Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

Implementing Pin Specific Interrupts in enCoRe™ II / enCoRe II LV

CE58786

Code Example Name: Example_enCoReII_GPIO
Programming Language: C
Associated Part Families: CY7C638xx, CY7C63310, CY7C601xx, CY7C602xx
Software Version: PSoC® Designer™ 5.3
Related Hardware: CY3655 Development kit
Author: Anandhakumar C

Code Example Objective

This is a basic example that demonstrates how to use the pin specific GPIO interrupts in enCoRe™ II/enCoRe II LV using the CY3655 development kit

Overview

This Code Example demonstrates how to use the dedicated pin GPIO interrupt INT0. When a switch connected to the port pin corresponding to INT0 (P0.2) is pressed, an LED connected to P1.3 glows. This Code Example was developed for CY7C60123-PVXC.

Pre-Requisites

enCoRe II / enCoRe II Low Voltage datasheet

System Requirements

Software: PSoC Designer 5.3, PSoC Programmer 3.17

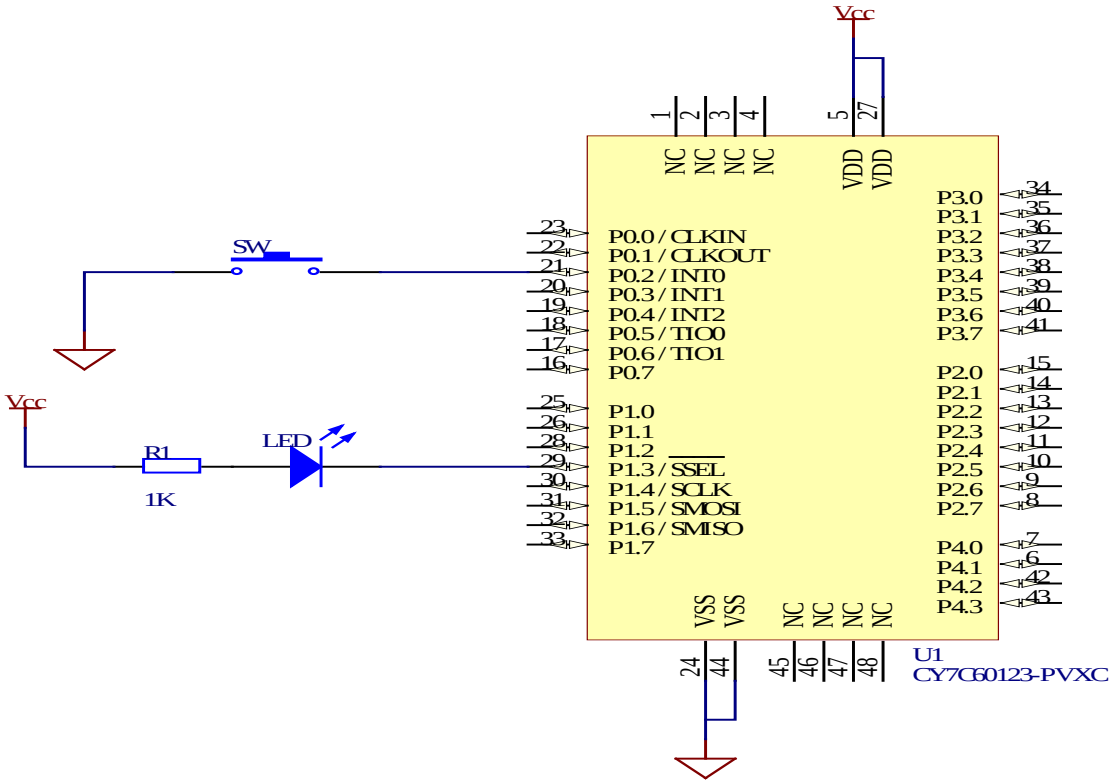
Hardware: CY3655 Development kit

Global Resources

Important Global Resources		
Parameter	**Value**	**Comments**
CPU Clock	Internal (24 MHz)	Internal 24 MHz oscillator is used
Watchdog Enable	Disable	Watchdog is disabled

Hardware Connections

The schematic diagram of the Code Example is shown.



The Code Example is tested using the CY3655 enCoRe II development kit. The Applications Board in this kit is used to develop applications for the enCoRe II low speed USB peripheral controller or Wireless enCoRe II microcontroller. For more details on using the kit, refer the CY3655 Development Kit Start Up Guide and CY3655 Hardware User Guide.

The following connections are made on the CY3655 board:

Place the jumper J11 for the push button S1.

Place the jumper J19 for the LED D3.

Testing the Code Example

In this Code Example, a pull down switch is connected to P0[2] and an active low LED to P1[3]. In the firmware, the INT0 interrupt is enabled by configuring the P0[2] pin as an input with the pull-up enabled and detecting a low level when the button is pressed. When the switch is pressed, the INT0 interrupt is raised and in the ISR for INT0, LED connected to P1[3] is made to glow.

Following are the steps to test the Code Example:

Download the hex file from the Code Example accompanying this document into the CY3655 kit using PSoC Programmer

Press Switch S1. The LED D3 will glow.

Device Details

The enCoRe II LV is a low voltage, low cost 8-bit Flash programmable microcontroller. The enCoRe II LV features up to 36 general-purpose I/O (GPIO) pins. The I/O pins are grouped into five ports (Port 0 to 4).The pins on Port 0 and Port 1 are each configured individually while the pins on Ports 2, 3, and 4 are only configured as a group. Each GPIO port supports high impedance inputs, configurable pull up, open drain output, CMOS/TTL inputs, and CMOS output with up to five pins that support programmable drive strength of up to 50 mA sink current. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Each GPIO port has its own GPIO interrupt vector with the exception of GPIO Port 0. GPIO Port 0 has in addition to the port interrupt vector, three dedicated pins that have independent interrupt vectors (P0.2–P0.4).

Beside their use as the P0.4–P0.2 GPIOs, these pins are also used as the Interrupt pins (INT0–INT2). To configure the P0.4–P0.2 pins, refer to the P0.2/INT0–P0.4/INT2 Configuration Registers. The P02CR–P04CR registers control the operation of pins P0.2–P0.4, respectively. These pins are shared between the P0.2–P0.4 GPIOs and the INT0–INT2. The INT0–INT2 interrupts are different from all other GPIO interrupts. These pins are connected directly to the interrupt controller to provide three edge-sensitive interrupts with independent interrupt vectors. These pins are enabled as interrupt sources in the interrupt controller registers (INT_MSKx).

By default all GPIOs are configured as inputs. To prevent the inputs from floating, the pull-up resistors are enabled. Firmware must configure each of the GPIOs before use.

To use these pins as interrupt inputs, configure them as inputs by clearing the corresponding Output Enable. If the INT0–INT2 pins are configured as outputs with interrupts enabled, the firmware can generate an interrupt by writing the appropriate value to the P0.2, P0.3, and P0.4 data bits in the P0 Data Register. Regardless of whether the pins are used as Interrupt or GPIO pins, the Int Enable, Int act Low, TTL Threshold, Open Drain, and Pull-up Enable bits control the behavior of the pin. The P0.2/INT0–P0.4/INT2 pins are individually configured with the P02CR (0x07), P03CR (0x08), and P04CR (0x09) respectively.

The interrupt controller and its associated registers allow your code to respond to an interrupt from almost every functional block in the enCoRe II LV devices. The registers associated with the interrupt controller can be disabled either globally or individually. The registers also provide a mechanism by which a user may clear all pending and posted interrupts, or clear individual posted or pending interrupts.

An interrupt is posted when its interrupt conditions occur. It remains posted until the interrupt is taken or it is cleared by writing to the appropriate INT_CLRx register. A posted interrupt is not pending unless it is enabled by setting its interrupt mask bit (in the appropriate INT_MSKx register). All pending interrupts are processed by the Priority Encoder to determine the highest priority interrupt which will be taken by the M8C if the Global Interrupt Enable bit is set in the CPU_F register.

The Interrupt Mask Registers (INT_MSKx) are used to enable the individual interrupt sources to create pending interrupts. If an INT_MSKx bit is set, the interrupt source associated with that mask bit can generate an interrupt that becomes a pending interrupt.

The Interrupt Clear Registers (INT_CLRx) are used to enable the individual interrupt sources to clear posted interrupts.

In the code, the INT0 interrupt is enabled by configuring the P0[2] pin as an input with the pull-up enabled and detecting a low level when the button is pressed. Three pushbuttons and three LEDs are available on the CY3655 board for general purpose use. The following table shows the enCoRe II signal name relation to the pushbutton or LED and its jumper on the CY3655 board.

Pushbutton or LED	Jumper	enCoRe II Signal Name
D1 LED	J12	P0.5
D2 LED	J17	P0.6
D3 LED	J19	P1.3
S1 Pushbutton	J11	P0.2
S2 Pushbutton	J16	P0.3
S3 Pushbutton	J18	P0.4

In this Code Example, the P0[2] is used as the switch button input and P1[3] as the LED output. Hence make sure you place the jumpers J11 and J19 on the board. Note that the LEDs D1, D2, and D3 are active low LEDs. Driving the enCoRe pin low illuminates the LEDs.

Following are the pin configurations for the port pins P0[2] and P1[3]. These are set in the pinout window in PSoC Designer.

Port pin	Name	Select	Drive	Interrupt
P0[2]	S1	Input-CMOS	Open Drain, Pullup	FallingEdge
P1[3]	D3	Output-8 mA	Open Drain	DisableInt

The port pin P1[3] is accessed using the P1DATA register. This register contains the data for Port 1. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 1 pins. Similarly, the port pin P0[2] is accessed using the P0DATA register.

For more details on the drive modes and pin configuration registers, refer to the enCoRe II datasheet.

Operation

On reset, all hardware settings from the device configuration are loaded into the device and *main.c* is executed.

The following operations are performed in *main.c*.

Global interrupts are enabled

INT0 interrupt is set in the INT_MSK0 register

When the INT0 interrupt is received (S1 is pressed), ISR() is executed.

The *boot.asm* file contains M8C boot code for the CY7C60100 microcontroller family. This file also contains the Interrupt Service Routines for enCoRe II GPIO interrupts: INT0, INT1, INT2, and the GPIO Port interrupts for port 0, port 1, port 2, port 3, and port 4.

Notes

The Device Editor in PSoC Designer™ uses a template file, BOOT.TPL, located in the Code Example's root directory to create BOOT.ASM. Any changes to BOOT.ASM are overwritten every time the Code Example is generated; therefore, changes should be made to BOOT.TPL not BOOT.ASM. When modifying BOOT.TPL make sure that replacement strings (such as @CODE EXAMPLE_NAME) are not accidentally modified.

The following code is added in the boot.tpl file to route the INT0 ISR to the custom ISR written in *main.c.*

```
;-------------------------------------------------------------------------
;  FUNCTION NAME: INT0_ISR
;
;  DESCRIPTION:   This is the ISR for the the INT0 GPIO interrupt
;
;-------------------------------------------------------------------------
INT0_ISR:
   ;@PSoC_UserCode_BODY_1@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------
   ;   NOTE: interrupt service routines must preserve
   ;   the values of the A and X CPU registers.
   lcall _ISR
   ;---------------------------------------------------
   ; Insert your custom code above this banner
   ;---------------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)
   RETI
;-------------------------------------------------------------------------
```

Upgrade Information

When upgrading the Code Example to newer versions of PSoC Designer, the changes in the boot.tpl file do not reflect in the newly generated Code Example. Code Example Update updates boot.tpl (the template for boot.asm). If you have modified boot.tpl in the previous version of PSoC Designer, Code Example Update places your custom file in the Backup folder of your Code Example directory. You can use this file for reference to manually modify the new boot.tpl installed by Code Example Update. All custom Interrupt Service Routine calls (such as a GPIO handler) must be manually copied from the backup boot.tpl to the new boot.tpl.

Document History

Document Title: Implementing Pin Specific Interrupts in enCoRe™ II / enCoRe II LV – CE58786

Document Number: 001-58786

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2858725	ANTG	01/19/2010	New example project
*A	3176055	ANTG	02/17/2011	Replaced 'example project' with 'code example' and updated the title. Upgraded to PD5.1 SP1. Added Pre-Requisites. Added System Requirements. Added Testing the Code Example.
*B	3918260	ANKC	03/01/2013	Updated Software Version as "PSoC® Designer™ 5.3". Updated Software as "PSoC Designer 5.3, PSoC Programmer 3.17" under System Requirements.

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," and PSoC Designer, and enCoRe are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/

© Cypress Semiconductor Corporation, 2010-2013. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.