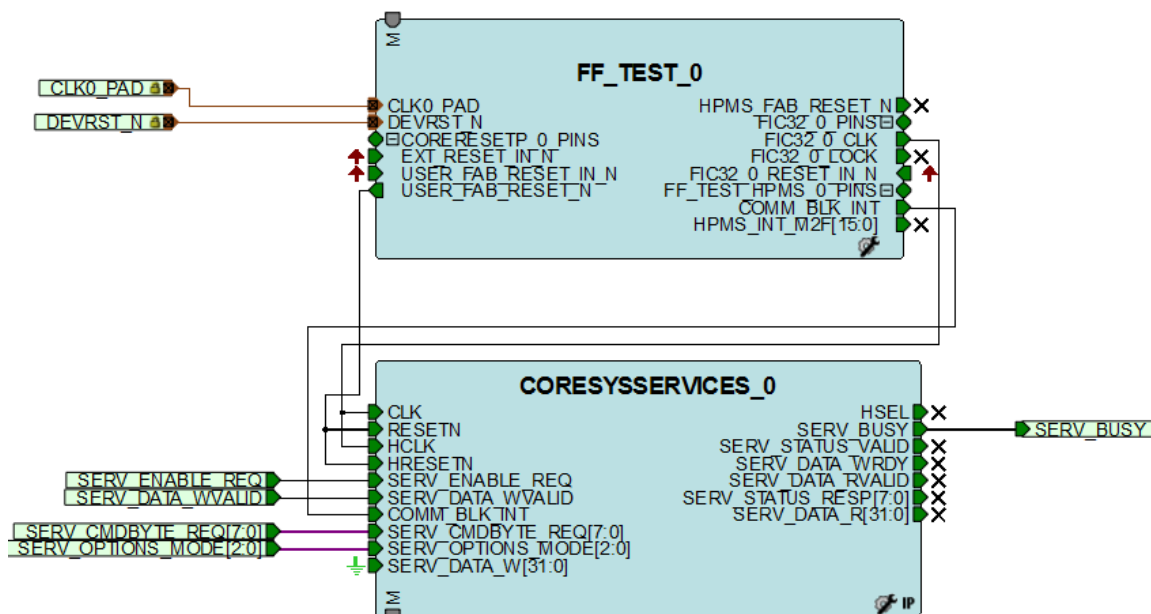
Testbench:

License:

Help OK Cancel

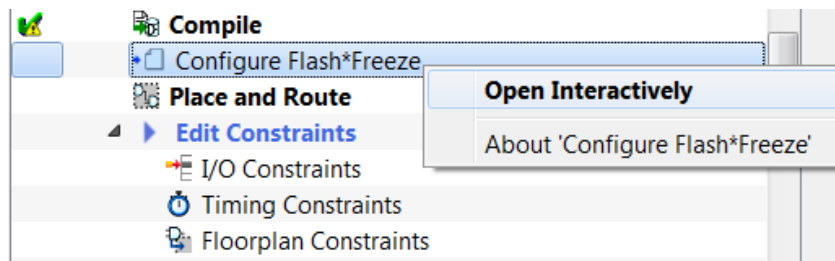
4. Connect CoreSysServices IP to the HPMS subsystem, as shown in the following figure.

Figure 17 • HPMS Subsystem Connections with the CoreSysServices AHB Bus Master



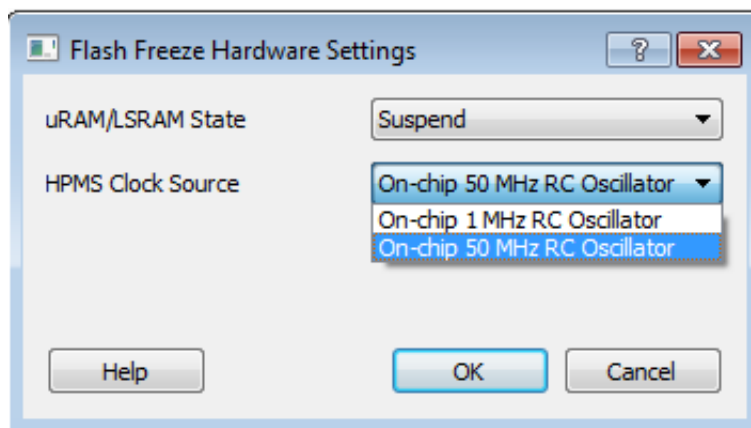
5. After generating the component in the **DesignFlow** tab under **Compile**, right-click **Configure Flash*Freeze** and select **Open Interactively**.

Figure 18 • Configure Flash*Freeze in Design Flow Tab



6. In the **Flash Freeze Hardware Settings** dialog box, select the required configuration for **uRAM/LSRAM State** and **HPMS Clock Source** during Flash*Freeze mode.

Figure 19 • Flash*Freeze Hardware Settings



Flash*Freeze Exit Phase

7. Configure an I/O pad as a **Wake_On_Change** input in the **I/O Attribute Editor**.
8. Connect the I/O pad to an external switch or any other source of trigger.

Figure 20 • Wake_on_Change Configuration

Wakeon_in	Input	LVC MOS25	Unassigned	--	--	TRISTATE	None	No	off	--	--	off	off
Wakeon_O	Output	LVC MOS25	Unassigned	--	--	TRISTATE	None	No	--	--	--	off	--

9. Write the user logic RTL and add a trigger switch to send the Flash*Freeze command—
 SERV_CMDBYTE_REQ = 0x00 (fabric power down) with a SERV_ENABLE_REQ pulse. The device enters into the Flash*Freeze mode.
 For more information on Flash*Freeze commands, see the [CoreSysServices v3.1 Handbook](#) and the [Flash*Freeze Service](#) section available in the [UG0450: SmartFusion2 and IGLOO2 FPGA System Controller User Guide](#).
10. To come out of Flash*Freeze phase, trigger the Wake_On_Change input signal externally.

2.5 Flash*Freeze Design

This section describes how reliable designs that use ultra-low power Flash*Freeze mode optimally can be created. It also gives specific recommendations on how to design and configure clocks.

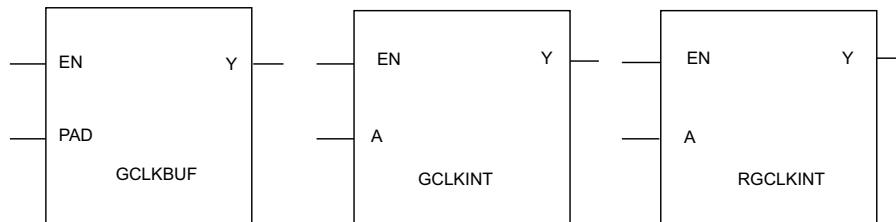
2.5.1 Clocks

Microsemi recommends using a completely synchronous design, cleanly gating all the internal and external clocks. This prevents narrow pulses upon entry to and exit from Flash*Freeze mode.

SmartFusion2 and IGLOO2 devices include sophisticated clock gating for all or portions of a global clock network. All library macros referencing global clock drivers include an enable input to gate the clock when required. The following figure shows global clock macros with the clock gating feature. The clock is passed when the enable pin (EN) is asserted high and held low when the EN pin is disabled.

To prevent glitches, the EN signal must be checked for basic setup and hold timing violations similar to a flip-flop. To clock gate a portion of a clock network, the RGCLKINT macro must be used.

Figure 21 • Global Gated Clock Macros



The following figures show example designs of using single and dual global gated clock buffer macros to control the fabric user logic.

Figure 22 • Gated External Clocks Using Single GCLKBUF Macro

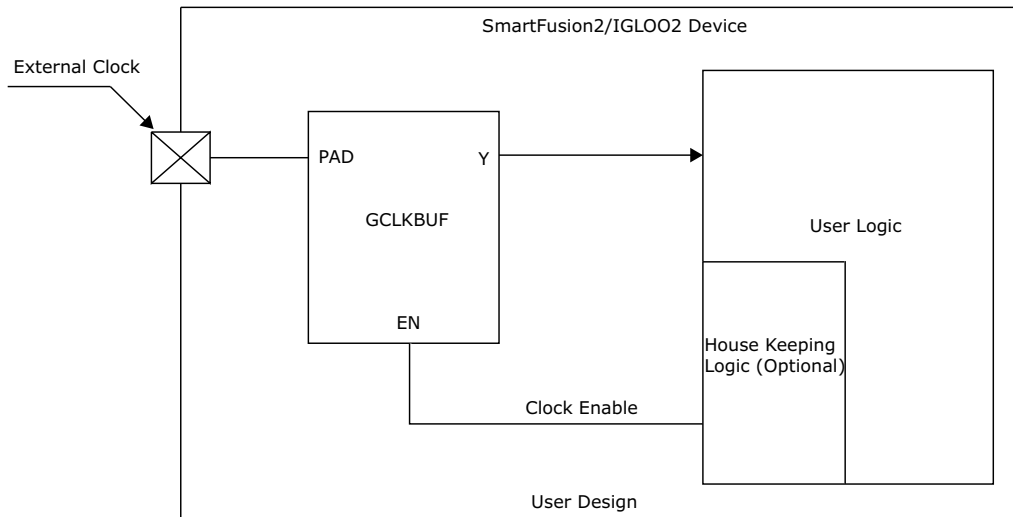
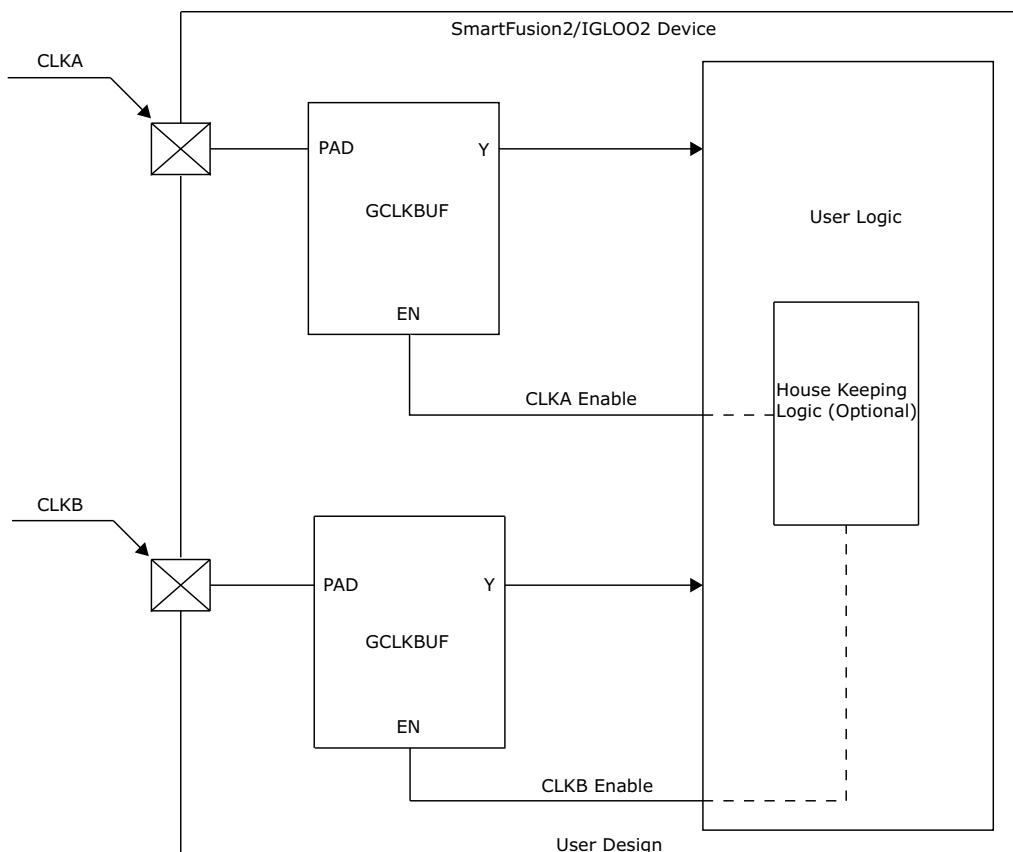


Figure 23 • Gated External Clock Using Multiple GCLKBUF Macros



Clocks can continue to drive the FPGA pins while the device is in Flash*Freeze mode, with virtually no power consumption. Upon exiting Flash*Freeze mode, the design must allow maximum acquisition time for the PLL to acquire the lock signal, and for a PLL clock to become active.

2.5.2 I/Os

The floating inputs can cause totem-pole currents on the input I/O circuitry when the device is in active mode. Microsemi recommends releasing the inputs only after the device enters Flash*Freeze mode.

2.5.3 FLASH_FREEZE Macro

The user service interface (USI) is an interface between the FPGA fabric and the system controller. USI provides two active high output signals: FF_TO_START and FF_DONE to the FPGA fabric, which are made available by instantiating the FLASH_FREEZE macro from the Libero IP catalog or by instantiating it directly inside an HDL file. The following figure shows the FLASH_FREEZE macro, which exposes the FF_TO_START and FF_DONE signals.

Figure 24 • FLASH_FREEZE Macro



FF_TO_START is asserted by the system controller to indicate that the FLASH*FREEZE request is about to start. 10 μ s are provided for housekeeping before the core is powered down. Microsemi recommends using this signal for clock gating the logic, and thereby ensuring that no glitches are transmitted to the sequential element in the design while entering Flash*Freeze.

FF_DONE is asserted by the system controller to indicate that the Flash*Freeze is completed. The same signal must be used to connect the CCC reset input with an inversion so that the CCC is reset after Flash*Freeze exit.

2.5.4 Flash*Freeze Guidelines

Follow these specific guidelines for reliable Flash*Freeze entry and exit:

- All asynchronous resets and presets including reset inputs to ASIC blocks (MDDR, FDDR, and SerDes) must be gated with the FF_DONE signal to ensure that no spurious resets are propagated from the fabric during Flash*Freeze exit.
- All fabric CCC resets must be connected to the FF_DONE signal with inversion to reset CCC during Flash*Freeze exit.
- When using CoreSysServicesIP, entry request (SERV_ENABLE_REQ) must not be held longer than one CoreSysServices clock cycle.
- A completely synchronous design must be used. If asynchronous clocks are involved, a simple resynchronization circuit (double flip-flop) can be added.

Signals from the FPGA fabric to the ASIC blocks in Smartfusion2/IGLOO2 (system controller, MSSDDR, FDDR, and SERDESIF) can go to an invalid state during recovery from suspend mode at the end of Flash*Freeze. These signals can be classified into the following groups:

- Resets
- Clocks
- AMBA bus interfaces (APB, AHB-Lite, AXI)
- MSS signals
- Generic signals

In some cases, it may be necessary to gate signals from the fabric with FF_DONE to ensure that they are in inactive state (which may be high or low depending on the signal) during recovery from suspend mode.

This section specifies how the preceding group of signals must be handled to avoid issues arising from a potentially invalid state during recovery from suspend mode.

2.5.4.1 Resets

Resets must be forced inactive during recovery from suspend mode. This is done by gating all resets generated from CoreResetP.

2.5.4.2 Clocks

The clocks from the fabric to the ASIC blocks do not need any additional gating, as they are shut off during Flash*Freeze mode.

2.5.4.3 AMBA Bus Interfaces

Each of the ASIC blocks has one or more AMBA bus interfaces (APB/AHB-Lite/AXI) connected to it. For any such fabric interface, which is being controlled in the fabric, the control signal that initiates transactions (PSEL for APB, HTRANS for AHB-Lite, and VALID signals for AXI) must be gated off with FF_DONE. However, as some of the bus interfaces are overlaid onto shared signals, gating is dependent on the configuration of the bus interface. The following tables specify the required gating of AMBA-related signals from the fabric to each ASIC block. Gating can be accomplished by incorporating the function internally within the various AMBA soft-IP cores. If a particular bus interface is held in reset (soft or hard) during Flash*Freeze, then gating need not be done. This is desirable if there are timing concerns at the fabric interface.

2.5.4.3.1 Fabric to MDDR

The following table specifies the gating of AMBA-related signals from the fabric to the MSSDDR. In each case, the signal is to be forced to logic 0.

Table 3 • Gating AMBA-related Signals from Fabric to MSSDDR

Function	Libero MSS Name
HTRANS (FIC_0 AHBL)	FIC_0_AHB_S_HTRANS[1]
HTRANS (FIC_1 AHBL)	FIC_1_AHB_S_HTRANS[1]
HTRANS (FIC64 AHBL0)	MDDR_DDR_AHB0_S_HTRANS
AWVALID (FIC64 AXI)	MDDR_DDR_AXI_S_AWVALID
WVALID (FIC64 AXI)	MDDR_DDR_AXI_S_WVALID
HTRANS (FIC64 AHBL1)	MDDR_DDR_AHB1_S_HTRANS
ARVALID (FIC64 AXI)	MDDR_DDR_AXI_S_ARVALID

2.5.4.3.2 Fabric to FDDR

The following table specifies the gating of AMBA-related signals from the fabric to the FDDR. In each case, the signal is to be forced to logic 0.

Table 4 • Gating AMBA-related Signals from Fabric to FDDR

Function	Libero FDDR Name
PSEL	APB_S_PSEL
HTRANS (AHBL0)	AHB0_S_HTRANS[1]
HTRANS (AHBL1)	AHB1_S_HTRANS[1]
AWVALID (AXI)	AXI_S_AWVALID
ARVALID (AXI)	AXI_S_ARVALID
WVALID (AXI)	AXI_S_WVALID

2.5.4.3.3 Fabric to SERDESIF

The following table specifies the gating of AMBA-related signals from the fabric to the SERDESIF. In each case, the signal is to be forced to logic 0.

Table 5 • Gating of AMBA-related Signals from Fabric to SERDESIF

Function	Libero SerDes Name
HTRANS (AHBL)	AHB_M_HTRANS[1]
AWVALID (AXI)	AXI_M_AWVALID
WVALID (AXI)	AXI_M_WVALID
ARVALID (AXI)	AXI_M_ARVALID
PSEL	APB_S_PSEL

2.5.4.4 MSS Signals

The section describes how to gate MSS-related signals with FF_DONE.

2.5.4.4.1 Interrupts

If the MSS needs to be kept operational during Flash*Freeze, the following interrupt signal must be gated off with FF_DONE. If not, the NVIC must be configured before requesting Flash*Freeze entry to mask off interrupts.

Table 6 • Gating MSS Interrupt Signal

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
16'b0	MSS_INT_F2M[15:0]

2.5.4.4.2 Non-Interrupt Signals

It is recommended that the following signals be gated to the level shown in the following table, using FF_DONE.

Table 7 • Gating MSS Non-Interrupt Signals

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
1'b1	M3_SLEEPHOLDREQ
1'b0	M3_RXEV

2.5.4.5 Generic Signals

These are signals that are not categorized in any of the preceding groups.

2.5.4.5.1 Fabric PLL Lock

It is recommended that the fabric PLL lock signal to MSSDDR be gated to the level shown in the following table, using FF_DONE.

Table 8 • Gating Fabric PLL Lock Signals

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
1'b1	MCCC_CLK_BASE_PLL_LOCK

2.5.4.5.2 Signals to MSS Peripherals

The following signals can be gated or not depending on whether the MSS peripheral associated with it is being used.

Table 9 • Gating MSS Peripheral Signals

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
8'b0	USB_UTMI_VSTATUS
2'b0	DMA_DMAREADY_FIC_0
2'b0	DMA_DMAREADY_FIC_1
1'b0	MAC_MII_CRS (MII/GMII)
1'b0	MAC_MII_COL (MII/GMII)
1'b0	MAC_MII_MDI (MII/TBI/GMII)
1'b0	MAC_MII_RX_DV (MII/GMII)
8'b0	MAC_MII_RXD (MII/GMII)
1'b0	MAC_MII_RX_ER (MII/GMII)
10'b0	MAC_TBI_RCGF
1'b0	I2C_0_SMBALERT_NI
1'b0	I2C_0_SMBSUS_NI
1'b0	I2C_1_SMBALERT_NI
1'b0	I2C_1_SMBSUS_NI
2'b0	USB_UTMI_LINE_STATE
1'b0	USB_UTMI_RX_ACTIVE
1'b0	USB_UTMI_RX_VALID
1'b0	USB_UTMI_RX_ERROR
8'b0	USB_UTMI_RX_DATA
1'b0	USB_UTMI_TX_READY
1'b0	USB_UTMI_VBUS_VALID
1'b0	USB_UTMI_AVALID
1'b0	USB_UTMI_SESSION_END
1'b0	USB_UTMI_ID_DIG
1'b0	USB_UTMI_HOST_DISCONNECT
1'b0	GPIO_0_F2M (FABRIC_A)
1'b0	I2C_1_SDA_F2M
1'b0	GPIO_1_F2M (FABRIC_A)
1'b0	I2C_1_SCL_F2M
1'b0	GPIO_2_F2M (FABRIC_A)
1'b0	GPIO_3_F2M (FABRIC_A)
1'b0	CAN_RX_F2M
1'b0	GPIO_4_F2M (FABRIC_A)
1'b0	GPIO_5_F2M (FABRIC_A)

Table 9 • Gating MSS Peripheral Signals (continued)

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
1'b0	SPI_0_DI_F2M
1'b0	GPIO_6_F2M (FABRIC_A)
1'b0	GPIO_7_F2M (FABRIC_A)
1'b0	SPI_0_SS0_F2M
1'b0	GPIO_8_F2M (FABRIC_A)
1'b0	GPIO_9_F2M (FABRIC_A)
1'b0	GPIO_10_F2M (FABRIC_A)
1'b0	GPIO_11_F2M (FABRIC_A)
1'b0	SPI_1_DI_F2M
1'b0	GPIO_12_F2M (FABRIC_A)
1'b0	GPIO_13_F2M (FABRIC_A)
1'b0	SPI_1_SS0_F2M
1'b0	GPIO_14_F2M (FABRIC_A)
1'b0	GPIO_15_F2M (FABRIC_A)
1'b0	GPIO_16_F2M (FABRIC_A)
1'b0	GPIO_11_F2M (FABRIC_B)
1'b0	MMUART_1_CTS_F2M
1'b0	MMUART_1_DSR_F2M
1'b0	MMUART_1_RI_F2M
1'b0	MMUART_1_DCD_F2M
1'b0	GPIO_24_F2M (FABRIC_B)
1'b0	MMUART_1_TXD_RXD_F2M MMUART_1_TXD_F2M
1'b0	GPIO_25_F2M (FABRIC_B)
1'b0	MMUART_1_TXD_RXD_F2M MMUART_1_RXD_F2M
1'b0	GPIO_26_F2M (FABRIC_B)
1'b0	MMUART_1_CLK_F2M
1'b0	GPIO_17_F2M (FABRIC_B)
1'b0	GPIO_18_F2M (FABRIC_B)
1'b0	GPIO_19_F2M (FABRIC_B)
1'b0	MMUART_0_CTS_F2M
1'b0	GPIO_20_F2M (FABRIC_B)
1'b0	MMUART_0_DSR_F2M
1'b0	GPIO_21_F2M (FABRIC_B)
1'b0	MMUART_0_RI_F2M
1'b0	GPIO_22_F2M (FABRIC_B)
1'b0	MMUART_0_DCD_F2M

Table 9 • Gating MSS Peripheral Signals (continued)

Required Gating Level if FF_DONE is Asserted	Libero MSS Name
1'b0	GPIO_27_F2M (FABRIC_B)
1'b0	MMUART_0_TXD_RXD_F2M MMUART_0_TXD_F2M
1'b0	GPIO_28_F2M (FABRIC_B)
1'b0	MMUART_0_TXD_RXD_F2M MMUART_0_RXD_F2M
1'b0	GPIO_29_F2M (FABRIC_B)
1'b0	MMUART_0_CLK_F2M
1'b0	GPIO_30_F2M (FABRIC_B)
1'b0	I2C_0_SDA_F2M
1'b0	GPIO_31_F2M (FABRIC_B)
1'b0	GPIO_0_F2M (FABRIC_A)

2.5.4.5.3 Signals to System Controller

It is recommended that the following response signals to the system controller be gated to the level shown in the following table, using FF_DONE.

Table 10 • Gating System Controller Response Signals

Required Gating Level if FF_DONE is Asserted	Libero Name on TAMPER Macro
3'b111	LOCKDOWN_ALL_N DISABLE_ALL_IOS_N RESET_N

2.6 SYSREG Control Registers for Flash*Freeze

The following table lists the registers for Flash*Freeze. For more information, see the System Register Block chapter of the [UG0448: IGLOO2 High Performance Memory Subsystem User Guide](#).

Table 11 • SYSREG Control Register

Register Name	Register Type	Flash Write Protect	Reset Source	Description
DEVICE_SR	RO-P	Register	SYSRESET_N	Device Status register
RTC_WAKEUP_CONFIG	RW-P	Register	SYSRESET_N	Masks RTC_WAKEUP interrupt to fabric, the Cortex-M3 processor, and the system controller.

2.7 Acronyms

AHB-Lite

AMBA High Performance Bus Lite

APB

AMBA peripheral bus

CCC

Clock conditioning circuit

COMBLK

COM block

ENVM

Embedded nonvolatile memory

LSRAM

Large SRAM

MDDR

Memory subsystem DDR

MSS

Microcontroller Subsystem

μSRAM

Micro SRAM

HPMS

High Performance Memory Subsystem

PDC

Physical Design Constraints

PLL

Phased-locked loop

SYSREG

System registers

2.8 Terminology

Flash*Freeze

Flash*Freeze technology provides an ultra-low power static mode for the SmartFusion2 and IGLOO2 devices.

Flash*Freeze Entry

Entry into ultra-low power static mode by the fabric master for the SmartFusion2 and IGLOO2 devices.

Flash*Freeze Exit

The exit sequence is performed by the system controller for Flash*Freeze mode exit.

Wakeup Mechanism

Exit from Flash*Freeze mode is initiated by the internal timed events or the external I/O events.