第 16 章 输出比较

目录

本章包括下列主题：

16.1 简介 16-2
16.2 输出比较寄存器 16-3
16.3 工作原理 16-34
16.4 中断 16-61
16.5 I/O 引脚控制 16-62
16.6 节能和调试模式下的操作 16-63
16.7 各种复位的影响 16-64
16.8 输出比较应用 16-64
16.9 设计技巧 16-67
16.10 相关应用笔记 16-68
16.11 版本历史 16-69

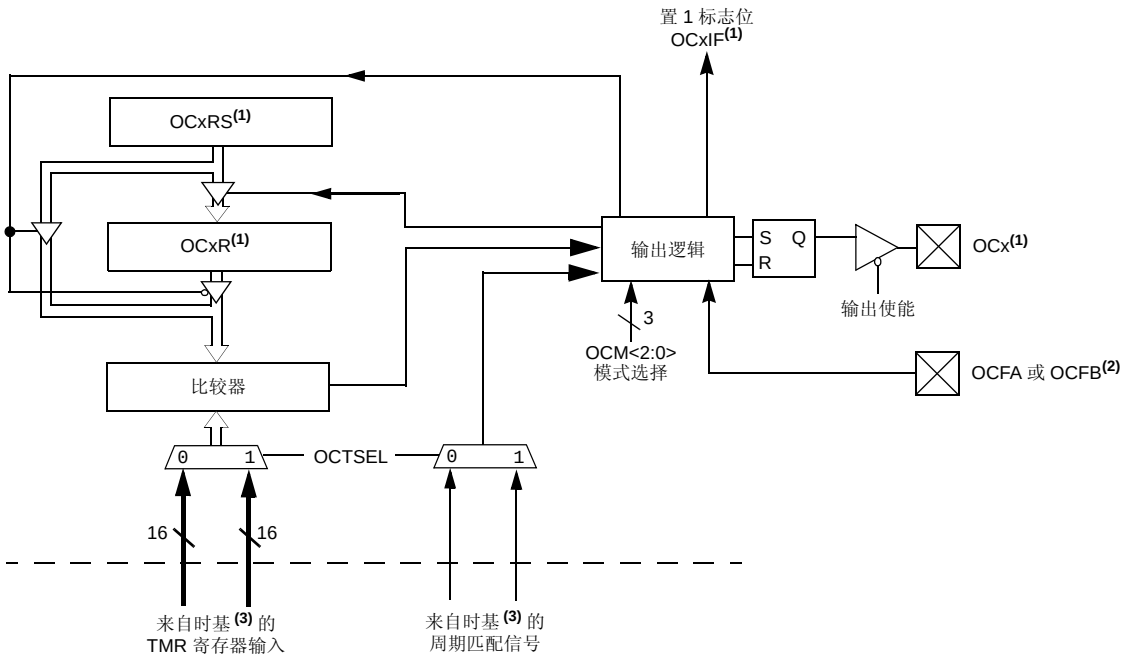
16.1 简介

输出比较模块主要用于在响应选定时基事件时产生单脉冲信号或一连串脉冲信号。

下面列出了输出比较模块的部分主要特性：

- 一个器件中可以有多个输出比较模块
- 单比较模式和双比较模式
- 产生单脉冲和连续脉冲输出
- 脉宽调制 （Pulse-Width Modulation， PWM）模式
- 在发生比较事件时产生可编程中断
- 基于硬件的 PWM 故障检测和自动输出禁止
- 可通过编程选择 16 位或 32 位时基
- 可通过两个可用 16 位时基中的任意一个工作，也可通过一个 32 位时基工作

图 16-1： 输出比较模块框图



注 1： 显示 “x” 之处与各自输出比较通道 1 到 5 相关的寄存器对应。

2： OCFA 引脚控制 OC1-OC4 通道。 OCFB 引脚控制 OC5 通道。

3： 每个输出比较通道可使用两个可选 16 位时基中的任意一个，或使用一个 32 位时基。关于与该模块相关的时基，请参见器件数据手册。

16.2 输出比较寄存器

> **注：** 每个不同的 PIC32MX 器件型号可能具有一个或多个输出比较模块。在引脚、控制 / 状态位和寄存器的名称中使用的 “x” 表示特定的模块。更多详细信息，请参见具体器件数据手册。

每个输出比较模块都包含以下特殊功能寄存器 （Special Function Register， SFR）：

- OCxCON：OCMP 模块 “x” 的控制寄存器
  OCxCONCLR、 OCxCONSET 和 OCxCONINV：OCxCON 的原子级位操作只写寄存器
- OCxR：模块 “x” 的数据寄存器
  OCxRCLR、 OCxRSET 和 OCxRINV：OCxR 的原子级位操作只写寄存器
- OCxRS：模块 “x” 的辅助数据寄存器
  OCxRSCLR、 OCxRSSET 和 OCxRSINV：OCxRS 的原子级位操作只写寄存器
- T2CON：时基寄存器
  T2CONCLR、 T2CONSET 和 T2CONINV：T2CON 的原子级位操作只写寄存器
- T3CON：时基寄存器
  T3CONCLR、 T3CONSET 和 T3CONINV：T3CON 的原子级位操作只写寄存器
- TMR2：定时器寄存器
  TMR2CLR、 TMR2SET 和 TMR2INV：TMR2 的原子级位操作只写寄存器
- TMR3：定时器寄存器
  TMR3CLR、 TMR3SET 和 TMR3INV：TMR3 的原子级位操作只写寄存器
- PR2：周期 2 寄存器
  PR2CLR、 PR2SET 和 PR2INV：PR2 的原子级位操作只写寄存器
- PR3：周期 3 寄存器
  PR3CLR、 PR3SET 和 PR3INV：PR3 的原子级位操作只写寄存器

每个定时器模块也都具有以下用于中断控制的相关位：

- OC5IF、 OC4IF、 OC3IF、 OC2IF 和 OC1IF：中断标志状态位 （在 IFS0 INT 寄存器中）
- OC5IE、 OC4IE、 OC3IE、 OC2IE 和 OC1IE：中断允许控制位 （在 IEC0 INT 寄存器中）
- OC1IP<2:0>：中断优先级控制位 （在 IPC1 INT 寄存器中）
- OC1IS<1:0>：中断子优先级控制位 （在 IPC1 INT 寄存器中）
- OC2IP<2:0>：中断优先级控制位 （在 IPC2 INT 寄存器中）
- OC2IS<1:0>：中断子优先级控制位 （在 IPC2 INT 寄存器中）
- OC3IP<2:0>：中断优先级控制位 （在 IPC3 INT 寄存器中）
- OC3IS<1:0>：中断子优先级控制位 （在 IPC3 INT 寄存器中）
- OC4IP<2:0>：中断优先级控制位 （在 IPC4 INT 寄存器中）
- OC4IS<1:0>：中断子优先级控制位 （在 IPC4 INT 寄存器中）
- OC5IP<2:0>：中断优先级控制位 （在 IPC5 INT 寄存器中）
- OC5IS<1:0>：中断子优先级控制位 （在 IPC5 INT 寄存器中）

下表汇总了所有与输出比较相关的寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。

表 16-1： 输出比较 SFR 汇总

名称		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
OCxCON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	FRZ	SIDL	—	—	—	—	—
	7:0	—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
OCxCONCLR	31:0	写入时会将 OCxCON 中的选定位清零，读取时获得的值未定义							
OCxCONSET	31:0	写入时会将 OCxCON 中的选定位置 1，读取时获得的值未定义							
OCxCONINV	31:0	写入时会将 OCxCON 中的选定位取反，读取时获得的值未定义							
OCxR	31:24	OCxR<31:24>							
	23:16	OCxR<23:16>							
	15:8	OCxR<15:8>							
	7:0	OCxR<7:0>							
OCxRCLR	31:0	写入时会将 OCxR 中的选定位清零，读取时获得的值未定义							
OCxRSET	31:0	写入时会将 OCxR 中的选定位置 1，读取时获得的值未定义							
OCxRINV	31:0	写入时会将 OCxR 中的选定位取反，读取时获得的值未定义							
OCxRS	31:24	OCxRS<31:24>							
	23:16	OCxRS<23:16>							
	15:8	OCxRS<15:8>							
	7:0	OCxRS<7:0>							
OCxRSCLR	31:0	写入时会将 OCxRS 中的选定位清零，读取时获得的值未定义							
OCxRSSET	31:0	写入时会将 OCxRS 中的选定位置 1，读取时获得的值未定义							
OCxRSINV	31:0	写入时会将 OCxRS 中的选定位取反，读取时获得的值未定义							
IFS0	31:24	I2C1MIF	I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
	23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
	15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC3IF	T2IF
	7:0	INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
IFS0CLR	31:0	写入时会将 IFS0 中的选定位清零，读取时获得的值未定义							
IFS0SET	31:0	写入时会将 IFS0 中的选定位置 1，读取时获得的值未定义							
IFS0INV	31:0	写入时会将 IFS0 中的选定位取反，读取时获得的值未定义							
IEC0	31:24	I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
	23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
	15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
	7:0	INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
IEC0CLR	31:0	写入时会将 IEC0 中的选定位清零，读取时获得的值未定义							
IEC0SET	31:0	写入时会将 IEC0 中的选定位置 1，读取时获得的值未定义							
IEC0INV	31:0	写入时会将 IEC0 中的选定位取反，读取时获得的值未定义							
IPC1	31:24	—	—	—	INT1IP<2:0>			INT1IS<1:0>	
	23:16	—	—	—	OC1IP<2:0>			OC1IS<1:0>	
	15:8	—	—	—	IC1IP<2:0>			IC1IS<1:0>	
	7:0	—	—	—	T1IP<2:0>			T1IS<1:0>	
IPC1CLR	31:0	写入时会将 IPC1 中的选定位清零，读取时获得的值未定义							
IPC1SET	31:0	写入时会将 IPC1 中的选定位置 1，读取时获得的值未定义							
IPC1INV	31:0	写入时会将 IPC1 中的选定位取反，读取时获得的值未定义							

表 16-1： 输出比较 SFR 汇总 （续）

名称		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
IPC2	31:24	—	—	—	INT2IP<2:0>			INT2IS<1:0>	
	23:16	—	—	—	OC2IP<2:0>			OC2IS<1:0>	
	15:8	—	—	—	IC2IP<2:0>			IC2IS<1:0>	
	7:0	—	—	—	T2IP<2:0>			T2IS<1:0>	
IPC2CLR	31:0	写入时会将 IPC2 中的选定位清零，读取时获得的值未定义							
IPC2SET	31:0	写入时会将 IPC2 中的选定位置 1，读取时获得的值未定义							
IPC2INV	31:0	写入时会将 IPC2 中的选定位取反，读取时获得的值未定义							
IPC3	31:24	—	—	—	INT3IP<2:0>			INT3IS<1:0>	
	23:16	—	—	—	OC3IP<2:0>			OC3IS<1:0>	
	15:8	—	—	—	IC3IP<2:0>			IC3IS<1:0>	
	7:0	—	—	—	T3IP<2:0>			T3IS<1:0>	
IPC3CLR	31:0	写入时会将 IPC3 中的选定位清零，读取时获得的值未定义							
IPC3SET	31:0	写入时会将 IPC3 中的选定位置 1，读取时获得的值未定义							
IPC3INV	31:0	写入时会将 IPC3 中的选定位取反，读取时获得的值未定义							
IPC4	31:24	—	—	—	INT4IP<2:0>			INT4IS<1:0>	
	23:16	—	—	—	OC4IP<2:0>			OC4IS<1:0>	
	15:8	—	—	—	IC4IP<2:0>			IC4IS<1:0>	
	7:0	—	—	—	T4IP<2:0>			T4IS<1:0>	
IPC4CLR	31:0	写入时会将 IPC4 中的选定位清零，读取时获得的值未定义							
IPC4SET	31:0	写入时会将 IPC4 中的选定位置 1，读取时获得的值未定义							
IPC4INV	31:0	写入时会将 IPC4 中的选定位取反，读取时获得的值未定义							
IPC5	31:24	—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
	23:16	—	—	—	OC5IP<2:0>			OC5IS<1:0>	
	15:8	—	—	—	IC5IP<2:0>			IC5IS<1:0>	
	7:0	—	—	—	T5IP<2:0>			T5IS<1:0>	
IPC5CLR	31:0	写入时会将 IPC5 中的选定位清零，读取时获得的值未定义							
IPC5SET	31:0	写入时会将 IPC5 中的选定位置 1，读取时获得的值未定义							
IPC5INV	31:0	写入时会将 IPC5 中的选定位取反，读取时获得的值未定义							
T2CON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	FRZ	SIDL	—	—	—	—	—
	7:0	TGATE	TCKPS<2:0>			T32	—	TCS	—
T2CONCLR	31:0	写入时会将 T2CON 中的选定位清零，读取时获得的值未定义							
T2CONSET	31:0	写入时会将 T2CON 中的选定位置 1，读取时获得的值未定义							
T2CONINV	31:0	写入时会将 T2CON 中的选定位取反，读取时获得的值未定义							
T3CON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ON	FRZ	SIDL	—	—	—	—	—
	7:0	TGATE	TCKPS<2:0>			—	—	TCS	—
T3CONCLR	31:0	写入时会将 T3CON 中的选定位清零，读取时获得的值未定义							
T3CONSET	31:0	写入时会将 T3CON 中的选定位置 1，读取时获得的值未定义							
T3CONINV	31:0	写入时会将 T3CON 中的选定位取反，读取时获得的值未定义							

表 16-1： 输出比较 SFR 汇总 （续）

名称		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
TMR2	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	TMRx<15:8>							
	7:0	TMRx<7:0>							
TMR2CLR	31:0	写入时会将 TMRx 中的选定位清零，读取时获得的值未定义							
TMR2SET	31:0	写入时会将 TMRx 中的选定位置 1，读取时获得的值未定义							
TMR2INV	31:0	写入时会将 TMRx 中的选定位取反，读取时获得的值未定义							
TMR3	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	TMRx<15:8>							
	7:0	TMRx<7:0>							
TMR3CLR	31:0	写入时会将 TMRx 中的选定位清零，读取时获得的值未定义							
TMR3SET	31:0	写入时会将 TMRx 中的选定位置 1，读取时获得的值未定义							
TMR3INV	31:0	写入时会将 TMRx 中的选定位取反，读取时获得的值未定义							
PR2	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	PR2<15:8>							
	7:0	PR2<7:0>							
PR2CLR	31:0	写入时会将 PR2 中的选定位清零，读取时获得的值未定义							
PR2SET	31:0	写入时会将 PR2 中的选定位置 1，读取时获得的值未定义							
PR2INV	31:0	写入时会将 PR2 中的选定位取反，读取时获得的值未定义							
PR3	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	PR3<15:8>							
	7:0	PR3<7:0>							
PR3CLR	31:0	写入时会将 PR3 中的选定位清零，读取时获得的值未定义							
PR3SET	31:0	写入时会将 PR3 中的选定位置 1，读取时获得的值未定义							
PR3INV	31:0	写入时会将 PR3 中的选定位取反，读取时获得的值未定义							

寄存器 16-1：　OCxCON：输出比较 x 控制寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

r-x	r-x	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OC32	OCFLT	OCTSEL	OCM<2:0>		
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16　**保留：**写入 0；忽略读操作

bit 15　**ON：**输出比较外设使能位
1 = 使能输出比较外设。
0 = 禁止输出比较外设，不会消耗电流。允许进行 SFR 修改。该寄存器中其他位的状态不会受该位置 1 或清零影响。
注： 使用 1:1 PBCLK 分频比时，在清零模块 ON 位的指令之后，用户的软件不应立即在 SYSCLK 周期中读 / 写外设的 SFR。

bit 14　**FRZ：**调试异常模式冻结位
1 = 在 CPU 进入调试异常模式时停止工作
0 = 在 CPU 进入调试异常模式时继续工作
注： FRZ 仅在调试异常模式下可写，在正常模式下强制为 0。

bit 13　**SIDL：**IDLE （空闲）模式停止位
1 = 在 CPU 进入 IDLE （空闲）模式时停止工作
0 = 在 IDLE （空闲）模式下继续工作

bit 12-6　**保留：**写入 0；忽略读操作

bit 5　**OC32：**32 位比较模式位
1 = OCxR<31:0> 和 / 或 OCxRS<31:0> 用于与 32 位定时器源进行比较
0 = OCxR<15:0> 和 OCxRS<15:0> 用于与 16 位定时器源进行比较

bit 4　**OCFLT：**PWM 故障条件状态位 **(1)**
1 = 发生了 PWM 故障条件 （仅可用硬件清零）
0 = 未发生 PWM 故障条件
注： 仅当 OCM<2:0> = 111 时，才使用该位。

bit 3　**OCTSEL：**输出比较定时器选择位
1 = Timer3 作为该 OCMP 模块的时钟源
0 = Timer2 作为该 OCMP 模块的时钟源
关于输出比较模块可用的特定时基，请参见器件数据手册。

注　1： 在除 PWM 模式外的其他模式下读为 0。

寄存器 16-1： OCxCON：输出比较 x 控制寄存器 （续）

bit 2-0 **OCM<2:0>：**输出比较模式选择位
111 = OCx 处于 PWM 模式；故障引脚使能
110 = OCx 处于 PWM 模式；故障引脚禁止
101 = 初始化 OCx 引脚为低电平；在 OCx 引脚上产生连续输出脉冲
100 = 初始化 OCx 引脚为低电平；在 OCx 引脚上产生单输出脉冲
011 = 比较事件使 OCx 引脚电平翻转
010 = 初始化 OCx 引脚为高电平；比较事件强制 OCx 引脚为低电平
001 = 初始化 OCx 引脚为低电平；比较事件强制 OCx 引脚为高电平
000 = 输出比较外设被禁止，但会继续消耗电流

注 1： 在除 PWM 模式外的其他模式下读为 0。

寄存器 16-2： OCxCONCLR：输出比较 x 控制清零寄存器

R/W-x	
写入时会将 OCxCON 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxCON 中的选定位清零**

在一个或多个位中写入 1 会将 OCxCON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxCONCLR = 0x00008001` 时，会将 OCxCON 寄存器中的 bit 15 和 bit 0 清零。
`localValue = OCxCONCLR` 时，将获得未定义的值。

寄存器 16-3： OCxCONSET：输出比较 x 控制置 1 寄存器

R/W-x	
写入时会将 OCxCON 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxCON 中的选定位置 1**

在一个或多个位中写入 1 会将 OCxCON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxCONSET = 0x00008001` 时，会将 OCxCON 寄存器中的 bit 15 和 bit 0 置 1。
`localValue = OCxCONSET` 时，将获得未定义的值。

寄存器 16-4： OCxCONINV：输出比较 x 控制取反寄存器

R/W-x	
写入时会将 OCxCON 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxCON 中的选定位取反**

在一个或多个位中写入 1 会将 OCxCON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxCONINV = 0x00008001` 时，会将 OCxCON 寄存器中的 bit 15 和 bit 0 取反。
`localValue = OCxCONINV` 时，将获得未定义的值。

寄存器 16-5： OCxR：输出比较 x 比较寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCR<31:24>							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCR<23:16>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **OCxR<31:16>：**当 OC32 （OCxCON<5>） = 1 时，为 32 位比较值的高 16 位

bit 15-0 **OCxR<15:0>：**为 32 位比较值的低 16 位，或者在 OC32 = 0 时，为 16 位比较值的全部 16 位

寄存器 16-6： OCxRCLR：输出比较 x 比较清零寄存器

R/W-x	
写入时会将 OCxR 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxR 中的选定位清零**

在一个或多个位中写入 1 会将 OCxR 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRCLR = 0x00008001` 时，会将 OCxR 寄存器中的 bit 15 和 bit 0 清零。
`localValue = OCxRCLR` 时，将获得未定义的值。

寄存器 16-7： OCxRSET：输出比较 x 比较置 1 寄存器

R/W-x	
写入时会将 OCxR 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxR 中的选定位置 1**

在一个或多个位中写入 1 会将 OCxR 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRSET = 0x00008001` 时，会将 OCxR 寄存器中的 bit 15 和 bit 0 置 1。
`localValue = OCxRSET` 时，将获得未定义的值。

寄存器 16-8： OCxRINV：输出比较 x 比较取反寄存器

R/W-x	
写入时会将 OCxR 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxR 中的选定位取反**

在一个或多个位中写入 1 会将 OCxR 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRINV = 0x00008001` 时，会将 OCxR 寄存器中的 bit 15 和 bit 0 取反。
`localValue = OCxRINV` 时，将获得未定义的值。

寄存器 16-9： OCxRS：输出比较 x 辅助比较寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCRS<31:24>							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCRS<23:16>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCRS<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OCRS<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **OCxRS<31:16>：**当 OC32 （OCxCON<5>） = 1 时，为 32 位比较值的高 16 位

bit 15-0 **OCxRS<15:0>：**为 32 位比较值的低 16 位，或者在 OC32 = 0 时，为 16 位比较值的全部 16 位

寄存器 16-10： OCxRSCLR：输出比较 x 辅助比较清零寄存器

R/W-x	
写入时会将 OCxRS 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxRS 中的选定位清零**

在一个或多个位中写入 1 会将 OCxRS 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRSCLR = 0x00008001` 时，会将 OCxRS 寄存器中的 bit 15 和 bit 0 清零。
`localValue = OCxRSCLR` 时，将获得未定义的值。

寄存器 16-11： OCxRSSET：输出比较 x 辅助比较置 1 寄存器

R/W-x	
写入时会将 OCxRS 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxRS 中的选定位置 1**

在一个或多个位中写入 1 会将 OCxRS 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRSSET = 0x00008001` 时，会将 OCxRS 寄存器中的 bit 15 和 bit 0 置 1。
`localValue = OCxRSSET` 时，将获得未定义的值。

寄存器 16-12： OCxRSINV：输出比较 x 辅助比较取反寄存器

R/W-x	
写入时会将 OCxRS 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 OCxRS 中的选定位取反**

在一个或多个位中写入 1 会将 OCxRS 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`OCxRSINV = 0x00008001` 时，会将 OCxRS 寄存器中的 bit 15 和 bit 0 取反。
`localValue = OCxRSINV` 时，将获得未定义的值。

寄存器 16-13： IFS0：中断标志状态寄存器 0(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF	SPI1EIF
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位，读为 0		-n = POR 时的值：（0， 1， x = 未知）	

bit 22 **OC5IF**：输出比较 5 中断请求标志位
1 = 产生了中断请求
0 = 未产生中断请求

bit 18 **OC4IF**：输出比较 4 中断请求标志位
1 = 产生了中断请求
0 = 未产生中断请求

bit 14 **OC3IF**：输出比较 3 中断请求标志位
1 = 产生了中断请求
0 = 未产生中断请求

bit 10 **OC2IF**：输出比较 2 中断请求标志位
1 = 产生了中断请求
0 = 未产生中断请求

bit 6 **OC1IF**：输出比较 1 中断请求标志位
1 = 产生了中断请求
0 = 未产生中断请求

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-14： IEC0：中断允许控制寄存器 0(1)

I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE	SPI1EIE
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位，读为 0		-n = POR 时的值：（0， 1， x = 未知）	

bit 22 **OC5IE**：输出比较 5 中断允许位
1 = 允许中断
0 = 禁止中断

bit 18 **OC4IE**：输出比较 4 中断允许位
1 = 允许中断
0 = 禁止中断

bit 14 **OC3IE**：输出比较 3 中断允许位
1 = 允许中断
0 = 禁止中断

bit 10 **OC2IE**：输出比较 2 中断允许位
1 = 允许中断
0 = 禁止中断

bit 6 **OC1IE**：输出比较 1 中断允许位
1 = 允许中断
0 = 禁止中断

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-15： IPC1：中断优先级控制寄存器 1(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT1IP<2:0>			INT1IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC1IP<2:0>			OC1IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC1IP<2:0>			IC1IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T1IP<2:0>			T1IS<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 20-18 **OC1IP<2:0>：**输出比较 1 中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 17-16 **OC1IS<1:0>：**输出比较 1 中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-16： IPC2：中断优先级控制寄存器 2(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT2IP<2:0>			INT2IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC2IP<2:0>			OC2IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC2IP<2:0>			IC2IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T2IP<2:0>			T2IS<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 20-18 **OC2IP<2:0>：**输出比较 2 中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 17-16 **OC2IS<1:0>：**输出比较 2 中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-17： IPC3：中断优先级控制寄存器 3(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT3IP<2:0>			INT3IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC3IP<2:0>			OC3IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC3IP<2:0>			IC3IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T3IP<2:0>			T3IS<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 20-18 **OC3IP<2:0>：**输出比较 3 中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 17-16 **OC3IS<1:0>：**输出比较 3 中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-18： IPC4：中断优先级控制寄存器 4(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4IP<2:0>			INT4IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC4IP<2:0>			OC4IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC4IP<2:0>			IC4IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T4IP<2:0>			T4IS<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 20-18 **OC4IP<2:0>：**输出比较 4 中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 17-16 **OC4IS<1:0>：**输出比较 4 中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-19： IPC5：中断优先级控制寄存器 5(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC5IP<2:0>			OC5IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC5IP<2:0>			IC5IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T5IP<2:0>			T5IS<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 20-18 **OC5IP<2:0>：**输出比较 5 中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 17-16 **OC5IS<1:0>：**输出比较 5 中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

注 1： 该中断寄存器中的阴影位名称用于控制其他 PIC32MX 外设，与输出比较模块无关。

寄存器 16-20： T2CON：时基寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	r-x	R/W-0	r-x
TGATE	TCKPS<2:0>			T32	—	TCS	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 15 **ON**：TMR2 使能位
1 = 使能外设
0 = 禁止外设
注： 使用 1:1 PBCLK 分频比时，在清零模块 ON 位的指令之后，用户的软件不应立即在 SYSCLK 周期中读 / 写外设的 SFR。

bit 14 **FRZ**：调试异常模式冻结位
1 = 在 CPU 处于调试异常模式时停止工作
0 = 即使在 CPU 处于调试异常模式时也继续工作
注： FRZ 仅在调试异常模式下可写，在正常模式下强制为 0。

bit 13 **SIDL**：IDLE （空闲）模式停止位
1 = 在器件进入 IDLE （空闲）模式时停止工作
0 = 即使在 IDLE （空闲）模式下也继续工作

bit 7 **TGATE**：定时器门控时间累加使能位
当 TCS = 1 时：
该位被忽略并读为 0
当 TCS = 0 时：
1 = 使能门控时间累加
0 = 禁止门控时间累加

bit 6-4 **TCKPS<2:0>**：定时器输入时钟预分频比选择位
111 = 预分频比为 1:256
110 = 预分频比为 1:64
101 = 预分频比为 1:32
100 = 预分频比为 1:16
011 = 预分频比为 1:8
010 = 预分频比为 1:4
001 = 预分频比为 1:2
000 = 预分频比为 1:1

bit 3 **T32**：32 位定时器模式选择位
1 = TMR2 和 TMR3 组成一个 32 位定时器
0 = TMR2 和 TMR3 是独立的 16 位定时器

寄存器 16-20： T2CON：时基寄存器 （续）

bit 1 **TCS**：TMR2 时钟源选择位
1 = 来自 T2CK 引脚的外部时钟
0 = 内部外设时钟

寄存器 16-21： T2CONCLR：时基清零寄存器

R/W-x															
写入时会将 T2CON 中的选定位清零，读取时获得的值未定义															
bit 31															bit 0

bit 31-0 **将 T2CON 中的选定位清零**

在一个或多个位中写入 1 会将 T2CON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T2CONCLR = 0x00008000` 时，会将 T2CON 寄存器中的 bit 15 清零。

寄存器 16-22： T2CONSET：时基置 1 寄存器

R/W-x															
写入时会将 T2CON 中的选定位置 1，读取时获得的值未定义															
bit 31															bit 0

bit 31-0 **将 T2CON 中的选定位置 1**

在一个或多个位中写入 1 会将 T2CON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T2CONSET = 0x00008000` 时，会将 T2CON 寄存器中的 bit 15 置 1。

寄存器 16-23： T2CONINV：时基取反寄存器

R/W-x															
写入时会将 T2CON 中的选定位取反，读取时获得的值未定义															
bit 31															bit 0

bit 31-0 **将 T2CON 中的选定位取反**

在一个或多个位中写入 1 会将 T2CON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T2CONINV = 0x00008000` 时，会将 T2CON 寄存器中的 bit 15 取反。

寄存器 16-24： TMR2：定时器寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 15-0 **TMR2<15:0>：**定时器计数寄存器

16 位模式：
这些位代表完整的 16 位定时器计数。

32 位模式 （仅适用于 B 类定时器）：
Timer2 和 Timer4
这些位代表 32 位定时器计数的低半字 （16 位）。

寄存器 16-25： TMR2CLR：定时器清零寄存器

写入时会将 TMR2 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR2 中的选定位清零**

在一个或多个位中写入 1 会将 TMR2 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR2CLR = 0x00008001` 时，会将 TMR2 寄存器中的 bit 15 和 bit 0 清零。

寄存器 16-26： TMR2SET：定时器置 1 寄存器

写入时会将 TMR2 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR2 中的选定位置 1**

在一个或多个位中写入 1 会将 TMR2 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR2SET = 0x00008001` 时，会将 TMR2 寄存器中的 bit 15 和 bit 0 置 1。

寄存器 16-27： TMR2INV：定时器取反寄存器

写入时会将 TMR2 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR2 中的选定位取反**

在一个或多个位中写入 1 会将 TMR2 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR2INV = 0x00008001` 时，会将 TMR2 寄存器中的 bit 15 和 bit 0 取反。

寄存器 16-28： PR2：周期寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **PR<31:16>：**未实现

bit 15-0 **PR<15:0>：**16 位 Timer2 周期匹配值。在 Timer2 和 Timer3 配置为组成一个 32 位定时器时，提供 32 位周期匹配值的低半部分。

寄存器 16-29： PR2CLR：周期 2 清零寄存器

R/W-x	
写入时会将 PR2 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR2 中的选定位清零**

在一个或多个位中写入 1 会将 PR2 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR2CLR = 0x00008001 时，会将 PR2 寄存器中的 bit 15 和 bit 0 清零。

寄存器 16-30： PR2SET：周期 2 置 1 寄存器

R/W-x	
写入时会将 PR2 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR2 中的选定位置 1**

在一个或多个位中写入 1 会将 PR2 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR2SET = 0x00008001 时，会将 PR2 寄存器中的 bit 15 和 bit 0 置 1。

寄存器 16-31： PR2INV：周期 2 取反寄存器

R/W-x	
写入时会将 PR2 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR2 中的选定位取反**

在一个或多个位中写入 1 会将 PR2 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR2INV = 0x00008001 时，会将 PR2 寄存器中的 bit 15 和 bit 0 取反。

寄存器 16-32： T3CON：时基寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	r-0	r-x	R/W-0	r-x
TGATE	TCKPS<2:0>			—	—	TCS	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 15 **ON**：TMR3 使能位
1 = 使能外设
0 = 禁止外设
注： 使用 1:1 PBCLK 分频比时，在清零模块 ON 位的指令之后，用户的软件不应立即在 SYSCLK 周期中读 / 写外设的 SFR。

bit 14 **FRZ**：调试异常模式冻结位
1 = 在 CPU 处于调试异常模式时停止工作
0 = 即使在 CPU 处于调试异常模式时也继续工作
注： FRZ 仅在调试异常模式下可写，在正常模式下强制为 0。

bit 13 **SIDL**：IDLE （空闲）模式停止位
1 = 在器件进入 IDLE （空闲）模式时停止工作
0 = 即使在 IDLE （空闲）模式下也继续工作

bit 7 **TGATE**：定时器门控时间累加使能位
当 TCS = 1 时：
该位被忽略并读为 0
当 TCS = 0 时：
1 = 使能门控时间累加
0 = 禁止门控时间累加

bit 6-4 **TCKPS<2:0>**：定时器输入时钟预分频比选择位
111 = 预分频比为 1:256
110 = 预分频比为 1:64
101 = 预分频比为 1:32
100 = 预分频比为 1:16
011 = 预分频比为 1:8
010 = 预分频比为 1:4
001 = 预分频比为 1:2
000 = 预分频比为 1:1

bit 1 **TCS**：TMR3 时钟源选择位
1 = 来自 T3CK 引脚的外部时钟
0 = 内部外设时钟

寄存器 16-33： T3CONCLR：时基清零寄存器

R/W-x	
写入时会将 T3CON 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 T3CON 中的选定位清零**

在一个或多个位中写入 1 会将 T3CON 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T3CONCLR = 0x00008000` 时，会将 T3CON 寄存器中的 bit 15 清零。

寄存器 16-34： T3CONSET：时基置 1 寄存器

R/W-x	
写入时会将 T3CON 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 T3CON 中的选定位置 1**

在一个或多个位中写入 1 会将 T3CON 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T3CONSET = 0x00008000` 时，会将 T3CON 寄存器中的 bit 15 置 1。

寄存器 16-35： T3CONINV：时基取反寄存器

R/W-x	
写入时会将 T3CON 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 T3CON 中的选定位取反**

在一个或多个位中写入 1 会将 T3CON 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`T3CONINV = 0x00008000` 时，会将 T3CON 寄存器中的 bit 15 取反。

寄存器 16-36： TMR3：定时器寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TMR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 15-0 **TMR3<15:0>：**定时器计数寄存器

16 位模式：
这些位代表完整的 16 位定时器计数。

32 位模式 （仅适用于 B 类定时器）：
Timer3 和 Timer5
这些位代表 32 位定时器计数的高半字 （16 位）。

寄存器 16-37： TMR3CLR：定时器清零寄存器

写入时会将 TMR3 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR3 中的选定位清零**

在一个或多个位中写入 1 会将 TMR3 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR3CLR = 0x00008001` 时，会将 TMR3 寄存器中的 bit 15 和 bit 0 清零。

寄存器 16-38： TMR3SET：定时器置 1 寄存器

写入时会将 TMR3 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR3 中的选定位置 1**

在一个或多个位中写入 1 会将 TMR3 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR3SET = 0x00008001` 时，会将 TMR3 寄存器中的 bit 15 和 bit 0 置 1。

寄存器 16-39： TMR3INV：定时器取反寄存器

写入时会将 TMR3 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 TMR3 中的选定位取反**

在一个或多个位中写入 1 会将 TMR3 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：`TMR3INV = 0x00008001` 时，会将 TMR3 寄存器中的 bit 15 和 bit 0 取反。

寄存器 16-40： PR3：周期 3 寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **PR<31:16>：**未实现

bit 15-0 **PR<15:0>：**16 位 Timer3 周期匹配值。在 Timer2 和 Timer3 配置为组成一个 32 位定时器时，提供 32 位周期匹配值的高半部分。

寄存器 16-41： PR3CLR：周期 3 清零寄存器

R/W-x	
写入时会将 PR3 中的选定位清零，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR3 中的选定位清零**

在一个或多个位中写入 1 会将 PR3 寄存器中的相应位清零，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR3CLR = 0x00008001 时，会将 PR3 寄存器中的 bit 15 和 bit 0 清零。

寄存器 16-42： PR3SET：周期 3 置 1 寄存器

R/W-x	
写入时会将 PR3 中的选定位置 1，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR3 中的选定位置 1**

在一个或多个位中写入 1 会将 PR3 寄存器中的相应位置 1，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR3SET = 0x00008001 时，会将 PR3 寄存器中的 bit 15 和 bit 0 置 1。

寄存器 16-43： PR3INV：周期 3 取反寄存器

R/W-x	
写入时会将 PR3 中的选定位取反，读取时获得的值未定义	
bit 31	bit 0

bit 31-0 **将 PR3 中的选定位取反**

在一个或多个位中写入 1 会将 PR3 寄存器中的相应位取反，但不会影响未实现位或只读位。写入 0 不会影响该寄存器。

示例：PR3INV = 0x00008001 时，会将 PR3 寄存器中的 bit 15 和 bit 0 取反。

16.3 工作原理

每个输出比较模块都具有以下工作模式：

- 单比较匹配模式
  - 输出驱动为高电平
  - 输出驱动为低电平
  - 输出驱动为翻转电平
- 双比较匹配模式
  - 单输出脉冲
  - 连续输出脉冲
- 简单脉宽调制模式
  - 不带故障保护输入
  - 带故障保护输入

注： 用户在切换到新模式之前，必须关闭输出比较模块（即，清零OCM<2:0>（OCxCON<2:0>））。在模块处于工作状态时更改模式可能会产生不可预料的结果。

在本章中，对与选定定时器源相关的任何 SFR 的引用，均用 “y” 后缀表示。例如， PRy 是选定定时器源的周期寄存器，而 TyCON 是选定定时器源的定时器控制寄存器。

16.3.1 单比较匹配模式

当控制位 OCM<2:0> （OCxCON<2:0>）设置为 001、 010 或 011 时，选定的输出比较通道将配置为三种单输出比较匹配模式中的一种。同时，必须使能比较时基。

在单比较模式下，将 OCxR 寄存器中装载的值与选定的递增定时器寄存器 TMRy 中的值作比较。在发生比较匹配事件时，将发生以下事件之一：

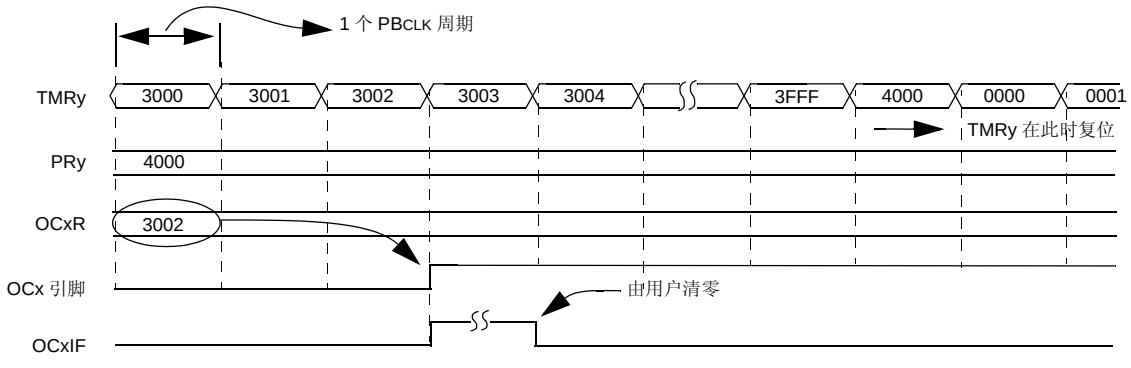
- 比较匹配事件强制OCx引脚为高电平；该引脚的初始状态为低电平。在发生单比较匹配事件时，产生中断。
- 比较匹配事件强制OCx引脚为低电平；该引脚的初始状态为高电平。在发生单比较匹配事件时，产生中断。
- 比较匹配事件使 OCx 引脚电平翻转。翻转事件是连续的，且每次翻转事件都会产生一次中断。

16.3.1.1 比较模式输出驱动为高电平

要将输出比较模块配置为该模式，需设置控制位 OCM<2:0> = 001。同时，必须使能比较时基。一旦使能了该比较模式，输出引脚 OCx 将被驱动为低电平，并保持低电平直到 TMRy 和 OCxR 寄存器之间发生匹配为止。请注意以下关键时序事件 （见图 16-2）：

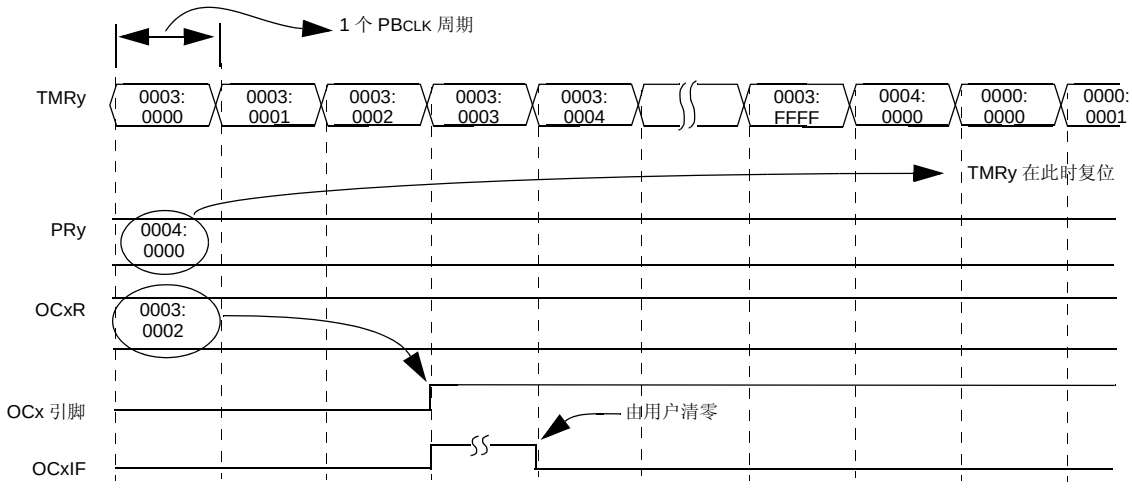
- 在比较时基和 OCxR 寄存器之间发生比较匹配后的一个外设时钟， OCx 引脚被驱动为高电平。 OCx 引脚将保持高电平直到模式发生改变或模块被禁止。
- 比较时基将计数到与关联周期寄存器中包含的值相等为止，然后在下一个 PBCLK 复位为 0x0000。
- 当 OCx 引脚被驱动为高电平时，相应的通道中断标志 OCxIF （关于每个输出比较通道的中断标志位的位置，请参见 IFS0 寄存器）会置为有效。

图 16-2： 单比较模式：在发生比较匹配事件时将 OCx 设置为高电平 （16 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-3： 单比较模式：在发生比较匹配事件时将 OCx 设置为高电平 （32 位模式）



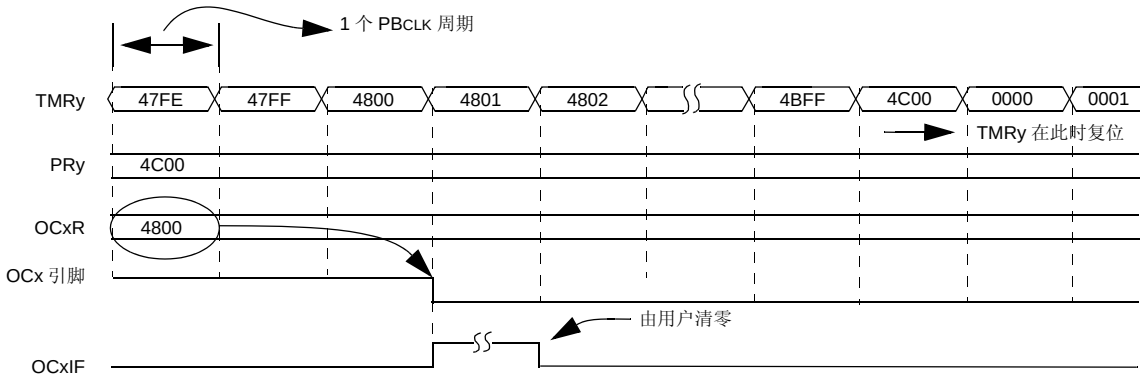
注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

16.3.1.2 比较模式输出驱动为低电平

要将输出比较模块配置为该模式，需设置控制位 OCM<2:0> = 010。同时，必须使能比较时基。一旦使能了该比较模式，输出引脚 OCx 将被驱动为高电平，并保持高电平直到定时器和 OCxR 寄存器之间发生匹配为止。请注意以下关键时序事件 （见图 16-4）：

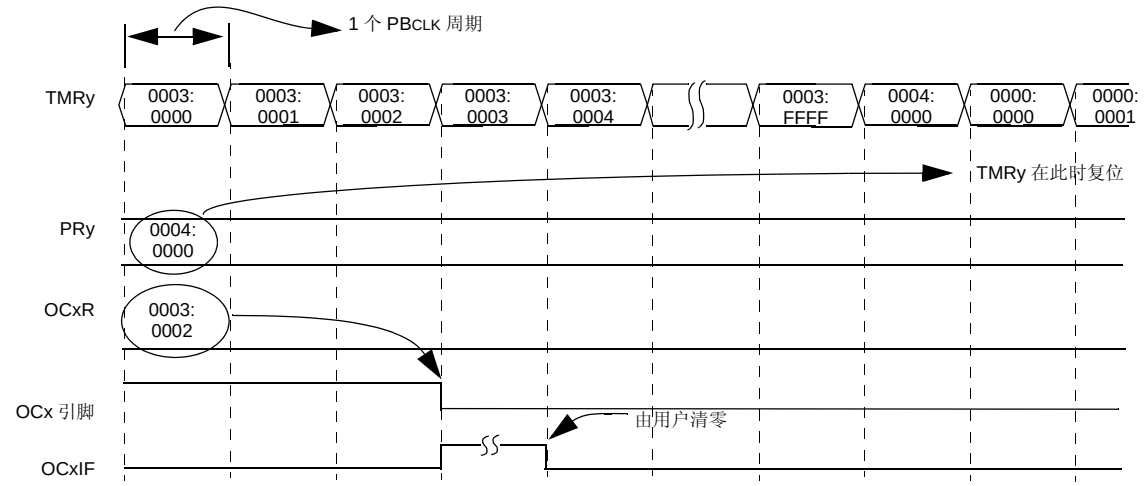
- 在比较时基和 OCxR 寄存器之间发生比较匹配后的一个 PBCLK，OCx 引脚被驱动为低电平。OCx 引脚将保持低电平直到模式发生改变或模块被禁止。
- 比较时基将计数到与关联周期寄存器中包含的值相等为止，然后在下一个 PBCLK 复位为 0x0000。
- 当 OCx 引脚被驱动为低电平时，相应通道的中断标志 OCxIF 会置为有效。

图 16-4： 单比较模式：在发生比较匹配事件时强制 OCx 为低电平 （16 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-5： 单比较模式：在发生比较匹配事件时将 OCx 设置为低电平 （32 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

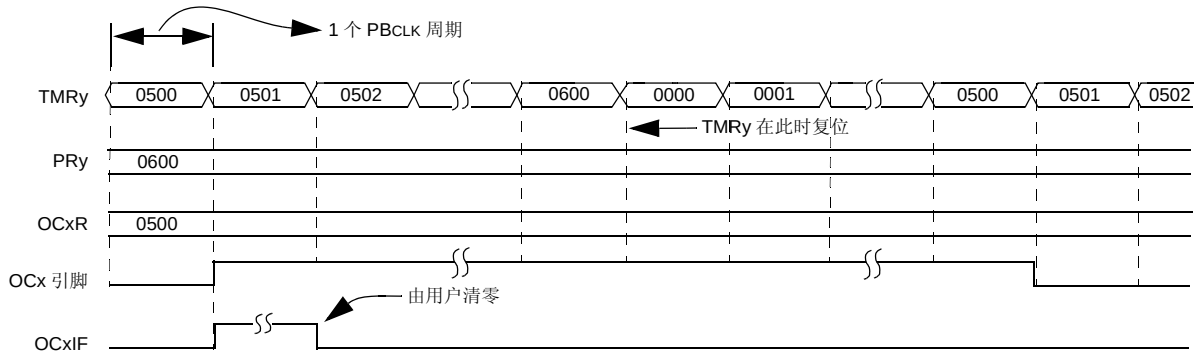
16.3.1.3 单比较模式翻转输出

要将输出比较模块配置为该模式，需设置控制位 OCM<2:0> = 011。此外，必须选择并使能 Timer2 或 Timer3。一旦使能了该比较模式，输出引脚 OCx 最初将被驱动为低电平，并在随后每当定时器和 OCxR 寄存器之间发生匹配事件时，交替输出高低电平。请注意以下关键时序事件（见图 16-6 和图 16-8）：

- 在比较时基和 OCxR 寄存器之间发生比较匹配后的一个 PBCLK 周期， OCx 引脚电平翻转。OCx 引脚将保持此新状态直到发生下一次翻转事件、模式发生改变或模块被禁止。
- 比较时基将计数到与周期寄存器中的值相等为止，然后在下一个 PBCLK 复位为 0x0000。
- 当 OCx 引脚电平翻转时，相应通道的中断标志 OCxIF 会置为有效。

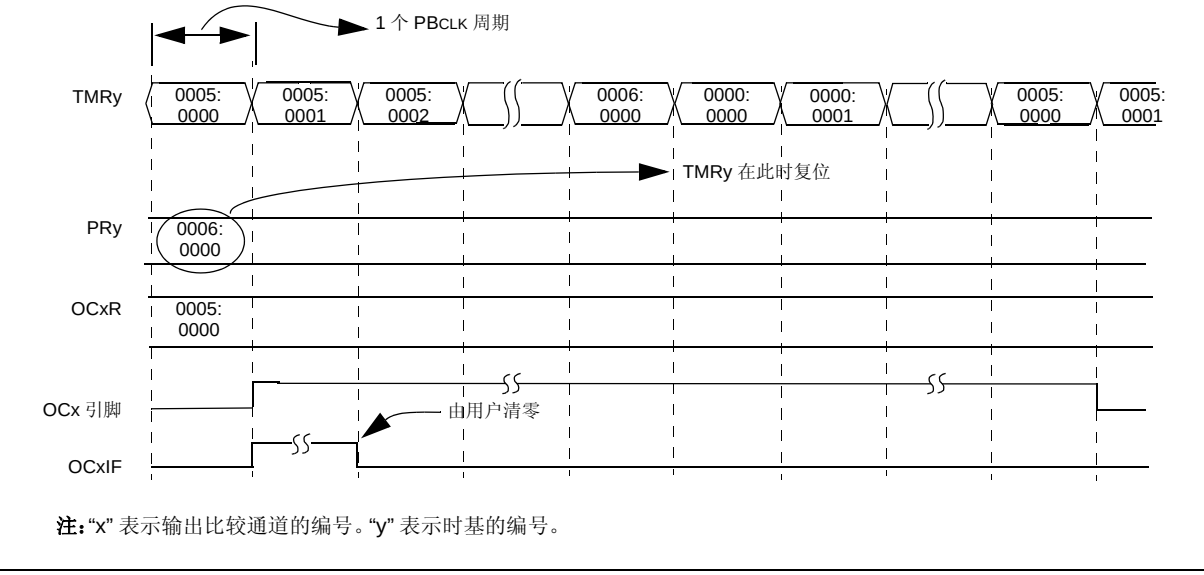
注： 器件复位时，内部 OCx 引脚输出逻辑被设置为逻辑 0。但是，在翻转模式下， OCx 引脚的工作状态可以通过用户软件设置。例 16-1 给出了在翻转工作模式下，定义所需的初始 OCx 引脚状态的代码示例。

图 16-6： 单比较模式：在发生比较匹配事件时翻转输出 （16 位模式）



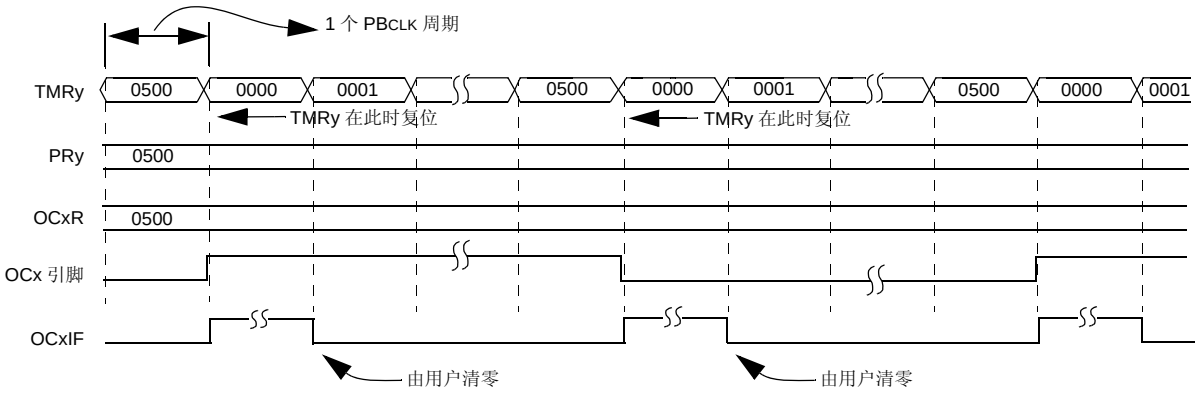
注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-7： 单比较模式：在发生比较匹配事件时翻转输出 （32 位模式）



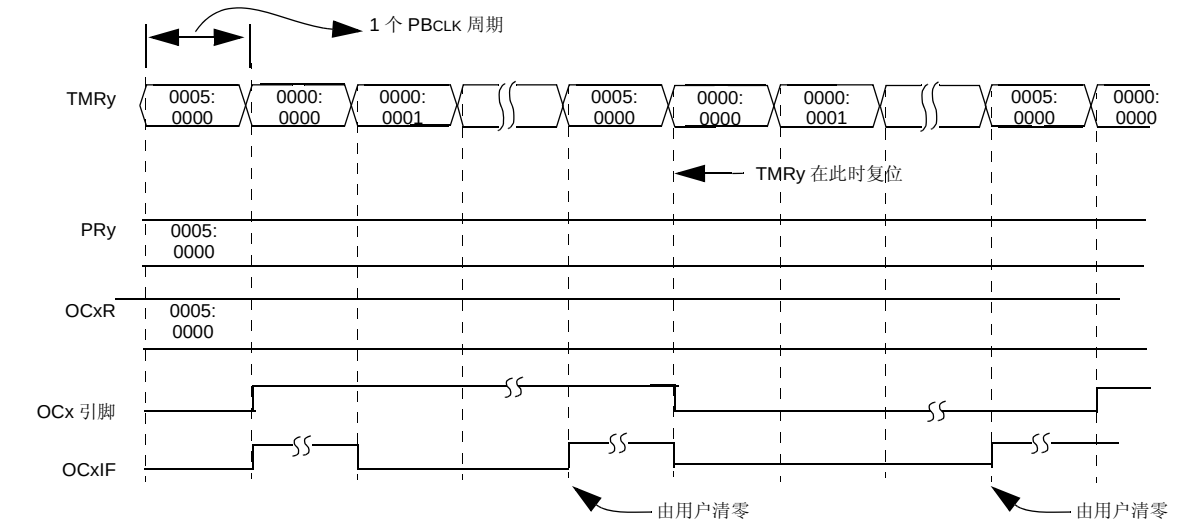
注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-8： 单比较模式：在发生比较匹配事件时翻转输出 （PRy = OCxR， 16 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-9： 单比较模式：在发生比较匹配事件时翻转输出 （PRy = OCxR， 32 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

例 16-1： 比较模式翻转模式引脚状态设置 （16 位模式）

```
// The following code example illustrates how to define the initial
// OC1 pin state for the output compare toggle mode of operation.

                                    // Toggle mode with initial OC1 pin state set low

OC1CON = 0x0001;                    // Configure module for OC1 pin low, toggle high
OC1CONSET = 0x8000;                 // Enable OC1 module
```


例 16-2： 比较模式翻转模式引脚状态设置 （32 位模式）

```
// The following code example illustrates how to define the initial
// OC1 pin state for the output compare toggle mode of operation.

                                    // Toggle mode with initial OC1 pin state set low

OC1CON = 0x0021;                    // Configure module for OC1 pin low, toggle high,
                                    // 32-bit mode
OC1CONSET = 0x8000;                 // Enable OC1 module
```

例 16-3 给出了单比较模式翻转事件的配置和中断处理的示例代码。

例 16-3： 比较模式翻转设置和中断处理 （16 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the toggle event and select Timer2 as the clock
// source for the compare time base.

T2CON = 0x0010;                     // Configure Timer2 for a prescaler of 2

OC1CON = 0x0000;                    // Turn off OC1 while doing setup.
OC1CON = 0x0003;                    // Configure for compare toggle mode
OC1R = 0x0500;                      // Initialize Compare Register 1
PR2 = 0x0500;                       // Set period

                                    // Configure int
IFS0CLR = 0x0040;                   // Clear the OC1 interrupt flag
IEC0SET = 0x040;                    // Enable OC1 interrupt
IPC1SET = 0x001C0000;               // Set OC1 interrupt priority to 7,
                                    // the highest level
IPC1SET = 0x00030000;               // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                  // Enable Timer 2
OC1CONSET = 0x8000;                 // Enable OC1

// Example code for Output Compare 1 ISR:

void __ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                   // Clear the OC1 interrupt flag
}
```

例 16-4：　比较模式翻转设置和中断处理 （32 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the toggle event and select the Timer2/Timer3 pair as
// the 32-bit as the clock source for the compare time base.

T2CON = 0x0018;                      // Configure Timer2 for 32-bit operation
                                     // with a prescaler of 2. The Timer2/Timer3
                                     // pair is accessed via registers associated
                                     // with the Timer2 register

OC1CON = 0x0000;                     // Turn off OC1 while doing setup.
OC1CON = 0x0023;                     // Configure for compare toggle mode
OC1R = 0x00500000;                   // Initialize Compare Register 1
PR2 = 0x00500000;                    // Set period (PR2 is now 32-bits wide)

                                     // configure int
IFS0CLR = 0x00000040;                // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                // Enable OC1 interrupt
IPC1SET = 0x001C0000;                // Set OC1 interrupt priority to 7,
                                     // the highest level
IPC1SET = 0x00030000;                // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                   // Enable Timer2
OC1CONSET = 0x8000;                  // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR (_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntlHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                    // Clear the OC1 interrupt flag
}
```

16.3.2 双比较匹配模式

当控制位 OCM<2:0>（OCxCON<2:0>）= 100 或 101 时，选定的输出比较通道被配置为以下两种双比较匹配模式之一：

- 单输出脉冲模式
- 连续输出脉冲模式

在双比较模式下，模块在处理比较匹配事件时使用 OCxR 和 OCxRS 这两个寄存器。将 OCxR 寄存器的值与递增定时器 TMRy 计数的值作比较，并且在发生比较匹配事件时，在 OCx 引脚上产生脉冲的前 （上升）沿。然后 OCxRS 寄存器的值与同一个递增定时器 TMRy 计数的值作比较，并且在发生比较匹配事件时，在 OCx 引脚上产生脉冲的后 （下降）沿。

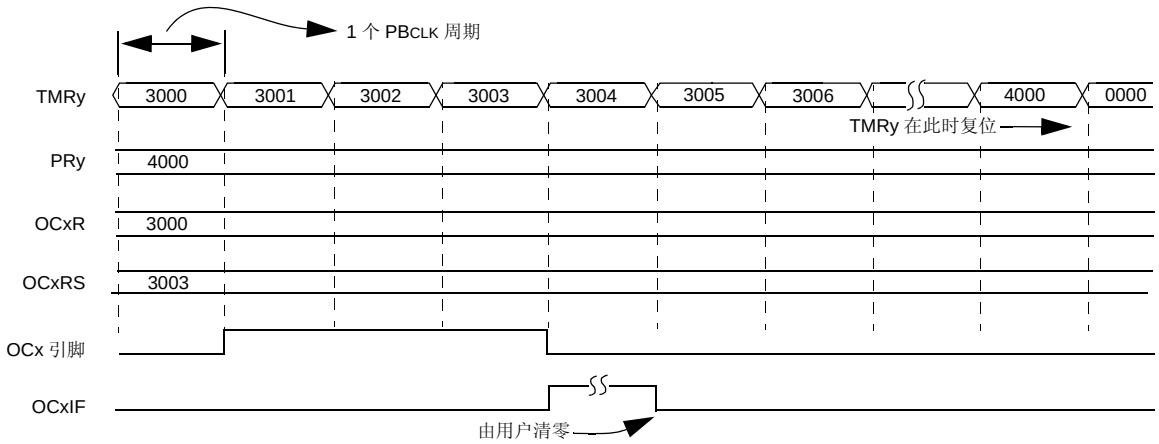
16.3.2.1 双比较模式：单输出脉冲

要将输出比较模块配置为单输出脉冲模式，需设置控制位 OCM<2:0> = 100。此外，必须选择并使能比较时基。一旦使能了该模式，输出引脚 OCx 将被驱动为低电平，并保持低电平直到时基和 OCxR 寄存器之间发生匹配为止。请注意以下关键时序事件 （见图 16-10 和图 16-12）：

- 在比较时基和 OCxR 寄存器之间发生比较匹配后的一个外设时钟， OCx 引脚被驱动为高电平。 OCx 引脚将保持高电平直到时基和 OCxRS 寄存器之间发生下一次匹配事件为止。此时，引脚将被驱动为低电平。 OCx 引脚将保持低电平直到模式发生改变或模块被禁止。
- 比较时基将计数到与关联周期寄存器中包含的值相等为止，然后在下一个指令时钟复位为 0x0000。
- 如果时基周期寄存器的内容小于 OCxRS 寄存器的内容，则不会产生脉冲的下降沿。OCx 引脚将保持高电平直到 OCxRS <= PR2、模式改变或复位条件产生为止。
- 当OCx引脚被驱动为低电平（单脉冲的下降沿）时，相应通道的中断标志OCxIF会置为有效。

图 16-10 给出了通用双比较模式产生单输出脉冲的图示。图 16-12 给出了另一个时序示例，其中 OCxRS > PR2。在该示例中，不产生脉冲的下降沿，因为比较时基在计数达到 0x4100 之前就复位了。

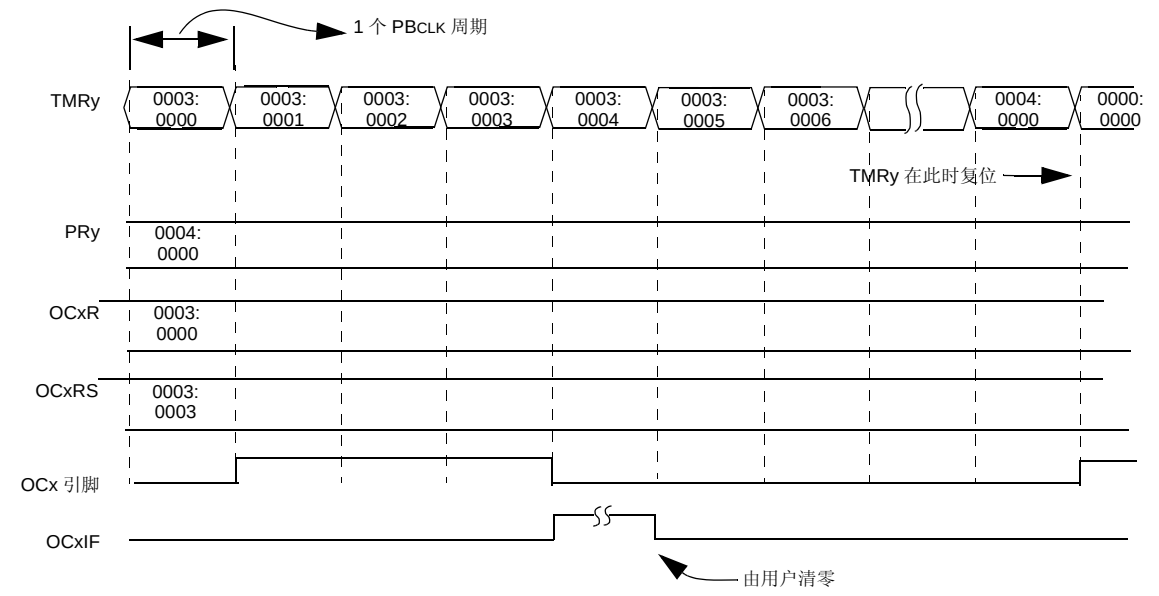
图 16-10： 双比较模式 （16 位模式）



注 1： “x” 表示输出比较通道的编号。 “y” 表示时基的编号。

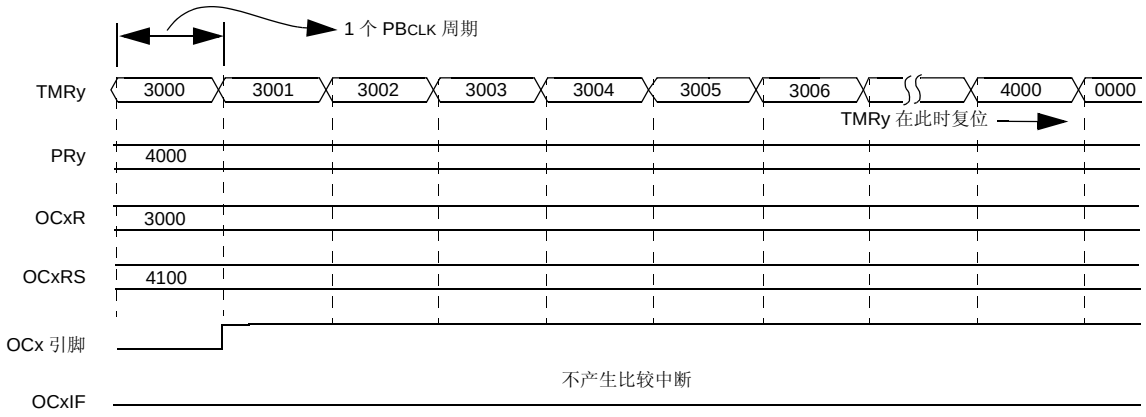
2： OCxR = 比较寄存器； OCxRS = 辅助比较寄存器。

图 16-11： 双比较模式 （32 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

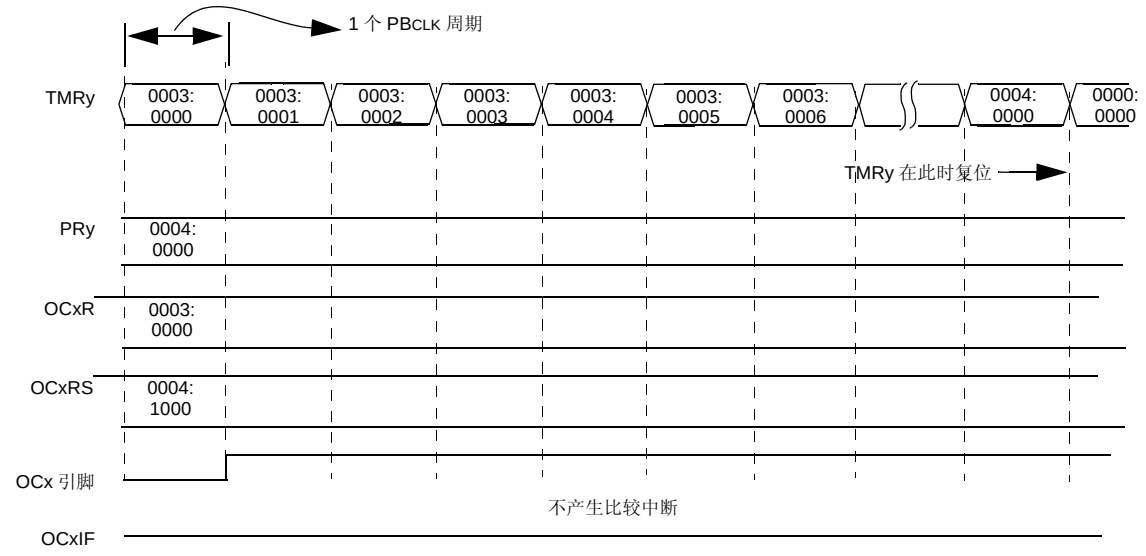
图 16-12： 双比较模式：单输出脉冲 （OCxRS > PRy， 16 位模式）



注 1： “x” 表示输出比较通道的编号。 “y” 表示时基的编号。

2： OCxR = 比较寄存器； OCxRS = 辅助比较寄存器。

图 16-13： 双比较模式：单输出脉冲 （OCxRS > PRy， 32 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

16.3.2.2 设置产生单输出脉冲

当控制位 OCM<2:0> （OCxCON<2:0>）被设置为 100 时，选定的输出比较通道将 OCx 引脚初始化为低电平状态并产生单输出脉冲。

若要产生单输出脉冲，需要遵循以下步骤（这些步骤假设定时器源在开始时是关闭的，但在模块工作时无此要求）：

1. 确定外设时钟周期时间。
2. 计算从 TMRy 起始值 （0x0000）到输出脉冲的上升沿所需的时间。
3. 根据所需的脉冲宽度和到脉冲上升沿的时间，计算出现脉冲下降沿的时间。
4. 将以上步骤2和步骤3中计算出的值分别写入比较寄存器OCxR和辅助比较寄存器OCxRS。
5. 将定时器周期寄存器 PRy 的值设置为等于或大于辅助比较寄存器 OCxRS 中的值。
6. 设置 OCM<2:0> = 100，并将 OCTSEL （OCxCON<3>）位设置为所需定时器源的对应值。此时 OCx 引脚状态被驱动为低电平。
7. 将 ON （TyCON<15>）位设为 1 以使能定时器。
8. 在 TMRy 和 OCxR 第一次匹配时， OCx 引脚将被驱动为高电平。
9. 当递增定时器 TMRy 和辅助比较寄存器 OCxRS 发生匹配时，在 OCx 引脚上驱动脉冲的第二个边沿（即后沿，从高电平到低电平）。OCx 引脚上不会驱动输出额外的脉冲，它将保持为低电平。第二次比较匹配事件会导致 OCxIF 中断标志位置 1，这将会产生中断（如果已通过将 OCxIE 位置 1 允许中断）。关于外设中断的更多信息，请参见**第 8 章 “ 中断 ”**（DS61108）。
10. 要启动另一个单脉冲输出，根据需要更改定时器和比较寄存器的设置，然后进行写操作将 OCM<2:0>（OCxCON<2:0>）位设置为 100。不需要禁止和重新使能定时器并清零 TMRy 寄存器，而且这样做有利于从已知事件时间边界定义脉冲。

在输出脉冲下降沿后不一定要禁止输出比较模块。通过重写 OCxCON 寄存器的值可以启动另一个脉冲。

例 16-5 和例 16-6 给出了单输出脉冲事件配置的示例代码。

例 16-5： 单输出脉冲设置和中断处理 （16 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the single pulse event and select Timer2
// as the clock source for the compare time base.

T2CON = 0x0010;                         // Configure Timer2 for a prescaler of 2

OC1CON = 0x0000;                        // Turn off OC1 while doing setup.
OC1CON = 0x0004;                        // Configure for single pulse mode
OC1R = 0x3000;                          // Initialize primary Compare Register
OC1RS = 0x3003;                         // Initialize secondary Compare Register
PR2 = 0x3003;                           // Set period (PR2 is now 32-bits wide)

                                        // configure int
IFS0CLR = 0x00000040;                   // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                   // Enable OC1 interrupt
IPC1SET = 0x001C0000;                   // Set OC1 interrupt priority to 7,
                                        // the highest level
IPC1SET = 0x00030000;                   // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                      // Enable Timer2
OC1CONSET = 0x8000;                     // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                       // Clear the OC1 interrupt flag
}
```

例 16-6： 单输出脉冲设置和中断处理 （32 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the single pulse event and select Timer2
// as the clock source for the compare time base.

T2CON = 0x0018;                      // Configure Timer2 for 32-bit operation
                                     // with a prescaler of 2. The Timer2/Timer3
                                     // pair is accessed via registers associated
                                     // with the Timer2 register

OC1CON = 0x0000;                     // Turn off OC1 while doing setup.
OC1CON = 0x0004;                     // Configure for single pulse mode
OC1R = 0x00203000;                   // Initialize primary Compare Register
OC1RS = 0x00203003;                  // Initialize secondary Compare Register
PR2 = 0x00500000;                    // Set period (PR2 is now 32-bits wide)

                                     // configure int
IFS0CLR = 0x00000040;                // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                // Enable OC1 interrupt
IPC1SET = 0x001C0000;                // Set OC1 interrupt priority to 7,
                                     // the highest level
IPC1SET = 0x00030000;                // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                   // Enable Timer2
OC1CONSET = 0x8000;                  // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                    // Clear the OC1 interrupt flag
}
```

16.3.2.3 双比较模式产生单输出脉冲的特殊情况

应了解根据 OCxR、OCxRS 和 PRy 值之间的关系，输出比较模块还有一些独特的条件。表 16-2 中说明了这些特殊条件及其所导致的模块行为。

表 16-2： 双比较模式产生单输出脉冲的特殊情况

SFR 逻辑关系	特殊条件	工作原理	OCx 上的输出
PRy >= OCxRS 且 OCxRS > OCxR	OCxR = 0 初始化 TMRy = 0	在第一次 TMRy 从 0x0000 计数至 PRy 时，OCx 引脚保持为低电平；不产生任何脉冲。在 TMRy 复位为零（在周期匹配时）之后，OCx 引脚在 TMRy 与 OCxR 匹配时变为高电平。在下一次 TMRy 与 OCxRS 匹配时， OCx 引脚变为低电平并保持低电平。第二次比较之后， OCxIF 位会被置 1。 有两种可选的初始条件需要考虑： a. 初始化 TMRy = 0 并设置 OCxR >= 1 b. 初始化 TMRy = PRy （PRy > 0）并设置 OCxR = 0	脉冲将根据设置进行延时，延时时间对应于 PRy 寄存器中的值
PRy >= OCxR 且 OCxR >= OCxRS	OCxR >= 1 且 PRy >= 1	TMRy 计数至 OCxR 的值，并在发生比较匹配事件（即，TMRy = OCxR）时，OCx 引脚被驱动为高电平状态。然后，TMRy 继续计数，最终在周期匹配（即， PRy = TMRy）时复位。然后，定时器从 0x0000 重新开始计数并计数至 OCxRS 的值。在发生比较匹配事件（即，TMRy = OCxRS）时，OCx 引脚被驱动为低电平状态。第二次比较之后， OCxIF 位会被置 1。	脉冲
OCxRS > PRy 且 PRy >= OCxR	无	OCx 引脚将只产生上升沿。 OCxIF 不会被置 1。	上升沿 / 跳变为高电平
OCxR > PRy	无	不受支持的模式；定时器在匹配条件之前复位。	保持为低电平

注 1： 这里考虑的所有情形，都假设 TMRy 寄存器初始化为 0x0000。

2： OCxR = 比较寄存器， OCxRS = 辅助比较寄存器， TMRy = Timery 计数， PRy = Timery 周期寄存器。

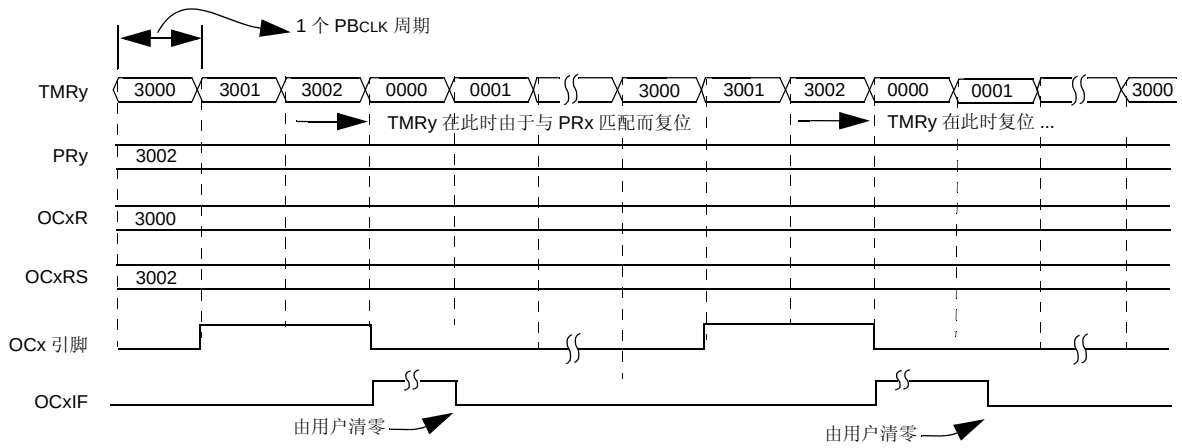
16.3.2.4 双比较模式：连续输出脉冲

要将输出比较模块配置为该模式，需设置控制位 OCM<2:0> = 101。此外，必须选择并使能比较时基。一旦使能了该模式，输出引脚OCx将被驱动为低电平，并保持低电平直到比较时基和OCxR寄存器之间发生匹配为止。请注意以下关键时序事件 （见图 16-14 和图 16-16）：

- 在比较时基和 OCxR 寄存器之间发生比较匹配后的一个 PBCLK 周期， OCx 引脚被驱动为高电平。OCx 引脚将保持高电平直到时基和 OCxRS 寄存器之间发生下一次匹配事件为止，此时引脚将被驱动为低电平。OCx 引脚将重复这种脉冲发生序列（即，从低电平变为高电平边沿，然后是从高电平变为低电平边沿），而无需用户进一步干预。
- OCx 引脚将产生连续脉冲，直到模式发生改变或模块被禁止为止。
- 比较时基将计数到与关联周期寄存器中包含的值相等为止，然后在下一个指令时钟复位为 0x0000。
- 如果比较时基周期寄存器的值小于 OCxRS 寄存器的值，则不会产生下降沿。OCx 引脚将保持高电平，直到 OCxRS <= PRy、模式发生改变或器件复位。
- 当OCx引脚被驱动为低电平（单脉冲的下降沿）时，相应通道的中断标志OCxIF会置为有效。

图 16-14 给出了通用双比较模式产生连续输出脉冲的图示。图 16-16 给出了另一个时序示例，其中 OCxRS > PRy。在该示例中，不产生脉冲的下降沿，因为时基将在计数达到 OCxRS 的内容之前复位。

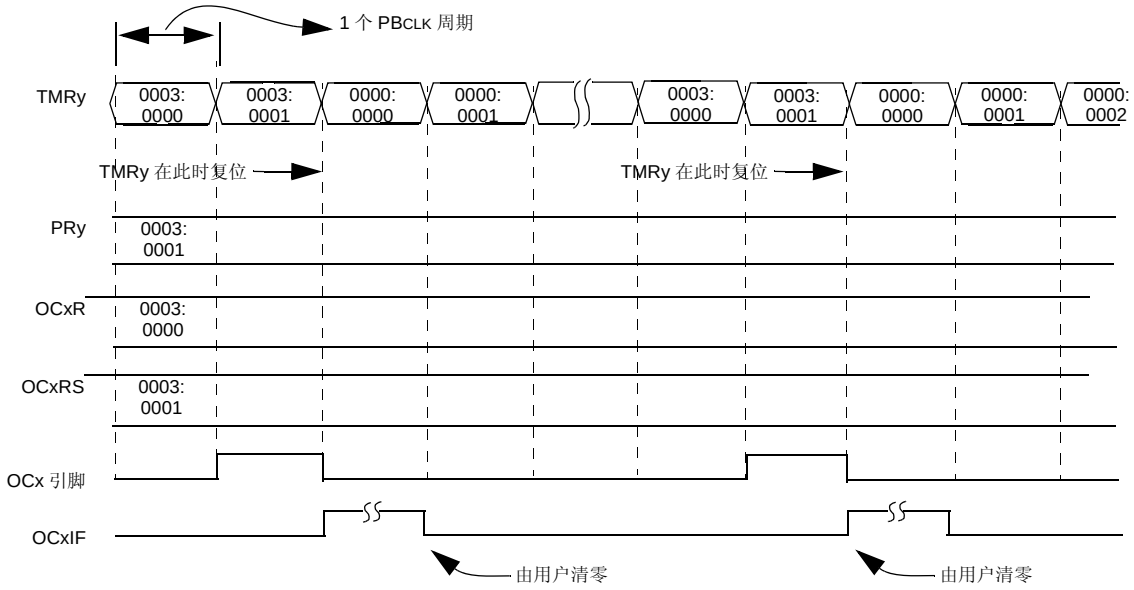
图 16-14： 双比较模式：连续输出脉冲 （PRy = OCxRS， 16 位模式）



注 1： “x” 表示输出比较通道的编号。 “y” 表示时基的编号。

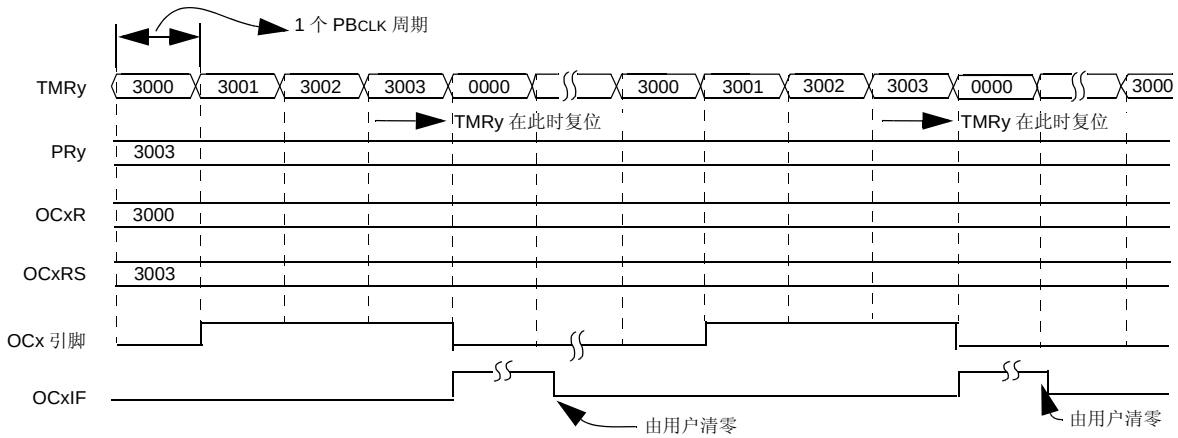
2： OCxR = 比较寄存器； OCxRS = 辅助比较寄存器。

图 16-15： 双比较模式：连续输出脉冲 （PRy = OCxRS， 32 位模式）



注：“x” 表示输出比较通道的编号。“y” 表示时基的编号。

图 16-16： 双比较模式：连续输出脉冲 （PRy = OCxRS， 16 位模式）



注 1： “x” 表示输出比较通道的编号。 “y” 表示时基的编号。

2： OCxR = 比较寄存器； OCxRS = 辅助比较寄存器。

16.3.2.5 设置产生连续输出脉冲

当控制位 OCM<2:0> （OCxCON<2:0>）被设置为 101 时，选定的输出比较通道将 OCx 引脚初始化为低电平状态，并在每次发生比较匹配事件时产生输出脉冲。

用户若要将模块配置为产生连续的输出脉冲流，需要遵循以下步骤（这些步骤假设定时器源在开始时是关闭的，但在模块工作时无此要求）：

1. 确定外设时钟周期时间。考虑定时器源（如果使用了一个定时器源）的外部时钟频率和定时器预分频比的设置。
2. 计算从 TMRy 起始值 （0x0000）到输出脉冲的上升沿所需的时间。
3. 根据所需的脉冲宽度和到脉冲上升沿的时间，计算出现脉冲下降沿的时间。
4. 将以上步骤2和步骤3中计算出的值分别写入比较寄存器OCxR和辅助比较寄存器OCxRS。
5. 将定时器周期寄存器 PRy 的值设置为等于或大于辅助比较寄存器 OCxRS 中的值。
6. 设置 OCM<2:0> = 101，并将 OCTSEL （OCxCON<3>）位设置为所需定时器源的对应值 （仅适用于 16 位模式）。此时 OCx 引脚状态被驱动为低电平。
7. 通过将 TON （TyCON<15>）位设为 1 使能比较时基。
8. 在 TMRy 和 OCxR 第一次匹配时， OCx 引脚将被驱动为高电平。
9. 当比较时基 TMRy 和辅助比较寄存器 OCxRS 发生匹配时，在 OCx 引脚上驱动脉冲的第二个边沿 （即后沿，从高电平到低电平）。
10. 第二次比较匹配事件会导致 OCxIF 中断标志位置 1。
11. 当比较时基和相应周期寄存器中的值匹配时，TMRy 寄存器复位为 0x0000 并重新开始计数。
12. 重复步骤 8 到步骤 11，可无限制地产生连续脉冲流。在每次发生 OCxRS-TMRy 比较匹配事件时， OCxIF 标志 （关于每个通道的中断标志位的位置，请参见 IF0 寄存器）会置 1。

例 16-7 给出了连续输出脉冲事件配置的示例代码。

例 16-7： 连续输出脉冲设置和中断处理 （16 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the continuous pulse event and select Timer2
// as the clock source for the compare time-base.

T2CON = 0x0010;                         // Configure Timer2 for a prescaler of 2

OC1CON = 0x0000;                        // disable OC1 module
OC1CON = 0x0005;                        // Configure OC1 module for Pulse output
OC1R = 0x3000;                          // Initialize Compare Register 1
OC1RS = 0x3003;                         // Initialize Secondary Compare Register 1
PR2 = 0x5000;                           // Set period


                                        // configure int
IFS0CLR = 0x00000040;                   // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                   // Enable OC1 interrupt
IPC1SET = 0x001C0000;                   // Set OC1 interrupt priority to 7,
                                        // the highest level
IPC1SET = 0x00030000;                   // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                      // Enable Timer2
OC1CONSET = 0x8000;                     // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR= 0x0040;                        // Clear the OC1 interrupt flag
}
```

例 16-8： 连续输出脉冲设置和中断处理 （32 位模式）

```
// The following code example will set the Output Compare 1 module
// for interrupts on the continuous pulse event and select Timer2
// as the clock source for the compare time-base.

T2CON = 0x0018;                      // Configure Timer2 for 32-bit operation
                                     // with a prescaler of 2. The Timer2/Timer3
                                     // pair is accessed via registers associated
                                     // with the Timer2 register

OC1CON = 0x0000;                     // disable OC1 module
OC1CON = 0x0005;                     // Configure OC1 module for Pulse output
OC1R = 0x3000;                       // Initialize Compare Register 1
OC1RS = 0x3003;                      // Initialize Secondary Compare Register 1
PR2 = 0x00500000;                    // Set period (PR2 is now 32-bits wide)


                                     // configure int
IFS0CLR = 0x00000040;                // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                // Enable OC1 interrupt
IPC1SET = 0x001C0000;                // Set OC1 interrupt priority to 7,
                                     // the highest level
IPC1SET = 0x00030000;                // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                   // Enable Timer2
OC1CONSET = 0x8000;                  // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                    // Clear the OC1 interrupt flag
}
```


16.3.2.6 双比较模式产生连续输出脉冲的特殊情况

根据 OCxR、OCxRS 和 PRy 值之间的关系，输出比较模块可能不会提供所期望的结果。表 16-3 中说明了这些特殊情况及其所导致的模块行为。

表 16-3： 双比较模式产生连续输出脉冲的特殊情况

SFR 逻辑关系	特殊条件	工作原理	OCx 上的输出
PRy >= OCxRS 且 OCxRS > OCxR	OCxR = 0 初始化 TMRy = 0	在第一次 TMRy 从 0x0000 计数至 PRy 时，OCx 引脚保持为低电平；不产生任何脉冲。在 TMRy 复位为零（在周期匹配时）之后， OCx 引脚变为高电平。在下一次 TMRy 与 OCxRS 匹配时，OCx 引脚变为低电平。如果 OCxR = 0 且 PRy = OCxRS，则引脚将保持一个时钟周期的低电平，然后在下一次 TMRy 与 OCxRS 匹配时被驱动为高电平。第二次比较之后， OCxIF 位会被置 1。 有两种可选的初始条件需要考虑： a. 初始化 TMRy = 0 并设置 OCxR >= 1 b. 初始化 TMRy = PRy （PRy > 0）并设置 OCxR = 0	输出连续的脉冲，第一个脉冲根据设置进行延时，延时时间对应于PRy 寄存器中的值。
PRy >= OCxR 且 OCxR >= OCxRS	OCxR >= 1 且 PRy >= 1	TMRy 计数至 OCxR 的值，并在发生比较匹配事件 （即，TMRy = OCxR）时， OCx 引脚被驱动为高电平状态。然后，TMRy 继续计数，最终在周期匹配（即，PRy = TMRy）时复位。然后，定时器从 0x0000 重新开始计数并计数至 OCxRS 的值。在发生比较匹配事件（即，TMRy = OCxR）时， OCx 引脚被驱动为低电平状态。第二次比较之后，OCxIF 位会被置 1。	连续脉冲
OCxRS > PRy 且 PRy >= OCxR	无	在 OCx 引脚将只产生一次电平跳变，直到 OCxRS 寄存器的内容更改为小于等于周期寄存器 （PRy）的内容为止。OCxIF 直到此时才会被置 1。	上升沿 / 跳变为高电平
OCxR > PRy	无	不受支持的模式；定时器在匹配条件之前复位。	保持为低电平

注 1： 这里考虑的所有情形，都假设 TMRy 寄存器初始化为 0x0000。

2： OCxR = 比较寄存器， OCxRS = 辅助比较寄存器， TMRy = Timery 计数， PRy = Timery 周期寄存器。

16.3.3 脉宽调制模式

当控制位 OCM<2:0> （OCxCON<2:0>）被设置为 110 或 111 时，选定的输出比较通道被配置为 PWM （脉宽调制）工作模式。

有以下两种 PWM 模式可供使用：

- 不带故障保护输入的 PWM
- 带故障保护输入的 PWM

第二种 PWM 模式需使用 OCFA 或 OCFB 故障输入引脚。在该模式下，OCFx 引脚上的异步逻辑电平 0 会使选定的 PWM 通道关闭。（见**第 16.3.3.1 节 “ 带故障保护输入引脚的 PWM”**。）

在 PWM 模式下， OCxR 寄存器是只读从占空比寄存器， OCxRS 是缓冲寄存器，由用户写入数据来更新 PWM 占空比。在每次发生定时器与周期寄存器的匹配事件时 （PWM 周期结束），占空比寄存器 OCxR 会装入 OCxRS 的内容。 TyIF 中断标志在每个 PWM 周期边界处置为有效。

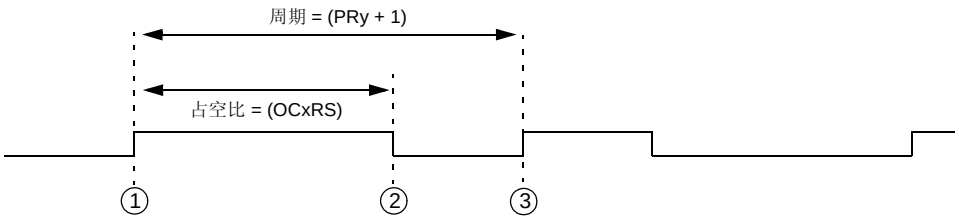
当将输出比较模块配置为 PWM 操作时，需要遵循以下步骤：

1. 通过写选定的定时器周期寄存器 （PRy），设置 PWM 周期。
2. 通过写 OCxRS 寄存器设置 PWM 占空比。
3. 向 OxCR 寄存器中写入初始占空比。
4. 如果需要，允许定时器和输出比较模块的中断。如果要使用 PWM 故障引脚，则必须设置输出比较中断。
5. 通过写输出比较模式位 OCM<2:0>（OCxCON<2:0>），将输出比较模块配置为两种 PWM 工作模式中的一种。
6. 设置 TMRy 预分频值，并通过设置 TON （TxCON<15>） = 1 使能时基。

注：	在第一次使能输出比较模块之前，必须先初始化 OCxR 寄存器。当模块工作于 PWM 模式时，OCxR 寄存器变为只读占空比寄存器。OCxR 中保存的值成为第一个 PWM 周期的 PWM 占空比。占空比缓冲寄存器 OCxRS 的内容在发生时基周期匹配之后才会被传送到 OCxR。

图 16-17 给出了 PWM 输出波形的示例。

图 16-17： PWM 输出波形



① 清零 Timery，将新的占空比值从 OCxRS 装入 OCxR。

② 定时器值等于 OCxR 寄存器中的值； OCx 引脚被驱动为低电平。

③ 定时器溢出； OCxRS 的值装入 OCxR ； OCx 引脚被驱动为高电平。
TyIF 中断标志置为有效。

16.3.3.1 带故障保护输入引脚的 PWM

当输出比较模式位 OCM<2:0>（OCxCON<2:0>）被设置为 111 时，选定的输出比较通道被配置为 PWM 工作模式。此时通道具有**第 16.3.3 节 “ 脉宽调制模式 ”** 中所述的所有功能，同时还具有输入故障保护功能。

故障保护通过 OCFA 和 OCFB 引脚提供。 OCFA 引脚与输出比较通道 1 至 4 关联，而 OCFB 引脚与输出比较通道 5 关联。

如果在 OCFA/OCFB 引脚检测到逻辑 0，则选定的 PWM 输出引脚被置为高阻抗状态。用户可以选择在 PWM 引脚连接下拉或上拉电阻，使得在发生故障条件时提供所需的状态。 PWM 输出立即关闭，不连接到器件时钟源。该状态将保持直到满足以下条件：

- 外部故障条件已经消除
- 通过写相应的模式位 OCM<2:0> （OCxCON<2:0>）重新使能 PWM 模式

发生故障条件后，相应的中断标志位 OCxIF 会置为有效，在允许中断的情况下将产生中断。在检测到故障条件时， OCFLT 位（OCxCON<4>）被驱动为高电平（逻辑 1）。该位是只读位，只有在以下情况下才会清零：外部故障条件已经消除，并通过写相应的模式位 OCM<2:0>（OCxCON<2:0>）重新使能 PWM 模式。

注：	在器件处于 SLEEP （休眠）或 IDLE （空闲）模式时，外部故障引脚 （如果使能）将继续控制 OCx 输出引脚。

16.3.3.2 PWM 周期

PWM 周期可通过写入 PRy （Timery 周期寄存器）来指定。 PWM 周期可由以下公式计算：

公式 16-1： 计算 PWM 周期

$$\text{PWM 周期} = [(\text{PR} + 1) \bullet T_{PB} \bullet (\text{TMR 预分频值})]$$

$$\text{PWM 频率} = 1/[\text{PWM 周期}]$$

PWM 周期一定不能超出所选定模式的周期寄存器宽度（对于 16 位模式为 16 位，对于 32 位模式为 32 位）。如果计算得到的周期太大，请选择较大的预分频比，以防止溢出。为了维持最大的 PWM 分辨率，请选择不会导致溢出的最小预分频比。

注：	如果 PRy 的值为 N，则会使 PWM 周期为 N + 1 个时基计数周期。例如，如果写入 PRy 寄存器的值为 7，则将产生由 8 个时基周期组成的 PWM 周期。

16.3.3.3 PWM 占空比

通过写入 OCxRS 寄存器来指定 PWM 占空比。可以在任何时候写 OCxRS 寄存器，但是在 PRy 和 TMRy 发生匹配（即周期结束）前占空比值不会被锁存到 OCxR 中。这可以为 PWM 占空比提供双重缓冲，对于 PWM 的无毛刺操作是极其重要的。在 PWM 模式下，OCxR 是只读寄存器。

PWM 占空比有一些重要的边界参数，包括：

- 如果占空比寄存器 OCxR 中装入 0x0000，则 OCx 引脚将保持低电平 （占空比为 0%）。
- 如果 OCxR 大于 PRy （定时器周期寄存器），则引脚将保持高电平 （占空比为 100%）。
- 如果 OCxR 等于 PRy，则 OCx 引脚在一个时基计数值内为低电平，而在所有其他计数值内均为高电平。

请参见图 16-18 了解 PWM 模式时序的详细信息。表 16-4 至表 16-9 给出了器件外设总线工作于各种不同频率时，所对应的 PWM 频率和分辨率的示例。

公式 16-2： 计算最大 PWM 分辨率

$$\text{最大 PWM 分辨率（位）} = \frac{\log_{10}\left(\frac{F_{PB}}{F_{PWM} \bullet TMRy \bullet \text{预分频比位}}\right)}{\log_{10}(2)}$$

公式 16-3： PWM 周期和占空比计算

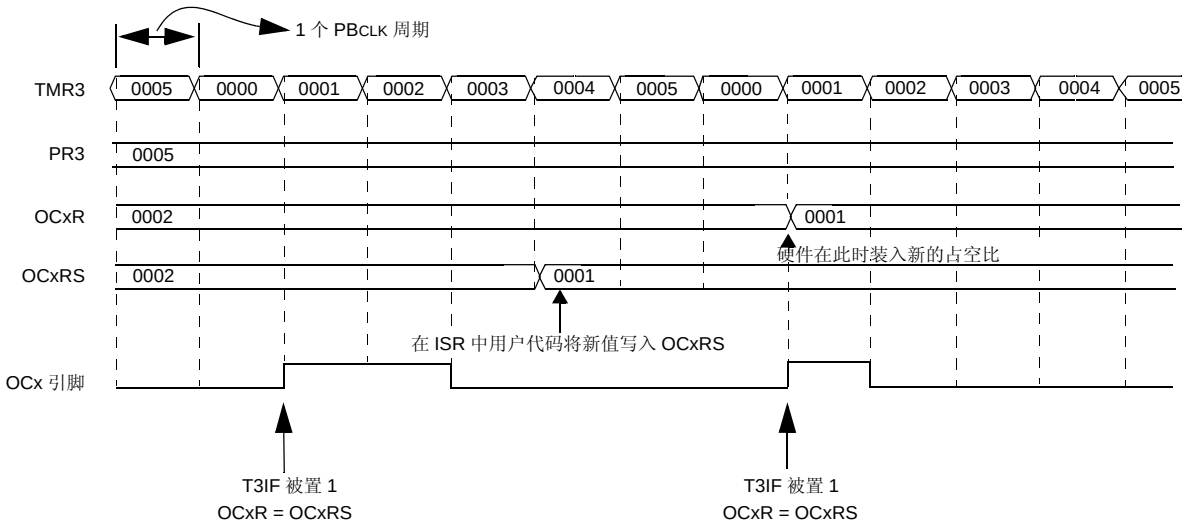
所需的 PWM 频率为 52.08 kHz
F_{PB} = 10 MHz
Timer2 预分频比设置：1:1

$$1/52.08\ \text{kHz} = (PR2 + 1) \bullet T_{PB} \bullet (\text{Timer2 预分频值})$$
$$19.20\ \mu s = (PR2 + 1) \bullet 0.1\ \mu s \bullet (1)$$
$$PR2 = 191$$

确定可用于 52.08 kHz PWM 频率和 10 MHz 外设总线时钟速率的占空比的最大分辨率。

$$1/52.08\ \text{kHz} = 2^{\text{PWM 分辨率}} \bullet 1/10\ \text{MHz} \bullet 1$$
$$19.20\ \mu s = 2^{\text{PWM 分辨率}} \bullet 100\ \text{ns} \bullet 1$$
$$192 = 2^{\text{PWM 分辨率}}$$
$$\log_{10}(192) = (\text{PWM 分辨率}) \bullet \log_{10}(2)$$
$$\text{PWM 分辨率} = 7.6\ \text{位}$$

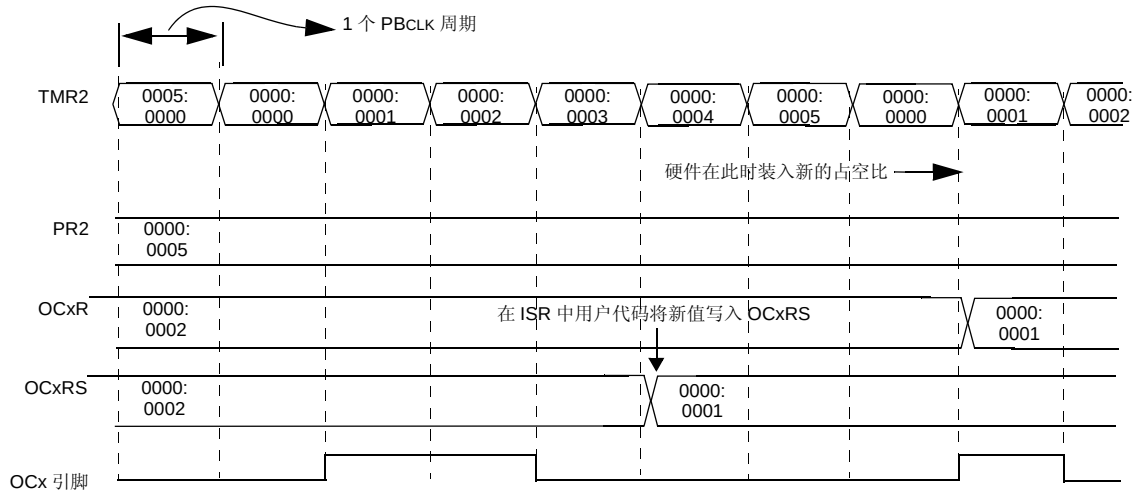
图 16-18： PWM 输出时序



注 1： “x” 表示输出比较通道的编号。

2： OCxR = 比较寄存器； OCxRS = 辅助比较寄存器。

图 16-19： 双比较模式：连续输出脉冲 （PR2 = OCxRS， 32 位模式）



注：“x” 表示输出比较通道的编号。

表 16-4： 采用 10 MHz （16 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	19 Hz	153 Hz	305 Hz	2.44 kHz	9.77 kHz	78.1 kHz	313 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器的值（十六进制）	0xFFFF	0xFFFF	0x7FFF	0x0FFF	0x03FF	0x007F	0x001F
分辨率 （位）（十进制）	16	16	15	12	10	7	5

表 16-5： 采用 30 MHz （16 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	58 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器的值（十六进制）	0xFC8E	0xFFDD	0x7FEE	0x1001	0x03FE	0x007F	0x001E
分辨率 （位）（十进制）	16	16	15	12	10	7	5

表 16-6： 采用 50 MHz （16 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	57 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
定时器预分频比	64	8	1	1	1	1	1
周期寄存器的值（十六进制）	0x349C	0x354D	0xD538	0x1AAD	0x06A9	0x00D4	0x0034
分辨率 （位）（十进制）	13.7	13.7	15.7	12.7	10.7	7.7	5.7

表 16-7： 采用 50 MHz （16 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
定时器预分频比	8	8	8	1	8	1	1
周期寄存器的值（十六进制）	0xF423	0x7A11	0x30D3	0xC34F	0x0C34	0x270F	0x1387
分辨率 （位）（十进制）	15.9	14.9	13.6	15.6	11.6	13.3	12.3

表 16-8： 采用 50 MHz （16 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
定时器预分频比	8	4	2	1	1	1	1
周期寄存器的值（十六进制）	0xF423	0xF423	0xC34F	0x0C34F	0x61A7	0x270F	0x1387
分辨率 （位）（十进制）	15.9	15.9	15.6	15.6	14.6	13.3	12.3

表 16-9： 采用 50 MHz （32 位模式）外设总线时钟时的 PWM 频率和分辨率示例

PWM 频率	100 Hz	200 Hz	500 Hz	1 kHz	2 kHz	5 kHz	10 kHz
定时器预分频比	1	1	1	1	1	8	1
周期寄存器的值（十六进制）	0x0007A11F	0x0003D08F	0x0001869F	0x0000C34F	0x000061A7	0x000004E1	0x00001387
分辨率 （位）（十进制）	18.9	17.9	16.6	15.6	14.6	10.3	12.3

例 16-9 给出了 PWM 工作模式的配置和中断处理代码。

例 16-9： PWM 模式设置和中断处理 （16 位模式）

```
// The following code example will set the Output Compare 1 module
// for PWM mode with FAULT pin disabled, a 50% duty cycle and a
// PWM frequency of 52.08 kHz at Fosc = 40 MHz.Timer2 is selected as
// the clock for the PWM time base and Timer2 interrupts
// are enabled.

OC1CON = 0x0000;                        // Turn off OC1 while doing setup.
OC1R = 0x0060;                          // Initialize primary Compare Register
OC1RS = 0x0060;                         // Initialize secondary Compare Register
OC1CON = 0x0006;                        // Configure for PWM mode
PR2 = 0x00BF;                           // Set period


                                        // configure int
IFS0CLR = 0x00000040;                   // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                   // Enable OC1 interrupt
IPC1SET = 0x001C0000;                   // Set OC1 interrupt priority to 7,
                                        // the highest level
IPC1SET = 0x00030000;                   // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                      // Enable Timer2
OC1CONSET = 0x8000;                     // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                       // Clear the OC1 interrupt flag
}
```

例 16-10： PWM 模式设置和中断处理 （32 位模式）

```
// The following code example will set the Output Compare 1 module
// for PWM mode with FAULT pin disabled, a 50% duty cycle and a
// PWM frequency of 52.08 kHz at Fosc = 40 MHz.Timer2 is selected as
// the clock for the PWM time base and Timer2 interrupts
// are enabled.

OC1CON = 0x0000;                        // Turn off OC1 while doing setup.
OC1R = 0x00600000;                      // Initialize primary Compare Register
OC1RS = 0x00600000;                     // Initialize secondary Compare Register
OC1CON = 0x0006;                        // Configure for single pulse mode
PR2 = 0x00600000;                       // Set period

                                        // configure int
IFS0CLR = 0x00000040;                   // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                   // Enable OC1 interrupt
IPC1SET = 0x001C0000;                   // Set OC1 interrupt priority to 7,
                                        // the highest level
IPC1SET = 0x00030000;                   // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                      // Enable Timer2
OC1CONSET = 0x8000;                     // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
// insert user code here
IFS0CLR = 0x0040;                       // Clear the OC1 interrupt flag
}
```

16.4 中断

每个可用的输出比较通道都具有专用的中断位 OCxIF，以及相应的中断允许 / 屏蔽位 OCxIE。这些位用于决定中断源和使能 / 禁止各个中断源。每个通道的优先级还可以独立于其他通道进行设置。

当输出比较通道检测到预定义的匹配条件（该条件定义为产生中断的事件）时， OCxIF 会置 1。OCxIF 位是否置 1 与相应 OCxIE 位的状态无关。如果需要，可以用软件查询 OCxIF 位。

OCxIE 位用于定义在相应 OCxIF 位置 1 时，向量中断控制器（Vector Interrupt Controller，VIC）的行为。当 OCxIE 位清零时， VIC 模块不会为事件产生 CPU 中断。如果 OCxIE 位置 1，则 VIC 模块会在相应的OCxIF位置1时向CPU产生中断（受以下段落中概述的优先级和子优先级制约）。

处理特定中断的程序需要负责在服务程序完成之前清零相应的中断标志位。

每个输出比较通道的优先级可以通过 OCxIP<2:0> 位独立设置。该优先级定义了中断源将分配到的优先级组。优先级组值的范围为 7 （最高优先级）到 0 （不产生中断）。较高优先级组中的中断会抢占正在处理、但优先级较低的中断。

子优先级位用于设置中断源在优先级组中的优先级。子优先级 OCxIS<1:0> 值的范围为 3 （最高优先级）到 0 （最低优先级）。处于相同优先级组，但具有更高子优先级值的中断会抢占子优先级较低、但正在进行的中断。

优先级组和子优先级位让多个中断源可以共用相同的优先级和子优先级。如果在该配置下同时发生若干个中断，则中断源在优先级 / 子优先级组对中的自然顺序将决定所产生的中断。自然优先级基于中断源的向量编号。向量编号越小，中断的自然优先级就越高。在当前中断的中断标志清零之后，所有不按照自然顺序执行的中断会产生相应的中断 （基于优先级、子优先级和自然顺序）。

产生允许的中断之后，CPU 将跳转到为该中断分配的向量处。该中断的向量编号与自然顺序编号相同。然后，CPU 将在向量地址处开始执行代码。该向量地址处的用户代码应执行任何所需的操作 （如重新装入占空比和清零中断标志），然后退出。关于向量地址表的详细信息和中断的更多信息，请参见**第 8 章 “ 中断 ”**（DS61108）。

16.5 I/O 引脚控制

当输出比较模块被使能时，I/O 引脚方向由比较模块控制。当比较模块被禁止时，它会将 I/O 引脚控制权归还给相应的引脚 LAT 和 TRIS 控制位。

当使能了具有故障保护输入模式的 PWM 时，必须通过将相应的 TRIS SFR 位置 1 以将 OCFx 故障引脚配置为输入。选择 PWM 故障模式时， OCFx 故障输入引脚不会自动配置为输入。

表 16-10： 与输出比较模块 1-5 相关的引脚

引脚名称	模块控制	引脚类型	缓冲器类型	说明
OC1	ON	O	—	输出比较 /PWM 通道 1
OC2	ON	O	—	输出比较 /PWM 通道 2
OC3	ON	O	—	输出比较 /PWM 通道 3
OC4	ON	O	—	输出比较 /PWM 通道 4
OC5	ON	O	—	输出比较 /PWM 通道 5
OCFA	ON	I	ST	PWM 故障保护 A 输入 （用于通道 1-4）
OCFB	ON	I	ST	PWM 故障保护 B 输入 （用于通道 5）

图注： ST = 带 CMOS 电平的施密特触发器输入
I = 输入
O = 输出

16.6 节能和调试模式下的操作

注： 在本手册中，对于特定模块中使用的功耗模式和器件使用的功耗模式进行了区分；例如，比较器的 Sleep（休眠）模式和 CPU 的 SLEEP（休眠）模式。为了指示所期望功耗模式的类型，模块功耗模式使用大写字母加小写字母（Sleep，Idle，Debug）（休眠、空闲和调试）来表示，器件功耗模式使用全大写字母（SLEEP，IDLE，DEBUG）（休眠、空闲和调试）来表示。

16.6.1 SLEEP（休眠）模式下的输出比较操作

当器件进入 SLEEP（休眠）模式时，系统时钟被禁止。在 SLEEP（休眠）模式期间，输出比较模块会将引脚驱动为与在进入 SLEEP（休眠）模式之前相同的有效状态。然后模块将暂停在该状态。

例如，如果引脚原先为高电平，则在 CPU 进入 SLEEP（休眠）状态后，引脚将保持高电平。类似地，如果引脚原先为低电平，则在 CPU 进入 SLEEP（休眠）状态后，引脚将保持低电平。在这两种情况下，当器件唤醒时，输出比较模块将继续工作。

当模块在 PWM 故障模式下工作时，故障电路的异步部分将保持工作状态。如果检测到故障，比较输出使能信号会置为无效，OCFLT（OCxCON<4>）被置 1。如果允许了相应中断，还将产生中断，并且器件将从 SLEEP（休眠）模式唤醒。

16.6.2 IDLE（空闲）模式下的输出比较操作

当器件进入 IDLE（空闲）模式时，系统时钟源保持工作，但 CPU 停止执行代码。SIDL 位（OCxCON<13>）用于选择比较模块在器件进入 IDLE（空闲）模式时是停止工作还是在 IDLE（空闲）模式下继续正常工作。

- 如果 SIDL = 1，则在 IDLE（空闲）模式下模块将停止工作。模块在 IDLE（空闲）模式下停止工作时将执行与在 SLEEP（休眠）模式下相同的程序。
- 如果 SIDL = 0，则只有选定时基设置为在 IDLE（空闲）模式下工作时，模块才能在 IDLE（空闲）模式下继续工作。如果 SIDL 位为逻辑 0，则输出比较通道将在 CPU IDLE（空闲）模式期间工作。此外，还必须将相应的 SIDL 位设为逻辑 0 以使能时基。

注： 在器件处于 SLEEP（休眠）或 IDLE（空闲）模式时，外部故障引脚（如果使能）将继续控制相关的 OCx 输出引脚。

- 当模块在 PWM 故障模式下工作时，故障电路的异步部分将保持工作状态。如果检测到故障，比较输出使能信号会置为无效，OCFLT（OCxCON<4>）被置 1。如果允许了相应中断，还将产生中断，并且器件将从 IDLE（空闲）模式唤醒。

16.6.3 DEBUG（调试）模式下的输出比较操作

FRZ 位（OCxCON<14>）决定 CPU 在 DEBUG（调试）模式下执行调试异常代码（即，应用程序暂停）时，输出比较模块是继续运行还是停止。如果 FRZ = 0，则在 DEBUG（调试）模式下，即使应用程序暂停，输出比较模块也会继续工作。当 FRZ = 1 且应用程序在 DEBUG（调试）模式下暂停时，模块将停止工作，并且不更改输出比较模块的状态。在 CPU 继续开始执行代码之后，模块将继续工作。

当模块在 PWM 故障模式下工作时，故障电路的异步部分将保持工作状态。如果检测到故障，比较输出使能信号会置为无效，OCFLT（OCxCON<4>）被置 1。如果允许了相应中断，还将产生中断。

注： 只有 CPU 在调试异常模式下执行时，FRZ 位才可读写。在所有其他模式下，FRZ 位读为 0。如果 FRZ 位在 DEBUG（调试）模式期间发生改变，则只有退出当前调试异常模式并重新进入该模式之后，新值才会生效。在调试异常模式期间，在进入 DEBUG（调试）模式时 FRZ 位会读取外设状态。

16.7 各种复位的影响

16.7.1 $\overline{\text{MCLR}}$ 复位

在发生 $\overline{\text{MCLR}}$ 事件之后，每个输出比较模块的 OCxCON、 OCxR 和 OCxRS 寄存器会复位为值 0x00000000。

16.7.2 上电复位

在发生上电 （Power-on， POR）事件之后，每个输出比较模块的 OCxCON、 OCxR 和 OCxRS 寄存器会复位为值 0x00000000。

16.7.3 看门狗定时器复位

在发生看门狗定时器 （Watchdog Timer， WDT）事件之后， OCMP 控制寄存器的状态取决于 WDT 事件之前 CPU 的工作模式。

如果器件不处于 SLEEP （休眠）模式，则 WDT 事件会将 OCxCON、 OCxR 和 OCxRS 寄存器强制为复位值 0x00000000。

如果在发生 WDT 事件时器件处于 SLEEP （休眠）模式，则 OCxCON、 OCxR 和 OCxRS 寄存器值不受影响。

16.8 输出比较应用

以下是一个使用输出比较模块的 PWM 模式来控制直流电机速度的示例应用。电机速度通过更改 PWM 占空比进行控制。

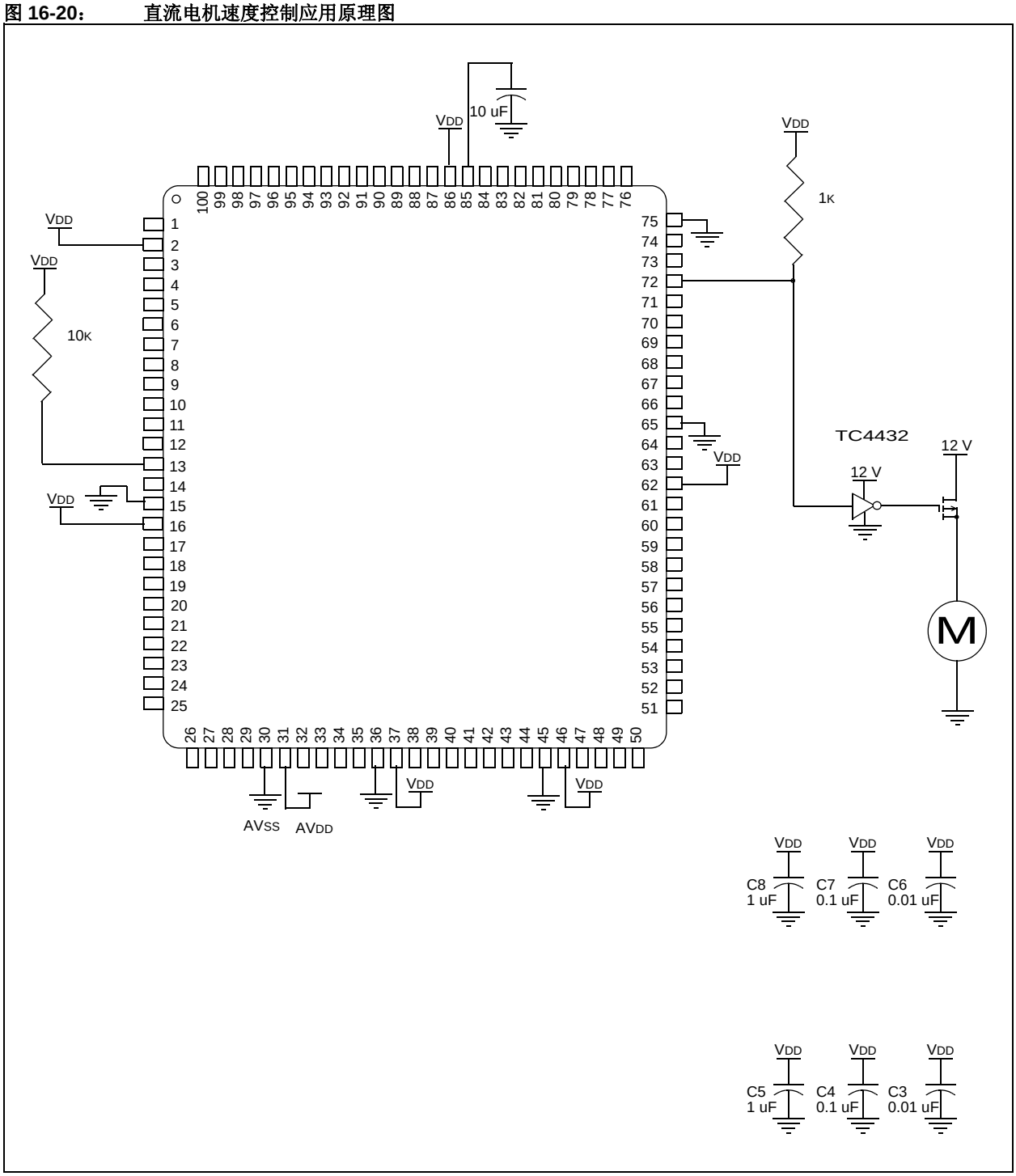
电路包含以下组成部分：

- PIC32MX 器件，用于产生 PWM。
- TC4431 或等效的 MOSFET 驱动器，用于驱动 MOSFET。
- MOSFET，用于驱动电机。
- 上拉电阻，用于在 PIC32MX 处于复位状态时将 MOSFET 驱动器的输入上拉为高电平。这可以防止在启动期间发生意外的电机操作。
- 直流电机。

例 16-11： PWM 模式示例应用 （16 位模式）

```
// The following code example will set the Output Compare 1 module
// for PWM mode w/o FAULT pin enabled, a 50% duty cycle and a
// PWM frequency of 52.08 kHz at FP = 40 MHz.Timer2 is selected as
// the clock for the PWM time base and Timer2 interrupts
// are enabled.This example ramps the PWM duty cycle from min to max, then
// from max to min and repeats.The rate at which the PWM duty cycle is
// changed can be adjusted by the rate at which the Timer2 overflows.
// The PWM period can be changed by writing a different value to the PR2
// register.If the PR2 value is adjusted the maximum PWM value will also
// have to be adjusted so that it is not greater than the PR2 value.

unsigned int Pwm;                       // variable to store calculated PWM value
unsigned char Mode = 0;                 // variable to determine ramp up or ramp down

OC1CON = 0x0000;                        // Turn off OC1 while doing setup.
OC1R = 0x0000;                          // Initialize primary Compare Register
OC1RS = 0x0000;                         // Initialize secondary Compare Register
OC1CON = 0x0006;                        // Configure for PWM mode
PR2 = 0xFFFF;                           // Set period

                                        // configure int
IFS0CLR = 0x00000040;                   // Clear the OC1 interrupt flag
IFS0SET = 0x00000040;                   // Enable OC1 interrupt
IPC1SET = 0x001C0000;                   // Set OC1 interrupt priority to 7,
                                        // the highest level
IPC1SET = 0x00030000;                   // Set Subpriority to 3, maximum

T2CONSET = 0x8000;                      // Enable Timer2
OC1CONSET = 0x8000;                     // Enable OC1

// Example code for Output Compare 1 ISR:

void__ISR(_OUTPUT_COMPARE_1_VECTOR, ipl7) OC1_IntHandler (void)
{
if ( Mode )
    {
        if ( Pwm < 0xFFFF )             // ramp up mode
        {
            Pwm ++;                     // If the duty cycle is not at max, increase
            OC1RS = Pwm;                // Write new duty cycle
        }
        else
        {
            Mode = 0;                   // PWM is at max, change mode to ramp down
        }
    }                                   // end of ramp up
    else
    {
        if ( !Pwm )                     // ramp down mode
        {
            Pwm --;                     // If the duty cycle is not at min, increase
            OC1RS = Pwm;                // Write new duty cycle
        }
        else
        {
            Mode = 1;                   // PWM is at min, change mode to ramp up
        }
    }                                   // end of ramp down

        // insert user code here
        IFS0CLR = 0x0040;               // Clear the OC1 interrupt flag
}
```

图 16-20：　直流电机速度控制应用原理图



16.9 设计技巧

问 1： ***即使当 SIDL 位未置 1 时，输出比较引脚仍停止工作。这是什么原因？***

答 1： 当相关定时器源的 SIDL 位（TxCON<13>）置 1 时，最可能发生此问题。因此，当执行 PWRSAV 指令时，实际上是定时器进入了 IDLE（空闲）模式。

问 2： ***是否能在选定时基配置为 32 位模式时使用输出比较模块？***

答 2： 可以。定时器可以在 32 位模式下使用，用作输出比较模块的时基，方法是将 T32 位（TxCON<3>）置 1。为了正确工作，输出比较模块必须配置为 32 位比较模式，方法是将 OC32 位（OCxCON<5>）置 1，使所有输出比较模块都使用 32 位定时器作为时基。

16.10 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与输出比较模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

16.11 版本历史

版本 A （2007 年 10 月）

这是本文档的初始版本。

版本 B （2007 年 10 月）

更新了文档 （删除了 “ 机密 ” 状态）。

版本 C （2008 年 4 月）

将状态修改为 “ 初稿 ” ； 将 U-0 修改为 r-x。

版本 D （2008 年 6 月）

修改了寄存器 16-1、16-20 和 16-32 ； 修改了例 16-3、16-4、16-5、16-6、16-7、16-8、16-9、16-10 和 16-11 ； 向汇总表中增加了 TMR1 和 TMR2 ； 修改了第 16.3 节的注释； 将保留位从 “ 保持为 ” 更改为 “ 写入 ” ； 为 ON 位 （OCxCON、 T2CON 和 T3CON 寄存器） 增加了注释。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “ 牢不可破 ” 的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

ISBN: 978-1-60932-327-1

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

亚太地区

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10