

Objective

This example explains how to setup a 32-bit free-running counter using two 16-bit counters of Multi-Counter Watchdog Timer (MCWDT) on PSoC® 6 MCU, using ModusToolbox™ IDE.

Requirements

Tool: [ModusToolbox™ IDE 1.1](#)

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 WiFi-BT Prototyping Kit](#)

Overview

There are many applications where there is a need to timestamp different events in a system for timekeeping or debugging. In this example, the time gap between successive switch press events is measured in seconds and displayed on a UART terminal.

Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that 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 If you do not upgrade, you will see an error like “unable to find CMSIS-DAP device” or “KitProg firmware is out of date”.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

This example uses a terminal emulator. Install one if you don't have one. The instructions use [Tera Term](#).

Operation

1. Connect the kit to your PC using the provided USB cable.
2. Open your terminal software and select the KitProg COM port. Set the other serial port parameters as follows:
 - a. Baud rate: 115200bps
 - b. Data: 8 bit
 - c. Parity: None
 - d. Stop: 1 bit
 - e. Flow control: None
3. Import the code example into a new workspace. See [KBA225201](#).
4. 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.
5. Press and release the switch SW2. Notice the time displayed in seconds.

Debugging

You can debug the example to step through the code. Use the **Debug (KitProg3)** configuration. See [KBA224621](#) to learn how to start a debug session with ModusToolbox IDE.

Design and Implementation

MCWDT has two 16-bit counters (Counter 0 and Counter 1) and one 32-bit counter (Counter 2). In the application, Counter 0 and Counter 1 are configured in free-running mode. Counter 0 is clocked by LFCLK and Counter 1 is clocked from the Counter 0 cascade. LFCLK source is set to ILO (nominal 32 kHz). The combined counter counts from 0 to 0xFFFFFFFF, which is equivalent to 134217 s (~1.5 day). The 32-bit counter, Counter 2, is not used.

A switch is used to mark the start and end points of MCWDT counting. Debounce logic is implemented in firmware to avoid false press events. The counter value is stored for each switch press. The time interval between two switch presses is evaluated in seconds and displayed on UART terminal.

In the case of failure in the initialization of MCWDT or UART, the red LED is turned ON.

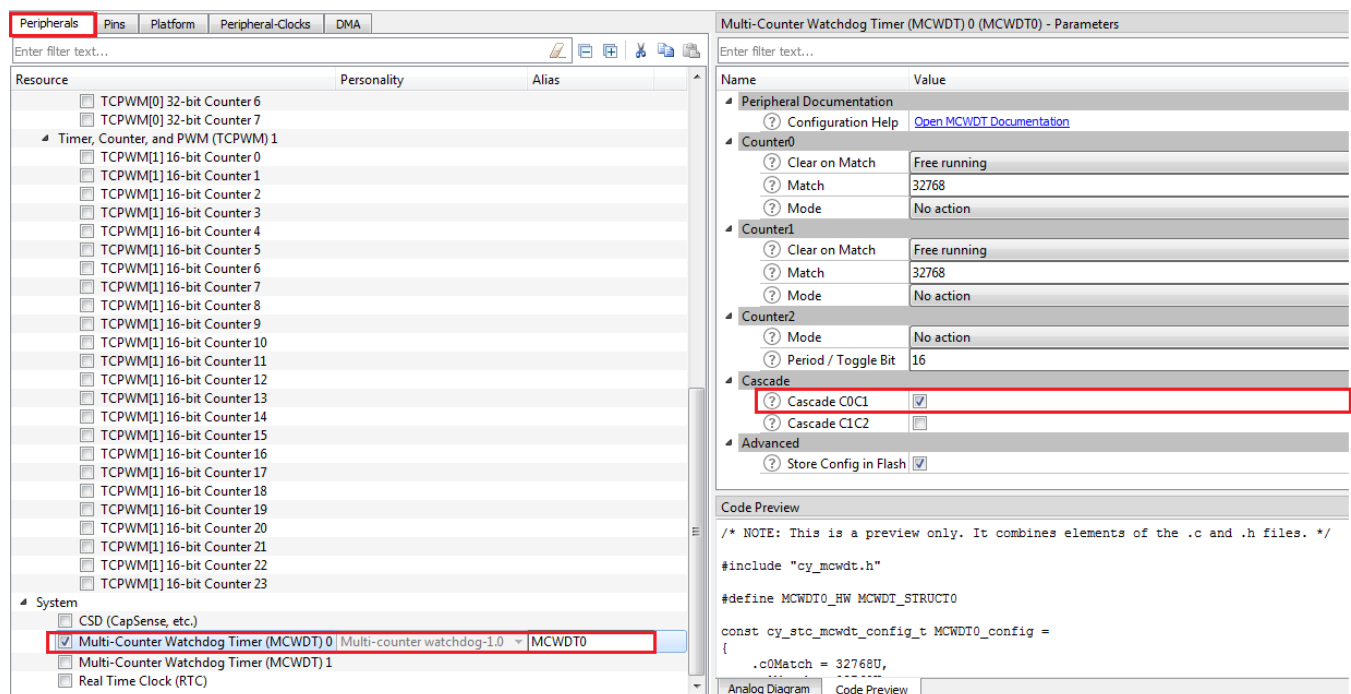
Resources and Settings

[Table 1](#) lists some of the ModusToolbox resources used in the example, and how they are used in the design. The *design.modus* file contains all the configuration settings. For example, for pin usage and configuration, open the **Pins** tab of the design file.

Table 1. ModusToolbox Resources

Resource	Alias	Purpose	Non-default Settings
Multi-Counter Watchdog Timer (MCWDT) 0	MCWDT0	Used to measure time between switch press events.	Counter 0 and counter 1 cascade enabled. See Figure 1 .
UART	KIT_UART	Used to communicate the measured time.	Clock source and TX pin are set. See Figure 2 .
Peripheral clock	KIT_UART_CLOCK	Clock input to the UART block.	Clock divider value and peripheral are set. See Figure 3 .

Figure 1. MCWDT Configuration



The screenshot shows the ModusToolbox IDE interface. On the left, the 'Peripherals' tab is active, displaying a list of resources. The 'Multi-Counter Watchdog Timer (MCWDT) 0' resource is selected and highlighted with a red box. On the right, the 'Multi-Counter Watchdog Timer (MCWDT) 0 (MCWDT0) - Parameters' panel is open. It shows the configuration for Counter0, Counter1, Counter2, and Cascade. The 'Cascade C0C1' checkbox is checked, indicating that Counter 1 is cascaded to Counter 0. The 'Advanced' section shows the 'Store Config in Flash' checkbox is also checked. Below the parameters, a 'Code Preview' section shows the generated C code for the MCWDT configuration.

```

/* NOTE: This is a preview only. It combines elements of the .c and .h files. */
#include "cy_mwtdt.h"
#define MCWDT0_HW MCWDT0_STRUCT0
const cy_stc_mwtdt_config_t MCWDT0_config =
{
    .c0Match = 32768U,
    .c0Mode = FreeRunning,
    .c1Match = 32768U,
    .c1Mode = FreeRunning,
    .c2Match = 0U,
    .c2Mode = NoAction,
    .c0C1Cascade = 1,
    .c1C2Cascade = 0,
    .storeConfigInFlash = 1
};
  
```

Figure 2. UART Configuration

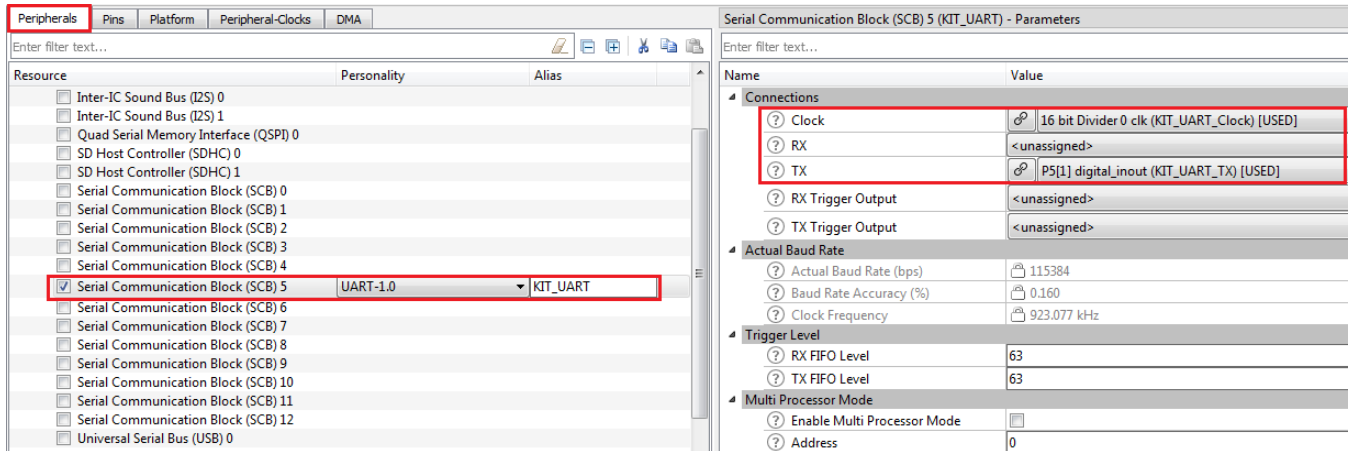
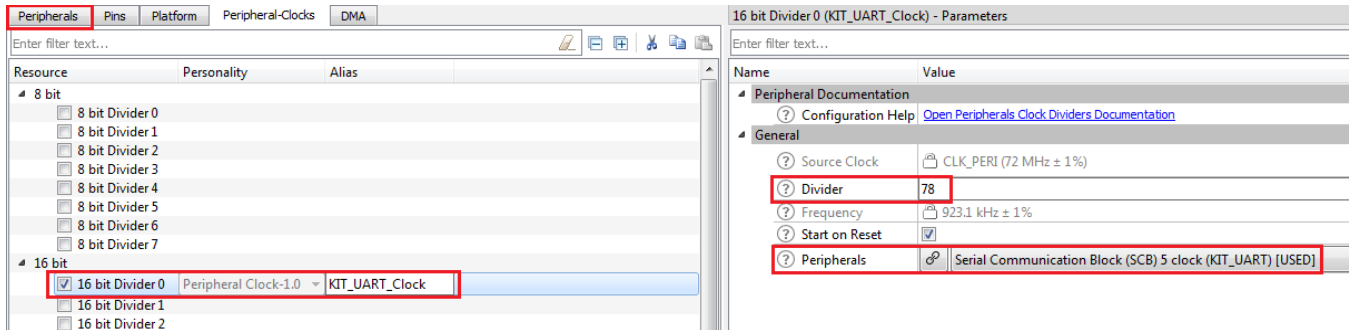


Figure 3. UART Clock Configuration



Reusing This Example

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

Table 2. Device and Pin Mapping across PSoC 6 MCU Kits

Kit Name	Device Used	LED	UART_TX	Switch
CY8CKIT-062-WiFi-BT	CY8C6247BZI-D54	P13[7]	P5[1]	P0[4]
CY8CKIT-062-BLE	CY8C6347BZI-BLD53	P13[7]	P5[1]	P0[4]
CY8CPROTO-062-4343W	CY8C624ABZI-D44	P13[7]	P5[1]	P0[4]

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: CE220498 – PSoC® 6 MCU Free-Running Multi-Counter Watchdog Timer

Document Number: 002-25571

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6374155	RJVB	11/21/2018	New code example
*A	6487889	RJVB	02/18/2019	Updated the project to ModusToolbox 1.1

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/pmhc
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

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

© Cypress Semiconductor Corporation, 2018-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.