Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-15482

Spec Title: AN15482 - USING CAPTURE TIMERS IN ENCORE(TM) II AND ENCORE II LV DEVICES

Replaced by: NONE

AN15482

Using Capture Timers in enCoRe™ II and enCoRe II LV Devices

Author: Jacob Tomy

Associated Part Family: CY7C63310, CY7C638xx, CY7C601xx, CY7C602xx

Associated Code Examples: None

Related Application Notes: None

To get the latest version of this application note, or the associated project file, visit http://www.cypress.com/go/AN15482.

AN15482 describes the features and architecture of the enCoRe™ II capture timer module, and explains its use. Assembly language and C language code examples are also provided as PSoC® Designer™ projects along with this application note.

OBSOLETE

Contents

1 Introduction....1
2 WirelessUSB Resources....2
3 PSoC Designer....2
4 enCoRe II Capture Timer Block Architecture....3
4.1 Introduction to the enCoRe II Timer Block....3
4.2 enCoRe II Capture Timer....3
4.3 Block Diagram of Capture Timer....4
4.4 Registers Associated with Capture Timers....5
5 General Usage Model....9
5.1 Hardware Setup....9
5.2 Firmware Setup....9
6 Capture Timer - Caveats....11
7 Using PSoC Designer to Set Up an enCoRe II Device for Capture Timer Operation....12
7.1 Device Editor....12
7.2 Application Editor....12
7.3 Debugger....12
8 Summary....12
Document History....13
Worldwide Sales and Design Support....14
Products....14
PSoC Solutions....14
Cypress Developer Community....14
Technical Support....14

1 Introduction

In embedded microcontroller systems, you will often need to record time instants when the state of a signal changes. enCoRe II devices make this possible by using capture timers. These are essentially registers that can store the value of a timer when the signal of interest changes state.

Some practical applications of capture timers include, but are not limited to:

- Decoding communication data. This is especially useful with infrared or 27-MHz frequencies. In these technologies, data is decoded based on the duration for which a signal is high or low.
- Gaming. Use a capture timer when you want to record button-press duration, the time between successive button presses, and so on.

2 WirelessUSB Resources

Cypress provides a wealth of data at www.cypress.com to help you select the right WirelessUSB device for your design, and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see the wireless webpage.

- **Overview**: Wireless Roadmap, Modules Roadmap, Wireless Portfolio
- **Product Selectors**: Wireless Product Selector
- **Datasheets**: Describe and provide electrical specifications for various device families. You can access the datasheets of all wireless products here.
- **Application Notes and Code Examples**: Cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples. You can access the complete list of wireless application notes here and code examples here.
- **Technical Reference Manuals (TRM)**: Provide detailed descriptions of the architecture and registers in each WirelessUSB device family. You can access the complete list of wireless product TRMs here.
- **Development Kits**: You can access the complete list of wireless kits and reference designs here.

Cypress also offers ARM® Cortex®-M0 based, single-chip Bluetooth® Low Energy (BLE) or Bluetooth Smart solutions. You can learn more about Cypress BLE devices here.

3 PSoC Designer

PSoC Designer is the revolutionary Integrated Design Environment (IDE) that you can use to customize PSoC to meet your specific application requirements. PSoC Designer software accelerates system bring-up and time-to-market. Develop your applications using a library of pre-characterized analog and digital peripherals in a drag-and-drop design environment. Then, customize your design leveraging the dynamically generated API libraries of code. Finally, debug and test your designs with the integrated debug environment including in-circuit emulation and standard software debug features.

- Application Editor GUI for device and User Module configuration and dynamic reconfiguration
- Extensive User Module catalog
- Integrated source code editor (C and Assembly)
- Free C compiler with no size restrictions or time limits
- Built-in debugger
- In-circuit emulation (ICE)
- Built-in support for communication interfaces:
  - Hardware and software I2C slaves and masters
  - Low/Full-speed USB 2.0
  - Up to four full-duplex UARTs, SPI master and slave, and wireless

Figure 1 shows the PSoC Designer window. **Note:** This is not the default view.

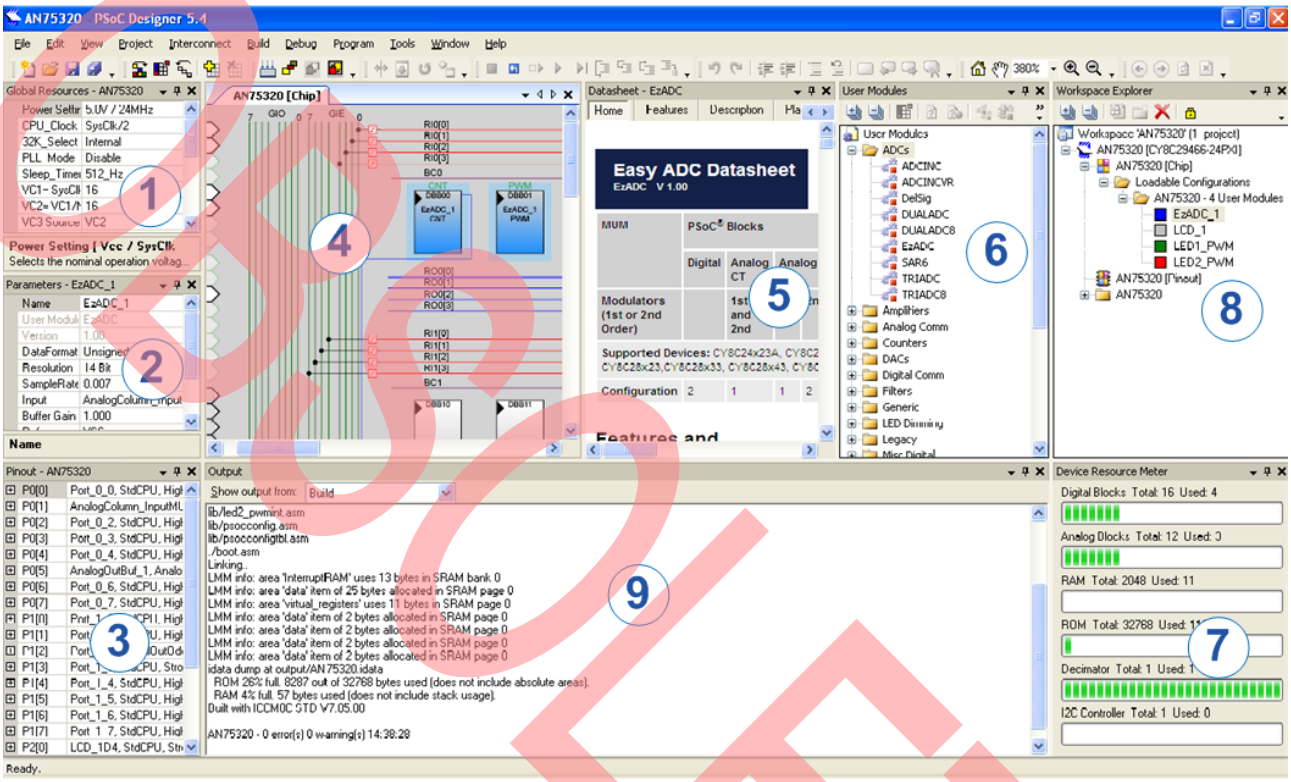
1. **Global Resources –** all device hardware settings
2. **Parameters –** the parameters of the currently selected User Modules
3. **Pinout –** information related to device pins
4. **Chip-Level Editor –** a diagram of the resources available on the selected chip
5. **Datasheet –** the datasheet for the currently selected User Module
6. **User Modules –** all available User Modules for the selected device
7. **Device Resource Meter –** device resource usage for the current project configuration

OBSOLETE

8. **Workspace –** a tree level diagram of files associated with the project
9. **Output –** output from project build and debug operations

Note: For detailed information on PSoC Designer, go to **PSoC Designer > Help > Documentation > Designer Specific Documents > IDE User Guide**.

Figure 1. PSoC Designer Layout



4 enCoRe II Capture Timer Block Architecture

This section provides an introduction to the enCoRe II Timer Block, discusses the enCoRe II capture timer, provides a block diagram of the capture timer, and lists details of the registers associated with capture timers.

4.1 Introduction to the enCoRe II Timer Block

All of the timer functions in the enCoRe II family of devices are provided by a single timer block. This timer block can be asynchronous from the CPU clock and includes:

- A 16-bit free-running timer (FRT) that is clocked by the Timer Capture Clock (TCAPCLK) signal.
- A 12-bit programmable interval timer that is clocked by the Interval Timer Clock (ITMRCLK) signal.
- Two 8-bit capture timer registers each for two capture inputs. These 8-bit registers can be concatenated to get a 16-bit capture value.

The TCAPCLK and ITMRCLK can in turn be sourced by the 24-MHz internal main oscillator (IMO), the 32-kHz low-power oscillator, or the external clock signal. In addition, the TCAPCLK can serve as a source for the ITMRCLK.

4.2 enCoRe II Capture Timer

Every enCoRe II device has two capture timer inputs bonded out as two pins: TIO0 and TIO1. Each of these inputs has two 8-bit registers associated with it to capture the rising and falling edge events. For example, the two registers associated for the TIO0 input are:

OBSOLETE

- Timer Capture 0 Rising register (TCAP0R)
- Timer Capture 0 Falling register (TCAP0F)

The capture timer inputs (TIO0 and TIO1) are registered synchronous to the TCAPCLK edges. The TCAPCLK also clocks the 16-bit free running timer in the device (see Figure 2).

TCAP0R holds the time instant of signal rising edges and TCAP0F holds the time instant of signal falling edges at the TIO0 input. Similar registers exist for the TIO1 input as well.

These time instants registered are actually the value of the 16-bit FRT. You can configure the desired eight bits to be captured from the 16-bit FRT. This is achieved by modifying the prescale bits [2:0] in the Timer Configuration register. This feature provides for eight possible capture timer resolutions with a single clock.

The Timer Configuration register also allows for:

- 16-bit capture mode at TIO0
  - The register can hold 16 bits of the free-running timer in this mode. It is available only for the TIO input. In this mode, Capture1 (TIO1) is disabled and Capture1 registers are used as an extension (16-bit) to the Capture0 registers.
- First or recent edge hold
  - When set, the time of the first edge is stored in the register. Subsequent edges are registered only if the capture timer register is read.
  - When clear, the most recent edge is stored in the capture timer registers.

In addition to the Timer Configuration register, there is a Capture Interrupt Enable register and a Capture Interrupt Status register. These two registers are linked with generating interrupts on rising or falling edges of capture timer inputs. All registers associated with the capture timer, their address locations, and bit settings and details are discussed in Registers Associated with Capture Timers.

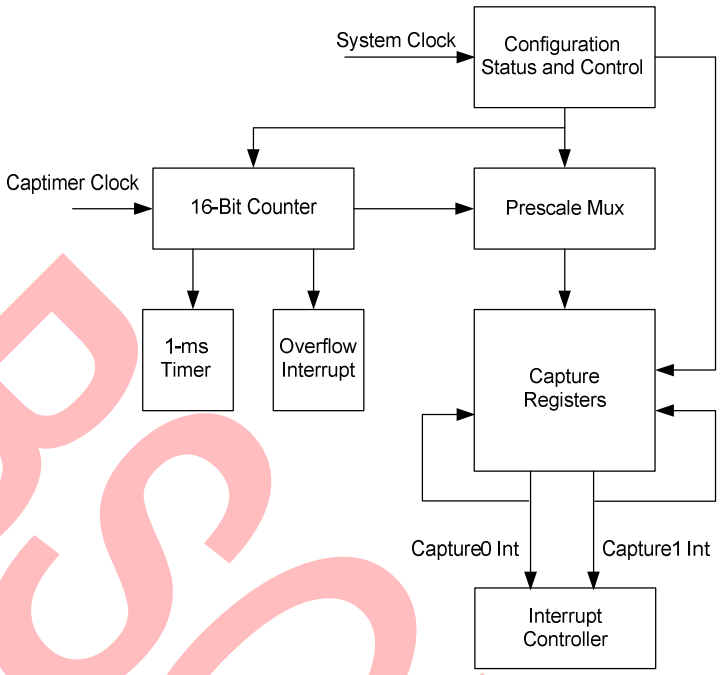
4.3 Block Diagram of Capture Timer

The block diagram provides an overview of how the different modules make up the capture timer block. The TCAPCLK is the source for the free-running timer. The capture timer registers hold the value of the 16-bit FRT depending on the values set in the configuration register. The capture timer registers can be made to generate an interrupt when they are loaded. enCoRe II devices provide two interrupts for capture timers: the Capture0 interrupt and Capture1 interrupt.

These interrupts are triggered when both the rising edge and falling edge registers associated with that input are loaded. Depending on the application requirements, user firmware must check to see if the interrupt was triggered by a falling or rising edge. The Capture Interrupt Status registers are used for this purpose.

OBSOLETE

Figure 2. Capture Timer Block Diagram



OBSOLETE

4.4 Registers Associated with Capture Timers

The following registers are associated with capture timers. Their addresses, bit settings, read/write settings, and default values are provided in this section.

Table 1. Timer Clock Configuration (TMRCLKCR) [0x31] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAPCLK Divider		TCAPCLK Select		ITMRCLK Divider		ITMRCLK Select	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	0	0	0	1	1	1	1

Bit	Name	Description
7:6	TCAPCLK Divider[1:0]	TCAPCLK Divider controls the TCAPCLK divisor.0 0 = Divider Value 2 0 1 = Divider Value 4 1 0 = Divider Value 6 1 1 = Divider Value 8
5:4	TCAPCLK Select[1:0]	TCAPCLK Select controls the source of the TCAPCLK. 0 0 = Internal 24-MHz oscillator 0 1 = External clock at CLKIN (P0.0) input 1 0 = Internal 32-kHz low-power oscillator 1 1 = TCAPCLK disabled **Note** The 1024-µs interval timer is based on the assumption that TCAPCLK is running at 4 MHz. Changes in TCAPCLK frequency cause a corresponding change in the 1024-µs interval timer frequency.
3:2	ITMRCLK Divider[1:0]	ITMRCLK Divider controls the ITMRCLK divisor. 0 0 = Divider value of 1 0 1 = Divider value of 2 1 0 = Divider value of 3

		1 1 = Divider value of 4
1:0	**ITMRCLK Select[1:0]**	ITMRCLK Select field controls the source of the ITMRCLK.0 0 = Internal 24-MHz oscillator 0 1 = External clock at CLKIN (P0.0) input 1 0 = Internal 32-kHz low-power oscillator 1 1 = TCAPCLK

Table 2. Timer Configuration (TMRCR) [0x2A] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	First Edge Hold	8-bit Capture Prescale[2:0]			Cap0 16-bit Enable	Reserved		
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7	**First Edge Hold**	The First Edge Hold function applies to all four capture timers. 0 = The time of the most recent edge is held in the Capture Timer Data register. If multiple edges have occurred since reading the capture timer, the time for the most recent one is read. 1 = The time of the first occurrence of an edge is held in the Capture Timer Data register until the data is read. Subsequent edges are ignored until the Capture Timer Data register is read.
6:4	**8-bit Capture Prescale[2:0]**	This field controls which eight bits of the 16-bit free-running timer are captured when in bit mode. 0 0 0 = capture timer[7:0] 0 0 1 = capture timer[8:1] 0 1 0 = capture timer[9:2] 0 1 1 = capture timer[10:3] 1 0 0 = capture timer[11:4] 1 0 1 = capture timer[12:5] 1 1 0 = capture timer[13:6] 1 1 1 = capture timer[14:7]
3	**Cap0 16-bit Enable**	0 = Capture 0 16-bit mode is disabled 1 = Capture 0 16-bit mode is enabled. Capture 1 is disabled and the Capture 1 rising and falling registers are used as an extension to the Capture 0 registers—extending them to 16 bits.
2:0	**Reserved**	

Table 3. Capture Interrupt Enable (TCAPINTE) [0x2B] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Cap1 Fall Enable	Cap1 Rise Enable	Cap0 Fall Enable	Cap0 Rise Enable
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:4	**Reserved**	
3	**Cap1 Fall Enable**	0 = Disable the capture 1 falling edge interrupt 1 = Enable the capture 1 falling edge interrupt
2	**Cap1 Rise Enable**	0 = Disable the capture 1 rising edge interrupt 1 = Enable the capture 1 rising edge interrupt

OBSOLETE

1	**Cap0 Fall Enable**	0 = Disable the capture 0 falling edge interrupt 1 = Enable the capture 0 falling edge interrupt
0	**Cap0 Rise Enable**	0 = Disable the capture 0 rising edge interrupt 1 = Enable the capture 0 rising edge interrupt

Table 4. Capture Interrupt Status (TCAPINTS) [0x2C] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Cap1 Fall Active	Cap1 Rise Active	Cap0 Fall Active	Cap0 Rise Active
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:4	**Reserved**	
3	**Cap1 Fall Active**	0 = No event 1 = A falling edge has occurred on Cap1
2	**Cap1 Rise Active**	0 = No event 1 = A rising edge has occurred on Cap1
1	**Cap0 Fall Active**	0 = No event 1 = A falling edge has occurred on Cap0
0	**Cap0 Rise Active**	0 = No event 1 = A rising edge has occurred on Cap0

Note: The interrupt status bits must be cleared by firmware to enable subsequent interrupts. This is achieved by writing a '1' to the corresponding Interrupt status bits.

Table 5. Timer Capture 0 Rising (TCAP0R) [0x22] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 0 Rising [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:0	**Capture 0 Rising[7:0]**	This register holds the value of the free-running timer at the point when the last rising edge occurred on the TCAP0 input. When Capture 0 is in 8-bit mode, the bits that are stored here are selected by the Prescale[2:0] bits in the Timer Configuration register. When Capture 0 is in 16-bit mode, this register holds the lower-order 8 bits of the 16-bit timer.

OBSOLETE

Table 6. Timer Capture 1 Rising (TCAP1R) [0x23] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 1 Rising [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:0	Capture 1 Rising[7:0]	This register holds the value of the free-running timer at the point when the last rising edge occurred on the TCAP1 input. The bits that are stored here are selected by the Prescale[2:0] bits in the Timer Configuration register. When Capture 1 is in 8-bit mode, this register holds the high-order 8 bits of the 16-bit timer from the last Capture 1 rising edge. When Capture 1 is in 16-bit mode, this register is loaded with the high-order 8 bits of the 16-bit timer on TCAP1 rising edge.

Table 7. Timer Capture 0 Falling (TCAP0F) [0x24] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 0 Falling [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:0	Capture 0 Falling[7:0]	This register holds the value of the free-running timer at the point when the last falling edge occurred on the TCAP0 input. When Capture 0 is in 8-bit mode, the bits that are stored here are selected by the Prescale[2:0] bits in the Timer Configuration register. When Capture 0 is in 16-bit mode, this register holds the lower-order 8 bits of the 16-bit timer.

Table 8. Timer Capture 1 Falling (TCAP1F) [0x25] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 1 Falling [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit	Name	Description
7:0	Capture 1 Falling[7:0]	This register holds the value of the free-running timer at the point when the last falling edge occurred on the TCAP1 input. The bits that are stored here are selected by the Prescale[2:0] bits in the Timer Configuration register. When Capture 1 is in 8-bit mode, this register holds the high-order 8 bits of the 16-bit timer from the last Capture 1 falling edge. When Capture 1 is in 16-bit mode, this register is loaded with the high-order 8 bits of the 16-bit timer on the TCAP1 falling edge

In addition to these registers, pins P0.5 and P0.6 should be configured as inputs. This is done using the GUI interface in PSoC Designer or by writing to registers P05CR and P06CR. Refer to the enCoRe II datasheet for further details.

OBSOLETE

5 General Usage Model

This section presents the hardware and software setup information.

5.1 Hardware Setup

Setting the hardware for capture timer operation is simple and straightforward. Connect the signal of interest to either P0.5 or P0.6 in the enCoRe II device.

5.2 Firmware Setup

The following updates must be made in the firmware to configure the capture timer for operation. Some of these can be alternatively set using the GUI interface in PSoC Designer. Follow the steps in the order presented.

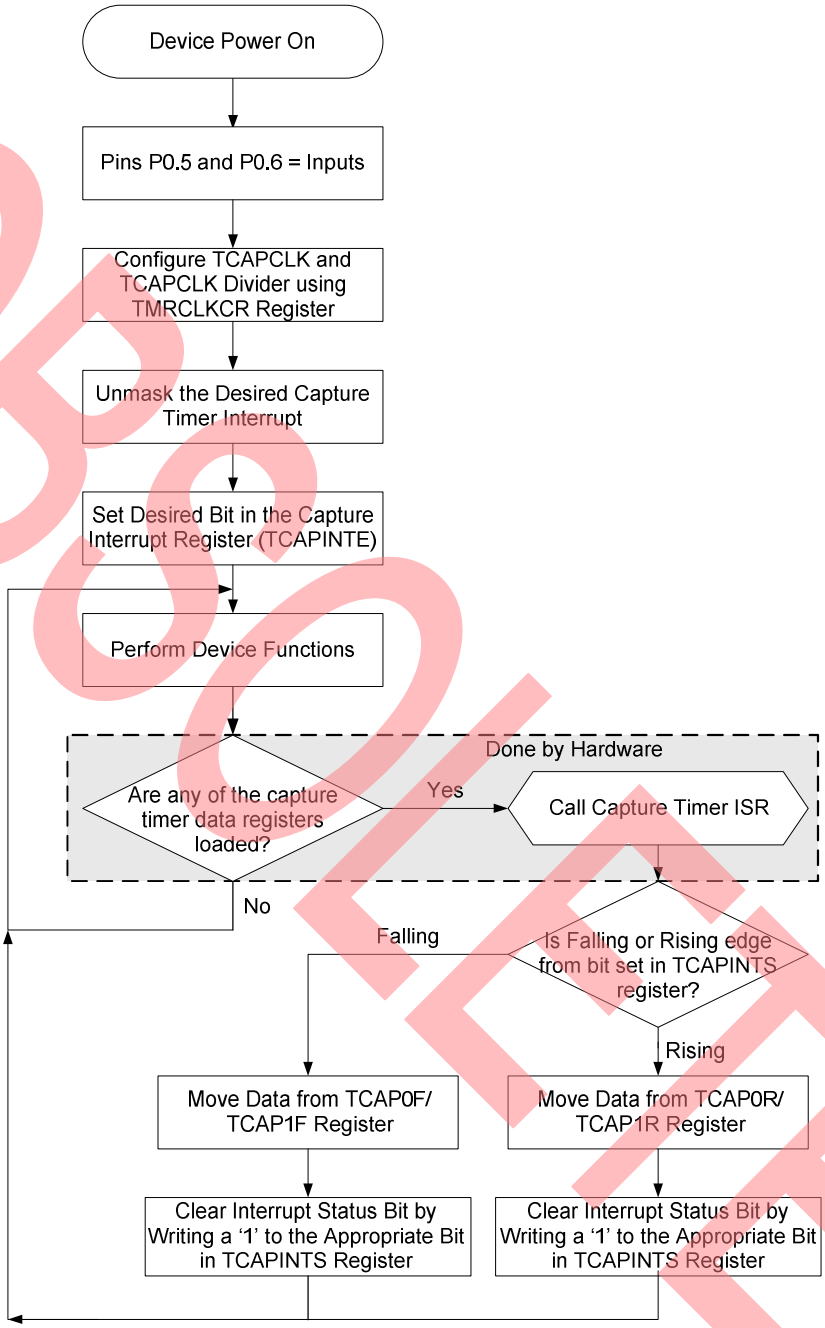
10. Configure pins P0.5 and P0.6 as inputs by clearing the output enable in registers P05CR and P06CR.
11. Configure TCAPCLK and TCAPCLK divider using register TMRCLKCR (Table 1 on page 5).
12. Use the TMRCR (Table 2 on page 6) to get the settings you want: first edge hold, prescale value, and 16-bit capture enable.
13. Unmask the Capture Timer Interrupt in the interrupt mask register.
14. Set the bit for the capture interrupt you want in the Capture Interrupt Enable register (TCAPINTE) (Table 3 on page 6).

If the device is started at this point, it captures the rising and falling edges on TIO0 and TIO1 inputs. This sets the corresponding bits in the Capture Interrupt Status register (TCAPINTS) (Table 4 on page 7). To allow subsequent interrupts, the status bits must be cleared. This is achieved by writing a ‘1’ to the corresponding bit location in the register. This action is generally done in the ISR after determining whether the interrupt was triggered by a rising or falling edge and then clearing the corresponding interrupt bit in the status register before exiting the ISR.

Figure 3 shows the steps to configure the capture timer.

OBSOLETE

Figure 3. Capture Timer Setup Flow Chart



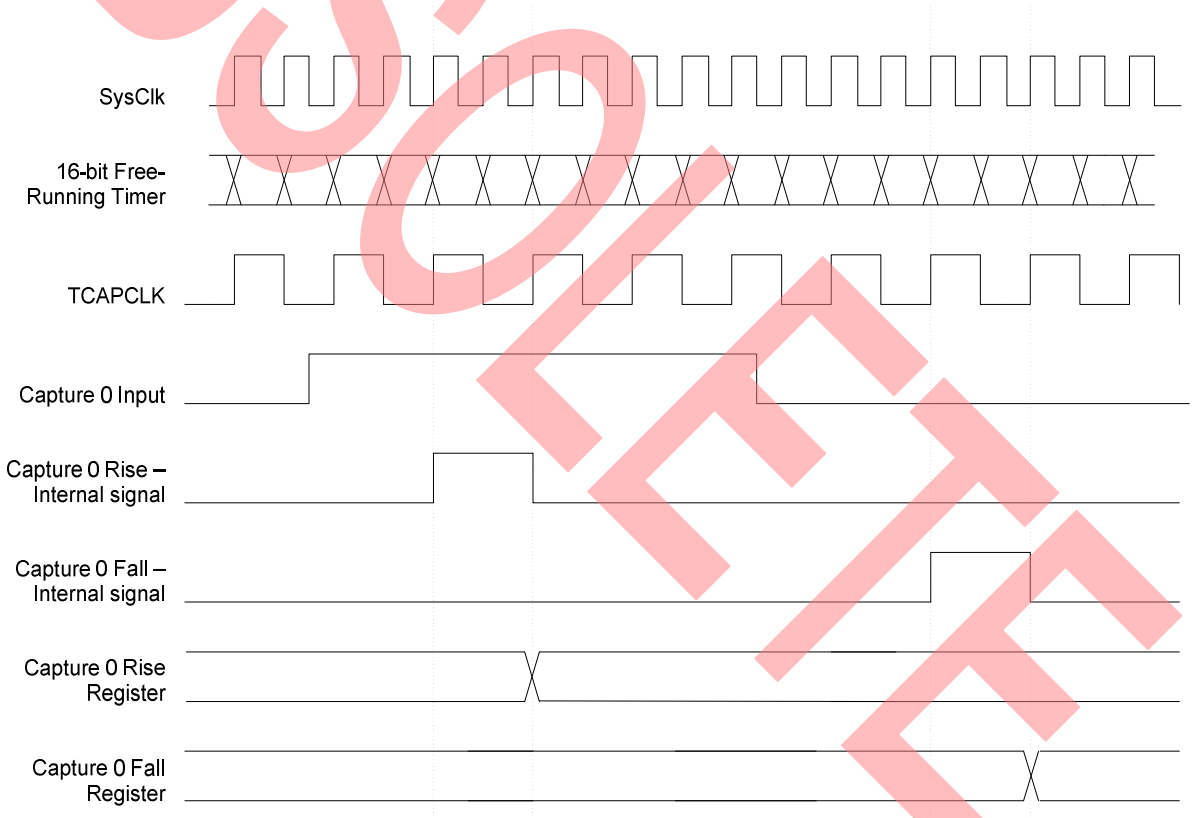
OBSOLETE

6 Capture Timer - Caveats

The way the capture timer is structured establishes some restrictions on its use. Figure 4 shows the capture timer functional timing diagram with reference to the Capture 0 input.

- From the figure, you can see that the capture timer data register is updated only on the third rising edge of the capture timer clock after the edge occurs on the input pin. Therefore, the capture timer clock imposes a restriction on the signal pulse width that can be captured.
- TCAPCLK can be clocked at a maximum frequency of 12 MHz. This limits the maximum frequency of the input signal to about 3 MHz. If the signal frequency is greater than 3 MHz, all edges may not be captured.
- If the 1-ms timer is used, then TCAPCLK is forced to 4 MHz. This further reduces the frequency of signals that can be fed to the capture timer input. You can overcome this by forcing the TCAPCLK to 24-MHz SysClk and TCAPCLK divider to '2' instead of '6' for the 1-ms timer. This causes a 1-ms timer interrupt to happen every 341.33 µs. A local variable can be incremented inside the ISR for every call and the 1-ms ISR operations performed when the variable reaches a value of '3'.
- The C compiler imposes a timing overhead (between when the edge occurs and the ISR is invoked). This is minimal (maximum of 5-µs difference when TCAPCLK is 4 MHz) and can be eliminated by using assembly code.

Figure 4. Capture Timer Functional Timing Diagram for Capture 0 Input



OBSOLETE

7 Using PSoC Designer to Set Up an enCoRe II Device for Capture Timer Operation

PSoC Designer provides a flexible environment for development and debugging.

7.1 Device Editor

PSoC Designer's Device Editor Interconnect view provides a GUI to set and configure clock and divider values for the capture timer. The desired source for the capture clock and an appropriate divider value can be chosen using the drop-down lists.

Configuration settings for the TMRCR register (Table 2 on page 6), capture edge (first or recent), and prescale bits are available through the GUI. However, this interface does not include a provision to enable the 16-bit capture.

The port pin configurations can be set in the Device Editor Interconnect view. Drop-down lists are available to select the drive and threshold modes (CMOS or TTL).

You can perform these steps in the Application Editor view.

7.2 Application Editor

The Application Editor is where firmware code is written to define the behavior of the device. This editor is used for functions such as initializing the device, enabling global interrupts, and initializing modules.

Capture Timer ISR is also written in the Application Editor. PSoC Designer provides the option of building firmware using either assembly or C language.

7.3 Debugger

The debugger provides the same view as the Application Editor, but is read-only. Using this view, code can be downloaded onto the emulation pod and run. The debugger allows you to set breakpoints and step through code.

Example capture timer projects in both assembly and C language are provided with this application note.

8 Summary

The enCoRe II and enCoRe II LV devices are widely used in the USB and wireless human-interface device (HID) market. These devices use the GUI-based tool, PSoC Designer, which makes development easy. This application note, along with the enCoRe II and enCoRe II LV datasheets, simplifies the task of configuring the device for capture timer operation.

OBSOLETE

Document History

Document Title: AN15482 - Using Capture Timers in enCoRe™ II and enCoRe II LV Devices

Document Number: 001-15482

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1082582	TYJ	05/18/2007	New application note
*A	3180633	NXZ	02/25/2011	Updated project to PD5.1 SP1. Added datasheet web links. Applied new template. Performed copy edit.
*B	3304111	CSAI	07/06/2011	No change
*C	3668235	DEJO	07/06/2012	Updated projects to PD5.2 SP1. Updated template.
*D	4854275	DEJO	07/23/2015	Updated template. Added the WirelessUSB Resources and PSoC Designer sections.
*E	5857304	ANKC	08/18/2017	Obsoleting the Spec

OBSOLETE

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

OBSOLETE

PSoC is a registered trademark and enCoRe and PSoC Designer are 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
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2007-2017. 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.