
Interrupting SmartFusion MSS Using FABINT

Table of Contents

Introduction	1
Design Example Overview	1
Interrupt Generator Block Description	2
Interface Description	4
Hardware Implementation	4
Software Implementation	4
Running the Design	6
Release Mode	7
Conclusion	7
Appendix A – Design Files	7
List of Changes	8

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 that can be connected to the fabric or outside world. There is a dedicated fabric interrupt available, FABINT, that is connected to the FPGA. The FABINT Interrupt signal is sourced by user logic to the NVIC on the Cortex-M3 processor, which is INTISR [31]. Refer to the [SmartFusion Microcontroller Subsystem User's Guide](#) for more details on interrupts.

This application note explains how to use FABINT to interrupt MSS from the FPGA fabric, and how to implement the interrupt handler on Cortex-M3 for FABINT.

Design Example Overview

This design example demonstrates using FABINT on SmartFusion Development Kit Board and SmartFusion Evaluation Kit Board. The design example consists of an interrupt generator block and an advanced peripheral bus interface (CoreAPB3 IP) implemented in the FPGA Fabric. The CoreAPB3 connects the interrupt generator with MSS. The interrupt generator has two timer blocks, pulse detector block, CoreGPIO IP and CoreInterrupt IP.

[Figure 1 on page 2](#) illustrates how to interface the interrupt generator with MSS to generate FABINT. The UART in MSS is used for printing Interrupt messages on PuTTY using the printf command.

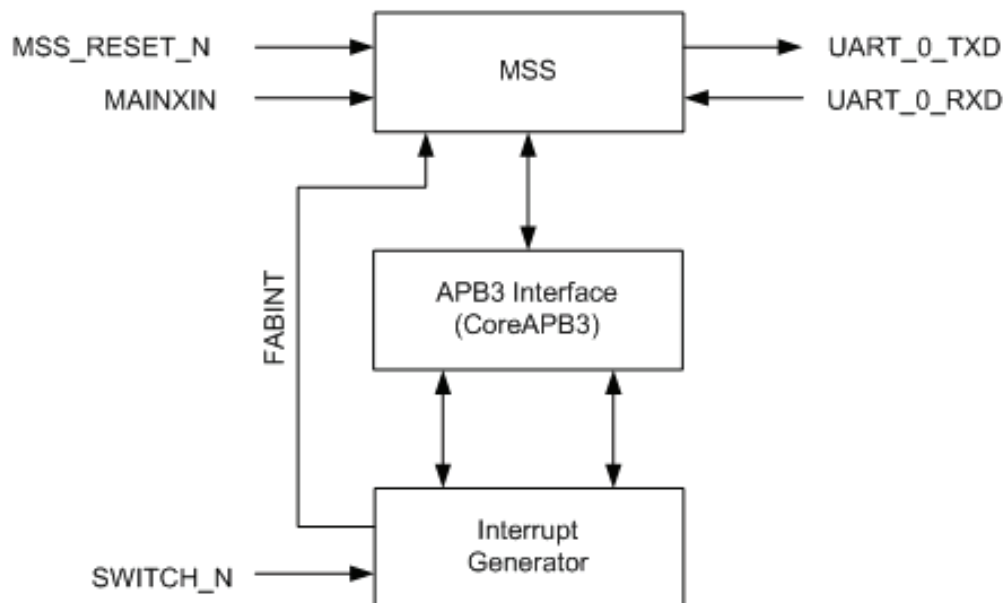


Figure 1 • Interfaces of Interrupt Generator Block with MSS

Interrupt Generator Block Description

The interrupt generator (Figure 2 on page 3) has two timer blocks, a pulse detector block, CoreGPIO IP, and CoreInterrupt IP. The two different timers are implemented in fabric and run with the same clock source. The clock conditioning circuit (CCC) in the MSS generates 25 MHz clock and acts as the clock source for the timers. This design example is implemented in the SmartFusion Evaluation Kit Board and the SmartFusion Development Kit Board.

This design example uses three interrupts. Timer 1 generates an interrupt for every 80 milliseconds, while Timer 2 generates an interrupt for every 400 milliseconds. For every five interrupts of Timer 1, there is one interrupt from the Timer 2. The third interrupt is generated by a pulse detector that detects a rising edge on pin "G19" driven by the SW1 on both kits. An interrupt is generated asynchronously whenever the SW1 is pressed. Refer to the [SmartFusion Development Kit User's Guide](#) and the [SmartFusion Evaluation Kit User's Guide](#), for more details on the development and evaluation kit boards.

All three interrupts are positive edge triggered interrupts and are connected to the CoreGPIO. This CoreGPIO is used to hold the Timer 1, Timer 2, and switch interrupts until Cortex-M3 processor reads the interrupt. The CoreGPIO provides an advanced peripheral bus (APB) register-based interface and supports 32 general purpose inputs and outputs (GPIOs). Refer to the [CoreGPIO Hand Book](#) for more details on CoreGPIO IP.

The CoreGPIO provides the interrupt signals to the interrupt controller (CoreInterrupt). The CoreInterrupt generates an interrupt that controls the MSS through FABINT. This interrupt signal is connected to the FABINT of MSS that interrupts the Cortex-M3 processor. The status register in the CoreInterrupt is updated during the occurrence of any interrupt (Timer 1, Timer 2, or switch interrupt) or any of the combinations. The processor can read the status register of CoreInterrupt and decides the source of the interrupts. The status register of the CoreGPIO and CoreInterrupt can be set or cleared by the MSS through the fabric interface controller (FIC). Refer to the [CoreInterrupt Datasheet](#) for more details on CoreInterrupt IP.

The CoreAPB3 acts as a bridge between the MSS and the interrupt generator block. It provides an advanced microcontroller bus architecture (AMBA3) APB3 fabric supporting up to 16 APB slaves. This design example uses two slave slots (Slot 0 and Slot 1) to interface with the CoreGPIO and CoreInterrupt and is configured with direct addressing mode. Refer to the [CoreAPB3 Hand Book](#) for more details on CoreAPB3 IP.

Figure 2 shows the block diagram of the interrupt generator along with the APB3 interface.

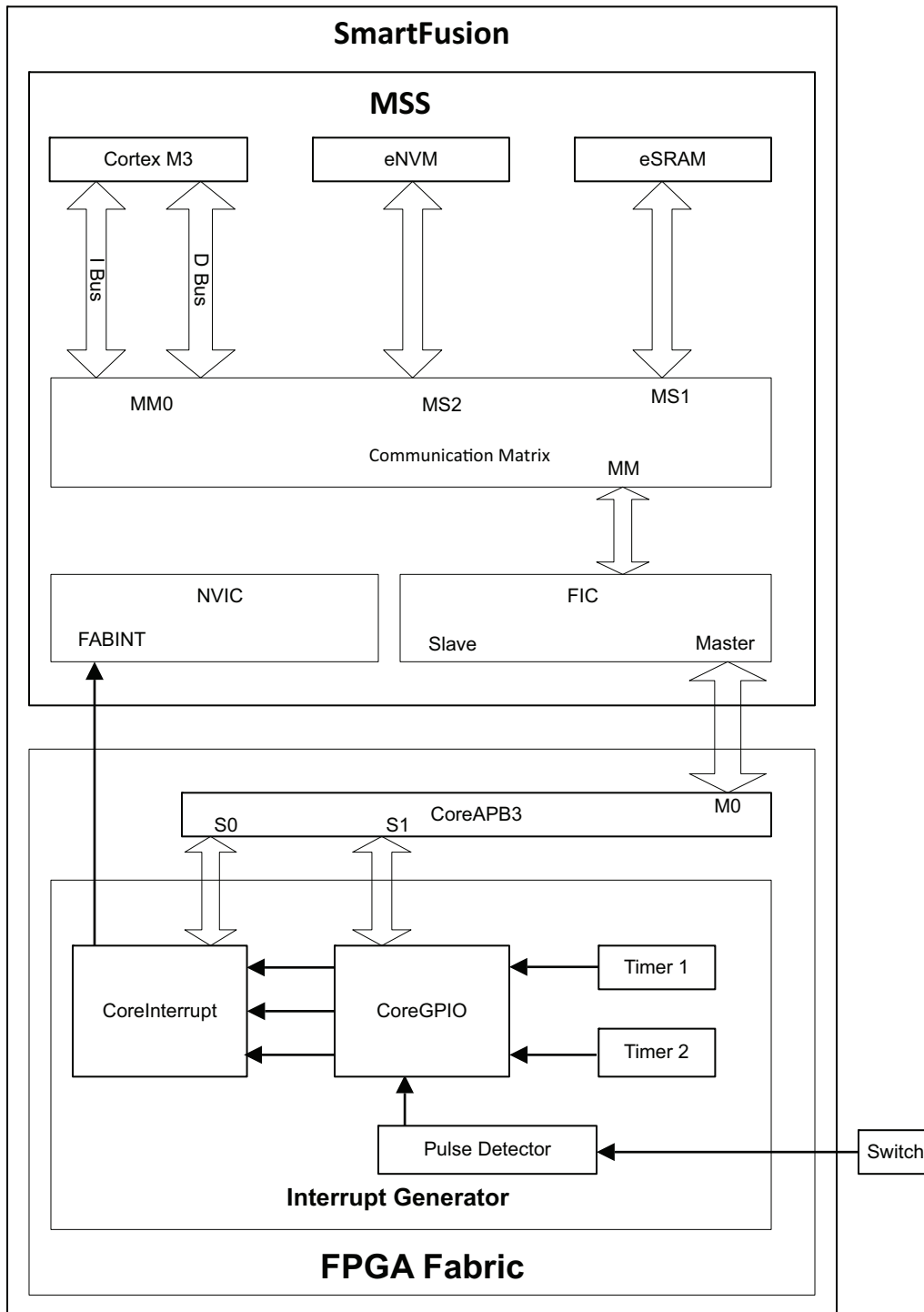


Figure 2 • Block Diagram of Interrupt Generator

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
SWITCH_N	Input	Active Low external interrupt signal
UART_0_TXD	Output	UART Transmit data
UART_0_RXD	Input	UART Receive data

Hardware Implementation

The design example is available in both VHDL and Verilog (refer to "Appendix A – Design Files" on page 7). To understand the interfacing of custom logic to MSS, refer to the *Connecting User Logic to the SmartFusion Microcontroller Subsystem* application note.

Software Implementation

The software design includes enabling the interrupt handler in Fabric. The MSS IRQ handler reads the status register of CoreInterrupt and prints the source of interrupt on PuTTY. Following is a description of software API's.

Enabling the Fabric Interrupt

To enable the fabric interrupt FABINT of Cortex-M3 NVIC call the following API:

```
NVIC_EnableIRQ(Fabric_IRQn);
```

Enabling Interrupts in the CoreInterrupt of Fabric

Timer 1, Timer 2, and switch interrupts are enabled in the CoreInterrupt of Fabric. The following code describes how to enable these interrupts, while Table 2 gives the description for CoreInterrupt Interrupt enable register.

```
((uint32_t volatile *) (FABRIC_INT_CTRL_BASE_ADDR + FABRIC_INTR_ENBL_OFF_ADDR)) = FABRIC_INTR_ENBL;
```

The base address of CoreInterrupt is 0x40050000 and offset address of Interrupt enable register is 0x20.

Table 2 • CoreInterrupt Interrupt Enable Register Description

Bit	Value	Description
[0]	1	This bit enables Timer 1 interrupt
[1]	1	This bit enables Timer 2 interrupt
[2]	1	This bit enables the Switch interrupt
[31:3]	1	Reserved

Fabric IRQ Handler

Fabric IRQ handler API makes decisions for the source of interrupt by reading the status register of fabric CoreInterrupt and printing it on HyperTerminal. The status register offset address is 0x2C.

Table 3 shows the status register descriptions. Then it clears the pending IRQ of Cortex-M3 processor and Interrupts in the CoreGPIO block.

The base address of CoreGPIO is 0x40050100 and offset address of Interrupt clear register is 0x80.

Table 3 • CoreInterrupt Status Register Description

Bit	Value	Description
[0]	1	Value 1 represents the occurrence of Timer 1 interrupt
[1]	1	Value 1 represents the occurrence of Timer 2 interrupt
[2]	1	Value 1 represents the occurrence of Switch interrupt
[31:3]	0	Reserved

Table 4 shows CoreGPIO Interrupt clear register description.

Table 4 • CoreGPIO Interrupt Clear Register Description

Bit	Value	Description
[0]	1	Value 1 clears Timer 1 interrupt
[1]	1	Value 1 clears Timer 2 interrupt
[2]	1	Value 1 clears Switch interrupt
[31:3]	1	Reserved

This is done by calling the following API:

```
void Fabric_IRQHandler( void )
```

Clear Pending IRQ

The following API clears the pending IRQ of Cortex-M3 processor:

```
NVIC_ClearPendingIRQ( Fabric_IRQn );
```

Enabling the printf Command Through UART

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

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

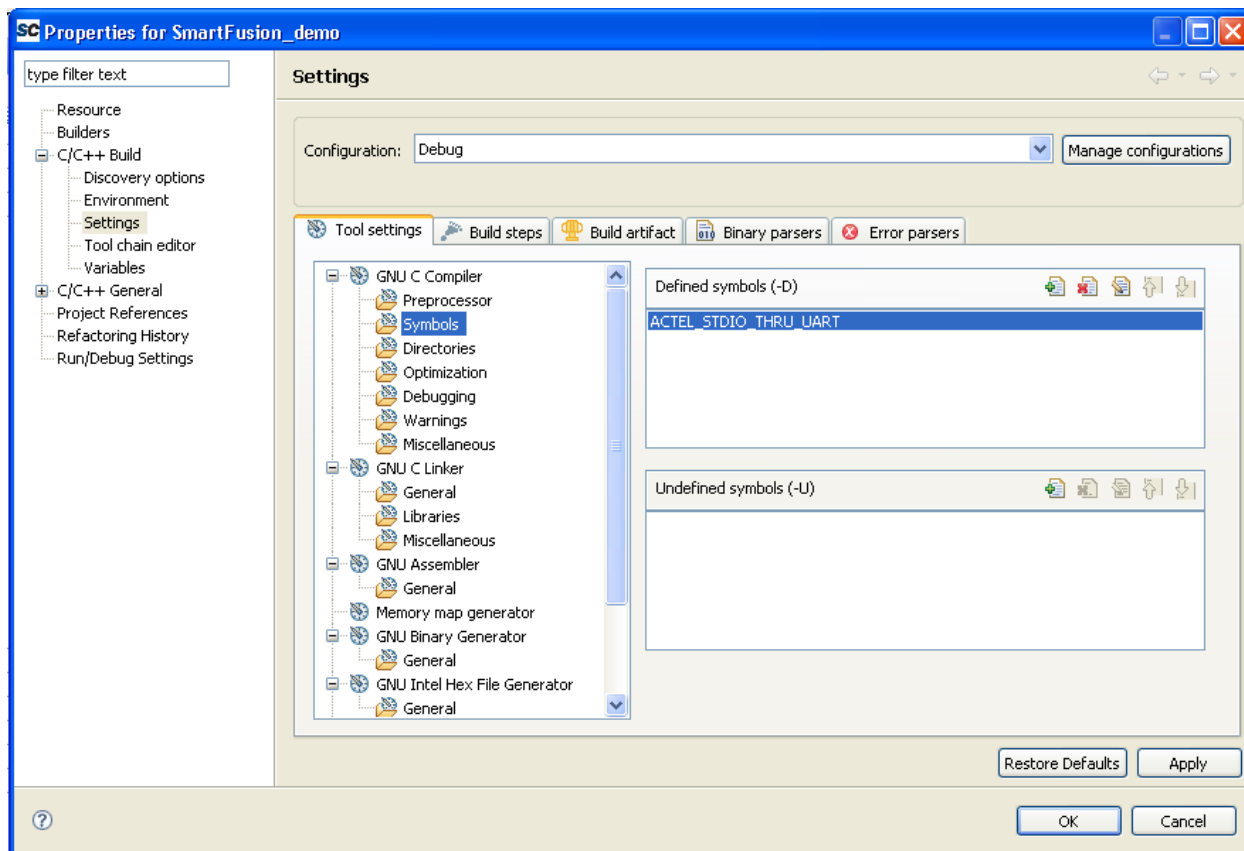


Figure 3 • SoftConsole Project Properties

Running the Design

Board Settings

The design example works on the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board with default board settings. Refer to the following user's guides for default board settings:

- [SmartFusion Development Kit User's Guide](#)
- [SmartFusion Evaluation Kit User's Guide](#)

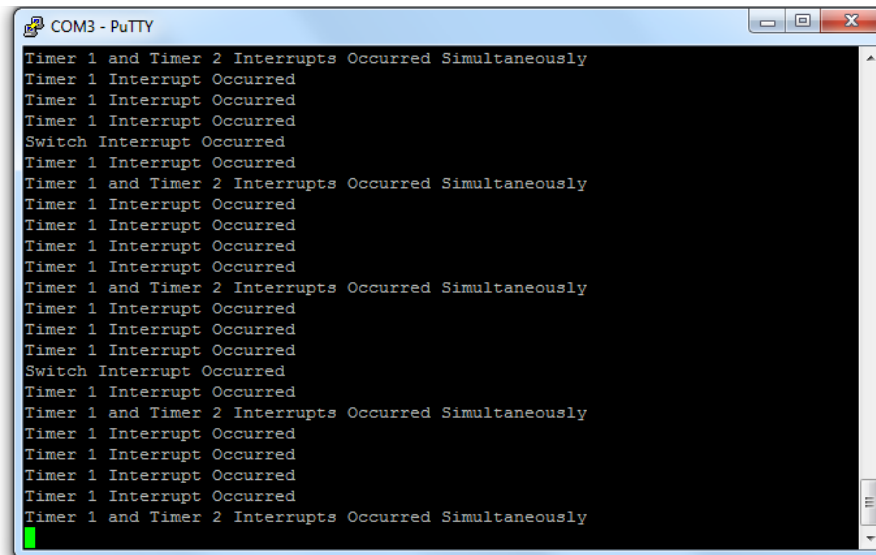
Program the Design and Running the Application

Program the SmartFusion Evaluation Kit Board or the Development Kit Board with the generated or provided *.stp file (refer to "Appendix A – Design Files" on page 7) 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 7) and launch the debugger. Start a PuTTY with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control.

If your PC does not have a PuTTY program, use any free serial terminal emulation program like HyperTerminal 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 PuTTY window prints the messages as interrupts, as shown in Figure 4.



```
COM3 - PuTTY
Timer 1 and Timer 2 Interrupts Occurred Simultaneously
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Switch Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 and Timer 2 Interrupts Occurred Simultaneously
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 and Timer 2 Interrupts Occurred Simultaneously
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Switch Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 and Timer 2 Interrupts Occurred Simultaneously
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 Interrupt Occurred
Timer 1 and Timer 2 Interrupts Occurred Simultaneously
```

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 *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 FABINT to Interrupt the MSS from FPGA fabric. This application note explains the implementation of various interrupts using the CoreGPIO IP and CoreInterrupt IP. The application note also demonstrates the interfacing of the interrupt generator with MSS via CoreAPB3 IP.

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_AC339_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_AC339_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 4 (January 2013)	Added "Board Settings" section and modified "Running the Design" section (SAR 43469).	6
Revision 3 (February 2012)	Removed ".zip" extension in the links (SAR 36763).	7
Revision 2 (January 2012)	Updated "Running the Design" for Libero v10.0 (SAR 35784).	6
	Updated "Appendix A – Design Files" for Libero v10.0 (SAR 35784).	7
	Changed Figure 4 for Libero v10.0 (SAR 35784).	7
Revision 1 (August 2010)	Modified the section "Running the Design" (SAR 27469).	6
	Removed Figure 4 • HyperTerminal Settings on page 7 (SAR 27469).	
	Modified the section "Appendix A – Design Files" (SAR 27469).	7
	Removed Table 2 • Design Files Description (SAR 27469).	

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.

© 2013 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.