16-Bit 定时器数据表 Timer16 V 1.1

Copyright © 2008-2010 Cypress Semiconductor Corporation. All Rights Reserved.

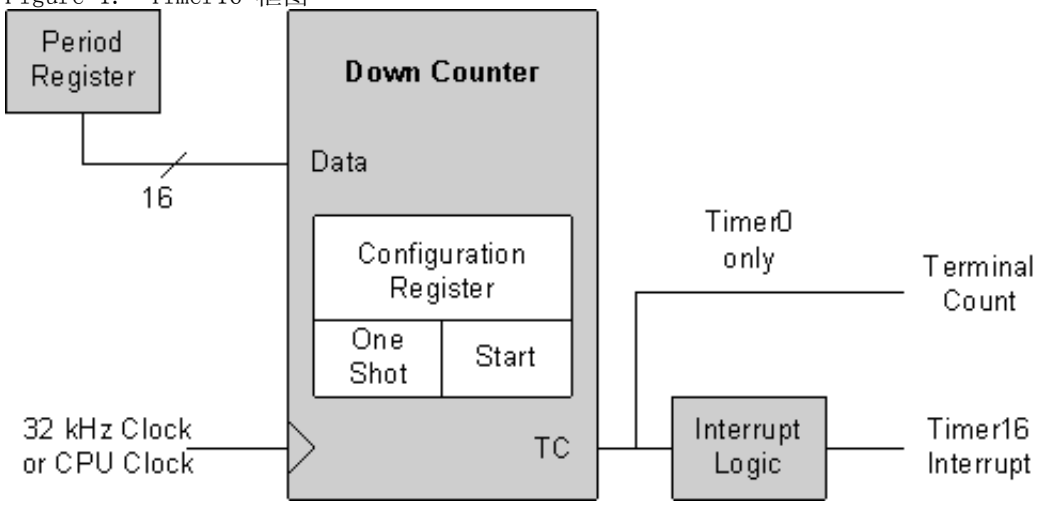
资源	PSoC® 模块				API 存储器 （字节）		每个传感器所使用的引脚数
	CapSense®	I2C/SPI	定时器	比较器	闪存	RAM	
CY8C20x66、CY8C20x36、CY8C20336AN、CY8C20436AN、CY8C20636AN、CY8C20x46、CY8C20x96、CY7C64xxx、CY7C60413、CY7C60424、CY7C6053x、CYONS2010、CYONS2011、CYONSFN2051、CYONSFN2053、CYONSFN2061、CYONSFN2151、CYONSFN2161、CYONSFN2162、CY8CTST200、CY8CTMG2xx							
			1		36	0	无

功能和概述

- 16-bit 可编程倒计时定时器
- 单触发倒计时选项，在终端计数上不重新加载周期
- 中断发生在终端计数上
- 使用内部 32 kHz 时钟或 CPU 时钟

用户模块是带有中断回调的 16-bit 可编程定时器。 它的时钟由内部 32 kHz 时钟或 CPU 时钟而定。

Figure 1. Timer16 框图



功能说明

Timer16 用户模块在一段给定的时间执行单触发倒计时 （单触发模式）或连续重复倒计时 （连续模式）。默认模式和周期值是在设备编辑器的参数部分中设置的，或者是根据应用程序编程接口 (API) 部分中所述使用 SetMode() 和 SetPeriod() 以编程方式设置。

直流和交流电气特性

Table 1. Timer16 DC 和 AC 电气特征

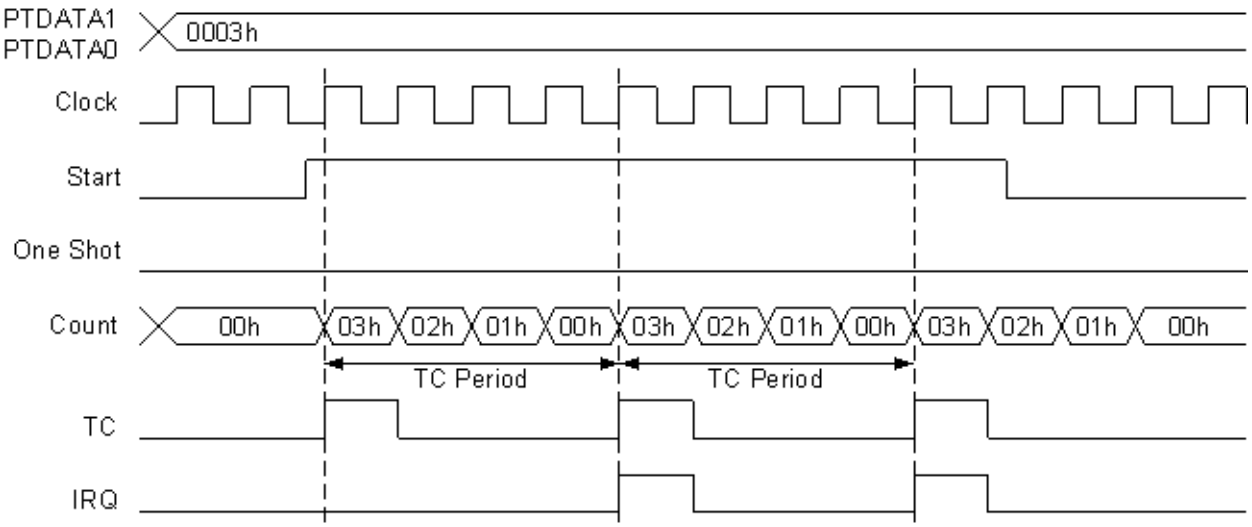
参数	条件和注释	典型值	限制	单位
F_{32K1}		32	-	kHz

定时

当启动时，可编程定时器加载其数据寄存器中包含的值，然后倒计时，直至终端计数为零。 达到终端计数时，定时器为一个时钟周期输出有效的终端计数高电平脉冲。 终端计数脉冲的低电平时间等于加载的十进制计数值乘以时钟周期。 (TCpw = 十进制计数值 * 时钟周期)。 终端计数输出的周期等于终端计数的脉冲宽度与一个时钟周期之和。 (TCperiod = TCpw + 时钟周期)。

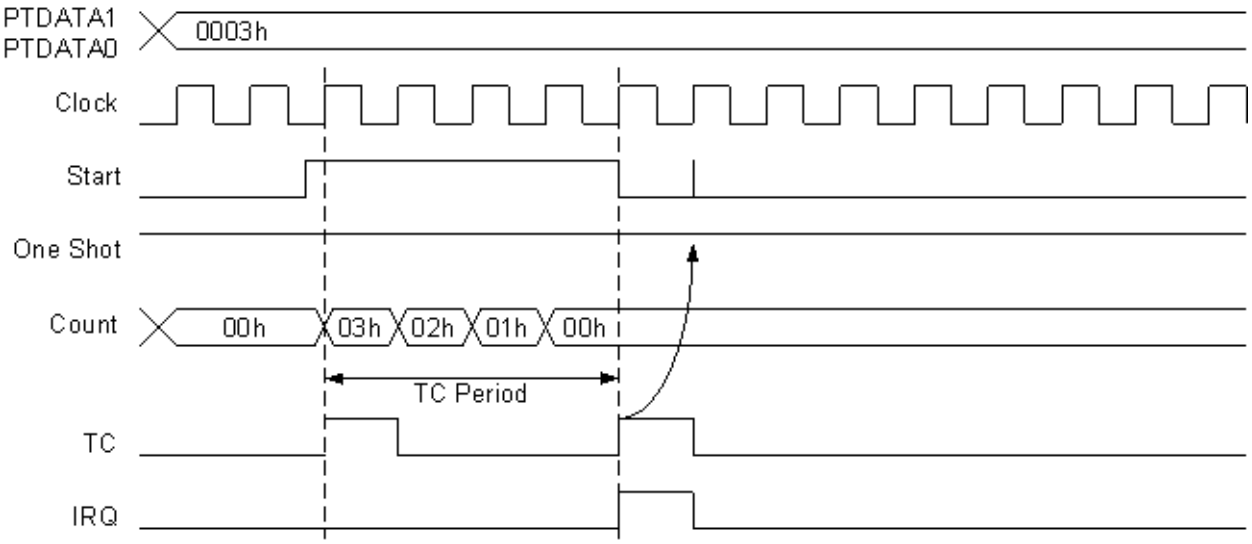
只有 TIMER0 输出其终端计数输出。 TIMER1 和 TIMER2 没有终端计数输出。

Figure 2. Timer16 连续操作



[+] Feedback

Figure 3. Timer16 单触发式操作



放置

放置 Timer16 可用 TIMER 块之一。 请注意，只有 TIMER0 会在终端计数中输出 TC 信号。 TIMER1 和 TIMER2 不这样做。

参数和资源

周期

一个从 1-65535 开始的 16-bit 值，用于确定倒计时周期。

模式

设置 Timer16 的模式。 两个选项为单触发式或连续模式。 在单触发模式中，定时器完成一个完整的计数周期，然后终止。 最终，清除此寄存器中的 START 位。 在连续模式中，定时器每次在完成其计数周期时重新加载计数值，并重复此操作。

时钟选择

选择输入时钟。 从可用时钟之一进行选择。

[+] Feedback

应用程序编程接口

应用程序编程接口 (API) 例程作为用户模块的一部分提供，从而允许以较高级别处理模块。 本节指定每个函数的接口，以及引用文件所提供的相关常量。

Note 在此，像在所有用户模块 API 中一样，可以通过调用 API 函数更改 A 和 X 寄存器的值。 如果在调用后需要 A 和 X 的值，则调用函数负责在调用前保留 A 和 X 的值。 选择此 “ 寄存器易失 ” 策略旨在提高效率，自 PSoC Designer 1.0 版起已强制使用此策略。 C 编译器自动遵循此要求。 汇编语言编程人员还必须确保其代码遵循此策略。 虽然某些用户模块 API 函数可能保持 A 和 X 不变，但是不保证它们将来也这样做。

下面是 Timer16 受支持 API 函数的列表：

Timer16_Start

说明：

通过向起始位写入一，启用 Timer16。

C 原型：

```
void  Timer16_Start(void)
```

汇编：

```
lcall  Timer16_Start
```

参数：

无

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

Timer16_Stop

说明：

通过清除起始位，禁用 Timer16。

C 原型：

```
void  Timer16_Stop(void)
```

汇编：

```
lcall  Timer16_Stop
```

参数：

无

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

Timer16_EnableInt

说明：

启用 Timer16 终端计数中断。

C 原型：

```
void Timer16_EnableInt(void)
```

汇编：

```
lcall  Timer16_EnableInt
```

参数：

无

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

Timer16_DisableInt

说明：

禁用 Timer16 终端计数中断。

C 原型：

```
void  Timer16_DisableInt(void)
```

汇编：

```
lcall  Timer16_DisableInt
```

参数：

无

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

Timer16_SetMode

说明：

通过将模式位值写入配置寄存器来设置模式。

C 原型：

```
void  Timer16_SetMode(BYTE bMode)
```

汇编：

```
mov   A, [bMode] ; place bMode in A
lcall  Timer16_SetMode
```

参数：

BYTE bMode： 单触发模式 （位值 1）或连续模式 （位值 0）。 定义要提供的内容。

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

Timer16_SetPeriod

说明：

通过将 16-bit 值写入周期寄存器来设置周期。

C 原型：

```
void  Timer16_SetPeriod(WORD wPeriod)
```

汇编：

```
mov   X, [wPeriod+0] ; place MSB in X
mov   A, [wPeriod+1] ; place LSB in A
lcall  Timer16_SetPeriod
```

参数：

wPeriod - 16-bit 周期值

返回值：

无

副作用：

此函数会更改 A 和 X 寄存器。

固件源代码示例

在下面的 C 和汇编示例代码中，启用全局中断和定时器中断以允许中断处理程序执行。 必须通过在用户代码标志之间添加 “ljmp _myTimer_ISR_Handler” 行来修改 timer16int.asm 中的 Timer16_ISR 例程。 然后调用启动例程以启动倒计时周期。 每次定时器达到终端计数 (0) 时，定时器计时单元都会递增。 对于每个 254 定时器触发，增大和降低 GPIO 引脚 Port_1_0。

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include "PSoCGPIOInt.h"

BYTE  timerTicks;

#pragma interrupt_handler myTimer_ISR_Handler;
void              myTimer_ISR_Handler( void );

void main(void)
{
    timerTicks = 0;
    M8C_EnableGInt;

    Timer16_EnableInt();
    Timer16_Start();

    while( 1 )
    {
        if (timerTicks > 254)
        {
            timerTicks = 0;
            //Port_1_0 is set to a Strong Drive
            Port_1_0_Data_ADDR |= Port_1_0_MASK;  //Raise Port_1_0
            Port_1_0_Data_ADDR &= ~Port_1_0_MASK; //Lower Port_1_0
        }
     }
}


//--------------------------------------------------------------------------
// myTimer_ISR_Handler    // The handler for the Timer ISR
//--------------------------------------------------------------------------
void myTimer_ISR_Handler(void)
{
timerTicks++;
}
```


[+] Feedback

下面是以汇编语言编写的等效代码示例：

```
include "PSoCAPI.inc"
include "m8c.inc"       ; part specific constants and macros
include "PSoCGPIOInt.inc"

area    bss     (RAM,REL)
timerTicks:                 blk     1

area text(ROM,REL)
export _main
export _myTimer_ISR_Handler

_main:
    mov   [timerTicks], 0
    M8C_EnableGInt
    lcall  _Timer16_EnableInt
    lcall  _Timer16_Start

loop:
    mov   A,254
    cmp   A,[timerTicks]
    jnc   loop
    mov   [timerTicks], 0
    ;Port_1_0 is set to a Strong Drive
    or    reg[Port_1_0_Data_ADDR],1   ;Raise Port_1_0
    and   reg[Port_1_0_Data_ADDR],254 ;Lower Port_1_0
    jmp   loop

;-----------------------------------------------------------------------
; myTimer_ISR_Handler    // The handler for the Timer ISR
;-----------------------------------------------------------------------
_myTimer_ISR_Handler:
    inc   [timerTicks]
    reti
```

配置寄存器

下面介绍用于配置此用户模块的定时器 PSoC 块寄存器：

Table 2. 块 Timer16、Timer0 配置寄存器 PTx_CFG

位	7	6	5	4	3	2	1	0
值	Reserved					CLKSEL	单触发式	START

CLKSEL – 此位确定定时器是以 32 kHz 时钟还是以 CPU 时钟运行。 如果此位设置为 1’b1，则定时器以 CPU 时钟运行，否则定时器以 32 kHz 时钟运行。

单触发式 – 此位确定定时器是在单触发模式还是在连续模式下运行。 在单触发模式中，定时器完成一个完整计数循环，然后终止。 最终，清除此寄存器中的 START 位。 在连续模式中，定时器每当完成其计数循环时会重新加载计数值，然后重复。

START – 此位从完整计数启动定时器计数。 完整计数由加载到 DATA 寄存器中的值决定。 如果定时器在单触发模式下运行，当完成完整计数循环时清除此位。

Table 3. 模块 Timer16、数据寄存器 1、 PTx_DATA1

位	7	6	5	4	3	2	1	0
值	Data							

这些位保留定时器的 16-bit 计数值的高 8 位。

Table 4. 模块 Timer16、数据寄存器 0、 PTx_DATA0

位	7	6	5	4	3	2	1	0
值	Data							

这些位保留定时器的 16-bit 计数值的低 8 位。

这些位保留定时器的 16-bit 计数值的低 8 位。

[+] Feedback

版本历史记录

版本	创作者	说明
1.1	DHA	添加的寄存器别名为 Timer16.h 和 Timer16.inc 文件。

Note PSoC Designer 5.1 在所有的用户模块数据表中提供版本历史记录。 本数据表详细介绍了当前和先前用户模块版本之间的区别。

Copyright © 2008-2010 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.