
Interrupting SmartFusion MSS Using GPIO and FABINT

Table of Contents

Introduction	1
Design Example Overview	1
Interrupt Generator Block Description	2
Interface Description	3
Software Implementation	3
Running the Design	5
Release Mode	6
Conclusion	6
Appendix A – Design Files	6
List of Changes	7

Introduction

The SmartFusion[®] customizable system-on-chip (cSoC) FPGA devices contain a hard embedded microcontroller subsystem (MSS), programmable analog circuitry, and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of a 100 MHz ARM[®] Cortex[™]-M3 processor, communications matrix, system registers, Ethernet MAC, DMA engine, real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), and fabric interface controller (FIC). The Cortex-M3 processor includes an interrupt controller called the nested vectored interrupt controller (NVIC).

There are 151 interrupts available in NVIC, of which 32 are general purpose interrupts they can be connected to the fabric or outside world. There is a dedicated fabric interrupt available, FABINT, that is connected to the FPGA. Refer to the *SmartFusion Microcontroller Subsystem User's Guide*, for more details on interrupts.

This application note explains how to use the general purpose input/output (GPIO) and FABINT to interrupt MSS from the FPGA fabric, and how to implement the interrupt handler on Cortex-M3. You should have a basic understanding of SmartFusion design flow. Refer to the *Using UART with SmartFusion* along with the *Libero SoC User's Guide* to understand the SmartFusion design flow.

Design Example Overview

This design example demonstrates using GPIO and FABINT on SmartFusion Development Kit Board and SmartFusion Evaluation Kit Board. The design example consists of an interrupt generator block which has two timer blocks. [Figure 1 on page 2](#) illustrates how to interface an interrupt generator with MSS to generate GPIO and FABINT interrupts. The universal asynchronous receiver/transmitter (UART) in MSS is used for printing interrupt messages on the HyperTerminal using the printf command.

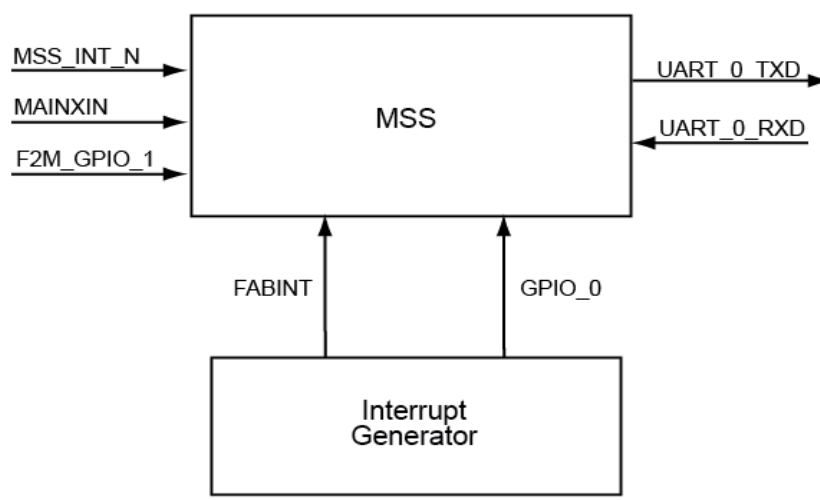


Figure 1 • Interfaces of Interrupt Generator Block with MSS

Interrupt Generator Block Description

The interrupt generator (Figure 2 on page 3) has two timer blocks. Two different timers are implemented in the FPGA fabric and run with the same clock source. The clock conditioning circuit (CCC) in the MSS generates a 12.5 MHz clock and acts as the clock source for the timers. This design example is implemented on the SmartFusion Evaluation Kit Board and SmartFusion Development Kit Board.

This design example uses 3 interrupts. Timer 1 generates an interrupt for every 76 milliseconds, while Timer 2 generates an interrupt for every 236 milliseconds. For every 3 interrupts of Timer 1, there is one interrupt from the Timer 2. The third interrupt is generated by the SW1 that is connected to FPGA I/O "G19" on both kits. An interrupt is generated asynchronously whenever the SW1 is pressed.

The Timer 1 block is connected to the FABINT of the MSS and the Timer 2 block is connected to the GPIO_0 of the MSS, which is configured to connect with the fabric. Figure 2 on page 3 shows the block diagram of the interrupt generator.

Whenever any of the interrupt occurs (the Timer 1 or, Timer 2 interrupt, or the switch interrupt), the processor executes the corresponding interrupt service routine and the source of the interrupt is printed on HyperTerminal.

This design example is available in both VHDL and Verilog (refer to "Appendix A – Design Files" on page 6).

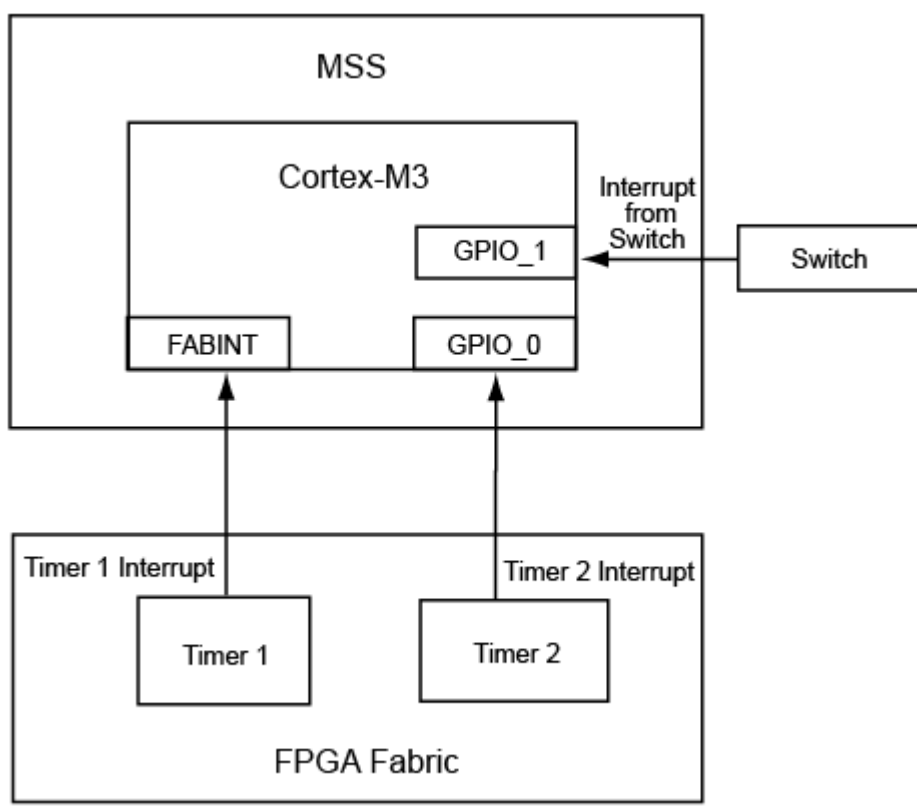


Figure 2 • Block Diagram of Interrupt Generator

Interface Description

Table 1 shows the top-level interface signal descriptions.

Table 1 • Interface Description

Signal	Direction	Description
MSS_RESET_N	Input	Active Low reset signal for the microcontroller subsystem.
MAINXIN	Input	Main crystal oscillator circuit. Input to the crystal oscillator circuit. Pin for connecting an external crystal, ceramic resonator, or RC network.
F2M_GPI_1	Input	Active Low external interrupt signal.
UART_0_TXD	Output	UART Transmit data.
UART_0_RXD	Input	UART Receive data.

Software Implementation

The software design consists of enabling the interrupts (GPIO_0, GPIO_1, and FABINT) and the implementation of interrupt handlers. This interrupt handler executes on the occurrence of interrupts and prints the source of interrupt on HyperTerminal. The following is a description of software Application Programming Interfaces (APIs).

Enabling the Fabric Interrupt

It enables the fabric interrupt, FABINT, in the NVIC interrupt controller by calling the following API:

```
NVIC_EnableIRQ(Fabric_IRQn);
```

Enabling and Configuring the GPIO Interrupts

The *MSS_GPIO_enable_irq()* function is used to enable the interrupt generation for the specified GPIO input. The interrupts are generated based on the state of the GPIO input and the interrupt mode configured for it by *MSS_GPIO_config()*. This is implemented by using the following API:

```
/* Enabling and Configuring GPIO 0, Positive edge sensitive */
```

```
NVIC_EnableIRQ(GPIO0_IRQn);  
MSS_GPIO_config( MSS_GPIO_0, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_POSITIVE );  
MSS_GPIO_enable_irq( MSS_GPIO_0 );
```

```
/* Enabling and Configuring GPIO 1, Negative edge sensitive */
```

```
NVIC_EnableIRQ(GPIO1_IRQn);  
MSS_GPIO_config( MSS_GPIO_1, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_NEGATIVE );  
MSS_GPIO_enable_irq( MSS_GPIO_1 );
```

Fabric Interrupt Handler

This interrupt handler executes on the occurrence of the fabric interrupt (Timer 1 interrupt) and prints the source of interrupt on HyperTerminal. This is done by calling the following API:

```
Void Fabric_IRQHandler( void )
```

GPIO_0 Interrupt Handler

This interrupt handler executes on the occurrence of the GPIO_0 interrupt (Timer 2 interrupt) and prints the source of interrupt on HyperTerminal. This is done by calling the following API:

```
Void GPIO0_IRQHandler( void )
```

GPIO_1 Interrupt Handler

This interrupt handler executes on the occurrence of the GPIO_1 interrupt (the switch interrupt) and prints the source of interrupt on HyperTerminal. This is done by calling the following API:

```
Void GPIO1_IRQHandler( void )
```

Clearing the Pending Interrupt

The *NVIC_ClearPendingIRQ()* function is used to clear the interrupt in the Cortex-M3 interrupt controller (NVIC). The following API is used to clear the fabric interrupt:

```
NVIC_ClearPendingIRQ( Fabric_IRQn );
```

The *MSS_GPIO_clear_irq()* function is used to clear a pending interrupt from GPIO_0 and GPIO_1. The following API is used to clear the GPIO interrupts:

```
MSS_GPIO_clear_irq( MSS_GPIO_0 );  
MSS_GPIO_clear_irq( MSS_GPIO_1 );
```

Note: The *MSS_GPIO_clear_irq()* function must be called as part of any GPIO interrupt service routine (ISR) to prevent the same interrupt event re-triggering a call to the GPIO ISR. This function also clears the interrupt in the Cortex-M3 interrupt controller through a call to the *NVIC_ClearPendingIRQ()*.

Enabling the printf Command Through UART

The printf statement is used for printing messages through UART on HyperTerminal. To enable this feature, add the symbol ACTEL_STUDIO_THRU_UART in the SoftConsole project properties.

Figure 3 shows the SoftConsole project properties window and can be referred to while adding the symbol for enabling printf.

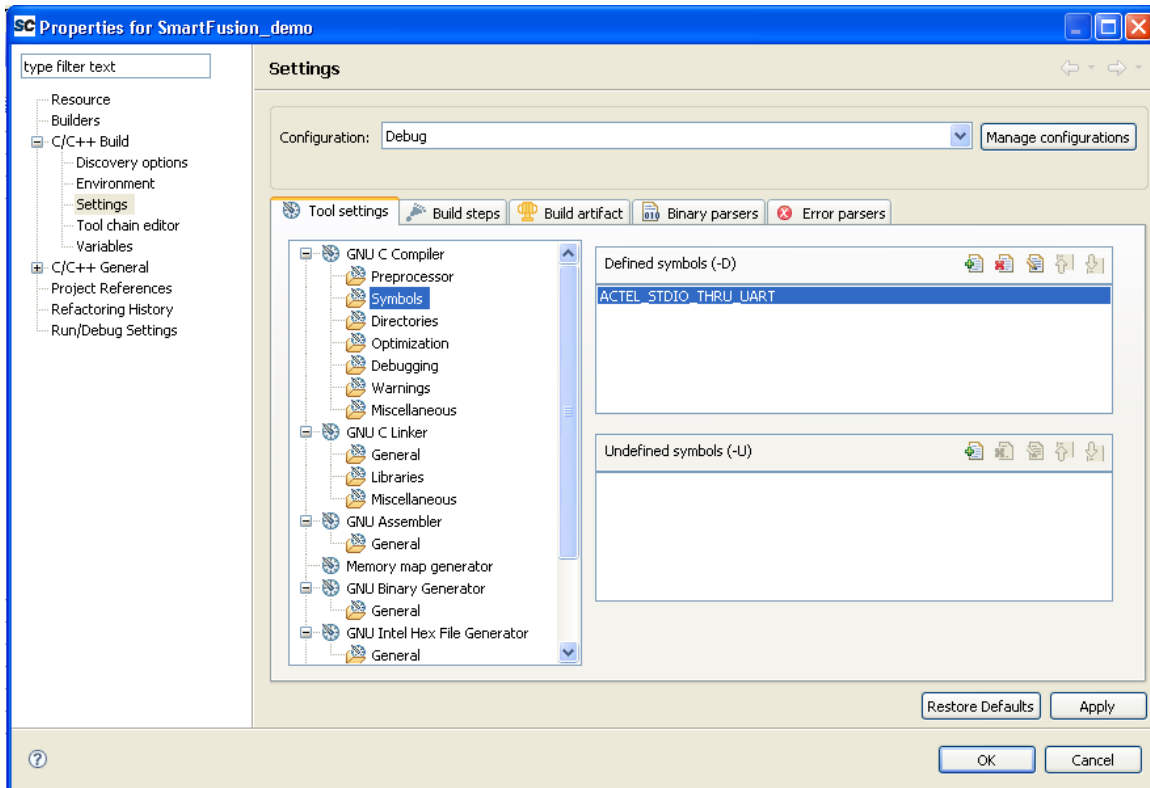


Figure 3 • SoftConsole Project Properties

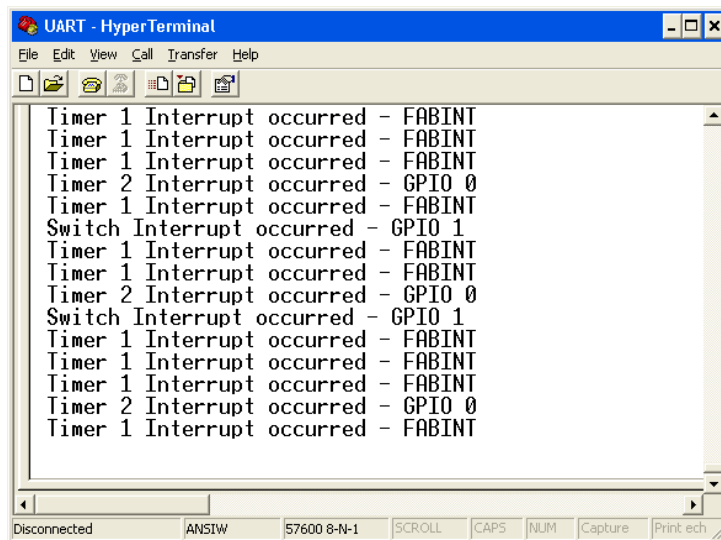
Running the Design

Program the SmartFusion Evaluation Kit Board or the Development Kit Board with the generate or provided *.stp file (refer to "Appendix A – Design Files" on page 6) using FlashPro, and then power cycle the board.

Invoke the SoftConsole IDE by clicking on the **Write Application Code** under Develop Firmware in Libero SoC (refer to "Appendix A – Design Files" on page 6) and launch the debugger. Start a HyperTerminal with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control.

If your PC does not have a HyperTerminal program, use any free serial terminal emulation program like PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs](#) tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the HyperTerminal window prints the messages as interrupts. Figure 4 displays a HyperTerminal with interrupt messages.



```
UART - HyperTerminal
File Edit View Call Transfer Help
Timer 1 Interrupt occurred - FABINT
Timer 1 Interrupt occurred - FABINT
Timer 1 Interrupt occurred - FABINT
Timer 2 Interrupt occurred - GPIO 0
Timer 1 Interrupt occurred - FABINT
Switch Interrupt occurred - GPIO 1
Timer 1 Interrupt occurred - FABINT
Timer 1 Interrupt occurred - FABINT
Timer 2 Interrupt occurred - GPIO 0
Switch Interrupt occurred - GPIO 1
Timer 1 Interrupt occurred - FABINT
Timer 1 Interrupt occurred - FABINT
Timer 1 Interrupt occurred - FABINT
Timer 2 Interrupt occurred - GPIO 0
Timer 1 Interrupt occurred - FABINT
Disconnected ANSIW 57600 8-N-1 SCROLL CAPS NUM Capture Print ech
```

Figure 4 • HyperTerminal Display

Release Mode

The release mode programming file (STAPL) is provided. If not, refer to "Appendix A – Design Files" for more information on downloading the programming files. Refer to the Readme.txt file included in the programming zip file for more information.

Refer to the [Building Executable Image in Release Mode and Loading into eNVM](#) tutorial for more information on building an application in release mode.

Conclusion

The design example demonstrated the usage of GPIO and FABINT to interrupt the MSS from the FPGA fabric. This application note explains the implementation of various interrupts and the interrupts handler.

Appendix A – Design Files

You can download the design files from the Microsemi SoC Products Group website:

www.microsemi.com/soc/download/rsc/?f=A2F_AC338_DF.

The design file consists of Libero Verilog and VHDL projects, SoftConsole software project, and programming files (*.stp) for A2F500-DEV-KIT and A2F-EVAL-KIT. Refer to the Readme.txt file included in the design file for the directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC338_PF.

The programming zip file consists of STAPL programming file (*.stp) for A2F500-DEV-KIT, A2F-EVAL-KIT, and a Readme.txt file.

List of Changes

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

Revision*	Changes	Page
Revision 2 (January 2012)	Updated "Running the Design" for Libero v10.0 (SAR 35782).	5
	Updated "Appendix A – Design Files" for Libero v10.0 (SAR 35782).	6
Revision 1 (August 2010)	Modified the section "Running the Design" (SAR 27465).	5
	Removed Figure 4 • HyperTerminal Settings on page 6 (SAR 27465).	
	Modified the section "Appendix A – Design Files" (SAR 27465).	6
	Removed Table 2 • Design Files Description (SAR 27465).	

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



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

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at www.microsemi.com.

© 2012 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.