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.

AN205275

F²MC-8FX 家族 MB95200H/210H 系列 8/16 位多功能定时器

本文档介绍了如何在各种模式下使用 8/16 位多功能定时器。本文档同时介绍了 8/16 位多功能定时器的特性和一些实例。

目录

1 概要 1
2 8/16 位多功能定时器 1
2.1 主要特性.....1
2.2 结构图 2
2.3 寄存器 3
2.4 操作模式.....4
3 操作和实例.....4
3.1 间隔定时器功能的操作（单次模式）.....4
3.2 间隔定时器的操作（连续模式）.....6
3.3 间隔定时器功能的操作（自由运行模式）.....8
3.4 PWM 定时器功能的操作（固定周期模式）.....10
3.5 PWM 定时器功能的操作（可变周期模式）.....11
3.6 PWC 定时器功能的操作.....13
3.7 输入捕捉功能的运算.....15
4 使用 8/16 位多功能定时器的注意事项.....18
5 更多信息 18
A 附录 19
A.1 代码范例 19
文档修改记录.....29

1 概要

本文档介绍了如何在各种模式下使用 8/16 位多功能定时器。

本文档同时介绍了 8/16 位多功能定时器的特性和一些实例。

2 8/16 位多功能定时器

本章介绍了 8/16 位多功能定时器的基本功能。

2.1 主要特性

8/16 位多功能定时器包括两个 8 位计数器，可以用作两个 8 位定时器或者级联后用作一个 16 位定时器。

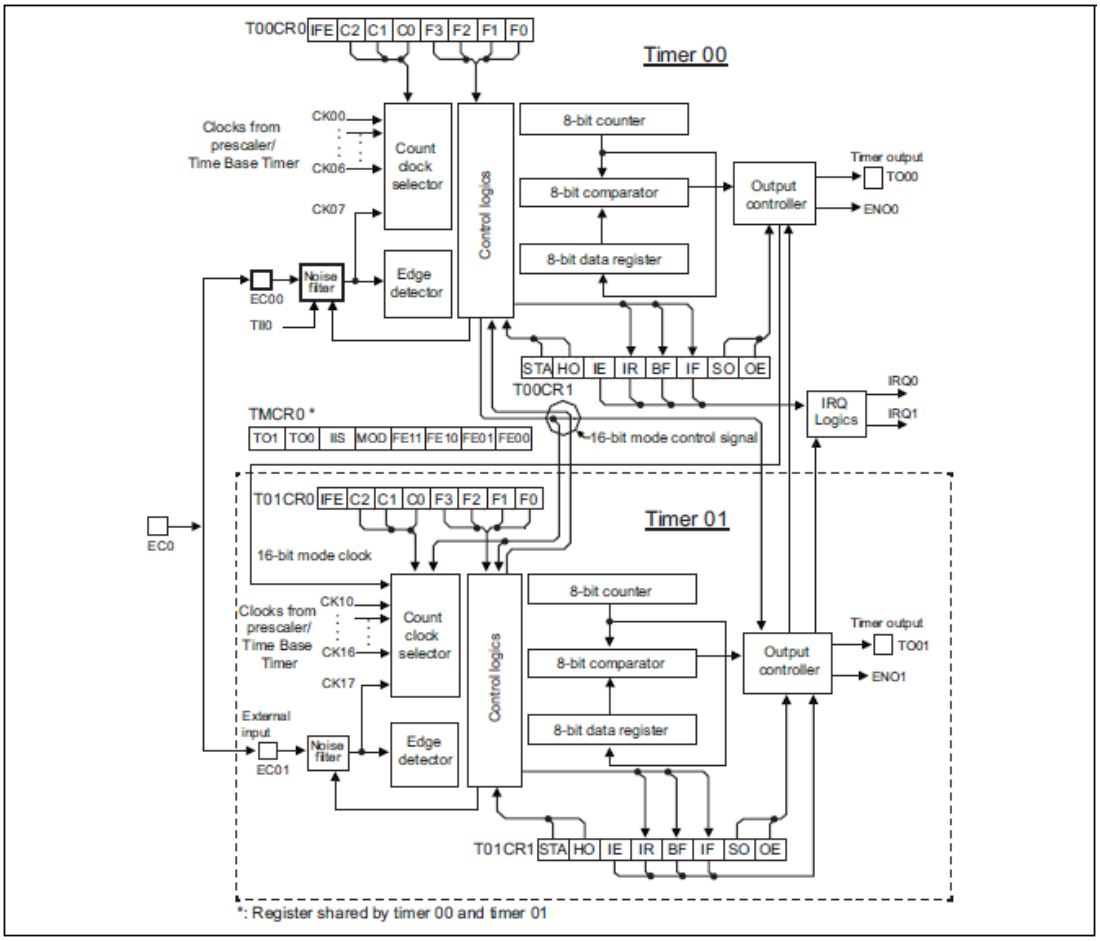
8/16 位多功能定时器有以下功能。

- 间隔定时器功能
- PWM 定时器功能
- PWC 定时器功能（脉冲宽度测定）
- 输入捕捉功能

2.2 结构图

图 1 显示了 8/16 位多功能定时器的内部结构图。

图 1. 8/16 位多功能定时器结构图



2.3 寄存器

参见 MB95200H/210H 系列硬件手册的第 14 章了解关于寄存器设置的更多信息。

2.3.1 8/16 位多功能定时器 00/01 控制状态寄存器 0（T00CR0/T01CR0）

该寄存器用于选择定时器操作模式，选择计数时钟以及启用或禁用 IF 标记中断。寄存器 T00CR0 对应定时器 00；寄存器 T01CR0 对应定时器 01。

图 2. T00CR0/T01CR0

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0F92H T01CR0	IFE	C2	C1	C0	F3	F2	F1	F0	00000000_B
0F93H T00CR0	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

2.3.2 8/16 位多功能定时器 00/01 控制状态寄存器 1（T00CR1/T01CR1）

该寄存器用于控制中断标记，定时器输出以及定时器运算。寄存器 T00CR1 对应定时器 00 ； T01CR1 对应定时器 01 。

图 3. T00CR1/T01CR1

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0036H T01CR1	STA	HO	IE	IR	BF	IF	SO	OE	00000000_B
0037H T00CR1	R/W	R/W	R/W	R (RM1), W	R/WX	R (RM1), W	R/W	R/W	

2.3.3 8/16 位多功能定时器 00/01 定时器模式控制寄存器（TMCR0）

该寄存器用于选择噪声滤波器功能，8 位或 16 位操作模式，输入到定时器 00 的信号，以及定时器输出值。该寄存器服务于定时器 00 和定时器 01。

图 4. TMCR0

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0F96H TMCR0	TO1	TO0	IIS	MOD	FE11	FE10	FE01	FE00	00000000_B
	R/WX	R/WX	R/W	R/W	R/W	R/W	R/W	R/W	

2.3.4 8/16 位多功能定时器 00/01 数据寄存器 ch.0（T00DR/T01DR）

该寄存器用于在间隔定时器或 PWM 定时器操作期间烧写计数的最大值，以及在 PWC 定时器或输入捕捉操作期间读取计数值。寄存器 T00DR 对应定时器 00；T01DR 对应定时器 01。

图 5. T00DR/T01DR

地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	初始值
0F94H T01DR	TDR7	TDR6	TDR5	TDR4	TDR3	TDR2	TDR1	TDR0	00000000_B
0F95H T00DR	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W： 可读/可写（读值与写值相同）

R/WX： 只读（可读，烧写操作无效）

R（RM1），W： 可读/可写（读值与写值不同，读取-修改-烧写指令读取“1”。）

2.4 操作模式

以下列出了 7 种操作模式下的 8/16 位多功能定时器功能。

- 间隔定时器功能（单次模式）
- 间隔定时器功能（连续模式）
- 间隔定时器功能（自由运行模式）
- PWM 定时器功能（固定周期模式）
- PWM 定时器功能（可变周期模式）
- PWC 定时器功能
- 输入捕捉功能

3 操作和实例

本章举例说明了 8/16 位多功能定时器。

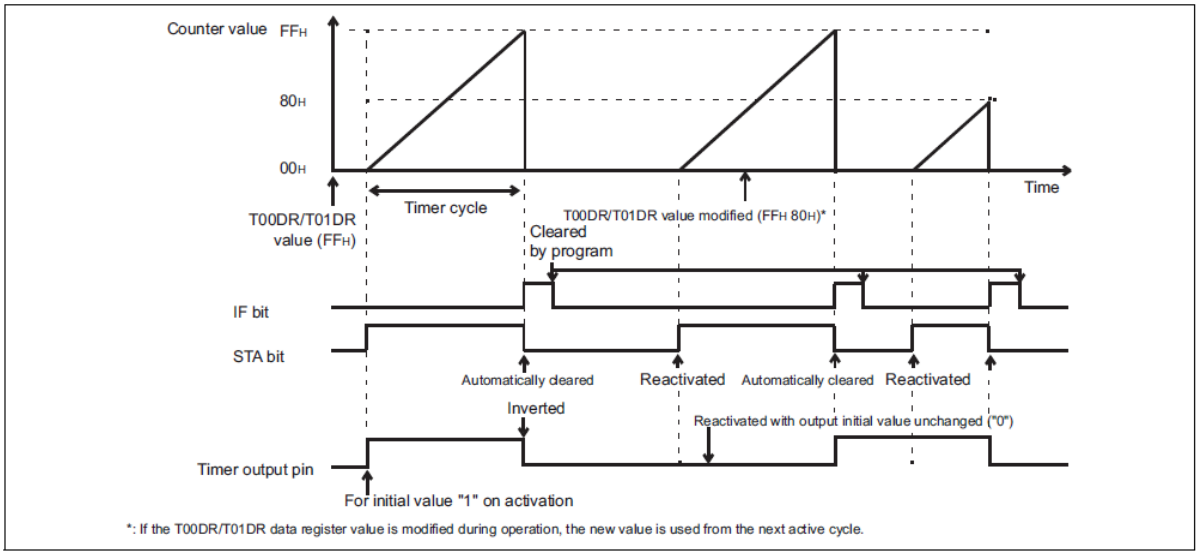
本部分描述了 8/16 位多功能定时器的各种模式并提供了一些实例。

3.1 间隔定时器功能的操作（单次模式）

如果选择间隔定时器功能（单次模式），定时器激活后，计数器从“00H”开始计数。当计数器值符合寄存器的设定值时，定时器输出反向。产生中断请求，计数器停止计数。

图 6 显示了 8 位模式中间隔定时器功能的操作。

图 6. 8 位模式中间隔定时器功能的操作（定时器 0）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器的间隔定时器功能（单次模式， 8 位）。

1. 设置模式（T00CR0 ： [F3..F0] = 0 0 0 0）
2. 设置计数器时钟（T00CR0 ： [C2..C0]）
3. 设置计数器值（T00DR）
4. 如有必要，设置中断级别（ILR1）
5. 启用定时器输出（T00CR1 ： OE = 1）
6. 启动定时器（T00CR1 ： STA = 1）

如果启用溢出中断，ISR 中的标志将被清除。（T00CR1 ： IF = 0）

以下代码显示了如何设置 8/16 位多功能定时器 0 在单次模式下的操作。

参见附录代码范例中的 One-shot 工程。

```
/* initial timer for interval timer (One-shot) function */
void InitCompTimer (void)
{
       T00DR = 0xFF;        // set the count value
       TMCR0 = 0x00;        // 8-bit, no filtering
       T00CR0 = 0x00;       // interval timer in one-shot mode
       T00CR1 = 0x01;       // disable interrupt, enable output
}

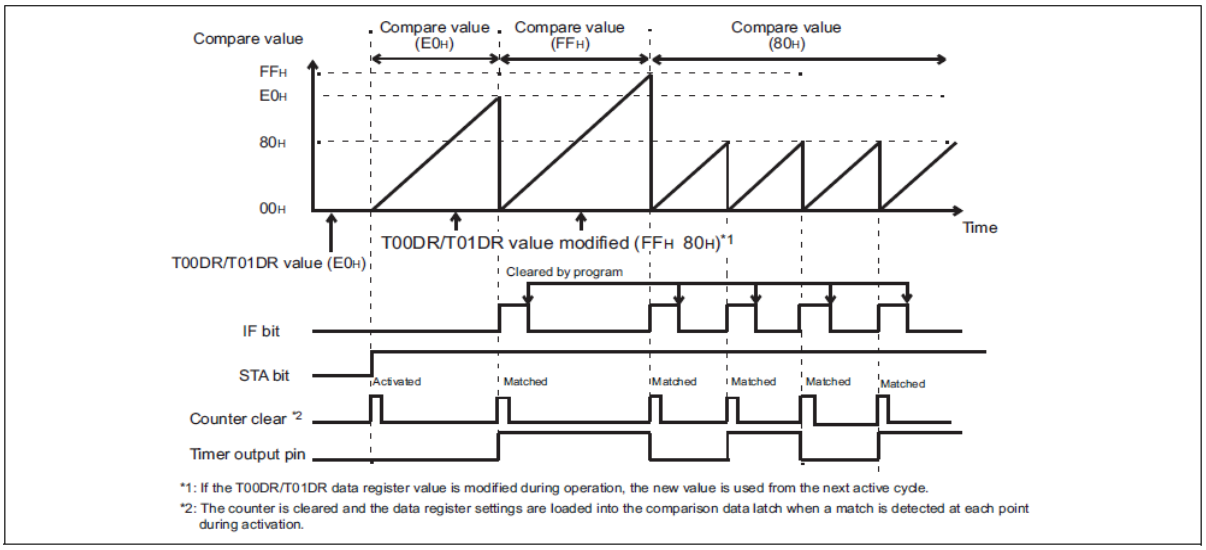

/* main routine */
void main (void)
{
       InitComTimer();
       ...
       T00CR1_STA = 1;      // start the timer
}
```

3.2 间隔定时器的操作（连续模式）

如果选择间隔定时器功能（连续模式），定时器激活后，计数器从“00H”开始计数。当计数器值符合寄存器的设定值时，定时器输出反向，产生中断请求，计数器返回“00H”并重新开始计数。此重复操作使定时器输出一个方波。

图 7 显示了 8 位模式下间隔定时器功能的操作。

图 7. 间隔定时器功能的操作图（连续模式）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器的间隔定时器功能（连续模式）。

1. 设置模式（T00CR0： [F3..F0] = 0 0 0 1）
2. 设置计数器时钟（T00CR0： [C2..C0]）
3. 设置计数器值（T00DR）
4. 如有必要，设置中断级别（ILR1）
5. 启用定时器输出（T00CR1： OE = 1）
6. 启动定时器（T00CR1： STA = 1）

如果启用溢出中断，ISR 中的标志将被清除。(T00CR1： IF = 0）

以下代码显示了如何设置 8/16 位多功能定时器 0 的中断操作（连续模式，16 位）。

```
/* initial timer for interval timer (continuous) function */
void InitCompTimer (void)
{
       T01DR = 0x01;        // set the count value (upper 8 bits)
       T00DR = 0xFF;        // set the count value (lower 8 bits)
       TMCR0 = 0x10;        // 16-bit, no filtering
       T00CR0 = 0x81;       // interval timer in the continuous mode
                            // enable IF flag interrupt
       T00CR1 = 0xA1;       // enable the interrupt, enable the output
                            // start the timer
}
/* enter ISR while the counter value is matchesthe pre-set value */
__interrupt void CompTimer (void)
{
       T00CR1_IE = 0;       // disable the interrupt
       T00CR1_IF = 0;       // clear the flag

       ...                  // interrupt service routine

       T00CR1_IE = 1;       // enable the interrupt
}
/* main routine */
void main (void)
{
       ...
       InitCompTimer();
       ...
       while (1);
}
```

注意：对应的中断向量和级别将在 Cypress 标准模板工程的 vector.c 模块中定义。

参见附录代码范例中的 Continuous 工程。

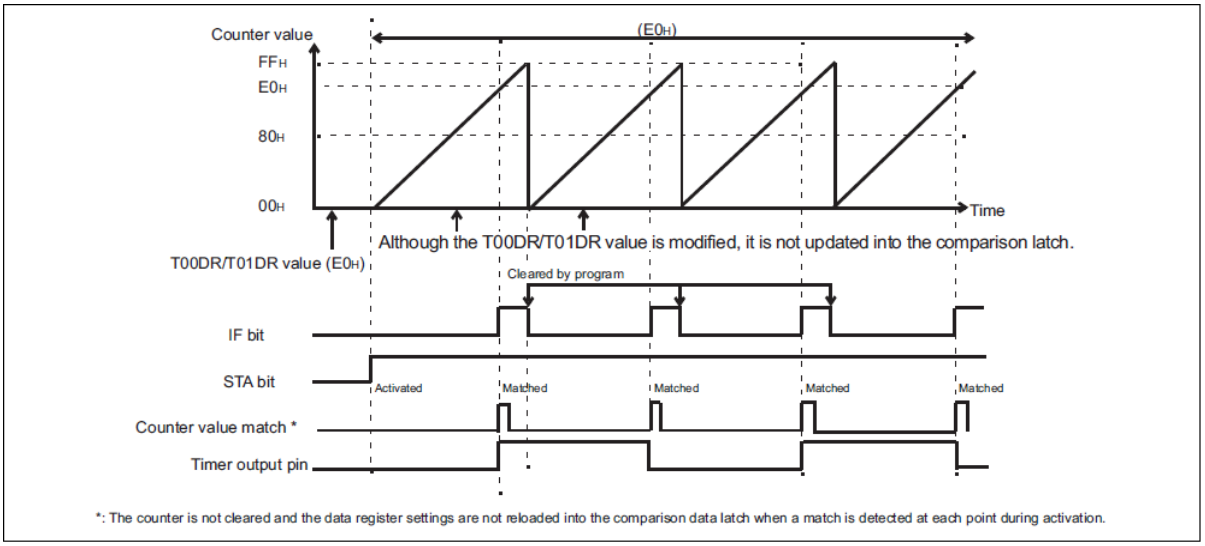
```
/* interrupt level setting */
void InitIrqLevels (void)
{
       ...
       ILR1 = 0xF3;         //IRQ5: 8/16-bit timer ch.0 (lower)
       ...
}
__interrupt void CompTimer (void);   //Prototype
       ...

#pragma intvect CompTimer  5   //8/16-bit timer ch.0 (lower)
       ...
```

3.3 间隔定时器功能的操作（自由运行模式）

如果选择间隔定时器功能（自由运行模式），计数器从“00H”开始计数。当计数器值符合寄存器设定值时，定时器输出反向，产生中断请求。计数器继续计数至“FFH”，然后返回“00F”并重新开始计数。此重复操作使定时器输出一个方波。

图 8. 间隔定时器功能的操作图（自由运行模式）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器的间隔定时器功能（自由运行模式）。

1. 设置模式（T00CR0： [F3..F0] = 0 0 1 0）
2. 设置计数器时钟（T00CR0： [C2..C0]）
3. 设置计数器值（T00DR）
4. 如有必要，设置中断级别（ILR1）
5. 启用定时器输出（T00CR1： OE = 1）
6. 启动定时器（T00CR1： STA = 1）

如果启用溢出中断，ISR 中的标志将被清除。（T00CR1： IF = 0）

以下代码显示了如何设置 8/16 位多功能定时器 0 在自由运行模式下的操作。

```
/* initial timer for interval timer (continuous) function */
void InitCompTimer (void)
{
      T01DR = 0x01;      // set the count value (upper 8 bits)
      T00DR = 0xFF;      // set the count value (lower 8 bits)
      TMCR0 = 0x10;      // 16-bit, no filtering
      T00CR0 = 0x82;     // interval timer in free run mode
                         // enable IF flag interrupt
      T00CR1 = 0xA1;     // enable the interrupt, enable output
                         // start the timer
}

/* enter ISR while counter value matchesthe pre-set value */
__interrupt void CompTimer (void)
{
      T00CR1_IE = 0;     // disable the interrupt
      T00CR1_IF = 0;     // clear the flag

      ...                // interrupt service routine

      T00CR1_IE = 1;     // enable the interrupt
}

/* main routine */
void main (void)
{
      ...
      InitCompTimer();

      ...
      while (1);
}
```

注意：对应的中断向量和级别将在 Cypress 标准模板工程的 vector.c 模块中定义。

```
/* interrupt level setting */
void InitIrqLevels (void)
{
        ...

        ILR1 = 0xF3;          //IRQ5: 8/16-bit timer ch.0 (lower)
        ...
}

__interrupt void CompTimer (void);   //Prototype

        ...

#pragma intvect CompTimer  5   //8/16-bit timer ch.0 (lower)

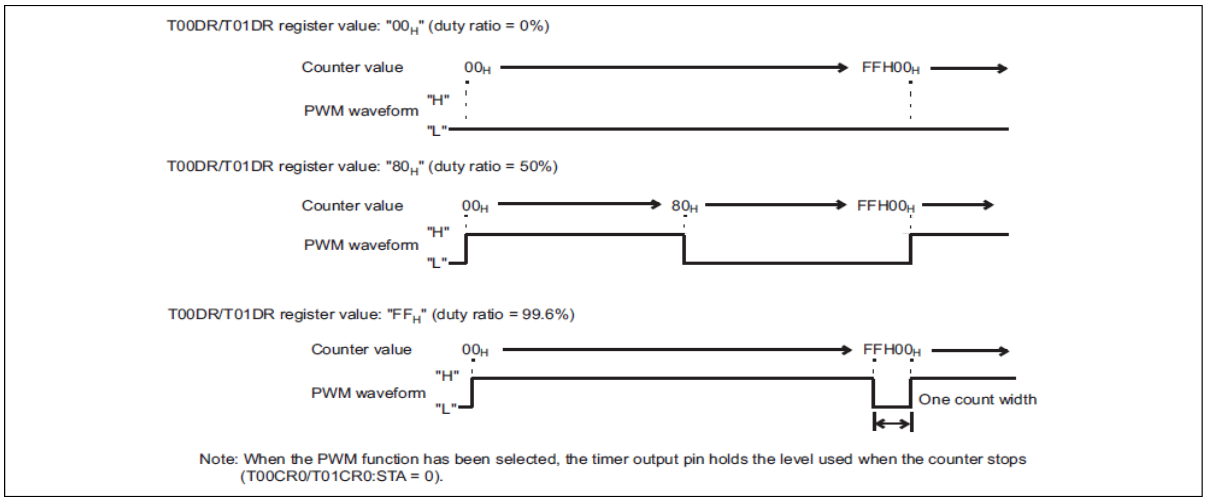
        ...
```

参见附录代码范例中的 Free-run 工程。

3.4 PWM 定时器功能的操作（固定周期模式）

如果选择 PWM 定时器功能（固定周期模式），在固定周期内将产生一个带有变量“H”脉冲宽度的 PWM 信号。8 位操作的周期固定为“FFH”；16 位操作的周期固定为“FFFFH”。该时间取决于所选的计数时钟。通过设置相应寄存器可指定“H”脉冲宽度。

图 9. PWM 定时器功能的操作图（固定周期模式）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器在 PWM 定时器功能下的操作（固定周期模式）。

1. 设置模式（T00CR0： [F3..F0] = 0 0 1 1）
2. 设置计数时钟（T00CR0： [C2..C0]）
3. 通过设置 T00DR，设置占空比。
4. 启动定时器（T00CR1： STA = 1）

以下代码显示了如何设置 8/16 位多功能定时器 0 在 PWM 定时器功能中的操作（固定周期模式）。

```
/* initial timer for the PWM function (fixed cycle) */
void InitCompTimer (void)
{
       T00DR = 0x40;        // set the PWM timer duty
       TMCR0 = 0x00;        // 8-bit, no filtering

       T00CR0 = 0x03;       // PWM timer function (fixed cycle mode)
       T00CR1 = 0x01;       // enable the output
}
       ...

/* main routine */
void main (void)
{
       ...

       InitCompTimer ();

       ...

       T00CR1_STA = 1;      //start the timer

       While(1);
}
```

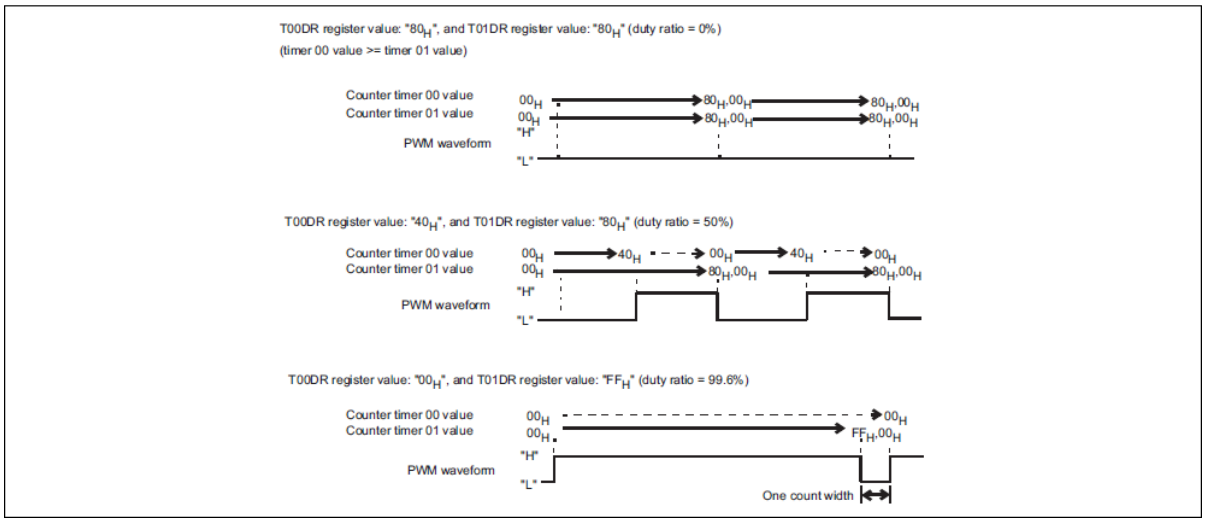
参见附录代码范例中的 PWM_Fixed 工程。

3.5 PWM 定时器功能的操作（可变周期模式）

如果选择 PWM 定时器功能（可变周期模式），两个 8 位计数器用于在任何周期内产生一个 8 位 PWM 信号，并基于对应寄存器指定的“L”和脉冲宽度产生占空比。

在此操作模式下，因为使用了两个 8 位计数器，该多功能定时器不能形成 16 位计数器。

图 10. PWM 定时器功能的操作图（可变周期模式）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器在 PWM 定时器功能下的操作（可变周期模式）。

1. 设置模式（T00CR0： [F3..F0] = 0 1 0 0）
2. 设置计数器时钟（T00CR0： [C2..C0]）
3. 设置计数器时钟（T01CR0： [C2..C0]）
4. 通过设置 T00DR，设置占空比。
5. 通过设置 T01DR，设置周期。
6. 启用定时器输出（T00CR1： OE = 1）
7. 启动定时器（T00CR1： STA = 1）

以下代码显示了如何设置 8/16 位多功能定时器在 PWM 定时器功能中的操作（可变周期模式）。

```
/* initial timer for PWM function (variable cycle) */
void InitCompTimer (void)
{
       T01DR = 0x80;        // counter timer 01 value for duty
       T00DR = 0x40;        // counter timer 00 value for cycle
                            // duty ratio 50%
       TMCR0 = 0x00;        // 8-bit, no filtering

       T00CR0 = 0x04;       // PWM timer function (variable-cycle mode)
       T01CR0 = 0x04;       // choose the same count clock as Timer 00
       T00CR1 = 0x01;       // enable the output
}
       ...

/* main routine */
void main (void)
{
       ...
       InitCompTimer ();
       ...

       T00CR1_STA = 1;  //start the timer

       while(1);
}
```

参见附录代码范例中的 PWM_Vari 工程。

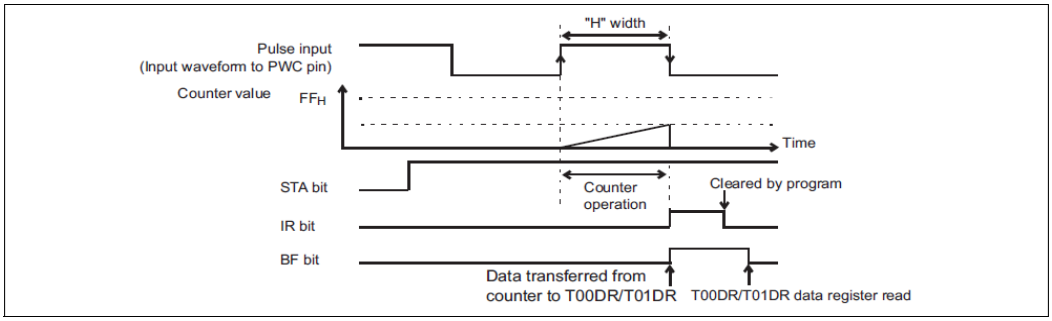
注意：使用该功能时，定时器 00 和定时器 01 将使用同一个计数器时钟。

3.6 PWC 定时器功能的操作

如果选择 PWC 定时器功能，外部输入脉冲的宽度和周期将被测定。

在此操作模式下，检测到内部输入信号的计数开始沿后，计数器从“00H”开始计数，并把计数值发送给寄存器，检测到计数结束沿后，产生一个中断。

图 11. PWC 定时器的操作图（H 脉冲宽度测定）



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器在 PWC 定时器功能下的操作（“H”脉冲）。

1. 设置定时器的输入（DDR0_ P04 = 0; AIDRL_ P04 = 1）
2. 选择输入端口（SYSC_EC0SL = 1）
3. 设置模式（T00CR0： [F3..F0] = 0 1 0 1）
4. 设置计数器时钟（T00CR0： [C2..C0]）
5. 如有必要，设置中断级别（ILR10)
6. 启用定时器输出（T00CR1： OE = 1）
7. 启动定时器（T00CR1： STA = 1）
8. 测定完成后，读取该值。

如果启用中断，ISR 中的标志将被清除。（T00CR1： IR = 0）。

PWC 定时器有以下五种测定模式。

表 1. T00CR0/T01CR_F3..F0 和 PWC 定时器功能的关系

T00CR0/T01CR0 [F3..F0]	说明
0 1 0 1	“H”脉冲 = 上升到下降
0 1 1 0	“L”脉冲 = 下降到上升
0 1 1 1	循环 = 上升到上升
1 0 0 0	循环 = 下降到下降
1 0 0 1	“H”脉冲 = 上升到下降；循环 = 上升到上升

以下代码显示了如何设置 8/16 位多功能定时器 0 在 PWC 定时器功能中的操作。

```
/* initial timer for the PWC function */
void InitCompTimer (void)
{
       SYSC_EC0SL = 1;      //P04 as EC0
       DDR0_P04 = 0;        //P04 input
       AIDRL |= 0x10;       //disable AN04 input
       TMCR0 = 0x00;        // 8-bit, no filtering
       T00CR0 = 0x05;       // PWC timer ("H" pulse = from rising to falling)
       T00CR1 = 0xA0;       // enable an interrupt, disable the output
                            // start the timer
}


/* enter ISR while measurement completes or overflows */
__interrupt void CompTimer (void)
{
       T00CR1_IE = 0;       // disable an interrupt

       if(T00CR1_BF)        // measurement data present in data register ?
       {
              ucHCycle = T00DR;    // read value
       }
       T00CR1_IR = 0;              // clear the flag
       T00CR1_IE = 1;              // enable the interrupt
}

/* main routine */
void main (void)
{
       ...

       InitCompTimer ();

       ...                  // input signal to be measured

       while(1);
}
```

注意：对应的中断向量和级别将在 Cypress 标准模板工程的 vector.c 模块中定义。

```
/* interrupt level setting */
void InitIrqLevels (void)
{
       ...

       ILR1 = 0xF3;         //IRQ5: 8/16-bit timer ch.0 (lower)
       ...
}

__interrupt void CompTimer (void);   //Prototype
       ...

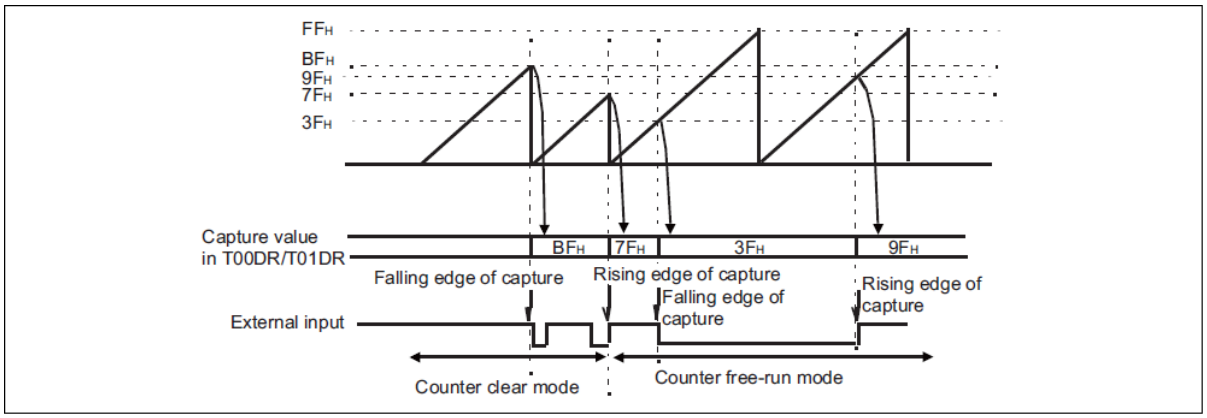
#pragma intvect CompTimer  5         //8/16-bit timer ch.0 (lower)
       ...
```

参见附录代码范例中的 PWC 工程。

3.7 输入捕捉功能的运算

如果选择输入捕捉功能，在检测到外部输入信号的一个沿后，计数值被保存在寄存器中。

图 12. 输入捕捉功能的操作图



设置程序示例

以下程序显示了如何设置 8/16 位多功能定时器在输入捕捉功能中的操作（上升，计数器清零)。

1. 设置定时器的输入（DDR0_ P04 = 0; AIDRL_ P04 = 1）
2. 选择输入端口（SYSC_EC0SL = 1）
3. 设置模式（T00CR0 ： [F3..F0] = 1 1 0 1）
4. 设置计数器时钟（T00CR0 ： [C2..C0]）
5. 如有必要，设置中断级别（ILR1）
6. 启用定时器输出（T00CR1 ： OE = 1）
7. 启动定时器（T00CR1 ： STA = 1）
8. 完成测定后，读取该值。

如果启用中断，ISR 中的标志将被清除。（T00CR1 ： IR = 0）

输入捕捉有以下六种捕捉模式。

表 2. T00CR0/T01CR_F3..F0 和输入捕捉功能的关系

T00CR0/T01CR0 [F3..F0]	说明
1 0 1 0	上升，自由运行计数器
1 0 1 1	下降，自由运行计数器
1 1 0 0	双沿，自由运行计数器
1 1 0 1	上升，计数器清零
1 1 1 0	下降，计数器清零
1 1 1 1	双沿，计数器清零

以下代码显示了如何设置 8/16 位多功能定时器 0 在输入捕捉功能中的操作。

```
/* initial timer for the input capture function */
void InitCompTimer (void)
{
       SYSC_EC0SL = 1;      //P04 as EC0
       DDR0_P04 = 0;        //P04 input
       AIDRL |= 0x10;       //disable AN04 input

       TMCR0 = 0x00;        // 8-bit, no filtering

       T00CR0 = 0x0D;       // Input capture (rising, counter clear)

       T00CR1 = 0xA0;       // enable the interrupt, disable the output
                            // start the timer
}


/* Enter ISR while capturing the input */
__interrupt void CompTimer (void)
{
       T00CR1_IE = 0;       // disable the interrupt

       ucHCycle = T00DR;    // read value

       T00CR1_IR = 0;       // clear the flag
       T00CR1_IE = 1;       // enable the interrupt
}

/* main routine */
void main (void)
{
       ...

       InitCompTimer ();

       ...                  // input signal to be captured

       while(1);
}
```

注意：对应的中断向量和级别将在 Cypress 标准模板工程的 vector.c 模块中定义。

```
/* interrupt level setting */
void InitIrqLevels (void)
{
       ...

       ILR1 = 0xF3;         //IRQ5: 8/16-bit timer ch.0 (lower)
       ...
}

__interrupt void CompTimer (void);   //Prototype

       ...

#pragma intvect CompTimer     5      //8/16-bit timer ch.0 (lower)

       ...
```

参见附录代码范例中的 Capture 工程。

4 使用 8/16 位多功能定时器的注意事项

本章介绍了使用 8/16 位多功能定时器的注意事项。

- 要使用定时器操作模式选择位（T00CR0/T01CR0：F3、F2、F1、F0）选择定时器功能，首先停止定时器操作（T00CR1/T01CR1：STA=0），然后清除中断标志（T00CR1/T01CR1：IF、IR），中断启用位（T00CR1/T01CR1：IE，T00CR0/T01CR0：IFE）和缓冲标志（T00CR1/T01CR1：BF）。
- 在 PWC 功能或输入捕捉功能的操作中，中断可能在定时器（STA = 0）激活以前产生。因此，应使定时器启动前的 8/16 位多功能定时器 00/01 数据寄存器（T00DR/T01DR）的值无效。

5 更多信息

如欲了解有关 Cypress 微控制器产品的更多详情，敬请访问以下网址：

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

http://www.cypress.com/cypress-mcu-product-softwareexamples

A 附录

A.1 代码范例

A.1.1 工程名称：One-shot

间隔定时器功能（单次模式）

main.c

```
#include "mb95200.h"
/*-------------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the interval timer (One-shot) function
-------------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        T00DR = 0xFF;           // set the count value
        TMCR0 = 0x00;           // 8-bit, no filtering
        T00CR0 = 0x00;      // interval timer in one-shot mode
        T00CR1 = 0x01;      // disable the interrupt, enable the output
}
/*-------------------------------------------------------------------------------
  name: main();
  function: main loop
-------------------------------------------------------------------------------*/
void main(void)
{
        InitCompTimer();
        T00CR1_STA = 1;         // start the timer
}
```

A.1.2 工程名称：Continuous

间隔定时器功能（连续模式）

main.c

```
#include "mb95200.h"
/*-------------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the interval timer (continuous) function
-------------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        T01DR = 0x01;           // set the count value (upper 8 bits)
        T00DR = 0xFF;           // set the count value (lower 8 bits)
        TMCR0 = 0x10;           // 16-bit, no filtering
        T00CR0 = 0x81;      // interval timer in continuous mode
                                // enable the IF flag interrupt
        T00CR1 = 0xA1;      // enable the interrupt, enable the output
                                // start the timer
}

/*-------------------------------------------------------------------------------
  name: CompTimer ();
  function: enter ISR while the counter value matches the pre-set value
-------------------------------------------------------------------------------*/
__interrupt void CompTimer (void)
{
        T00CR1_IE = 0;      // disable the interrupt
        T00CR1_IF = 0;      // clear the flag
        //...               // interrupt service routine
        T00CR1_IE = 1;      // enable the interrupt
}
/*-------------------------------------------------------------------------------
  name: main ();
  function: main loop
-------------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        InitIrqLevels();
        __EI();
        while (1);
}
```

vector.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
  name: InitIrqLevels ();
  function: Interrupt level (priority) setting
-----------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
        ILR1 = 0xF3;            // IRQ4:  UART/SIO ch0
                                // IRQ5:  8/16-bit timer ch.0 (lower)
                                // IRQ6:  8/16-bit timer ch.0 (upper)
                                // IRQ7:  LIN-UART (reception)
}


/*-----------------------------------------------------------------------------
  Prototypes
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void);

/*-----------------------------------------------------------------------------
  Vector definition
-----------------------------------------------------------------------------*/
#pragma intvect CompTimer 5   // IRQ5:  8/16-bit timer ch.0 (lower)
```

A.1.3 工程名称：Free-run

间隔定时器功能（自由运行模式）

main.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the interval timer (Free-run) function
-----------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        T01DR = 0x01;           // set count value (upper 8 bits)
        T00DR = 0xFF;           // set count value (lower 8 bits)
        TMCR0 = 0x10;           // 16-bit, no filtering
        T00CR0 = 0x82;      // interval timer in a free run mode
                                // enable the IF flag interrupt
        T00CR1 = 0xA1;      // enable the interrupt, enable the output
                                // start the timer
}

/*-----------------------------------------------------------------------------
  name: CompTimer ();
  function: enter ISR while the counter value matches the pre-set value
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void)
{
        T00CR1_IE = 0;      // disable the interrupt
        T00CR1_IF = 0;      // clear the flag
        //...               // interrupt service routine
        T00CR1_IE = 1;      // enable the interrupt
}
/*-----------------------------------------------------------------------------
  name: main ();
  function: main loop
-----------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        InitIrqLevels();
        __EI();
        while (1);
}
```

vector.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
  name: InitIrqLevels ();
  function: Interrupt level (priority) setting
-----------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
        ILR1 = 0xF3;        // IRQ4:  UART/SIO ch.0
                            // IRQ5:  8/16-bit timer ch.0 (lower)
                            // IRQ6:  8/16-bit timer ch.0 (upper)
                            // IRQ7:  LIN-UART (reception)
}

/*-----------------------------------------------------------------------------
  Prototypes
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void);

/*-----------------------------------------------------------------------------
  Vector definition
-----------------------------------------------------------------------------*/
#pragma intvect CompTimer 5   // IRQ5:  8/16-bit timer ch0 (lower)
```

A.1.4 工程名称：PWM_Fixed

PWM 定时器功能（固定周期模式）

main.c

```
#include “mb95200.h”
/*-------------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the PWM (fixed-cycle) function
-------------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        T00DR = 0x40;           // set the PWM timer duty
        TMCR0 = 0x00;           // 8-bit, no filtering
        T00CR0 = 0x03;      // PWM timer function (fixed cycle mode)
        T00CR1 = 0x01;      // enable the output
}
/*-------------------------------------------------------------------------------
  name: main ();
  function: main loop
-------------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        T00CR1_STA = 1;         // start the timer
        while(1);
}
```

A.1.5 工程名称：PWM_Vari

PWM 定时器功能（可变周期模式）

main .c

```
#include "mb95200.h"
/*-------------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for PWM (variable cycle) function
-------------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        T01DR = 0x80;           // set the PWM timer cycle
        T00DR = 0x40;           // set the PWM timer duty
                                // duty ratio 50%
        TMCR0 = 0x00;           // 8-bit, no filtering
        T00CR0 = 0x04;      // PWM timer function (variable cycle mode)
        T01CR0 = 0x04;      // choose the same count clock as Timer 00
        T00CR1 = 0x01;      // enable the output
}
/*-------------------------------------------------------------------------------
  name: main ();
  function: main loop
-------------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        T00CR1_STA = 1;         // start the timer
        while(1);
}
```

A.1.6 工程名称：PWC

PWC 定时器功能

main.c

```
#include "mb95200.h"
unsigned char ucHCycle = 0xFF;
/*-----------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the PWC function
-----------------------------------------------------------------------------*/
void InitCompTimer (void)
{
        SYSC_EC0SL = 1;         // P04 as EC0
        DDR0_P04 = 0;           // P04 input
        AIDRL |= 0x10;          // disable AN04 input
        TMCR0 = 0x00;           // 8-bit, no filtering
        T00CR0 = 0x05;          // PWC timer ("H" pulse = from rising to falling)
        T00CR1 = 0xA0;          // enable the interrupt, disable the output
                                // start the timer
}
/*-----------------------------------------------------------------------------
  name: CompTimer ();
  function: enter ISR while the counter value matches the pre-set value
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void)
{
        T00CR1_IE = 0;          // disable the interrupt
        if (T00CR1_BF)          // measurement data present in a data register ?
{
        ucHCycle = T00DR;       // read the value
    }
        T00CR1_IR = 0;          // clear the flag
        T00CR1_IE = 1;          // enable the interrupt
}
/*-----------------------------------------------------------------------------
  name: main ();
  function: main loop
-----------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        InitIrqLevels();
        __EI();
                                // input signal to be measured
        while (1);
}
```

vector.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
  name: InitIrqLevels ();
  function: Interrupt level (priority) setting
-----------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
        ILR1 = 0xF3;        // IRQ4:  UART/SIO ch.0
                            // IRQ5:  8/16-bit timer ch.0 (lower)
                            // IRQ6:  8/16-bit timer ch.0 (upper)
                            // IRQ7:  LIN-UART (reception)
}

/*-----------------------------------------------------------------------------
  Prototypes
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void);

/*-----------------------------------------------------------------------------
  Vector definition
-----------------------------------------------------------------------------*/
#pragma intvect CompTimer 5   // IRQ5:  8/16-bit timer ch.0 (lower)
```

A.1.7 工程名称：Capture

输入捕捉功能

main.c

```
#include "mb95200.h"
unsigned char ucHCycle = 0xFF;
/*-----------------------------------------------------------------------------
  name: InitCompTimer();
  function: initial timer for the capture input function
-----------------------------------------------------------------------------*/

void InitCompTimer (void)
{
        SYSC_EC0SL = 1;         // P04 as EC0
        DDR0_P04 = 0;       // P04 input
        AIDRL |= 0x10;      // disable AN04 input
        TMCR0 = 0x00;       // 8-bit, no filtering
        T00CR0 = 0x0D;      // Input capture (rising, counter clear)
        T00CR1 = 0xA0;      // enable the interrupt, disable the output
                                // start the timer
}
/*-----------------------------------------------------------------------------
  name: CompTimer ();
  function: enter ISR while capturing the input
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void)
{
        T00CR1_IE = 0;      // disable the interrupt
        ucHCycle = T00DR;   // read the value
        T00CR1_IR = 0;      // clear the flag
        T00CR1_IE = 1;      // enable the interrupt
}
/*-----------------------------------------------------------------------------
  name: main ();
  function: main loop
-----------------------------------------------------------------------------*/
void main (void)
{
        InitCompTimer();
        InitIrqLevels();
        __EI();
                                // input signal to be captured
        while (1);
}
```

vector.c

```
#include "mb95200.h"
/*-----------------------------------------------------------------------------
  name: InitIrqLevels ();
  function: Interrupt level (priority) setting
-----------------------------------------------------------------------------*/
void InitIrqLevels(void)
{
        ILR1 = 0xF3;        //  IRQ4:  UART/SIO ch0
                            //  IRQ5:  8/16-bit timer ch0 (lower)
                            //  IRQ6:  8/16-bit timer ch0 (upper)
                            //  IRQ7:  LIN-UART (reception)
}

/*-----------------------------------------------------------------------------
  Prototypes
-----------------------------------------------------------------------------*/
__interrupt void CompTimer (void);

/*-----------------------------------------------------------------------------
  Vector definition
-----------------------------------------------------------------------------*/
#pragma intvect CompTimer 5   //  IRQ5:  8/16-bit timer ch0 (lower)
```

文档修改记录

文档标题：AN205275 - F²MC-8FX 家族 MB95200H/210H 系列 8/16 位多功能定时器

文档编号：002-05739

修订版	ECN	变更者	提交日期	变更说明
**	—	HUAL	03/20/2008	初稿
			07/18/2008	在第六章“更多信息”中增加 URL；修改部分代码示例。
*A	5327469	HUAL	06/30/2016	已将 Spansion 应用手册《MCU-AN-500004-Z-11》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，2008-2016 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。