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.

AN43353

Using enCoRe V 16-Bit Timer Modules as PWMs

Author: Jacob Tomy

Associated Project: Yes

Associated Part Family: CY7C6431x, CY7C64345, CY7C6435x, CY7C604x5, CY7C60456

Related Application Notes: None

AN43353 describes how the 16-bit Timer User Module in enCoRe™ V can be leveraged to build a PWM using the part. The PSoC® Designer™ project accompanying this application note discusses the method of implementation.

Introduction

The Cypress enCoRe V device is a full-speed USB peripheral controller with configurable resources. To help you design easily, the enCoRe V development tool kit provides predefined readymade firmware code called user modules (UM). These user modules help configure the available resources to function as required.

The enCoRe V device has three 16-bit hardware timers, which can be used with the 16-bit timer user module (Timer16 User Module). The development tool, however, does not support 16-bit PWMs in the form of an additional user module. This application note explains how to build an 8-bit and 16-bit PWM from the 16-bit timer user module using the timer hardware.

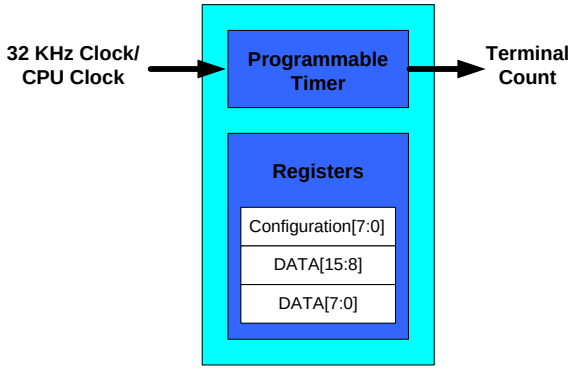
PWMs are widely used for force feedback in electronic game controllers. The enCoRe V devices are also used in gaming applications.

16-Bit Timers in enCoRe V Devices

The enCoRe V has three programmable timers, TIMER0, TIMER1, and TIMER2. The programmable timers are individually controlled 16-bit down counters.

The timers have one configuration and two data registers associated with each of them, as shown in Figure 1. The timer is started when the START bit is set in the configuration register. When started, the timers always count down from the value loaded into the data registers.

Figure 1. 16-Bit Timer Resources



The timer works in two modes:

- **Continuous mode:** Continuously repeats the countdown depending on the period set in the configuration register.
- **One shot mode:** Completes one full-count cycle and stops. The START bit in the configuration register is cleared after completing one full count cycle. Set the START bit again to restart the timer.

The programmable timer loads the value in its data registers during start and counts down to zero. TIMER0 has a terminal count output and sends a pulse for one clock cycle on reaching the terminal count. TIMER1 and TIMER2 do not have a terminal count outputs.

Clocking

The timers work with both the 32-kHz clock and the CPU clock. Use the CLKSEL bits in the respective configuration registers to select the clock. Select the clock before setting the START bit. This ensures that the clock frequency does not change when the timer is running and the timing is not affected.

As shown in Figure 1 and Figure 2, there are two main data registers and one configuration register residing in register bank0. These registers exist for each Timer16 block in enCoRe V.

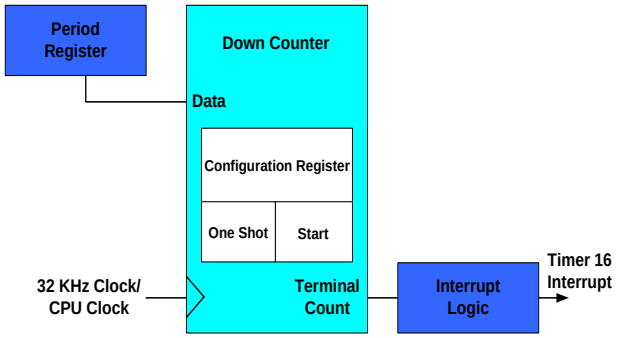
The configuration register and data registers, DATA1 and DATA0, are crucial to the functioning of the 16-bit timer.

The configuration registers store the following details about the timers:

- CPU clock to timer setting
- One shot or continuous mode setting
- ON and OFF control of the timer provided by the START bit

The data registers (or the period registers) are set with the value of counts, depending on the input clock. After the data register is set with the value, the START bit is set and the timer counts down from the value to zero.

Figure 2. 16-Bit Timer Hardware Block Diagram



Interrupt

The Timer16 interrupt is triggered when the timer counts down to zero and reloads the period value. In Timer0 this interrupt can be routed to an external GPIO. However, interrupts in Timer1 and Timer2 are only internal signals and cannot be routed to an external pin.

Registers Associated with 16-Bit Timer

The 16-bit timer is personalized and parameterized through three registers. All three 16-bit enCoRe V timer blocks have separate registers. The registers are updated when you choose options from the Chip Layout view under user module parameters. You can also edit the registers from the main user firmware code.

The following tables give the personality [1] values as constants and the parameters as named bit fields with brief descriptions for the registers.

Table 1. TimerX Configuration Register - PTx_CFG

Bit #	7	6	5	4	3	2	1	0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Value Setting	Reserved	Reserved	Reserved	Reserved	Reserved	CLKSEL	One Shot	START

CLKSEL: This bit determines whether the timer runs on a 32-kHz clock or a CPU clock. If the bit is set to 1'b1, the timer runs on the CPU clock; otherwise, the timer runs on the 32-kHz clock.

One Shot: This bit determines whether the timer runs in one shot mode or continuous mode. If the bit is set to 1'b1, the timer runs on one short mode; otherwise, the timer runs on continuous mode. In one shot mode, the timer completes one full count cycle and terminates. At termination, the START bit in this register is cleared. In the continuous mode, the timer reloads the count value every time it completes its count cycle.

START: This bit starts the timer counting from a full count. The full count is determined by the value set in the data registers. This bit is cleared on completion of a full count cycle when the timer is running in one shot mode.

[1] For further details see the enCoRe V TRM from www.cypress.com.

Table 2. TimerX Data Register1 – PTx_DATA1

Bit #	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Value Setting	Data							

These bits hold the upper eight bits of the timer's 16-bit count or period value.

Table 3. TimerX Data Register0 – PTx_DATA0

Bit #	7	6	5	4	3	2	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Value Setting	Data							

These bits hold the lower eight bits of the timer's 16-bit count or period value.

Configuring and Modifying the 16-Bit Timer to Behave as a PWM

Digital microcontroller circuits, especially those requiring modulated control signals, need configurable PWMs. You can configure the 16-bit timer of the enCoRe V along with a GPIO for PWM operations. By modifying its usage options, maintaining all its registers, and using a GPIO, you can configure the Timer 16 User Module to operate as a PWM.

Load the 16-bit timer's data register with a known value and let it run continuously. In the timer ISR, increment a variable and turn ON/OFF a GPIO based on this variable count value. This gives a pulse width-modulated output based on the interrupt frequency and the variable in the ISR.

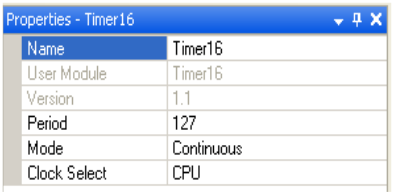
The following two steps are required to configure the 16-bit timer for PWM operation using PSoC Designer 5.0.

Configuration Settings using the PSoC Designer Chip Layout View

1. Start a PSoC Designer project with enCoRe V using C as the development language.
2. In the Chip Layout, select the Timer16 User Module and place it. PSoC Designer allows you to place the module or assign it to any of the three hardware timer blocks. Timer1 is chosen for this application note.
3. In the Global Resources setting, you can either choose the external clock or the internal main oscillator (IMO) as the SysClk source. If the IMO is used, its frequency can be configured to 6, 12, or 24 MHz. The CPU clock speed is set using the clock divider.

Figure 3 shows the user module parameters for Timer16.

Figure 3. User Module Parameters for Timer16



The period value is written to the data register from where the reload occurs on terminal count. The mode is written to the configuration register as either one shot or continuous. The clock select is written to the configuration register as either the CPU or 32-kHz clock.

The value 127 is randomly chosen for demonstration purposes. You can use any value between 1 and 65535.

4. In the User Module Properties for the Timer16 section, choose the following:
   - Clock = CPU (to select the CPU clock)
   - Mode = Continuous (to enable continuous count)
   - Period = 127
5. Set port pin P1.2 to 'Strong' drive, StdCPU select and name it PWM_Output. Port pin P1.2 is chosen as the PWM output because this is one of the pins that have only a digital I/O without any dual functions.

Application Firmware Code for PWM Operation

1. Click Generate Application and navigate to the *main.c* file under 'Source Files' folder in your project.
2. In this file:
   - Create a variable of type BYTE - pulseWidth and set it to a value of 127.
   - Enable global interrupts.
   - Enable Timer16 interrupt.
   - Start Timer16.
3. In the defined Timer16 ISR (Timer_ISR_Handler):
   - Maintain a variable of type BYTE - timerTicks.
   - Increment this variable each time the ISR is entered.
   - Check if the timerTicks variable is greater than or less than 'pulseWidth'.
   - If the timerTicks variable is less than pulseWidth, assert the GPIO pin high on Port1.1.
   - If the timerTicks variable is greater than pulseWidth, then pull the GPIO P1.1 down to 0.
4. Step 3 gives a pulse width that is a multiple of 127 CPU clocks from the 127 in the period/data register of Timer16.
5. The resolution of the PWM can be increased by:
   - Using low values for the Timer16 period. This gives an increased resolution based on the frequency of interrupts.
   - Increasing the timerTicks counts resolution to WORD (16-bits) instead of a BYTE.

Calculating PWM Timing using Timer16

The following example shows how to calculate the PWM timing using Timer16 for a 6-MHz CPU source clock. You can use the same calculation for the 32-kHz clock.

Example

If the Timer16 period and pulseWidth variable are set to 127, then for a 6-MHz clock, interrupt occurs after 127, 6-MHz clock cycles (that is, in 21.16 µs).

If pulseWidth is set to 127, the application code forces GPIO P1.0 to remain high for 127 x 21.16 µs = 2.68 ms and low for approximately the same time. This gives a 50% duty cycle.

Figure 4 shows the relation between timer clock frequency, period value, and variable timerTicks size (BYTE or WORD).

Figure 4. Parameters Influencing PWM Timing

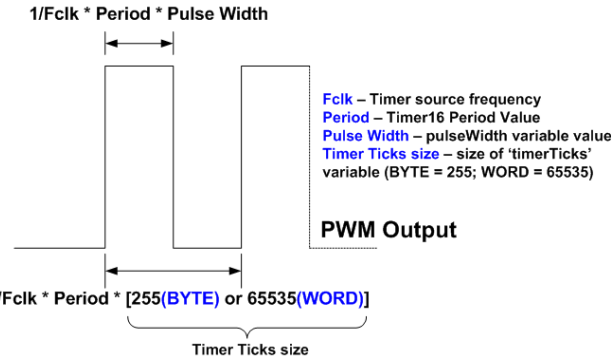
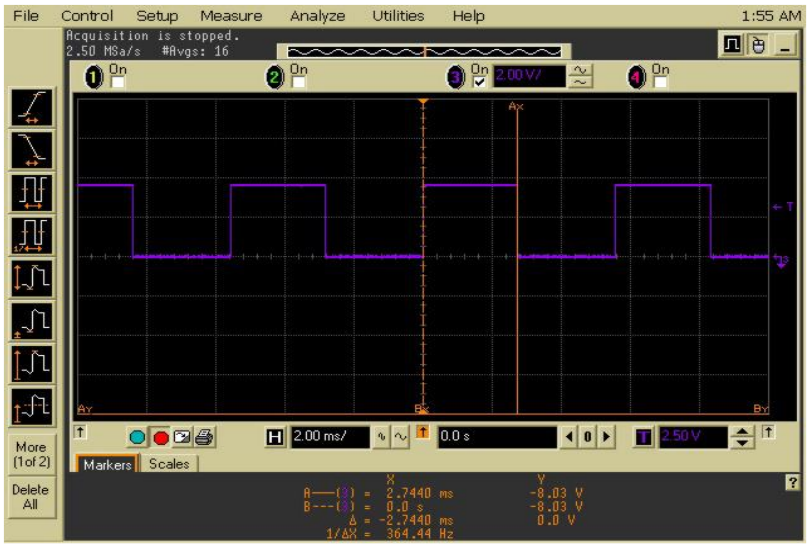


Figure 5 shows an oscilloscope capture of a PWM signal configured according to the above example.

Figure 5. Capture of PWM Signal from the Example



PWM Output Accuracy

The two different clock sources (32-kHz oscillator, CPU clock) of the Timer16 blocks are sourced by the internal low-speed oscillator (ILO) and the IMO or the external clock respectively. The PWM's output is thus dependent on the frequency of the clock source being used in its design.

If the 32-kHz clock source is chosen (and indirectly the ILO), the PWM's accuracy can vary between -40.625% and +56.25% of the expected value across the operating voltage and temperature ranges of enCoRe V and enCoRe V LV devices.

If the CPU clock source is chosen and is, in turn, driven by an external clock, the PWM's output accuracy depends on the accuracy of the externally connected clock source.

If the CPU clock is driven by the IMO, then the PWM's accuracy can vary between ±5% of the expected value across the operating voltage and temperature ranges of enCoRe V and enCoRe V LV devices. However, in enCoRe V (USB) devices, if the full-speed USB is operational, the IMO accuracy and hence the PWM output accuracy increases to ±0.25% of the expected value.

Limitations of the PWM Output

As discussed earlier, the resolution of the PWM output can be increased by using low values for the Timer16 period. This gives an increased resolution based on the frequency of interrupts.

Using too low a value at high CPU clock frequency can also cause issues.

The Port0 and Port1 GPIOs are only capable of a 12-MHz maximum frequency. Ideally, if the Timer16 Period and pusleWidth variables are set to '1' and CPU is set to 24 MHz, it results in 24 MHz at the output, if there were no maximum frequency restrictions on the GPIOs.

Even if the PWM output is slowed down to theoretically get 12 MHz at the output, in reality the output is much slower. This is because of the CPU cycles involved in taking the ISR after the interrupt occurs and switching the GPIO (PWM output).

The minimum pulse width that is achieved depends on the style of implementation, coding method (C or assembly) and the compiler used. The minimum pulse width achieved using the implementation mentioned in this application note with C language coding and Hi-Tech compiler is approximately 3.3105 µs.

Changing the PWM Pulse Width or Period

Changing the pulseWidth value changes the ON time of the constructed PWM. The firmware can monitor certain factors and change the value of the pulseWidth variable appropriately.

To change the values of the period and data registers or the configuration register of Timer16 on the fly, follow these steps in the given order:

1. Disable interrupts for Timer16
2. Stop Timer16
3. Change the data register or period value
4. Enable interrupts for Timer16
5. Start Timer16

Note Ensure that you write to the registers corresponding to the chosen timer blocks.

Summary

PWMs with 8-bit and 16-bit capabilities are used widely in digital microcontrollers that target control applications and gaming. This application note demonstrates how the available enCoRe V 16-bit timers can be configured to operate as PWMs.

Document History

Document Title: AN43353 - Using enCoRe V 16-Bit Timer Modules as PWMs

Document Number: 001-43353

Revision	ECN	Orig. of Change	Description of Change
**	2590711	10/16/2008	New application note
*A	3512322	01/30/2012	Updated template No technical updates
*B	4666641	02/20/2015	No content update. Updated format based on the template.
*C	5834804	07/27/2017	Updated Cypress logo and Copyright information.
*D	6829089	03/11/2020	Sunset review Updated description of CLKSEL and One Shot Updated template

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

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709