

CE58033

Code Example: Pulse_Width_Measurement

Programming Language: C

Associated Part Families: CY8C27x43, CY8C29x66, CY8C24x23, CY8C21x34, CY8C24x94, CY8CLEDxx

Software Version: PSoC® Designer™ 5.3

Related Hardware: CY3210 PSoC Eval

Author: Umanath R Kamath

Objective

This code example demonstrates functionality of Capture feature of timer user module of PSoC® 1.

Overview

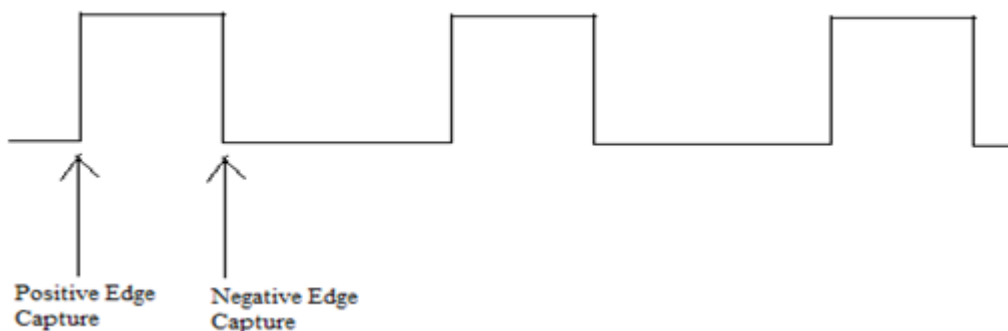
Pulse width measurement is a feature needed in many embedded systems. There are several reasons to find the width of a given pulse for example, to measure the duration of a given event or to find the interval between the occurrences of two events.

There are different methods of Pulse Width Measurement such as

1. Using timer capture method which is discussed in this code example.
2. Using GPIO: Based on GPIO interrupt whose source is change from read. In this case, the first read on GPIO
3. Loads/starts the timer in the assembly code of GPIO_ISR while the next read stops the timer. Thereby the pulse width can be calculated based on these values.

As shown in [Figure 1](#), the instances at which positive and negative edges occur for the test signal are recorded. The pulse width is calculated by finding the difference between these two values.

Figure 1. Test Signal whose Pulse Width needs to be Measure



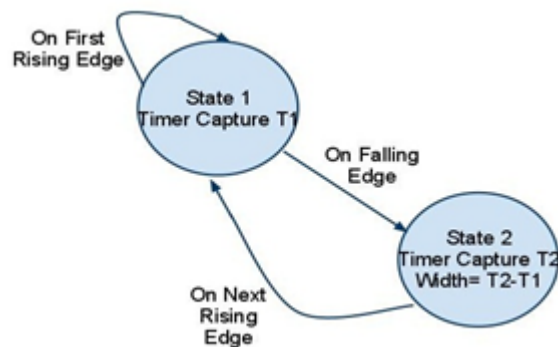
Principle

The principle of Pulse Width measurement can be understood by the simple state diagram shown in [Figure 2](#). As seen from the figure, the system can be in any of the two states based on the edge of the test signal detected.

The 16 bit Timer is initially interrupted by the positive edge (which is the capture edge) of the test signal in State 1. The capture value is then stored and the edge is changed to negative edge (Negative edge detection by Timer).

During the subsequent negative edge which follows the FSM is in State 2 wherein the capture value corresponding to negative edge is stored and the difference is the pulse width of the signal. The Timer is now changed to positive edge capture and the cycle follows by going to State 1 on the next positive edge.

Figure 2. State Diagram of Pulse Width Measurement System



User Module List and Placement

This table lists the user modules used in this example and the hardware resources occupied by each user module.

User Module	Placement
Timer (16-bit)	DBB00 and DBB01
PWM (optional)	DCB02
LCD	Software

User Module Parameter Settings

These tables show the user module parameter settings for each user module used in the example.

Timer		
Parameter	Value	Comments
Clock	VC2	VC2 is used to generate a 1 MHz clock input to the Timer. This results in a resolution of measurement of 1uS
Period	65535	The period is set to 65535 to avoid any underflow between the two instances.
Capture	Row0 Input 2	The input signal is routed to the capture input from P0[2] through Row0_Input2
Interrupt Type	Capture	The timer generates an interrupt on a capture event
ClockSync	Sync to SysClk	As the clock to the timer is derived from SysClk, the clock sync is set to "SyncToSysClk"
PWM		
Parameter	Value	Comments
Clock	VC2	Clock source for PWM is VC2
Period	250	The period of the PWM generated signal set to 250, so that the output frequency is around 4 KHz. This is the test signal whose pulse width needs to be measured.
Pulsewidth	100	Test pulse-width to be measures for verification
Compare type	Less than	Compare value should be less than the set threshold
ClockSync	Sync to SysClk	Clock should be synchronized to SysClk

LCD		
Parameter	Value	Comments
LCDPort	Port2	Port 2 is used for LCD connection
BarGraph	Disable	Bar graph feature is disabled

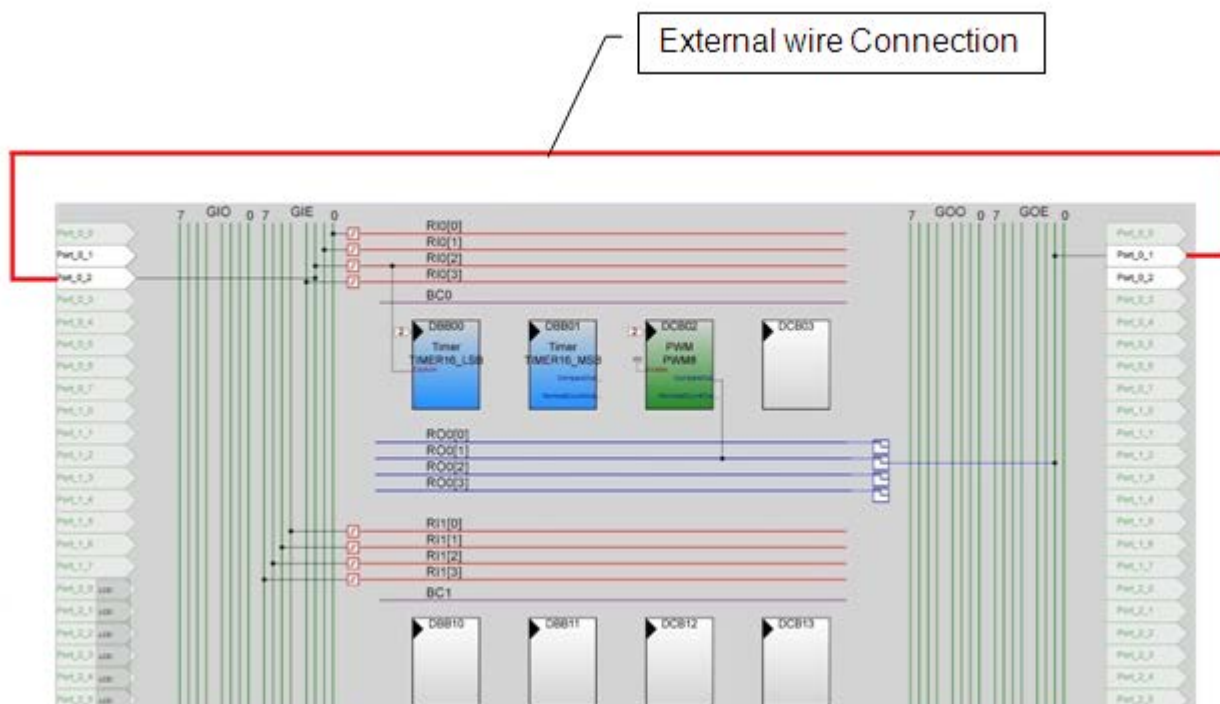
Notes

- Any of the associated part numbers for this example support the above resources and hence can be used to implement this design.
- The PWM is used only to generate a test signal to verify the operation. This can be avoided and a generic test signal from an external function generator can be used to feed input at P0.2. But make sure the frequency of such a signal is within the measurable range (Here the timer is fed by VC2 = 1 MHz, therefore the pulse frequency should be less than 1 MHz).

Global Resources

Important Global Resources		
Parameter	Value	Comments
VC1	12	$VC1 = 24 \text{ MHz} / 12 = 2 \text{ MHz}$
VC2	2	$VC2 = VC1 / 2 = 1 \text{ MHz}$
CPU Clock	24 MHz	Sets CPU clock to 24 MHz

Chip View under PSoC[®] Designer™



Hardware Connections

The Capture input to the Timer module is provided from P0.2. The test signal whose pulse width is to be measured is provided on P0.1 from the PWM module or by an external signal source.

Here the example is tested on CY3210 – PSoC Eval1 board.

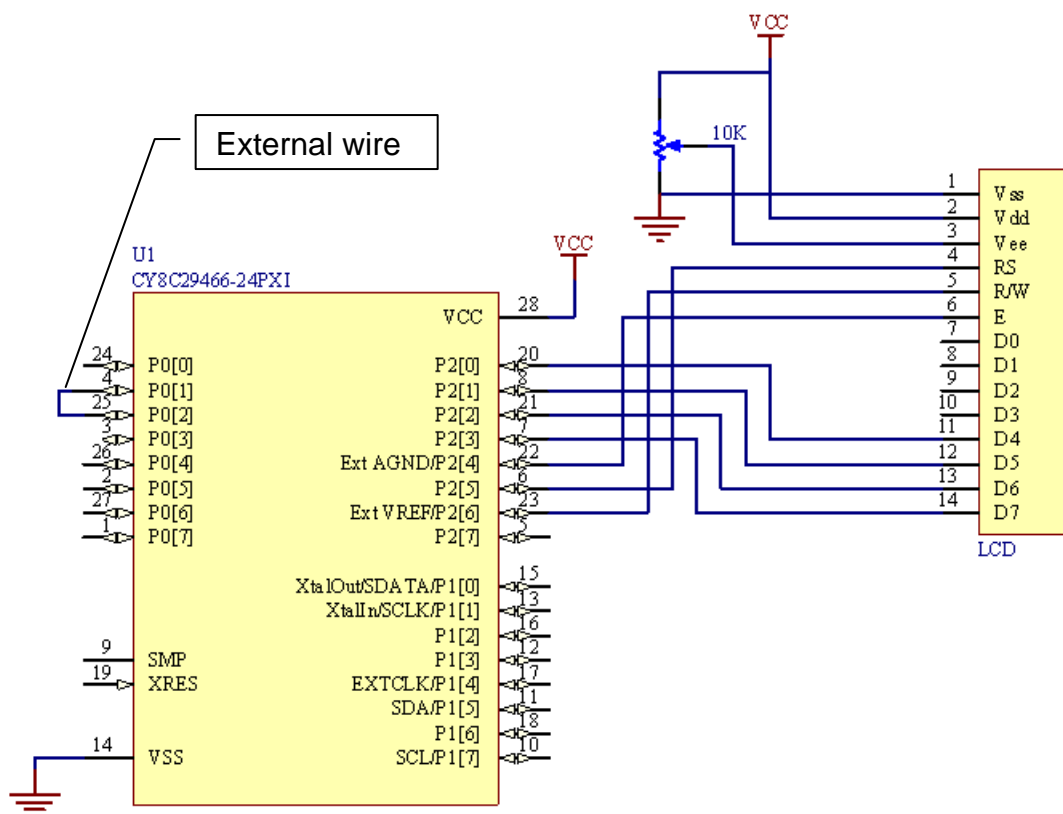
Hardware connection comprises of the following:

Connect wire between P0.1 to P0.2 which feeds the test signal generated in PSoC to the Timers for measurement.

Notes

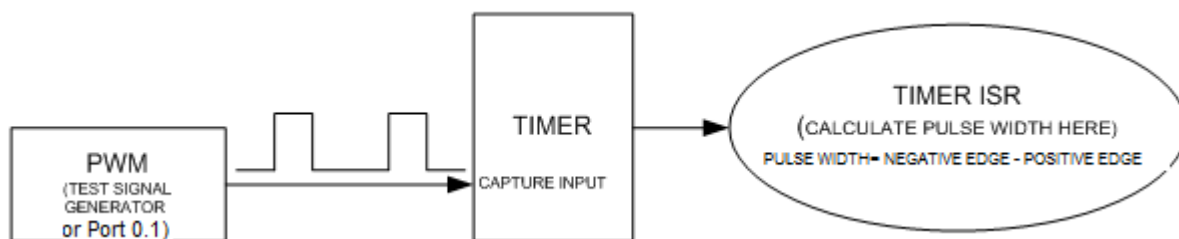
- The CPU system clock frequency should be high when compared to the range of pulse width to be measured. This is because there could be the edges from the test signal which could be missed when the controller is servicing the Timer Capture ISR. Therefore, note that when this example is cloned for another chip, the CPU_clock should be fixed to SysClk/1.

The following schematic diagram shows the Pulse Width Measurement.



Operation

Figure 3. Functional Diagram of the Measurement Setup



During Rising edge

Using the API `Timer_wReadCompareValue()`, the timer value at the moment of the rising edge is stored into the variable `CapturePosEdge`.

Using the statement, `Timer_FUNC_LSB_REG |= 0x80`; we set the 7th bit of the function register of the timer. By doing this, we change the capture type to falling edge. The falling edge flag is also set. Hence, the next time the interrupt is generated, the falling edge state is visited.

During Falling edge

Using the API `Timer_wReadCompareValue()`, the timer value at the moment of the falling edge is stored into the variable `CaptureNegEdge`.

Using the statement, `Timer_FUNC_LSB_REG &= ~0x80`; we clear the 7th bit of the Function register of the timer. By doing this, we change the capture type to rising edge. The falling edge flag is cleared, the pulse width is calculated and the data available flag is set.

Conclusion

The pulse width of the test signal is displayed on the LCD. In our case, 064 (hex equivalent of 100) is displayed on the LCD. To test the validity, one can vary the pulse width in the PWM user module and program the PSoC again. This would result in a new value being updated on the LCD during the next operation. Alternatively any test signal whose pulse width needs to be measured can be fed to P0.2.

Upgrade Information

As the `ljmp` instruction to the C ISR for the Timer16 Module is placed inside the user code markers inside the `TimerINT.asm` file, this change is preserved when the project is generated and built. If the source file for the `TimerINT.asm` file changes in a future release of PSoC Designer™, this instruction may get overwritten. So, if you have upgraded the PSoC Designer and find that this example is not working, check the following.

Open `TimerINT.asm` file and check if there is an `'ljmp _TimerCaptureISR'` instruction present in the user code area in the `_Timer_ISR` function. If not, add this line of code within the user code markers.

Document History

Document Title: Pulse Width Measurement using Timer capture in PSoC® 1 – CE58033

Document Number: 001-58033

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2819817	PRKR	12/02/2009	New example project
*A	3110066	UMRK	12/13/2010	1. Figure1 and Overview section were added. 2. State Diagram explaining the two functional states were shown in Figure 2 3. A new chip view in PSoC Designer was added. 4. Figure 3 shows the Functional Diagram of the project.
*B	3242304	UMRK	04/27/2011	1. Explanation of different methods to measure pulse width and the reason for the choice of method in the given example project. 2. Inclusion of state diagram which explains the flow of the project. 3. Inclusion of waveforms and modification of diagrams for better clarity in setup.
*C	3598234	KUK	01/16/2013	Obsolete document.
*D	3972748	KUK	04/18/2013	Change status from Obsolete to Active. Updated Software Version as "PSoC® Designer™ 5.3". Removed attachment file "Pulse_Width_Measurement".

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," PSoC Designer, and 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, 2009-2013. 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.