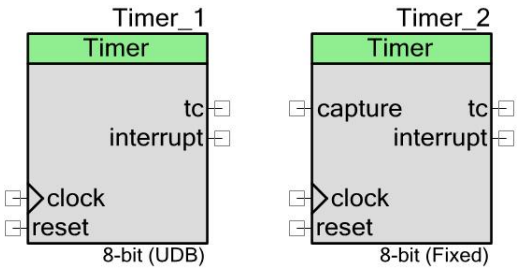
定时器
2.50

特性



- 支持 PSoC 3 和 PSoC5 LP 中的固定功能（FF - Fixed Function）实现
- 8、16、24 或 32 位定时器
- 可选捕获输入
- 使能、触发和复位输入，用于与其他组件同步
- 连续或单次触发模式

概述

定时器组件可以提供一种方法用于测量时间间隔。它不但可以实现基本的定时器功能，而且还可以提供捕获、多中断触发，DMA 触发等高级功能。

对于 PSoC 3 和 PSoC5 LP 器件，组件可以使用 FF 模块或 UDB 实现。而 PSoC 4 器件仅支持 UDB 实现。与 FF 实现相比，UDB 实现通常拥有更多特性。如果设计足够简单，可考虑使用 FF 定时器，以节省 UDB 资源用于其它用途。

下表显示了 FF 定时器与 UDB 定时器之间的主要特性差异。FF 定时器与 UDB 定时器之间还有很多功能差异，不同设备中的 FF 定时器功能也不相同。有关两者的详细差异，请参阅配置一节的时序波形。

特性	Fixed Function	UDB
位数	8或16	8、16、24或32
运行模式	连续或单次触发	连续、单次触发或中断停止单次触发
计数模式	只进行递减计数	只进行递减计数
使能输入	是（硬件或软件使能）	是（硬件或软件使能）
捕获输入	是	是
捕获模式	仅上升沿	上升沿、下降沿、任一边沿或软件控制
捕获FIFO	否（一个捕获寄存器）	是（最多可捕获四次）

特性	Fixed Function	UDB
触发输入	否	是
触发模式	无	上升沿、下降沿、任一边沿或软件控制
复位输入	是	是
终端计数输出	是	是
中断输出	是（仅PSoC 3）	是
中断条件	TC、捕获	TC、捕获和FIFO满位
捕获输出	否	是
周期寄存器	是	是
周期重新加载	是（始终在复位或TC时重新加载）	是（始终在复位或TC时重新加载）
时钟输入	仅限于时钟系统中的数字时钟	任何信号

何时使用定时器

定时器的默认用途是产生周期性事件或中断信号。但是，还有其它潜在用途：

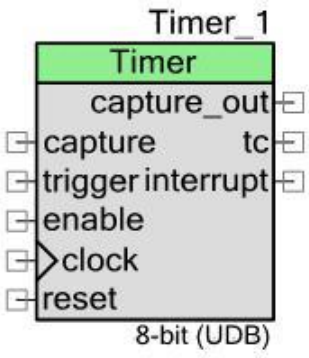
- 通过将时钟连接到时钟输入，并使用终端计数输出作为分频的时钟输出，可以创建时钟分频器。
- 通过将时钟连接到时钟输入，并将测试信号发送到使能或捕获输入，可以测量各硬件事件之间的时长。

注意：计数器组件适合用于有关计数事件的情况。而 PWM 组件则适合用于需要多个比较输出，并要求中心对齐、输出非同步停止输入和死区输出等多个控制功能的情况。

定时器通常用于记录各事件间时钟周期的数量。例如，测量转速计传感器生成的两个上升沿之间的时钟计数。更复杂的用途是测量 PWM 输入的周期和占空比。对于 PWM 测量，定时器组件设置为在上升沿时进行启动，在下一个下降沿时进行捕获，并在下一个上升沿时进行捕获和停止。最终捕获后进行中断，向 CPU 发出表示 FIFO 中所有捕获的值都已就绪的信号。

输入/输出接口

本章节介绍的是定时器上各种输入和输出连接。某些 I/O 符号在其描述中所列的特定条件下可能不显示。



注意：除非另有说明，否则所有信号都是高电平有效。

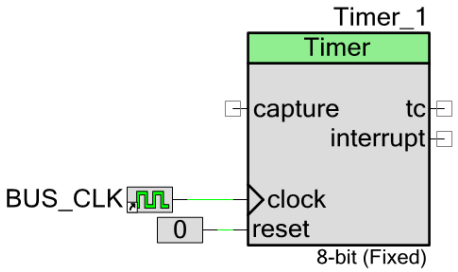
输入	是否隐藏	说明
clock	否	时钟输入定义了定时器组件的工作频率。即当使能了定时器组件时，定时器周期计数器值在此输入的上升沿被递减。
reset	否	该输入为同步复位。它需要至少一个时钟的上升沿来完成计数器和捕获计数器的复位。它将周期计数器复位为周期值，并且复位捕获计数器。对于FF定时器，复位信号强制计数器仅在活动模式从周期寄存器加载。即是否已通过调用Timer_Start()函数启用定时器。
使能	是	此输入为定时器硬件使能。此连接使周期计数器能够在时钟的每个上升沿上被递减。如果此输入处于低电平，输出仍会处于活动状态，但定时器组件不会改变状态。当**Enable Mode**（使能模式）参数设置为**Hardware Only**（仅硬件）或**Software and Hardware**（软件和硬件）时，此输入可见。
捕获	是	捕获输入将当前计数值捕获到捕获寄存器或FIFO中。如果**Capture Mode**（捕获模式）参数设置为**None**（无）之外的任何其它模式，此输入可见。捕获可能发生在上升沿、下降沿，或应用于该输入的任一沿上，这取决于**捕获模式**的设置。在时钟输入上对捕获输入进行采样。如果定时器已被禁用，则不会捕获任何值。捕获输入可以悬空，即没有任何外部连接。如果捕获输入无任何连接，则此组件将为其分配常数逻辑0。
触发	是	触发输入允许定时器根据可配置的硬件事件开始计数。如果将**Trigger Mode**（触发模式）参数设置为非**None**（无），则该输入可见。它会使定时器延迟计数，直至检测到合适的边沿为止。触发边沿既不产生捕获，也不会生成中断。

输出	是否隐藏	说明
tc	否	终端计数是同步输出，用于表示计数值等于零。此输出与定时器的时钟输入同步。此输出的定时精度取决于器件以及使用的是UDB还是FF定时器。

输出	是否隐藏	说明
interrupt	否	此中断输出由硬件中所配置的中断源驱动。对所有源进行“或”运算以创建最终输出信号。中断源可以是：终端计数、捕获或FIFO已满。 触发中断后，会持续设置中断输出，直至读取状态寄存器为止。 PSoC 5LP 上的 FF 定时器支持中断，但中断信号不能引出，即中断接口不能连接其他模块，包括中断模块。如果需要此功能，可将中断组件连接到tc信号，或使用UDB定时器。
capture_out	是	capture_out是一项指示输出，用于指示硬件捕获已触发。此信号仅适用于UDB定时器。此输出与定时器的时钟输入同步。

原理图宏信息

组件目录中的定时器是使用带默认设置的定时器组件的原理图宏。它连接到总线时钟和逻辑低电平组件。



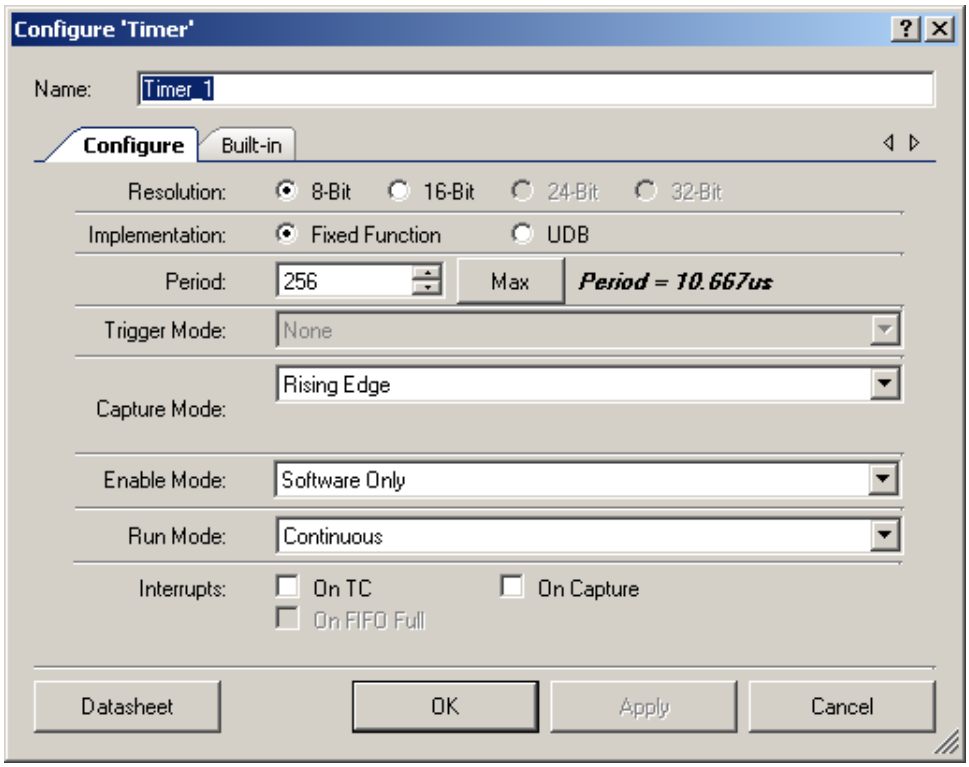
组件参数

将定时器拖入设计中，双击定时器以打开 **Configure**（配置）对话框。

硬件与软件配置选项

硬件配置选项用于更改项目合成和放置在硬件中的方式。如果您对任何这些选项进行了更改，则必须重新编译硬件。软件配置选项不影响合成或放置。如果在构建之前设置这些参数，则需要设置其初始值。可随时使用所提供的 API 修改这些初始值。下面章节中描述的大多数参数是硬件选项的。软件选项也将一同说明。

Configure 选项卡



分辨率

Resolution（分辨率）参数用于定义定时器的位宽。可根据最大计数值 255、65535、16777215 和 4294967295，分别将此值设置为 8、16、24 或 32。而对于 FF 定时器，分辨率仅限于 8 或 16 位。

实现

Implementation（实现）参数允许您选择计数器的固定功能模块实现或 UDB 实现。如果选择了 FF，UDB 函数将被禁用。

周期（软件选项）

Period（周期）参数用于定义计数器的周期。定时器组件的最大计数值（或翻转点）等于 **Period** 减去 1。**Period** 减去 1 是加载到周期寄存器中的初始值。软件可随时使用 Timer_WritePeriod() API 更改周期寄存器。要使用此 API 获取等同的结果，必须将定制器中的**周期**值减去 1 用作为此函数中的参数。

该值的限制由 **Resolution**（分辨率）参数定义。对于 8、16、24 和 32 位 **Resolution** 参数，**Period** 值分别为：2^8、2^16、2^24 和 2^32，或 256、65536、16777216 和 4294967296。

触发模式（软件选项）

Trigger Mode（触发模式）参数用于配置触发输入的实现。只有将 **Implementation**（实现）设置为 **UDB** 时，此参数才有效。

可将 **Trigger Mode**（触发模式）设置为以下任一值：

- **None**（无）（默认值） — 不执行任何触发，且触发输入引脚处于隐藏状态
- **Rising Edge**（上升沿） — 在触发输入的第一个上升沿上触发（启用）计数
- **Falling Edge**（下降沿） — 在触发输入的第一个下降沿上触发（启用）计数
- **Either Edge**（任一沿） — 在触发输入的第一个沿（上升或下降沿）上触发（启用）计数
- **Software Controlled**（软件控制） — 运行时，可通过调用 Timer_SetTriggerMode() API 将触发模式设置为上述四种触发模式之一。默认触发为 **None**（无），直至使用此 API 设置其它值。

捕获模式（软件选项）

捕获模式部分包含三个参数：**Capture Mode Value**（捕获模式值）、**Enable Capture Counter**（使能捕获计数器）和 **Capture Count**（捕获计数）。

捕获模式

Capture Mode（捕获模式）参数用于配置发生捕获的时间。在时钟输入的上升沿上对捕获输入进行采样。此模式可设置为以下任一值（对于固定功能实现，仅 **None** 和 **Rising Edge** 可用）：

- **None** — 不会实现任何捕获，且捕获输入引脚处于隐藏状态
- **Rising Edge**（上升沿） — 根据时钟输入，在捕获输入的上升沿上捕获计数器值。
- **Falling Edge**（下降沿） — 根据时钟输入，在捕获输入的下降沿上捕获计数器值。
- **Either Edge**（任一沿） — 根据时钟输入，在捕获输入的任一沿上捕获计数器值。
- **Software Controlled** — 运行时，可通过调用 Timer_SetCaptureMode() API 将捕获模式设置为上述四种捕获模式之一。默认触发为 **None**（无），直至使用此 API 设置其它值为止。

使能捕获计数器（软件选项）

Enable Capture Counter（使能捕获计数器）参数用于定义在实际捕获计数器前需要发生多少个捕获事件。例如，如果需要每隔三个事件捕获一次，那么捕获计数器值必须设置为 3。此参数仅适用于 UDB 定时器。

捕获计数（软件选项）

Capture Count（捕获计数）参数用于设置在实际捕获计数器前发生的捕获事件的初始数量。它可设置为介于 2 到 127 之间值。运行时，可通过调用 API 函数 Timer_SetCaptureCount() 修改捕捉计数值。此参数仅适用于 UDB 定时器。

使能模式

Enable Mode （使能模式）参数用于配置定时器的使能条件。在时钟输入的上升沿上对使能输入进行采样。可将该模式设置为以下任何值：

- **Software Only**（仅软件） — 仅根据控制寄存器的使能位启用定时器。
- **Hardware Only**（仅硬件） — 仅根据使能输入启用定时器。（仅 UDB）
- **Hardware and Software**（硬件和软件） — 仅当硬件和软件使能都为“true”时，才会使能计数器。

运行模式

Run Mode（运行模式）参数用于将定时器组件配置为连续运行模式或单次触发模式：

- **Continuous**（连续） — 定时器在启用后连续运行。
- **One Shot**（单次触发） — 定时器开始计数并在到达零时停止计数。复位后，它将开始另一个计数周期。停止后，对于 UDB 定时器，它会将 **Period** 重新加载到计数寄存器内；对于 FF 定时器，计数寄存器中的终端计数保持不变。
- **One Shot (Halt on Interrupt)** （单次触发（中断停止））— 定时器开始计数并在到达零时或发生中断时停止计数。复位后，它将开始另一个计数周期。停止后，对于 UDB 定时器，它会将 **Period** 重新加载到计数寄存器中；对于 FF 定时器，计数寄存器中的终端计数保持不变。

  注意：为了确保单次触发不会过早的让定时器计数，推荐使用**触发模式**控制开始时间，或使用某种形式的软件使能模式（**Software Only** 或 **Software and Hardware**）。

中断（软件选项）

Interrupt（中断） 参数用于配置初始中断源。当发生以下所选的一个或多个事件时，会生成中断。软件可随时重新配置此模式；此参数用于定义初始配置。

- **On TC**（TC 时）— 此参数始终有效；默认情况下，它被清除。
- **On Capture (1-4)** （捕获时（1-4））— 可在给定的捕获次数后生成中断；默认情况下，它被清除。
- **On FIFO Full**（FIFO 已满时）— 可在捕获 FIFO 已满后生成中断；默认情况下，它被清除。

时钟选择

固定功能组件

当被配置为使用设备中的 FF 模块时，定时组件具有以下限制：

- 时钟输入必须是时钟系统中的数字时钟。
- 如果此时钟的频率要与总线时钟的频率相同，则此时钟必须是总线时钟。

  打开相应时钟组件的 **Configure** 对话框，然后将 **Clock Type**（时钟类型）参数配置为 **Existing**（现有），并将 **Source**（源）参数配置为 **BUS_CLK**。该频率下的时钟不能是从主时钟、IMO 等任何其它源分出来的。

对于基于 UDB 的组件

对于 PSoC3/5, 任何来源的任何数字信号都可连接到时钟输入，但在 PSoC4 中只有时钟组件可以为计数器提供时钟输入。该信号的频率限于本数据手册直流电和交流电电气特性（UDB 实现） 一节中定义的频率范围。

应用编程接口

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下面的表格列出并说明了每个函数的接口。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“Timer_1”分配给指定设计中组件的第一个实例。您可以将其重命名为遵循标识符语法规则的任何唯一值。实例名称会成为每个全局函数名称、变量和符号常量的前缀。出于可读性考虑，下表中使用的实例名称为“Timer”。

函数	说明
Timer_Start()	设置initVar变量，调用Timer_Init()函数，然后调用Enable函数。
Timer_Stop()	禁用定时器。
Timer_SetInterruptMode()	启用或禁用中断输出源。
Timer_ReadStatusRegister()	返回状态寄存器的当前状态。
Timer_ReadControlRegister()	返回控制寄存器的当前状态。
Timer_WriteControlRegister()	设置控制寄存器的位域。
Timer_WriteCounter()	将新值直接写入到计数器寄存器中。（仅UDB）
Timer_ReadCounter()	强制进行捕获，然后返回捕获值。
Timer_WritePeriod()	写入周期寄存器。

函数	说明
Timer_ReadPeriod()	读取周期寄存器。
Timer_ReadCapture()	返回捕获寄存器的内容或FIFO的输出。
Timer_SetCaptureMode()	设置发生捕获的硬件或软件条件。
Timer_SetCaptureCount()	设置在将计数寄存器捕获到FIFO之前要计数的捕获事件数量。
Timer_ReadCaptureCount()	报告当前捕获事件的数量。
Timer_SoftwareCapture()	强制将计数器捕获到捕获FIFO
Timer_SetTriggerMode()	设置发生触发事件的硬件或软件条件。
Timer_EnableTrigger()	使能定时器的触发模式。
Timer_DisableTrigger()	禁用定时器的触发模式。
Timer_SetInterruptCount()	设置在触发中断之前要计数的捕获数量。
Timer_ClearFIFO()	清除捕获FIFO。
Timer_Sleep()	停止定时器并保存其当前配置。
Timer_Wakeup()	恢复定时器配置并重新使能定时器。
Timer_Init()	根据Configure（配置）对话框的设置来初始化或恢复定时器。
Timer_Enable()	使能定时器。
Timer_SaveConfig()	保存定时器的当前配置。
Timer_RestoreConfig()	恢复定时器的配置。

全局变量

变量	说明
Timer_initVar	指示定时器是否已初始化。变量将初始化为0，并在第一次调用Timer_Start()时设置为1。这样，第一次调用Timer_Start()子程序后，组件不用重新初始化即可重启。 如果需要重新初始化此组件，则可在Timer_Start()或Timer_Enable()函数之前调用Timer_Init()函数。

void Timer_Start(void)

说明： 这是开始执行组件操作的首选方法。Timer_Start()设置initVar变量，调用Timer_Init()函数，然后调用Timer_Enable()函数。

参数： 无

返回值： 无

其他影响： 如果已设置initVar变量，则该函数仅调用Timer_Enable()函数。

void Timer_Stop(void)

说明： 对于FF定时器，它将禁用定时器并关闭其电源。对于UDB定时器，定时器仅在软件使能模式中被禁用。

参数： 无

返回值： 无

其他影响： 由于固定功能的定时器的电源被此函数关闭，TC输出将被驱动至低电平。

void Timer_SetInterruptMode(uint8 interruptMode)

说明： 启用或禁用中断输出源。

参数： uint8：中断源。有关位定义，请参阅本数据手册的模式寄存器一节。

返回值： 无

其他影响： FF和UDB的位位置不同。掩码#defines用于封装这种差异。

uint8 Timer_ReadStatusRegister(void)

说明： 返回状态寄存器的当前状态。

参数： 无

返回值： uint8：状态寄存器的当前值。
有关位定义，请参阅本数据手册的状态寄存器一节。

其他影响： 读取状态寄存器时，将清除这些位中的一部分。本数据手册的状态寄存器一节中定义了“读取时清除”位。

uint8 Timer_ReadControlRegister(void)

说明： 返回控制寄存器的当前状态。在不需要控制寄存器的特殊情况中，此API不可用。（特殊情况分别为UDB定时器，使能模式为“仅硬件”，捕获模式非软件控制以及触发模式非软件控制）。

参数： 无

返回值： uint8：控制寄存器位域

有关位的定义，请参阅本数据手册中的控制寄存器一节。

其他影响： 无

void Timer_WriteControlRegister(uint8 control)

说明： 设置控制寄存器的位域。在不需要控制寄存器的特殊情况中，此API不可用。（特殊情况分别为UDB定时器，使能模式为“仅硬件”，捕获模式非软件控制以及触发模式非软件控制）。

参数： uint8：控制寄存器位域

有关位的定义，请参阅本数据手册的控制寄存器一节。

返回值： 无

其他影响： 无

void Timer_WriteCounter(uint8/16/32 counter)

说明： 将新值直接写入到计数器寄存器中。此函数仅适用于UDB定时器。

参数： uint8/16/32：新的计数器值。对于24位定时器，该参数为uint32。

返回值： 无

其他影响： 覆盖计数器值。这可能会导致周期宽度被更改以及TC输出状态不确定等问题。这不是原子写操作，且该函数可能被中断。调用此函数前需要先禁用定时器。

uint8/16/32 Timer_ReadCounter(void)

说明： 强制进行捕获，然后返回捕获值。

参数： 无

返回值： uint8/16/32：当前计数器值。对于24位定时器，返回类型为uint32。

其他影响： 返回捕获寄存器的内容或FIFO的输出（仅适用于UDB）。

void Timer_WritePeriod(uint8/16/32 period)

说明： 写入周期寄存器。

参数： uint8/16/32：新的周期值。对于24位定时器，该参数为uint32。

返回值： 无

其他影响： 定时器周期保持不变，直至从周期寄存器重载计数器为止。

uint8/16/32 Timer_ReadPeriod(void)

说明： 读取周期寄存器。

参数： 无

返回值： uint8/16/32：当前的周期值。对于24位定时器，返回类型为uint32。

其他影响： 无

uint8/16/32 Timer_ReadCapture(void)

说明： 返回捕获寄存器的内容或FIFO的输出（UDB）。

参数： 无

返回值： uint8/16/32：当前捕获值。对于24位定时器，返回类型为uint32。

其他影响： 在UDB定时器中，将从FIFO中移除此值。

void Timer_SetCaptureMode(uint8 captureMode)

说明： 设置捕获模式。此函数仅用于UDB定时器，并且仅在 **Capture Mode**（捕获模式）参数被设为 **Software Controlled**（软件控制）时才可用。

参数： uint8：所列举的捕获模式。此外，请参阅控制寄存器 一节：

```
Timer__B_TIMER__CM_NONE
Timer__B_TIMER__CM_RISINGEDGE
Timer__B_TIMER__CM_FALLINGEDGE
Timer__B_TIMER__CM_EITHEREDGE
Timer__B_TIMER__CM_SOFTWARE
```

返回值： 无

其他影响： 无

void Timer_SetCaptureCount(uint8 captureCount)

说明： 设置在执行某个捕获前要计数的捕获事件数量。此函数仅适用于UDB定时器，并且仅在Configure（配置）对话框中选定了**Enable Capture Counter**（启用捕获计数器）参数时才可用。

参数： uint8 captureCount：在将计数器值捕获至捕获FIFO之前要计数的捕获事件数量。有效范围值为2到127。

返回值： 无

其他影响： 无

uint8 Timer_ReadCaptureCount(void)

说明： 读取Timer_SetCaptureCount()函数中设置的captureCount参数的当前值。此函数仅适用于UDB定时器，并且仅在Configure（配置）对话框中选定了**Enable Capture Counter**（启用捕获计数器）参数时才可用。

参数： 无

返回值： uint8：当前捕获计数

其他影响： 无

void Timer_SoftwareCapture(void)

说明： 强制将当前计数器值的软件捕获至FIFO。此函数仅可用于UDB定时器。

参数： 无

返回值： 无

其他影响： 无

void Timer_SetTriggerMode(uint8 triggerMode)

说明： 设置触发模式。此函数仅适用于UDB定时器，并且仅在Trigger Mode（触发模式）参数设为Software Controlled（软件控制）时才可用。

参数： uint8：所列举的捕获模式。此外，请参阅控制寄存器一节。

```
Timer__B_TIMER__TM_NONE
Timer__B_TIMER__TM_RISINGEDGE
Timer__B_TIMER__TM_FALLINGEDGE
Timer__B_TIMER__TM_EITHEREDGE
Timer__B_TIMER__TM_SOFTWARE
```

返回值： 无

其他影响： 无

void Timer_EnableTrigger(void)

说明： 启用触发。此函数仅在**Trigger Mode**（触发模式）设为**Software Controlled**（软件控制）时才可用。

参数： 无

返回值： 无

其他影响： 无

void Timer_DisableTrigger(void)

说明： 禁用触发。此函数仅在**Trigger Mode**（触发模式）被设为**Software Controlled**（软件控制）时才可用。

参数： 无

返回值： 无

其他影响： 无

void Timer_SetInterruptCount(uint8 interruptCount)

说明： 设置在为InterruptOnCapture源生成中断之前要计数的捕获的数量。此函数仅在使能了InterruptOnCaptureCount时可用。

参数： uint8 interruptCount：生成捕获中断之前要计数的捕获事件的数量。有效范围值为0到3。

返回值： 无

其他影响： 无

void Timer_ClearFIFO(void)

说明： 清除捕获FIFO。此函数仅适用于UDB定时器。请参阅本数据手册的MISRA合规性中UDB FIFO一节。

参数： 无

返回值： 无

其他影响： 无

void Timer_Sleep(void)

说明： 这是组件准备进入睡眠模式时的首选子程序。Timer_Sleep()保存当前组件的状态，然后调用Timer_Stop()函数，并调用Timer_SaveConfig()以保存硬件配置。

在调用CyPmSleep()或CyPmHibernate()函数之前调用Timer_Sleep()函数。有关功耗管理函数的详细信息，请参考*《系统参考指南》*中的“PSoC Creator”一节。

参数： 无

返回值： 无

其他影响： 对于FF定时器，将在所有低功耗模式中保留所有寄存器。对于UDB定时器，将保存和恢复控制寄存器和计数器值寄存器。此外，如果调用Timer_Sleep() 时未调用 Timer_Stop()，则将存储启用状态。

void Timer_Wakeup(void)

说明： 此函数是将组件恢复到调用Timer_Sleep()时的状态的首选子程序。Timer_Wakeup()函数调用Timer_RestoreConfig()函数以恢复配置。如果在调用Timer_Sleep()函数前启用了组件，则Timer_Wakeup()函数将重新启用组件。

参数： 无

返回值： 无

其他影响： 调用Timer_Wakeup()函数前未调用Timer_Sleep()或Timer_SaveConfig()函数，则可能产生意外行为。

void Timer_Init(void)

说明： 根据自定义程序“Configure”对话框设置，初始化或恢复组件。无需调用Timer_Init()，因为Timer_Start()子程序会调用该函数，这是开始组件操作的首选方法。

参数： 无

返回值： 无

其他影响： 根据自定义程序“Configure”对话框中的内容设置所有寄存器。

void Timer_Enable(void)

说明： 激活硬件，开始组件操作。无需调用Timer_Enable()，因为Timer_Start()子程序会调用该函数，这是开始组件操作的首选方法。此函数针对任一软件控制使能模式启用定时器。

参数： 无

返回值： 无

其他影响： 如果**Enable Mode** （使能模式）参数被设置为**Hardware Only**（仅硬件），则此函数不对定时器的操作产生任何影响。

void Timer_SaveConfig(void)

说明： 此函数会保存组件配置以及非保留寄存器。它还保存Configure（配置）对话框中定义的或通过相应API修改的当前器件参数值。此函数由Timer_Sleep()函数调用。

参数： 无

返回值： 无

其他影响： 无

void Timer_RestoreConfig(void)

说明： 此函数会恢复组件配置以及非保留寄存器。它还将组件参数值恢复为在调用Timer_Sleep()函数之前的值。

参数： 无

返回值： 无

其他影响： 调用此函数前未调用Timer_Sleep()或Timer_SaveConfig()函数可能会产生意外行为。

示例固件源代码

PSoC Creator 在“Find Example Project”对话框中提供了包括原理图和代码示例的许多示例项目。要查看特定组件实例，请打开“Component Catalog”中的对话框或者原理图中的组件样例。要查看通用示例，请打开 Start Page 或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》中主题为“查找示例项目”中的内容。

MISRA 合规性

本节介绍了MISRA-C:2004合规性和本组件的偏差情况。定义了两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节介绍了有关组件特定偏差的信息。*《系统参考指南》* 的“MISRA 合规性”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

此定时器组件没有任何特定偏差。

功能说明

如上文所述，可配置定时器组件用于多种用途。本节将更详细地介绍这些配置。

常规操作

在时钟输入的各个上升沿上，定时器组件始终进行倒计时。在计数器达到零值后的下一个时钟边沿上，定时器组件将从周期寄存器中重新加载计数器寄存器。

定时器保持禁用状态，直至由硬件或软件启用，这取决于配置设置。用户不可使用此组件，直至调用 Timer_Start()，因为组件需要该函数来配置。

定时器输出

可监控并重新加载计数器寄存器。tc 输出可用于监控计数器寄存器的当前值；当计数器为零时它为高电平。

定时器输入

可以在硬件或固件中执行捕获操作。计数寄存器中的当前值被复制到一个捕获寄存器或 FIFO 中。这样，固件稍后可读取此捕获值。

复位和启用功能可用于使定时器组件与其它组件保持同步。定时器组件仅在已启用且复位后未保留时进行计数。也可针对触发输入事件启动计数。可由硬件或固件复位或启用。所有的触发都是硬件触发的。

注意：FF 定时器实现（捕获、复位和启用）的所有输入都将在 FF 定时器中进行双同步。同步器以 BUS_CLK 速度运行。这导致应用这些信号的时间与这些信号生效的时间之间出现延迟。延迟取决于 BUS_CLK 与运行定时器的时钟的比率。针对 FF 定时器显示的所有波形都将同步后的信号。

定时器中断

中断输出可用来与 CPU 或其他组件进行关于事件发生的通信。可针对一个或多个事件的组合将中断设为活动状态。应仔细设计中断处理程序，以便确定中断源以及中断是边沿敏感型还是电平敏感型，并清除中断源。

定时器寄存器

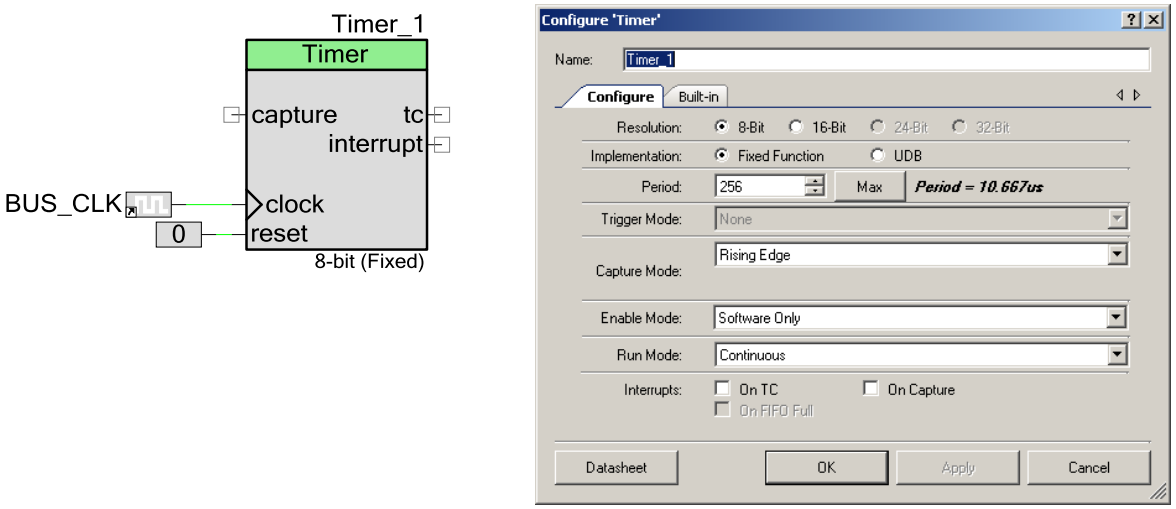
有三种寄存器：模式寄存器、状态寄存器和控制寄存器。请参考寄存器一节的内容。

配置

默认配置

将定时器组件拖动到 PSoC Creator 原理图上时，默认配置为 8 位 FF 定时器，此定时器在时钟输入的上升沿上递减计数器寄存器。图 1 显示的是默认原理图宏和 Configure（配置）对话框的设置。

图 1. 默认定时器配置



此定时器的实际功能因不同实现方法和芯片而异。下面各图显示的是此定时器在 UDB 实现中的功能，以及它在不同芯片上 FF 实现中的功能。

图 2 中显示的是 UDB 定时器的默认配置的功能。

在定时器配置期间预加载计数器，并在每次计数器达到零时都将重新加载计数器。在默认配置中，**Period** 被设为 256。这将导致将 0xFF 加载到计数器中，因为从 0xFF 计数至 0 会产生 256 的周期。

复位信号强制计数器从其周期寄存器中重新加载。此计数器保持在此状态，直至移除复位信号。

终端计数表示此定时器已倒计时至零。它在时钟循环上仍处于活动状态，并在计数值达到零后遵循此时钟循环。终端计数信号并非是基于复位事件而生成的。

默认情况下，捕获功能配置为在捕获输入的每个上升沿上进行捕获。无论捕获脉冲宽度如何，都将捕获单个值。在此例中，将捕获 0xFE 和 0x01 值，这些值可由 CPU 读取。

图 2. 默认 UDB 定时器实现的示例波形

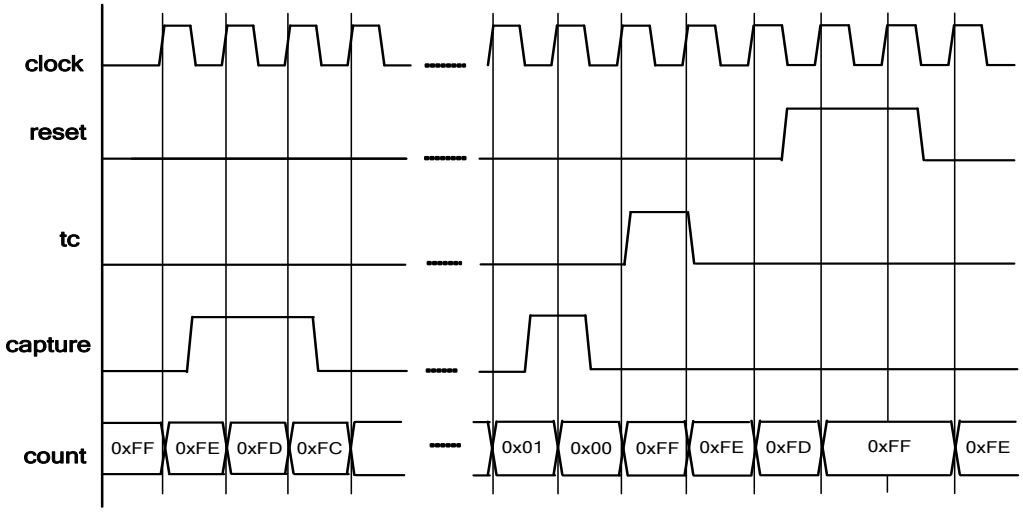


图 3 中显示的是用于 PSoC 3 FF 定时器的默认配置的功能。

对于 FF 定时器，配置期间将不预加载计数器值，计数器将从零值开始。对于 PSoC 3，这将导致 FF 定时器与 UDB 定时器出现三个时钟周期的初始延迟时间。定时器开始计数之前有两个时钟周期的延迟，并用一个周期从周期寄存器中加载定时器。定时器运行之后，周期与 UDB 定时器相同。

复位信号强制计数器从周期寄存器中进行加载，并保留该计数直至移除复位。一旦移除了复位，在计数器开始倒计时之前有两个时钟周期的延迟。

终端计数表示此定时器已倒计时至零。它在时钟循环上仍处于活动状态，并在计数值达到零后遵循此时钟循环。终端计数信号并非是基于复位事件或因初始计数器值为零而生成的。

默认情况下，捕获功能配置为在捕获输入的每个上升沿上进行捕获。无论捕获脉冲宽度如何，都将捕获单个值。在此例中，将捕获 0xFF 和 0x01 值，这些值可由 CPU 读取。此功能与 UDB 定时器相同。

图 3. 默认 PSoC 3 FF 定时器实现示例波形

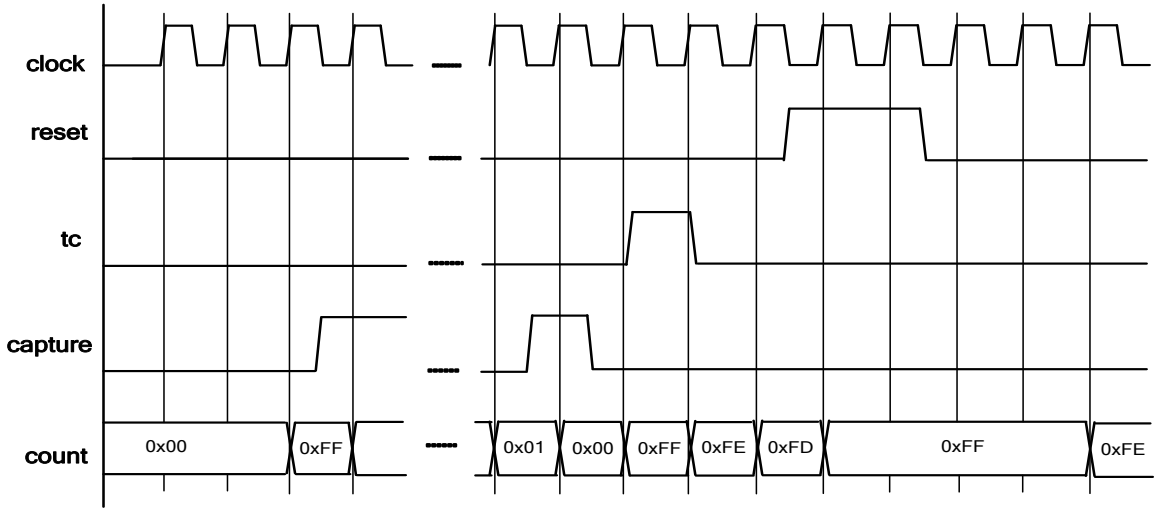


图 4 中显示用于 PSoC5 LP 上固定功能实现的默认配置的功能。

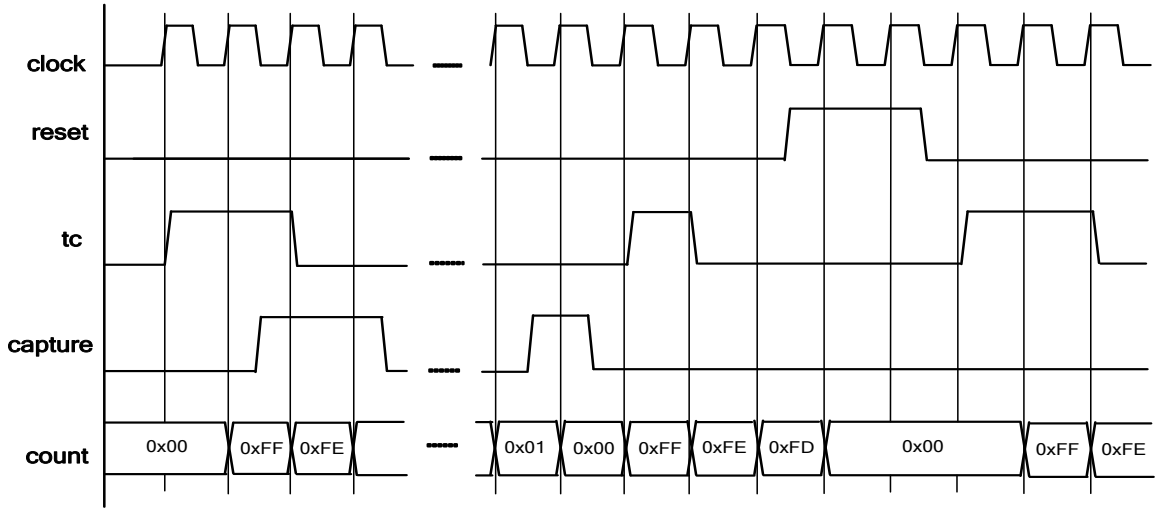
对于固定功能实现，配置期间将不预加载计数器值，计数器将从零值开始。对于 PSoC5 LP，这将导致 FF 定时器与 UDB 定时器出现两个时钟周期的初始延迟时间。定时器开始计数之前有一个时钟周期的延迟，并用一个时钟周期从周期寄存器中加载定时器。定时器运行之后，周期与 UDB 定时器相同。

复位信号强制计数器清除，并保留在零值状态，直至移除复位。复位后的功能与初始状态的功能相同，第一个周期比 UDB 定时器长两个时钟周期。

终端计数表示此定时器值为零。与计数器的初始值和复位时的值组合时，将导致初始化时和复位后都有两个时钟周期的 TC 脉冲。复位为活动状态时 TC 保持低电平，但在移除复位后会变为高电平两个时钟周期。

默认情况下，捕获功能配置为在捕获输入的每个上升沿上进行捕获。无论捕获脉冲宽度如何，都将捕获单个值。在此例中，将捕获 0xFF 和 0x01 值，这些值可由 CPU 读取。此功能与 UDB 定时器相同。

图 4. 默认 PSoC5 LP FF 定时器实现示例波形



软件和硬件使能配置

硬件使能的功能因特定实现而异。图 5 中显示为具备 UDB 定时器的软件和硬件使能配置的定时器的功能。

启用定时器后，计数器在每个循环中递减。在从周期寄存器中重新加载计数器的循环内，将生成单个时钟周期终端计数脉冲。TC 信号将始终是单个时钟周期脉冲。注意：它在重新加载周期内出现。如果重新加载因计数器到达零计数时被禁用而延迟，TC 脉冲也会延迟，直至重新启用计数器并重新加载计数器。如果因复位信号而强制计数器重新加载，则不生成 TC 脉冲。

图 5. 软件和硬件使能 UDB 定时器实现示例波形

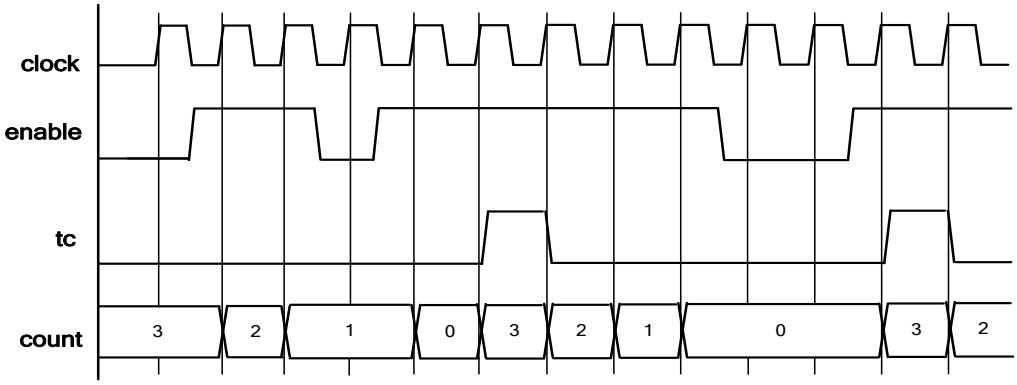


图 6 中显示为具备
PSoC 3 FF 定时器的软件和硬件使能配置的定时器的功能。

硬件使能与计数器的有效启用之间有两个时钟周期的延迟。结果是，如果两个时钟循环之前的使能信号为高电平，则计数器将递减。启用和禁用计数器时都会出现此延迟。在从周期寄存器中重新加载计数器的循环内，将生成单周期计数脉冲。TC 信号始终是单个时钟脉冲。

注意：在计数器到达零值前的两个时钟周期内，如果定时器的使能信号为低电平，则不会为此定时器周期生成 TC 输出脉冲。当重新启用定时器时，将重新加载此定时器，而不会生成 TC 信号。将在示例波形中显示此操作。

图 6. 软件和硬件使能 PSoC 3 FF 定时器实现示例波形

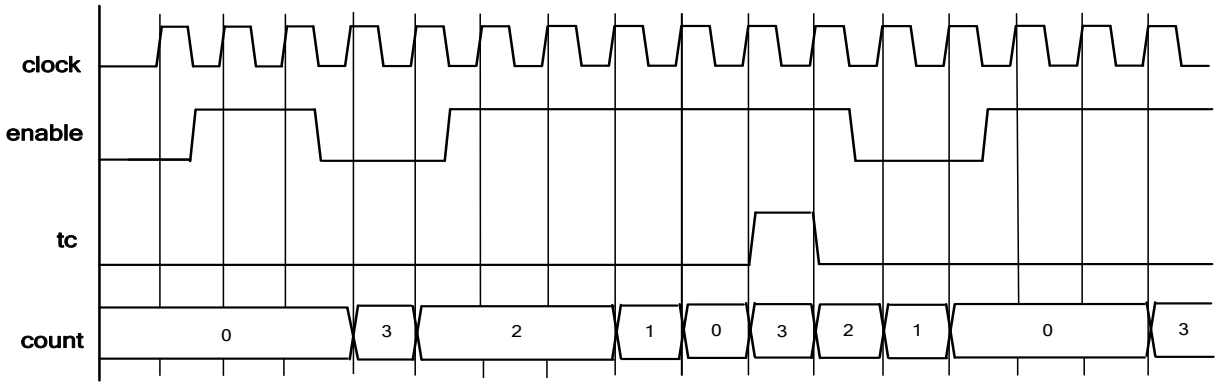
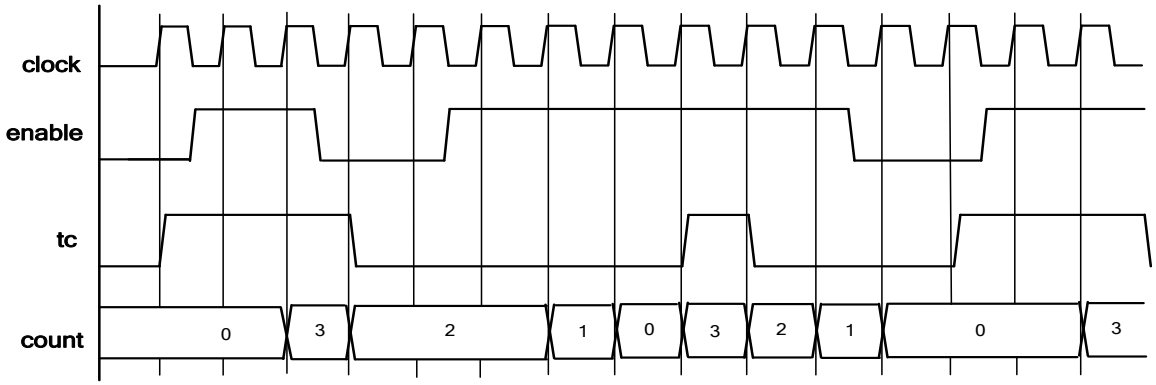


图 7 中显示为具备 PSoC5 LP FF 定时器的软件和硬件使能配置的定时器的功能。

硬件使能与计数器的有效启用之间有一个时钟循周期的延迟。结果是，如果一个时钟周期之前的使能信号为高电平，则计数器将递减。启用和禁用计数器时都会出现此延迟。当计数器值等于零时将生成终端计数信号，并有一个时钟周期的延迟。此情况发生在初始配置时。当计数器等于零时，如果使能信号导致定时器停止，则 TC 信号保持高电平。

注意：如果使能脉冲保持非活动状态一个循环，硬件使能信号不能正常工作。单周期禁用脉冲在该计数时锁定定时器，直至定时器再次禁用然后重新启用。因此，硬件禁用始终必须需要两个或更多的时钟周期。单时钟周期使能将正常工作。

图 7. 软件和硬件使能 PSoC5 LP FF 定时器实现示例波形



单次触发配置

单次触发模式的功能因特定实现而异。图 8 中显示为具备 UDB 定时器的单次触发配置的定时器的功能。

硬件使能与计数器的有效启用之间有一个时钟循周期的延迟。结果是，如果一个时钟周期之前的使能信号为高电平，则计数器将递减。启用和禁用计数器时都会出现此延迟。这与连续运行模式下的行为不同，连续运行模式计数时无延迟。

TC 信号始终是单个时钟脉冲。注意：它在重新加载周期内出现。如果重新加载因计数器到达零计数时被禁用而延迟，TC 脉冲也会延迟，直至重新启用计数器并重新加载计数器。如果因复位信号而强制计数器重新加载，则不生成 TC 脉冲。

单次触发周期完成后，可使用硬件复位设置定时器以运行另一个周期。硬件复位从周期寄存器中重新加载计数器。移除复位后的一个周期内，将在硬件使能处于活动状态时启用定时器以再次倒计时。

图 8. 单次触发式操作 UDB 定时器实现示例波形

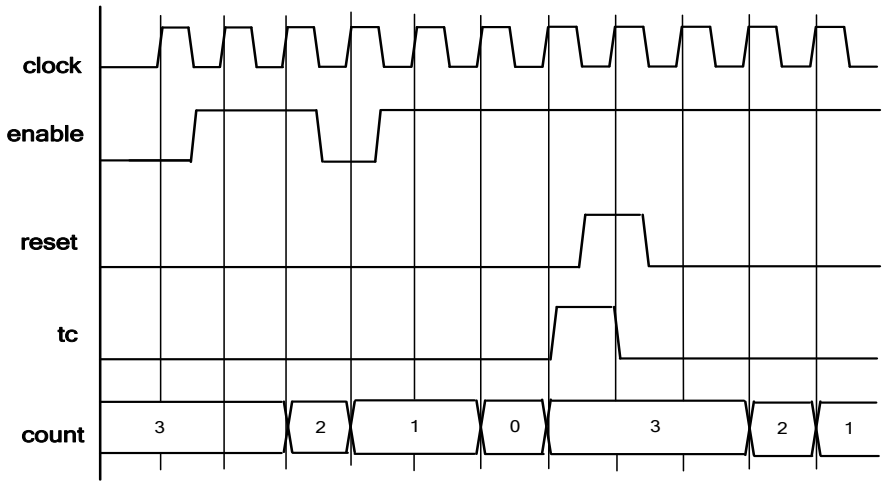


图 9 中显示为具备 PSoC 3 FF 定时器的单次触发式操作配置的定时器的功能。

硬件使能与计数器的有效启用之间有两个时钟周期的延迟。结果是，如果两个时钟循环之前的使能信号为高电平，则计数器将递减。启用和禁用计数器时都会出现此延迟。在从周期寄存器中重新加载计数器的循环内，将生成单周期计数脉冲。TC 信号始终是单个时钟脉冲。这与连续运行模式中的操作完全一致。

单次触发模式仅适用于此实现的额外功能是，一旦定时器开始计数，使能信号首次变为低电平时将在该值停止计数器。要重新开始计数，定时器必须复位。

单次触发周期完成或已因使能信号禁用而停止后，可使用硬件复位设置定时器以运行另一个周期。硬件复位从周期寄存器中重新加载计数器。释放复位后有两个循环的延迟，直至启用定时器再次倒计时。

注意：对于此实现，仅可使用 Timer_Stop() API 后跟 Timer_Start() API 以重新启动定时器。这样使计数器能够继续计数，但不会重新加载计数器值，因此此方法仅适用于计数器已完成一个周期并已重新加载的情况。

图 9. 单次触发式操作 FF PSoC 3 定时器实现示例波形

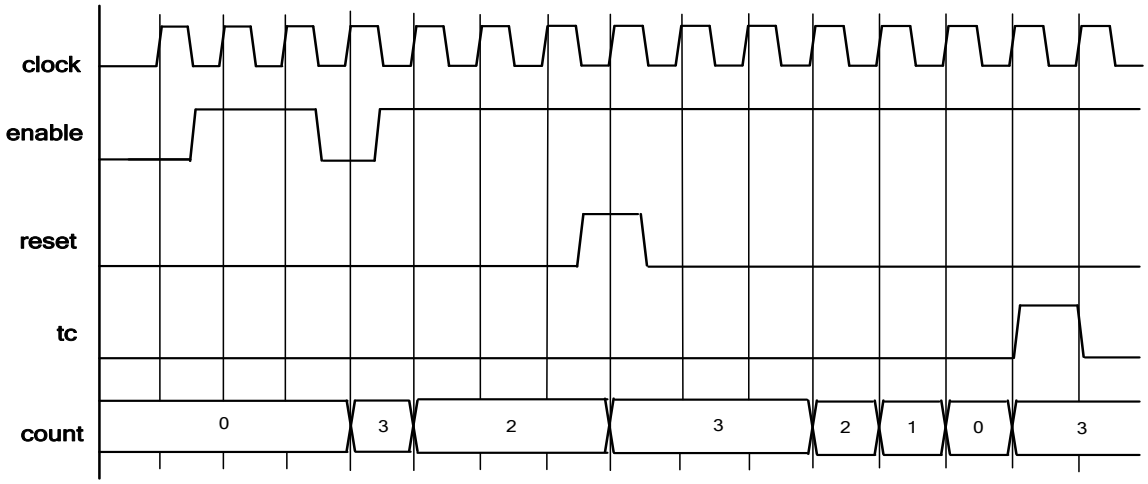


图 10 中显示为具备 PSoC5 LP 定时器的单次触发式操作配置的定时器的功能。

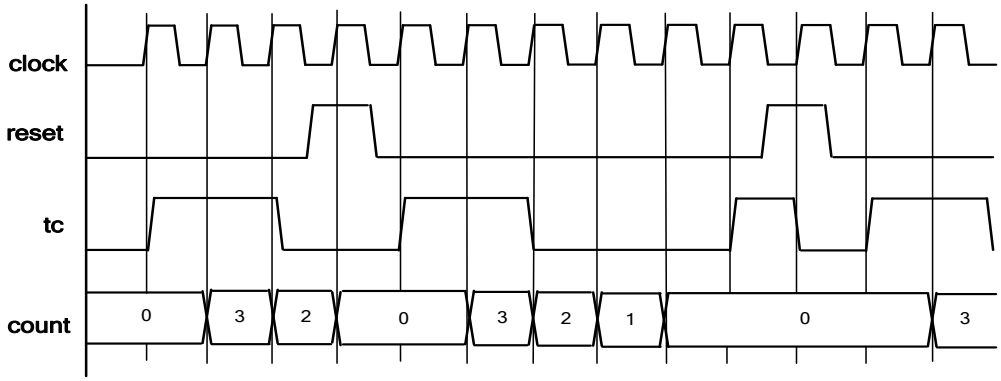
当计数器值等于零时将生成终端计数信号，并有一个时钟周期的延迟。此情况发生在初始配置时。完成一个单次触发周期后，TC 信号将保持高电平，因为计数器值将保持在零值。计数器为零值时生成 TC 信号的一个例外情况是，当复位信号为活动状态时，TC 始终保持在零值。

单次触发周期完成后，可使用硬件复位设置定时器以运行另一个周期。硬件复位重新加载计数器（零值），并配置定时器以重新运行。释放复位后有一个时钟周期的延迟，直至启用定时器再次倒计时。

注意：PSoC5 LP FF 配置不支持使用硬件使能的单次触发模式。

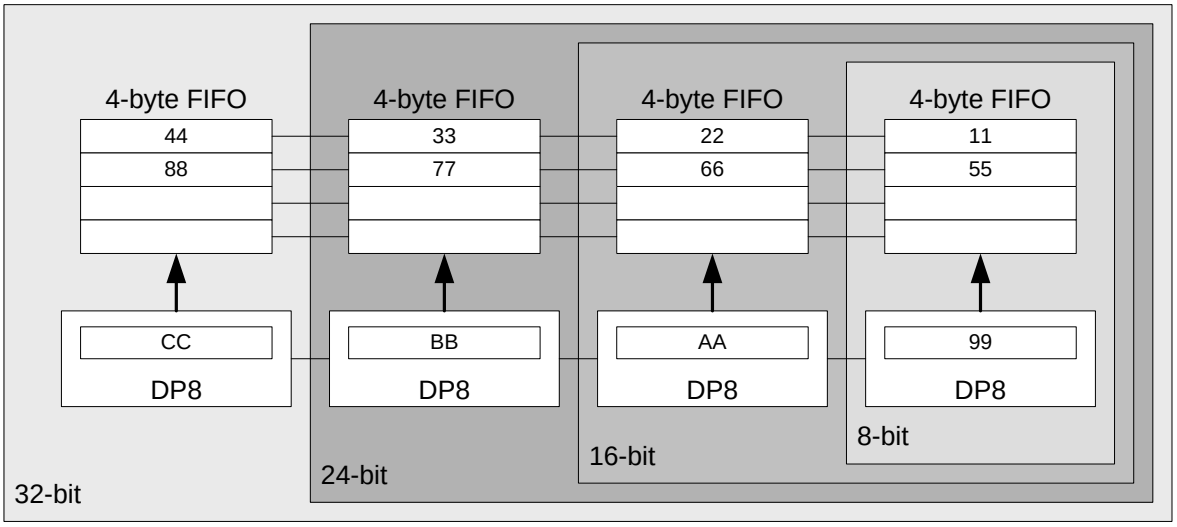
注意：由于当复位信号处于活动状态时 TC 信号保持低电平，因此每次单次触发运行完成时都会生成两个 TC 脉冲。第一个脉冲是在计数器计数至零时生成的。第二个脉冲是在移除复位后且计数器开始计数前生成的。将在示例波形中显示此操作。

图 10. 单次触发式操作 FF PSoC5 LP 定时器实现示例波形



UDB FIFO

UDB 数据路径 FIFO 用于捕获计数器值。每个 FIFO 的深度为四个字节。对于多字节配置，将在相关 UDB 的 FIFO 中同步捕获计数器的各个字节。因此，在 CPU 必须读取捕获寄存器以避免丢失数据之前，可以完成最多四次捕获。



Capture Value #1 = 0x44332211
Capture Value #2 = 0x88776655
Accumulator = 0xCCBBAA99

寄存器

状态寄存器

状态寄存器是只读寄存器，包含为定时器定义的各种状态位。使用 Timer_ReadStatusRegister() 函数读取状态寄存器值。所有在状态寄存器上的操作必须使用以下针对位域的定义，因为 FF 实现和 UDB 实现的位域可能不同。

状态寄存器中的一些位是粘滞的，意味着当它们设为 1 时，它们将保持该状态，直至它们在寄存器读取时被清除。状态数据在定时器的输入时钟边沿处寄存，使得所有粘滞位具有了定时器的时序分辨率。所有非粘滞位都是透明的，可以直接从输入中读取到状态寄存器内。

Timer_Status（UDB 定时器）

位	7	6	5	4	3	2	1	0
名称	RSVD	RSVD	RSVD	RSVD	FIFO非空	FIFO已满	捕获	TC
粘滞	N/A	N/A	N/A	N/A	FALSE	FALSE	TRUE	TRUE

Timer_Status（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	TC	捕获	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
粘滞	TRUE	TRUE	N/A	N/A	N/A	N/A	N/A	N/A

位名	头文件中的#define	说明
TC	Timer_STATUS_TC	当计数器值等于零时，此位变为1。
捕获	Timer_STATUS_CAPTURE	当触发有效捕获事件时，此位变为1。但不包括软件捕获。
FIFO已满	Timer_STATUS_FIFOFULL	当UDB FIFO达到被四个条目定义的已满状态时，此位就变为1。
FIFO非空	Timer_STATUS_FIFONEMP	当UDB FIFO包含至少一个条目时，此位就变为1。

模式寄存器

模式寄存器是一个读/写寄存器，它包含为计数器定义的中断屏蔽位。使用 Timer_SetInterruptMode()函数设置模式位。所有在模式寄存器上的操作必须使用以下针对位域的定义，因为 FF 实现和 UDB 实现的位域可能不同。

定时器组件中断输入是所有中断源的 OR 函数。每个源都可以通过模式寄存器中相应的位来启用或屏蔽。

Timer_Mode（UDB 定时器）

位	7	6	5	4	3	2	1	0
名称	RSVD	RSVD	RSVD	RSVD	RSVD	FIFO已满	捕获	TC

Timer_Mode（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	RSVD	RSVD	RSVD	RSVD	TC	捕获	RSVD	RSVD

位名	头文件中的#define	使能中断输出
TC	Timer_STATUS_TC_INT_MASK	计数寄存器等于0
捕获	Timer_STATUS_CAPTURE_INT_MASK	捕获
FIFO已满	Timer_STATUS_FIFOFULL_INT_MASK	UDB FIFO已满

控制寄存器

控制寄存器用于控制计数器的常规操作。此寄存器写入时需要调用 Counter_WriteControlRegister()函数，并使用 Counter_ReadControlRegister()函数读取。在控制寄存器上执行的所有操作都必须使用以下针对位域的定义，因为这些位域对于 FF 定时器和 UDB 定时器可能各有不同。

注意：写入到控制寄存器时，绝不能更改任何保留位。所有操作必须读取-修改-写入，并且屏蔽保留位。

Timer_Control（UDB 定时器）

位	7	6	5	4	3	2	1	0
名称	使能	捕获模式 [1:0]		触发使能	触发模式 [1:0]		中断计数 [1:0]	

Timer_Control1（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	使能

位名	头文件中的#define	说明/枚举类型
中断计数	Timer_CTRL_INTCNT_MASK	中断计数位定义开始中断前要计数的捕获事件数。
触发模式	Timer_CTRL_TRIG_MODE_MASK	触发模式控制位定义预期的触发输入功能。此位域在初始化时进行配置，并在**Trigger Mode**（触发模式）参数中定义触发模式。 ▪ Timer__B_TIMER__TM_NONE ▪ Timer__B_TIMER__TM_RISINGEDGE ▪ Timer__B_TIMER__TM_FALLINGEDGE ▪ Timer__B_TIMER__TM_EITHEREDGE ▪ Timer__B_TIMER__TM_SOFTWARE
触发使能	Timer_CTRL_TRIG_EN	触发启用位可实现对何时准备触发事件的软件控制。
捕获模式	Timer_CTRL_CAP_MODE_MASK	捕获模式控制位是两位域，用于定义预期的捕获输入操作。此位域是在初始化时使用**Capture Mode**参数中所定义的捕获模式下配置的。 ▪ Timer__B_TIMER__CM_NONE ▪ Timer__B_TIMER__CM_RISINGEDGE ▪ Timer__B_TIMER__CM_FALLINGEDGE ▪ Timer__B_TIMER__CM_EITHEREDGE ▪ Timer__B_TIMER__CM_SOFTWARE
使能	Timer_CTRL_ENABLE	在软件控制下启用计数。此位仅在**Enable Mode**（使能模式）参数设为**Software Only**（仅软件）或**Software and Hardware**（软件和硬件）时才有效。

计数器（8、16、24 或 32 位，根据分辨率）

计数寄存器包含当前的计数器值。此寄存器递减，以响应所有时钟输入的上升沿。可通过调用 Timer_ReadCounter()函数随时读取此寄存器。

捕获（8、16、24 或 32 位，根据分辨率）

捕获寄存器包含被捕获的计数器值。任何捕获事件都会将计数寄存器值复制到该寄存器中。在 UDB 实现中，该寄存器实际上是一个 FIFO。请参考 UDB FIFO 一节中的内容，了解更详细的信息。

周期（8、16、24 或 32 位，根据分辨率）

周期寄存器包含用户通过 Timer_WritePeriod() 函数设置的周期值和 **Period**（周期）参数在初始化期间定义的周期值。发生重载事件时，该周期寄存器会被复制到计数寄存器中。

组件调试窗口

定时器组件支持 PSoC Creator 组件调试窗口。调试窗口中显示了以下各寄存器。某些寄存器可用于 UDB 定时器（用*表示），某些寄存器仅可用于固定功能实现（用**表示）。所有其它寄存器都适用于任一配置。

寄存器： Timer_CONTROL

名称： 控制寄存器

说明： 有关位域定义，请参阅此数据手册中早先的Timer_Control寄存器说明。

寄存器： Timer_CONTROL2 **

名称： 固定功能控制寄存器#2

说明： FF定时器模块具有第二个配置寄存器。有关位域定义，请参阅技术参考手册。

寄存器： Timer_STATUS_MASK *

名称： 状态寄存器中断掩码配置

说明： 允许您在组件的中断输出引脚上启用任意状态位以作为中断源使用。有关位域定义的一对一关联，请参阅此数据手册中早先的Timer_Status寄存器说明。

寄存器： Timer_STATUS_AUX_CTRL *

名称： 状态寄存器的辅助控制寄存器

说明： 允许您通过位域INT_EN启用内部状态寄存器的中断输出。有关位域定义，请参阅技术参考手册。

寄存器： Timer_PERIOD

名称： 定时器周期寄存器

说明： 定义定时器各个循环开始时重新加载到周期计数器中的周期值。

寄存器： Timer_COUNTER

名称： 定时器计数器寄存器

说明： 表示当前定时器当前计数器值（在时钟循环中，从**周期**降至零）。

寄存器： Timer_GLOBAL_ENABLE **

名称： FF定时器全局使能寄存器

说明： 启用FF定时器以便操作。有关位域定义，请参阅技术参考手册。

资源

定时器组件基于 **Implementation**（实现）参数放置于设备中。如果此参数设为 **Fixed Function**（固定功能），则此组件使用 FF 计数器/定时器模块。如果此参数设为 **UDB**，则此组件使用以下资源。

配置[1]	资源类型					
	数据路径单元	宏单元	状态单元	控制单元	DMA通道	中断
8位UDB定时器触发模式 = 上升沿	1	6	1	1	–	–
16位UDB定时器触发模式 = 上升沿	2	6	1	1	–	–
24位UDB定时器触发模式 = 上升沿	3	6	1	1	–	–
32位UDB定时器触发模式 = 上升沿	4	6	1	1	–	–
8位UDB定时器单次触发模式 = 上升沿	1	9	1	1	–	–

1. 对于所有配置，通用设置为：Enable mode（使能模式）= Software only（仅软件），Capture mode（捕获模式）= None（无），Interrupt（中断）= On TC（TC 上）

16位UDB定时器单次触发模式 = 上升沿	2	9	1	1	–	–

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况不同，组件的存储器大小也不一样。下表提供了组件配置中所有 API 占用的存储器大小。

通过使用“释放”模式下的相应编译器，可以进行测量操作。在该模式下，存储器的大小得到优化。对于特定设计，可以分析编译器生成的映射文件，从而确定存储器的使用大小。

配置[2]	PSoC 3 （Keil_PK51）		PSoC 4（GCC）		PSoC5 LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
8位UDB定时器	255	5	396	5	444	5
8位FF定时器	234	2	N/A	N/A	366	5
16位UDB定时器	299	6	396	5	444	9
16位FF定时器	265	2	N/A	N/A	380	5
24位UDB定时器	287	8	412	5	452	13
32位UDB定时器	287	8	396	5	444	13
8位UDB定时器单次触发	255	5	396	5	444	5
16位UDB定时器单次触发	299	6	396	5	444	9

PSoC 3 的直流和交流电气特性（FF 定时器）

除非另有说明，否则这些规范的适用条件是：-40 °C ≤ TA ≤ 85 °C 且 TJ ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流电特性

参数	说明	条件	最小值	典型值	最大值	单位
	16位定时器模块的电流消耗	输入时钟频率 — 3 MHz	–	15	–	µA

2.对于所有配置，通用设置为：Enable mode（使能模式）= Software only（仅软件），Capture mode（捕获模式）= None（无），Interrupt（中断）= On TC（TC 上）

参数	说明	条件	最小值	典型值	最大值	单位
		输入时钟频率 — 12 MHz	–	60	–	µA
		输入时钟频率 — 48 MHz	–	260	–	µA
		输入时钟频率 — 67 MHz	–	350	–	µA

交流电特性

参数	说明	条件	最小值	典型值	最大值	单位
	工作频率		DC	–	67.01	MHz
	捕获脉冲宽度（内部）		15	–	–	ns
	捕获脉冲宽度（外部）		30	–	–	ns
	定时器分辨率		15	–	–	ns
	使能脉冲宽度		15	–	–	ns
	使能脉冲宽度（外部）		30	–	–	ns
	复位脉冲宽度		15	–	–	ns
	复位脉冲宽度（外部）		30	–	–	ns

PSoC5 LP 的直流和交流电气特性（FF 定时器）

除非另有说明，否则这些规范的适用条件是：-40°C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 2.7 V 到 5.5 V。

直流电特性

参数	说明	条件	最小值	典型值	最大值	单位
	16位定时器模块电流消耗	输入时钟频率 — 3 MHz	–	65	–	µA
		输入时钟频率 — 12 MHz	–	170	–	µA
		输入时钟频率 — 48 MHz	–	650	–	µA
		输入时钟频率 — 67 MHz	–	900	–	µA

交流电特性

参数	说明	条件	最小值	典型值	最大值	单位
	工作频率		DC	–	67.01	MHz
	捕获脉冲宽度（内部）		13	–	–	ns
	捕获脉冲宽度（外部）		30	–	–	ns
	定时器分辨率		13	–	–	ns
	使能脉冲宽度		13	–	–	ns
	使能脉冲宽度（外部）		30	–	–	ns
	复位脉冲宽度		13	–	–	ns
	复位脉冲宽度（外部）		30	–	–	ns

直流电和交流电电气特性（UDB 实现）

除非另有说明，否则这些规范的适用条件是：-40°C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流电特性

参数	说明[1]	最小值	典型值[2]	最大值	单位
I_{DD}	组件电流消耗				
	8位UDB，连续，触发 = 无	–	6	–	µA/MHz
	8位UDB，单次触发，触发 = 无	–	5	–	µA/MHz
	16位UDB，连续，触发 = 上升沿	–	8	–	µA/MHz
	16位UDB，单次触发，触发 = 上升沿	–	8	–	µA/MHz
	24位UDB，连续，触发 = 任一沿	–	10	–	µA/MHz
	32位UDB，连续，触发 = 软件控制	–	13	–	µA/MHz

1. 对于所有配置，通用设置为：Enable mode（使能模式）= Software only（仅软件），Capture mode（捕获模式）= None（无），Interrupt（中断）= On TC（TC 上）

2. 未包括器件的 IO 和时钟分配的电流。这些值是在温度是 25 °C 时的值。

交流电特性

参数	说明[3]	最小值	典型值	最大值[4]	单位
f_{CLOCK}	组件时钟频率				
	8位UDB，连续，触发 = 无	–	–	44	MHz
	8位UDB，单次触发，触发 = 无	–	–	44	MHz
	16位UDB，连续，触发 = 上升沿	–	–	33	MHz
	16位UDB，单次触发，触发 = 上升沿	–	–	33	MHz
	24位UDB，连续，触发 = 任一沿	–	–	28	MHz
	32位UDB，连续，触发 = 软件控制	–	–	25	MHz

3. 对于所有配置，通用设置为：Enable mode（使能模式）= Software only（仅软件），Capture mode（捕获模式）= None（无），Interrupt（中断）= On TC（TC 上）

4. 这些值提供了此组件的最大安全工作频率。可以在更高的时钟频率运行此组件，在该频率将需要使用 STA 结果验证时序要求。

组件更改

本节列出了各版本组件的主要更改。

版本	更改内容	更改原因/影响
2.50.a	对数据手册进行了编辑，以删除PSoC 5的引用。	使用PSoC5 LP来取代PSoC 5。
2.50	更新了数据手册和PSoC 4内存的使用情况。	支持新的器件。
2.40	已添加了MISRA合规性章节。	该组件没有任何特定偏差。
2.30	添加了PSoC5 LP支持。	
	更新了自定义程序，以在单次触发硬件使能模式中移除弹出的警告。	
	更新了直流和交流电气特性。 更新了资源和API存储器使用章节。	
	从符号文件中移除了芯片修订版枚举。添加了正式参数的说明。	
2.20	UDB定时器的Verilog更改。	修复一个问题，即当使用硬件使能信号时，在某些情况下TC输出会丢失。
	记录中断信号不可用于PSoC5 FF定时器。	此功能已移除，因为芯片不支持此功能。
	更新了自定义程序，使Cancel（取消）按键始终可用。	在某些错误情况下，Cancel（取消）按键不可用。
	广泛的数据手册更新。	定时器的各个实现（UDB、PSoC 3 FF、PSoC 5 FF）各有不同，这些不同之处并未充分描述。特别要查看“功能说明”的“配置”一节中提供的波形。
2.10	Verilog更新和自定义程序相关的更新。	用触发逻辑修复小问题，并修复GUI相关问题。
	当 Capture Mode（捕获模式）设为 None（无）时，禁用“Interrupt on Capture”（捕获中断方式）。	即使Capture Mode（捕获模式）设为“None”（无）且不可用时，“Interrupt on Capture”（捕获中断方式）复选框选项也仍然可用。
2.0	同步的输入。	所有输入都是在模块的输出处的固定功能实现中同步的。
	Timer_GetInterruptSource() 函数转换为宏。	Timer_GetInterruptSource()函数与Timer_ReadStatusRegister()函数具有完全相同的实现。为了节省代码空间，该函数转换为Timer_ReadStatusRegister()函数的宏替换。
	现在将输出寄存到组件时钟上。	为了避免组件输出中出现故障，需要将所有输出同步。如果可能，此同步在数据路径内部完成，以避免过多使用资源。

版本	更改内容	更改原因/影响
	在写入到辅助控制寄存器时执行了关键区域。	当写入辅助控制寄存器时使用CyEnterCriticalSection和CyExitCriticalSections函数，以便它不会被任何其他进程线程修改。
	矫正了错误的掩码，同时使用SetCaptureMode() API设置捕获模式。	用于设置捕获模式的掩码值错误。
	向数据手册中添加了特性数据。	
	对数据手册进行了少量编辑和更新。	