The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205007

F²MC-8FX 家族 MB95430 系列 16 位 FRT 和 OCU 应用手册

相关器件系列：MB95430 系列

本文档介绍了 16 位定时器和输出比较单元的功能及配置方法，同时给出了相关代码。

目录

1 概要 2
2 16 位 OCU 和 FRT 的特性.....2
2.1 OCU 和 FRT 的结构图.....2
2.2 OCU 和 FRT 记数器 4
2.3 OCU 的引脚设置 5
2.4 与电压比较器的关系.....5
2.5 中断.....5
3 软件操作 6
3.1 设置程序 6
3.2 操作模式 7
4 部分位的影响.....13
5 更多信息.....16
A 附录 17
A.1 范例代码 17
文档修改记录.....23

1 概要

本文档介绍了 16 位定时器和输出比较单元的功能及配置方法，同时给出了相关代码。

2 16 位 OCU 和 FRT 的特性

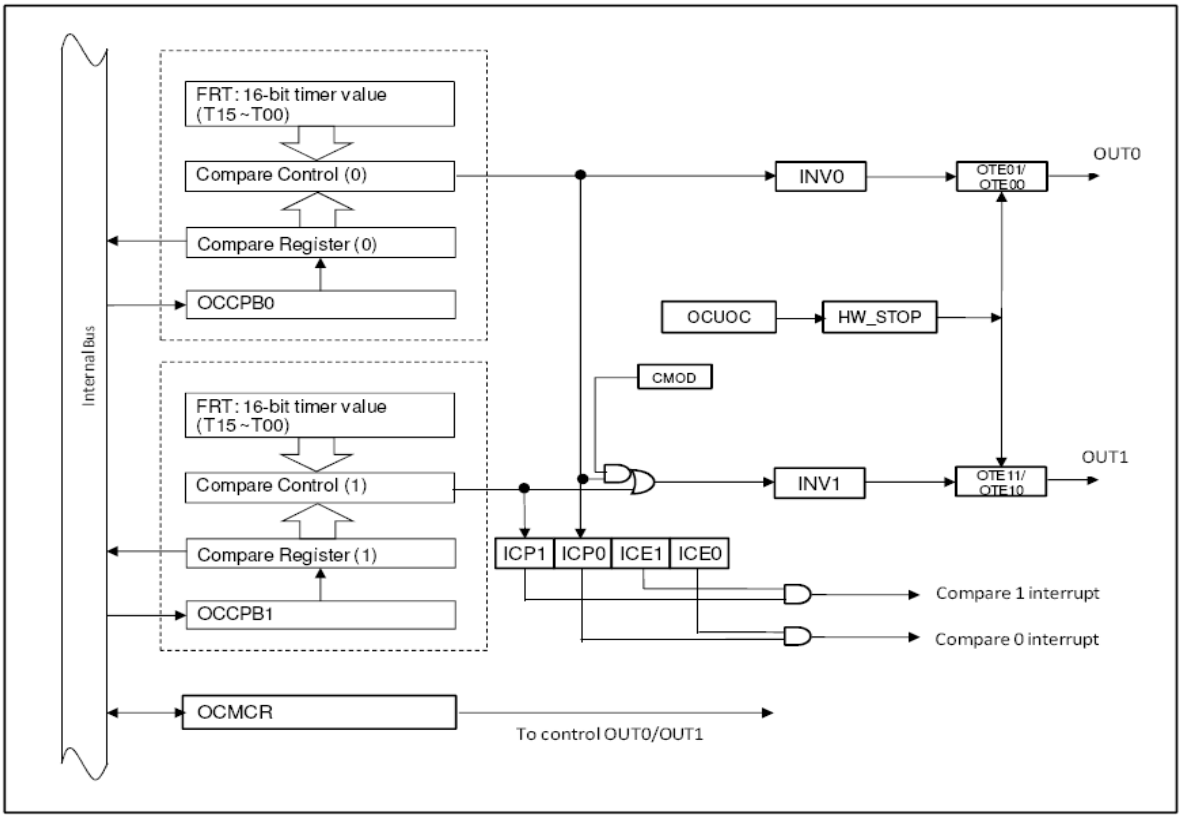
本章介绍了 OCU 和 FRT 的特性。

2.1 OCU 和 FRT 的结构图

16 位输出比较单元用于生成脉冲序列。它包括一个 16 位定时器，两个比较寄存器，两个比较缓冲寄存器，两个比较输出引脚，以及多个控制寄存器。如果烧入比较寄存器的值与 16 位定时器的计数值匹配，引脚的输出水平将被触发，产生中断。

图 2-1 显示了 16 位 OCU 的结构图。

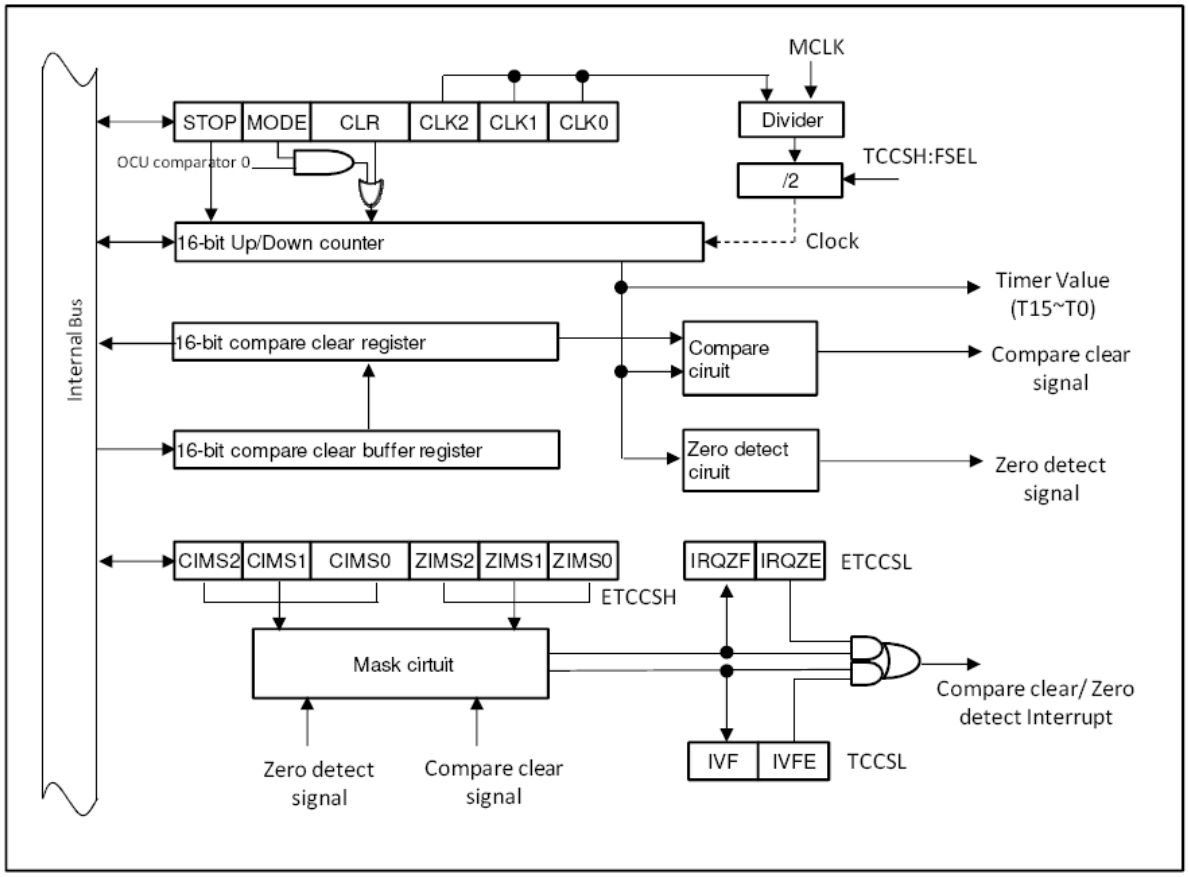
图 2-1. OCU 结构图



该 16 位定时器包括一个 16 位上/下计数器，一个比较清除缓冲寄存器，以及一个控制状态记数器。
定时器的计数值用作输出比较单元的时基。

图 2-2 显示了 16 位定时器的结构图。

图 2-2. FRT 结构图



2.2 OCU 和 FRT 记数器

2.2.1 与输出比较单元相关的寄存器

OCU 模块有两个通道：OUT0 和 OUT1。每个通道有一个 16 位输出比较寄存器和一个 16 位缓冲寄存器。 输出寄存器和缓冲寄存器位于同一地址。输出寄存器为只读；缓冲记数器只写。用户可以使用 2.2.2 节中介绍的方法访问这两种寄存器。

表 2-1 列出了与输出比较单元相关的寄存器。

表 2-1. OCU 寄存器

寄存器名	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	缺省值
OCCP0/B0 H	输出比较寄存器 0 / 缓冲寄存器 0 H								b’ 00000000
OCCP0/B0 L	输出比较寄存器 0 / 缓冲寄存器 0 L								b’ 00000000
OCCP1/B1 H	输出比较寄存器 1 / 缓冲寄存器 1 H								b’ 00000000
OCCP1/B1 L	输出比较寄存器 1 / 缓冲寄存器 1 L								b’ 00000000
OCSL	ICP1	ICP0	ICE1	ICE0	--	--	CST1	CST0	b’ 00000000
OCSH	--	CMOD0	OTE11	OTE10	OTE01	OTE00	OTD1	OTD0	b’ 01000000
EOCS	HW_STOP	--	BTS1	BUF1	--	--	BTS0	BUF0	b’ 00000000
OCMCR	--	FDEN1	INV1	CMPMD1	--	FDEN0	INV0	CMPMD0	b’ 00000000
OCUOC	--	--	--	--	--	OCSTPSEL			b’ 00000000

2.2.2 与定时器相关的寄存器

表 2-2 列出了与定时器相关的寄存器。

表 2-2. FRT 寄存器

寄存器名	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit1	Bit 0	缺省值
CPCLRH	定时器比较清除寄存器寄存器 H								b’ 11111111
CPCLRL	定时器比较清除寄存器 L								b’ 11111111
CPCLRBH	定时器比较清除缓冲寄存器 H								b’ 11111111
CPCLRBL	定时器比较清除缓冲寄存器 L								b’ 11111111
TCCSL	IVF	IVFE	中止	模式	CLR	CLK2	CLK1	CLK0	b’ 00000000
TCCSH	--	FSEL	--	--	--	--	--	--	b’ 01000000
ETCCSL	--	--	CNTDIR	--	BFE	IRQZF	IRQZE	CNTMD	b’ 00000000
ETCCSH	--	CIMS2	CIMS1	CIMS0	--	ZIMS2	ZIMS1	ZIMS0	b’ 00000000

比较清除缓冲寄存器（CPCLRBH/L）和比较清除记数器（CPCLRH/L）位于同一地址。比较清除缓冲寄存器为只写寄存器；比较清除寄存器为只读寄存器。两者都是 16 位寄存器，可使用以下方法进行读写。

- 使用 16 位访问指令读写（例如“MOVW”）。
- 首先使用字节访问指令读写高位，然后读写低位（例如“MOV”）。

2.3 OCU 的引脚设置

OCU 模块有两对输出。PG0 或 P70 可配置为 OUT0，PG1 或 P73 可配置为 OUT1，如下所示。

SYSC2_OUTSEL0 = 0： OUT0 -- P70.

SYSC2_OUTSEL0 = 1： OUT0 -- PG0.

SYSC2_OUTSEL1 = 0： OUT1 -- P73.

SYSC2_OUTSEL1 = 1： OUT1 -- PG1.

为输出比较信号，用户应通过 OCSH_ OTEn 启用通道；否则引脚状态将由 GPIO 或其他外围设备控制。
建议在禁用 OCU 通道前，初始化引脚至安全状态。

2.4 与电压比较器的关系

如果 OTE00，OTE 01（OTE10，OTE11）设置为 11，OCU 模块可由电压比较器输出停止。用户可通过配置寄存器 OCUOC，选择停止 OCU 输出的比较器和边沿方向。

2.5 中断

2.5.1 定时器中断

对于 FRT，如果首先启用中断，根据设置的不同，中断请求将在零检测或比较清除点发生。 用户可通过设置 ETCCSH 寄存器多次匹配中断标志。

2.5.2 OCU 中断

OCU 的每个通道都有一个中断。输出比较发生时，设置中断标志。

如果启用中断，中断的优先级应在 vector.c 文件中设置。下表列出了中断向量。

表 2-3. 中断向量

中断源	中断请求编号	中断优先级设置寄存器		向量表地址	
		记数器	设置位	高位	低位
输出比较 ch0 与匹配	IRQ07	ILR07	L07 [1:0]	FFECH	FFEDH
输出比较 ch1 与匹配	IRQ08	ILR08	L08 [1:0]	FFEAH	FFEBH
16 位定时器	IRQ14	ILR14	L14 [1:0]	FFDEH	FFDFH

3 软件操作

本章介绍了输出比较单元的软件操作。

OCU 有一个用于提供时基的 16 位定时器，设置该定时器是先决条件。 本文档给出了 FRT 的范例代码。参见“16 位定时器”了解关于该定时器的更多信息。

3.1 设置程序

3.1.1 16 位定时器

通过 TCCSL_CLK0~CLK2 设置定时器计数时钟
通过 ETCCSL_CNTMD 选择 FRT 计数模式
烧写比较清除值至 CPCLRH/L
通过 TCCSL_CLR 清除定时器计数器
如有必要，启用缓冲器
如有必要，启用溢出或过零中断
如有必要，设置溢出或过零中断标志
配置 OCU 后，启用定时器

3.1.2 输出比较单元

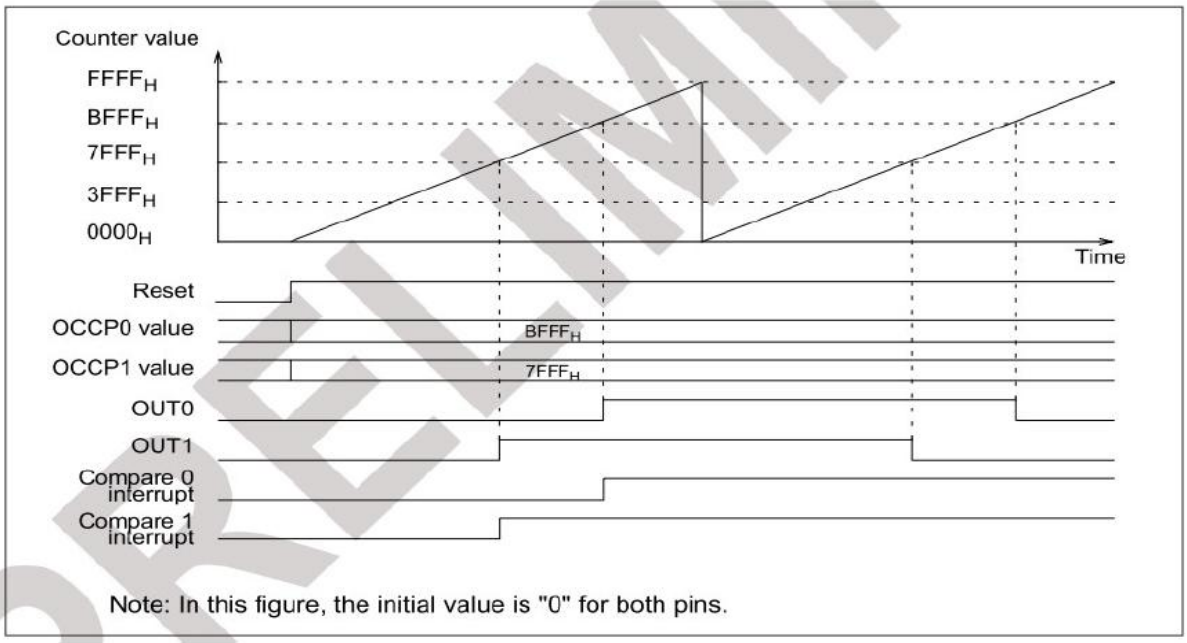
选择输出引脚
烧写输出比较值至 OCCP0 和 OCCP1
通过 OCMCR_CMPMDn 设置比较匹配输出
如有必要，启用输出比较缓冲器
如果缓冲器由 EOCS_BTSn 启用，选择数据传送点
通过 OCSH_CMOD0 为通道 1 定义比较输出模式
使用 OCSH_OTDn 选择输出比较引脚的初始水平
如有必要，选择硬件停止触发器来源
如有必要，启用中断
通过 OCSH_OTE 启用输出比较引脚
通过 OCSL_CSTn 启用输出比较操作

3.2 操作模式

3.2.1 有相位差的输出脉冲序列

图 3-1 显示了有相位差的脉冲序列。

图 3-1. 有相位的差脉冲序列



如图 3-1 所示，OUT0 和 OUT1 的周期和占空比相同，但上升沿和下降沿发生的点不同。 上升沿间的不同就是相位差。为输出期望的序列，16 位定时器在向上计数模式下运行。如果时钟输入相同，比较清除记数器的值将决定脉冲序列的周期。用户可通过烧写不同的值至寄存器 CPCLRH/L，修改比较清除记数器的值。OCCP0 和 OCCP1 的不同造成脉冲序列的相位差。

下式用于计算周期（向上计数模式）

Period =2* (1/MCLK)*(1/Pre-scale)*CPCLRH/L. (3-1)

预分频为 16 位定时器的预分频。

因此，

CPCLRH/L = Period*MCLK*Pre-scale/2. (3-2)

如果：

FCH = 4M， MCLK =4M，频率 = 20k，预分频 =1/4

则有：

CPCLH/L = 25

- 初始化

图 3-2 中的代码说明了配置 FRT 和 OCU 模块的步骤。P70 和 P73 用作输出引脚，定时器处于向上计数模式，预分频为 1 ：4，缓冲器被启用。16 位定时器和 OCU 功能在初始化代码中被禁用。

注意：务必在启用比较操作之前，写一个值至比较寄存器。

图 3-2. 初始化代码

```
SYSC2_OUTSEL0 = 0;              // select OUT0 output pin
SYSC2_OUTSEL1 = 0;              // select OUT1 output pin
DDR7_P70 = 1;                   // OUT0 output
DDR7_P73 = 1;                   // OUT1 output
PDR7_P70 = 0;                   // OUT0 output 0 first
PDR7_P73 = 0;                   // OUT1 output 0 first

TCCSH = 0x40;                   // count clock have no division
TCCSL = 0x2A;                   // clear timer
                                // disable timer first
                                // count clock=1/4*MCLK
ETCCSH = 0x0;                   // no interrupt mask
ETCCSL = 0x08;                  // up count mode
                                // interrupt disabled
CPCLRH = 0xFF;                  // write upper register first
CPCLRL = 0xFF;                  // then down register

OCSL = 0x00;                    // disable compare operation first
OCSH = 0x00;                    // initialize ch0&ch1 to 0
                                // disable output
                                // ch1 reverses upon comp reg1
EOCS = 0x11;                    // buffer enabled
                                // transfer data at zero point
OCCP0H = 0xBF;                  // write upper register first
OCCP0L = 0xFF;                  // then down register

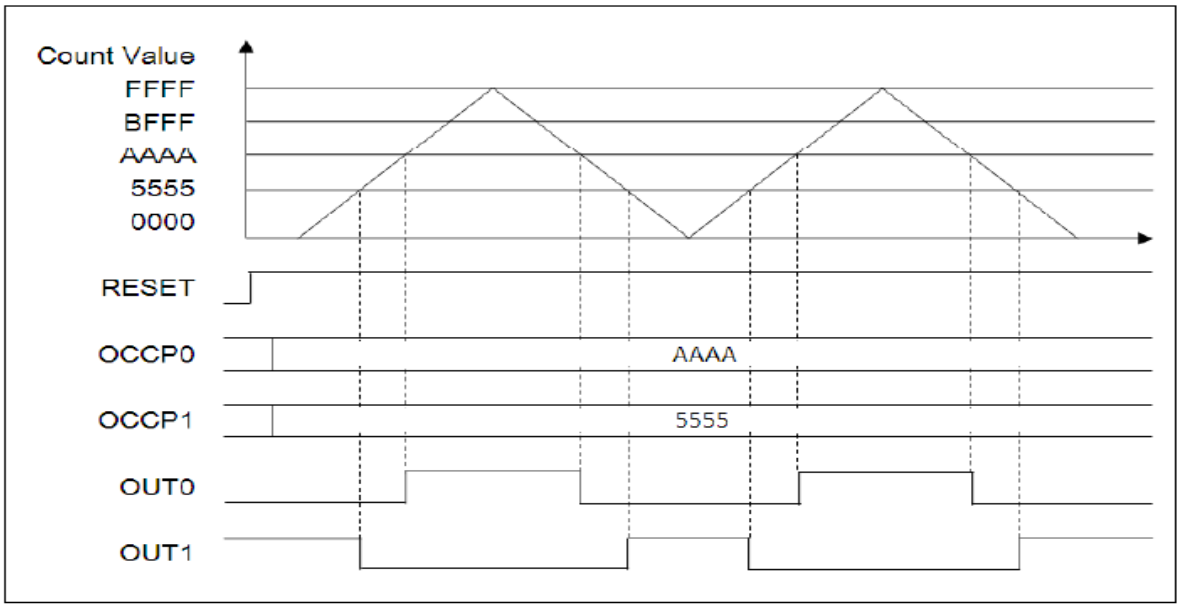
OCCP1H = 0x7F;
OCCP1L = 0xFF;
```

3.2.2 有死区时间的脉冲序列

有死区时间的脉冲序列通常用于驱动 H 桥。OCU 不能输出两个通道的 PWM 信号，但可以改变两个脉冲序列的频率。因此，PFM 方法对于驱动 H 桥非常有用。图 3-3 显示了波形。

CPCLRH/L 可通过式（3-2）算出，脉冲序列的占空比为 50%。

图 3-3. 有失效时间的脉冲序列



死区时间的计算如下：

Dead Time = (OCCP0-OCCP1) *(1/MCLK)*(1/Pre-scale)

改变 OCCP0-OCCP1 的值将改变死区时间。

- 初始化

16 位定时器处于上/下模式。CPCLRH/L 的缺省值为 0XFFFF。OCCP0 和 OCCP1 分别烧写为 0XAAAA 和 0X5555，该值的不同导致死区时间的产生。 图 3-4 显示了初始化代码。

图 3-4. 有死区时间的脉冲序列的初始化

```
SYSC2_OUTSEL0 = 0;                  // select OUT0 output pin
SYSC2_OUTSEL1 = 0;                  // select OUT1 output pin
DDR7_P70 = 1;                       // OUT0 output
DDR7_P73 = 1;                       // OUT1 output
PDR7_P70 = 0;                       // OUT0 output 0 first
PDR7_P73 = 0;                       // OUT1 output 0 first

TCCSH = 0x40;                       // count clock have no division
TCCSL = 0x2A;                       // clear timer
                                    // disable timer first
                                    // count clock=1/4*MCLK
ETCCSH = 0x0;                       // no interrupt mask
ETCCSL = 0x09;                      // up/down count mode
                                    // interrupt disabled
CPCLRH = 0xFF;                      // write upper register first
CPCLRL = 0xFF;                      // then down register

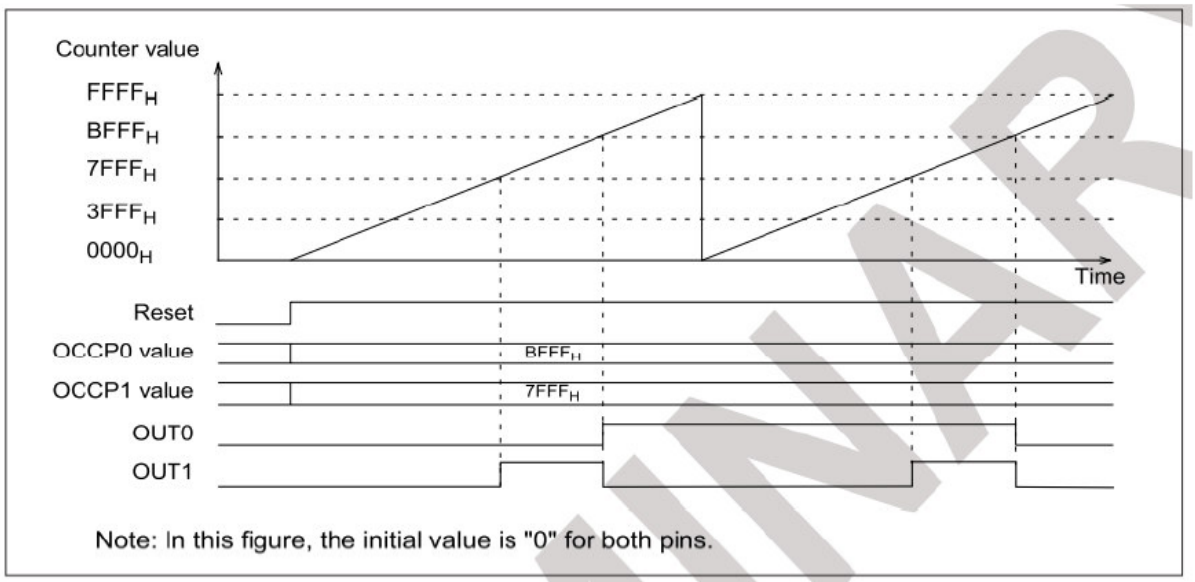
OCSL = 0x00;                        // disable compare operation first
OCSH = 0x00;                        // initialize ch0&ch1 to 0
                                    // disable output
                                    // ch1 reverses upon comp reg1
EOCS = 0x11;                        // buffer enabled
                                    // transfer data at zero point
OCCP0H = 0xAA;                      // write upper register first
OCCP0L = 0xAA;                      // then down register

OCCP1H = 0x55;
OCCP1L = 0x55;
```

3.2.3 输出 PWM

OCSH_CMOD0 位设置为 1 时，OCU 模块可通过 OUT1 产生一个通道的 PWM。图 3-5 显示了 PWM 信号波形。

图 3-5. PWM 信号波形



周期的计算公式如下：

Period = (1/MCLK)*(1/Pre-scale)*CPCLRH/L (3-3)

预分频是 16 位定时器的预分频

因此，

CPCLRH/L = Period*MCLK*Pre-scale (3-4)

如果 16 位定时器处于上/下模式，应使用式（3-1）计算周期。

改变 OCCP0 的值将改变占空比。保持 OCCP1 不变，修改 OCCP0 为 0XFFFF，OUT1 的占空比将变大。

- 初始化

在该模式下，输出通道 1 配置为与比较记数器 0，1 匹配时反向。改变 OCCP0 的值将改变 PWM 波形的占空比。

图 3-6 显示了 PWM 的初始化代码。

图 3-6. PWM 的初始化代码

```
SYSC2_OUTSEL0 = 0;              // select OUT0 output pin
SYSC2_OUTSEL1 = 0;              // select OUT1 output pin
DDR7_P70 = 1;                   // OUT0 output
DDR7_P73 = 1;                   // OUT1 output
PDR7_P70 = 0;                   // OUT0 output 0 first
PDR7_P73 = 0;                   // OUT1 output 0 first

TCCSH = 0x40;                   // count clock have no division
TCCSL = 0x2A;                   // clear timer
                                // disable timer first
                                // count clock=1/4*MCLK
ETCCSH = 0x0;                   // no interrupt mask
ETCCSL = 0x08;                  // up count mode
                                // interrupt disabled
CPCLRH = 0xFF;                  // write upper register first
CPCLRL = 0xFF;                  // then down register

OCSL = 0x00;                    // disable compare operation first
OCSH = 0x40;                    // initialize ch0&ch1 to 0
                                // disable output
                                // ch1 reverses upon comp reg0,1
EOCS = 0x11;                    // buffer enabled
                                // transfer data at zero point
OCCP0H = 0xBF;                  // write upper register first
OCCP0L = 0xFF;                  // then down register

OCCP1H = 0x7F;
OCCP1L = 0xFF;
```

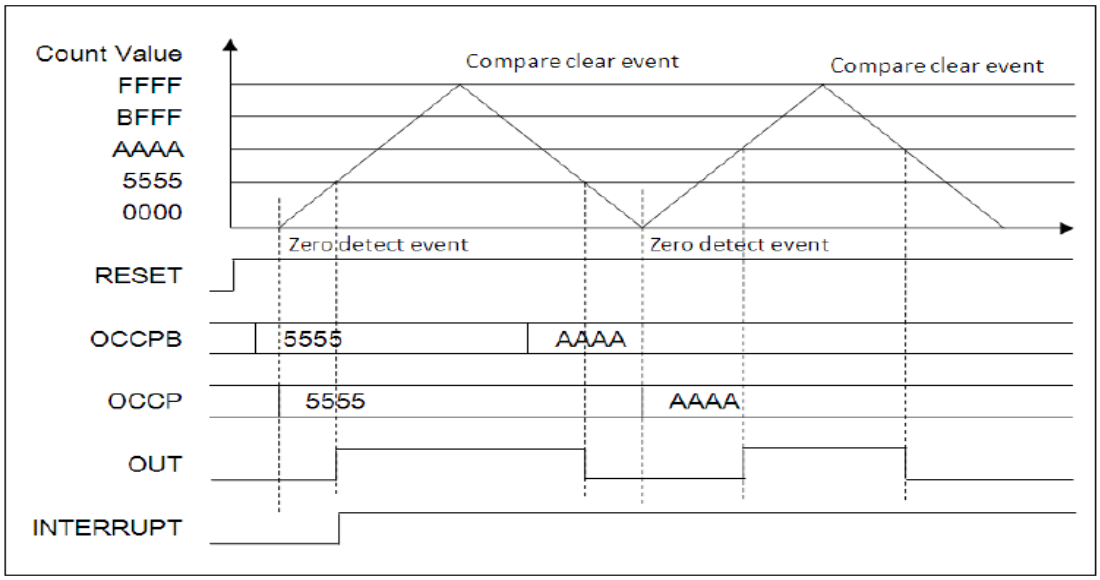
4 部分位的影响

4.1 EOCS_BTSn

EOCS_BTSn 决定了启用缓冲器功能时，输出比较缓冲器寄存器的值转移至输出比较寄存器的时间点。

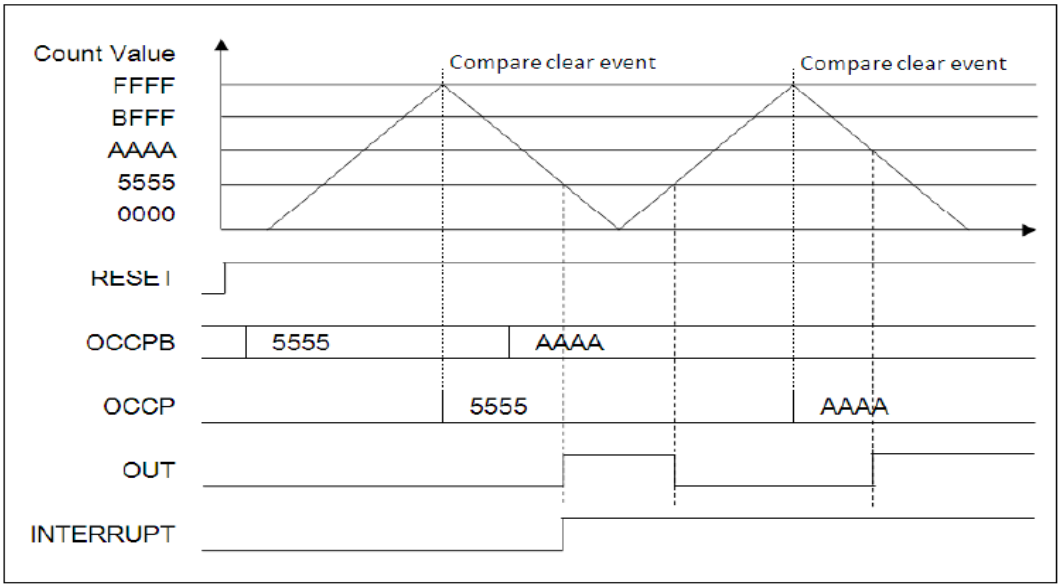
- EOCS_BTSn = 0，转移发生在 FRT 的过零点。

图 4-1. 过零点的转移



EOCS_BTSn = 1，转移发生在 FRT 的溢出点。

图 4-2. 在溢出点转移



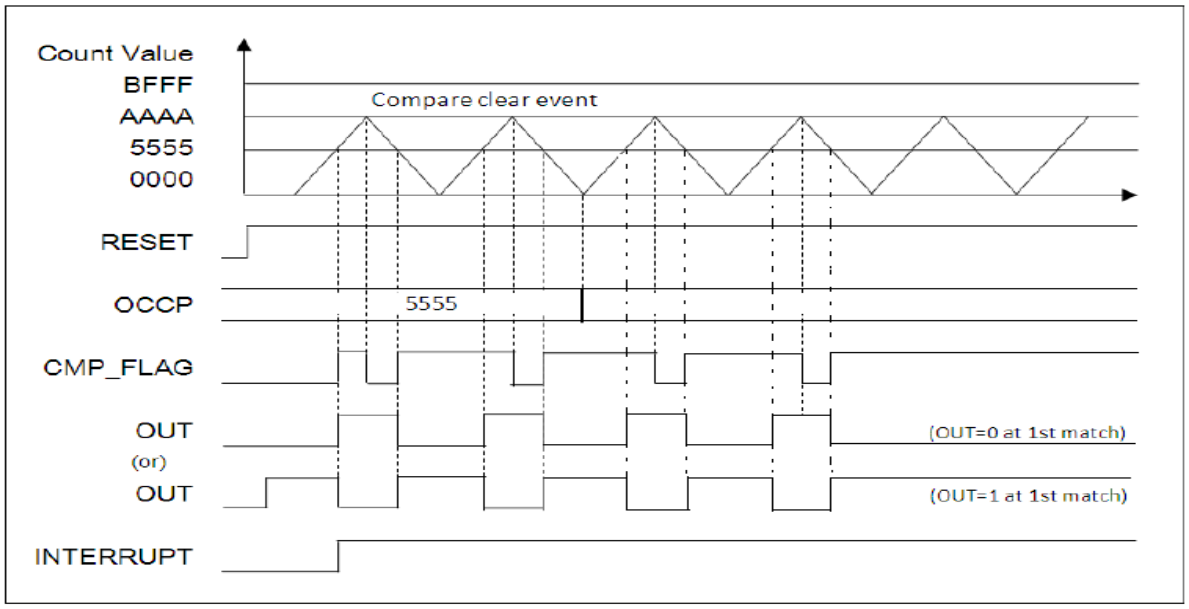
4.2 OCMCR_CMPMDn

启用引脚输出时，该位用于在匹配发生后立即改变引脚的输出水平。

- OCMCR_CMPMDn = 0

在该模式下，引脚水平与之前的值相反。因此，需注意引脚的初始水平。图 4-3 显示了信号波形。

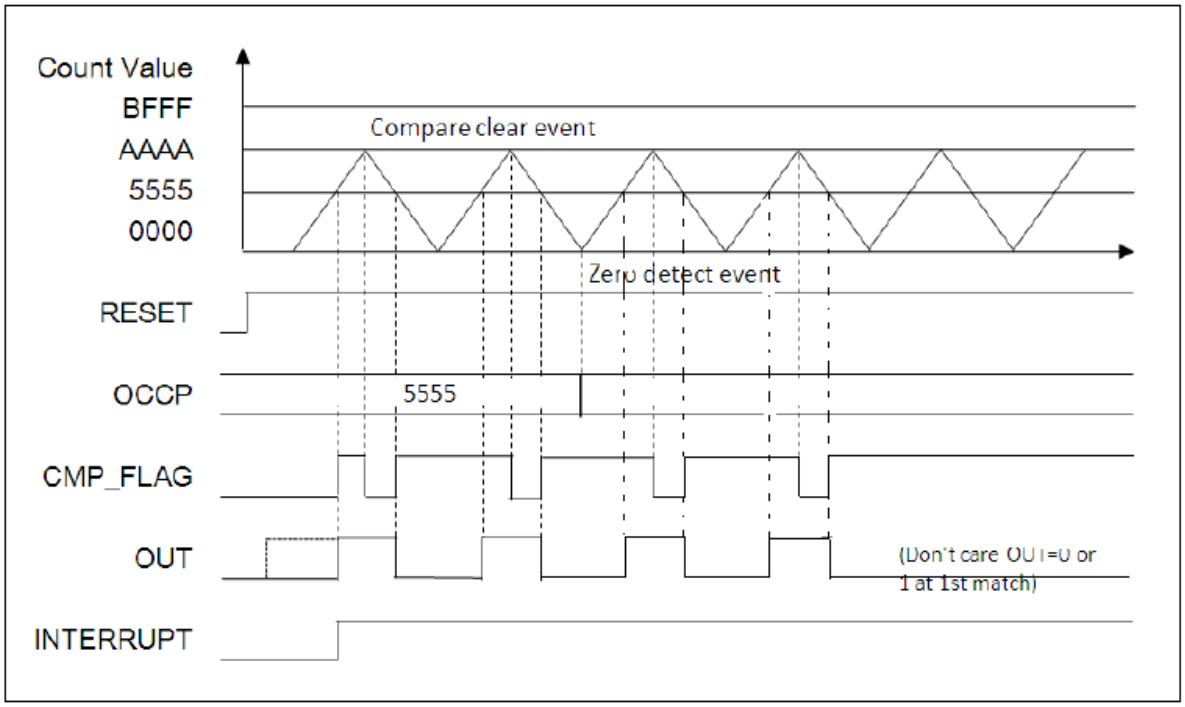
图 4-3. CMPMDn=0 的信号波形



■ OCMCR_CMPMDn = 1

16 位定时器处于向上计数模式时，输出引脚水平设置为 1；16 位定时器处于向下计数模式时，输出引脚水平被清除。该模式不考虑输出引脚的初始值。

图 4-4. CMPMDn= 1 的信号波形



4.3 OCMCR_FDENn (n= 0, 1)

该位用于 ch0 或 ch1 的满占空比功能。

FDENn = 0 时，满占空比功能特性被禁用。

FDENn =1 时，满占空比功能特性被启用（必须设置 OCSH_CMOD0 = 0）。

如果 FDENn = 0，

16 位定时器的值与比较寄存器（OCCPn）匹配时，触发输出水平。

如果 FDENn =1， 且 COMD0=0，

OCCPn= 0 时，OCU 的输出与 16 位定时器的值无关。

OUTn = OCMCR_INVn

0<OCCP0<CPCLR 时，

OCU 的输出取决于 CMPMDn 位。

OCCP0>= CPCLR 时，

OCU 的输出取决于比较清除事件的 OCU 比较标志。

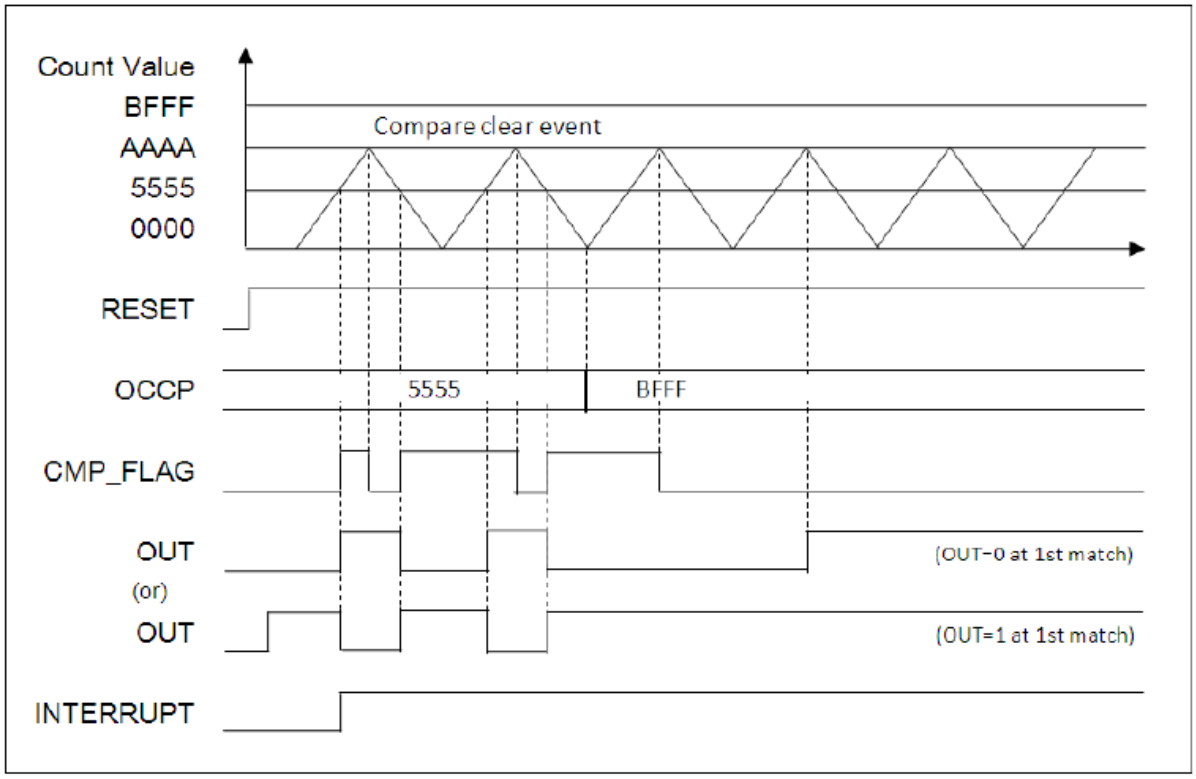
如果 CMP_FLAG =0，OUTn =-OCMCR_INVn

如果 CMP_FLAG =1， OUTn 不变。

CMP_FLAG 由比较匹配事件设置，由比较清除事件重置。

图 4-5 显示了 FDEN =1，CMPMD0= 0，INV0 = 0 时的信号波形。

图 4-5. 满占空比的信号波形



5 更多信息

如欲了解有关 Cypress 产品的更多详情，敬请访问以下网址：

www.cypress.com/cypress-microcontrollers

A 附录

A.1 范例代码

Project1: Pulse sequences with phase difference

```
#include "mb95430.h"
/*------------------------------------------------------------------------------
Name        : InitFRTandOCU ()
Function : Initialize the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------------*/
void InitFRTandOCU (void)
{
    SYSC2_OUTSEL0 = 0;                  // select OUT0 output pin
    SYSC2_OUTSEL1 = 0;                  // select OUT1 output pin
    DDR7_P70 = 1;                       // OUT0 output
    DDR7_P73 = 1;                       // OUT1 output
    PDR7_P70 = 0;                       // OUT0 output 0 first
    PDR7_P73 = 0;                       // OUT1 output 0 first

    TCCSH = 0x40;                       // count clock have no division
    TCCSL = 0x2A;                       // clear timer
                                        // disable timer first
                                        // count clock=1/4*MCLK

    ETCCSH = 0x0;                       // no interrupt mask
    ETCCSL = 0x08;                      // up count mode
                                        // interrupt disabled
    CPCLRH = 0xFF;                      // write upper register first
    CPCLRL = 0xFF;                      // then down register

    OCSL = 0x00;                                // disable compare operation first
    OCSH = 0x00;                                // initialize ch0&ch1 to 0
                                        // disable output

                                        // ch1 reverses upon comp reg1
    EOCS = 0x11;                                // buffer enabled
                                        // transfer data at zero point
    OCCP0H = 0xBF;                      // write upper register first
    OCCP0L = 0xFF;                      // then down register

    OCCP1H = 0x7F;
    OCCP1L = 0xFF;
}
/*------------------------------------------------------------------------------
Name        : EnableFRTandOCU ()
Function : Enable the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------------*/
void EnableFRTandOCU (void)
{
    TCCSL_CLR = 1;                      // clear FRT counter

    /*
    take care here, strongly recommend to write 0 to TCDTH/L !!!!
```

```
    */
    TCDTH = 0;
    TCDTL = 0;

    OCSH = 0x28;                                    // output channels are enabled
                                                    // if HW_STOP=0
    OCSL_CST0 = 1;                                  // enable output compare0 operation
    OCSL_CST1 = 1;                                  // enable output compare1 operation

    TCCSL_STOP = 1;                                 // start free-run timer counting
}
/*------------------------------------------------------------------------------
Name        : DisableFRTandOCU ()
Function: Disable the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------------*/
void DisableFRTandOCU(void)
{
    OCSH = 0;                                       // output channels are disabled
                                                    // the pin are controlled by GPIO
    TCCSL_STOP = 0;                                 // stop free-run timer counting

    OCSL_CST0 = 0;                                  // disable operation
    OCSL_CST1 = 0;
}
/* delay */
void Delayms (unsigned char cnt_Dly)
{
    unsigned char i;
    for(; cnt_Dly > 0 ; cnt_Dly-- )
    {
        for(i = 250 ; i > 0 ; i-- )
        {
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
        }
    }
}

main()
{
    __DI();                                         // disable interrupt
    SYCC = 0x01;                                    // MCLK = (1/4)*8M
    InitFRTandOCU ();                       // enable interrupt
    __EI();

    while(1)
    {
        EnableFRTandOCU ();
        Delayms(100);
        DisableFRTandOCU ();
        Delayms(100);
    }
}
```

Project 2: Output Pulse Sequences with Dead Time

```
#include "mb95430.h"
/*------------------------------------------------------------------------
Name        : InitFRTandOCU ()
Function : Initialize the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------*/
void InitFRTandOCU (void)
{
    SYSC2_OUTSEL0 = 0;                  // select OUT0 output pin
    SYSC2_OUTSEL1 = 0;                  // select OUT1 output pin
    DDR7_P70 = 1;                       // OUT0 output
    DDR7_P73 = 1;                       // OUT1 output
    PDR7_P70 = 0;                       // OUT0 output 0 first
    PDR7_P73 = 0;                       // OUT1 output 0 first

    TCCSH = 0x40;                       // count clock have no division
    TCCSL = 0x2A;                       // clear timer
                                        // disable timer first
                                        // count clock=1/4*MCLK

    ETCCSH = 0x0;                       // no interrupt mask
    ETCCSL = 0x08;                      // up count mode
                                        // interrupt disabled
    CPCLRH = 0xFF;                      // write upper register first
    CPCLRL = 0xFF;                      // then down register

    OCSL = 0x00;                                // disable compare operation first
    OCSH = 0x40;                                // initialize ch0&ch1 to 0
                                        // disable output
                                        // ch1 reverses upon comp reg0,1
    EOCS = 0x11;                                // buffer enabled
                                        // transfer data at zero point
    OCCP0H = 0xBF;                      // write upper register first
    OCCP0L = 0xFF;                      // then down register

    OCCP1H = 0x7F;
    OCCP1L = 0xFF;
}
/*------------------------------------------------------------------------
Name        : EnableFRTandOCU ()
Function : Enable the Free-Run Timer and OCU module
Input       : void
Output      : void
------------------------------------------------------------------------*/
void EnableFRTandOCU(void)
{
    TCCSL_CLR = 1;                      // clear FRT counter
    /*
    take care here, strongly recommend to write 0 to TCDTH/L !!!!
    */
    TCDTH = 0;
    TCDTL = 0;

    OCSH = 0x68;                                // output channels are enabled
                                        // if HW_STOP=0
    OCSL_CST0 = 1;                      // enable output compare0 operation
```

```
    OCSL_CST1 = 1;                          // enable output compare1 operation

    TCCSL_STOP = 1;                         // start free-run timer counting
}
/*------------------------------------------------------------------------
Name        : DisableFRTandOCU ()
Function : Disable the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------*/
void DisableFRTandOCU (void)
{
    TCCSL_STOP = 0;                         // stop free-run timer counting
    OCSH = 0x80;                                    // output channels are disabled
                                            // the pin are controlled by GPIO

    OCSL_CST0 = 0;                          // disable operation
    OCSL_CST1 = 0;
}
/*delay*/
void Delayms (unsigned char cnt_Dly)
{
    unsigned char i;
    for(; cnt_Dly > 0 ; cnt_Dly-- )
    {
        for(i = 250 ; i > 0 ; i-- )
        {
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
        }
    }
}
main()
{
    __DI();                                 // disable interrupt
    SYCC = 0x01;                                    // MCLK = (1/4)*8M
    InitFRTandOCU ();                       // enable interrupt
    __EI();

    while(1)
    {
        EnableFRTandOCU ();
        Delayms(100);
        DisableFRTandOCU ();
        Delayms(100);
    }
}
```

Project 3: Output PWM

```
#include "mb95430.h"
/*------------------------------------------------------------------------
Name          : InitFRTandOCU ()
Function : Initialize the Free-Run Timer and OCU module
Input         : void
Output              : void
------------------------------------------------------------------------*/
void InitFRTandOCU(void)
{
    SYSC2_OUTSEL0 = 0;                  // select OUT0 output pin
    SYSC2_OUTSEL1 = 0;                  // select OUT1 output pin
    DDR7_P70 = 1;                       // OUT0 output
    DDR7_P73 = 1;                       // OUT1 output
    PDR7_P70 = 0;                       // OUT0 output 0 first
    PDR7_P73 = 0;                       // OUT1 output 0 first

    TCCSH = 0x40;                       // count clock have no division
    TCCSL = 0x2A;                       // clear timer
                                                // disable timer first
                                                // count clock=1/4*MCLK

    ETCCSH = 0x0;                       // no interrupt mask
    ETCCSL = 0x08;                      // up count mode
                                                // interrupt disabled
    CPCLRH = 0xFF;                      // write upper register first
    CPCLRL = 0xFF;                      // then down register

    OCSL = 0x00;                                // disable compare operation first
    OCSH = 0x40;                                // initialize ch0&ch1 to 0
                                        // disable output

                                        // ch1 reverses upon comp reg0,1
    EOCS = 0x11;                                // buffer enabled
                                        // transfer data at zero point
    OCCP0H = 0xBF;                      // write upper register first
    OCCP0L = 0xFF;                      // then down register

    OCCP1H = 0x7F;
    OCCP1L = 0xFF;
}
/*------------------------------------------------------------------------
Name          : EnableFRTandOCU ()
Function : Enable the Free-Run Timer and OCU module
Input         : void
Output              : void
------------------------------------------------------------------------*/
void EnableFRTandOCU(void)
{
    TCCSL_CLR = 1;                      // clear FRT counter
    /*
    take care here, strongly recommend to write 0 to TCDTH/L !!!!
    */
    TCDTH = 0;
    TCDTL = 0;

    OCSH = 0x68;                                // output channels are enabled
                                        // if HW_STOP=0
```

```
    OCSL_CST0 = 1;                          // enable output compare0 operation
    OCSL_CST1 = 1;                          // enable output compare1 operation

    TCCSL_STOP = 1;                         // start free-run timer counting
}
/*------------------------------------------------------------------------
Name        : DisableFRTandOCU ()
Function : Disable the Free-Run Timer and OCU module
Input       : void
Output          : void
------------------------------------------------------------------------*/
void DisableFRTandOCU(void)
{
    TCCSL_STOP = 0;                         // stop free-run timer counting
    OCSH = 0x80;                                    // output channels are disabled
                                            // the pin are controlled by GPIO

    OCSL_CST0 = 0;                          // disable operation
    OCSL_CST1 = 0;
}
/*delay*/
void Delayms (unsigned char cnt_Dly)
{
    unsigned char i;
    for(; cnt_Dly > 0 ; cnt_Dly-- )
    {
        for(i = 250 ; i > 0 ; i-- )
        {
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
            __wait_nop ();
        }
    }
}
main()
{
    __DI();                                 // disable interrupt
    SYCC = 0x01;                                    // MCLK = (1/4)*8M
    InitFRTandOCU ();               // enable interrupt
    __EI();

    while(1)
    {
        EnableFRTandOCU ();
        Delayms(100);
        DisableFRTandOCU ();
        Delayms(100);
    }
}
```

文档修改记录

文档标题：AN205007 - F²MC-8FX 家族 MB95430 系列 16 位 FRT 和 OCU 应用手册

文档编号：002-05710

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/12/2010	初稿
			04/12/2010	更新
			09/27/2010	更新源代码
*A	5327330	HUAL	06/30/2016	已将 Spansion 应用手册《MCU-AN-500077-Z-12》转换成 Cypress 格式。

全球销售和设计支持

赛普拉斯公司拥有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。如果想要查找离您最近的办事处，请访问赛普拉斯所在地。

产品

ARM® Cortex® 微控制器	cypress.com/arm
汽车级	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
照明和电源控制	cypress.com/powerpsoc
存储器	cypress.com/memory
PSoC	cypress.com/psoc
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线/射频	cypress.com/wireless

PSoC® 解决方案

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

赛普拉斯开发者社区

论坛 | 项目 | 视频 | 博客 | 培训 | 组件

技术支持

cypress.com/support

PSoC 是赛普拉斯半导体公司的注册商标。PSoC Creator 是赛普拉斯半导体公司的商标。 此处引用的所有其他商标或注册商标都归其各自所有者所有。

赛普拉斯半导体
198 Champion Court
San Jose, CA 95134-1709

电话 : 408-943-2600
传真 : 408-943-4730
网站地址 : www.cypress.com

©赛普拉斯半导体公司，2010-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属个人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码的形式向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。
在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。在适用法律允许的限度内，赛普拉斯保留更改本文件的权利，届时将不另行通知。赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失的其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何索赔、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的索赔，并使之免受损失。
赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。敬请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。