CE220060 – PSoC 6 MCU Watchdog Timer

Objective

This example demonstrates the Watchdog Timer (WDT) on the PSoC® 6 MCU, using ModusToolbox™ IDE.

Requirements

Tool: ModusToolbox™ IDE 1.0

Programming Language: C

Associated Parts: All PSoC 6 MCU parts

Related Hardware: PSoC 6 BLE Pioneer Kit, PSoC 6 WiFi-BT Pioneer Kit, PSoC 6 Wi-Fi BT Prototyping Kit

Overview

This example explains the two use cases of WDT – as a watchdog that causes a device reset in the case of a malfunction, and as a periodic interrupt source.

A macro definition determines the mode to use. Out of the box, this example demonstrates the periodic interrupt. The red LED toggles on every interrupt at an interval of ~1 s.

For reset mode, you change the macro definition and enable an infinite loop in the `main()` function, to block the execution. The device resets in ~6 s. The red LED blinks twice after the device comes out of reset. If you use the reset mode without blocking the execution, the device does not reset. The red LED toggles every 1 s in the main loop to indicate that the CPU is in action.

In addition, the red LED blinks once for power cycling or an external reset event.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure the kit is configured correctly.

Note: The PSoC 6 BLE Pioneer kit and the PSoC 6 WiFi-BT Pioneer Kit ships with KitProg2. ModusToolbox only works with KitProg3. Before using this code example, make sure that the kit is upgraded to KitProg3. See ModusToolbox Help > ModusToolbox IDE Documentation > User Guide; section PSoC 6 MCU KitProg Firmware Loader. If you do not upgrade, you will see an error like "unable to find CMSIS-DAP device" or "KitProg firmware is out of date".

Software Setup

See the Operation section for information on how to modify the code for each demo.

Operation

1. Connect the Pioneer kit to your PC using the provided USB cable.
2. Import the application project into a new workspace. If you aren't familiar with this process, see KBA225201.
3. Modify the source code if you want to use RESET mode. By default, the example is in INTERRUPT mode.
   a. Open *main.c* from **CE220060_PSoC6_WatchdogTimer_mainapp** in the workspace.
   b. Set `WDT_DEMO to WDT_RESET_DEMO` for reset demonstration.
   c. In the innermost `for` loop, uncomment the line of code `//while(1);` to cause the reset to happen.
4. When `WDT_INTERRUPT_DEMO` is selected, you can optionally put the device in Deep Sleep mode by uncommenting the function call `Cy_SysPm_CpuEnterDeepSleep()` in the main loop. The device wakes up from Deep Sleep mode on a WDT interrupt.

5. Program the PSoC 6 MCU device. In the project explorer, select the **mainapp** project. In the Quick Panel, scroll to the **Launches** section and click the **Program (KitProg3)** configuration.
6. Observe the status of LEDs based on different events summarized in Table 1.

Table 1. LED Status

Project Setting	LED Status
`WDT_DEMO` set to `WDT_INTERRUPT_DEMO`	Red LED toggles on every WDT interrupt (interval of 1 s).
`WDT_DEMO` set to `WDT_RESET_DEMO` with the blocking function	After approximately 6 s, the device resets and the red LED blinks twice within a second to indicate a WDT reset.
`WDT_DEMO` set to `WDT_RESET_DEMO` without the blocking function	Red LED toggles every 1 s to indicate that the CPU is in action.

Note that red LED blinks once on a power cycle or an external reset event.

Design and Implementation

The WDT in PSoC 6 MCU is a 16-bit timer and uses the Internal Low-Speed Oscillator (ILO) clock of 32 kHz.

The WDT is configured in 8 steps in the project as described below:

1. Unlock the WDT to enable configuration.
2. Set the ignore bits for the match resolution. In the project, it is set to '0'; that means full 16-bit resolution for the match count, which gives an interval of 2.048 s ($2^{16} \div 32$ kHz) for the match event.
3. Write the match value. The WDT can generate an interrupt (if enabled) when the WDT counter reaches the match count. The project configures the match count using the `WDT_MATCH_COUNT` macro. Note that the interrupt handler modifies the match count when `WDT_INTERRUPT_DEMO` is selected to generate a periodic interrupt[1] every 1 s. The match count, however, is set to zero in WDT_RESET_DEMO mode to generate match events at equal intervals of time[2]. The device resets after three WDT match events; that is, 2.048 s x 3 = 6.144 s; if the match event is not cleared. You can reduce the duration of the match event by introducing ignore bits in Step 2.
4. Clear the pending WDT interrupt.
5. Enable the ILO, which is the source for the WDT.
6. Enable the interrupt generation if `WDT_INTERRUPT_DEMO` mode is selected and assign the handler. Interrupt should be disabled when using the WDT_RESET_DEMO mode[2].
7. Enable WDT.
8. Lock the WDT to prevent inadvertent changes.

Notes:

(1) The WDT generates an interrupt on match. However, the counter is not reset on match. It continues to count across the full 16-bit resolution. For this reason, the match count is updated on every WDT interrupt when `WDT_INTERRUPT_DEMO` is selected.

(2) Interrupt should not be enabled when the reset mode of the WDT is being used. If interrupt is enabled, upon an interrupt, it needs to be cleared to avoid repeated execution of the WDT interrupt handler. This removes the actual purpose of the WDT. Thus, interrupt generation should never be enabled and the WDT match event should always be cleared in the main loop.

Settings

Table 2 lists the macros in this example, and how they are used in the project.

Table 2. Macros

Macro	Value	Purpose
`WDT_DEMO`	WDT_RESET_DEMO	In this mode of the WDT, interrupt generation is not enabled and the WDT match event is cleared in the main loop.
	WDT_INTERRUPT_DEMO (default)	In this mode of the WDT, interrupt generation is enabled and the match event is cleared in the interrupt handler. The match count is updated to get the next interrupt after the same interval.
`WDT_INTERRUPT_INTERVAL`	1000 (default)	Specifies the interrupt interval in milliseconds. In this case, it is set to 1000 millisecond. Change this value to get a different interrupt interval.

To simulate a malfunction, the main loop contains a blocking function (infinite `while` loop). Enabling this blocking function causes WDT match events not to be cleared. After three match events, the device resets. The firmware blinks the red LED twice when the device comes out of reset.

For pin usage and configuration, open the **Pins** tab of the *design.modus* file.

Reusing This Example

This example is configured for the CY8CKIT-062-BLE kit. To port the design to a different PSoC 6 MCU device, right-click the application project and choose **Change ModusToolbox Device**. If changing to a different kit, you may need to reassign pins.

Table 3. Device and Pin Mapping Across PSoC 6 MCU Kits

Kit Name	Device Used	LED
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P13[7]
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P13[7]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P13[7]

In some cases, a resource used by a code example (for example, an IP block) is not supported on another device. In that case, the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on which resources a device supports.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox application and PSoC Creator project
Code Examples	
Visit the Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
PSoC 6 MCU: PSoC 62 Datasheet	PSoC 6 MCU: PSoC 62 Architecture Technical Reference Manual (TRM)

Development Kits	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit	
Tool Documentation	
ModusToolbox IDE	The Cypress IDE for IoT designers

Cypress Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right device, and quickly and effectively integrate the device into your design.

For the PSoC 6 MCU devices, see KBA223067 in the Cypress community for a comprehensive list of PSoC 6 MCU resources.

Document History

Document Title: CE220060 – PSoC 6 MCU Watchdog Timer

Document Number: 002-25468

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6372119	RJVB	11/21/2018	New code example

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709