8-Bit Software Pulse Width Modulator Datasheet PWM8_SW V 1.00

Copyright © 2013 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C20xx6, CY8C20xx7, CY8C24x93, CY8C20065						
8-bit	1	0	0	120	4	1

Features and Overview

- 8 -bit period/compare registers
- 16-bit input clock divider
- Output clock is up to 80 kHz (for the 24-MHz CPU clock)
- 2-input clock options – ILO and CPU
- Two supported alignments – edge aligned and center aligned
- Four supported compare modes – less than, less than or equal to, greater than, and greater than or equal to
- Any GPIO pin can be used as a PWM output

Functional Description

The PWM8SW UM implements a simple counter with comparison using the Timer16 ISR. The underlying Timer16 block generates the ISR/Clock for each PWM counter step. In the ISR, the counter is incremented by 1 and the comparison condition, based on the user selection, is applied along with the user- entered compare value. For center-aligned implementation (Figure 1), after the counter reaches its period value (user defined) the counter starts to count down to 0. For edge aligned (Figure 2), the counter resets (value = 0) when its value reaches the period value. Whenever the comparison condition is met, the pin selected by the user is driven high, else driven low.

Figure 1. Operation of Center Aligned PWM

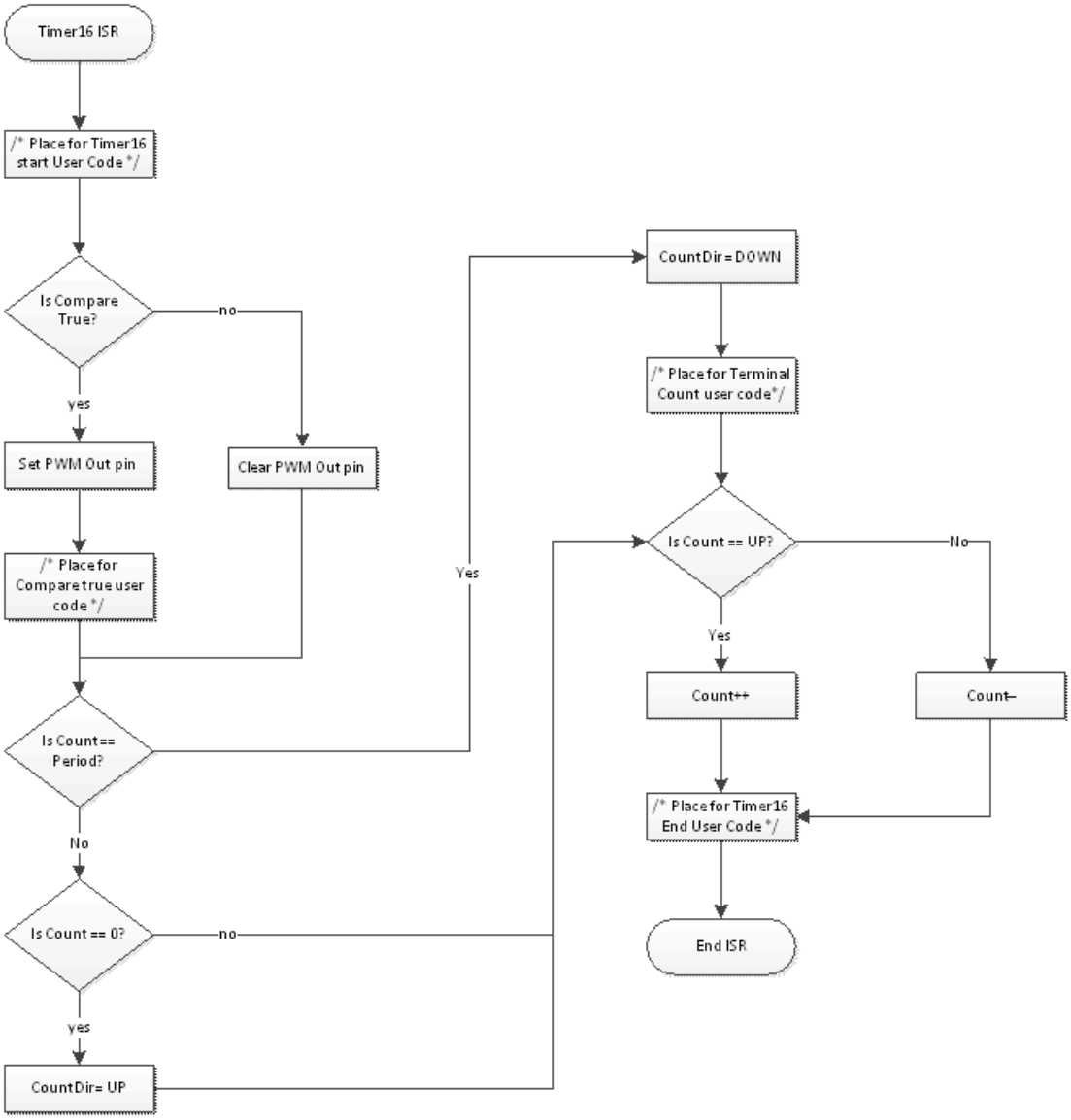
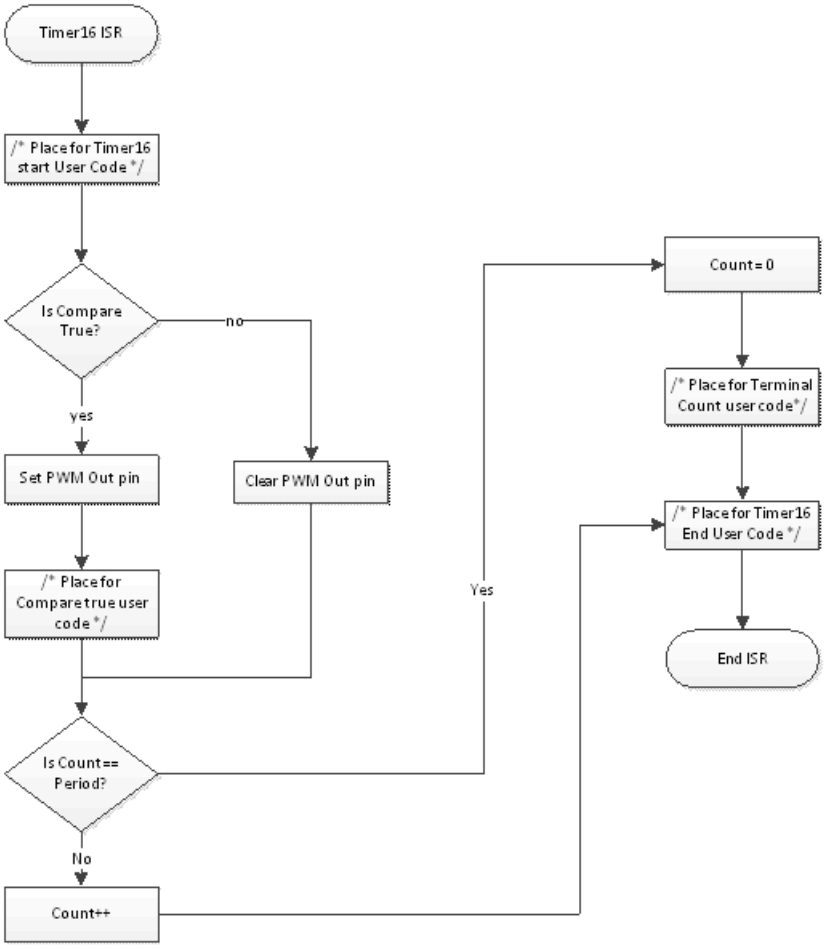
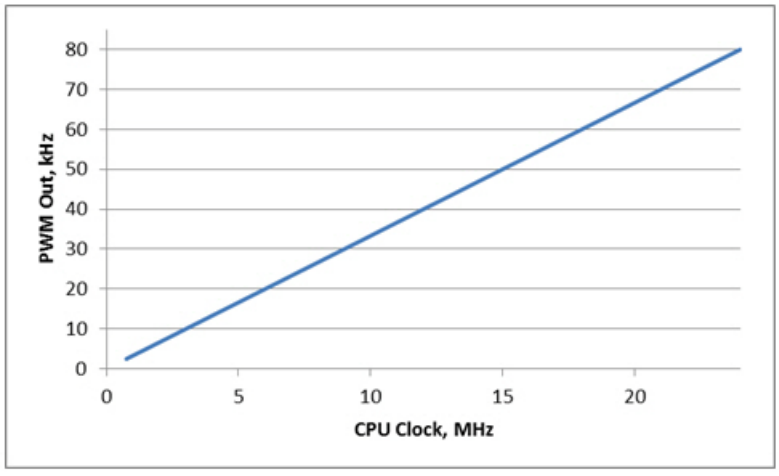


Figure 2. Operation of Edge aligned PWM



The maximum PWM Output clock versus CPU clock dependence is shown in the following figure.



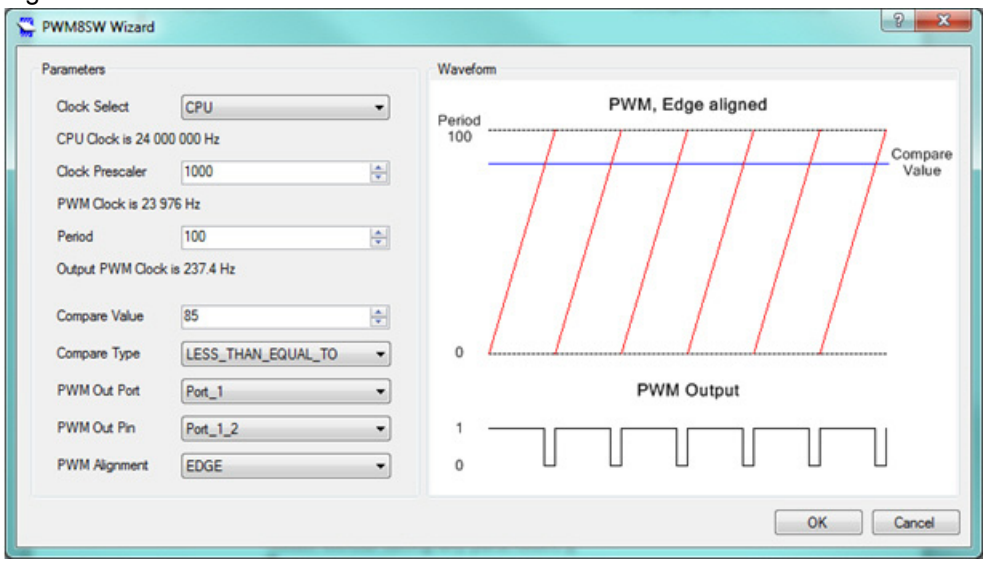
Wizard Requirements

The user module wizard helps you to configure PWM. To access the Wizard, right click the user module in the **Workspace Explorer** or the **Chip Editor** and select the "PWM8SW Wizard...".

All the user module parameters are present in the UM Parameters window. No additional parameters are present. The wizard displays a lot of helpful information that cannot be shown in the UM Parameter window. With this information, the user can configure the PWM8SW User Module quicker because there is no need to perform any calculations of used and produced clocks. All needed calculations are done by the Wizard.

Wizard View

Figure 3. Wizard View of the PWM8SW User Module



Based on the information entered in the user parameters, the Wizard calculates the PWM Input and Output clocks and graphically shows the PWM Output signal.

Wizard Parameters

The wizard contains all the parameters that are present in the UM parameter window. These parameters are described in the Parameters and Resources section in detail. The preferred method of setting UM parameters is through the Wizard.

Wizard Buttons

There are three buttons on the wizard:

- "OK" - this button checks parameters and if the parameters are correct, the Wizard saves the parameters and closes. If the parameters are incorrect, the Wizard shows the appropriate warning message.
- "Cancel" - this button closes the Wizard without saving the parameters.
- "Close" - this is a standard window close button located at the top right corner of the Wizard form. If the user clicks this button, the Wizard closes without saving any parameters.

Parameters and Resources

Clock Select

This parameter selects the input clock source for the PWM.

Type	ENUM
Range	ILO, CPU
Default	CPU

Affected registers: PTx_CFG

Clock Prescaler

This parameter sets the clock divider value for PWM's input clock.

Type	WORD
Range	149-65535 for CPU clock Select 1-65535 for ILO clock Select
Default	1000

Dividers that have less number of cycles than the ones used to serve the Timer16 ISR are removed.

Affected registers: PTx_DATA0 and PTx_DATA1

Period

This parameter sets PWM period.

Type	BYTE
Range	1-255
Default	100

Compare Value

This parameter sets the compare value for the PWM.

Type	BYTE
Range	1-Period
Default	1

The parameter value cannot exceed value of the Period parameter.

Compare Type

This parameter sets the compare condition for the PWM.

Type	ENUM
Range	LESS_THAN LESS_THAN_EQUAL_TO GREATER_THAN GREATER_THAN_EQUAL_TO
Default	LESS_THAN_EQUAL_TO

PWM_Out_Port

This parameter sets the PWM Output port.

Type	ENUM
Range	All available ports
Default	None

Affected registers: PRTxDR, PRTxDM0, PRTxDM1

PWM_Out_Pin

This parameter sets the PWM Output pin.

Type	ENUM
Range	All available port pin options
Default	None

Affected registers: PRTxDR, PRTxDM0, PRTxDM1

PWM Alignment

This parameter sets the PWM alignment.

Type	ENUM
Range	EDGE CENTER
Default	EDGE

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This sections specifies the interface to each function together with related constants provided by the “include” files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This “registers are volatile” policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API functions may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

PWM8SW_Start

Description:

Starts the PWM by setting the PWM period, compare values, and then starts the Timer16 block and enables its ISR.

C Prototype:

```
void  PWM8SW_Start(void)
```

Assembly:

```
lcall  PWM8SW_Start
```

Parameters:

None

Return Value:

None

PWM8SW_Stop

Description:

Stops the PWM block by disabling the Timer16 block and its ISR with the default value.

C Prototype:

```
void  PWM8SW_Stop(void)
```

Assembly:

```
lcall  PWM8SW_Stop
```

Parameters:

None

Return Value:

None

PWM8SW_WritePeriod

Description:

Sets the PWM period value.

C Prototype:

```
void  PWM8SW_WritePeriod(BYTE bPeriod)
```

Assembly:

```
mov   A, bPeriod
lcall  PWM8SW_WritePeriod
```

Parameters:

bPeriod: bPeriod value is a value from 1 to 255.

Return Value:

None

PWM8SW_WriteCompare

Description:

Sets the PWM Compare value.

C Prototype:

```
void  PWM8SW_WriteCompare(BYTE bCompare)
```

Assembly:

```
mov   A, bCompare
lcall  PWM8SW_WriteCompare
```

Parameters:

bCompare: 1-255; if greater than Period, the value is forced to period value.

Return Value:

None

PWM8SW_bReadCounter

Description:

Returns the current PWM counter count.

C Prototype:

```
BYTE  PWM8SW_bReadCounter(void)
```

Assembly:

```
lcall  PWM8SW_bReadCounter
```

Parameters:

None

Return Value:

BYTE PWM Counter value.

Sample Firmware Source Code

The following C sample code demonstrates the PWM8SW UM functionality:

```
//-------------------------------------------------------------------------
```

```
// C main line
//---------------------------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main(void)
{
    M8C_EnableGInt;           // Enable Global Interrupts
    PWM8SW_Start();           // Start software PWM
    PWM8SW_WritePeriod(199);  // Set period value
    PWM8SW_WriteCompare(99);  // Set compare value
}
```

The following is the equivalent code written in assembly:

```
;---------------------------------------------------------------------------
; Assembly main line
;---------------------------------------------------------------------------

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:

    M8C_EnableGInt          ; Enable Global Interrupts

    lcall PWM8SW_Start      ; Start software PWM

    mov A, 199
    lcall PWM8SW_WritePeriod ; Set period value

    mov A, 99
    lcall PWM8SW_WriteCompare      ; Set compare value

.terminate:
    jmp .terminate
```

Configuration Registers

The Timer PSoC block registers used to configure this user module are described here:

Table 1. Block Timer16, Configuration Register PTx_CFG

Bit	7	6	5	4	3	2	1	0
Value	Reserved					CLKSEL	–	START

CLKSEL - This bit determines if the timer runs the ILO clock or CPU clock. If the bit is set, the timer runs on the CPU clock; if not, the timer runs on the ILO clock.

START - This bit starts the timer counting from a full count. The full count is determined by the value loaded into the DATA registers. In the one shot mode, this bit is cleared when the timer is running and a full count cycle completes.

Table 2. Block Timer16, Data Register 1, PTx_DATA1

Bit	7	6	5	4	3	2	1	0
Value	Data							

These bits hold the upper 8 bits of the timer's 16-bit period value.

Table 3. Block Timer16, Data Register 0, PTx_DATA0

Bit	7	6	5	4	3	2	1	0
Value	Data							

These bits hold the lower 8 bits of the timer's16-bit period value.

Version History

Version	Originator	Description
1.00	HPHA	Initial version

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2013 Cypress Semiconductor Corporation. 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.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any 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.