AN99235

FM4 S6E2HG Series MCU – 16-Bit PWM Using a Base Timer

Author: Arthur Zhong
Associated Part Family: FM4
Associated Code Examples: None
Related Application Notes: None

AN99235 discusses how to configure the base timer of an S6E2HG (FM4) series MCU to generate a PWM signal with a variable period and duty cycle. A code example using FM4-120L-S6E2HG starter-kit is included to demonstrate this operation.

Contents

1 Introduction 1
2 PWM Principle 1
Set of Registers 2
2.1 Base Timer Mode Select 2
2.2 Pin Allocation for PWM 2
2.3 Clock for PWM 3
2.4 PWM Control Register 4
3 PWM Configuration 7
3.1 Configuration Flow Chart 7
3.2 PWM Operation Mode 8
Summary 16
Document History 17
Worldwide Sales and Design Support 18
Products 18
PSoC® Solutions 18
Cypress Developer Community 18
Technical Support 18

1 Introduction

Pulse width modulation (PWM) is a general function for house-hold and industry applications such as power management, lighting, and motor control.

Most ARM® Cortex®-M4 based microcontrollers have a base timer that can be used to generate a PWM signal.

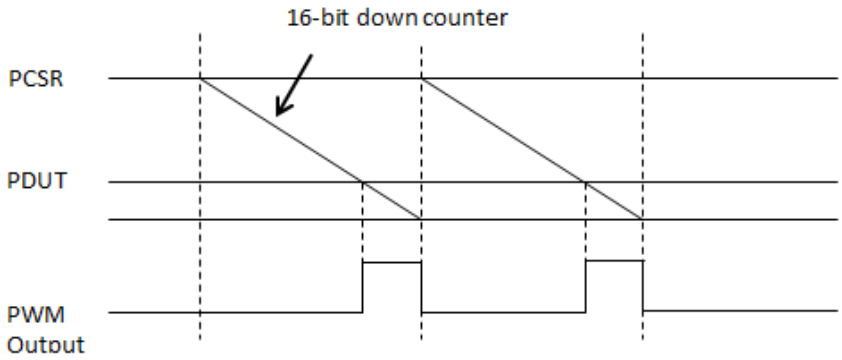
This application note discusses how to use the base timer of an S6E2HG (FM4) series MCU to generate a PWM signal with a variable period and duty cycle. This application note covers the principle of PWM generation, definition of related registers with base timer, flow of configuring the PWM timer mode, and a code example of configuring the running modes (one-shot mode, continuous mode, restart enable mode, restart disable mode, etc). The sample code and illustrations based on the starter-kit (FM4-120L-S6E2HG) are included to demonstrate this operation.

2 PWM Principle

Each channel of the base timer can be used as one of the four kinds of timers: 16-bit PWM timer, 16-bit PPG timer, 16- or 32-bit reload timer and 16-or 32-bit PWC timer. In this application note, 16-bit PWM timer mode is used to generate a PWM signal.

The generation of a PWM signal using the base timer of an S6E2HG series MCU works by comparing the register values with the 16-bit down counter. The period of PWM is determined by the PCSR register, which stores the initial value of 16-bit down counter. When the counter value is equal to zero, an underflow interrupt is occurred, and the value in PCSR is transferred to the 16-bit down counter. The PWM duty cycle is determined by the PDUT register. When the counter value is equal to the value in PDUT, the polarity of the PWM waveform is inverted.

Figure 1. PWM Principle Explanation



Set of Registers

2.1 Base Timer Mode Select

The base timer function can be set as a 16-bit PWM timer, 16-bit PPG timer, 16/32-bit reload timer, or a 16/32-bit PWC timer using the FMD2, FMD1, and FMD0 bits in the Timer Control Register (TMCR). Table 1 provides an overview of each selectable timer function.

Table 1.Relationship Between Mode Setting and Timer Functions

Settings of FMD2,FMD1, And FMD0 bits	Function
0b000	Reset mode
0b001	16-bit PWM timer
0b010	16-bit PPG timer
0b011	16/32-bit reload timer
0b100	16/32-bit PWC timer

The PWM timer consists of a 16-bit down counter, a 16-bit data register with a period set buffer, a 16-bit compare register with a duty cycle set buffer, and a pin controller.

The period and duty cycle data are in a buffered register and therefore can be rewritten while the timer is operating.

When triggered, the 16-bit PWM timer starts countdown from the period set value. First, it outputs a LOW pulse. When the 16-bit down counter matches the value set in the PWM Duty Set Register, the output inverts to HIGH. After this, the output inverts again to LOW when a counter underflow occurs. This sequence can generate waveforms with any period and duty cycle.

2.2 Pin Allocation for PWM

Select the pin and its corresponding base timer. This example uses the pin P3A (TIOA0_1); therefore, set the output for TIOA0 as described in Table 2.

Table 2. [bit 3:2] TIOA0E: TIOA0 Output Select Bit

bit3:2	Description
Reading	Reads out the register value
Writing 00	Does not produce output for BT-ch0-TIOA (initial value)
Writing 01	Uses TIOA0_0 as the output pin of BT-ch0-TIOA
Writing 10	Uses TIOA0_1 as the output pin of BT-ch0-TIOA
Writing 11	Uses TIOA0_2 as the output pin of BT-ch0-TIOA

2.3 Clock for PWM

Perform the following sequence:

1. Set HCLK.
2. Set PCLK1 (which is connected to HCLK).
3. Set the PWM timer (which is connected to PCLK1).

- Set the HCLK (base timer). HCLK is a clock for the macro connected to the AHB bus. The clock frequency can be set between 1/1 and 1/16 of the frequency of the master clock.

  The following code sets the base clock frequency to 80 MHz:

  ```
  FM_CRG->BSC_PSR_f.BSR = 1u;
  ```

  The Base Clock Prescaler Register (BSC_PSR)

  Set the frequency division ratio of the base clock per Table 3.

Table 3. [bit 2:0] BSR: Base Clock Frequency Division Ratio Setting Bit

Bit2	Bit1	Bit0	Description
0	0	0	1/1
0	0	1	½
0	1	0	1/3
0	1	1	¼
1	0	0	1/6
1	0	1	1/8
1	1	0	1/16
1	1	1	Reserved

- Set the APB1 bus clock (PCLK1). PCLK1 is a clock for the macro connected to the base timer bus. The clock frequency can be set between 1/1 and 1/8 of the frequency of the base clock.

  The following code sets the APB1 bus clock frequency to 20 MHz:

  ```
  FM_CRG->APBC1_PSR_f.APBC1 = 2u;
  ```

  APBC1_PSR sets the APB1 bus clock frequency division ratio as listed in Table 4.

Table 4. [bit 1:0] APBC1: APB1 Bus Clock Frequency Division Ratio Setting Bit

Bit2	Bit1	Description
0	0	1/1(Initial value)
0	1	½
1	0	¼
1	1	1/8

- The Timer Control Register (TMCR) controls the PWM timer. Set bit14:12 CKS3 to CKS0 (count clock selection bit).

  The following code sets the PWM clock frequency to 5MHz:

  ```
  TMCR_f.CKS = 1u;
  ```

Select the count clock for the 16-bit counter, as described in Table 5.

Table 5. [bit 14:12, TMCR2: bit8] CKS3 to CK0

CKS3	CKS2	CKS1	CKS0	Description
0	0	0	0	PCLK1
0	0	0	1	PCLK1/4
0	0	1	0	PCLK1/16
0	0	1	1	PCLK1/128
0	1	0	0	PCLK1/256
0	1	0	1	External clock (rising-edge event)
0	1	1	0	External clock (falling-edge event)
0	1	1	1	External clock (rising-and-falling edge event)
1	0	0	0	PCLK1/512
1	0	0	1	PCLK1/1024
1	0	1	0	PCLK1/2048
Others				Setting disabled

2.4 PWM Control Register

2.4.1 Timer Control Registers (TMCR)

Timer control register (TMCR) determines the function of base timer, sets the polarity of PWM output, sets the output as continuous pulse or one-shot pulse, enables or disables restart, and so on..

- The following code sets the PWM continuous pulse output:

```
TMCR_f.MDSE = 0u;
```

  The mode selection bit (MDSE) selects continuous pulse output or one-shot pulse output.

- The following code disables the external trigger:

```
TMCR_f.EGS =  0u;
```

  [bit 9:8] EGS1 and EGS0 are trigger input edge selection bits.

  These bits select a valid edge for input waveforms as an external start cause and set the trigger condition, as summarized in Table 6.

  When the initial value or "0b00" is set, the timer is not started by external waveforms because this setting indicates that no valid edge is selected for input waveforms.

Table 6. [bit 9:8] EGS1, EGS0: Trigger Input Edge Selection Bits

Bit8	Bit9	Description
0	0	Trigger input disabled
0	1	Rising edge
1	0	Falling edge
1	1	Both edges

- The following code sets the normal output mode:

```
TMCR_f.PMSK = 0u;
```

The pulse output mask bit (PMSK) controls the output level of the PWM output waveforms, as described in Table 7.

Note: When output polarity specification bit (OSEL) of Timer Control Register (TMCR) is set to inverted output, setting PMSK to "1" masks the output with HIGH.

Table 7. [bit 10] PMSK: Pulse Output Mask Bit

Bit	Description
0	Normal polarity
1	Inverted polarity

- The following code sets the output polarity, as described in Table 8:

  TMCR_f.OSEL = 0u;

Table 8. [bit 3] OSEL: Output Polarity Specification Bit

Bit	Polarity	After Reset	Match in Duty	Underflow
0	Normal	LOW output		
1	Inverted	HIGH output		

- The following code sets restart enable as listed in Table 9:

  TMCR_f.RTGEN = 1u;

  The restart enable bit (RTGEN) enables restart by a software trigger or trigger input.

Table 9. [bit 11] RTGEN: Restart Enable Bit

Bit	Description
0	Restart disabled
1	Restart enabled

- The following code enables the count operation:

  TMCR_f.CTEN = 1u;

  The count operation enable bit (CTEN) enables the operation of the down counter.

  When the counter is in an operation-enabled status (the CTEN bit is ‘1’), writing ‘0’ to this bit stops the counter.

- The following code starts software triggering:

  TMCR_f.STRG = 1u;

  STRG: is the software trigger bit.

  When the CTEN bit is ‘1’, writing ‘1’ to the STRG bit enables software triggering.

  The read value of the STRG bit is always ‘0’.

2.4.2 PWM Cycle Set Register (PCSR)

- The following code sets the PWM period:

  ((volatile FM_BT_PWM_TypeDef*) FM_BT0_PWM_BASE)->PCSR = 999; /*PWM period = (1+999) * 0.2us = 200us*/

  The PWM Cycle Set Register (PCSR) is a buffered register for setting the PWM period. Transfer to the Timer Register is performed at startup or underflow.

  When initializing or rewriting the PWM Cycle Set Register, be sure to write to the PWM Duty Set Register after writing to the PWM Cycle Set Register.

  Access the PCSR register with 16-bit data.

  Set the period using PCSR register after setting the PWM function using the FMD2, FMD1, and FMD0 bits in the TMCR register.

2.4.3 PWM Duty Set Register (PDUT)

- The following code sets the duty cycle:

  ((volatile FM_BT_PWM_TypeDef*) FM_BT0_PWM_BASE)->PDUT = 499; /*Duty cycle = (1+499) / (1+999) = 50%*/

  The PWM Duty Set Register (PDUT) is a buffered register for setting the duty cycle. Transfer from the buffer is performed at an underflow.

  When the value of PDUT is equal to the value of PCSR, an all-HIGH pulse is output under normal polarity and an all-LOW pulse is output under inverted polarity.

  Do not set a value that makes PCSR < PDUT. The PWM output becomes undefined.

  Access the PDUT register with 16-bit data.

  Set the duty cycle using PDUT register after setting the PWM function using the FMD2, FMD1, and FMD0 bits in the TMCR register.

  Calculate the real value of period and duty cycle: (PWM clock frequency = 5 MHz, PWM clock cycle = 0.2 µs):

  - PWM period = (1 + m) * PWM clock cycle = 200 µs (m: PCSR value, configure it to 999 in the sample project)
  - Duty cycle = (1 + n) * PWM clock cycle / PWM period = 50% (n: PDUT value, configure it to 499 in sample project)

  Note: Setting the PDUT register to ‘0’ cannot get LOW output, To get LOW output, consider using the PMSK bit in TMCR register (set PMSK to 1).

2.4.4 Status Control Register (STC)

- The following syntax sets the interrupt enable:

  STC_f.TGIE = 1u; /* TGIE is the trigger interrupt request enable bit */

  STC_f.DTIE = 1u; /*DTIE is the duty match interrupt request enable bit*/

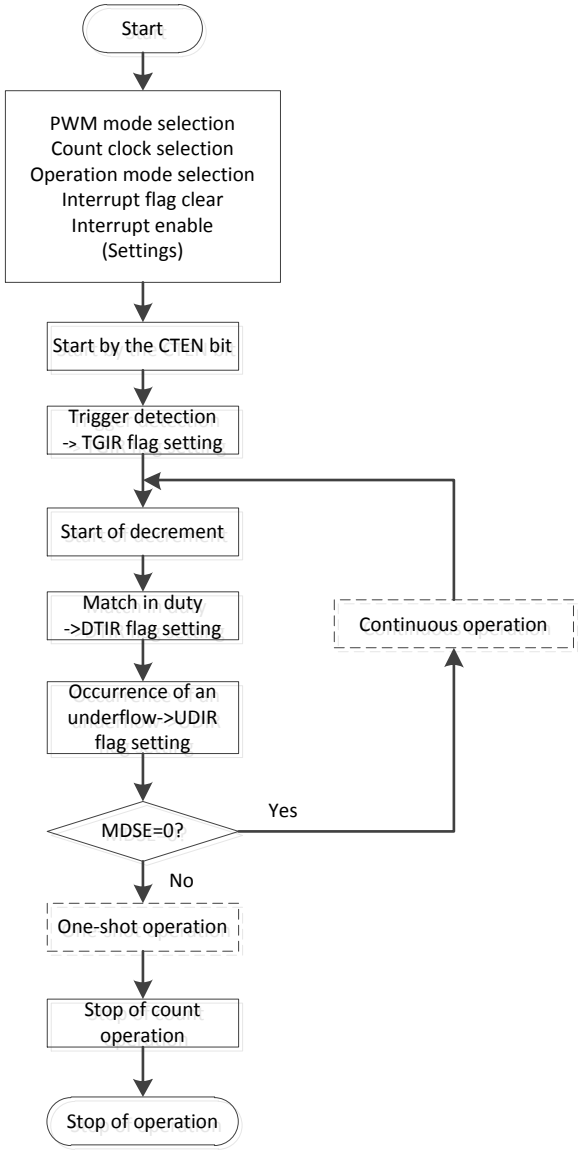
  STC_f.UDIE = 1u; /*UDIE is the underflow interrupt request enable bit*/

3 PWM Configuration

3.1 Configuration Flow Chart

This section presents the flow chart of the PWM timer operation as Figure 2, and then introduces the four PWM modes, lists the register to be used, and provides the code to configure the register.

Figure 2. PWM Timer Operation Flow Chart



3.2 PWM Operation Mode

3.2.1 Continuous Operation, Restart Enable Mode

```
PwmInitOutput();          // Set basetimer IO port

// Initialize interrupt
stcPwmConfig.pstcPwmIrqEn = &stcIrqEn;
stcPwmConfig.pstcPwmIrqCb = &stcIrqCb;
stcPwmConfig1.pstcPwmIrqEn = &stcIrqEn;
stcPwmConfig1.pstcPwmIrqCb = &stcIrqCb;

Bt_ConfigIOMode(&USER_BT, BtIoMode0);   // Set requested I/O mode

// Initialize PWM timer
stcPwmConfig.enPres = PwmPres1Div4;  // PWM clock = 5MHz @ PCLK=20MHz
stcPwmConfig.enMode = PwmContinuous;    //Continuous mode
stcPwmConfig.enExtTrig = PwmExtTrigDisable;   //Software trigger
stcPwmConfig.enOutputMask = PwmOutputNormal;   // Normal polarity
stcPwmConfig.enOutputPolarity = PwmPolarityLow;   // Low polarity
stcPwmConfig.enRestartEn = PwmRestartEnable;   // Enable restart mode
stcPwmConfig.pstcPwmIrqEn->bPwmTrigIrq = 1; //trigger interrupt request enable
stcPwmConfig.pstcPwmIrqEn->bPwmDutyMatchIrq = 1; //duty match interrupt request enable
stcPwmConfig.pstcPwmIrqEn->bPwmUnderflowIrq = 1; // underflow interrupt request enable
stcPwmConfig.bTouchNvic = TRUE; // NVIC(Nested Vectored Interrupt Controller) setting

Bt_Pwm_Init(&USER_BT , &stcPwmConfig); // Initialize PWM function of BT
// Set cycle and duty value
Bt_Pwm_WriteCycleVal(&USER_BT, 999); // Cycle = (1+m)*PWM clock cycle = 200us
Bt_Pwm_WriteDutyVal(&USER_BT, 499);  // Duty = (1+m)*PWM clock cycle = 40us

Bt_Pwm_EnableCount(&USER_BT); // Enable count operation
Bt_Pwm_EnableSwTrig(&USER_BT); // Start triggered by software
```

From this code (for the configuration of the registers), you get a 5-kHz and fifty-percent-duty-cycle PWM from pin P3A, as shown in Figure 3 and Figure 4. Thus, PWM works at continuous pulse output and in enable restart mode.

Figure 3. 5-kHz, Fifty-Percent Duty Cycle PWM from Pin P3A (Period = 200 µs)

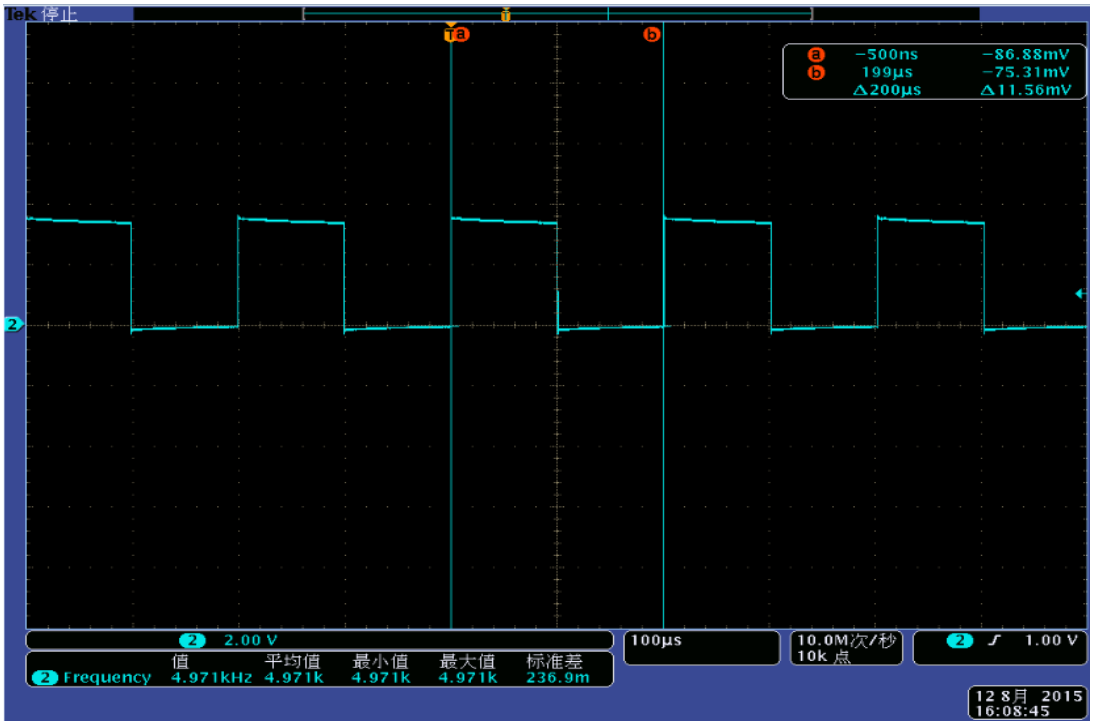
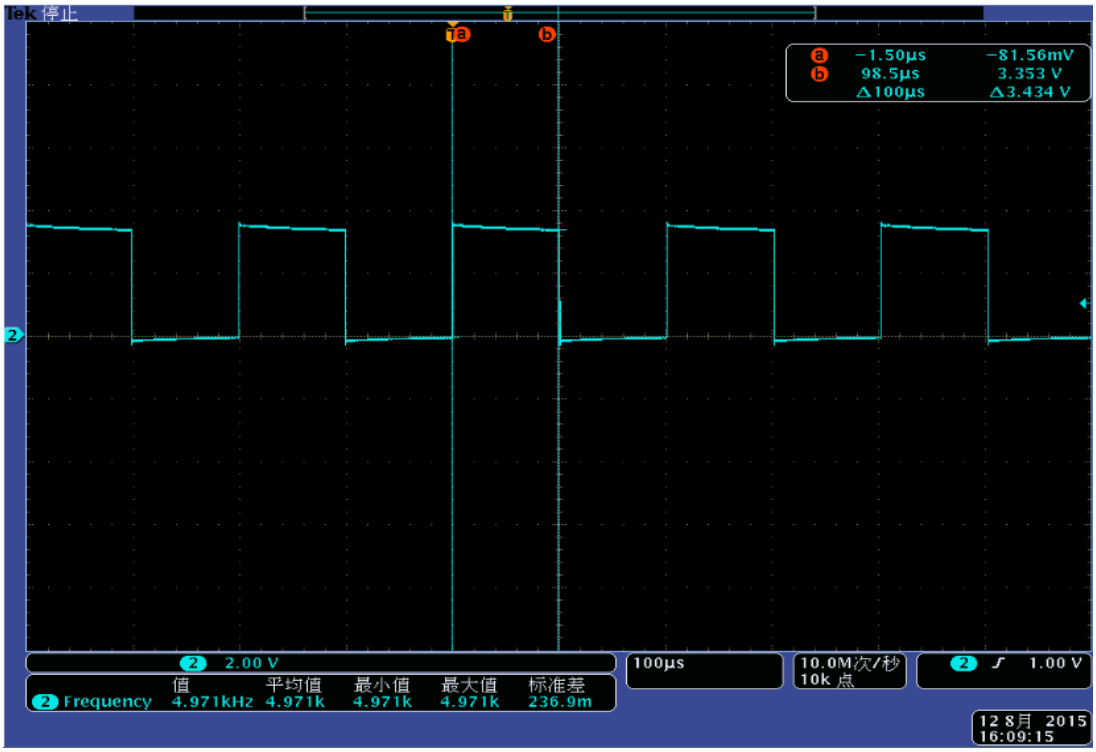


Figure 4. 5kHz Fifty Percent Duty Cycle PWM from Pin P3A (Duty Cycle = 50%)



In this mode, if a trigger occurs twice and retriggering occurs during operation, the waveform appears as shown in Figure 5.

Figure 5. PWM Operation Timing Chart (Continuous Operation, Restart Enable Mode)

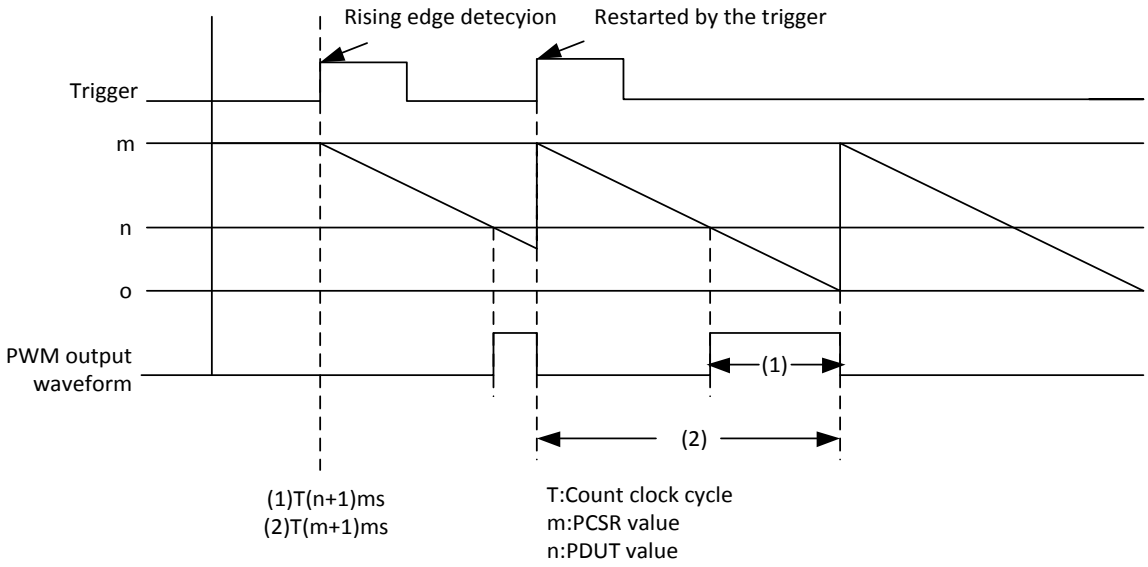
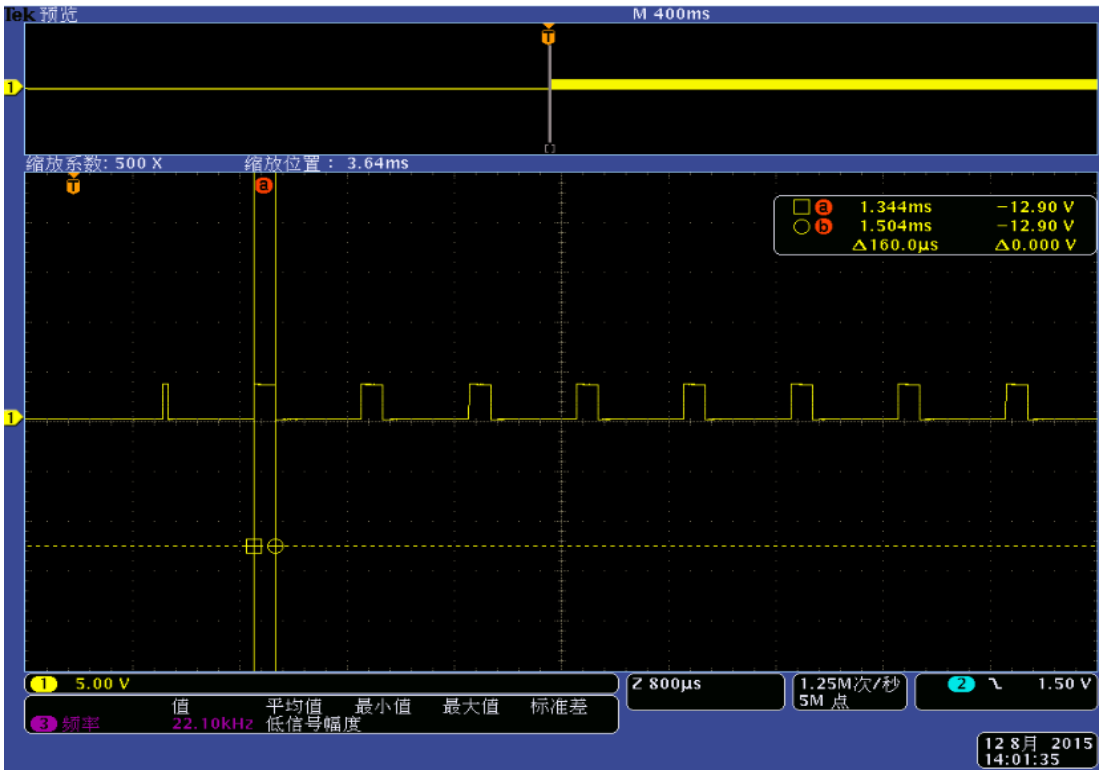


Figure 6 illustrates the waveform if the timer is restarted by triggering in continuous operation.

Figure 6. P3B in Continuous Operation, Restart Enable Mode (Duty Cycle = 20%)



3.2.2 Continuous Operation, Restart Disable Mode

If you want to select "restart disable" mode, configure the restart enable bit (RTGEN) as shown in the code below. Keep the other code unchanged.

```
stcPwmConfig.enRestartEn = PwmRestartDisable;   // Restart disable mode
```

In this mode, if trigger occurs twice and retriggering occurs during operation, the waveform appears as shown in Figure 7.

Figure 7. PWM Operation Timing Chart (Continuous Operation, Restart Disable Mode)

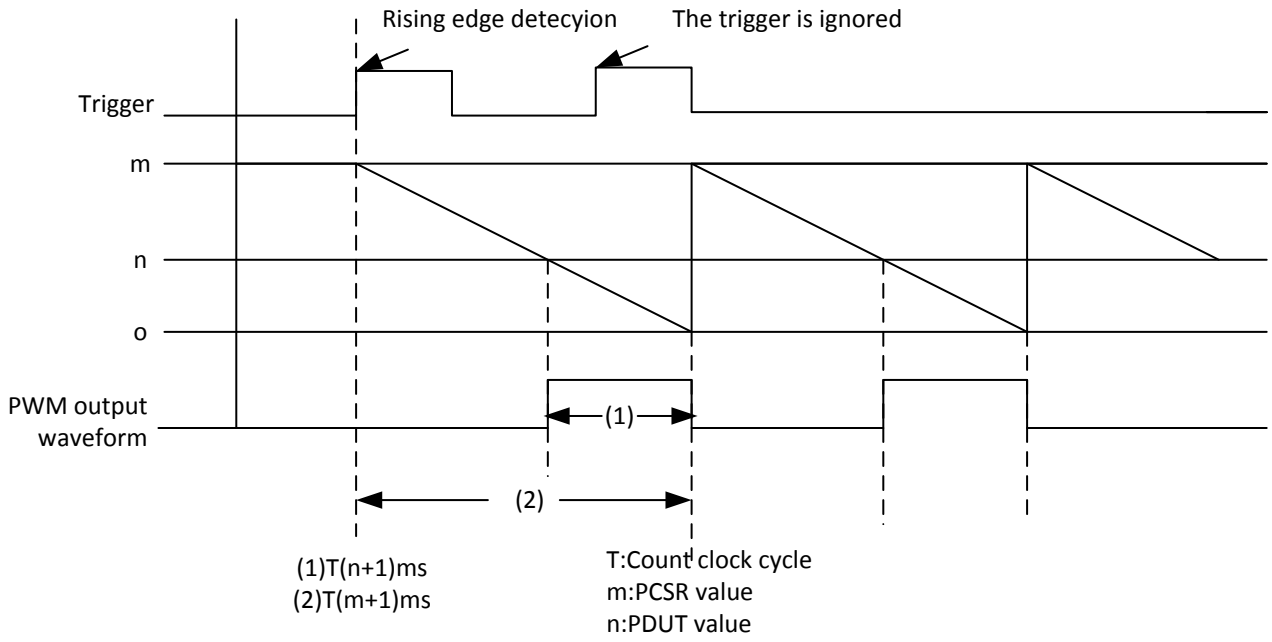
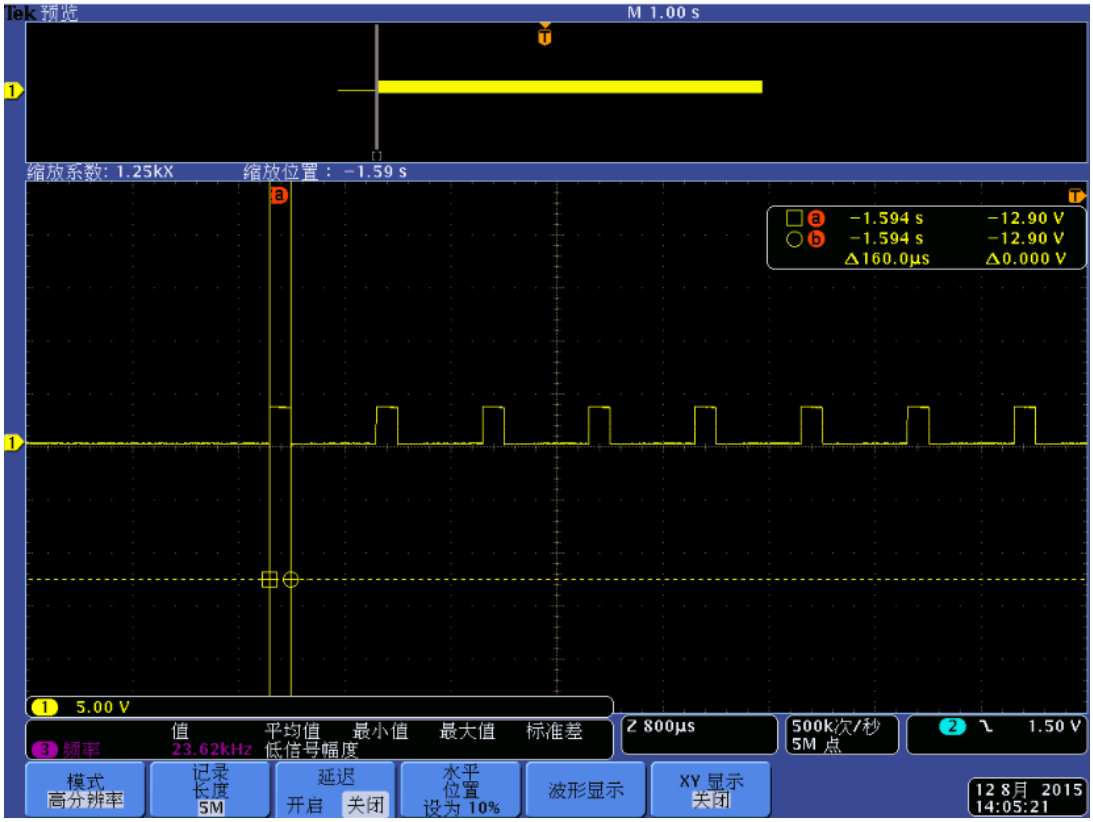


Figure 8 illustrates the waveform if the timer is triggered twice and restart is disabled in continuous operation.

Figure 8. P3B in Continuous Operation, Restart Disable Mode (Duty Cycle = 20%)



3.2.3 One-Shot Operation, Enable Restart Mode

If you want to select "on-shot" mode, configure the mode selection bit (MDSE) per the code below. Keep the other code unchanged.

```
stcPwmConfig1.enMode = PwmOneshot; // One-shot mode
```

In one-shot operation, a single pulse of any width can be output using a trigger. When restart is enabled, the counter is reloaded when an edge is detected during operation.

In this mode, if trigger occurs twice and retriggering occurs during operation, the waveform appears as shown in Figure 9.

Figure 9. One-Shot Operation Timing Chart (One-shot Operation, Restart Enable Mode)

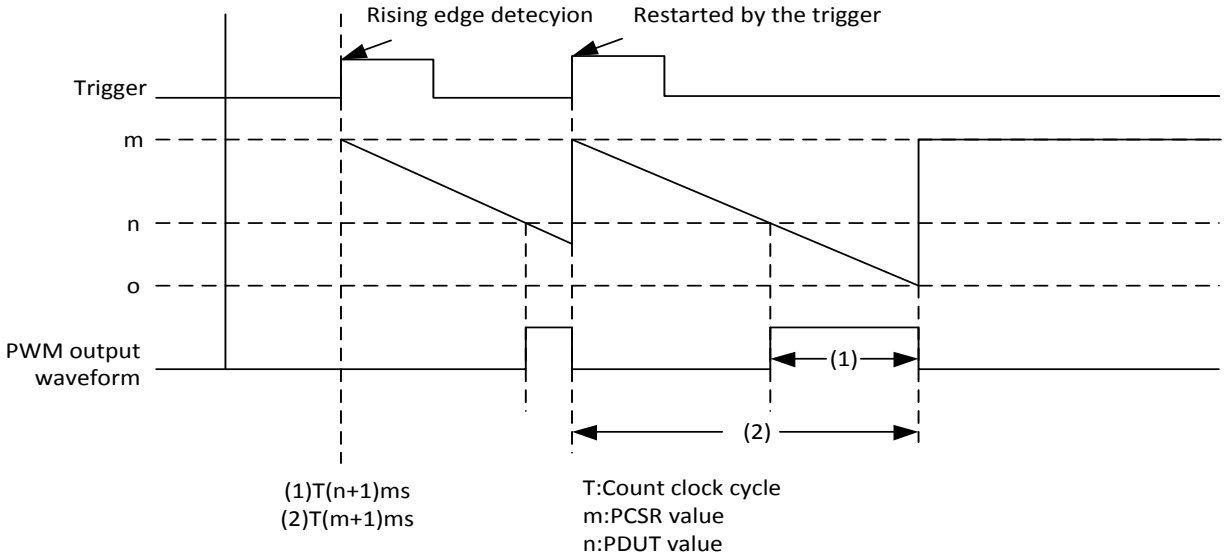
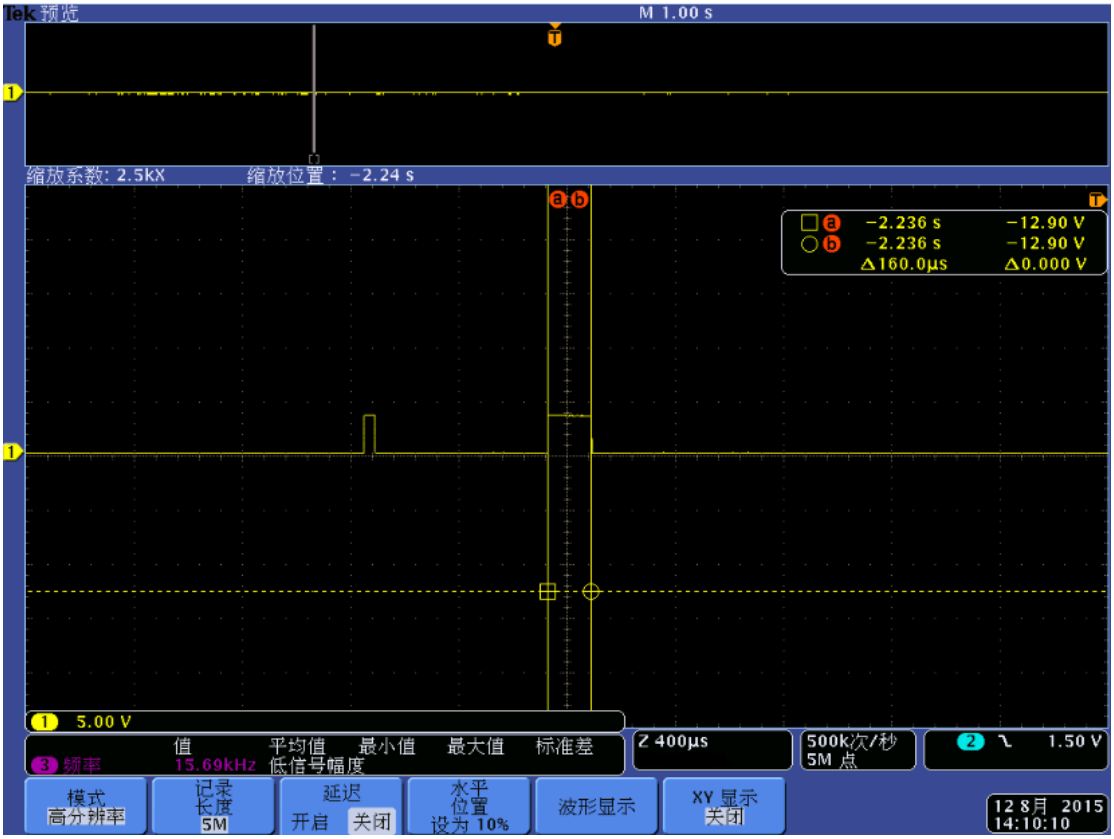


Figure 10 illustrates the waveform if the timer is triggered twice and restarted during one-shot operation.

Figure 10. P3B in One-Shot Operation, Restart Enable Mode (Duty Cycle = 20%)



3.2.4 One-Shot Operation, Disable Restart Mode

If you want to select "one-shot, restart disable" mode, configure the mode selection bit (MDSE) and restart enable bit (RTGEN) as below. Keep the other code unchanged.

```
stcPwmConfig.enMode = PwmOneshot; // One-shot mode
stcPwmConfig.enRestartEn = PwmRestartDisable;  // Disable restart mode
```

Figure 11. One-Shot Operation Timing Chart (One-Shot Operation, Restart Disable Mode)

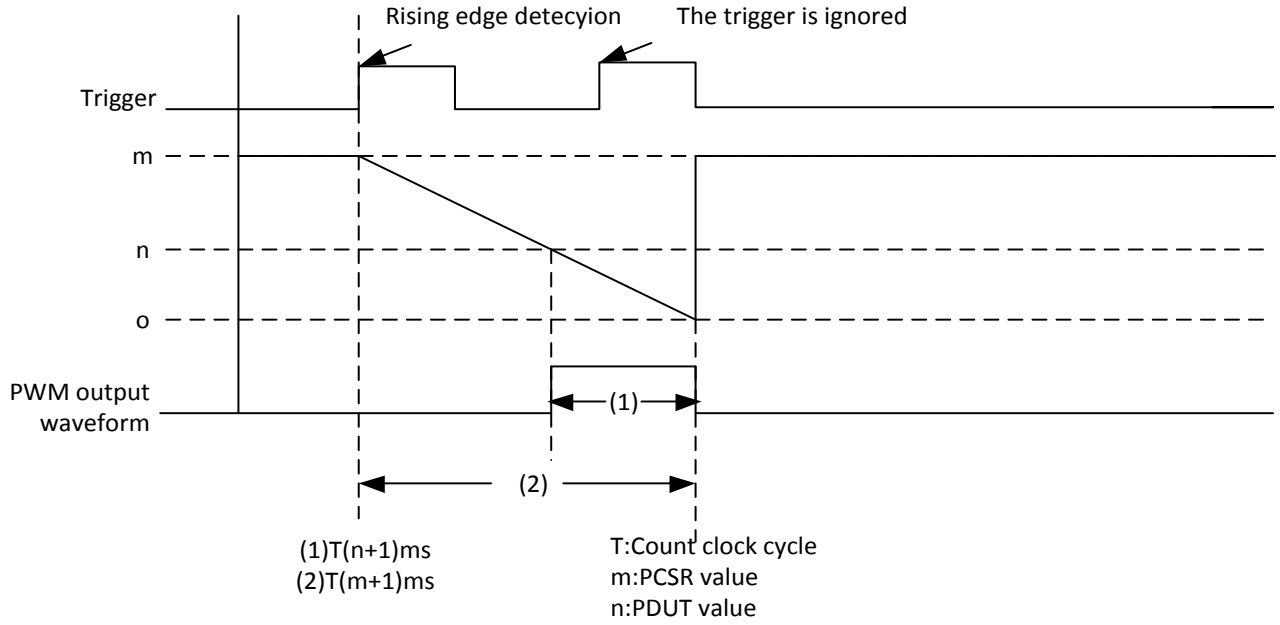
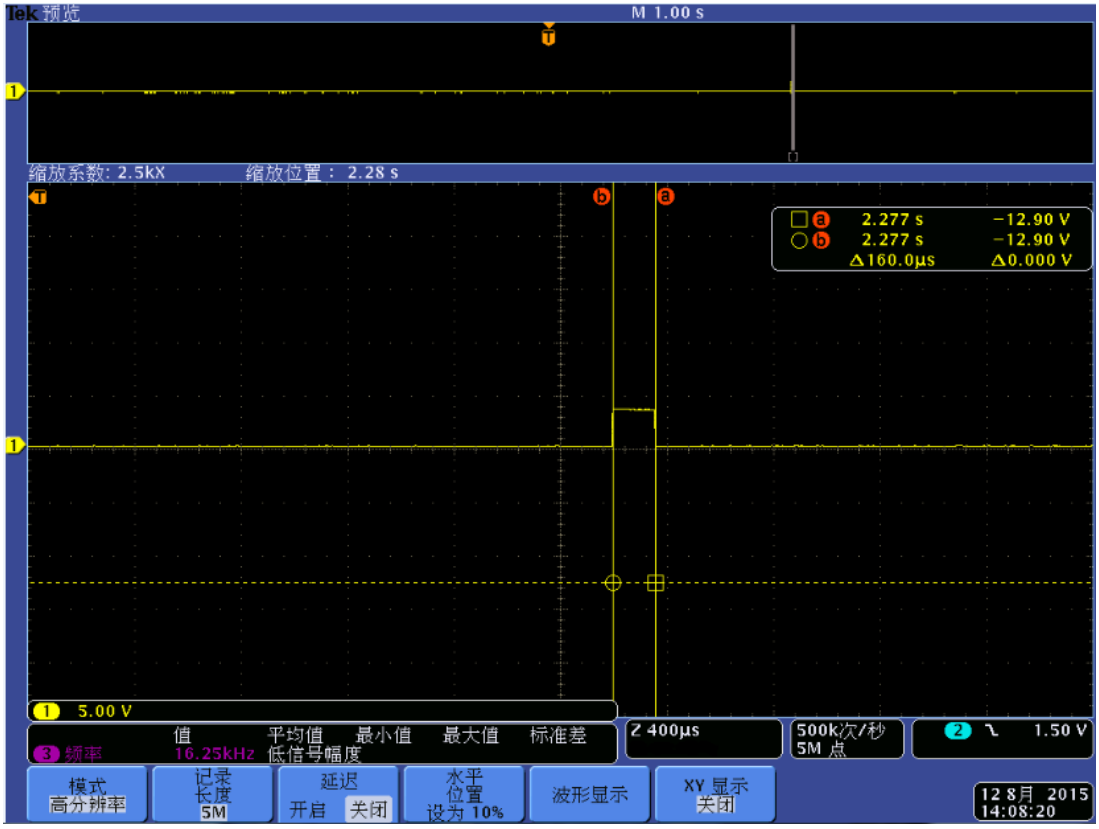


Figure 12 illustrates the waveform if the timer is retriggered and restart is disabled in one-shot operation.

Figure 12. P3B in One-Shot Operation, Trigger Restart Is Disabled (Duty Cycle = 20%)



Summary

This application note covers the principle of PWM generation, the definition of related registers with base timer, the flow of configuring PWM timer mode, the code example of configuring different work mode (one-shot mode, continuous mode, restart mode, restart disable mode, etc).

Document History

Document Title: AN99235 – FM4 S6E2HG Series MCU – 16-Bit PWM Using a Base Timer

Document Number: 001-99235

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5005098	CBZH	11/06/2015	New application note

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

PSoC is a registered trademark 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, 2015. 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.