# 16-Bit 脉冲宽度调制器数据表 PWM16 V 2.5

Copyright © 2000-2011 Cypress Semiconductor Corporation. All Rights Reserved.

| 资源 | PSoC® 模块 | | | API 内存 （字节） | | 引脚 （每个外部 I/O） |
|---|---|---|---|---|---|---|
| | 数字 | 模拟 CT | 模拟 SC | 闪存 | RAM | |
| CY8C29/27/24/22/21xxx、CY8C23x33、CY7C64215/603xx、CYWUSB6953、CY8CLED02/04/08/16、CY8CLED0xD、CY8CLED0xG、CY8CNP102、CY8CTST110、CY8CTMG110、CY8CTST120、CY8CTMG120、CY8CTMA120、CY8C21x45、CY8C22x45、CY8CTMA30xx、CY8C28x45、CY8CPLC20、CY8CLED16P01、 CY8C28xxx | | | | | | |
| 16-bit | 2 | 0 | 0 | 89 | 0 | 1 |
| CY8C26/25xxx | | | | | | |
| 16-bit | 2 | 0 | 0 | 138 | 0 | 1 |

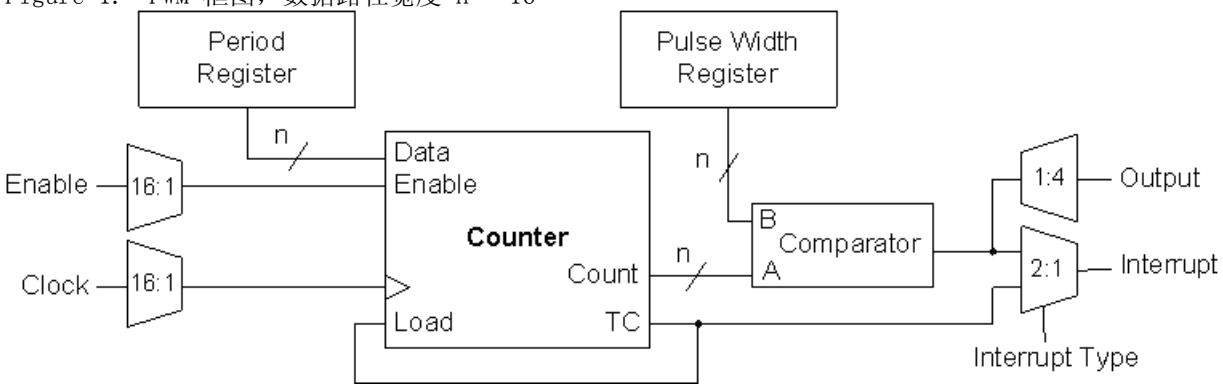
如需一个或多个使用此用户模块的完全配置的功能性示例工程，请转到 www.cypress.com/psocexampleprojects.

## 特性与概述

- 16-bit 通用脉冲宽度调制器使用两个 PSoC 模块
- 源时钟频率最大为 48 MHz
- 为每次脉冲周期自动重新载入周期
- 可编程脉冲宽度
- 输入启用 / 禁用连续计数器操作
- 基于输出上升沿或终端计数的中断选项

16-bit PWM 用户模块是一个脉冲宽度调制器，具有可编程的周期和脉冲宽度。 可以从多个源中选择时钟以及使能信号。 输出信号可以路由到引脚或全局输出总线之一，以供其他用户模块内部使用。 通过编程，可以在输出的上升沿或计数器达到终端计数时触发中断。

Figure 1. PWM 框图，数据路径宽度 n = 16



## 功能说明

PWM 用户模块采用两个数字 PSoC 模块，每个模块提供 8 位分辨率。 为形成 16-bit 脉冲宽度调制器，两个连续的模块是相连的，所以它们的内部进位位、终端计数和比较信号为同步链接。 这将同步各个计数、周期和比较寄存器 （分别为数据寄存器 DR0、DR1 和 DR2），以提供所需的 16-bit 分辨率。

PWM API 提供可从 C 语言程序和汇编语言程序调用的函数，以停止和启动计数器操作，并读取和写入各种数据寄存器。 数据寄存器值还可通过使用器件编辑器建立。 启动后，计数寄存器会在高电平有效使能输入信号被置位的每个时钟周期上升沿出现时递减。 在终端计数 （当计数寄存器达到零时）后的风险时钟沿上，计数寄存器重新加载为周期寄存器中的值。

随时可以将周期寄存器修改为新值。 PWM 停止时，向周期寄存器写入值同样会更改计数寄存器中的值。 PWM 运行时，写入周期寄存器不会将新的周期值更新到计数寄存器，直至终端计数后发生再次重新载入。 由于计数为零时达到终端计数，因此操作和输出信号的周期比周期寄存器中存储的值大 1。 以下等式说明了 PWM 输出与输入时钟和周期寄存器值之间的关系。

TOUT = (PeriodValue+1)/FCLOCK

$$FOUT = FCLOCK/(PeriodValue+1) \quad \textbf{Equation 1}$$

这里的 *F*OUT 是 PWM 的输出频率，*T*OUT 是 PWM 的输出周期，*F*CLOCK 是输入时钟频率，*PeriodValue* 是周期的输入值。

PWM 停止时将输出置为低电平。 运行时，由比较器控制输出信号的占空比。 在每个时钟周期中，此比较器测试计数寄存器的值与脉冲宽度寄存器之间的关系，根据使用器件编辑器时选择的选项执行 “ 小于 ”(Less Than) 或 “ 小于或等于 ”(Less Than Or Equal) 测试。 PWM 在执行比较后的周期时钟上升沿置位该比较的高电平有效真值。 脉冲宽度值和周期的比率设置输出波形的占空比。 可以使用下列等式计算占空比。

如果脉冲宽度值 < 周期值：

**Equation 2**

$$DutyCycle = \begin{cases} \frac{PulseWidthValue}{PeriodValue+1}, \text{For } Less\ Than \text{ comparison} \\ \frac{PulseWidthValue+1}{PeriodValue+1}, \text{For } Less\ Than\ Or\ Equal\ To \text{ comparison} \end{cases}$$

如果脉冲宽度值 >= 周期值

占空比 = 100%

下表汇总了基于周期、脉冲宽度和比较操作设置的部分特殊输出信号条件。

Table 1. 计数器特殊输出信号条件

| 周期寄存器值 | 比较类型 | 脉冲宽度寄存器值 | 脉冲宽度高电平时间和周期的比率 |
|---|---|---|---|
| 0 | 无需关注 | > 0 | 1.0 |
| 0 | ≤ | 0 | 1.0 |
| 0 | < | 0 | 0.0 |
| > 0 | ≤ | 0 | 1/ （周期 +1） |



[+] Feedback

| 周期寄存器值 | 比较类型 | 脉冲宽度寄存器值 | 脉冲宽度高电平时间和周期的比率 |
|---|---|---|---|
| > 0 | < | 0 | 0.0 |
| 周期 = 脉冲宽度 | ≤ | 周期 = 脉冲宽度 | 1.0 |
| 周期 = 脉冲宽度 | < | 周期 = 脉冲宽度 | 周期 / （周期 +1） |
| 脉冲宽度值 > 周期 | 无需关注 | 脉冲宽度值 > 周期 | 1.0 |

可使用器件编辑器或在运行期间使用 API 设置脉冲宽度寄存器值。 周期寄存器在终端计数前缓冲计数寄存器，但在脉冲宽度寄存器中未提供缓冲。 因此，脉冲宽度寄存器的更改会在下一个时钟周期影响比较输出，而非在终端计数后。 这会产生包含多个脉冲的周期。

在 CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx 器件系列中，PWM 用户模块提供终端计数信号作为辅助输出。 在终端计数后的时钟周期上升沿置位高电平有效信号，此时计数寄存器从周期寄存器载入。

可以对中断进行编程，使其基于终端计数或当比较结果为真时发生。 比较器输出在输出信号的上升沿触发中断，终端计数在输出信号下降沿之前的半个时钟周期上触发中断。 此选项可用器件编辑器设置。 启用或禁用中断通过使用计数器 API 在运行时完成。 全局中断必须在计数器中断触发前启用。

修改脉冲宽度寄存器时必须谨慎，因为其值与当前计数值一起决定 PWM 的输出状态。 为防止可能出现输出信号低电平和潜在故障，必须在检测到终端计数条件使用中断后修改脉冲宽度寄存器。

对于要求更快占空比更新间隔的应用程序，PWM 输出可以路由到处于轮询状态的引脚。 一旦检测到输出从高电平跃变到低电平，脉冲宽度即可更新。 请注意，如果脉冲宽度引发 “ 比较结果为真 ” 条件，则会在下一个时钟上将输出置为高电平。

必须谨慎获取计数寄存器值。 读取计数寄存器将使其内容锁存到脉冲宽度寄存器中。 这会引起输出占空比更改。

如果您需要实时读取计数寄存器，可以调用 ReadCounter() API 函数。 此函数会暂时禁用时钟、保存脉冲宽度寄存器内容、读取计数寄存器、读取脉冲宽度寄存器、存储脉冲宽度寄存器然后存储时钟。 有关可能出现的副作用，请参见 “ 应用程序编程接口 ” 一节关于 ReadCounter() 函数的说明。

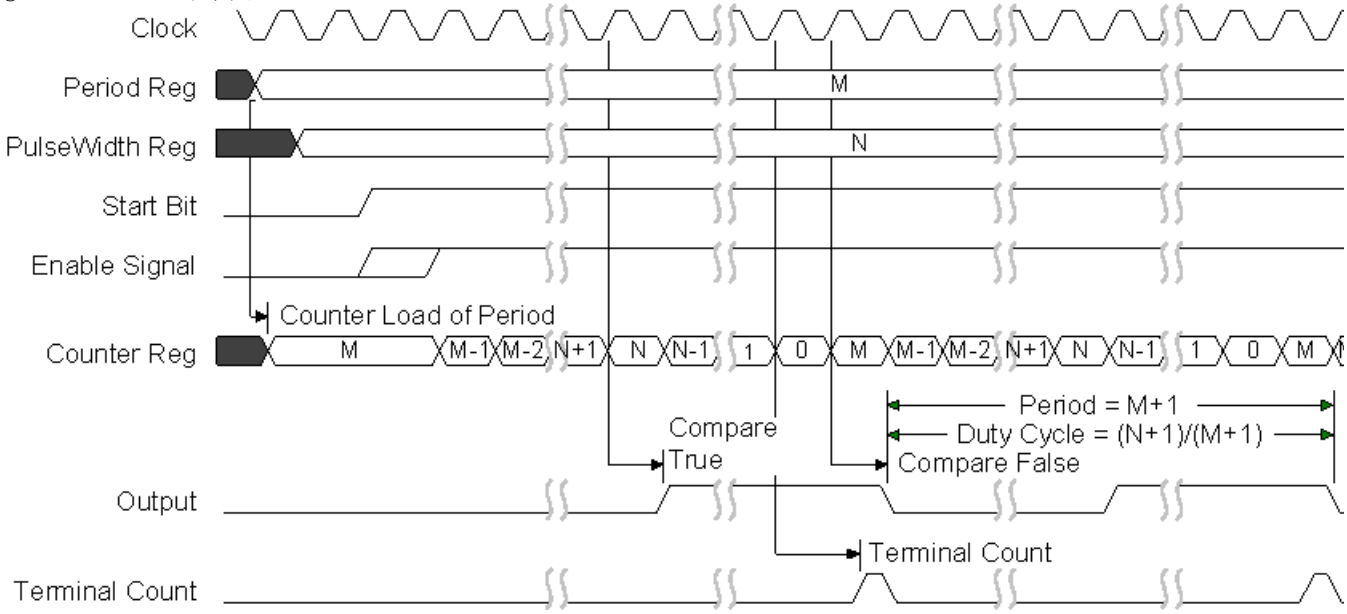


[+] Feedback

## 时序

PWM 操作可被置为接通和关断，或由路由至 PWM 的外部引脚来计时，路由通过器件全局总线特性来完成。

Figure 2. PWM 时序图




## 直流和交流电气特性

Table 2. PWM 直流和交流电气特性

| 参数 | 典型值 | 限值 | 单位 | 条件和注释 |
|---|---|---|---|---|
| $FOutput_{max}$ | -- | 24[1] | MHz | 5.0V 和 48 MHz 输入时钟 |
| | -- | 12[2] | MHz | 3.3V 和 24 MHz 输入时钟 |

电气特性注释

1. 若通过全局总线路由输出，则频率会限制在最大 12 MHz 以内。
2. 工作电压为 3.3V 时 PSoC 模块的最快时钟为 24 MHz。

## 放置

PWM 每 8 位分辨率使用一个数字 PSoC 模块。 模块由器件编辑器按照模块编号升序从最低有效位 (LSB) 到最高有效位 (MSB) 连续放置。 每个模块都具有符号名称，在放置期间和放置后由器件编辑器显示该名称。 API 会使用用户分配的实例名称和模块名称来描述所有寄存器的名称，以便从 API 包含文件直接访问 PWM 寄存器。 下表给出了各种宽度使用的模块名称。

Table 3. PWM 符号 PSoC 模块名称

| PSoC 模块 | 16-Bit PWM |
|---|---|
| 1 | PWM16_LSB |
| 2 | PWM16_MSB |



[+] Feedback

# 参数和资源

### 时钟

可以从 16 个源中选得 “ 时钟 ”(Clock) 参数。 这些源包括 48 MHz 振荡器(工作电压只能为 5.0 V)、从 24 MHz 系统时钟向下分频的较低频率 (VC1、VC2 和 VC3)、其他 PSoC 模块以及通过全局输入和输出路由的外部输入。 在模块中使用外部数字时钟时，为达到最高精度并使用睡眠操作应关闭行输入同步。

### 使能

可以从 16 个源中选得 “ 使能 ”(Enable) 参数。 高输入可启用连续的计数，而低使能可在不重置计数器的情况下禁用计数。

### CompareOut

比较输出可能被禁用 (不干扰中断操作)或连接到任何行输出总线。 不受设置的影响，此参数始终可输入到下一个更高的数字 PSoC 模块和模拟列时钟选择多路器。 此参数仅在 PSoC 器件的 CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx 系列产品中显示。

### TerminalCountOut

终端计数输出是一种辅助计数器输出。 此参数允许禁用或连接到任何行输出总线。 此参数仅在 PSoC 器件的 CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx 系列产品中显示。

### 周期

此参数设置计数器的周期。 PWM8 的允许值范围为 0 至 255。 PWM16 的允许值范围为 0 至 $2^{16}$-1。 周期将载入周期寄存器。 PWM16 的有效输出波形周期为周期计数加 1。 可使用 API 修改该值。

### PulseWidth

设置 PWM 输出的脉冲宽度。 允许值范围为零至周期值。 可使用 API 修改该值。

### InterruptType

此参数设置中断触发类型。 可以设置中断，使其在输出信号的上升沿或计数器寄存器终端计数上触发。 单独的寄存器可以独自启用中断。

### CompareType

此参数可以设置比较函数类型为 “ 小于 ”(Less Than) 或 “ 小于或等于 ”(Less Than or Equal To)。

### ClockSync

在 PSoC 器件中，数字模块可以在系统时钟以外提供时钟源。 数字时钟源甚至可以用连锁方式串联起来。 这样就会引起与系统时钟相关的时滞。 由于各种数据路径优化，在 CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx PSoC 器件系列中，这些时滞更具危害性，特别是应用于系统总线的部分。 此参数可用于控制时钟时滞，确保读取和写入 PSoC 模块寄存器值时进行正确操作。 此参数的正确数值必须根据下表决定。



[+] Feedback

| 时钟同步值 | 使用说明 |
|---|---|
| 同步到 SysClk | 当使用任何由 24 MHz (SysClk) 所衍生出来的低于 24 MHz 的输入时钟源，使用此设定。 这样的时钟源包括 VC1、VC2、VC3 （在 VC3 由 SysClk 驱动时）、32KHz 和以 SysClk 作为时钟源的数字 PSoC 模块。 外部生成的时钟源也应当使用此数值来确保产生正确的同步。 |
| Sync 到 SysClk*2 | 当使用任何基于 48 MHz (SysClk*2) 的低于 48 MHz 的输入时钟，使用此设定。 |
| 使用 SysClk Direct | 需要 24 MHz (SysClk/1) 的时钟时使用。 此项选择并不真正执行同步，但提供了对系统时钟本身的低时滞访问方式。 如果选择此项，则此选项将覆盖之前 “ 时钟 ”(Clock) 参数的设置。 在所有分频器组合起来最终生成了 24 MHz 的输出时，一定要使用此项，而不使用 VC1、VC2、VC3 或数字模块。 |
| Unsynchronized | 选中 48 MHz (SysClk*2) 输入时使用。<br>在需要未同步输入时使用。 一般来说，只有在中断生成是计数器的唯一应用时才推荐使用此选项。 在睡眠期间依然保持活动状态的模块需要进行此设置。 |

**InvertEnable**

此参数确定使能输入信号的意义。 选中 “ 正常 ”(Normal) 时，使能输入为高电平有效。 选择 “ 反相 ” (Invert) 则解释为低电平有效。 InvertEnable 仅适用于 PSoC 器件的 CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx 系列。

## 中断产生控制 (Interrupt Generation Control)

当选中 PSoC Designer 中的 **“ 启用中断产生控制 ”** 复选框时，有两个附加参数将变为可用。 此复选框位于 **“ 项目 ” > “ 设置 ” > “ 芯片编辑器 ” 之下。** 当拥有多个程序层而且多个用户模块在不同的程序层共享中断时，中断生成控制是非常重要的：

- 中断 API (Interrupt API)
- IntDispatchMode

**InterruptAPI**

InterruptAPI 参数允许有条件地生成一个用户模块的中断处理程序和中断矢量表入口。 选择 “ 启用 ”(Enable) 以生成中断处理程序和中断矢量表条目。 选择 “ 禁用 ”(Disable) 以取消生成中断处理程序和中断矢量表条目。 在那些拥有多个程序层而且有多个程序层使用同一模块资源的项目中，特别推荐要正确地选择是否要生成中断 API。 仅在必要时选择生成中断 API，这样可以避免生成中断调度代码，从而减少开销。

**IntDispatchMode**

IntDispatchMode 参数用于指定如何处理中断请求（当处于同一模块的多个用户模块在不同的程序层共享该中断时）。 选择 “ActiveStatus” 会导致固件在为共享的中断请求提供服务之前测试哪一个程序层正处于活动状态。 这项测试发生在每次请求共享中断的时候。 这样会导致延迟增加并且会产生一个服务于共享中断请求的非确定性的程序，但并不要求任何 RAM 资源。 选择 “OffsetPreCalc” 参数会导致固件只在最初已经有一个程序层运行时计算共享中断请求的来源。 这项计算减少了中断延迟会产生一个服务于共享中断请求的确定性程序，但其代价是 1 个字节的 RAM 资源。



[+] Feedback

# 应用程序编程接口

应用程序编程接口 (API) 子程序作为用户模块的一部分提供，使设计人员能够在较高的层级处理模块。本部分具体说明了每个函数对应的接口以及 “include” 文件所提供的相关常量。

Note 在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能因调用 API 函数而更改。 发起调用 API 的函数有责任在调用之前保留 A 和 X 的值 （如果调用后需要再次用到它们）。 选择这种 “ 寄存器易变 ” 策略是为了提高效率，并且自从 PsoC Designer 的 1.0 版本起使用。 C 编译器自动遵循此要求。 汇编语言程序员也必须确保其代码遵守这一策略。 虽然有些用户模块的 API 函数可能保留 A 和 X 不变，但并不保证在未来也将如此。

应用程序编程接口 (API) 子程序作为用户模块的一部分提供，使设计人员能够在较高的层级处理模块。以下是为 PWM16 提供的 API 编程子程序。

## PWM16_PERIOD

**说明：**

表示器件编辑器中为 PWM16 的 “ 周期 ”(Period) 字段选择的值。 该值的范围介于 0 到 65535 之间。

## PWM16_PULSE_WIDTH

**说明：**

表示器件编辑器中为 PWM16 的 PulseWidth 字段选择的值。 该值的范围介于 0 到 65535 之间。

## PWM16_EnableInt

**说明：**

启用中断模式操作。

**C 原型：**

```
void  PWM16_EnableInt(void);
```

**汇编：**

```
lcall  PWM16_EnableInt
```

**参数：**

无

**返回值：**

无

**副作用：**

此函数可能更改 A 和 X 寄存器。

## PWM16_DisableInt

**说明：**

禁用中断模式操作。

**C 原型：**

```
void  PWM16_DisableInt(void);
```



[+] Feedback

**汇编：**

```
lcall  PWM16_DisableInt
```

**参数：**

无

**返回值：**

无

**副作用：**

此函数可能更改 A 和 X 寄存器。

### PWM16_Start

**说明：**

启动 PWM16 用户模块。 如果使能输入处于高电平，计数器就开始递减。

**C 原型：**

```
void  PWM16_Start(void);
```

**汇编：**

```
lcall  PWM16_Start
```

**参数：**

无

**返回值：**

无

**副作用：**

此函数可能更改 A 和 X 寄存器。

### PWM16_Stop

**说明：**

停止计数器操作。

**C 原型：**

```
void  PWM16_Stop(void);
```

**汇编：**

```
lcall  PWM16_Stop
```

**参数：**

无

**返回值：**

无

**副作用：**

输出复位为低电平，写入周期寄存器会导致计数器寄存器更新为新周期值。 此函数可能更改 A 和 X 寄存器。

## PWM16_WritePeriod

**说明：**

将周期值写入到周期寄存器。 如果 PWM16 停止或者计数器达到零计数时，周期值立即从周期寄存器传输至计数器寄存器。

**C 原型：**

```
void  PWM16_WritePeriod(WORD wPeriod);
```

**汇编：**

```
mov   X, [wPeriod]
mov   A, [wPeriod+1]
lcall  PWM16_WritePeriod
```

**参数：**

wPeriod：wPeriod 值的范围为 0 到 $2^{16}$-1。 最高有效位在 X 寄存器中传递，而最低有效位在累加器中传递。

**返回值：**

无

**副作用：**

此函数可能更改 A 和 X 寄存器。

## PWM16_WritePulseWidth

**说明：**

将脉冲宽度值写入脉冲宽度寄存器。

**C 原型：**

```
void  PWM16_WritePulseWidth(WORD wPulseWidth);
```

**汇编：**

```
mov   X, [wPulseWidth]
mov   A, [wPulseWidth+1]
lcall  PWM16_WritePulseWidth
```

**参数：**

wPulseWidth：wPulseWidth 值的范围为 0 到周期值。 最高有效位在 X 寄存器中传递，而最低有效位在累加器中传递。

**返回值：**

无

**副作用：**

计数器处于活动状态时写入脉冲宽度寄存器会改变输出的占空比。 这可能导致输出故障或者不慎更改。 此函数可能更改 A 和 X 寄存器。

## PWM16_wReadPulseWidth

**说明：**

读取脉冲宽度寄存器。



[+] Feedback

**C 原型：**

```
WORD  PWM16_wReadPulseWidth();
```

**汇编：**

```
lcall  PWM16_wReadPulseWidth
mov   [wPulseWidth], X
mov   [wPulseWidth+1], A
```

**参数：**

无

**返回值：**

脉冲宽度值存储于脉冲宽度寄存器中。 最高有效位在 X 寄存器中传递，而最低有效位在累加器中传递。

**副作用：**

此函数可能更改 A 和 X 寄存器。

## PWM16_wReadCounter

**说明：**

读取计数器寄存器。

请注意此函数针对的是必须读取动态计数器寄存器的应用程序，而且会产生一些副作用。

**C 原型：**

```
BYTE  PWM16_wReadCounter();
```

**汇编：**

```
lcall  PWM16_wReadCounter
mov   [wCounter], X
mov   [wCounter+1], A
```

**参数：**

无

**返回值：**

返回计数器寄存器值。 最高有效位在 X 寄存器中传递，而最低有效位在累加器中传递。

**副作用：**

为了读取 PWM16 计数器寄存器，必须暂时修改脉冲宽度寄存器。 这可能导致 PWM16 计数器寄存器操作延缓一次或更多计数。 另外，这可能导致不慎中断条件。 此函数可能更改 A 和 X 寄存器。

## 固件源代码示例

在以下示例中，C 语言和汇编语言代码之间的对应关系简单且直接。 显示的周期值和比较值都与基本值相差一，因为寄存器是从零开始的，即零是递减计数循环的终端计数。 在 A 寄存器中而非堆栈中传递单一字节参数，这是汇编程序和 C 语言程序针对用户模块 API 使用的性能优化方式。 当在 PWM16.h 文件中遇到 #pragma 快速调用声明时，C 语言编译器会对 “INT” 类型应用此机制，而不会将参数推入堆栈。

以下汇编语言源代码说明了 API 的使用。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Function:  GenerateOneThirdDutyCycle
;   Description:
;      This sample shows how to create a 33% duty cycle output
;      pulse. The clock selected should be 1000 times the required period.
;      The comparator operation is specified to be "Less than or Equal".
;
;   Parameters:  none
;   Returns:     none
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include "PWM16.inc" ; include the PWM16 API include file

GenerateOneThirdDutyCycle:
   mov   A, E7h                 ; set period to be 1000 counts of the clock
   mov   X, 03h                 ; Period is set to 1000 - 1 = 999 (0x3E7).
   call  PWM16_WritePeriod
   mov   A, 4ch                 ; set PulseWidth to generate a 33% duty cycle
   mov   X, 01h                 ; Pulse Width = 1000/3 - 1 = 332 (0x14C).
   call  PWM16_WritePulseWidth
   call  PWM16_DisableInt       ; ensure that interrupts are disabled
   call  PWM16_Start            ; start the PWM16 - counter will start to
   ret                          ; count when the enable input is asserted
                                ; high
```

同一代码用 C 语言表示为：

```
/* include the Counter16 API header file    */
#include "PWM16.h"

/* function prototype */
void GenerateOneThirdDutyCycle(void);

/* Divide by eight function */
void GenerateOneThirdDutyCycle(void)
{
    /* set period to eight clocks */
    PWM16_WritePeriod(999);
    /* set pulse width to generate a 33% duty cycle */
    PWM16_WritePulseWidth(332);
    /* ensure interrupt is disabled */
    PWM16_DisableInt();
    /* start the PWM16! */
    PWM16_Start();
}
```



[+] Feedback

## 配置寄存器

除特别说明以外，本节所给的寄存器规范适用于所有 PSoC 器件系列。

16-bit PWM 使用两个数字 PSoC 模块。 按照从左到右的放置顺序，分别为 PWM16_LSB 和 PWM16_MSB. 通过 7 个寄存器对每个模块进行了个性化和参数化设置。 以下表格给出了作为常量和参数的 “ 特性 ” 值，命名为带有简要描述的位字段。 寄存器符号名在用户模块实例的 C 语言和汇编语言接口文件 （后缀名为 “.h” 和 “.inc” 的文件）中定义。

Table 4. 功能寄存器，组 1 CY8C26/25xxx

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 0 | 0 | 1 | 比较类型 | 中断类型 | 0 | 0 | 1 |
| LSB | 0 | 0 | 1 | 比较类型 | 0 | 0 | 0 | 1 |

Table 5. 功能寄存器，组 1 CY8C29/27/24/22/21xxx 和 CY8CLED04/08/16

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 数据反相 | 0 | 1 | 比较类型 | 中断类型 | 0 | 0 | 1 |
| LSB | 0 | BCEN | 1 | 比较类型 | 0 | 0 | 0 | 1 |

BCEN 将比较输出传送至行广播总线。 此位字段在器件编辑器中通过直接配置广播线进行设置。 “ 数据反相 ” 标志用于控制捕获输入信号的意义，此参数通过显示在器件编辑器中的用户模块参数进行设置。 CompareType 标志表示比较函数的设置为 “ 小于或等于 ”(Less than or Equal to) 或 “ 小于 ”(Less Than)。InterruptType 标志确定通过比较事件或终端计数触发中断。 CompareType 和 InterruptType 均在器件编辑器中直接通过用户模块参数进行设置，这些参数在之前相关主题部分中有所介绍。

Table 6. 输入寄存器，组 1

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 0 | 0 | 1 | 1 | 时钟 | | | |
| LSB | 使能 | | | | 时钟 | | | |

使能 (Enable) 在 16 个源中选择具有相同名称的输入信号。 器件编辑器中的用户模块 “ 使能 ”(Enable) 参数设置决定其值。 同样，用户模块 “ 时钟 ”(Clock) 参数设置决定此值。

Table 7. 输出寄存器，组 1 CY8C26/25xxx

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 0 | 0 | 0 | 0 | 0 | 输出使能 (Out Enable) | OutputSelect | |
| LSB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



[+] Feedback

Table 8. 输出寄存器，组 1 CY8C29/27/24/22/21xxx 和 CY8CLED04/08/16

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | AuxClk | | AuxEnable | AuxSelect | | OutEnable | OutputSelect | |
| LSB | AuxClk | | 0 | 0 | 0 | 0 | 0 | 0 |

器件编辑器中的用户模块 “ClockSync” 参数决定 AuxClk 位的值。 尽管命名类似，AuxEnable 和 AuxSelect 位却与 OutEnable 和 OutSelect 位字段有关。 AuxEnable 和 AuxSelect 允许将终端计数输出信号输出到其中一个行输出总线，这两个参数通过在 “ 器件编辑器互连视图 ” 中以图形方式操纵行总线来控制。 当终端计数输出被输出到某个行或全局输出总线时，将设置 OutEnable。 OutputSelect 控制哪条总线将从比较输出中输出。

Table 9. 计数寄存器 (DR0)，组 0

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 计数 (MSB) | | | | | | | |
| LSB | 计数 (LSB) | | | | | | | |

“ 计数 ”(Count) 是 PWM16 MSB 和 LSB 递减 PWM。 两者都可以使用 PWM16 API 读取。

Table 10. 周期寄存器 (DR1)，组 0

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 周期 (MSB) | | | | | | | |
| LSB | 周期 (LSB) | | | | | | | |

“ 周期 ”(Period) 保留周期值的 MSB 和 LSB，周期值根据使能或终端计数条件加载到计数器寄存器。 两者都可以在器件编辑器和 PWM16 API 中进行设置。

Table 11. 脉冲宽度寄存器 (DR2)，组 0

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 脉冲宽度 (MSB) | | | | | | | |
| LSB | 脉冲宽度 (LSB) | | | | | | | |

PulseWidth 保留用于生成比较事件的脉冲宽度值的 MSB 和 LSB。 两者都可以在器件编辑器和 PWM16 API 中进行设置。

Table 12. 控制寄存器 (CR0)，组 0

| 模块 / 位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MSB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0[1] |
| LSB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 启动 / 停止 |

若设置了 “ 启动 / 停止 ”(Start/Stop)，则表示 PWM16 处于使能状态。 使用 PWM16 API 对它进行修改。

“ 启动 / 停止 ”(Start/Stop) 由串联 PSoC 模块中的 LSB 控制寄存器控制，设置为零。



[+] Feedback

# 版本历史记录

| 版本 | 创作者 | 说明 |
|---|---|---|
| 2.5 | TDU | 更新了时钟说明，内容包括： 在模块中使用外部数字时钟时，为达到最高精度并使用睡眠操作应关闭行输入同步。 |

**Note** PSoC Designer 5.1 在所有用户模块数据表中引入了版本历史。 此部分记录了当前用户模块版本和以前用户模块版本之间区别的高级描述。



Copyright © 2000-2011 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.