Using PSoC® 3 / PSoC 5 to Measure Interval between Two Events

.CE64441
Associated Part Families: CY8C38xx/CY8C55xx
Software: PSoC® Creator™
Related Hardware: CY8CKIT-001
Author: Lakshmi Natarajan

Code Example Objective

CE64441 explains how to use the counter in PSoC® 3 / PSoC 5 to measure the interval between two events.

Overview

In real-time systems, it may be important to know the interval between two events. This project explains how the counter can be configured to measure time between two events. The rising edge on a particular signal is considered as an event. This project measures time between two rising edges.

The project can also be used to measure low frequencies if the rising edges are periodic. If the rising edges are not periodic, the project simply does the interval measurement. The technique used to measure high frequency signals is described in Frequency Measurement with Counter.

The project uses the capture feature of the counter to measure the time interval. The time interval is displayed in microseconds on the LCD in hexadecimal format.

The project can start measuring from one microsecond. The upper limit for measurement is set by various factors explained in the following sections.

Component List

Instance Name	Component Name	Component Category	Comments
Counter	Counter	Digital → Functions	Configured as up counter with rising edge capture.
LCD	Character LCD	Display	Displays the time interval between two events
Pin_InputSignal	Digital Input	Ports and Pins	The input pin to which the signal is given. Connected to P3[0] in the DVK.
Clock_Counter	Clock	System	Configured for 3 MHz. Better precision can be achieved if a higher frequency clock is used.
ZeroTerminal_1	Logic Low “0”	Digital → Logic	Given to the reset pin of the counter
ISR_Counter	Interrupt	System	Interrupts on Capture of the Counter
Clock_Sync	Clock	System	Counter component is synchronous to this clock

Top Design

Figure 1 illustrates the components and their routing.

Figure 1. Component Routing

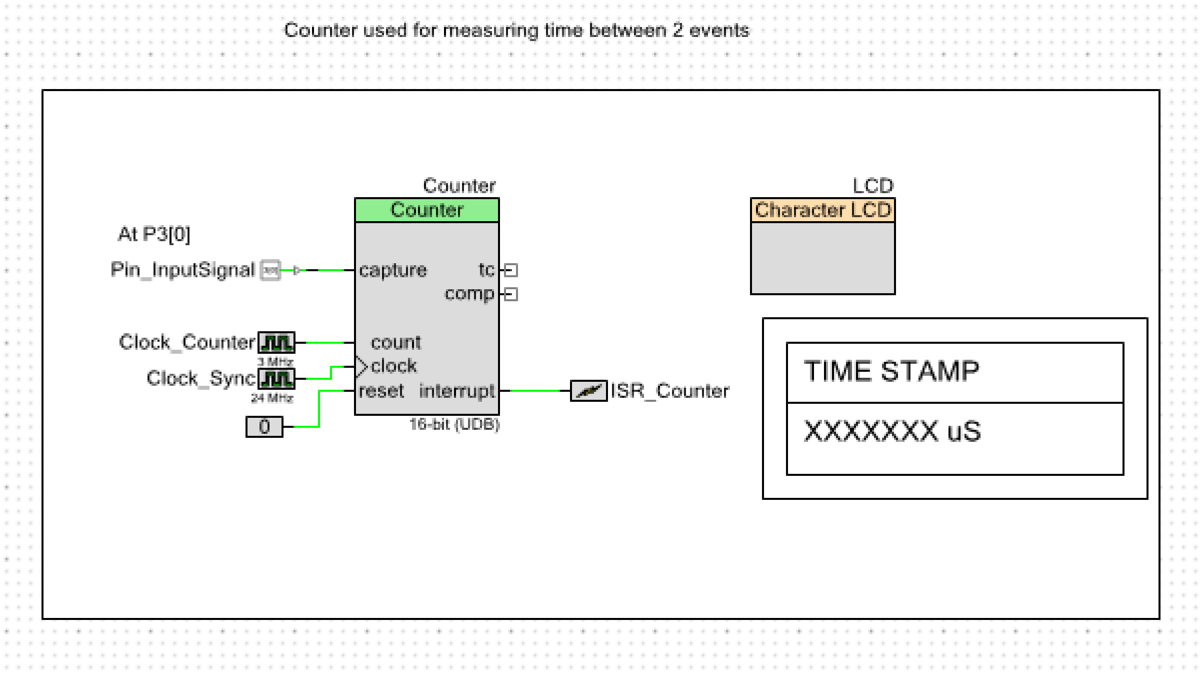
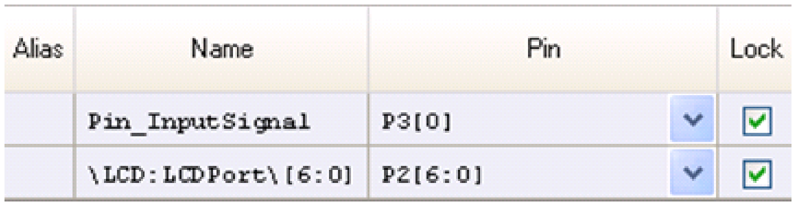


Figure 2 shows the pin placement (as in the *.cydwr* file).

Figure 2. Pin Placement

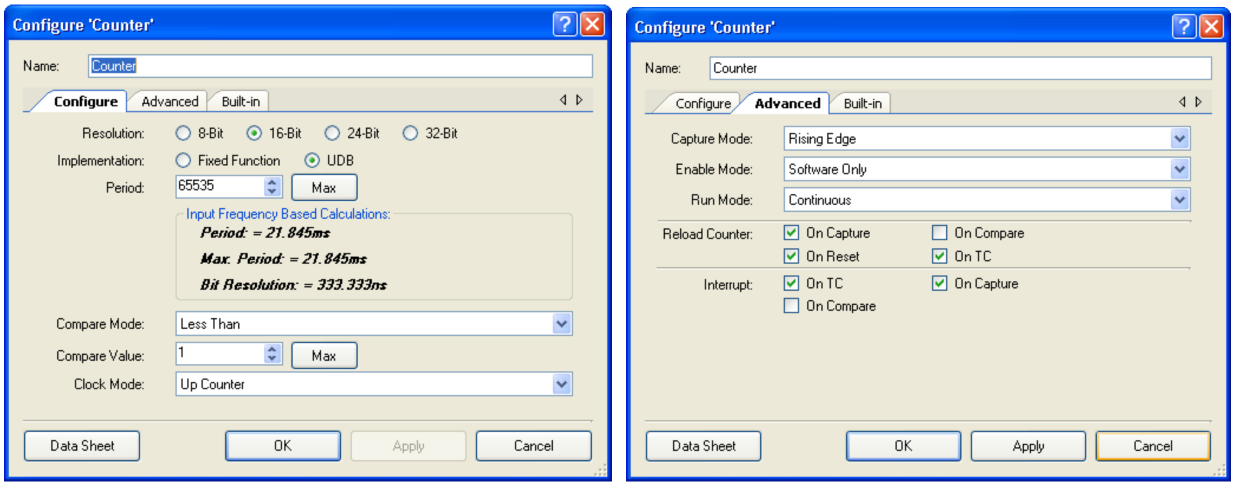


Alias	Name	Pin	Lock
	Pin_InputSignal	P3[0]	✓
	\LCD:LCDPort\[6:0]	P2[6:0]	✓

Component Configuration

Counter

Figure 3. Counter Configuration



Configure 'Counter' – Configure tab

Name: Counter

Resolution: 8-Bit [✓ 16-Bit] 24-Bit 32-Bit

Implementation: Fixed Function [✓ UDB]

Period: 65535 Max

Input Frequency Based Calculations:

Period: = 21.845ms

Max. Period: = 21.845ms

Bit Resolution: = 333.333ns

Compare Mode: Less Than

Compare Value: 1 Max

Clock Mode: Up Counter

Configure 'Counter' – Advanced tab

Name: Counter

Capture Mode: Rising Edge

Enable Mode: Software Only

Run Mode: Continuous

Reload Counter: ✓ On Capture, ☐ On Compare, ✓ On Reset, ✓ On TC

Interrupt: ✓ On TC, ✓ On Capture, ☐ On Compare

A 16-bit counter implemented in a universal digital block (UDB) is selected because we need the up counter feature, which is not supported by a fixed-function counter.

The “Clock Mode” of the counter is set to “Up Counter”. Counter will have two inputs in this mode: “count” and “clock.” Count is the input clock on whose rising edge the counter increments, and “clock” is the synchronizing clock for the counter component. Counter is configured for interrupt on capture; this brings another input called “capture” for the counter. The input signal whose time interval is to be calculated is connected to the “capture” input.

The counter is configured as an up counter because counting from zero makes the calculation easier. When the counter is configured as a down counter, to get the exact count value, subtract the captured value from the counter period before proceeding with calculations.

The counter is configured for a rising edge capture. The event whose time interval is to be measured is connected to the capture signal.

The counter is configured to “Reload on Capture”. This feature comes in handy when measuring time. The counter captures the value whenever the rising edge occurs at the capture signal. After it captures the value, it automatically reloads the counter with a zero. Because the counter is configured as an up counter, counting from zero, this feature gives a direct measure of the number of counts between two rising edges.

The counter is configured for Interrupt on Terminal Count and Interrupt on Capture. When the Capture Interrupt occurs, the captured count is read and the time value is calculated. When the Terminal Count Interrupt occurs, the number of times the counter overflows prior to the next capture is counted. This count value is used to measure the time between two capture edges.

Design Wide Resources

The project uses the default system wide resources settings. Refer to the *TimeStamp_Measurement.cydwr* file for the settings.

Operation

- The project uses the counter for the time measurement. The counter is configured for using capture at the rising edge. The signal whose time interval is to be measured is connected to the capture signal. The rising edge is considered as an event. The counter is an up counter and is reloaded with zero on capture. When a rising edge occurs, the counter captures the count value into the FIFO and reloads the counter. This captured count value is used for time measurement.

- The counter triggers an interrupt on the terminal count and capture event. There is only one interrupt that occurs for both the events. To find the cause of the interrupt, we should read the counter status register and find the bits for the interrupts.

- Refer to the *ISR_Counter.c* file where the status register is read and stored to a variable. This variable is checked in the *main.c* for the interrupt. The variable is 'AND'ed with predefined macros for the status bits. Refer to *Counter.h* for the definitions (for example, STATUS_OVERFLOW_INT_EN_MASK)

When the capture interrupt occurs, the time value is calculated and displayed on the character LCD (in microseconds). When the terminal count interrupt occurs, the number of times the counter overflows is counted. The terminal count interrupt occurs only when the counter overflows. Because the counter is configured for only a16-bit resolution, the time interval is limited by the period of the counter. Note that the period of the counter is limited by the configured period count and the clock. For example, when the period count is 0xFFFF and the clock is 3 MHz, the counter can measure only for 21.845 ms. To overcome this limitation, consider the counter overflow. While calculating the time interval, consider the number of times the counter overflows. Therefore, the formula to measure the time interval is:

Time interval = (Period of the counter * number of times the counter overflows) + (Time measured using the capture value)

This formula helps to measure huge time intervals. The upper limit for the time interval is given by the 32-bit overflow limit. Therefore, changing the period value and the clock for the counter helps to achieve greater time intervals.

When calculating the period and the time for capture value, note that the count value is always incremented by one before the time calculation (for example, see the calculation for the counter period before the "for" loop in main.c).

Hardware Connections

The project is tested with the PSoC development kit CY8CKIT-001.

Use the kit with the jumpers in their default state. Refer to the *PSoC Development Kit Board Guide* supplied with the kit.

- Place the character LCD on P18 of the CY8CKIT-001 DVK.

- Enable the LCD power by placing the jumper J12 to ON position.

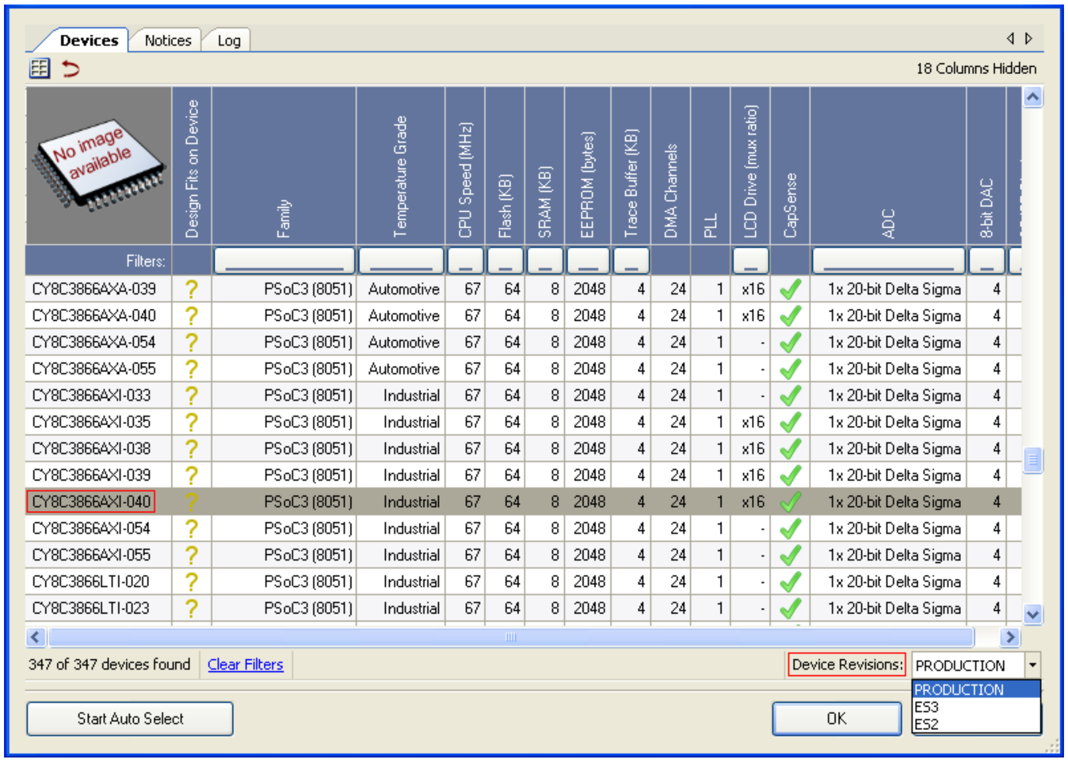
- Connect the function generator output to Pin 'P3[0]' in the DVK. This is the input to the counter.

- The following configuration should be set in the function generator:
  - Load impedance: HI-Z
  - Period: The lower limit is 1 µs. The upper limit can be measured based on the above explanation.
  - Voltage levels:
    - Low level – 0 V
    - High level – Vddd (Vddd is the limit set in the DVK for the digital voltage)

Output

- Use device selector (Project->Device Selector) window in PSoC Creator to select the appropriate device and Device Revision.

  If you are using PSoC3 device (for example, CY8C3866AXI-040) with production revision, then use the following selection:



	Design Fits on Device	Family	Temperature Grade	CPU Speed (MHz)	Flash (KB)	SRAM (KB)	EEPROM (bytes)	Trace Buffer (KB)	DMA Channels	PLL	LCD Drive (mux ratio)	CapSense	ADC	8-bit DAC
CY8C3866AXA-039	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-040	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-054	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-055	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-033	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-035	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-038	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-039	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-040	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-054	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-055	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866LTI-020	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866LTI-023	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4

347 of 347 devices found Clear Filters

Device Revisions: PRODUCTION

PRODUCTION
ES3
ES2

  Similarly, select appropriate device number to work with PSoC5 Device family (for example, CY8C5588AXI-060)

  Note: For engineering samples, device revision is marked on the package as part of the device number. Production silicon will not have an ES marking.

- Build the project and program the PSoC 3 device
- Reset the device by pressing SW4 (Reset switch)
- The project is designed to display the time interval in microseconds
- Set the period in the function generator to 10 microseconds
- The LCD display should show 0000000A uS. (Hex value for 10 microseconds)
- Change the period and observe the value on LCD

Document History

Document Title: Using PSoC® 3 / PSoC 5 to Measure Interval between Two Events – CE64441

Document Number: 001-64441

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3049552	LNAT	10/06/2010	New example project (TimeStamp Measurement)
*A	3153594	DASG	01/25/2011	Updated components in the project to the latest version. Deleted he component version number from the component table in the documentation. Changed screen shots according to the changes in the counter component. Updated the input names for the counter input for better understanding. Updated the document to explain new counter component changes. Edited the output section for selecting the device and silicon revision versions. Removed the "project name", "programming language" and "prerequisites" from the header of the example project.
*B	3223246	DASG	04/12/2011	In the component list, error is rectified by changing 'lower frequency' to 'higher frequency'. In page 4, micro second is rectified with ,milli second. In main.c of the project, the LCD output API is rectified from (1,3) to (1,4) so that one nibble is prevented from being lost. Micro second is changed from 'US' to 'uS'. This is also reflected in the document.
*C	3246098	AESA	05/02/2011	Fixed Typo in Document History title

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