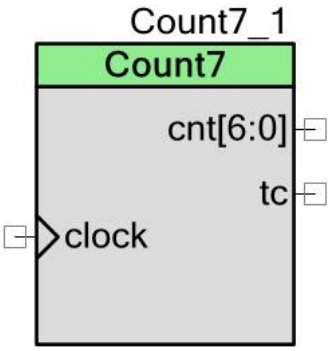
# 7 位的递减计数器（Count7）

1.0




## 特性

- 7 位读/写周期寄存器
- 7 位读/写计数寄存器
- 计数结束后，自动重新加载周期到计数寄存器
- 加载和使能信号可被连接控制

## 概述

Count7 组件是一个 7 位递减计数器，其计数值可作为硬件信号使用。通过通用数字模块（UDB）的特定配置，可以使用该计数器。在使用 UDB 中特定的计数器逻辑资源的同时，还使用了控制和状态寄存器中的某些资源来实现该计数器。

### 何时使用 Count7

PSoC 提供了多个可以实现计数器功能的组件。Count7 组件为 7 位计数器提供了最节约资源的实现方式。该计数器直接将计数值作为硬件信号，因此，Count7 能够用于状态机功能的硬件实现，计数器周期不超过 128 个周期。

## 输入/输出连接

本节介绍了 Count7 的各种输入和输出连接。I/O 列表中的星号（*）表示，在 I/O 说明部分所列出的内容中，该 I/O 可能不可见。

### en — 输入*

同步高电平有效使能信号。如果选中了 **EnableSignal** 参数的 **Enabled**（使能），此信号可用。当它可见时，将需要一个连接。高电平使能信号允许计数器在时钟上升沿递减计数。

## load — 输入*

同步高电平有效加载信号。如果选中 **LoadSignal** 参数的 **Enabled**（使能），此信号可用。当它可见时，将需要一个连接。如果使能了计数器，高电平加载信号会将周期值加载载入到计数器中。如果禁用了计数器，加载信号将无效。

## clock — 输入

组件时钟。组件可支持频率高达 67 MHz 的时钟源。

## reset — 输入*

异步高电平有效复位信号。如果选中了 **ResetSignal** 参数中的 **Enabled**（使能），则此信号可用。当它可见时，将需要一个连接。高电平信号立即将计数器的值清零。对一个被使能的组件，当复位信号返回低状态时，计数器将在时钟的下一个上升沿中加载周期值。

## cnt[6:0] — 输出

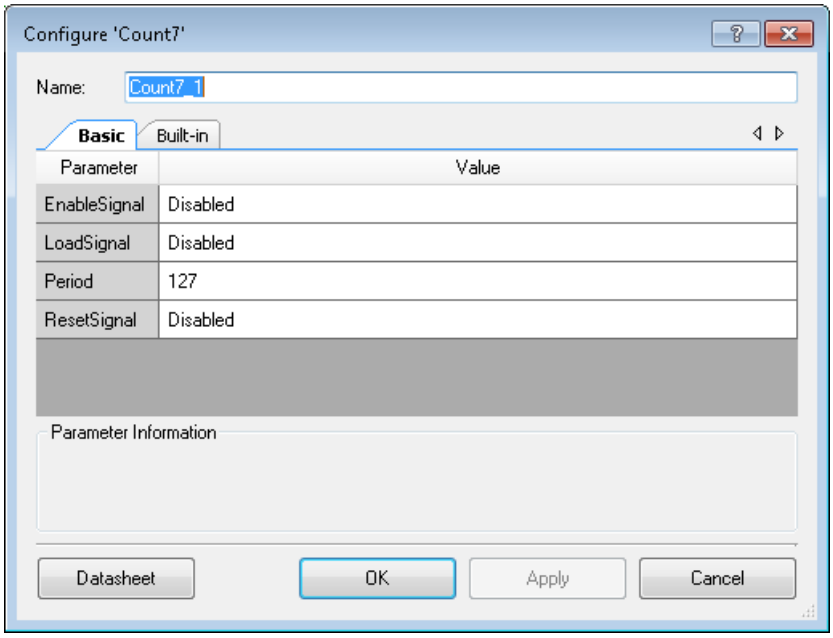
7 位计数器的值。

## tc — 输出

当计数器等于零时，终端计数输出高电平。如果使能了计数器，此输出会在单个周期内保持为高电平状态，此外，计数器会在下一个周期内重新加载周期值。如果周期计数器等于零，那么终端计数信号会保持高电平状态，直到周期计数器转为非零值为止。终端计数信号会被计数时钟同步，所以它比计数器值的变化延迟一个时钟周期。

# 组件参数

将 Count7 拖入您的设计中，双击它以打开 Configure（配置）对话框。




Count7 提供下列参数：

## 基本选项

### EnableSignal

使能硬件使能信号的连接。

### LoadSignal

使能硬件加载信号的连接。

### Period

定义初始的周期寄存器值。对于 N 个时钟的一个周期，周期值应设置为 N-1。计数器将从 N-1 向下计数递减至‘0’，这样会完成一个 N 时钟循环周期。数值为‘0’的周期寄存器不受支持，因此它使中断计数输出保持恒定的高电平状态。

### ResetSignal

使能异步复位信号的连接。

# 时钟选择

该组件中没有内部时钟。所以您必须添加一个时钟源。该组件的最高时钟频率可达 67 MHz。

# 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。下表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“Count7_1”分配给指定设计中组件的第一个实例。您可以将该实例重新命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。为便于阅读，下表使用了实例名称为“Count7”。

## 函数

| 函数 | 说明 |
|---|---|
| Count7_Start() | 对组件执行所有需要的初始化，并使能计数器。 |
| Count7_Init() | 根据定制器的设置，初始化或恢复组件。 |
| Count7_Enable() | 使能计数器的软件使能。 |
| Count7_Stop() | 禁用计数器的软件使能。 |
| Count7_WriteCounter() | 通过此函数，可直接写入计数器。调用此函数前，需要先禁用计数器。 |
| Count7_ReadCounter() | 此函数会读取计数器值。 |
| Count7_WritePeriod() | 通过此函数，可写入周期寄存器。 |
| Count7_ReadPeriod() | 此函数会读取周期寄存器。 |
| Count7_Sleep() | 这是使组件进入低功耗模式的首选API。 |
| Count7_Wakeup() | 这是将组件恢复到调用Count7_Sleep()时状态的首选API。 |
| Count7_SaveConfig() | 通过此函数可以保存在进入低功耗模式前组件计数寄存器中的值。 |
| Count7_RestoreConfig() | 通过此函数可以恢复先前保存的组件计数寄存器中的值。 |

## 全局变量

| 变量 | 说明 |
|---|---|
| Count7_initVar | 指示Count7是否已被初始化。该变量的初始化值为‘0’，并在第一次调用Count7_Start()时设置为‘1’。这样，第一次调用Count7_Start()子程序后，组件不用重新初始化即可重新启动。<br>如果需要重新初始化组件，则可以在 Count7_Start()或Count7_Enable()函数之前调用Count7_Init()函数。 |

## void Count7_Start(void)

**说明：** 对组件执行所有需要的初始化，并使能计数器。第一次执行子程序时，周期会被设置为定制器中的配置。调用Count7_Stop()后调用此变量以重新启动计数器时，当前的周期值仍被保留。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_Init(void)

**说明：** 根据定制器的设置，初始化或恢复组件。无需调用Count7_Init()，因为Count7_Start() API会调用此函数，这是开始组件操作的首选方法。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_Enable(void)

**说明：** 使能计数器的软件使能。计数器由软件使能和一个可选的硬件使能控制。无需调用Count7_Enable()，因为Count7_Start() API会调用此函数，这是启动组件操作的首选方法。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_WriteCounter(uint8 count)

**说明：** 通过此函数，可直接写入计数器。调用此函数前，需要先禁用计数器。

**参数：** uint8 count：要写入计数器内的值。
有效值范围为0至127。

**返回值：** 无

**其他影响：** 无

## uint8 Count7_ReadCounter(void)

**说明：** 此函数会读取计数器的值。

**参数：** 无

**返回值：** uint8：计数器的当前值。

**其他影响：** 无

## void Count7_WritePeriod(uint8 period)

**说明：** 通过此函数，可写入周期寄存器。实际周期比周期寄存器中的值大1，这因为计数序列以周期寄存器值开始，然后向下递减计数至0（包括0在内）。因终端计数值为零或硬件加载信号而导致计数器重新加载前，计数器输出的周期会保持不变。

**参数：** uint8 period：需要写入的周期值。
有效的值范围为0至127。

**返回值：** 无

**其他影响：** 无

## uint8 Count7_ReadPeriod(void)

**说明：** 该函数会读取周期寄存器。

**参数：** 无

**返回值：** uint8：当前的周期值。

**其他影响：** 无

## void Count7_Sleep(void)

**说明：** 这是将组件进入低功耗模式的首选API。Count7_Sleep() API使用Count7_SaveConfig()函数保存当前的组件状态，并禁用计数器。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_Wakeup(void)

**说明：** 这是将组件恢复到调用Count7_Sleep()时状态的首选API。Count7_Wakeup()函数通过调用Count7_RestoreConfig()函数恢复配置。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_SaveConfig(void)

**说明：** 此函数保存在进入低功耗模式前的当前计数器值。此函数由Count7_Sleep()函数调用。

**参数：** 无

**返回值：** 无

**其他影响：** 无

## void Count7_RestoreConfig(void)

**说明：** 此函数恢复先前保存的计数器值。此函数由Count7_Wakeup()函数调用。

**参数：** 无

**返回值：** 无

**其他影响：** 无

# MISRA 合规性

本节介绍了 MISRA-C:2004 兼容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。本节提供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。

Count7 组件没有任何特定偏差。

# 采样固件源代码

在“Find Example Project”对话框中，PSoC Creator 提供了大量的示例项目，包括原理图和代码的。要获取组件特定的示例，请打开组件目录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** 项来限定可选的项目列表。

有关更多信息，请参考 PSoC Creator 帮助中的“Find Example Project”（查找示例项目）主题。

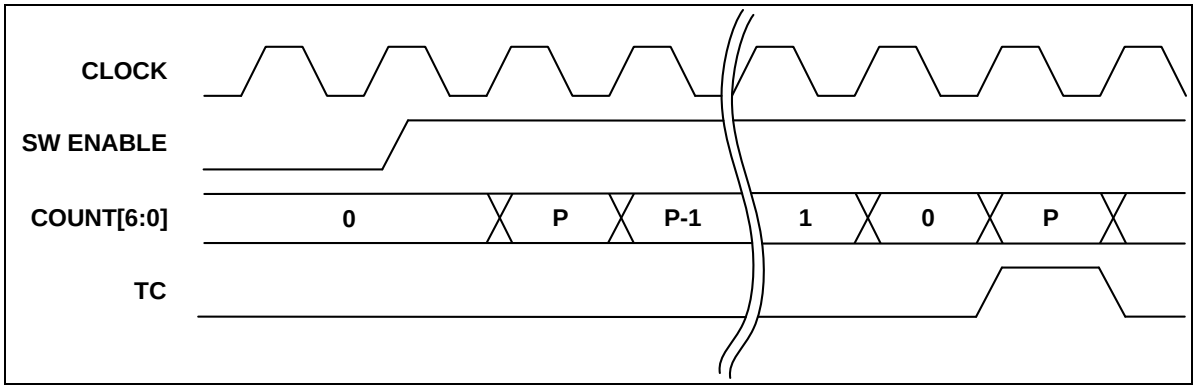
# 功能说明

每个 UDB 具有一个数据通道处理器、一个控制寄存器、一个状态寄存器以及两个 PLD。此外，还有一个额外的资源，即 Count7。Count7 是一个 7 位的递减计数器。它借用了 UDB 中其他部分的资源。具体包括，Count7 会使用控制寄存器和状态寄存器的掩码寄存器。如果采用了硬件负载或硬件使能，该计数器会使用状态寄存器所使用的输入。如果既不用硬件加载也不用硬件使能信号，那么，UDB 中的状态寄存器可以被使用，但中断功能不可用。

## 默认配置

默认情况下，Count7 组件仍有未显示在符号上的所有可选硬件输入，如 load（加载）、en（使能）和 reset（复位）。这是常见的使用情况。在此配置中，调用 Count7_Start()后，计数器会开

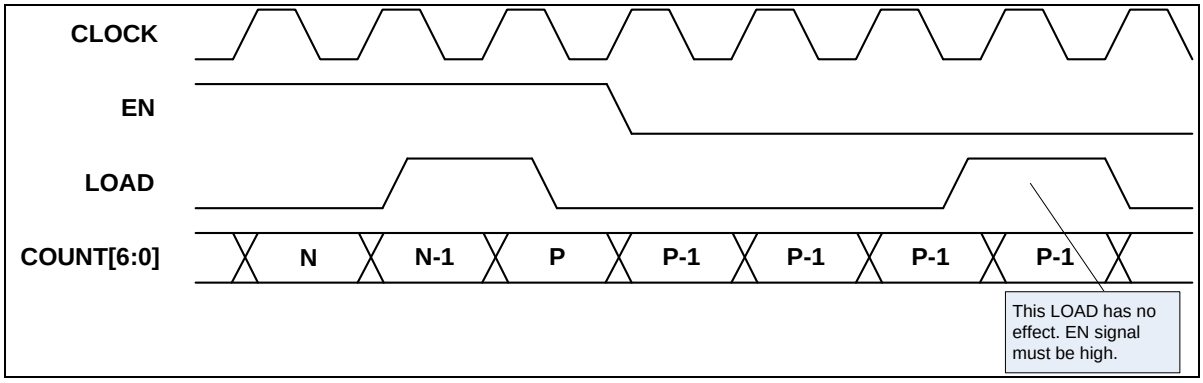
始计数，并在每个时长为一个周期终端计数脉冲的周期内循环各个计数值。在下面的框图中，SW ENABLE 为内部软件使能信号（由 Count7_Start()设置），P 值为已配置的周期值：




由于计数器从周期值向下递减计数至‘0’，所以整体周期会比周期寄存器中的值大 1。注意，计数值为零时，终端计数信号将在下一个时钟周期内设置为高电平。

### 使用可选的输入

可以选择性地使用一个硬件加载信号重载周期值。注意，如果未使能计数器，将忽略加载信号。




组件也可以选择性地使用异步复位输入。如果复位信号处于高电平状态，计数器的值会始终为零。在这种情况下，其他控制信号的状态不会起任何作用。

## 寄存器

Count7 组件使用状态和控制模块的计数和周期寄存器。不应该直接访问这些寄存器。而应该使用所提供的 API 函数实现该操作。

## 资源

Count7 组件放置在整个 UDB 阵列中。它只占用一个 Count7 单元，而不会使用其他任何资源。

# API 存储器大小

根据不同的编译程序、器件、所使用的 API 数量以及组件的配置情况，组件所用的存储空间大小也不一样。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。有关特定的设计，可分析编译器生成的映射文件以确定存储器的大小。

| 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 |
| 默认值 | 141 | 3 | 256 | 3 | 264 | 3 |

# 组件调试窗口

Count7 组件支持 PSoC Creator 组件调试窗口。Count7 组件调试窗口显示了以下寄存器。

## COUNT（计数）

此寄存器包含 COUNT 寄存器的当前值。

## PERIOD（周期）

此寄存器包含 PERIOD 寄存器的当前值。

# 直流和交流的电气特性

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。

## 直流规范

| 参数 | 说明 | 最小值 | 典型值[1] | 最大值 | 单位 |
|---|---|---|---|---|---|
| $I_{DD}$ | 模块电流消耗 | - | 2 | - | µA/MHz[2] |

[1] 当前未包括器件的 IO 和时钟分配的电流。这些值是在温度是 25 °C 时的值。

[2] 每 MHz 的输入组件时钟的电流消耗。

## 交流规范

| 参数 | 说明 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|
| $F_{clk}$ | 时钟频率 | - | - | 67[3] | MHz |

# 组件更改

本节列出了各版本的主要组件更改内容。

| 版本 | 更改内容 | 更改原因/影响 |
|---|---|---|
| 1.0 | 版本1.0是Count7组件的首次发行版 | |

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，WICED，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。

[3]Count7 组件能够在高达 67 MHz 的时钟频率操作。但是，PSoC 4 的最大时钟频率为 48 MHz。