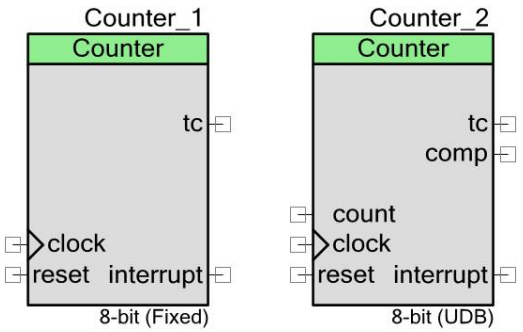
计数器
2.40



特性

- 支持 PSoC 3 和 PSoC 5LP 中的固定功能（FF - Fixed Function）实现
- 8、16、24 或 32 位计数器
- 递增、递减，或递增递减计数配置
- 可选的比较输出
- 可选的捕获输入
- 使能和复位输入，用于与其他组件进行同步
- 连续或单触发运行模式

概述

计数器组件提供了一种计数事件的方法。它可以执行基本的计数器功能，还提供了一些高级特性，如捕获、比较输出和计数方向控制。

对于 PSoC 3 和 PSoC 5LP 器件，可以使用 FF 模块或 UDB 实现组件。而对于 PSoC 4 器件，则仅支持 UDB 实现。与 FF 实现相比，UDB 实现通常拥有更多特性。如果您的设计足够简单，可考虑使用 FF，这样可以节省 UDB 资源用于其他目的。

下表显示了 FF 和 UDB 之间的主要区别。更多有关器件中 FF 资源的详细信息，请参考适用的器件数据手册或技术参考手册。

特性	FF	UDB
位数	8或16	8、16、24或32
运行模式	连续或单触发模式	连续或单触发模式
计数器模式	只进行递减计数	进行递增、递减，或递增递减计数
使能输入	无（只有软件使能）	有（硬件或软件使能）
捕获输入	无	有
捕获模式	无	上升沿、下降沿、任一边沿或软件控制

特性	FF	UDB
捕获FIFO	无（一个捕获寄存器）	有（最多4个捕获寄存器）
复位输入	有	有
终端计数输出	有	有
比较输出	无	有
比较模式	无	<、 ≤、=、≥、>，或软件控制
中断输出	有	有
中断条件	TC	TC、捕获和比较
周期寄存器	有	有
周期重新加载	有（复位或终端计数时总是重新加载）	有（发生一次或多次复位、TC、捕获和比较时重新加载）
时钟输入	仅限于时钟系统中的数字时钟	所有信号
取样时钟输入	无	需要有一个明确的时钟信号（组件时钟），以对该组件进行输入信号取样。

何时使用计数器

计数器默认用于对计数输入上的边沿事件进行计数。然而，计数器还有其他几种用途。

- 时钟分频器：驱动时钟连接计数输入并将比较或终端计数输出当作分频时钟输出。
- 频率计数器：通过将一个已知周期的信号连接至计数器的使能输入，同时计数信号来测量计数输入。
- 用作测量互补事件（如正交解码器的输出）的工具

注意：定时器组件最适合用于测量各事件之间的时间间隔。PWM 组件最适合用于以下情况：要求多个比较输出并带有更多的控制特性，比如中心对齐、输出非同步停止和死区输出等。

输入/输出接口

本节介绍了计数器组件的各种输入和输接口。某些 I/O 在其描述中所罗列的条件下可能不在符号上显示。

注意：除非另有说明，否则所有信号都是高电平有效。所有对计数器的输入必须在计数器之外同步进行。

输入	是否被隐藏	说明
Clock（时钟）	否	与UDB计数器相比，固定功能计数器的时钟输入的功能行为有所不同。 ■ 如果是固定功能计数器，则没有计数输入。每当时钟输入的上升沿到来时，固定功能计数器将进行更新（递减其内部计数器）。 ■ 如果是 UDB 计数器，则时钟输入和计数输入都会在计数器符号上显示。时钟是用来采样计数器组件的输入。UDB 计数器以一个同步计数器来实现，该同步计数器仅将时钟输入作为同步时钟。计数器的所有输入必须与时钟输入同步，以避免设置出现冲击。这也能确保 UDB 计数器实现的边沿检测电路正常工作。
Count（计数）	是	在固定功能计数器中，计数器符号上没有任何计数输入。每当时钟输入的上升沿到来时，固定功能计数器将更新其内部计数。 对于UDB计数器，计数器按照边沿检测逻辑更新其内部计数器，以确定一次更新事件。更新事件的源信号取决于UDB计数器的时钟模式。通过使用时钟输入，可以给边沿检测逻辑提供时钟脉冲。计数输入的两个边沿都必须符合时钟输入的设置，因此最大计数输入等于时钟输入频率的一半。 ■ 当 UDB 计数器处于 **Up Counter**（递增计数器）或 **Down Counter**（递减计数器）时钟模式时，边沿检测逻辑将检测与时钟输入同步的计数输入的上升沿。根据计数器被配置为递增计数器还是递减计数器，计数输入的边沿检测事件将分别递增或递减计数器。请参见第 24 页上的图 5，了解递增/递减计数器的功能说明。 ■ 当 UDB 计数器处于 **Count Input and Direction**（计数输入和方向）时钟模式时，边沿检测逻辑将检测与时钟输入同步的计数输入的上升沿。根据“增减”信号是 1 还是 0，计数输入的边沿检测事件将分别递增或递减计数器。请参阅第 25 页上的图 7，了解处于 **Count Input and Direction** 时钟模式的计数器的功能说明。 ■ 当 UDB 计数器处于 **Clock with UpCnt & DwnCnt**（带有递增计数和递减计数的时钟）的时钟模式时，则没有计数输入。计数器更新事件由对与时钟输入同步的递增计数器和递减计数器信号的边沿检测逻辑的结合来确定。为了能更新计数器，在边沿检测的整个过程中，UpCnt 或 DwnCnt 信号必须处于低电平状态。请参阅第 26 页上的图 9，了解处于 **Clock with UpCnt & DwnCnt** 时钟模式的计数器的功能说明。
upCnt（递增计数）	是	增加至计数器的信号。当将**Clock Mode**（时钟模式）设置为**Clock With UpCnt & DwnCnt**模式时，这一输入与递减计数输入和时钟输入一起使用，以允许计数器能作为一个编码器使用。这一输入的上升沿会递增计数值。递增计数输入的两个边沿都必须符合时钟输入的设置，因此最大计数输入是时钟输入频率的一半。

输入	是否被隐藏	说明
dwnCnt（递减计数）	是	减少至计数器的信号。当将计数器的**Clock Mode**设置为**Clock With UpCnt & DwnCnt**模式时，这一输入与递增计数输入和时钟输入一起使用，以允许该计数器能作为一个编码器使用。这一输入的上升沿会递减计数值。递减计数输入的两个边沿都必须符合时钟输入的设置，因此最大计数输入是时钟输入频率的一半。
up_ndown（递增和递减）	是	规定计数器的计数方向。仅在将**Clock Mode**设为**Count Input and Direction**（计数输入和方向）以使能方向控制时，才能使用该输入。在该计数输入的上升沿上，该输入上的‘1’会导致计数器递增，而这一输入上的‘0’会引起计数器递减。
Reset（复位）	否	该复位输入使计数器复归初始值。 ▪ 对于 **Up Counter** 配置，初始值为零。 ▪ 对于 **Down Counter**、**Count Input and Direction** 以及 **Clock With UpCnt & DwnCnt** 等配置，初始值设置为当前周期寄存器值。 在计数/时钟输入上对复位输入进行采样。 对于PSoC 3 Production或更高版本的芯片，固定功能计数器的终端计数引脚在复位期间保持低电平。复位输入用于复位PSoC 3 Production、PSoC 4和PSoC 5LP计数器实现中的控制寄存器，另外，还实现单触发运行模式功能。 UDB单触发模式需要一个复位脉冲来使能计数。如果计数器在加电后仅以单触发模式运行，那么只有在应用和移除复位后它才会开始计数。
Enable（使能）	是	计数器的硬件使能。仅在将**Enable Mode**（使能模式）参数设为**Hardware**（硬件）时，才能看到该输入。
Capture（捕获）	是	将当前的计数值捕获至一个捕获寄存器或FIFO上。当**Capture Mode**（捕获模式）参数被设为除**None**（无）之外的任何其他模式时，该输入均可见。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**Capture Mode**的设置。 在时钟输入上对捕获输入进行采样。

输出	是否被隐藏	说明
tc	否	终端计数是一种同步输出，表明计数值等于终端计数。该输出与计数器的时钟输入是同步的。计数值与终端计数一致后的一个时钟周期内，信号会变为高电平；并在计数值等于终端计数时，则信号会一直保持为高电平。如果在计数值等于终端计数时禁用计数器，输出将保持高电平，直到重新使能计数器为止。
Comp（比较）	是	比较输出表示：将计数器值与**Compare Mode**（比较模式）参数所配置的比较值进行比较。在比较事件发生后的一个时钟周期延迟内，比较输出会变为高电平。
interrupt（中断）	否	此中断输出由硬件中所配置的中断源驱动。所有源一起执行“或”运算以创建最终输出信号。中断源可能是：比较、终端计数或捕获。

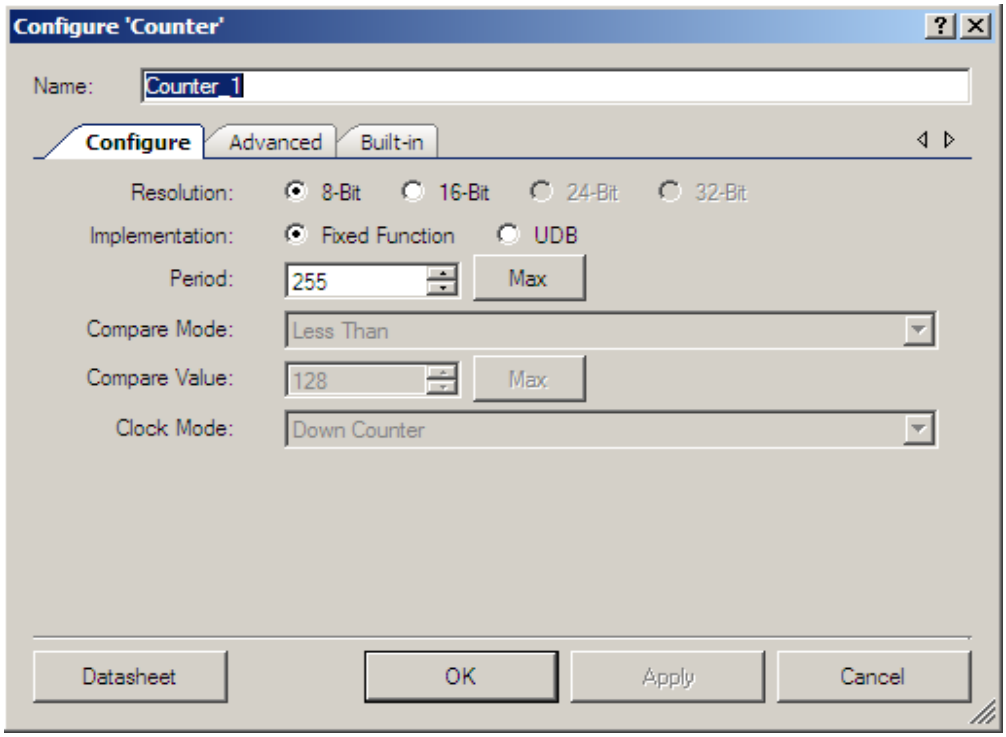
组件参数

将计数器拖入您的设计中，并双击它以打开 **Configure** 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并放置在硬件中的方式。如果您对这些项中的任何选项进行了更改，则必须重新编译硬件。则软件配置选项不影响项目的合成或放置。如果在构建前设置这些参数，即正在设置其初始值，那么这些初始值随时可能由所提供的 API 更改。下面章节中所描述的大多数参数都是硬件选项。软件选项也将同样说明。

Configure（配置）选项卡



Resolution（分辨率）

Resolution 参数定义了计数器组件的位宽度分辨率。该值可设置为 8、16、24 或 32 位，而与其相应的最大计数值分别为 255、65535、16777215 和 4294967295。

Implementation（实现）

Implementation 参数允许您可以选择计数器的固定功能模块实现或 UDB 实现。如果选中 **Fixed Function**（固定功能），就会禁用 UDB 功能。

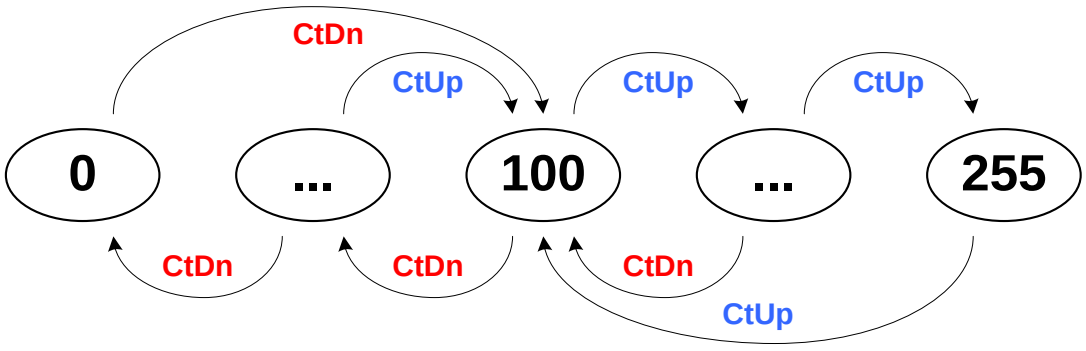
Period（周期 — 软件选项）

Period 参数定义了计数器组件的最大计数值（或翻转点）。该参数定义了所加载至周期寄存器上的初始值，软件可通过 Counter_WritePeriod() API 随时对其进行修改。

该值的限制由 **Resolution** 参数定义。如果 **Resolution** 参数设置为 8 位、16 位、24 位和 32 位，那么相应的 **Period**（周期）最大值分别定义为(2^8) – 1、(2^16) – 1、(2^24) – 1 和(2^32) – 1，或分别为 255、65535、16777215 和 4294967295。

当将**Clock Mode**配置为**Clock with UpCnt & DwnCnt**或**Count Input and Direction**时，计数器会在使能时或计数器上溢或下溢时设置周期。在这些时钟模式下，请勿将周期值设置为全 1 或全 0。如果周期值被设为全 1 或全 0，则计数器将无法工作。反而，一般将周期值设置为周期范围的中间点（例如：计数器的分辨率为 8 位，就将周期值设为 0x7F）。图 1 显示的是**Clock Mode**被设为**Count Input and Direction**。

图 1. 时钟模式



Compare Mode（比较模式 — 软件选项）

Compare Mode 参数配置了比较输出信号的操作。该信号是比较值参数和当前计数器值相比较的状态。该参数定义了初始设置。您可以随时修改该参数，以重新配置计数器组件的比较操作。

Compare Mode 可被设为以下的任意值：

- **Less Than**（小于）— 计数器值小于比较值。
- **Less Than Or Equal** （小于或等于）— 计数器值小于或等于比较值。
- **Equal To**（等于）— 计数器值等于比较值。
- **Greater Than**（大于）— 计数器值大于比较值。
- **Greater Than Or Equal**（大于或等于）— 计数器值大于或等于比较值。
- **Software Controlled**（软件控制）— 在运行时间通过调用 Counter_SetCompareMode() API，可将比较模式设置为上述五种比较模式中的任何一个。

Compare Value（比较值 — 软件选项）

Compare Value 参数定义了所加载到计数器的比较寄存器上的初始值。该值与 **Compare Mode** 参数一起用，以定义比较输出操作。

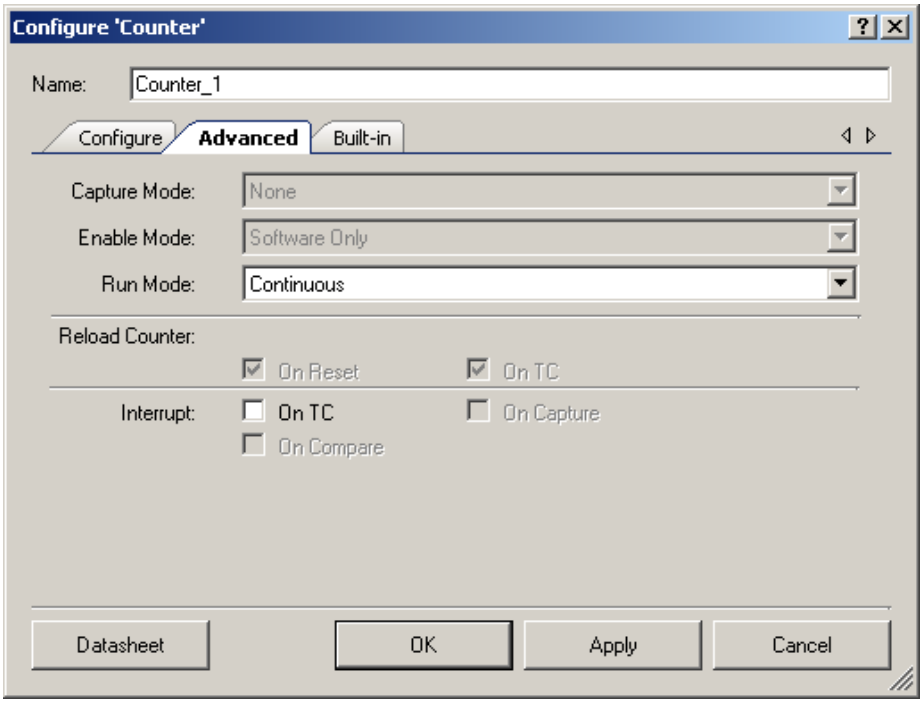
该值可以是 0 到（2^分辨率 – 1）之间的任何无符号整数，但必须小于或等于周期值。

Clock Mode（时钟模式）

Clock Mode 参数配置了计数器的计数方式。可将该模式设置为以下的任意值：

- **Count Input and Direction**（计数输入和方向）— 计数器为双向计数器。当输入时钟的每个上升沿的 up_ndown 输入为高电平时，计数器将递增计数；相反，当输入时钟的每个上升沿上的 up_ndown 输入为低电平时，则计数器将递减计数。
- **Clock With UpCnt & DwnCnt**（带递增计数和递减计数的时钟）— 计数器为双向计数器。根据时钟输入，它在 upCnt 输入的每个上升沿上会递增计数器值，则在 dwnCnt 输入的每个上升沿上会递减计数器值。在这种模式下，时钟输入频率必须至少为 upCnt 和 dwnCnt 输入频率的 2 倍。
- **Up Counter**（递增计数器）— 计数器只能递增计数。根据时钟信号，当使能计数器时，它在计数输入的上升沿上进行递增计数。
- **Down Counter**（递减计数器）— 计数器只能递减计数。根据时钟信号，当使能计数器时，它在计数输入的上升沿上进行递减计数。

Advanced（高级）选项卡



Capture Mode（捕获模式）

Capture Mode 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。可将该模式设置为以下的任意值：

- **None**（无）— 没有实现捕获。捕获输入引脚被隐藏。
- **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**（软件控制）— 在运行时间中，通过在控制寄存器 Counter_CTRL_CAPMODE_MASK 中设置捕获模式位，可以将该模式设置为上述所列举的各种捕获模式。*Counter.h* 头文件中定义了它们。

Enable Mode（使能模式）

Enable Mode 参数配置了计数器的使能实现。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的任意值：

- **Software Only**（仅使能软件）— 只有通过控制寄存器的使能位才可使能计数器。
- **Hardware Only**（仅使能硬件）— 只有通过使能输入才能使能计数器。
- **Hardware and Software**（硬件和软件）— 仅当硬件和软件使能值都为“真”时，才能使能计数器。

Run Mode （运行模式）

Run Mode 参数允许您可配置计数器组件连续运行或者以单触发模式运行。

- **Continuous**（连续）— 计数器使能后将连续运行。
- **One Shot**（单触发）— 计数器在一个单周期内运行，并在终端计数停止运行。复位后，计数器将开始另一个单周期。停止运行时，UDB 计数器会将周期值重新加载入计数寄存器，而对于固定功能计数器，计数寄存器仍保持终端计数值。

Reload Counter（重新加载计数器）

通过 **Reload Counter** 参数，您可以配置何时重新加载计数器值。如果发生以下所选的一个或多个事件，计数器值将重新加载。重新加载事件发生时，计数器将重新加载其起始值。

- **On Capture**（发生捕获）— 当发生捕获事件时，会重新加载计数器值。默认情况下，此参数会被清除。仅当 **Implementation** 被选为 **UDB** 时，该参数才会显示。
- **On Compare**（发生比较）— 当发生“比较结果为真”的事件时，会重新加载计数器值。默认情况下，该参数会被清除。仅当 **Implementation** 被选为 **UDB** 时，该参数才会显示。
- **On Reset**（发生复位）— 发生复位事件时，会重新加载计数器值。默认选择该参数。该参数始终显示。如果选择了固定功能计数器，该参数不能修改。如果是 UDB 计数器，该参数可被关闭。
- **On TC**（发生终端计数）— 当计数器上溢出（处于递增计数模式）或下溢出（处于递减计数模式）时，将重新加载计数器值。默认选择该参数。如果是固定功能计数器，该参数不能修改。如果是 UDB 计数器，该参数可被关闭。

  当时钟模式被设为 **Clock With UpCnt & DwnCnt** 时，如果计数器值为 0x00 或全是 0xFF，该选项会重新加载周期值。

下表列出了计数器组件各种时钟模式的重新加载和终端计数条件。

时钟模式	终端计数条件	计数器起始值以及当发生重新加载事件时的重新加载值
固定功能计数器	当计数器值为0时。	计数器的起始值和重新加载值均为该组件的**Period**（周期）值。
UDB递增计数器	当计数器值等于**Period**值时。	计数器的起始值和重新加载值均为0。
UDB递减计数器	当计数器值为0时。	计数器的起始值和重新加载值均为该组件的**Period**值。
UDB计数输入和方向	当计数器递增计数时，它的计数值为全1（即8位、16位、24位和32位计数器的值分别为0xFF、0xFFFF、0xFFFFFF和0xFFFFFFFF）。如果数器递减计数，计数值等于0。	计数器的起始值和重新加载值均为该组件的**Period**值。如果没有重新加载条件，计数器将返转，继续计数而不发生计数寄存器重新加载事件。
UDB带递增计数和递减计数的时钟	当计数器递增计数时，它的计数值为全1（即8位、16位、24位和32位计数器的值分别为0xFF、0xFFFF、0xFFFFFF和0xFFFFFFFF）。如果数器递减计数，那么计数值等于0。	计数器的起始值和重新加载值均为该组件的**Period**值。如果没有重新加载条件，计数器将返转，继续计数而不发生计数寄存器重新加载事件。

Interrupt（中断）

通过 **Interrupt** 参数，您可以配置初始中断源。当发生以下所选的一个或多个事件时，会生成中断。该参数定义了初始配置。软件可以随时重新配置该模式。

- **On TC**（发生终端计数）— 该参数始终有效；默认情况下不选择该参数。
- **On Capture**（发生捕获）— 默认情况下不选择该参数。该参数始终显示，但只在 **Implementation** 参数被设为 **UDB** 时才有效。
- **On Compare**（发生比较）— 默认情况下不选择该参数。该参数始终显示，但只在 **Implementation** 参数被设为 **UDB** 时才有效。

时钟模式的选择

对于计数器组件：

- 当 **Clock Mode** 参数被设为 **Up Counter** 或 **Down Counter** 时，计数输入可以是上升沿计数的任一信号。该组件的时钟输入对计数输入进行采样，并且上升沿和下降沿都必须符合时钟的设置要求。

- 如果 **Clock Mode** 参数被设为 **Count Input and Direction**，则计数输入可以是上升沿计数的任一信号。该组件的时钟输入对计数输入进行采样，并且上升沿和下降沿都必须符合时钟的设置要求。根据 up_ndown 输入，计数器会进行递增或递减计数。
- 将 **Clock Mode** 参数设为 **Clock With UpCnt & DwnCnt** 时，根据时钟输入对 upCnt 和 dwnCnt 上升沿进行采样。计数器会在 upCnt 信号的上升沿上进行递增计数，则在 dwnCnt 信号的上升沿上递减计数。upCnt 和 dwnCnt 的上升沿和下降沿都必须符合时钟的设置要求。

请参阅时钟组件数据手册和相应的器件数据手册，以了解更多有关 PSoC 3、PSoC 4 或 PSoC 5LP 时钟系统的详细信息。

固定功能组件

如果配置计数器组件能够使用器件的固定功能模块，则它有以下限制：

- 计数输入必须是时钟系统中的数字时钟。
- 如果时钟的频率与总线时钟的频率相同，则其必须即为总线时钟。

  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。

基于 UDB 的组件

对于PSoC 3/PSoC 5LP，您可以将来自任何源的任何数字信号连接至计数/时钟输入。该信号的频率范围有一定的限制，请参考该数据手册中的直流电和交流电电气特性（UDB 实现）一节。计数输入的频率最大不能超过任何计数器时钟模式下的时钟输入频率的一半。

对于 PSoC 4，您不能将时钟信号直接连接到该计数输入。

放置

PSoC Creator 根据 **Implementation** 参数将计数器组件放置在器件中。如果该参数被设为 **Fixed Function**，可将该组件放置在任何可用的固定功能计数器/定时器模块中。如果此参数被设为 **UDB**，则该组件将以最佳的配置放置在 UDB 阵列周围。

应用编程接口

通过应用编程接口（API），您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。以下各节将更加详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“Counter_1”分配给所提供的设计中的第一个器件实例。您可以将实例名称重命名为任何遵循标识符语法规则的唯一值。实例名称会成为与该组件相关的每个全局函数名称、变量和常量符号的前缀。为了增加可读性，下表中使用了实例名称“Counter”。

函数	说明
Counter_Start()	设置initVar变量，并调用Counter_Init()函数，然后调用Enable函数。
Counter_Stop()	禁用计数器
Counter_SetInterruptMode()	使能或禁用中断输出的源
Counter_ReadStatusRegister()	返回状态寄存器的当前状态
Counter_ReadControlRegister()	返回控制寄存器的当前状态
Counter_WriteControlRegister()	设置控制寄存器的位字段
Counter_WriteCounter()	将一个新的值直接写入到计数寄存器中。
Counter_ReadCounter()	强制执行捕获，然后返回捕获值
Counter_ReadCapture()	返回捕获寄存器的内容或FIFO的输出
Counter_WritePeriod()	写入周期寄存器
Counter_ReadPeriod()	读取周期寄存器
Counter_WriteCompare()	对比较寄存器进行写操作
Counter_ReadCompare()	读取比较寄存器
Counter_SetCompareMode()	设置比较模式
Counter_SetCaptureMode()	设置捕获模式
Counter_ClearFIFO()	清除捕获FIFO
Counter_Sleep()	停止计数器，并保存用户配置。
Counter_Wakeup()	恢复并使能用户配置
Counter_Init()	根据Configure对话框设的设置，初始化或恢复计数器。
Counter_Enable()	使能计数器
Counter_SaveConfig()	保存计数器配置
Counter_RestoreConfig()	恢复计数器配置

全局变量

变量	说明
Counter_initVar	指示计数器是否已初始化。该变量初始时为0，而在第一次调用Counter_Start()时被设为1。这允许组件第一次调用Counter_Start()子程序后无需重新初始化而仍可重新使能。 如果需要重新初始化组件，则可以在调用Counter_Start()或Counter_Enable()函数前先调用Counter_Init()函数。

void Counter_Start(void)

说明： 这是开始执行组件操作的首选方法。将Counter_Start()设为initVar变量，调用Counter_Init()函数，然后调用Counter_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 Counter_Enable()函数。

void Counter_Stop(void)

说明： 该函数只能禁用处于软件使能模式的计数器。

参数： 无

返回值： 无

其他影响： 如果**Enable Mode**（使能模式）被设为**Hardware Only**（仅使能硬件），则该函数并不起作用。

void Counter_SetInterruptMode(uint8 interruptSource)

说明： 该函数可使能或禁用中断输出的源。

参数： uint 8：中断源。请参阅本数据手册中状态寄存器一节，了解位定义。

返回值： 无

其他影响： FF和UDB的位位置不同。掩码#定义用于封装此不同之处。

uint8 Counter_ReadStatusRegister(void)

说明： 该函数可返回状态寄存器的当前状态。

参数： 无

返回值： uint8：当前的状态寄存器值。状态寄存器的位有：

[7]：未使用（0）

[6]：FIFO非空

[5]：FIFO满载

[4]：捕捕获状态

[3]：下溢状态

[2]：溢出状态

[1]： A0 Zero状态

[0]：比较输出

请参阅本数据手册中状态寄存器一节，了解位定义。

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

uint8 Counter_ReadControlRegister(void)

说明： 该函数可返回控制寄存器的当前状态。仅在控制寄存器中定义的模式之一被实际使用时，该功能才可用。

参数： 无

返回值： uint8：当前的控制寄存器值。控制寄存器位为：

[7]：计数器使能

[6:5]：未使用

[4:3]：捕获模式选择

[2:0]：比较模式选择

请参阅本数据手册中控制寄存器一节，了解位定义。

其他影响： 无

void Counter_WriteControlRegister(uint8 control)

说明： 该函数可设置控制寄存器的位字段。仅在控制寄存器中定义的模式之一被实际使用时，该功能才可用。

参数： uint8：控制寄存器位字段。控制寄存器位为：

[7]：计数器使能

[6:5]：未使用

[4:3]：捕获模式选择

[2:0]：比较模式选择

请参阅本数据手册中控制寄存器一节，了解位定义。

返回值： 无

其他影响： 无

void Counter_WriteCounter(uint8/16/32 count)

说明： 该函数可将一个新的值直接写入计数寄存器中。

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

返回值： 无

其他影响： 覆盖计数器值。这可能导致对比较输出、终端计数输出或周期宽度进行的不必要行为。这不是原子写操作，该函数可能被中断。调用该函数前必先禁用计数器。

uint8/16/32 Counter_ReadCounter(void)

说明： 该函数强制执行捕获，然后返回捕获值。

发生的捕获不视为捕获事件，它不会导致计数器复位，也不会触发中断。

参数： 无

返回值： uint8/16/32：当前计数器值。对于24位计数器，返回类型是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）。

uint8/16/32 Counter_ReadCapture(void)

说明： 该函数会返回捕获寄存器的内容或FIFO的输出（仅适用于UDB）。

参数： 无

返回值： uint8/16/32：当前捕获值。对于24位计数器，返回类型是uint32。

其他影响： 无

void Counter_WritePeriod(uint8/16/32 period)

说明： 该函数会对周期寄存器进行写操作。

参数： uint8/16/32：新的周期值。对于24位计数器，该参数为uint32。

返回值： 无

其他影响： 计数器输出的周期不会改变，直到计数器重新加载为止。该周期将根据**Reload Counter**参数的配置进行重新加载。

uint8/16/32 Counter_ReadPeriod(void)

说明： 该函数会读取周期寄存器。

参数： 无

返回值： uint8/16/32：当前的周期值。对于24位计数器，返回类型是uint32。

其他影响： 无

void Counter_WriteCompare(uint8/16/32 比较值)

说明： 该函数会写入比较寄存器中。它仅在选择UDB实现时才可用。

参数： uint8/16/32：新的比较值。对于24位计数器，该参数为uint32。

返回值： 无

其他影响： 根据所写入的值和计数器的当前值，比较输出可能立即改变。

uint8/16/32 Counter_ReadCompare(void)

说明： 该函数会读取比较寄存器。它仅在选择UDB实现时才可用。

参数： 无

返回值： uint8/16/32：当前比较值。对于24位计数器，返回类型是uint32。

其他影响： 无

void Counter_SetCompareMode(uint8 compareMode)

说明： 该函数用于设置比较模式。 仅在UDB实现的情况下，并且选择了软件比较模式时，该函数才可用。

参数： uint8：查看所列举的比较模式。也可以参考控制寄存器一节中的内容。

```
Counter__B_COUNTER__LESS_THAN
Counter__B_COUNTER__LESS_THAN_OR_EQUAL
Counter__B_COUNTER__EQUAL
Counter__B_COUNTER__GREATER_THAN
Counter__B_COUNTER__GREATER_THAN_OR_EQUAL
Counter__B_COUNTER__SOFTWARE
```

返回值： 无

其他影响： 根据所写入的值和计数器的当前值，比较输出可能立即改变。

void Counter_SetCaptureMode(uint8 captureMode)

说明： 该函数设置了捕获模式。 仅在UDB实现的情况下并且将**Capture Mode** 参数设为**Software Controlled**时，该函数才可用。

参数： uint8：查看所列举的捕获模式。也可以参考控制寄存器一节中的内容。

```
Counter__B_COUNTER__NONE
Counter__B_COUNTER__RISING_EDGE
Counter__B_COUNTER__FALLING_EDGE
Counter__B_COUNTER__EITHER_EDGE
Counter__B_COUNTER__SOFTWARE_CONTROL
```

返回值： 无

其他影响： 无

void Counter_ClearFIFO(void)

说明： 该函数清除了捕获FIFO。它仅在选择UDB实现时才可用。

请参阅本数据手册中功能说明部分的UDB FIFO一节所介绍的内容。

参数： 无

返回值： 无

其他影响： 无

void Counter_Sleep(void)

说明： 这是组件准备进入睡眠模式时的首选子程序。Counter_Sleep()子程序保存当前组件状态。然后它依次调用Counter_Stop()和Counter_SaveConfig()函数来保存硬件配置。

调用CyPmSleep()或CyPmHibernate()函数前，先调用Counter_Sleep()函数。欲了解更多有关功耗管理函数的详细信息，请参考*《系统参考指南》*中“PSoC Creator”章节的内容。

参数： 无

返回值： 无

其他影响： 对于FF实现，所有寄存器将保持处于低功耗模式。对于UDB实现，会保存并恢复控制寄存器和计数器值寄存器。

此外，只要调用Counter_Sleep()就可以保存使能状态（而不需要调用Counter_Stop()函数）。

void Counter_Wakeup(void)

说明： 这是将组件恢复为调用Counter_Sleep()时的状态的首选子程序。Counter_Wakeup()函数调用了Counter_RestoreConfig()函数以恢复配置。如果在调用Counter_Sleep()函数前使能了组件，则Counter_Wakeup()函数又重新使能组件。

参数： 无

返回值： 无

其他影响： 如果调用Counter_Wakeup()数前未调用Counter_Sleep()或Counter_SaveConfig()函数，可能会产生意外行为。

void Counter_Init(void)

说明： 根据自定义程序“Configure”对话框的设置，初始化或恢复组件。通常不需要调用Counter_Init()，因为Counter_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，对所有寄存器进行设置。

void Counter_Enable(void)

说明： 激活硬件，并开始执行组件操作。通常不需要调用Counter_Enable()，因为Counter_Start()子程序会调用该函数，这是开始组件操作的首选方法。该函数可在任何软件控制的使能模式下使能计数器。

参数： 无

返回值： 无

其他影响： 如果将**Enable Mode**参数设为**Hardware Only**，那么该函数不会对计数器的操作产生任何影响。

void Counter_SaveConfig(void)

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

参数： 无

返回值： 无

其他影响： 无

void Counter_RestoreConfig(void)

说明： 该函数会恢复组件配置和非保留寄存器。它还将组件参数值恢复为在调用Counter_Sleep()函数之前的值。

参数： 无

返回值： 无

其他影响： 调用该函数前未调用Counter_Sleep()或Counter_SaveConfig()函数，可能会产生意外行为。

有条件编译信息

计数器组件 API 文件要求两个有条件的编译定义来处理计数器必须支持的多项配置。该 API 文件必须有条件地在 FF 或 UDB 模块之间选择的 **Resolution**（分辨率）和 **Implementation**（实现）参数上进行编译。所定义的两个条件基于这些参数。该 API 文件绝不能直接使用这些参数，而应使用此处列明的两个定义。

Counter_DataWidth

在构建时，为 **Resolution**（分辨率）值分配了数据宽度定义。它被应用在整个 API 中，以便根据该信息的 API 函数的正确数据宽度类型进行编译。

Counter_UsingFixedFunction

UsingFixedFunction 定义多用于头文件中，用来进行正确的寄存器分配。这是必需的，因为固定功能模块中所提供的寄存器不同于在 UDB 中实现该组件时使用的寄存器。在某些情况下，该定义也与数据宽度定义一起使用，因为固定功能模块的最大数据宽度被限制为 16 位。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本器件的偏差情况。定义了下面两种类型的偏差：

- 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。

该计数器组件没有任何特定偏差。

固件源代码示例

PSoC Creator 在“Find Example Project”（查找示例项目）对话框中提供了多种包括原理图和代码示例的示例工程。要查看特定组件示例，请打开“Component Catalog”中的对话框或原理图中的组件实例。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》部分中主题为“查找示例项目”的内容。

功能说明

常规操作

计数器可以单向（递增或递减）或双向计数，取决于 **Clock Mode** 参数的设置。

- 如果它被设为 **Up Counter** 或 **Down Counter**，组件只能单向计数。对于在根据时钟输入的计数输入上的每一个上升沿，计数寄存器可递增或递减一次。
- 如果被设为 **Clock Input and Direction** 或 **Clock With UpCnt & DwnCnt**，该组件可根据 upCnt、dwnCnt 和 up_ndown 等输入来进行双向计数。这些输入在本数据手册的输入/输出接口一节中有详细说明。

计数器上溢/下溢

计数器在任何时钟模式下均可发生下溢和上溢。在状态寄存器中的有效位指示何时发生了上溢或下溢。在模式寄存器中的有效位控制了在这些条件下是否生成中断。

时钟模式	发生上溢的条件	发生下溢的条件
递减计数器	未定义。应屏蔽中断生成。	计数寄存器值等于0。
递增计数器	计数寄存器值等于周期寄存器值	未定义。应屏蔽中断生成。
时钟输入和方向	计数寄存器值等于0xFF、0xFFFF、0xFFFFFF或0xFFFFFFFF	计数寄存器值等于0。
带递增计数和递减计数的时钟	计数寄存器值等于0xFF、0xFFFF、0xFFFFFF或0xFFFFFFFF	计数寄存器值等于0。

计数器输出

可监控并重新加载计数器寄存器。有两种输出（即为终端计数和比较输出），可用于监控计数寄存器的当前值，并可以配置为重新加载事件。请参考输入/输出接口一节了解更多详细信息。

从周期寄存器中重新加载计数寄存器值。下表显示的是在每一种 **Clock Mode** 的设置中，终端计数和重新加载如何工作：

时钟模式	终端计数输出有效的条件	计数器重新加载的内容
递减计数器	计数寄存器值等于0后的一个时钟输入周期	计数寄存器值等于0时周期寄存器的内容。
递增计数器	计数寄存器值等于周期寄存器值后的一个时钟输入周期	当计数寄存器值等于周期寄存器值的时候，计数器立即复位为0。
时钟输入和方向	计数寄存器翻转回0后的一个时钟输入周期	无 — 计数器翻转
带递增计数和递减计数的时钟	计数寄存器值等于0后的一个时钟输入周期	无 — 计数器翻转

比较输出持续不断地指示与比较值相比较的计数器值。**Compare Mode** 参数可以配置为所有的标准模式（例如，**Less Than Or Equal**、**Greater Than**）。当计数器正在进行计数时，它可以用于创建不同的输出波形。该比较输出与计数器的时钟输入同步。

计数器输入

可以在硬件或固件中执行捕获操作。将计数器寄存器的当前值复制到捕获寄存器或 FIFO 中。这样，固件稍后可读取此捕获值。

复位和使能特性允许计数器组件与其他组件同步。计数器组件仅在使能并且不保持复位状态时才能计数。可以通过硬件或固件来复位或使能它。

计数器中断

可以使用中断输出向 CPU 或其他组件进行关于事件发生的通信。该中断可以设置为在一个或多个事件共同发生时有效。中断处理程序在设计时应该周全考虑到如何确定中断源，中断是否对边沿或电平敏感，以及如何清除中断源。

计数寄存器

有两种寄存器：状态寄存器和控制寄存器。请参考寄存器一节的内容。

配置

以下章节讲述了一些不同的时钟组件配置。

默认配置

当您将一个计数器组件拖入 PSoC Creator 原理图中，默认配置是一个 8 位固定功能计数器，它在时钟输入的上升沿上对寄存器进行递减计数。图 2 显示的是默认组件符号和 **Configure** 对话框选项卡。

图 2. 默认配置

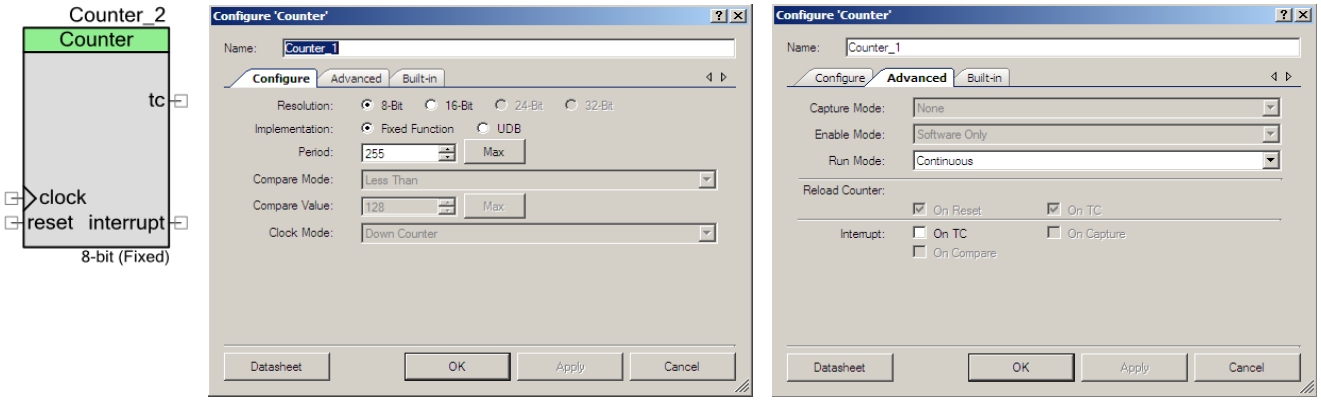
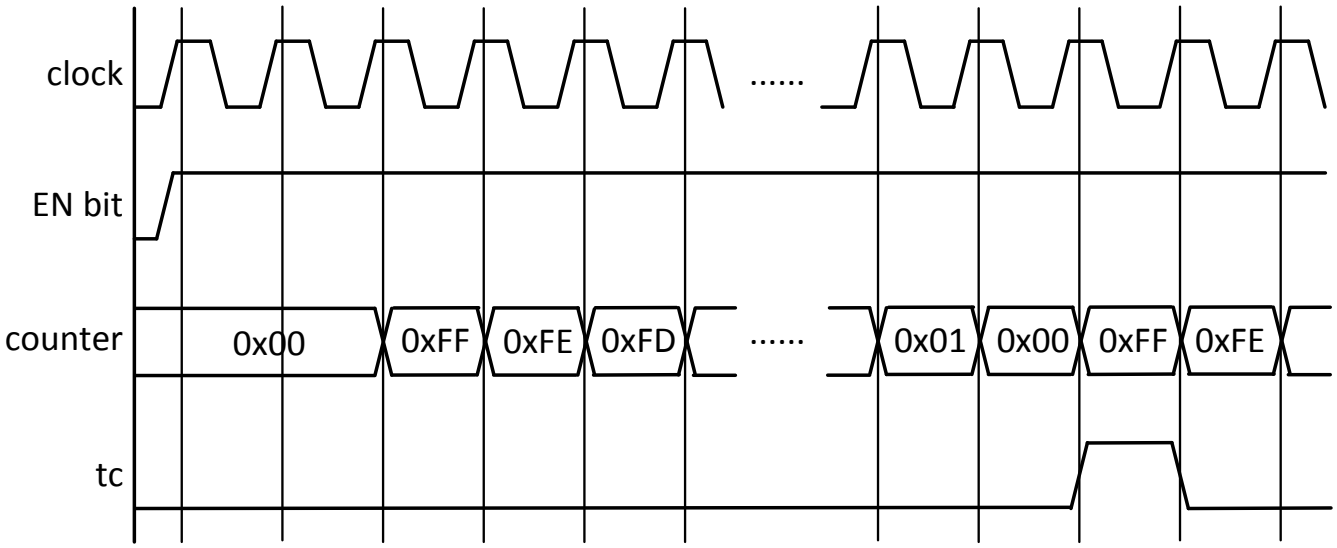


图 3 显示的是默认配置的时序图。

图 3. 默认配置波形



UDB 8 位递增/递减计数器配置

在该配置中，根据所选的时钟模式，计数寄存器将在时钟输入的上升沿上进行递增或递减计数。如果所选的时钟模式是 **Up Counter**，计数寄存器从 0 递增到周期值。如果所选的时钟模式是 **Down Counter**，则计数寄存器从周期值递减到 0。

图 4 显示了 UDB 8 位递增/递减计数器符号和 **Configure** 对话框选项卡。

图 4. UDB 8 位递增/递减计数器配置

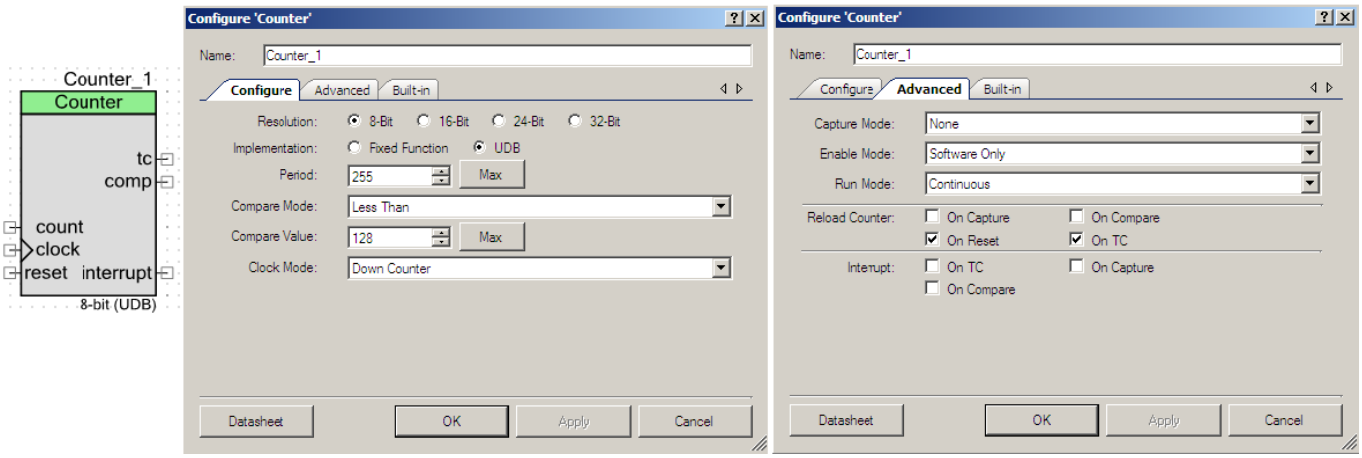
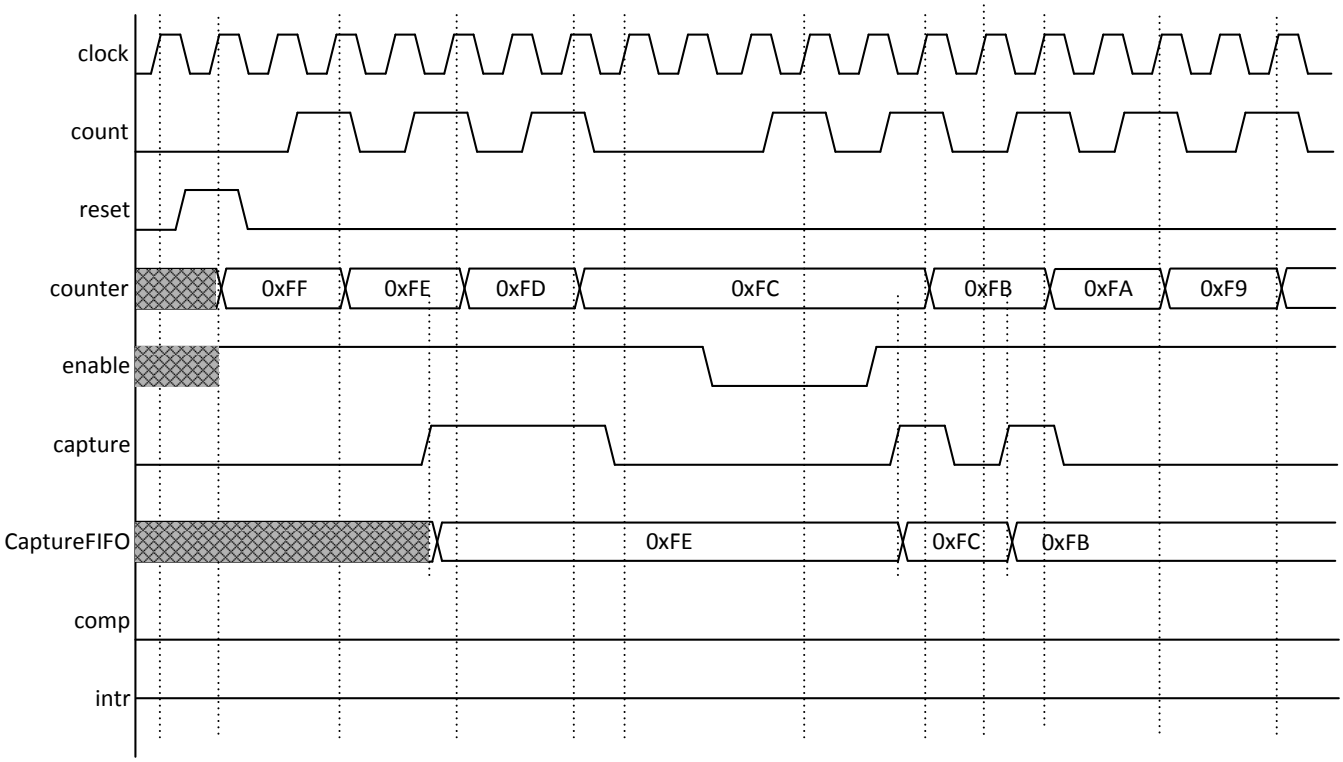


图 5 显示的是 UDB 8 位递增/递减计数器配置的时序图。

图 5. UDB 8 位递增/递减计数器配置波形



时钟输入和方向的配置

在该配置中，根据 up_ndown 输入终端的信号，计数寄存器会递增或递减计数。当 up_ndown 输入接收到一个高电平信号时，计数器在计数输入的上升沿上递增计数。如果 up_ndown 输入接收到一个低电平信号，则计数器在计数输入的上升沿上递减计数。

图 6 显示了 **Clock input and Direction**（时钟输入和方向）的配置符号和 **Configure** 对话框选项卡。

图 6. 时钟输入和方向的配置

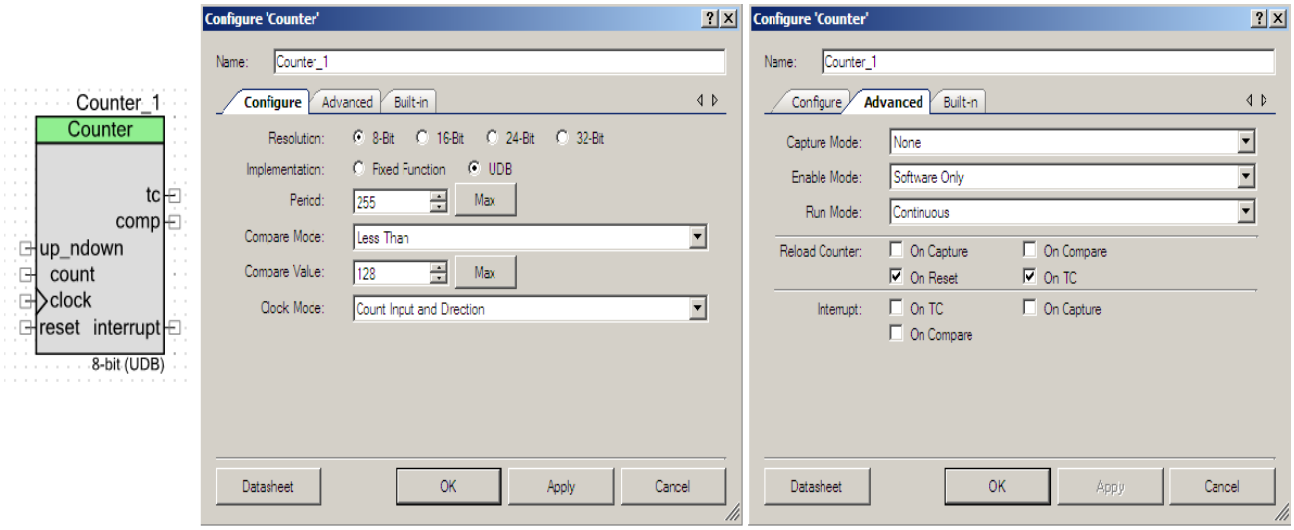
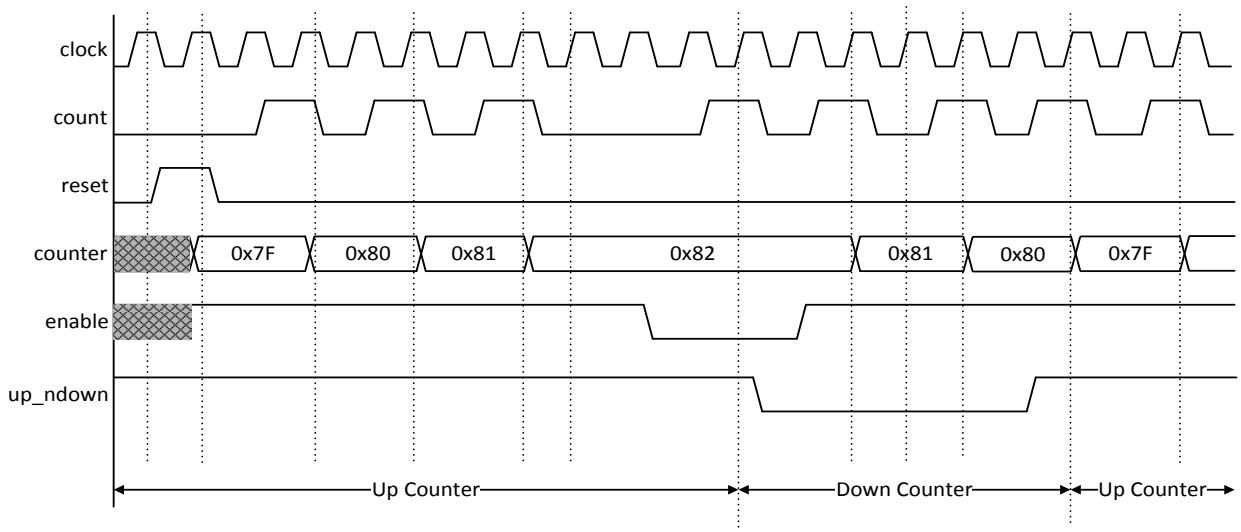


图 7 显示的是 **Clock Input and Direction** 模式配置的时序图。

图 7. 时钟输入和方向模式配置波形



Timing Diagram for Clock with Direction

带递增计数和递减计数的时钟配置

在该配置中，计数输入缺失。根据对递增计数器和递减计数器输入的信号，计数器会递增或递减计数。

图 8 显示了 **Clock with UpCnt & DwnCnt** 配置符号和 **Configure** 对话框选项卡。

图 8. 带递增计数和递减计数的时钟模式配置

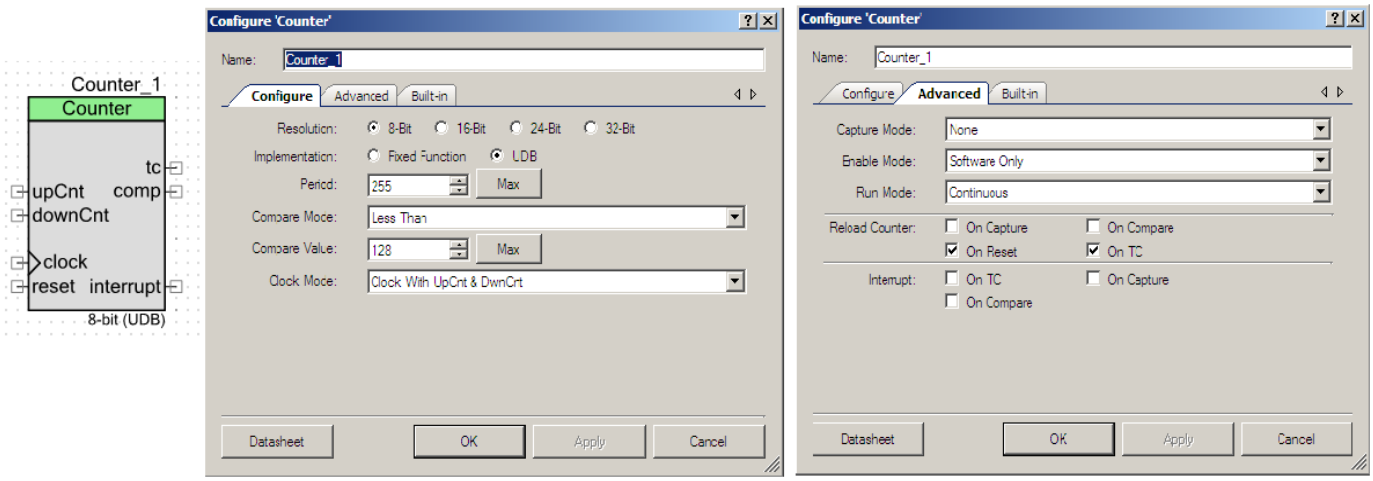
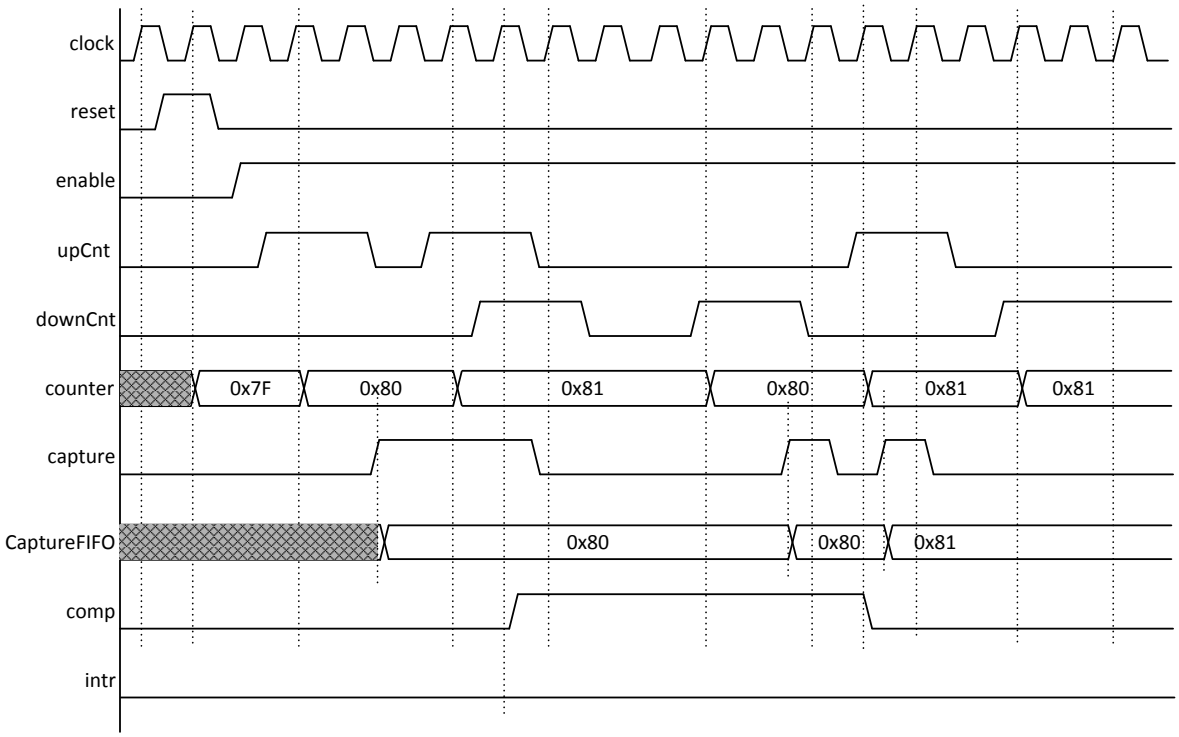


图 9 显示了 **Clock with UpCnt & DwnCnt** 模式配置的时序图。

图 9. 带递增计数和递减计数的时钟模式配置波形



事件计数器配置

当将 **Implementation** 参数设置为 **Fixed Function** 时，对于可以应用到计数输入的信号是有限制的。因此，将组件设置为 **UDB** 更容易创建事件计数器。在该配置中，间歇性异步事件可以被检测到并加以处理，以生成一个脉冲。时钟输入可用于取样该计数输入，以生成一个上升沿，该上升沿根据 **Clock Mode** 的设置，会导致计数器递增或递减计数。CPU 捕获并读取计数寄存器，以确定已发生的事件数量。

时钟分频器配置

将 **Implementation** 参数设为 **UDB** 也能够使能一个比较输出。该输出可以用于创建一个具备可编程频率和占空比的时钟分频器。在默认配置下，比较输出是一个占空比为~50%且频率为输入时钟频率的 1/256 的时钟。

图 10. 时钟分频器配置

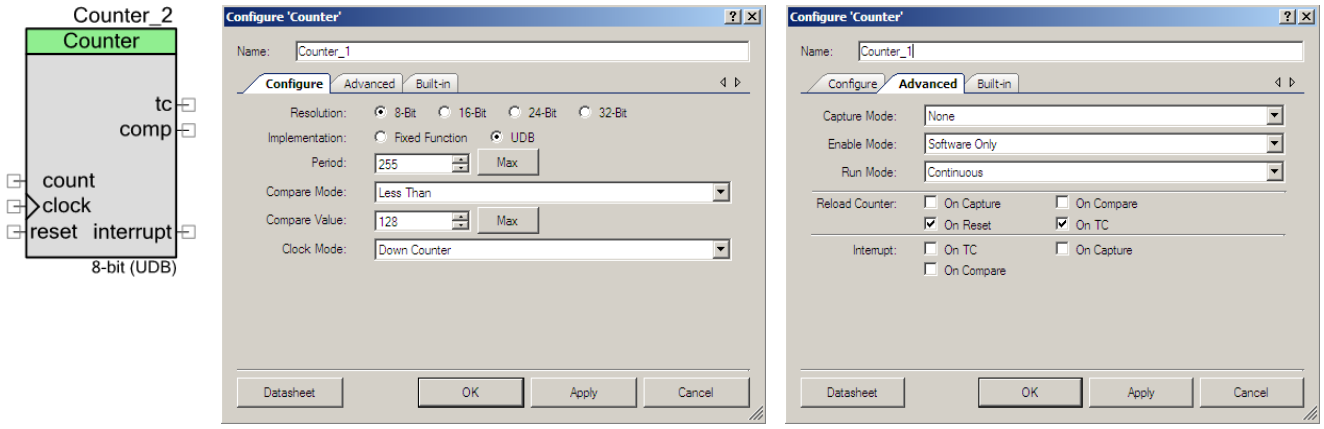
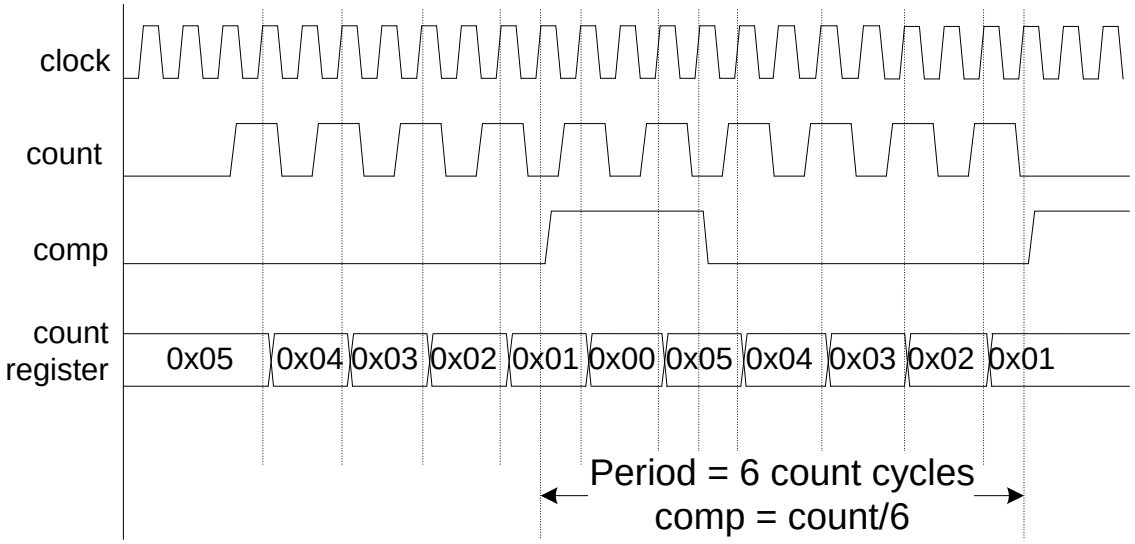


图 11 是一个波形示例，其周期为 6，比较值为 2，**Compare Mode** 参数被设为 **Less Than**。

图 11. 时钟分频器配置波形示例



频率计数器配置

添加硬件使能能够使计数器实现一个频率计数器功能。如果使能输入由一个已知信号驱动，那么就可以确定计数输入的信号频率。如果 **Clock Mode** 参数被设为 **Up Counter** 而不是 **Down Counter**，就可能够简化计算方式。

图 12. 频率计数器配置

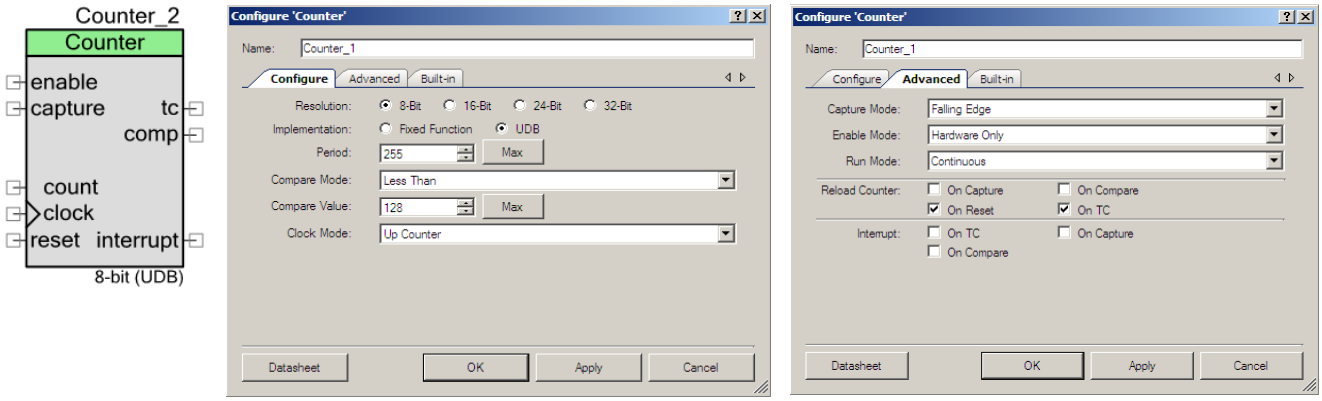
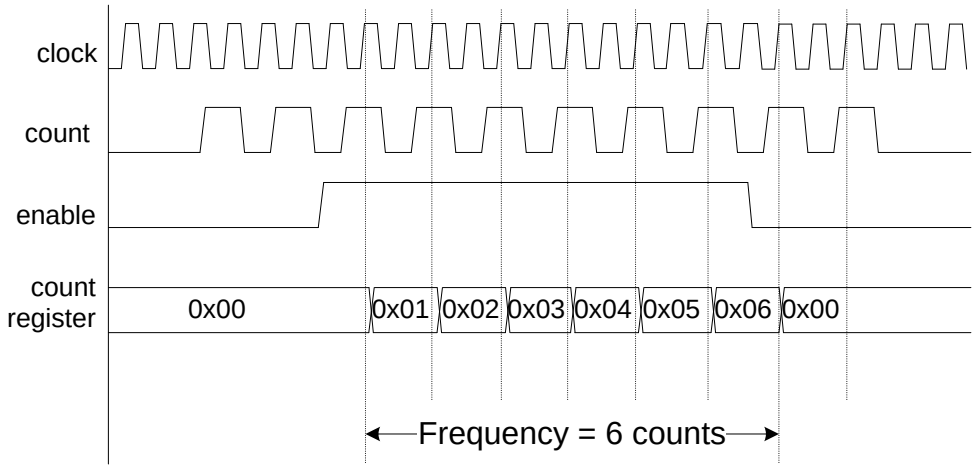


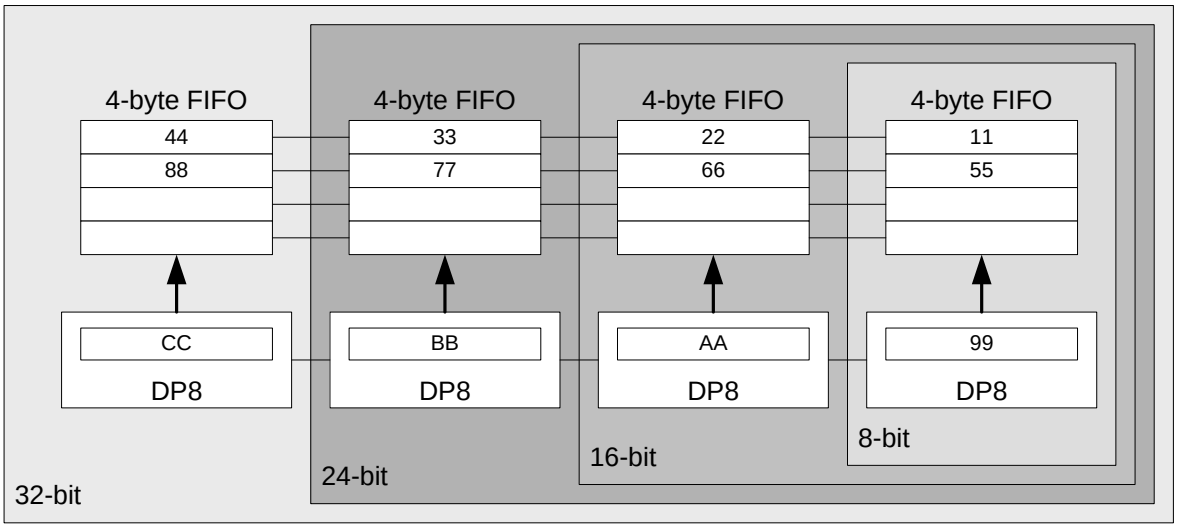
图 13. 频率计数器配置波形示例



UDB FIFO

每个 UDB 数据路径含有两个 8 位 FIFO 寄存器：F0 和 F1（请参考可应用的器件数据手册或技术参考手册了解更详细的信息）。每个 FIFO 的深度为 4 个字节。计数器 UDB 实现利用其中一个 FIFO 作为捕获寄存器。其他数据路径里的 FIFO 可用于 16 位、24 位和 32 计数器。因此，在 CPU 必须读取捕获寄存器以避免丢失数据之前，可以执行最多四次捕获。

如果 FIFO 为满，并且发生后续写入操作（上溢），那么新的数据将覆盖 FIFO 的前部分（当前数据作为输出，读取得到下个数据）。



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

寄存器

定义了几项常量来寻址所有寄存器。每个寄存器定义都需要一个指向寄存器数据或寄存器地址的指针。由于不同的编译器有不同的字节顺序设置，因此当寄存器访问大于 8 位时使用宏 CY_GET_REGX 和 CY_SET_REGX，每个寄存器都有_PTR 定义。所生成的头文件中提供了_PTR 定义。

状态寄存器

状态寄存器是一个只读寄存器，包含为计数器定义的状态位。使用 Counter_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字段的定义，因为 FF 实现和 UDB 实现的位字段可能有所不同。

状态寄存器中的某些位是粘滞的，表明它们被设置为 1 后，就保持此状态，一直到读取寄存器后将其清除为止。粘滞位的状态数据在计数器的输入时钟沿处寄存，使得所有粘滞位具有了计数器的时序分辨率。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。

Counter_Status（UDB 实现）

位	7	6	5	4	3	2	1	0
名称	RSVD	FIFO非空	FIFO已满	捕获	下溢	上溢	零	Cmp
粘滞	不可用	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE

Counter_Status（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	终端计数	捕获	使能	停止	保留	保留	保留	保留
粘滞	TRUE	TRUE	TRUE	TRUE	不可用	不可用	不可用	不可用

有关寄存器的详细信息，请参见芯片的《技术参考手册》（TRM）。

位名	头文件中的#define	说明
Cmp	Counter_STATUS_CMP	比较输出为高电平时，此位变为1。
零	Counter_STATUS_ZERO	计数器值等于零时，此位变为1。
上溢	Counter_STATUS_OVERFLOW	计数器值等于周期值时，此位变为1。
下溢	Counter_STATUS_UNDERFLOW	计数器值等于零时，此位变为高电平。
捕获	Counter_STATUS_CAPTURE	每当触发有效的捕获事件，该位就变为1。但不包括软件捕获。

位名	头文件中的#define	说明
FIFO已满	Counter_STATUS_FIFOFULL	当UDB FIFO达到已满状态（四个条目）时，此位变为1。
FIFO非空	Counter_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 Counter_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字段的定义，因为 FF 实现和 UDB 实现的位字段可能有所不同。

计数器组件中断输出是所有中断源的“或”函数。可以通过模式寄存器中相应的位来使能或屏蔽每个源。

Counter_Mode（UDB 实现）

位	7	6	5	4	3	2	1	0
名称	保留	FIFO非空	FIFO已满	捕获	下溢	上溢	零	Cmp

Counter_Mode（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	保留	保留	保留	保留	终端计数	捕获	使能	停止

位名	头文件中的#define	使能中断输出的条件
Cmp	Counter_STATUS_CMP_INT_MASK	比较
零	Counter_STATUS_ZERO_INT_MASK	计数寄存器值等于0
上溢	Counter_STATUS_OVERFLOW_INT_MASK	计数寄存器上溢
下溢	Counter_STATUS_UNDERFLOW_INT_MASK	计数寄存器下溢
捕获	Counter_STATUS_CAPTURE_INT_MASK	捕获
FIFO已满	Counter_STATUS_FIFOFULL_INT_MASK	UDB FIFO已满
FIFO非空	Counter_STATUS_FIFONEMP_INT_MASK	UDB FIFO非空

控制寄存器

控制寄存器允许您可控制计数器的常规操作。分别使用 Counter_WriteControlRegister()和 Counter_ReadControlRegister()函数对该寄存器进行写入和读取操作。所有在控制寄存器上的操作必须使用以下针对位字段的定义，因为 FF 实现和 UDB 实现的位字段可能有所不同。

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

Counter_Control（UDB 实现）

位	7	6	5	4	3	2	1	0
名称	使能	保留	保留	捕获模式[1:0]		比较模式[2:0]		

Counter_Control（固定功能实现）

位	7	6	5	4	3	2	1	0
名称	保留	保留	保留	保留	保留	单触发	保留	使能

位名	头文件中的#define	说明/枚举类型
比较模式	Counter_CTRL_CMPMODE_MASK	比较模式控制位定义了预期的比较输出操作。该位字段是在初始化时使用**Compare Mode**参数中所定义的比较模式下配置的。 ■ Counter__B_COUNTER__CM_LESSTHAN ■ Counter__B_COUNTER__CM_LESSTHANOREQUAL ■ Counter__B_COUNTER__CM_EQUAL ■ Counter__B_COUNTER__CM_GREATERTHAN ■ Counter__B_COUNTER__CM_GREATERTHANOREQUAL
捕获模式	Counter_CTRL_CAPMODE_MASK	捕获模式控制位是一个两位字段，定义预计的捕获输入操作。该位字段是在初始化时使用**Capture Mode**参数中所定义的捕获模式下配置的。 ■ Counter__B_COUNTER__CPTM_NONE ■ Counter__B_COUNTER__CPTM_RISINGEDGE ■ Counter__B_COUNTER__CPTM_FALLINGEDGE ■ Counter__B_COUNTER__CPTM_EITHEREDGE
使能	Counter_CTRL_ENABLE	该位使能了软件控制条件下的计数。仅在**Enable Mode**参数被设为**Software Only**或**Hardware and Software**时，此位才有效

位名	头文件中的#define	说明/枚举类型
单触发	Counter_ONESHOT	此位会选择单触发模式或连续运行模式。 ■ 1：表示单触发模式 ■ 0：表示连续运行模式

计数器（根据分辨率不同而分为 8 位、16 位、24 位或 32 位）

计数寄存器包含当前的计数器值。会对该寄存器进行递增或递减计数，以响应多种不同的计数/时钟输入。可以使用 Counter_ReadCounter()函数随时读取该寄存器。

捕获（根据分辨率不同分为 8 位、18 位、24 位或 32 位）

捕获寄存器包含捕获的计数器值。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 一节中的内容以了解更详细的信息。

周期（根据分辨率不同而分为 8 位、16 位、24 位或 32 位）

周期寄存器包含 Counter_WritePeriod()函数所设置并由 **Period** 参数在初始化时定义的周期值。发生重载事件时，该周期寄存器会被复制到计数寄存器中。

比较（根据分辨率不同而分为 8 位、16 位、24 位或 32 位）

比较寄存器包含用于确定比较（comp）输出状态的比较值。

使用资源

根据实现参数，计数器组件使用一个 FF 计数器/定时器模块或被放置于整个 UDB 阵列中。UDB 实现使用了以下资源。

配置	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位递增计数器	1	6	1	1	–	–
8位有方向计数器	1	9	1	1	–	–
16位递增计数器	2	7	1	1	–	–
16位有方向计数器	2	9	1	1	–	_
24位递增计数器	3	6	1	1	–	_
32位递增计数器	4	7	1	1	–	–

API 存储器的使用情况

根据编译器、器件、所使用的 API 数量以及组件的配置情况的不同，组件所用的存储器使用情况也不一样。下表提供了在某一器件配置中的所有 API 使用的存储器使用情况。

通过使用“Release”模式中的相应编译器，可以进行测量操作。在该模式下，存储器的大小得到优化。对于特定的设计，分析完编译器生成的映射文件后可以确定存储器的使用情况。

配置	PSoC 3（Keil_PK51）		PSoC 4（GCC）		PSoC 5LP（GCC）	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
8位FF计数器	257	5	不可用	不可用	376	5
16位FF计数器	261	6	不可用	不可用	376	9
8位UDB递增计数器	265	5	412	5	460	5
8位有方向UDB计数器	265	5	412	5	460	5
16位递增计数器	316	6	412	5	460	9
16位有方向计数器	317	6	412	5	460	9
24位UDB递增计数器	307	8	436	5	472	13
32位UDB递增计数器	306	8	412	5	460	13

PSoC 3 的直流电和交流电电气特性（FF 实现）

下面各值表示其预计性能，它们基于初始特性数据。

直流电规范

参数	说明	条件	最小值	典型值	最大值	单位
	模块电流消耗	16位计数器，在所列出的输入时钟频率下	–	–	_	µA
	3 MHz		–	15	–	µA
	12 MHz		–	60	_	µA
	48 MHz		–	260	–	µA
	67 MHz		–	350	–	µA

交流规范

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		15	–	–	ns
	分辨率		15	–	–	ns
	脉冲宽度		15	–	–	ns
	脉冲宽度（外部）		30	–	–	ns
	使能脉冲宽度		15	–	–	ns
	使能脉冲宽度（外部时钟）		30	–	–	ns
	复位脉冲宽度		15	–	–	ns
	复位脉冲宽度（外部时钟）		30	–	–	ns

PSoC 5LP 的直流和交流电气特性（FF 定时器）

下面各值表示其预计性能，它们基于初始特性数据。

直流电规范

参数	说明	条件	最小值	典型值	最大值	单位
	模块的电流消耗	16位计数器，在所列出的输入时钟频率下	–	–	_	µA
	3 MHz		–	15	–	µA
	12 MHz		–	60	_	µA
	48 MHz		–	260	–	µA
	67 MHz		–	350	–	µA

交流规范

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		15	–	–	ns
	分辨率		15	–	–	ns
	脉冲宽度		15	–	–	ns
	脉冲宽度（外部）		30	–	–	ns
	使能脉冲宽度		15	–	–	ns
	使能脉冲宽度（外部时钟）		30	–	–	ns
	复位脉冲宽度		15	–	–	ns
	复位脉冲宽度（外部时钟）		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，T_J ≤ 100 °C，电压范围为 1.71 V 到 5.5 V。

直流特性

参数	说明	最小值	典型值[1]	最大值	单位
I_{DD}	组件电流消耗				
	8位UDB递增计数器	–	10	–	µA/MHz
	8位有方向UDB计数器	–	10	–	µA/MHz
	16位递增计数器	–	16	–	µA/MHz
	16位有方向计数器	–	15	–	µA/MHz
	24位UDB递增计数器	–	31	–	µA/MHz
	32位UDB递增计数器	–	32	–	µA/MHz

交流特性

参数	说明	最小值	典型值	最大值[2]	单位
f_{CLOCK}	组件时钟频率				
	8位UDB递增计数器	–	–	39	MHz
	8位有方向UDB计数器	–	–	39	MHz
	16位递增计数器	–	–	33	MHz
	16位有方向计数器	–	–	33	MHz
	24位UDB递增计数器	–	–	29	MHz
	32位UDB递增计数器	–	–	26	MHz

1. 未包括设备 IO 和时钟分配的电流。这些都是在温度为 25 °C 时的值。
2. 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下运行该组件，此时需要验证时序要求和 STA 的结果。

组件更改

本节列出了该组件各版本中的主要更改内容。

版本	更改说明	更改原因/影响
2.40.b	更新了数据手册。	阐明了“基于UDB的组件”一节。 阐明了“输入/输出接口”一节（计数输入）。 阐明了WritePeriod API的其他影响。 在FIFO一节中添加了备注，使之更加清晰明了，并在“特性”一节中更新了一些数值，使之更加准确。
2.40.a	更新了默认配置波形。	计数器数据手册中显示的默认配置波形是错误的。
2.40	更新了数据手册和PSoC 4内存使用情况	
2.30	添加了“MISRA合规性”章节。	该组件没有任何特定偏差。
2.20	已添加了Counter_WriteCounter() API	固定功能计数器的Counter_WriteCounter() API的相关问题。
	更新了计数器数据手册中的默认配置波形。	计数器数据手册中显示的默认配置波形是错误的。
	更新了数据手册中Counter_Control寄存器（固定功能）的说明部分。	固定功能Counter_Control寄存器说明有错误。
	更新了数据手册中计数器配置窗口的截图	已对从计数寄存器窗口中移除频率计算字段进行了更新。
2.10	为UDB实现进行了自定义程序相关的更新。	现在，比较值可以采取递增或递减计数等模式下分辨率全部范围内的任何值。
	更新了Counter_RestoreConfig() API	用于修复与从低功耗模式唤醒后的中断触发器有关的问题。
	在数据手册中已经添加了PSoC 5直流和交流的固定功能电气特性	
2.0.a	更新了数据手册中的资源信息	
	在数据手册中已添加了其他配置章节	
2.0	重新设计了一个带有在所有模式中的过采样功能的同步计数器。	该器件的架构和工具已证明了：同步设计是唯一可行的解决方案。仍然支持所有模式，但需要一个过采样时钟。
	从固定功能实现中移除了“comp”终端。	该实现不支持比较输出。目前，该引脚已正确地隐藏。
	同步的输入	所有输入都是在固定功能实现的该模块输入同步的。

版本	更改说明	更改原因/影响
	已经将Counter_GetInterruptSource()函数转换为宏	Counter_GetInterruptSource()函数与Counter_ReadStatusRegister()函数的功能实现完全一致。为了节省代码空间，该函数已被转换为Counter_ReadStatusRegister()函数的宏。
	现在将输出寄存到组件时钟	为了避免组件输出上出现短时脉冲干扰，需要使所有输出同步。如果可能，在数据路径内部进行该操作，以避免使用过多资源。
	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，以便不被任何其他进程线程修改。
	向数据手册中添加了特性数据	
	对数据手册进行了少量的编辑和更新	

赛普拉斯半导体公司，2013-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）：（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。**在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。**赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。**针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。