CE220692 – PSoC 6 MCU Frequency Measurement Using TCPWM

-

Objective

This code example demonstrates how to use the TCPWM Component in PSoC® 6 MCU to measure the frequency of a periodic digital signal.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.1

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

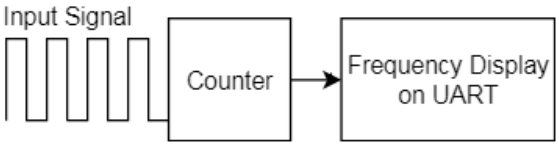
Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Overview

Frequency measurement is counting the number of edges (rising, falling, or both) that occur within a known time interval. To find this number, this project uses a one-second time window and determines the number of counts (rising edges) within that time window.

Note: This project measures digital signals. If you want to measure the frequency of other waveforms, the waveform must be converted to a digital signal before inputting it to the counter.



This code example assumes that you are familiar with the PSoC 6 MCU device and the PSoC Creator™ IDE. If you are new to PSoC 6 MCU, see the application note AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity.

Hardware Setup

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

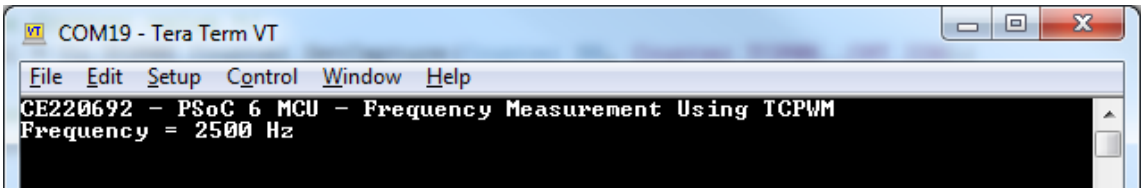
Software Setup

This code example requires a PC terminal emulator such as PuTTY or Tera Term to view the output. For terminal emulator setup, see Operation below.

Operation

1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Open terminal emulator software such as PuTTY or Tera Term.
   a. If using PuTTY, select **Serial** under **Connection type**, under **Serial line**, enter the COM port that KitProg is connected to, and under **Speed**, enter **115200**.
   b. If using Tera Term, select **Serial**, and from the drop-down menu, select the COM port connected to KitProg and click OK. Then under the **Setup** tab, select **Serial port** and set the **Speed** to **115200**.
3. Build the project and program it into the PSoC 6 MCU device. Choose **Debug** > **Program**. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.
4. On successful programming, the UART terminal displays the code example title and the frequency as shown in Figure 1.

Figure 1. UART Terminal Displaying Starting Message and Result



5. To test the code example, change the **Period** and **Compare** values of the PWM_FreqGen Component to generate a different frequency.

Design and Implementation

This example uses three TCPWM Components.

One is set up as a timer/counter (**OneSecTimer**), with a 1-kHz clock, and a period of 999. The overflow output will generate a pulse every second and connect directly to the capture input of the counter.

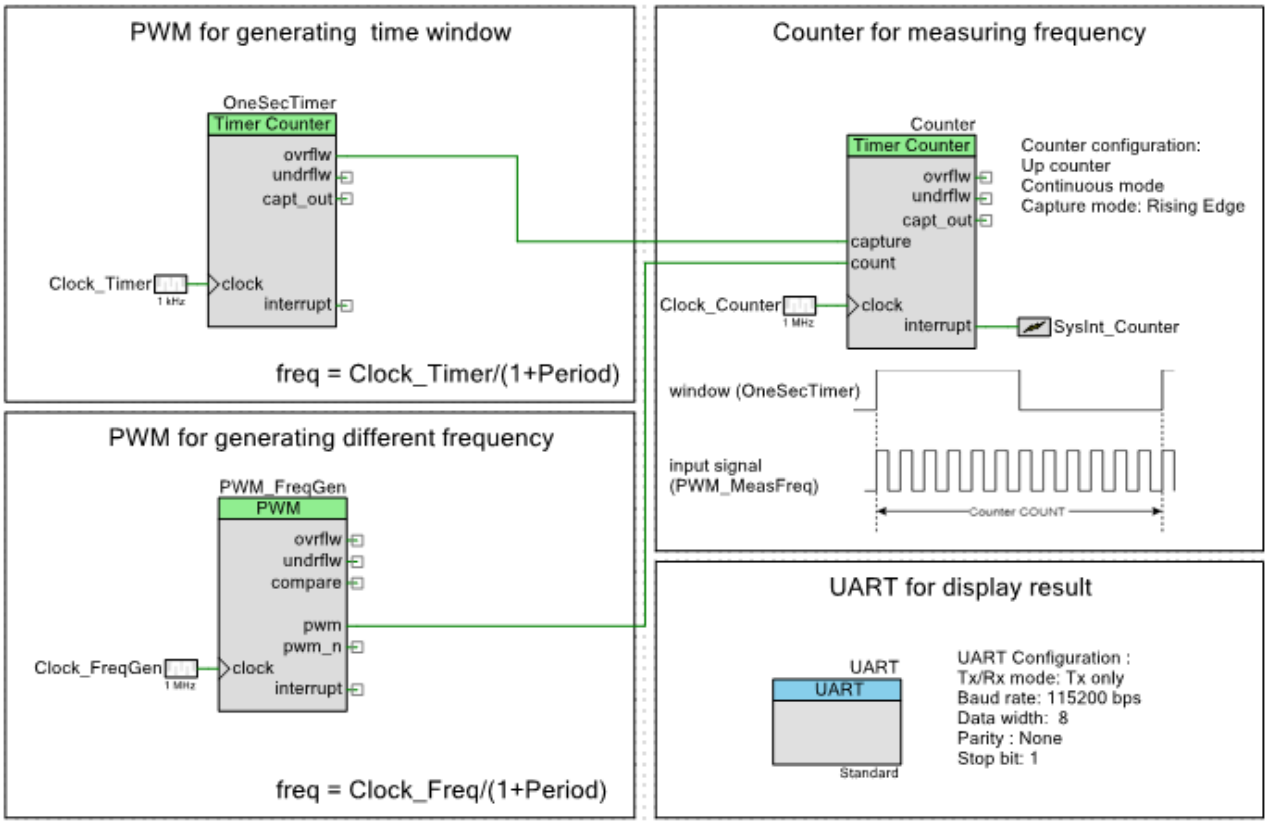
A second is set up as a PWM (**PWM_FreqGen**) with a 1-MHz clock. Based on the “Period” and “Compare” values, it generates a signal at a particular frequency. By default, the Component is configured to generate a signal at 2500 Hz (2.5 kHz).

The third Component is set up as a counter (**Counter**). It receives the capture input from the OneSecTimer, and the input signal from the frequency generator. Once per second, it reads the count from the frequency generator. Firmware performs the calculation to identify the input frequency based on how the count changes each second.

A UART PSoC Creator Component with baud rate of 115200 bps is configured to display the result in a terminal application such as Tera Term of PuTTY.

Figure 2 shows the PSoC Creator schematic for this code example.

Figure 2. TopDesign Schematic



Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1: PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
Timer Counter (TCPWM)	Counter	Count the number of rising edges of the input signal	**[General tab]** Capture input: Rising Edge Interrupt Source: Overflow/Underflow and Compare/Capture **[Input tab]** Count input: Rising edge
	OneSecTimer	Generates a one-second time window	**[Basic tab]** Period:999u
PWM (TCPWM)	PWM_FreqGen	Generates a signal at a particular frequency, based on the period and compare values	**[General tab]** Period:399u Compare:200u
Clock	Clock_Timer	Drive the OneSecTimer at 1 kHz	**Frequency**: 1 kHz
	Clock_FreqGen	Drive the PWM_FreqGen at 1 MHz	**Frequency**: 1 MHz
	Clock_Counter	Drive the Counter at 1 MHz	**Frequency**: 1 MHz
Interrupt	SysInt_Counter	Configure the interrupt	Default settings only
UART (SCB)	UART	Serial communication block for output on a terminal emulator	**[Basic tab]** TX/RX Mode: Tx only

For information on the hardware resources used by a Component, see the Component datasheet.

Figure 3 shows the configuration for the **Interrupts** tab in the **Design Wide Resources** window. The interrupt is enabled on CM4, and given a priority of 7. These are the default settings for an interrupt. The ISR is compiled as part of the CM4 code.

Figure 3. Interrupt Assignments

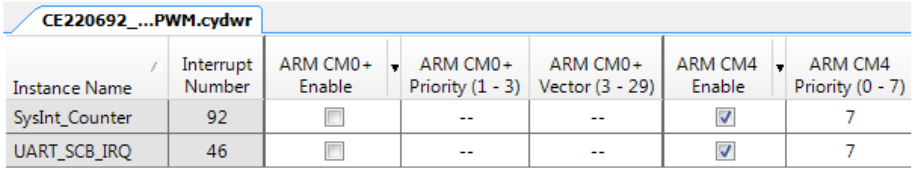
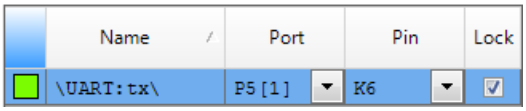


Figure 4 shows the pin assignment for the project done through the **Pins** tab in the **Design Wide Resources** window. These assignments are compatible with CY8CKIT-062-BLE.

Figure 4. Pin Assignments



Reusing This Example

This example is designed for the CY8CKIT-062-BLE pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed. For single-CPU PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c*.

In some cases, a resource used by a code example (for example, a Universal Digital 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 what a particular device supports.

Related Documents

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

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with BLE Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project
AN215656 – PSoC 6 MCU Dual-Core CPU system Design	Describes the dual-core CPU architecture in PSoC 6 MCU, and shows how to build a simple dual-core design
AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project	Describes how to import the code generated by PSoC Creator into your preferred IDE
PSoC Creator Component Datasheets	
Pins	Supports connection of hardware resources to physical pins
Timer Counter (TCPWM)	Supports fixed-function Timer/Counter implementation
Clock	Supports local clock generation
Interrupt	Supports generating interrupts from hardware signals
UART	Provides asynchronous communication interface using SCB hardware
Related Code Examples	
CE220290	PSoC 6 MCU: TCPWM Breathing LED
CE220169	PSoC 6 MCU: Periodic Interrupt using TCPWM
CE220291	PSoC 6 MCU: TCPWM Square Wave
CE220799	PSoC 6 MCU: Direction Detection Using Quadrature Decoder
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
Tool Documentation	
PSoC Creator	Look in the downloads tab for Quick Start and User Guides

Document History

Document Title: CE220692 – PSoC 6 MCU Frequency Measurement Using TCPWM

Document Number: 002-20692

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5896719	AJYA	09/26/2017	New code example
*A	6177969	AJYA	05/18/2018	Updated for PSoC Creator 4.2
*B	6624067	NRSH	07/22/2019	Update document and Creator project for clarity.

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

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, 2017-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("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 shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. 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. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, 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 any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

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.