AN205009

F2MC-8FX Family MB95F430 Series 16-bit PPG Timer Application Note

Associated Part Family: MB95F430 Series

In this application Note, we will introduce how to use the PPG function on the MB95F430 series.

Contents

1 Introduction

In this document, we will introduce how to use the PPG function on the MB95F430 series.

Section 2 introduces background and PPG formats.

Section 3 introduces PPG block, PPG registers, and the setting procedure.

Section 4 introduces PPG interrupt.

Section 5 introduces PPG drivers.

Section 6 introduces PPG application demo.

2 PPG Overview

This chapter gives an overview on PPG.

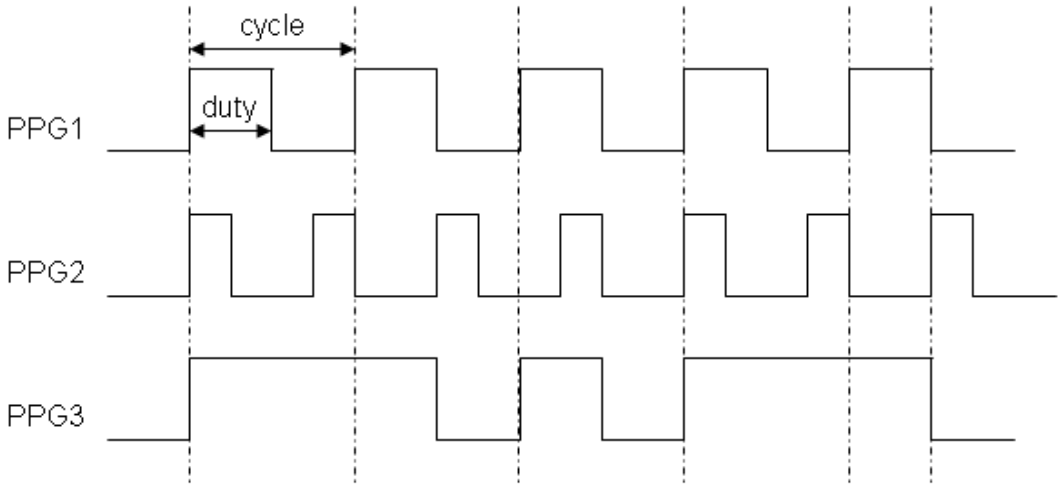
2.1 Background

It normally is used in communication, electronic, automation and so on. In communication, the duty or frequency can be used to identify the protocol. In electronic, it can use to design switching power supply. In automation, it can use to control motors.

2.2 PPG Formats

PPG format consists of frequency and duty. PPG's wave is below.

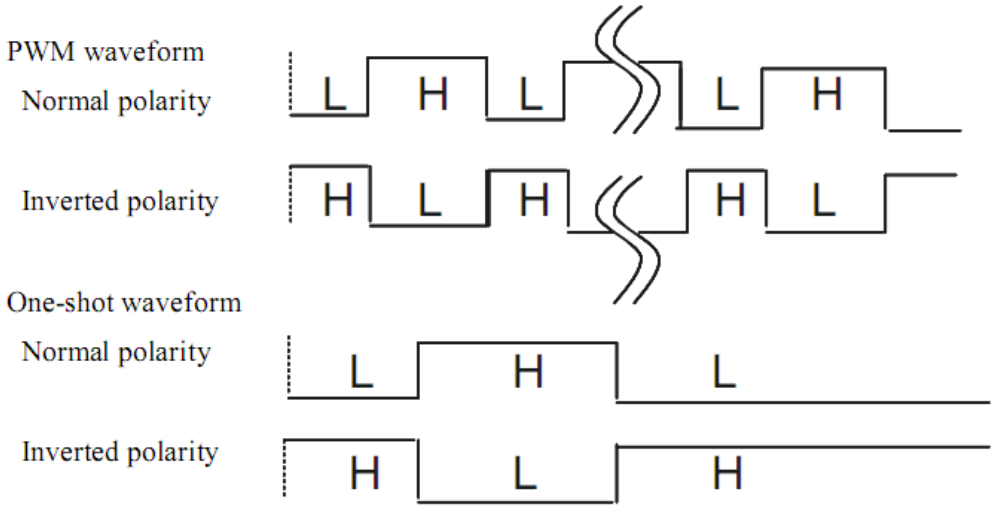
Figure 2-1: PPG Formats



3 16-bit PPG Timer Description

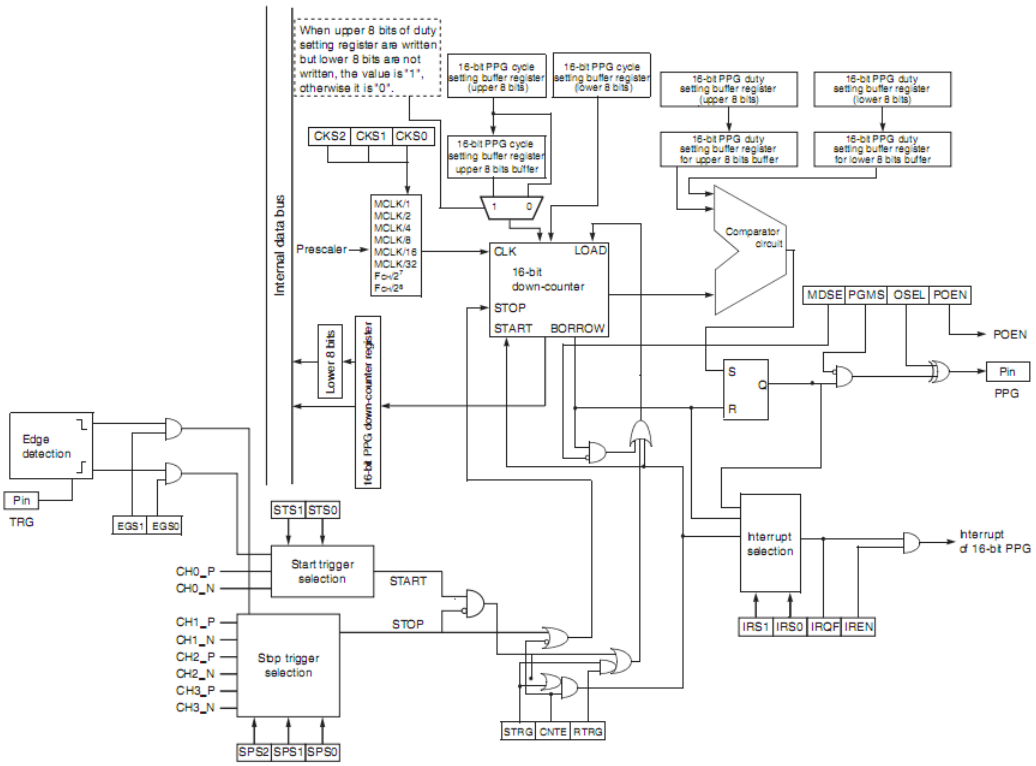
16-bit PPG timer can output the PWM output and the one shot. The output waveform can be reversed by setting the register (Normal polarity ↔ Inverted polarity).

Figure 3-1: 16-bit PPG Timer



3.1 Block Diagram of 16-bit PPG Timer

Figure 3-2: Block Diagram of 16-bit PPG Timer



3.2 Pins of 16-bit PPG Timer

The pin related to the 16-bit PPG timer is namely the PPG0 pin, TRG0 pin.

PPG0 is output pin. The PPG waveform can be outputted by using the 16-bit PPG status control register to enable output (PCNTL0: POEN=1). TRG0 is used to start 16-bit PPG timer by hardware trigger.

3.3 Registers List of 16-bit PPG Timer

Register	Description
PDCRH0	16-bit PPG down counter register (upper)
PDCRL0	16-bit PPG down counter register (lower)
PCSRH0	16-bit PPG cycle setting buffer register (upper)
PCSRL0	16-bit PPG cycle setting buffer register (lower)
PDUTH0	16-bit PPG duty setting buffer register (upper)
PDUTL0	16-bit PPG duty setting buffer register (lower)
PCNTH0	16-bit PPG status control register (upper)
PCNTL0	16-bit PPG status control register (lower)
PTGS0	16-bit PPG trigger source control register

3.4 16-bit PPG Down Counter Registers

The 16-bit PPG down counter registers (upper, lower) (PDCRH0, PDCRL0) form a 16-bit register which is used to read the count value from the 16-bit PPG down-counter.

The initial values of the register are all "0".

Always use one of the following procedures to read from this register.

Use the "MOVW" instruction (use a 16-bit access instruction to read the PDCRH0 register address)

Use the "MOV" instruction and read PDCRH0 first and PDCRL0 second (reading PDCRH0 automatically copies the lower 8 bits of the down-counter to PDCRL0)

These registers are read-only and writing has no effect on the operation.

Figure 3-3: 16-bit PPG Down Counter Register

16-bit PPG down counter register (upper) PDCRH0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FAA_H$ PDCRH0	DC15	DC14	DC13	DC12	DC11	DC10	DC09	DC08	00000000_B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

16-bit PPG down counter register (lower) PDCRL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FAB_H$ PDCRL0	DC07	DC06	DC05	DC04	DC03	DC02	DC01	DC00	00000000_B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

3.5 16-bit PPG Cycle Setting Buffer Registers

The 16-bit PPG cycle setting buffer registers are used to set the cycle for the output pulses generated by the PPG.

These registers form a 16-bit register which sets the period for the output pulses generated by the PPG. The values set in these registers are loaded to the down-counter.

When writing to these registers, always use one of the following procedures.

- Use the "MOVW" instruction (use a 16-bit access instruction to write to the PCSRH0 register address)
- Use the "MOV" instruction and write to PCSRH0 first and PCSRL0 second

If a down-counter load occurs after writing data to PCSRH0 (but before writing data to PCSRL0), the previous valid PCSRH0/PCSRL0 value will be loaded to the down-counter. If the PCSRH0/PCSRL0 value is modified during counting, the modified value will become effective from the next load of the down-counter.

- Do not set PCSRH0 and PCSRL0 to "00H", or PCSRH0 to "01H" and PCSRL0 to "01H".

Figure 3-4: 16-bit PPG Cycle Register

16-bit PPG cycle setting buffer register (upper) PCSRH0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
$0FAC_H$ PCSRH0	CS15	CS14	CS13	CS12	CS11	CS10	CS09	CS08	11111111_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

16-bit PPG cycle setting buffer register (lower) PCSRL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FAD_H$ PCSRL0	CS07	CS06	CS05	CS04	CS03	CS02	CS01	CS00	11111111_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

3.6 16-bit PPG Duty Setting Buffer Registers

The 16-bit PPG duty setting buffer registers control the duty ratio for the output pulses generated by the PPG.

These registers form a 16-bit register which controls the duty ratio for the output pulses generated by the PPG. Transfer of the data from the 16-bit PPG duty setting buffer registers to the duty setting registers is performed at the same timing as the down-counter read.

When writing to these registers, always use one of the following procedures.

- Use the "MOVW" instruction (use a 16-bit access instruction to write to the PDUTH0 register address)
- Use the "MOV" instruction and write to PDUTH0 first and PDUTL0 second

If a down-counter load occurs after writing data to PDUTH0 (but before writing data to PDUTL0), the value of the 16-bit PPG duty setting buffer registers is not transferred to the duty setting registers.

The relation between the value of the 16-bit PPG duty setting registers and output pulse is as follows:

- When the same value is set in both the 16-bit PPG cycle setting buffer registers and duty setting registers, the "H" level will always be outputted if normal polarity is set, or the "L" level will always be outputted if inverted polarity is set.
- When the duty setting registers are set to "00B", the "L" level will always be outputted if normal polarity is set, or the "H" level will always be outputted if inverted polarity is set.
- When the value set in the duty setting registers is greater than the value in the 16-bit PPG cycle setting buffer registers, the "L" level will always be outputted if normal polarity is set, and the "H" level will always be outputted if inverted polarity is set.

Figure 3-5: 16-bit PPG Duty Register

16-bit PPG duty setting buffer register (upper) PDUTH0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
$0FAE_H$ PDUTH0	DU15	DU14	DU13	DU12	DU11	DU10	DU09	DU08	11111111_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

16-bit PPG duty setting buffer register (lower) PDUTL0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0FAF_H$ PDUTL0	DU07	DU06	DU05	DU04	DU03	DU02	DU01	DU01	11111111_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

3.7 16-bit PPG Status Control Registers

The 16-bit PPG status control register is used to enable and disable the 16-bit PPG timer and also to set the operating status for the software trigger, retrigger control interrupt, and output polarity. This register can also check the operation status.

Figure 3-6: 16-bit PPG Status Control Register, Upper Byte (PCNTH0)

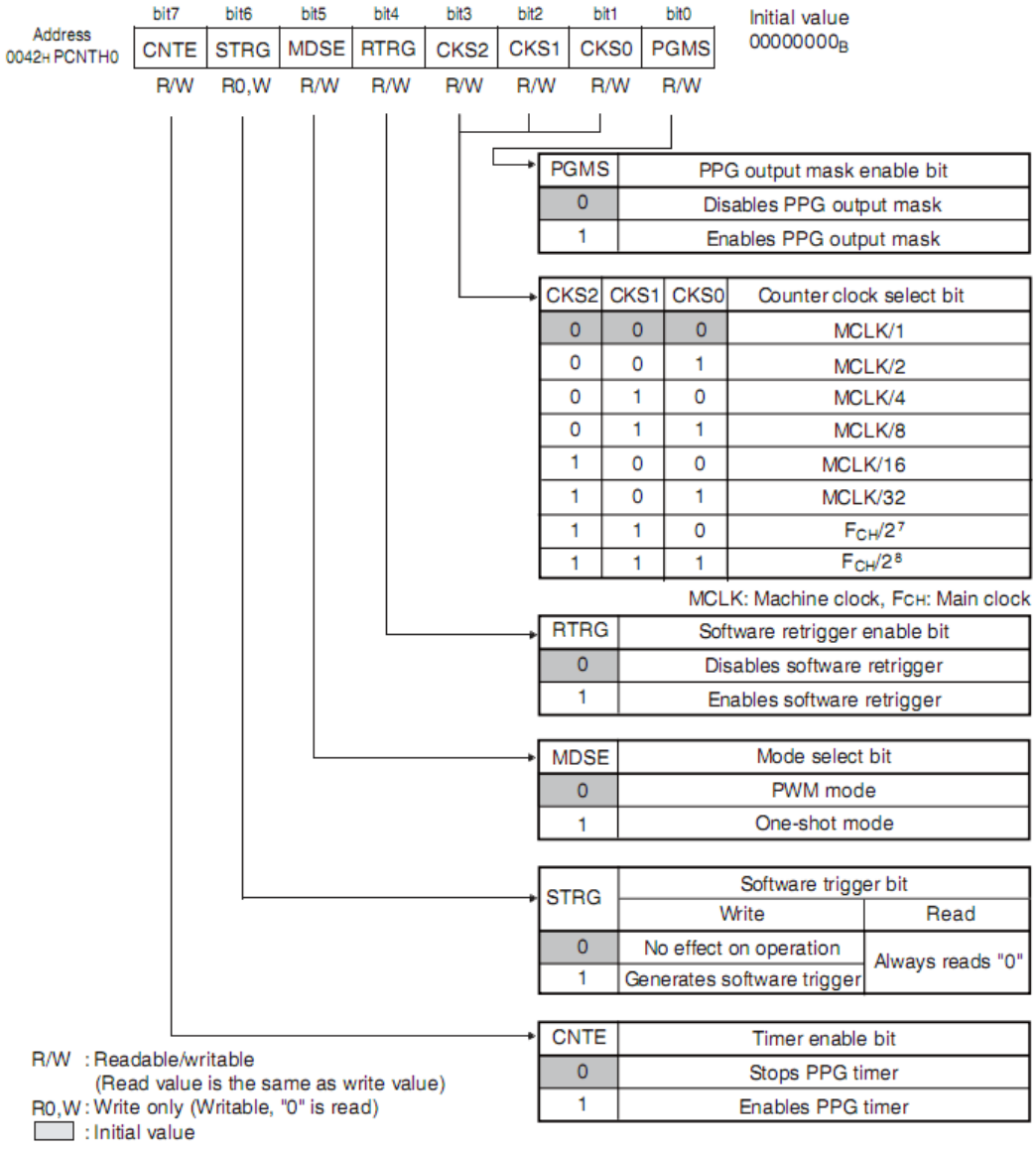
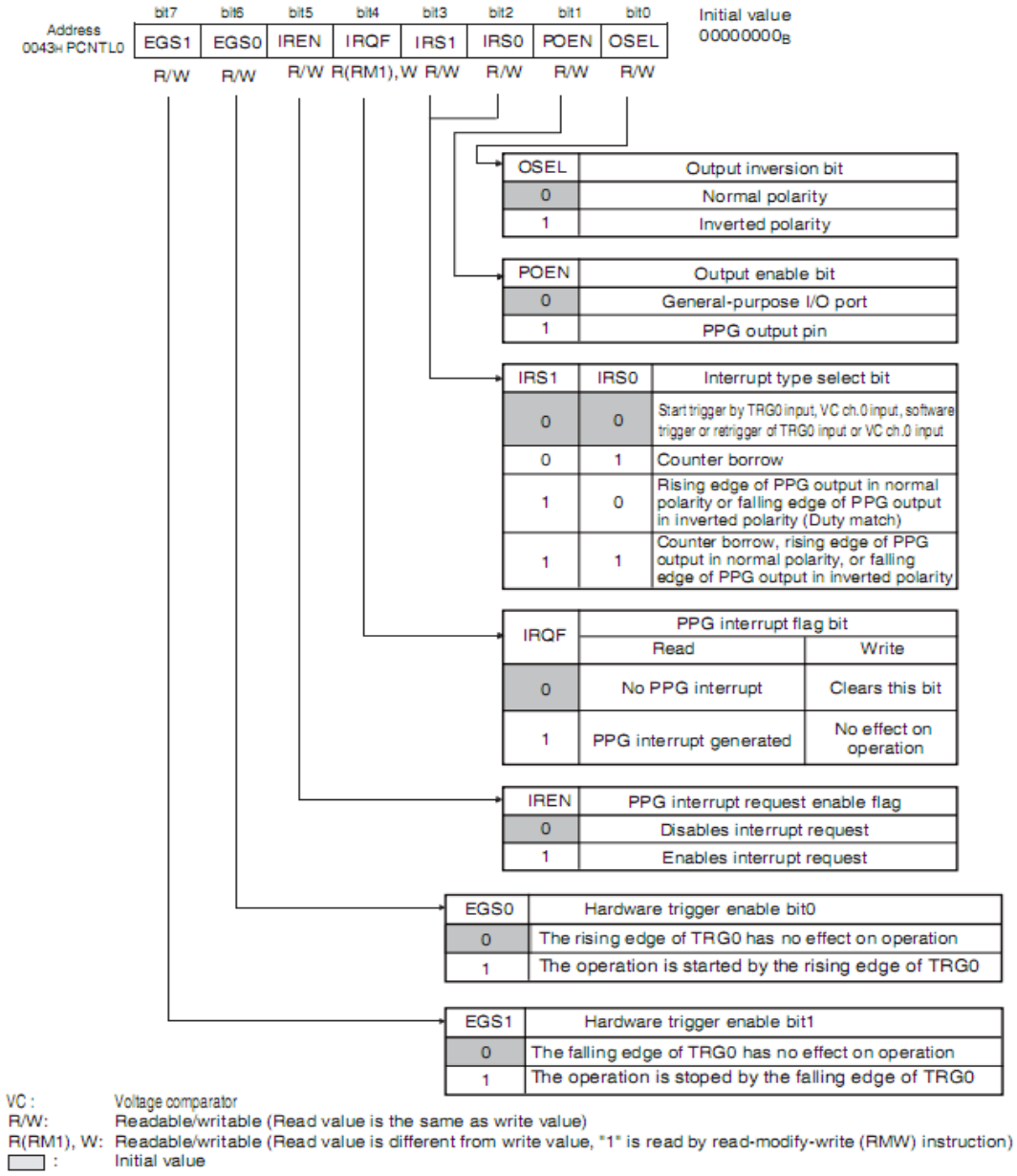


Figure 3-7: 16-bit PPG Status Control Register, Lower Byte (PCNTL0)



Address 0043H PCNTL0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value 00000000B
	EGS1	EGS0	IREN	IRQF	IRS1	IRS0	POEN	OSEL	
	R/W	R/W	R/W	R(RM1), W	R/W	R/W	R/W	R/W	

OSEL	Output inversion bit
0	Normal polarity
1	Inverted polarity

POEN	Output enable bit
0	General-purpose I/O port
1	PPG output pin

IRS1	IRS0	Interrupt type select bit
0	0	Start trigger by TRG0 input, VC ch.0 input, software trigger or retrigger of TRG0 input or VC ch.0 input
0	1	Counter borrow
1	0	Rising edge of PPG output in normal polarity or falling edge of PPG output in inverted polarity (Duty match)
1	1	Counter borrow, rising edge of PPG output in normal polarity, or falling edge of PPG output in inverted polarity

IRQF	PPG interrupt flag bit	
	Read	Write
0	No PPG interrupt	Clears this bit
1	PPG interrupt generated	No effect on operation

IREN	PPG interrupt request enable flag
0	Disables interrupt request
1	Enables interrupt request

EGS0	Hardware trigger enable bit0
0	The rising edge of TRG0 has no effect on operation
1	The operation is started by the rising edge of TRG0

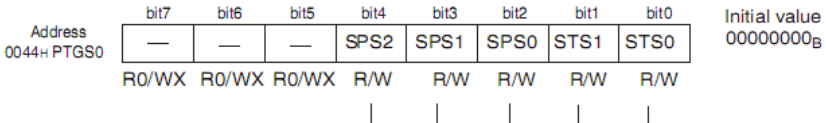
EGS1	Hardware trigger enable bit1
0	The falling edge of TRG0 has no effect on operation
1	The operation is stoped by the falling edge of TRG0

VC : Voltage comparator
R/W: Readable/writable (Read value is the same as write value)
R(RM1), W: Readable/writable (Read value is different from write value, "1" is read by read-modify-write (RMW) instruction)
□ : Initial value

3.8 16-bit PPG trigger source control Register

The 16-bit PPG trigger source control register controls the trigger source of the 16-bit PPG timer.

Figure 3-8: 16-bit PPG Trigger Source Control Register (PTGS0)



STS1	STS0	hardware start trigger select bits
0	0	Enable select rising edge of TRG0 as PPG start trigger
0	1	Select rising edge of VC ch.0 as PPG start trigger
1	0	Select falling edge of VC ch.0 as PPG start trigger
1	1	Select both edges of VC ch.0 as PPG start trigger

SPS2	SPS1	SPS0	hardware stop trigger select bits
0	0	0	Enable select falling edge of TRG0 as PPG stop trigger
0	0	1	Select rising edge of VC ch.1 as PPG stop trigger
0	1	0	Select falling edge of VC ch.1 as PPG stop trigger
0	1	1	Select both edges of VC ch.1 as PPG stop trigger
1	0	0	Select rising edge of VC ch.1/ch.2/ch.3 as PPG stop trigger
1	0	1	Select falling edge of VC ch.1/ch.2/ch.3 as PPG stop trigger
1	1	0	Setting prohibited
1	1	1	Setting prohibited

3.9 PPG Macro General Setting Procedure

Initial setup

1. Set the interrupt level. (ILR3, ILR4)
2. Enable the hardware trigger and interrupts, select the interrupt type, and enable output. (PCNTL0)
3. Select the count clock and the mode, and enable timer operation. (PCNTH0)
4. Set the cycle. (PCSRH0, PCSRL0)
5. Set the duty. (PDUTH0, PDUTL0)
6. Start the PPG by the software trigger. (PCNTH0: STRG = 1)

Interrupt processing

1. Process any interrupt.
2. Clear the interrupt request flag. (PCNTL0: IRQF)

4 Interrupts of 16-bit PPG Timer

PPG interrupt will be described.

The 16-bit PPG timer can generate interrupt requests in the following cases:

When a trigger or counter borrow occurs

When a rising edge of PPG is generated in normal polarity

When a falling edge of PPG is generated in inverted polarity

The interrupt operation is controlled by IRS1 (bit3) and IRS0 (bit2) in the PCNTL register.

4.1 Interrupt Control Bits and Interrupt Sources of 16-bit PPG Timer

Item	Description
Interrupt flag bit	PCNTL0:IRQF
Interrupt request enable bit	PCNTL0:IREN
Interrupt type select bits	PCNTL0:IRS1, IRS0
Interrupt sources	PCNTL0:IRS1, IRS0=00_B Hardware trigger by TRG Pin input of 16-bit down-counter, software trigger and retrigger
	PCNTL0:IRS1, IRS0=01_B Counter borrow of 16-bit down-counter
	PCNTL0:IRS1, IRS0=10_B Rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity
	PCNTL0:IRS1, IRS0=11_B Counter borrow of 16-bit down-counter, rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity

4.2 Register and Vector Table Addresses Related to Interrupts of 16-bit PPG Timer

Interrupt source	Interrupt request no.	Interrupt level setting register		Vector table address	
		Register	Setting bit	Upper	Lower
16-bit PPG timer ch. 1	IRQ17	ILR4	L17	$FFD8_H$	$FFD9_H$

5 PPG Trigger

PPG trigger is described here.

PPG activation by signal input to the TRG input pin or internal voltage comparator (VC) input.

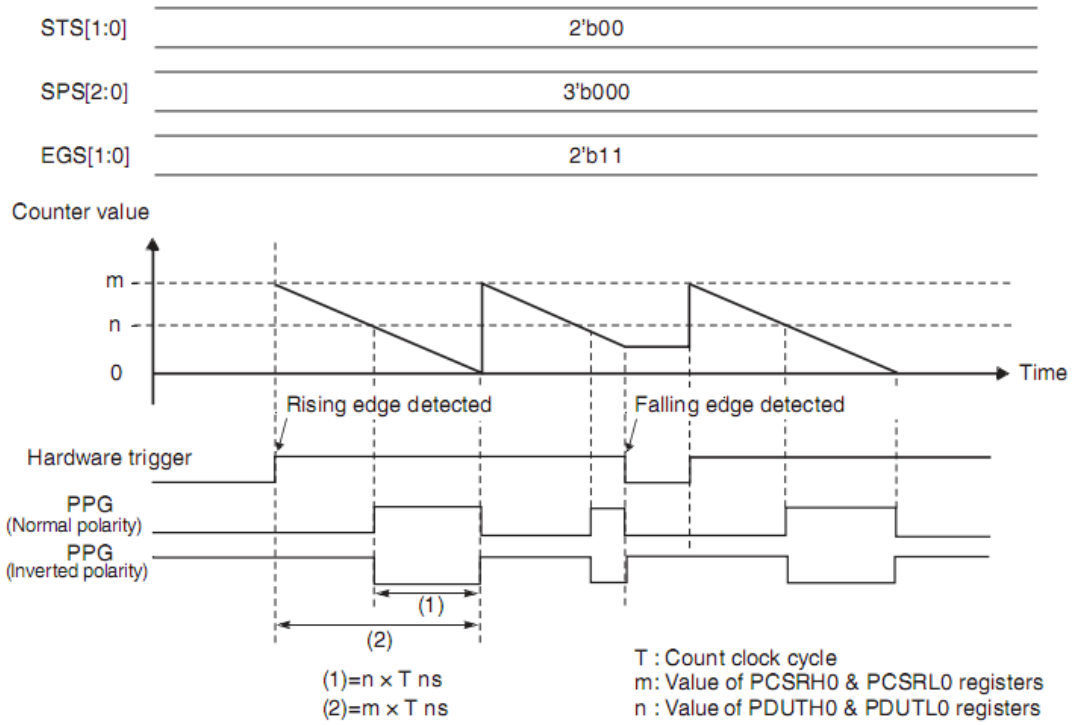
Condition 1: PPG start triggering and stop triggering do not occur simultaneously.

1. TRG

When STS1 and STS0 are set to "00B", SPS2, SPS1 and SPS0 to "000B", and EGS1 and EGS0 to "11B" and the hardware trigger input from TRG is used, the 16-bit PPG timer starts operating at a rising edge and stops upon the detection of a falling edge. Moreover, the 16-bit PPG timer also starts operating at the following rising edge from the beginning.

The operation can be re-triggered by a valid TRG input hardware trigger regardless of the retrigger setting of the RTRG bit when the TRG input hardware trigger has been selected.

Figure 5-1: Trigger from TRG in PWM Mode



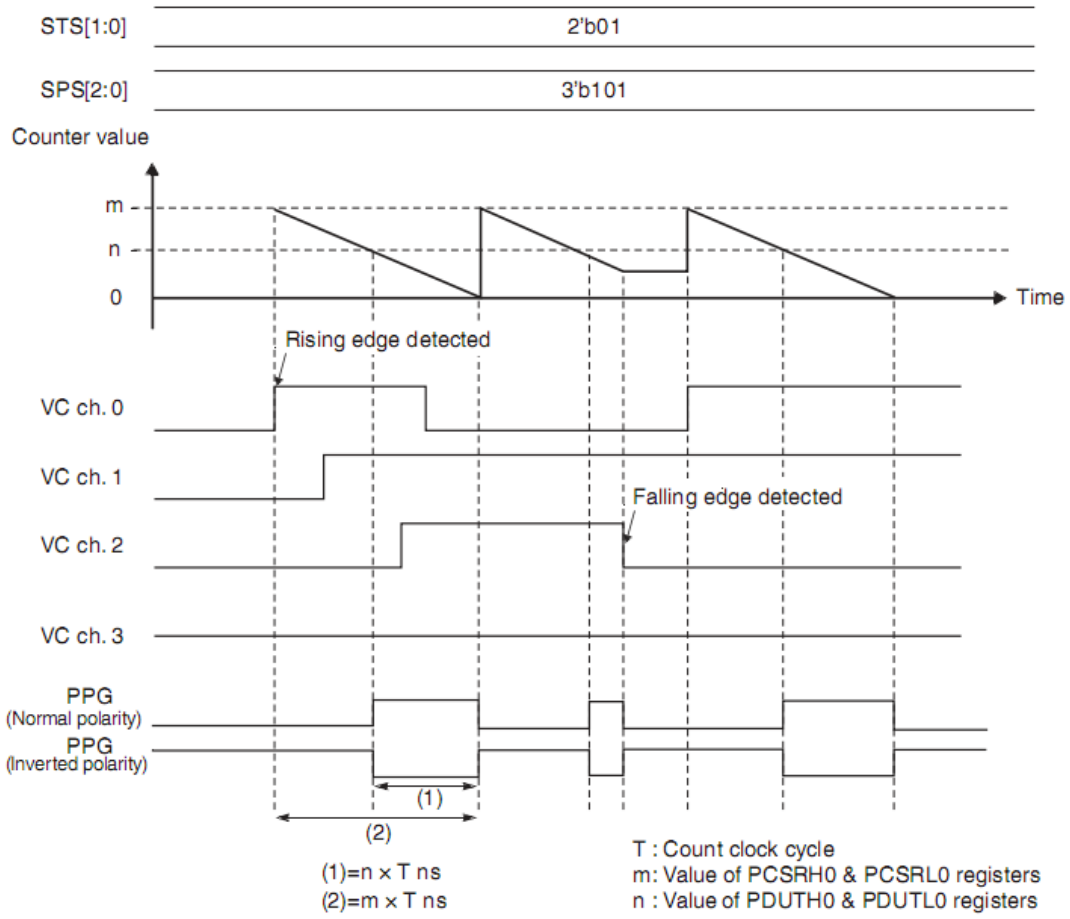
2. Voltage comparator (VC)

When STS1 and STS0 are set to "01B", SPS2, SPS1 and SPS0 to "101B", and the hardware trigger input from the VC ch. 0 is used, the 16-bit PPG timer starts operating at a rising edge of VC ch. 0 and stops upon the detection of a falling edge of VC ch.1/2/3.

Moreover, the 16-bit PPG timer also starts operating at the following rising edge of VC ch. 0 from the beginning.

The operation can be re-triggered by a valid rising edge of VC ch. 0 input hardware trigger regardless of the retrigger setting of the RTRG bit when the VC ch. 0 input hardware trigger has been selected.

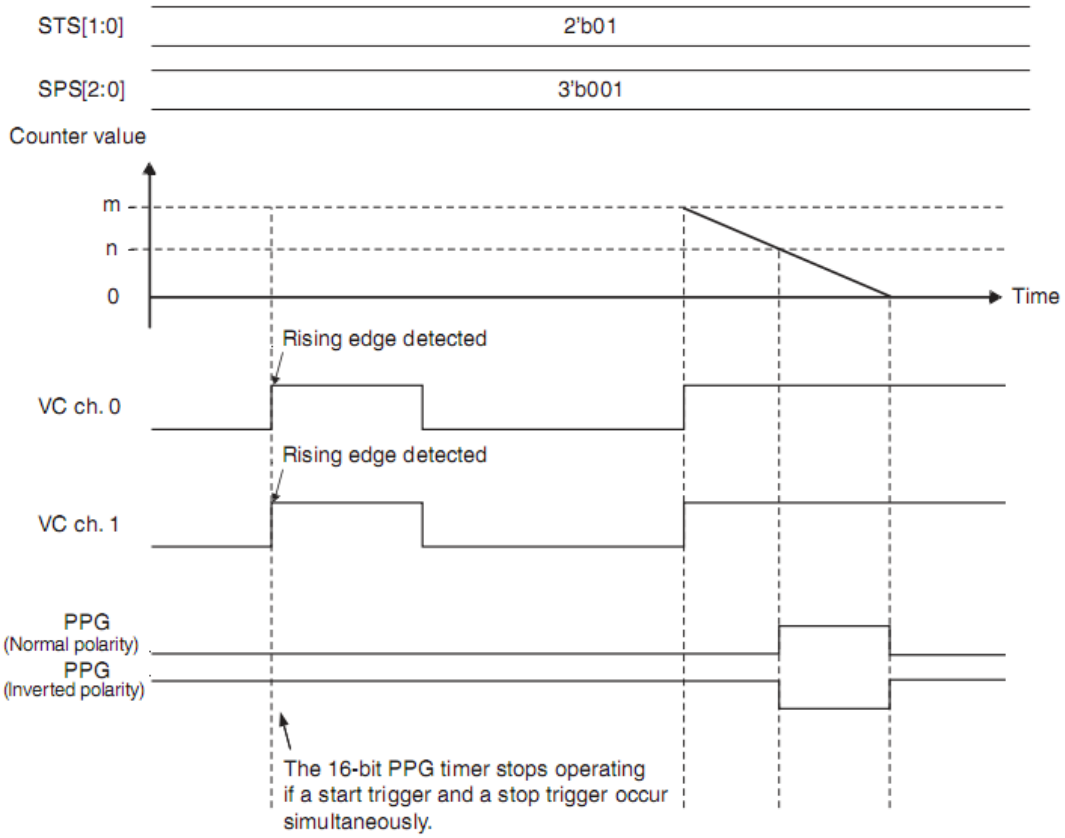
Figure 5-2: Trigger from TRG in PWM Mode



Condition 2: PPG start triggering and stop triggering occur simultaneously.

When STS1 and STS0 are set to "01B", SPS2, SPS1 and SPS0 to "001B", and if the hardware trigger from a rising edge of VC ch. 0 and another rising edge of VC ch. 1 occur simultaneously, the PPG will halt (the stop trigger has higher priority than the start trigger.).

Figure 5-3: Triggers Start and Stop Occur Simultaneously in PWM Mode



6 PPG Driver

This chapter describes the PPG driver.

6.1 Peripheral Usage

The MCU pins used as below:

PPG0, used as PPG wave output;

TRG0, used as PPG trigger;

6.2 Driver Code

6.2.1 General Definition

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;          /* Unsigned  8 bit quantity */
typedef signed   char  INT8S;          /* Signed    8 bit quantity */
typedef unsigned int   INT16U;         /* Unsigned 16 bit quantity */
typedef signed   int   INT16S;         /* Signed   16 bit quantity */
typedef unsigned long  INT32U;         /* Unsigned 32 bit quantity */
typedef signed   long  INT32S;         /* Signed   32 bit quantity */

#define BOOL                    BOOLEAN
#define BYTE                    INT8U
#define UBYTE                   INT8U
#define WORD                    INT16U
#define UWORD                   INT16U
#define LONG                    INT32S
#define ULONG       INT32U
#define UCHAR       INT8U
#define UINT                            INT16U
#define DWORD                   INT32U

#define TRUE         1
#define FALSE        0

#define BYTE_LO(w)     ((UBYTE)(w))
#define BYTE_HI(w)     ((UBYTE)(((UWORD)(w)>>8)&0xFF))
```

6.2.2 PPG Routine

void PPGOpen()

Return : none.

Parameters : none.

Description : open the PPG function.

Example : PPGOpen();

```
void PPGOpen()
{
      DDR7_P73=1;
      PCNTH0_PGMS=0;//Open PPG
}
```

void PPGClose ()

Return : none.

Parameters : none.

Description : close the PPG function.

Example : PPGClose();

```
void PPGClose()
{
      DDR7_P73=1;
      PCNTH0_PGMS=1;//Close PPG
}
```

void PPGSet(WORD usCycle,WORD usDuty)

Return : none.

Parameters : usCycle, is PPG cycle(us)

usDuty, is PPG duty(us)

Description : set the PPG function.

Example : PPGSet(40,10);

```
void PPGSet(WORD usCycle,WORD usDuty)
{
      DDR7_P73=1;
//PPG0
      PCSRH0=BYTE_HI(usCycle);//Upper Byte
      PCSRL0=BYTE_LO(usCycle);//Lower Byte
      PDUTH0=BYTE_HI(usDuty);//Upper Byte
      PDUTL0=BYTE_LO(usDuty);//Lower Byte
      PCNTH0=0xF7;//Mask=1,1.000us(MCLK/8),OneShot,Soft Trigger
      PCNTL0=0x0E;//OSEL=L,IREN=0
}
```

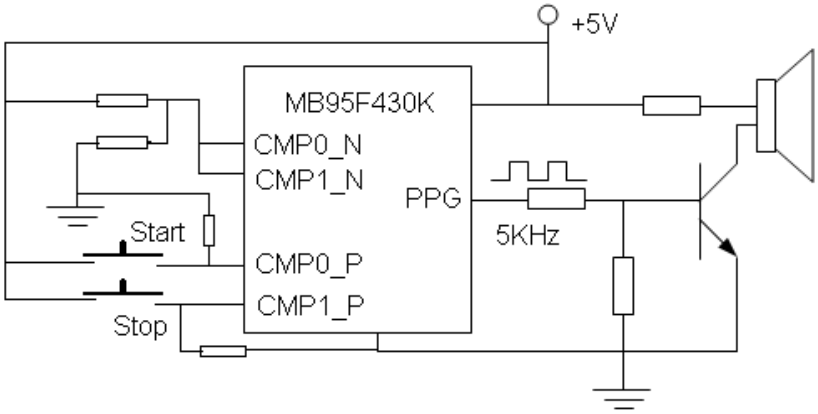
7 Typical Application

The PPG typical application will be introduce here.

7.1 HW Design

In this application, we will use the PPG to drive a buzzer. VC0 rising edge start PPG, VC1 rising edge stop PPG (PTGS0=0x05). The MCU used is MB95F430K. The HW is designed as below.

Figure 7-1: Hardware Design



7.2 Sample Code

void main(void)

Return : none.

Parameters : none;

Description : system main programm.

Example : main();

```
void main(void)
{
      __DI();
      __set_il(3);
      InitIrqLevels();

      WDTH =0xA5;//Disable WTG
      WDTL =0x96;

      WATR =0xEE;
      SYCC =0xF0;//Main Clock
      SYCC2=0xF4;//Main Clock
      SYSC  =0xBC;//BUZZ(P01)
      SYSC2 =0x02;//PPG(P73),Disable I2C
      while(!STBC_MRDY);
      __EI();

      PTGS0=0x05;//VC0 rising edge start PPG,VC1 rising edge stop PPG
      PPGSet(200,10);
      PPGOpen();
      PPGClose();
}
```

8 Additional Information

For more information on Cypress products, please visit the following website:

http://www.cypress.com/cypress-microcontrollers

A Appendix

A.1 Sample Codes

main.c

```
/*-----------------------------------------------------------------------------*/
/*    Title : Sample main program for sensorless DC inverter control           */
/*    Aurthor     : Folix Li
                        */
/*    Date  : 26th Nov 2999
                    */
/*-----------------------------------------------------------------------------*/
#include "mb95430.h"
#include "TypeDef.h"

/*-----------------------------------------------------------------------------*/
/* PWM Control
/*-----------------------------------------------------------------------------*/
void PPGOpen()
{
      DDR7_P73=1;
      PCNTH0_PGMS=0;
}

void PPGClose()
{
      DDR7_P73=1;
      PCNTH0_PGMS=1;
}

void PPGSet(WORD usCycle,WORD usDuty)
{
      DDR7_P73=1;
//PPG0
      PCSRH0=BYTE_HI(usCycle);//Upper Byte
      PCSRL0=BYTE_LO(usCycle);//Lower Byte
      PDUTH0=BYTE_HI(usDuty);//Upper Byte
      PDUTL0=BYTE_LO(usDuty);//Lower Byte
      PCNTH0=0xF7;//Mask=1,1.000us(MCLK/8),OneShot,Soft Trigger
      PCNTL0=0x0E;//OSEL=L,IREN=0
}


void main(void)
{
      __DI();
      __set_il(3);
      InitIrqLevels();

      WDTH =0xA5;
      WDTL =0x96;

      WATR =0xEE;
      SYCC =0xF0;//Main Clock
      SYCC2=0xF4;//Main Clock
```

```
      SYSC  =0xBC;//BUZZ(P01)
      SYSC2 =0x02;//PPG(P73),Disable I2C
      while(!STBC_MRDY);
      __EI();

      PTGS0=0x05;//VC0 rising edge start PPG,VC1 rising edge stop PPG
      PPGSet(200,10);
      PPGOpen();
      PPGClose();
}


VECTORS.C

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* SEMICONDUCTOR ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES.                                          */
/*                 (C) Fujitsu Semiconductor Europe GmbH                  */
/*------------------------------------------------------------------------
  VECTORS.C
  - Interrupt level (priority) setting
  - Interrupt vector definition
  29.09.04  1.00   HWe    V30L29
------------------------------------------------------------------------*/

 #include "mb95430.h"

/*------------------------------------------------------------------------
   InitIrqLevels()

   This function  pre-sets all interrupt control registers. It can be used
   to set all interrupt priorities in static applications. If this file
   contains assignments to dedicated resources, verify  that the
   appropriate controller is used.

   NOTE: value 0xFF disables the interrupt and value 0 sets highest priority.
   NOTE: For all resource interrupts exists 3 Interrupt level registers (ILRx).
         Each register sets the level for 4 different resources (IRQx).
   NOTE: This list is prepared for the 8FX-emulation chip MB95FV100 'Horn'.
         Not all resources will be supported by all 8FX-devices
------------------------------------------------------------------------*/

void InitIrqLevels(void)
{
/*  ILRx                  IRQs defined by ILRx */

    ILR0 = 0xFF;      //  IRQ0:  external interrupt ch0 | ch4
                      //  IRQ1:  external interrupt ch1 | ch5
                      //  IRQ2:  external interrupt ch2 | ch6
                      //  IRQ3:  external interrupt ch3 | ch7

    ILR1 = 0xFF;      //  IRQ4:  UART/SIO ch0
                      //  IRQ5:  8/16-bit timer ch0 (lower)
                      //  IRQ6:  8/16-bit timer ch0 (upper)
                      //  IRQ7:  Output Compare ch0
```

```
    ILR2 = 0xFF;        //  IRQ8:  Output Compare ch1
                        //  IRQ9:  none
                        //  IRQ10: Voltage Compare ch0
                        //  IRQ11: Voltage Compare ch1

    ILR3 = 0xFF;        //  IRQ12: Voltage Compare ch2
                        //  IRQ13: Voltage Compare ch3
                        //  IRQ14: 16-bit free run timer
                        //  IRQ15: 16-bit PPG0

    ILR4 = 0xFF;        //  IRQ16: I2C ch0
                        //  IRQ17: none
                        //  IRQ18: 10-bit A/D-converter
                        //  IRQ19: Timebase timer

    ILR5 = 0xFF;        //  IRQ20: Watch timer
                        //  IRQ21: none
                        //  IRQ22: none
                        //  IRQ23: Flash Memory
}

/*-----------------------------------------------------------------------
   Prototypes

   Add your own prototypes here. Each vector definition needs is proto-
   type. Either do it here or include a header file containing them.
-----------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void);

/*-----------------------------------------------------------------------
   Vector definiton

   Use following statements to define vectors.
   All resource related vectors are predefined.
   Remaining software interrupts can be added hereas well.
-----------------------------------------------------------------------*/
#pragma intvect DefaultIRQHandler 0   //  IRQ0:  external interrupt ch0 | ch4
#pragma intvect DefaultIRQHandler 1   //  IRQ1:  external interrupt ch1 | ch5
#pragma intvect DefaultIRQHandler 2   //  IRQ2:  external interrupt ch2 | ch6
#pragma intvect DefaultIRQHandler 3   //  IRQ3:  external interrupt ch3 | ch7

#pragma intvect DefaultIRQHandler 4   //  IRQ4:  UART/SIO ch0
#pragma intvect DefaultIRQHandler 5   //  IRQ5:  8/16-bit timer ch0 (lower)
#pragma intvect DefaultIRQHandler 6   //  IRQ6:  8/16-bit timer ch0 (upper)
#pragma intvect DefaultIRQHandler 7   //  IRQ7:  Output Compare ch0

#pragma intvect DefaultIRQHandler 8   //  IRQ8:  Output Compare ch1
#pragma intvect DefaultIRQHandler 9   //  IRQ9:  none
#pragma intvect DefaultIRQHandler 10  //  IRQ10: Voltage Compare ch0
#pragma intvect DefaultIRQHandler 11  //  IRQ11: Voltage Compare ch1

#pragma intvect DefaultIRQHandler 12  //  IRQ12: Voltage Compare ch2
#pragma intvect DefaultIRQHandler 13  //  IRQ13: Voltage Compare ch3
#pragma intvect DefaultIRQHandler 14  //  IRQ14: 16-bit free run timer
#pragma intvect DefaultIRQHandler 15  //  IRQ15: 16-bit PPG0
```

```
#pragma intvect DefaultIRQHandler 16  //  IRQ16: I2C ch0
#pragma intvect DefaultIRQHandler 17  //  IRQ17: none
#pragma intvect DefaultIRQHandler 18  //  IRQ18: 10-bit A/D-converter
#pragma intvect DefaultIRQHandler 19  //  IRQ19: Timebase timer

#pragma intvect DefaultIRQHandler 20  //  IRQ20: Watch timer
#pragma intvect DefaultIRQHandler 21  //  IRQ21: none
#pragma intvect DefaultIRQHandler 22  //  IRQ22: none
#pragma intvect DefaultIRQHandler 23  //  IRQ23: Flash Memory

/*-----------------------------------------------------------------------------
   DefaultIRQHandler()

   This function is a placeholder for all vector definitions.
   Either use your own placeholder or add necessary code here
   (the real used resource interrupt handlers should be defined in the main.c).
-----------------------------------------------------------------------------*/
__interrupt void DefaultIRQHandler(void)
{
      __DI();                                   // disable interrupts
      while(1)
            __wait_nop();                 // halt system
}
```

Document History

Document Title: AN205009 - F2MC-8FX Family MB95F430 Series 16-bit PPG Timer Application Note

Document Number: 002-05009

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	HUAL	03/16/2010	Initial release
*A	5264535	HUAL	05/09/2016	Migrated Spansion Application Note “MCU-AN-500078-E-10” to Cypress format.
*B	5851036	MALI	08/11/2017	Updated logo and copyright

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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