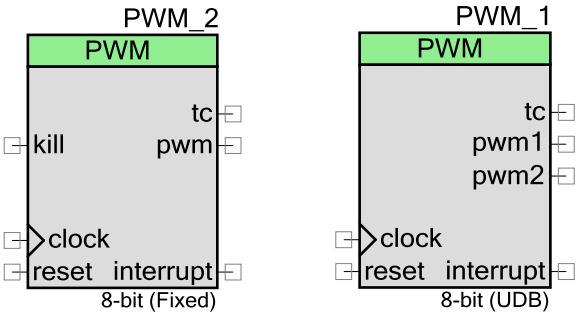
脉冲宽度调制器 (PWM)

2.10

特点



- 8 位或 16 位分辨率
- 多个脉冲宽度输出模式
- 可配置触发器
- 可配置捕获
- 可配置硬件/软件使能
- 可配置死区
- 多种可配置非同步停止输入模式
- 自定义配置工具

概述

PWM 组件提供了比较输出，通过硬件来产生单次或连续时序和控制信号。PWM 组件的设计提供了一种在 CPU 干预最少的情况下准确生成复杂实时事件的简便方法。PWM 组件可以与其他模拟和数字组件组合使用，以创建自定义外围设备。

PWM 最多可生成两个左对齐或右对齐 PWM 输出，或者一个中心对齐或双边 PWM 输出。PWM 输出经过双缓冲，以避免由于运行时的占空比更改而产生短时脉冲。左对齐 PWM 用于大多数常规 PWM 使用场合。右对齐 PWM 通常仅用于需要与左对齐 PWM 相反的对齐方式的特殊场合。中心对齐 PWM 大多用于交流电机控制以保持相位对齐。双边 PWM 最适用于必须调整相位对齐的电源转换场合。

可选死区提供了带有可调整死时间的互补输出，每次跃变之间这两个输出都很低。互补输出和死区时间通常用来驱动半桥电路中的功率器件，避免功率器件同时导通造成的电路短路和器件损坏。PWM 组件还提供了非同步停止输入，它在使能后可立即禁用死区输出（ph1 和 ph2）。提供了三种非同步停止输入模式，以支持多种使用场合。

提供了两种硬件颤振模式以提高 PWM 灵活性。当资源或时钟频率在 PWM 计数器中防碍标准实现时，第一种颤振模式可将有效分辨率提高两比特。第二种颤振模式使用数字输入逐周期地选择两种 PWM 输出中的一种，通常用于在电源转换中提供快速瞬态响应。

触发和复位输入允许 PWM 与其他内部或外部硬件同步。可选触发输入是可配置的，以便上升沿启动 PWM。复位输入中的上升沿会复位 PWM 的内部计数器。使能输入提供了硬件使能，以根据硬件信号使能/禁止 PWM 操作。

可以设置当 PWM 计数达到终端值 (terminal count) 或者比较器输出为高时产生中断。

何时使用 PWM

PWM 的最常见使用是生成具有可调整占空比的周期性波形。PWM 还为功率控制、电机控制、开关稳压器和照明控制提供优化功能。通过给 PWM 的时钟输入提供时钟源，并使用 PWM 的 tc 输出或 PWM 输出作为分频的时钟输出，还可以将 PWM 用作时钟分频器。

PWM、定时器和计数器有许多相同的功能，但是每个又具备了其他两个所没有的特定功能。计数器组件最适合于需要对大量事件计数的场合，但还提供了上升沿捕获输入和比较输出。定时器组件最适合于对事件长度定时、测量多个上升沿和/或下降沿的间隔的场合，或者用于多个捕获事件。

输入/输出连接

本节介绍 PWM 的各种输入和输出连接。某些输入/输出端口会根据 PWM 组件的配置隐藏或显现。

注意除非另有指定，否则所有信号都是高电平有效。

输入	可否隐藏	说明
时钟	N	时钟输入定义要计数的信号。计数器在时钟的每个上升沿递增或递减。
复位	N	将周期计数器复位为**周期**，然后继续正常操作。 **注意**在复位过程中，禁用 pwm、pwm1 或 pwm2 输出（驱动到“0”）。 对于固定功能实现的 PWM 模块（fix-function based），在复位过程中 PWM 输出为“1”。 在功能描述一章中的固定功能模块中的复位这一节，提供了具体的图示说明。 对于 PSoC 3 ES2 芯片，固定功能 PWM 的 tc 输出端在复位中保持高电平。为 PWM 输出提供的图示说明同样适用于 tc 输出端。对于PSoC3 Production 或更高版本的芯片，固定功能 PWM 的 tc 端在复位中保持低电平
使能	Y	使能输入与软件使能和触发器输入（如果使能了触发器输入）一起用于使能周期计数器。如果 **Enable Mode**（使能模式）参数设置为 **Software Only**（仅软件），则使能输入不可见。当选择 “固定功能” PWM 实现时，此输入不可用。
非同步停止输入	Y	非同步停止输入禁用 PWM 输出。提供了若干非同步停止输入模式，它们都依赖于此输入来实现输出信号的最终非同步停止输入。如果实现了死区，则仅禁用死区输出（ph1 和 ph2），不禁用 pwm、pwm1 和 pwm2 输出。如果 **Kill Mode**（非同步停止输入模式）参数设置为 **Disabled**（禁用），则非同步停止输入不可见。如果选择 “固定功能” PWM 实现并启用了死区，则非同步停止输入仅停止死区输出。对于 PSoC 5，当禁用死区时，非同步停止输入将不停止比较器输出。对于 PSoC 3，当禁用死区时，非同步停止输入将停止比较器输出。

输入	可否隐藏	说明
cmp_sel	Y	cmp_sel 输入选择 pwm1 或 pwm2 输出作为 pwm 终端的最终输出。如配置工具波形查看器所示，当输入为“0”（低电平）时，pwm 输出为 pwm1；当输入为“1”（高电平）时，pwm 输出为 pwm2。当 **PWM Mode**（PWM 模式）参数设置为 **Hardware Select**（硬件选择）时，cmp_sel 输入可见。
捕获	Y	捕获输入将周期计数器的当前值强制读取到 FIFO 中。模块为 **Capture Mode**（捕获模式）参数定义了几种不同的工作模式。如果 **Capture Mode**（捕获模式）参数设置为 **None**（无），则捕获输入不可见。当选择“固定功能” PWM 实现时，捕获输入始终与上升沿关联。
触发	Y	触发输入使能 PWM 的操作。此输入的功能由**Trigger Mode**（触发模式）和**Run Mode**（运行模式）参数定义。在触发功能使能后， PWM_Start() API 命令只启用 PWM，但是计数器不递减，直到发生触发条件。对于 “UDB ” 实现的 PWM ，使用输入时钟寄存触发输入，因此计数器在触发输入使能一个时钟周期后启动。触发条件是由**Trigger Mode**（触发模式）参数设置的。如果该参数设置为 **None**（无），则触发输入不可见。

输出	可否隐藏	说明
tc	N	当周期计数器等于零时，终端计数端 （tc端） 输出为“1”。在正常操作中，当周期寄存器重新加载计数值时此终端输出一个高脉冲。脉冲宽度为1个 PWM clock 时钟周期。如果 PWM 在周期计数器等于零时停止，则此信号保持高电平，直到周期计数器不再为零。此输出与 PWM 组件 的 clock 信号同步。
中断	Y	中断输出是所有可能的中断源的逻辑或（OR）。任何已使能中断源为 true 时，此信号将变为高电平。中断输出保留置位状态，直至软件读出状态寄存器。为了接收后续中断，中断产生后应当通过使用 PWM_ReadStatusRegister() API 读取状态寄存器来清除中断。仅当设置了 **Use Interrupt**（使用中断）参数时，中断输出才可见。这允许根据需要删除状态寄存器以实现资源优化。
pwm/pwm1	Y	pwm 或 pwm1 输出是第一个或唯一一个脉冲宽度调制输出。根据 **Configure**（配置）对话框中的波形所示，此信号由 **PWM Mode**（PWM 模式）、比较模式和比较值定义。如果实例是在 **One Output**（单输出）、**Dual Edge**（双边）、**Hardware Select**（硬件选择）、**Center Align**（中心对齐）或 **Dither**（颤振）PWM 模式下配置的，则输出“pwm”是可见的。否则，pwm1和另一个脉宽信号 pwm2 可见。此输出与 PWM 组件的 clock 信号同步。
pwm2	Y	pwm2 输出是第二个脉冲宽度调制输出。只有当 **PWM Mode**（PWM 模式）设置为 **Two Outputs**（双输出）时，pwm2 输出才可见。此输出与 PWM 组件的 clock 信号同步。
ph1/ph2	Y	ph1 和 ph2 输出是 PWM 的死区相位输出。在所有工作模式下，只有当 PWM 输出可见时，ph1/ph2才可能可见。在 **Two Outputs**（双输出）模式下，这些信号只是 pwm1 信号的相位输出。只有在死区设置为 "2 to 4 Clock Cycles" 或 "2 to 256 Clock Cycles"的情况下两个输出才可见；如果禁用死区，则它们不可见。此输出与 PWM 组件的 clock 信号同步。

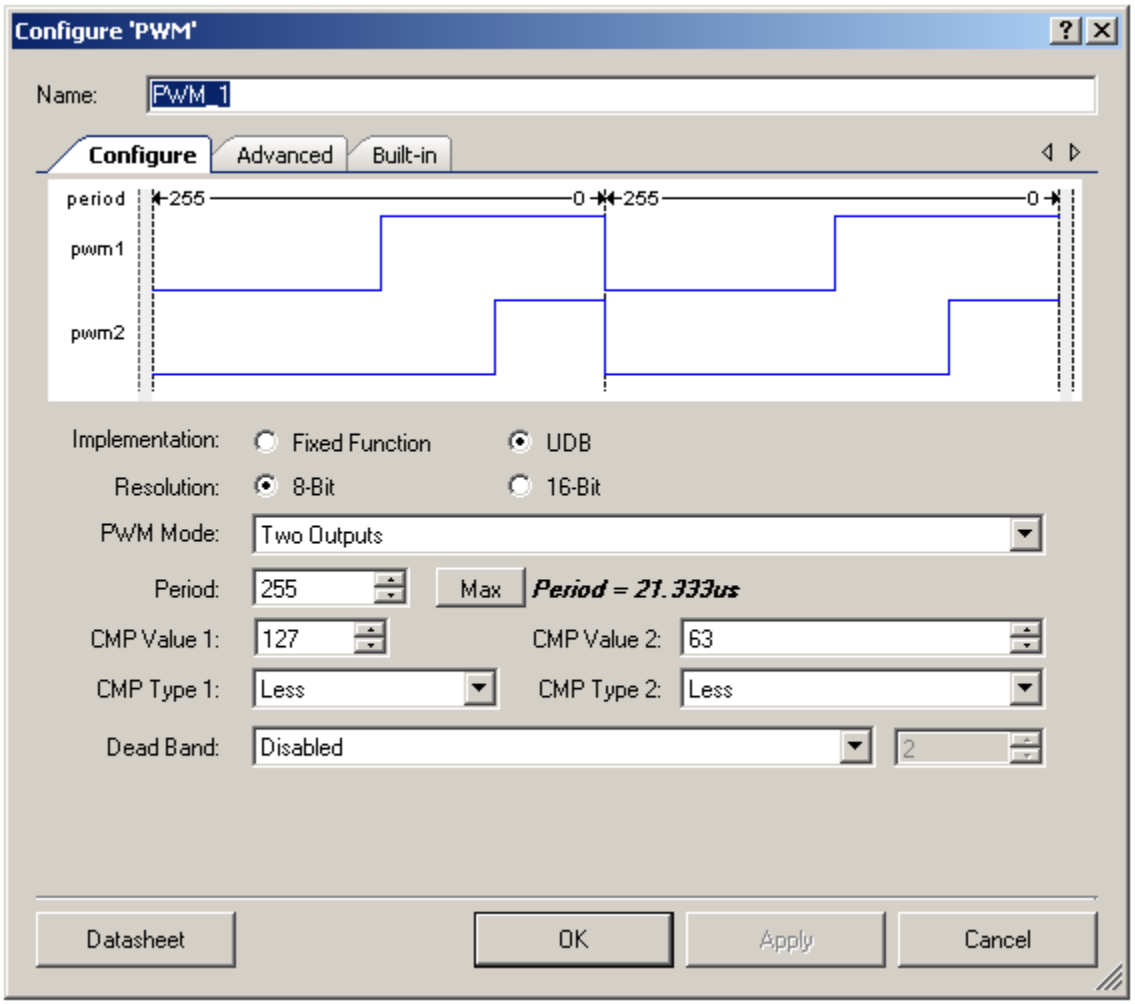
组件参数

将一个 PWM 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。**Configure PWM**（配置 PWM）对话框包含两个选项卡：**Configure**（配置）和 **Advanced**（高级）。

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，则必须重新构建硬件。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，这些初始值随时可能随提供的 API 修改。下面章节中描述的大多数参数是硬件选项。软件选项也将一同说明。

“配置”选项卡



实现

此参数允许您在 PWM 的 **Fixed Function**（固定功能）和 **UDB** 实现之间进行选择。如果此参数设置为 **Fixed Function**（固定功能），则 PWM 在具有关联限制的固定功能模块中实现。

分辨率

Resolution（分辨率）参数定义周期计数器的位宽度分辨率。

分辨率	最大周期计数值
8（默认值）	255
16	65,535

注意：如果 **PWM Mode**（PWM 模式）设置为 **Center Align**（中心对齐），则要求递增计数到周期值，然后递减计数到零，从而 PWM 周期将加倍。在此模式下，8 位 PWM 的限制为 254 循环（x2 = 508 循环），16 位 PWM 的限制为 65,534（x2 = 131,068 循环）。

PWM 模式

PWM 模式参数定义 PWM 的整体功能。如果 **Implementation**（实现）设置为 **Fixed Function**（固定功能），将禁用此参数。

此参数对组件图标上的可见引脚以及配置工具中显示的 pwm、pwm1 和 pwm2 的功能有很大影响。选项包括：

- **One Output**（单输出）— 只有单一 PWM 输出。在此模式下，pwm 输出可见
- **Two Output**（双输出） — 两个可单独配置的 PWM 输出。在此模式下，pwm1 和 pwm2 输出可见
- **Dual Edge**（双边）— 由与操作（ ANDing) 以及 pwm1 和 pwm2 信号创建的单一双边输出。在此模式下，pwm 输出可见。
- **Center Align**（中心对齐）— 根据比较值的创建一个中心对齐的脉冲波形，波形通过计数器递增计数到周期值，然后再递减计数到零，来创建单一中心对齐输出。在此模式下，pwm 输出可见。
- **Dither**（颤振）— 由 pwm 硬件实现中包括的硬件状态机从两个内部 pwm 信号（pwm1 和 pwm2）选择的单一输出。您可选择 0.00、0.25、0.50 或 0.75（具体含义后面所有介绍）选项来改变 PWM 的输出切换。两个 PWM 输出的切换选择由硬件控制实现。在这种情况下，比较值设置为“比较”和“比较 + 1”。在此模式下，pwm 输出可见。

- **Hardware Select**（硬件选择）— 硬件输入引脚 cmp_sel 从两个内部 pwm 信号选择的单一输出。当 cmp_sel 为低电平时，pwm1 信号是 pwm 输出引脚上的输出；当 cmp_sel 为高电平时，pwm2 信号是 pwm 输出引脚上的输出。在此模式下，pwm 输出可见。

周期（软件）

Period（周期）参数定义计数器的初始启动值，任何时候达到终端计数时，PWM 模式将允许重新加载周期计数器。

PWM 的周期计数器为 **Period**（周期）值到零计数的递减计数器。周期必须大于 1，且受 PWM 分辨率的上限限制。对于 8 位 PWM，周期值的最大值为 255。否则，周期值的最大值为 65535。当 PWM 模式配置为 Center Aligned（中心对齐）模式时，PWM 从零递增计数到周期值，然后递减计数到零。由于此特殊功能，中心对齐模式下的周期值是所有其他模式的两倍。可以随时通过 PWM_WritePeriod() API 更改周期值。该参数仅保留配置期间写入的初始值。

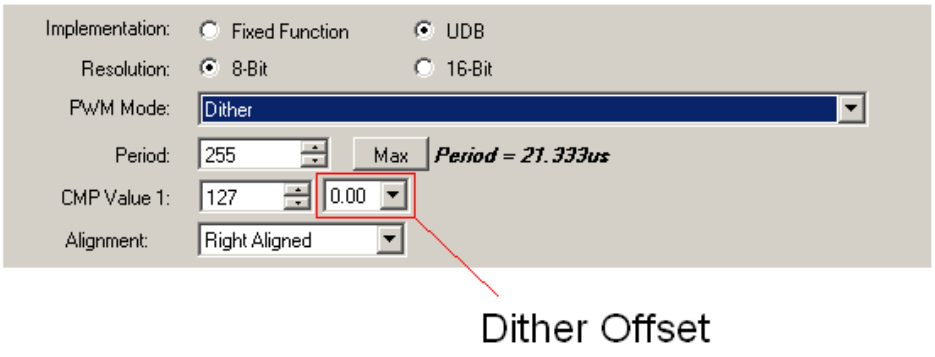
CMP Value 1 / CMP Value2（CMP 值 1 / CMP 值 2）（软件）

比较值与硬件**比较类型**选项一起定义比较输出功能。

比较值必须大于 1，且受 PWM 分辨率的上限限制。对于 8 位 PWM，比较值的最大值为 255。否则，比较值的最大值为 65535。比较值还受周期限制。当递减周期时，最大比较值设置为 **Period**（周期）— 1 以避免停用比较输出。可以随时通过调用 PWM_WriteCompare1() 和 PWM_WriteCompare2() API 更改比较值。这些参数仅保留配置期间写入的初始值。

颤振偏移

如果 PWM 是在 **Dither**（颤振）PWM 模式下配置的，**Dither Offset**（颤振偏移）参数配置 pwm 输出的功能。



颤振嵌入了内部状态机，以选择 pwm1 或 pwm2 输出作为最终 pwm 输出。pwm1 的比较值较 pwm2 小 1，其中 pwm1 适用于比较值，pwm2 适用于比较值+ 1。选项包括：

- **DO00** — 无颤振。输出始终为 pwm1。

- **DO25** — 0.25 颤振。对于四个周期计数中的三个周期计数，输出为 pwm1；对于一个周期计数，输出为 pwm2。
- **DO50** — 0.50 颤振。对于四个周期计数中的两个周期计数，输出为 pwm1；对于四个周期计数中的两个周期计数，输出为 pwm2。
- **DO75** — 0.75 颤振。对于四个周期计数中的一个周期计数，输出为 pwm1；对于四个周期计数中的三个周期计数，输出为 pwm2。

对齐

Alignment（对齐）参数在 **PWM Mode**（PWM 模式）设置为 **Dither**（颤振）时可用。选项包括：

- 右对齐
- 左对齐

CMP 类型 1 / CMP 类型 2（软件）

比较值参数定义两个构成 PWM 输出的周期计数器之间的比较关系。对于每个 PWM 模式，这些参数的实现方式不同，因此它们通常受配置工具的控制。两个比较模式参数中的每一个都可以独立设置成下列枚举类型之一。选项包括：

- **Less**（小于）— 如果周期计数器小于对应的比较值，则比较输出为 true。
- **Less or Equal**（小于或等于）— 如果周期计数器小于或等于对应的比较值，则比较输出为 true。
- **Greater**（大于）— 如果周期计数器大于对应的比较值，则比较输出为 true。
- **Greater or Equal**（大于或等于）— 如果周期计数器大于或等于对应的比较值，则比较输出为 true。
- **Equal**（等于）— 如果周期计数器等于对应的比较值，则比较输出为 true。
- **Firmware Control**（固件控制）— **Firmware Control**（固件控制）实现提供了更多资源用法模型，可用于在运行时期间设置比较模型。可以随时通过调用 PWM_WriteCompare1() 和 PWM_WriteCompare2() API 改比较模式。这些参数仅保留配置期间写入的初始模式。如果选择了除 **Firmware Control**（固件控制）之外的任何实现，则硬件经过预先配置，并在构建时固定在该配置。在这种情况下，WriteCompare API 会从编译中删除，因此不可用。

死区

Dead Band（死区）参数启用/禁用 PWM 的死区功能。死区模式在 “固定功能” 实现的 PWM 组件中略有不同。如果死区模式是两个已使能选项中的一项，则 ph1 和 ph2 输出可见。选项包括：

- **Disabled**（已禁用）— 无死区
- **0-3 Counts（0-3 个计数）**— 死区是在 pwm 或 pwm1 输出上实现的，最大值为 3 个计数。死区的实现是在 PLD 逻辑中实现，不使用数据路径（datapath） 模块。
- **2-4 Clock Cycles**（2-4 个时钟循环）— 死区是在 pwm 或 pwm1 输出上实现的，最大值为 4 个时钟循环。
- **2-256 Clock Cycles**（2-256 个时钟循环）— 死区是在 pwm 或 pwm1 输出上实现的，最大值为 256 个时钟循环。这个死区是使用数据路径（datapath） 模块实现的。

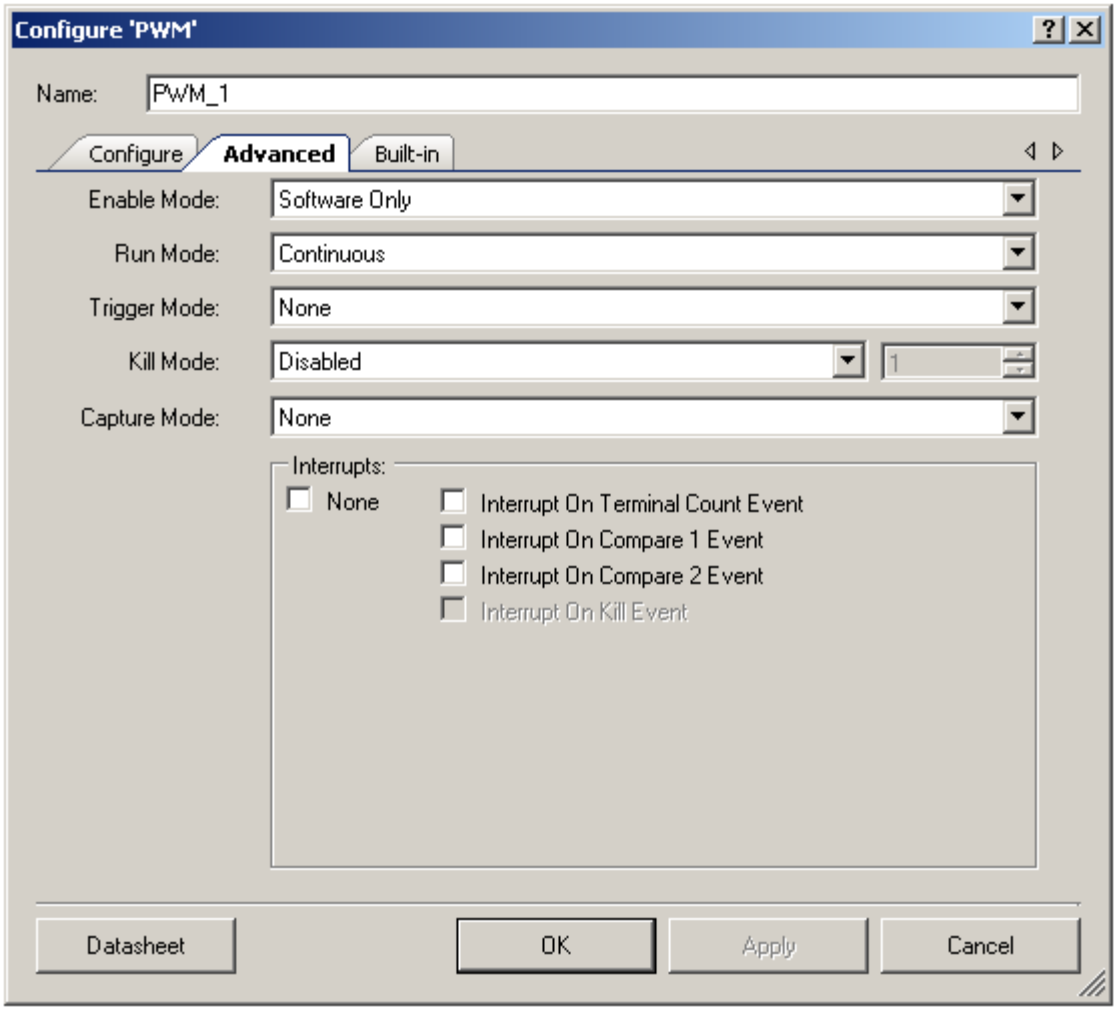
死区时间（软件）

死区时间值定义在死区输出信号 ph1 和 ph2 中实现的死区时间量。此参数仅在启用 **Dead Band**（死区）时才有效，并受限于 **Dead Band**（死区）参数中定义的硬件配置选项。



只有当使能了 “2 to 256 Clock Cycles” 选项是，死区时间选项才可配置。此数据受 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() API 控制。如果死区时间设置为 “2 to 4 Clock Cycles”, 死区时间由硬件控制，不可通过 API 函数更改。

“高级”选项卡



使能模式

Enable Mode（使能模式）参数用于定义需要使用哪个硬件和软件组合来使能 PWM 的整体功能。选项包括：

- **Software Only**（仅软件）— 只有当控制寄存器中的使能位由软件设置时，才使能 PWM。当使能模式设置为 **Software Only**（仅软件）时使能输入不可见。
- **Hardware Only**（仅硬件）— 只有当硬件使能输入有效（高电平）时，才启用 PWM。在此模式中，必须调用 PWM_Start() API，组件才能正确初始化以避免意外行为。
- **Hardware And Software**（硬件和软件）— 当控制寄存器和硬件输入中的位都有效（高电平）时，启用 PWM。

运行模式

Run Mode（运行模式）参数定义如何触发 PWM 以启动和继续运行。PWM 根据启用输入运行（如以下枚举值所述）。

- **Continuous**（连续）— 发生触发事件后 PWM 持续运行。
- **One Shot with Single Trigger**（单次触发）— 发生触发器事件后 PWM 仅运行一次，然后停止。下次触发需在组件重新 Reset 之后，否则触发不被识别。
- **One Shot with Multi Trigger**（多次触发）— 发生触发器事件后 PWM 运行一次。在每个周期完成时，PWM 暂停，直至发生下次触发器事件。

触发模式

触发模式参数用于定义有效触发事件的类型。当触发模式设置为 **None**（无）时，触发输入不可见。选项包括：

- **None**（无）— 不使能触发功能（触发始终视为 true）
- **Rising Edge**（上升沿）— 在触发输入的上升沿发出触发事件信号。
- **Falling Edge**（下降沿）— 在触发输入的下降沿发出触发事件信号。
- **Either Edge**（上升/下降沿）— 在触发输入的上升沿或下降沿发出触发事件信号。

非同步停止输出模式

Kill Mode（非同步停止输入模式）参数用于定义当硬件非同步停止输入模式输入有效时，硬件如何处理 pwm 输出。当非同步停止输入模式设置为 **Disabled**（已禁用）时，非同步停止输入不可见。选项包括：

- **Disabled**（已禁用）— 不启用非同步停止输入
- **Asynchronous**（异步）— 非同步停止输入有效时，禁用 pwm 输出。
- **Single Cycle**（单一循环）— 非同步停止输入有效时，禁用 pwm 输出，在到达周期结尾（即 tc）之前不会重新使能。
- **Latched**（锁存）— 非同步停止输入时禁用 pwm 输出，并保持禁用，直到 PWM 复位。
- **Minimum Time**（最小时间）— 非同步停止输入有效时禁用 pwm 输出，在经过最小时间后才会重新启用。

最小非同步停止输入时间（软件）

最小非同步停止输入时间参数用于定义当 **Kill Mode**（非同步停止输入模式）参数设置为 **Minimum Time**（最小时间）时，所需有效非同步停止输入信号的最小长度。



最小非同步停止输入时间值受 PWM_WriteKillTime() 和 PWM_ReadKillTime() API 控制，限制为值 1 到 255。

捕获模式

Capture Mode（捕获模式）参数用于定义什么硬件事件将驱使捕获当前周期计数器值的值并存入 FIFO。始终可以通过调用 PWM_ReadCounter() API 读取当前计数器值（即软件捕获）。当捕获模式设置为 **None**（无）时，捕获输入不可见。选项包括：

- **None**（无）— 不使能捕获
- **Rising Edge**（上升沿）— 在捕获输入的上升沿发出捕获事件的信号。
- **Falling Edge**（下降沿）— 在捕获输入的下降沿发出捕获事件的信号。
- **Either Edge**（任一沿）— 在捕获输入的上升沿或下降沿发出捕获事件的信号。

中断

Interrupts（中断）参数允许您配置初始中断源。这些中断的输出进行或（OR）操作，以提供最终的中断输出。只要 **Interrupts**（中断）未设置为 **None**（无），软件便可以随时重新配置此模式。此参数定义初始配置。

- **None**（无）— 不设置中断。
- **Interrupt On Terminal Count Event**（发生终端计数事件时中断）— 此选项始终可用；默认情况下会取消选择它。
- **Interrupt On Compare 1 Event**（发生比较 1 事件时中断）— 默认情况下会取消选择此选项。它始终处于显示状态。
- **Interrupt On Compare 2 Event**（发生比较 2 事件时中断）— 默认情况下会取消选择此选项。只有当为 **Implementation**（实现）选择了 **UDB** 且正确设置了 **PWM Mode**（PWM 模式）时，它才可用。

- **Interrupt On Kill Event**（发生非同步停止输入事件时中断）— 默认情况下会取消选择此选项。只有当为 **Implementation**（实现）选择了 **UDB** 且正确设置了 **PWM Mode**（PWM 模式）时，它才可用。

本地参数（供 API 使用）

这些参数在 API 中使用，未在 GUI 中公开。

- **FixedFunctionUsed** — 如果您已选择使用固定功能模块实现 PWM，则定义为“1”(true)。
- **KillModeMinTime** — 如果您将 **Kill Mode**（非同步停止输入模式）设置为 **Minimum Time**（最小时间），则定义为“1”(true)。如果有必要，这个参数可以被 PWM_WriteKillTIme() 和 PWM_ReadKillTime() 函数使用
- **PWMModeCenterAligned** — 如果您已将 **PWM Mode**（PWM 模式）设置为 **Center Aligned（中心对齐），则定义为“1”(true)。** 对于此模式，PWM_ReadCompare() 和 PWM_WriteCompare() 函数的定义方式与其他模式不同。此参数用于添加正确函数并删除不需要的函数。
- **DeadBandUsed**（使用的死区）— 如果您已选择使用 "2 to 256 Clock Cycles" 模式实现死区，则定义为“1”(true)。这用于有条件地包括 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() API 函数。
- **DeadBand2_4** — 如果您已选择 "2 to 4 Clock Cycles" 模式实现死区，则定义为“1”(true)。这是在处理死区时间不同操作的 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() 函数内部使用的。
- **UseStatus**（使用状态）— 当配置保证状态寄存器的使用时，定义为“1”(true)。这允许在设计中不需要状态寄存器资源时将其删除。
- **UseControl**（使用控制）— 当配置保证控制寄存器的使用时，定义为“1”(true)。这允许在设计中不需要控制寄存器资源时将其删除。
- **UseOneCompareMode**（使用单比较模式）— 当配置仅需要一个比较模式 API 可用时，定义为“1”(true)。这允许根据所选结构的定义来删除 API。

时钟选择

此组件中没有内部时钟。您必须附加时钟源。

警告：当 配置为“Fixed Function” 实现时，PWM 组件将具有下列限制：

- 时钟输入必须来自本地时钟，该本地时钟同步到总线时钟或直接源自总线时钟（将时钟类型配置为 Existing（现有），将源配置为 BUS_CLK）。
- 如果时钟频率与总线时钟匹配，则时钟必须直接连接到总线时钟（再次将时钟类型配置为 Existing（现有），将源配置为 BUS_CLK）。如果本地时钟具有与总线时钟匹配的频率，将在工程建立过程中导致错误。

对于基于 UDB 的组件

如果组件允许异步时钟，您可以使用器件频率范围内的任何时钟输入频率。

如果组件需要同步到总线时钟，则当使用路由时钟[1] 设置组件的时钟时，路由时钟的频率不能超过路由时钟源时钟频率的 1/2。

- 如果路由时钟与总线时钟同步，则它是总线时钟的 1/2。
- 如果路由时钟与时钟分频器之一同步，则其最大值为该时钟速率的 1/2。

放置

PWM 组件的放置基于组件实例的 **Implementation**（实现）参数选择。如果设置为 **Fixed Function**（固定功能），则 PWM 放置在固定功能模块之一中。如果设置为 **UDB**，则 PWM 放置在整个 UDB 阵列中，所有放置信息通过 *cyfitter.h* 文件提供给 API。

[1] 路由时钟不是直接连接到时钟输入的时钟符号。

资源

分辨率	数字模块					API 存储器（字节）		引脚（每个外部 I/O）
	数据路径（data path）	宏单元	状态寄存器	控制寄存器	计数器 7	闪存	RAM	
8 位 单输出模式[2]	1	6	1	1	0	226	5	-
8 位 双输出模式 2	1	9	1	1	0	237	5	-
8 位 双边模式 2	1	8	1	1	0	237	5	-
8 位 中心对齐模式 2	1	9	1	1	0	226	5	-
8 位 硬件选择模式 2	1	9	1	1	0	237	5	-
8 位 颤振模式 2	1	9	1	1	0	231	5	
16 位 单输出模式 2	2	6	1	1	0	275	5	1/2
具有死区的 PWM8 2-4 [3]	1	12	1	2	0	272	6	+2
具有死区的 PWM16 2-4 3	2	12	1	2	0	321	7	
具有死区的 PWM8 2-256 3	2	11	1	1	0	272	6	+2
具有死区的 PWM16 2-256 3	3	11	1	1	0	308	7	

[2] 配置 1：PWM 配置为相应的工作模式和分辨率，其他配置为：仅软件启动模式、连续运行模式、触发设置为无、禁用非同步停止输出功能、禁用捕获功能、不设置死区、使能 tc 中断。

[3] 配置 2：2-4 死区范围和 2-256 死区范围互斥。PWM 配置为单输出模式和相应的分辨率，其他配置为：仅软件启动模式、连续运行模式、触发设置为无、禁用非同步停止输出功能、禁用捕获功能、使能 tc 中断。

应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出了每个函数的接口定义，并进行了说明。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“PWM_1”分配给提供的设计中的第一个组件实例。可以将实例重命名为任何遵循标识符语法规则的唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。为增加可读性，下表中使用了实例名称“PWM”。

函数	说明
PWM_Start()	使用默认自定义的各项参数值初始化 PWM。
PWM_Stop()	禁用 PWM 操作。清除任一软件控制启用模式的控制寄存器的启用位。
PWM_SetInterruptMode()	配置中断源状态寄存器的中断掩码控制。
PWM_ReadStatusRegister()	返回状态寄存器的当前状态。
PWM_ReadControlRegister()	返回控制寄存器的当前状态。
PWM_WriteControlRegister()	设置控制寄存器的位字段。
PWM_SetCompareMode()	当设置为 Dither（颤振）模式、Center Align（中心对齐）模式或 One Output（单输出）模式时写入比较输出的比较模式。
PWM_SetCompareMode1()	将 compare1 输出的比较模式写入控制寄存器。
PWM_SetCompareMode2()	将 compare2 输出的比较模式写入控制寄存器。
PWM_ReadCounter()	读取当前计数器值（软件捕获）。
PWM_ReadCapture()	从捕获 FIFO 读取捕获值。
PWM_WriteCounter()	将新计数器值直接写入计数器寄存器。值只在当前的运行周期内生效。
PWM_WritePeriod()	写入 PWM 硬件所使用的周期值。
PWM_ReadPeriod()	读取 PWM 硬件所使用的周期值。
PWM_WriteCompare()	当实例定义为 Dither（颤振）模式、Center Align（中心对齐）模式或 One Output（单输出）模式时写入比较值。
PWM_ReadCompare()	当实例定义为 Dither（颤振）模式、Center Align（中心对齐）模式或 One Output（单输出）模式时读取比较值。
PWM_WriteCompare1()	为 compare1 写入比较值。
PWM_ReadCompare1()	读取 compare1 的比较值。
PWM_WriteCompare2()	为 compare2 写入比较值。
PWM_ReadCompare2()	读取 compare2 的比较值。
PWM_WriteDeadTime()	写入死区时间值

函数	说明
PWM_ReadDeadTime()	读取死区时间值。
PWM_WriteKillTime()	写入非同步停止输入模式设置为 Minimum Time（最小时间）时硬件所使用的非同步停止输入时间值。
PWM_ReadKillTime()	读取非同步停止输入模式设置为 Minimum Time（最小时间）时硬件所使用的非同步停止输入时间值。
PWM_ClearFIFO()	清除捕获 FIFO 中的所有捕获数据。
PWM_Sleep()	停止并保存用户配置。
PWM_Wakeup()	恢复并启用用户配置。
PWM_Init()	将组件参数初始化为图示中的自定义程序中设置的值。
PWM_Enable()	使能 PWM 模块操作。
PWM_SaveConfig()	保存组件的当前用户配置。
PWM_RestoreConfig()	恢复组件的当前用户配置。

全局变量

变量	说明
PWM_initVar	指示是否 PWM 已初始化。该变量初始化为 0，在第一次调用 PWM_Start() 时设置为 1。这允许第一次调用 PWM_Start() 子程式后组件无需重新初始化便可重新启动。 如果需要重新初始化组件，则可以在 PWM_Start() 或 PWM_Enable() 函数之前调用 PWM_Init() 函数。

void PWM_Start(void)

说明： 使用默认自定义程序值初始化 PWM。通过设置任一软件控制使能模式的控制寄存器的使能位来启用 PWM 操作。

参数： 无

返回值： 无

副作用： 设置 PWM 的控制寄存器中的使能位。如果 **Enable Mode**（启用模式）设置为 **Hardware Only**（仅硬件），则不对 PWM 产生任何影响。如果 **Enable Mode**（启用模式）设置为 **Hardware and Software**（硬件和软件），则只有当软件和硬件都被使能的情况下才可以启动 PWM。

void PWM_Stop(void)

说明： 通过复位任一软件控制使能模式控制寄存器的第七位，禁用 PWM 操作。禁用已选择的固定功能模块。

参数： 无

返回值： 无

副作用： 清除 PWM 的控制寄存器中的使能位。如果 **Enable Mode**（启用模式）设置为 **Hardware Only**（仅硬件），则此函数不对 PWM 产生任何影响。如果 **Enable Mode**（启用模式）设置为 **Hardware and Software**（硬件和软件），则将禁用此模式的软件部分，硬件输入不对 PWM 的使能产生任何进一步的影响。

void PWM_SetInterruptMode(uint8 interruptMode)

说明： 配置中断源状态寄存器的中断控制掩码。

参数： uint8 interruptMode：包含使能的中断源的位字段

返回值： 无

副作用： 无

uint8 PWM_ReadStatusRegister(void)

说明： 返回状态寄存器的当前状态。

参数： 无

返回值： uint8：当前状态寄存器值。状态寄存器的位有：

[7:6]：未使用 (0)

[5] ：非同步停止输入事件输出

[4] ：FIFO 非空

[3] ：FIFO 已满

[2] ：终端计数

[1] ：比较输出 2

[0] ：比较输出 1

副作用： 可以在读取时清除状态寄存器位。

uint8 PWM_ReadControlRegister(void)

说明： 返回控制寄存器的当前状态。只有当未删除控制寄存器时，此 API 才可用。

参数： 无

返回值： uint8：当前控制寄存器值

副作用： 无

void PWM_WriteControlRegister(uint8 control)

说明： 设置控制寄存器的位域。只有当未删除控制寄存器时，此 API 才可用。

参数： uint8 control：控制寄存器位位域，状态寄存器位有：

[7] ：PWM 使能

[6] ：复位

[5:3] ：比较模式 2

[2:0] ：比较模式 2

返回值： 无

副作用： 无

void PWM_SetCompareMode(enum comparemode)

说明： 当设置为 Dither（颤振）模式、Center Align（中心对齐）模式或 One Output（单输出）模式时写入比较模式。

参数： enum comparemode：比较模式枚举类型

返回值： 无

副作用： 无

void PWM_SetCompareMode1(enum comparemode)

说明： 将 compare1 输出的比较模式写入控制寄存器。

参数： enum comparemode：比较模式枚举类型

返回值： 无

副作用： 无

void PWM_SetCompareMode2(enum comparemode)

说明： 将 compare2 输出的比较模式写入控制寄存器。

参数： enum comparemode：比较模式枚举类型

返回值： 无

副作用： 无

uint8/16 PWM_ReadCounter(void)

说明： 读取当前计数器值（软件捕获）。

参数： 无

返回值： uint8/uint16：当前周期计数器值

副作用： 无

uint8/16 PWM_ReadCapture(void)

说明： 从捕获 FIFO 读取捕获值。

参数： 无

返回值： uint8/uint16：当前捕获值

副作用： 无

注意：在进入低功耗模式之后会清除 FIFO。必须先从捕获 FIFO 读取所有数据，然后才能进入低功耗模式（如果需要）。

void PWM_WriteCounter(uint8/16 period)

说明： 将新计数器值直接写入计数器寄存器。新的计数值在当前的运行周期写入，并只在当前周期内有效。

参数： uint8/uint16 period：周期计数器值

返回值： 无

副作用： 无

void PWM_WritePeriod(uint8/16 period)

说明： 写入 PWM 硬件所使用的周期值。

参数： period: uint8 或 16 取决于分辨率，新周期值

返回值： 无

副作用： 无

uint8/16 PWM_ReadPeriod(void)

说明： 读取 PWM 硬件所使用的周期值。

参数： 无

返回值： uint8/16：周期值

副作用： 无

void PWM_WriteCompare(uint8/16 compare)

说明： 当 **PWM Mode**（PWM 模式）参数设置为 **Dither**（颤振）模式、**Center Aligned**（中心对齐）模式或 **One Output**（单输出）模式时写入比较值。

参数： uint8/16：比较值

返回值： 无

副作用： 当 PWM 正在运行并且功能正常时使用 PWM_WriteCompare() API 可能会立即更改 PWM 输出，具体取决于指定的比较模式和比较类型。在将新比较值编程到比较寄存器时，比较输出会立即更改。如果启用了死区，则 PWM 输出的更改也会触发死区逻辑。

uint8/16 PWM_ReadCompare(void)

说明： 当 **PWM Mode**（PWM 模式）参数设置为 **Dither**（颤振）模式、**Center Aligned**（中心对齐）模式或 **One Output**（单输出）模式时读取比较值。

参数： 无

返回值： uint8/uint16：当前比较值

副作用： 只有当 **PWM Mode**（PWM 模式）参数设置为上述模式之一时，此函数才可用。否则必须调用 ReadCompare1/2 函数。

void PWM_WriteCompare1(uint8/16 compare)

说明： 为 compare1 写入新的比较值。

参数： uint8/uint16：pwm1 的新比较值

返回值： 无

副作用： 无

uint8/16 PWM_ReadCompare1(void)

说明： 读取 compare1 的比较值。

参数： 无

返回值： uint8/uint16：当前比较值 1

副作用： 无

void PWM_WriteCompare2(uint8/16 compare)

说明： 为 compare2 写入新的比较值。

参数： uint8/uint16：pwm2 的新比较值

返回值： 无

副作用： 无

uint8/16 PWM_ReadCompare2(void)

说明： 读取 compare2 的比较值。

参数： 无

返回值： uint8/uint16：当前比较值

副作用： 无

void PWM_WriteDeadTime(uint8 deadband)

说明： 为实现死区控制的硬件设置死区时间值。

参数： uint8：死区计数值

返回值： 无

副作用： 无

uint8 PWM_ReadDeadTime(void)

说明： 读取设置的死区时间值。

参数： 无

返回值： uint8：死区的当前设置值

副作用： 无

void PWM_WriteKillTime(uint8 killtime)

说明： 写入 **Kill Mode**（非同步停止输入模式）设置为 **Minimum Time**（最小时间）时硬件所使用的非同步停止输入时间值。

参数： uint8：非同步停止输入最小计数值

返回值： 无

副作用： 无

uint8 PWM_ReadKillTime(void)

说明： 读取 **Kill Mode**（非同步停止输入模式）设置为 **Minimum Time**（最小时间）时硬件所使用的非同步停止输入时间值。

参数： 无

返回值： uint8：当前非同步停止输入最小计数值

副作用： 无

void PWM_ClearFIFO(void)

说明： 清除以前捕获数据的 FIFO。在此处调用 PWM_ReadCapture()，直至 FIFO 为空。

参数： 无

返回值： 无

副作用： 无

void PWM_Sleep(void)

说明： 停止并保存用户配置。

参数： 无

返回值： 无

副作用： 无

void PWM_Wakeup(void)

说明： 恢复并启用用户配置。

参数： 无

返回值： 无

副作用： 无

void PWM_Init(void)

说明： 将组件参数初始化为图示中的自定义设置的值。通过设置控制寄存器的对应位，设置比较模式。中断被选作状态寄存器的输出。如果使用固定功能模式，则使能所选的固定功能模块。清除 FIFO，以使能要在状态寄存器中设置的 FIFO 满位。通常在 PWM_Start() 中调用。

参数： 无

返回值： 无

副作用： 所有寄存器将复位至初始值。这将重新初始化该部件。

void PWM_Enable(void)

说明： 通过设置控制寄存器的第七位启用 PWM 模块操作。

参数： 无

返回值： 无

副作用： 无

void PWM_SaveConfig(void)

说明： 保存组件的当前用户配置。保存周期寄存器、死区寄存器、计数器寄存器和控制寄存器值。

参数： 无

返回值： 无

副作用： 无

void PWM_RestoreConfig(void)

说明： 恢复组件的当前用户配置。

参数： 无

返回值： 无

副作用： 无

固件源代码示例

PSoC Creator 在 Find Example Project（查找示例项目）对话框中提供了许多包括原理图和示例代码的示例项目。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。要获取通用的示例，请打开开始页或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（滤波器选项）可缩小可选项目的列表。

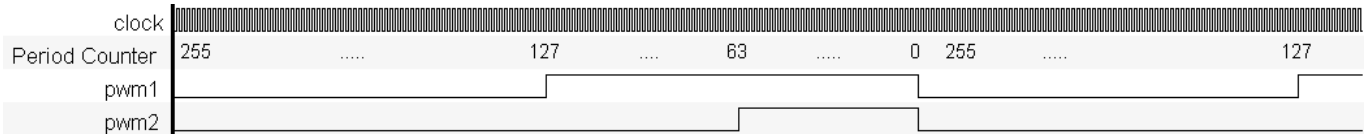
有关更多信息，请参考 PSoC Creator 帮助中的“查找示例项目”主题。

功能描述

默认配置

PWM 的默认配置是一个双输出 8 位 PWM，它将创建一个具有小于 127 的比较值的输出（周期为 255），以及另一个使用 12-MHz 时钟的小于 63 的输出。图 1 显示 PWM 保留在默认配置中时的输入和输出。

图 1. PWM 输入和输出



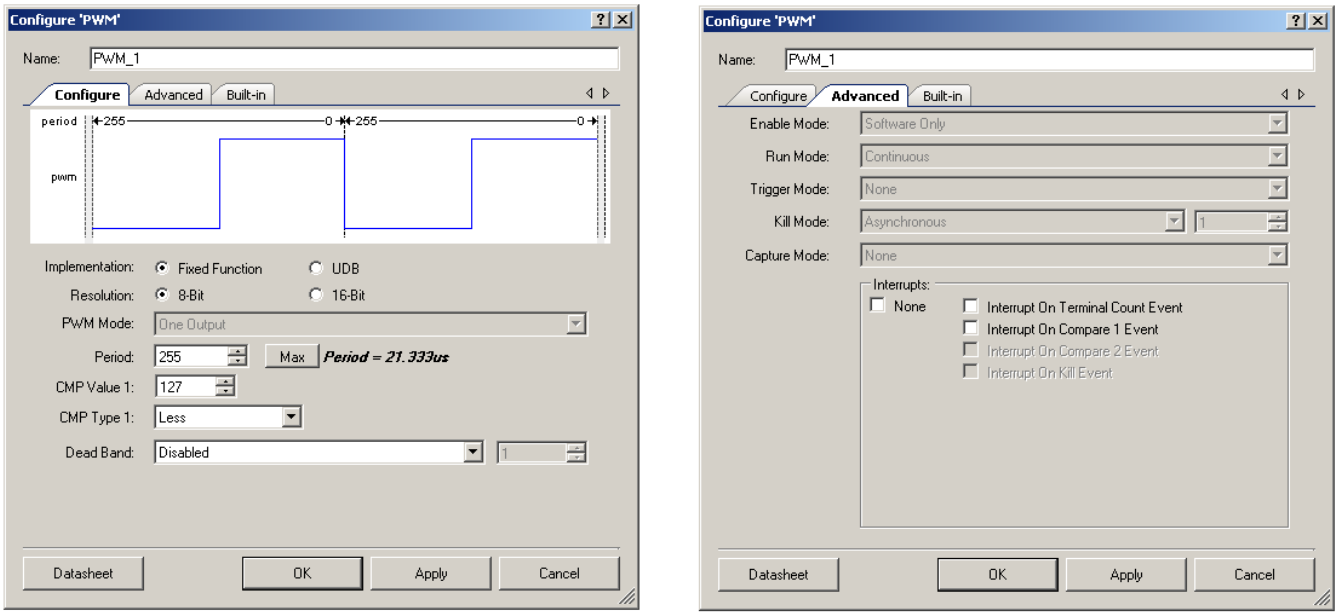
固定功能模块限制

使用 “Fixed Function” 实现的 PWM 组件是基于一个可配置的硬件模块。这个选项可以减少 UDB 模块的使用量但是实现的 PWM 功能有很多限制。这些模块之一中的 PWM 功能具有下列限制：

- 没有计数器值的访问权 — PWM_ReadCapture() 和 PWM_ReadCounter() 不可用。
- 仅限 One output（单输出）模式 — 没有 Center Align（中心对齐）、Dual Edge（双边）、Dither（颤振）或 Two Outputs（双输出）模式
- 仅限异步非同步停止输入模式
- 没有触发输入
- 仅限连续运行模式
- 仅限软件使能用 — 没有硬件使能模式
- 减少了死区功能 — 限制为 0 到 3 个时钟周期
- 当使用死区功能是减少了模块的输入输出 — TC 和 CMP1 相应变成 PH1 和 PH2。

当您选择 **Fixed Function**（固定功能）实现时，**Configure**（配置）选项卡对话框和 **Advanced**（高级）选项卡对话框中的诸多选项被禁用来指明这些限制，如图 2 所示。

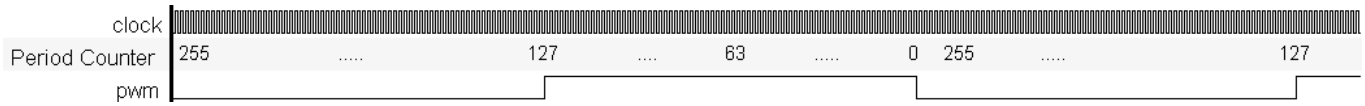
图 2. 固定功能设置



PWM 模式

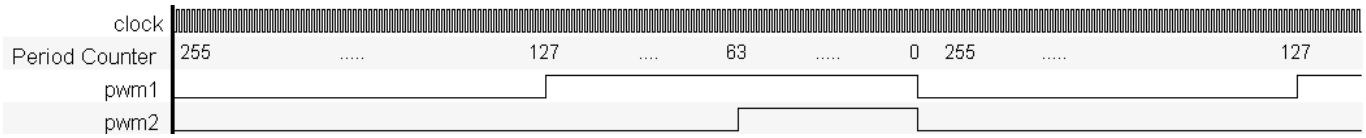
单输出

单输出 PWM 只有一个受单一比较值和单一比较模式控制的输出。对于 **Greater**（大于）或 **Greater or Equal**（大于或等于）比较模式，此波形可以左对齐；对于 **Less**（小于）或 **Less or Equal**（小于或等于）比较模式，此波形可以右对齐。



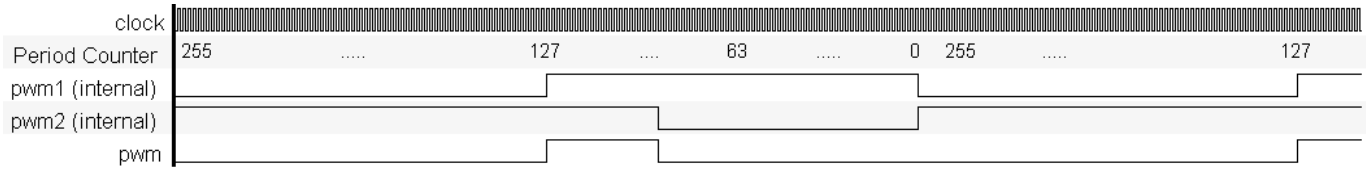
双输出

双输出 PWM 是默认配置。这两个 PWM 输出是使用两个比较值和两个比较模式彼此独立定义的。这两个输出中的每一个都可以按照以前在 单输出（单输出）模式中所述进行左对齐或右对齐。



双边

双边 PWM 使用两个比较输出和两个比较模式来生成单一 PWM 输出。最终输出是两个比较值和比较模式所定义的两个不同信号的“与”运算结果。此模式要求您对不同模式生成的结果有一些了解。参数编辑自定义程序中的波形示例提供了有关最终波形形状的帮助。但是，比较值、比较模式和周期值都可以在运行时设置。在不了解最终配置的情况下更改这些值可能很容易形成 0 值输出。

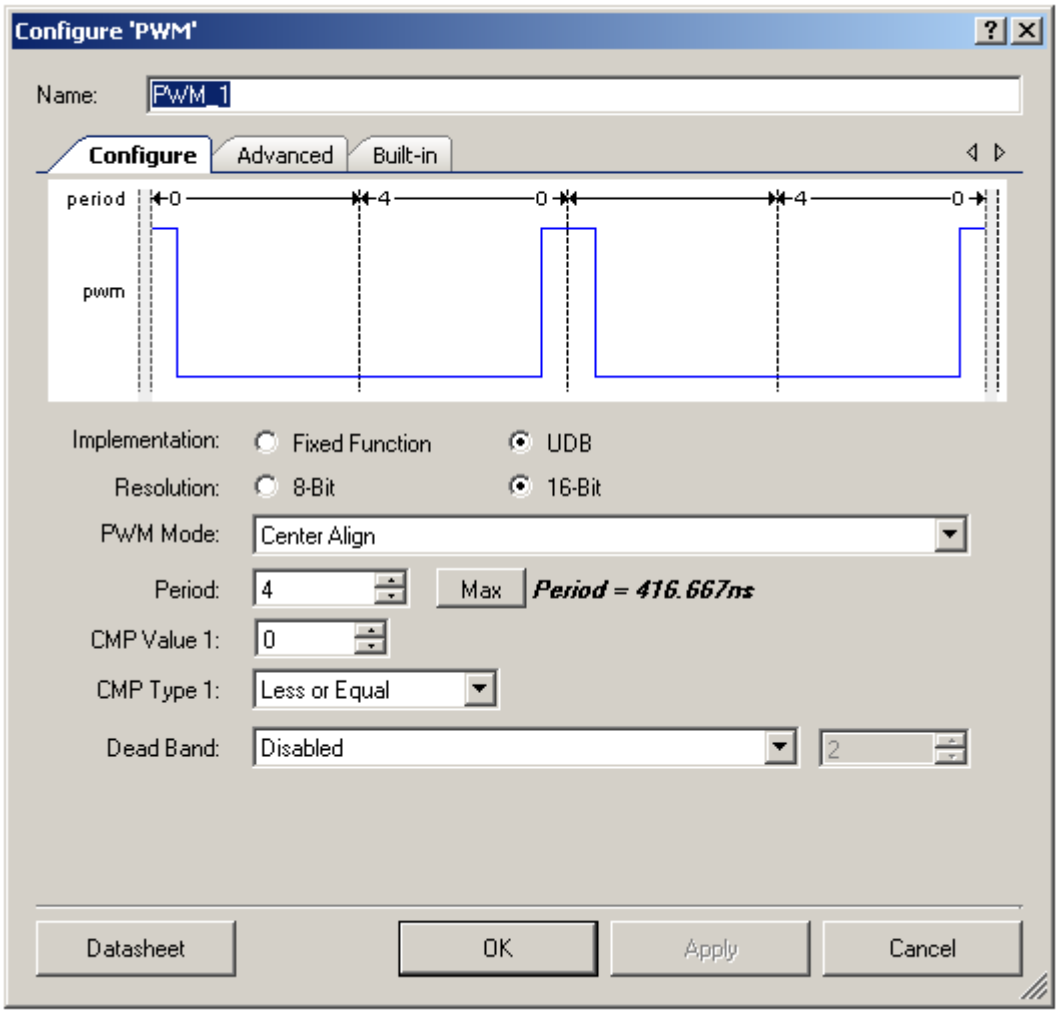


中心对齐

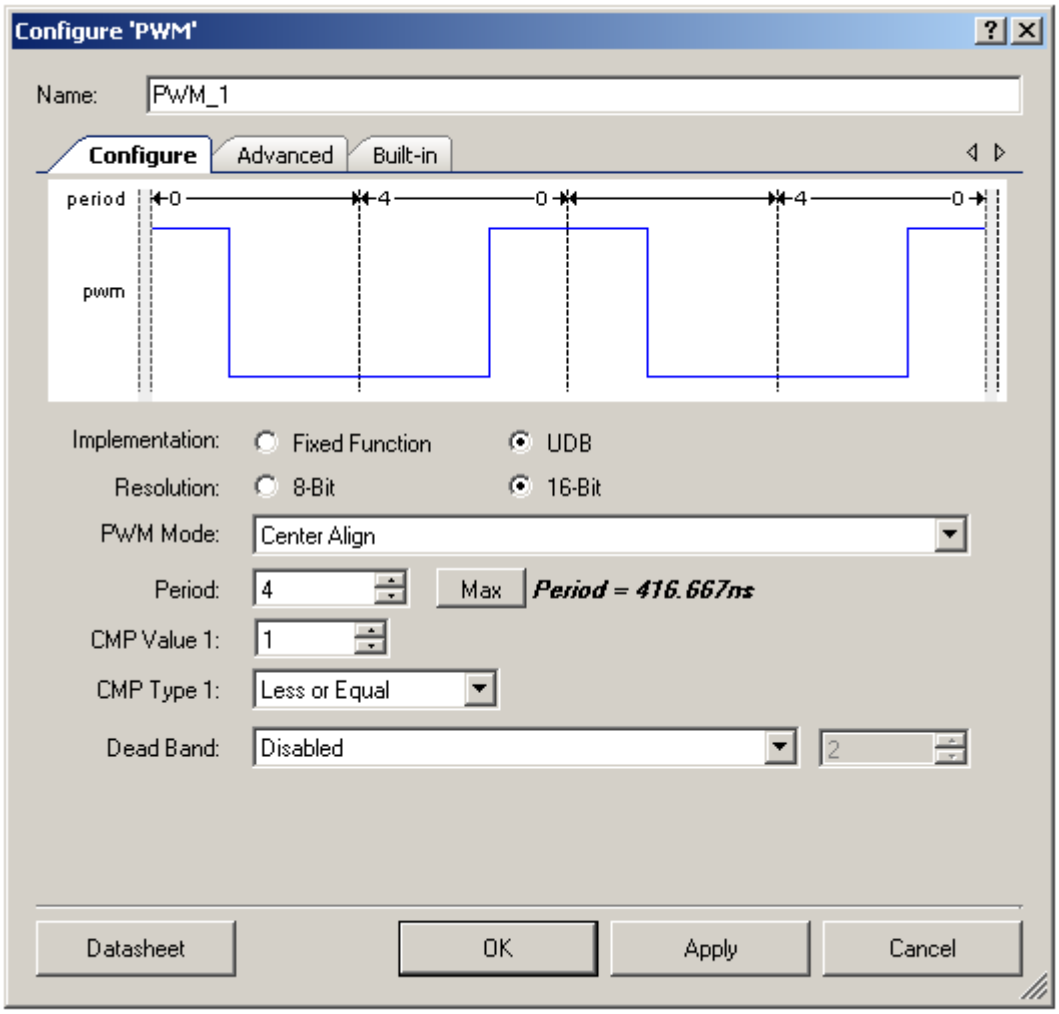
中心对齐 PWM 实现 PWM 的方式与所有其他模式不同。所需的输出要求周期计数器从零开始，递增计数到周期值，当达到周期值时，计数器将开始倒计数，直到零为止。在此模式下，周期值实际是最终输出的周期的一半。此功能可以使用单一比较值和比较模式。

PWM 的所有其他模式在周期设置递减计数到 0 时启动周期计数器，重新加载到周期值，得到实际周期时间的“周期 + 1”结果。自定义程序中显示的计算出的周期中表明了这一点。对于中心对齐模式，计算出的周期不 + 1。这是因为周期计数器从 0 计数到周期，然后立即开始递减计数。例如，如果计数器计数周期为 4，则 0、1、2、3、4、3、2、1、0、1、2… 构成周期 4 时钟循环。

自定义程序在自定义程序中显示的波形的开头和结尾分别用半位时间来表示这一结果。



在此配置中，对于在第一个周期开头和结尾表示为半位时间的单一时钟循环，计数器只是小于或等于零。下图表示，对于在波形所示的两个周期中每个周期开头和结尾表示为 1.5 时钟循环的 3 时钟循环而言，计数器小于或等于 1。



硬件选择

硬件选择 PWM 实现为两个输出 PWM，该实现具有两个独立比较值和比较模式。硬件输入 cmp_sel 选择两个输入中的哪一个输入将作为最终 PWM 输出。这允许您根据需要在两个预先配置的值之间切换，无需修改参数。

颤振

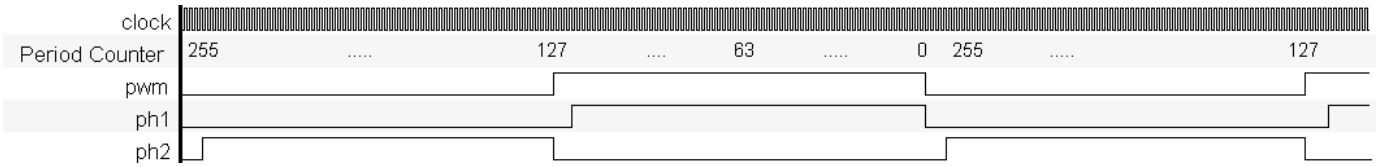
颤振模式 PWM 可以说是硬件选择模式 的一种：它的特点是两个输出比较值相差为 1，两个比较模式相等。用于控制硬件选择的是一个内置的状态机。在此模式下，cmp_sel 输入不可用。您可以将偏移设置为 0.00、0.25、0.50 和 0.75，参数字段在此模式下可见。如果偏移配置为 0.00，输出始终为 compare1 输出。当设置为 0.25 时，对于三个循环而言，输出为 compare1，对于单一循环而言，输出为 compare1 + 1。

颤振模式	循环 0	循环 1	循环 2	循环 3
0.00	Compare1	Compare1	Compare1	Compare1
0.25	Compare1 + 1	Compare1	Compare1	Compare1
0.50	Compare1	Compare1 + 1	Compare1	Compare1 + 1
0.75	Compare1 + 1	Compare1 + 1	Compare1 + 1	Compare 1

死区

死区是上述任意 PWM 模式的附加选项。当启用死区时，两个新输出 ph1 和 ph2（phase1 和 phase2）在配置界面中可见。死区输出适用于单一 PWM 输出。在除双输出模式之外的所有模式下，死区输出与单一 PWM 输出相关。在双输出模式下，死区仅在 pwm1 输出上实现。在所有死区模式下，提供了原始输出以及 ph1 和 ph2 输出。

死区的死区时间可以配置为具有 2 到 4 或 2 到 256 时钟周期范围。提供 2 到 4 时钟周期范围的目的是通过在 PLD 中实现计数器（而不是使用 datapath 模块）来减少资源使用。当选择 2 到 256 时钟周期范围死区时，死区时间是由 UDB 中的 datapath 以及必要的逻辑来实现的。



非同步停止输出模式

与死区类似，非同步停止输入模式是一个附加函数，它不在内部中断 PWM 的输出。此附加函数放置在 PWM 的输出，仅控制最终输出信号。如果不实现死区，则非同步停止输入操作通过将 PWM 输出拉至低电平来禁用 PWM 输出。如果实现死区，则非同步停止输入操作通过将 ph1 拉至低电平并将 ph2 拉至高电平来禁用 ph1 和 ph2。

异步

在异步非同步停止输入模式下，当非同步停止输入有效（高电平）时禁用输出，只要非同步停止输入变为无效，便重新启用输出。

单周期

在单周期非同步停止输入模式下，当非同步停止输入有效（高电平）时禁用输出，在下一周期的开始处重新启用输出。

锁存

在锁存非同步停止输入模式下，当非同步停止输入变为高电平时禁用输出。在 PWM 复位后，如果非同步停止输入仍未变为有效，则将重新启用 PWM 输出，否则它们将保留在非同步停止输入状态，直到使用非有效的非同步停止输入再次复位 PWM 为止。

最小时间

在最小时间非同步停止输入模式下，在非同步停止输入有效（高电平）时，会禁用输出。如果非同步停止输入不再有效，则在经过最小时间之后，会重新启用输出。对于此模式，您使用时钟计数 1 到 255 范围内的数字来定义最小非同步停止输入时间。只有当选择此非同步停止输入模式时，控制最小非同步停止输入时间计数所需的 API 才可用。

运行模式

连续

连续运行模式是 PWM 的默认配置。此模式允许 PWM 在使能后永远运行。只要启用 PWM，输出将不断重复循环周期以实现指定的脉冲宽度输出。

单次触发

单次触发运行模式将在发生有效触发事件时触发 PWM 运行一个周期。在周期完成之后，PWM 会停止。在周期寄存器中的值重新加载之后，PWM 便停止。硬件复位脉冲会重新激活 PWM 并允许下一个触发器事件使 PWM 运行其他周期。

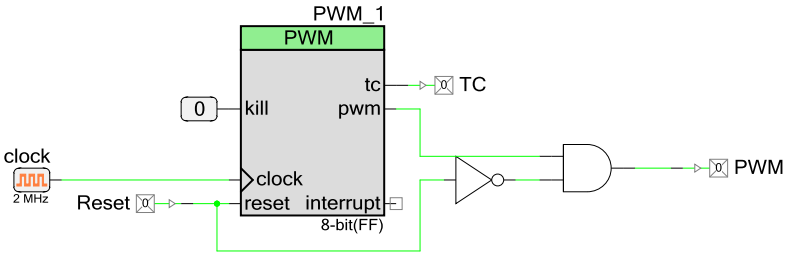
多次触发

多次触发运行模式将在发生有效触发事件时触发 PWM 运行一个周期。在周期完成之后，PWM 会停止，在等到下一个触发器事件时重新激活 在使用周期寄存器中的值重新加载计数器之后，PWM 会停止。单次触发模式与多次触发模式的区别在于，在无需硬件复位的前提下，多次触发可以被触发信号重新激活输出

固定功能模块中的复位

“固定功能” 实现的 PWM 组件与 UDB 实现的不同之处是，复位期间 pwm 输出变为高电平，而 UDB 实现的 PWM 组件 pwm 输出变为低电平。如图 3 所示，很容易更改这两个实现中任一实现的功能。图 3 显示的 “固定功能” 实现的 PWM 在复位输入有效时将 PWM 输出拉至低电平，因而提供了与同一组件的 UDB 实现相同功能的电路。

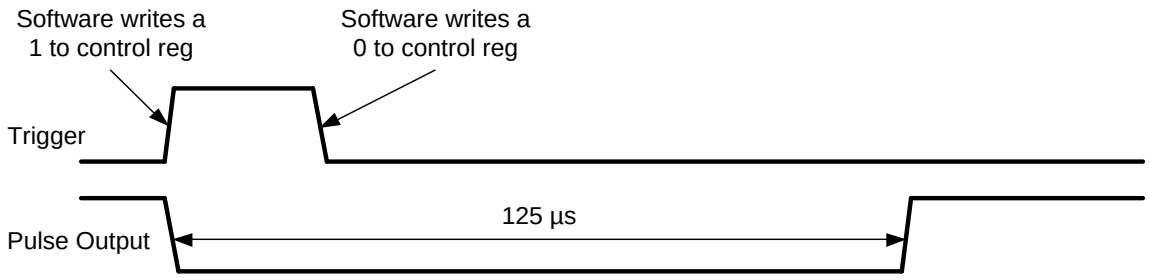
图 3. 固定功能 PWM 实现原理图



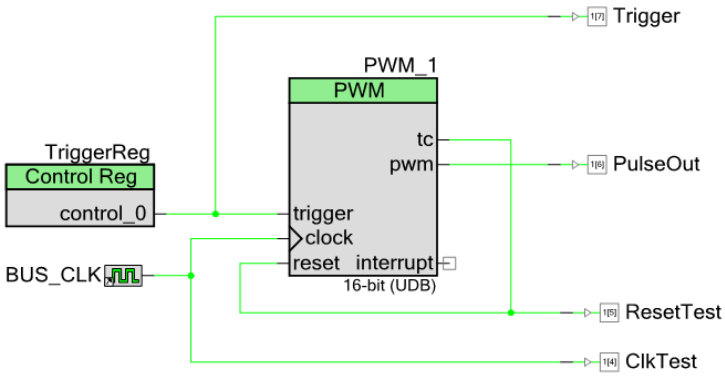
最好在 “固定功能” 的 PWM 组件上实现此更改，因为它只有一个要处理的输出，而 UDB 实现的模式中具有两个输出。在大多数情况下，还最好在复位期间将输出拉至低电平。

PWM 组件 用作脉冲发生器

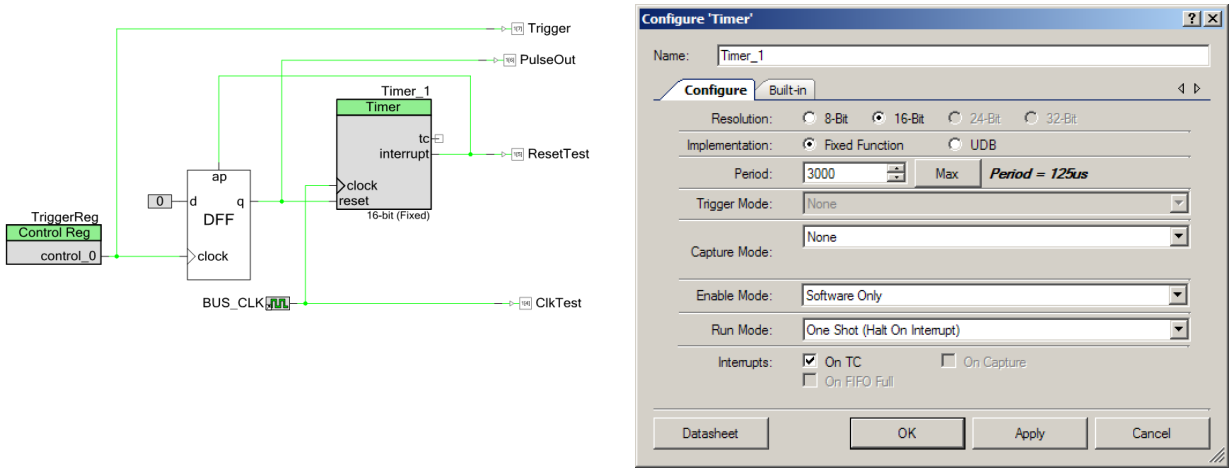
PWM 组件可以用于设计软件触发的脉冲发生器电路，以生成已知周期的时序脉冲。下面的时序图介绍了一个脉冲生成应用示例，该示例在软件触发器上生成 125 µs 的时序脉冲。



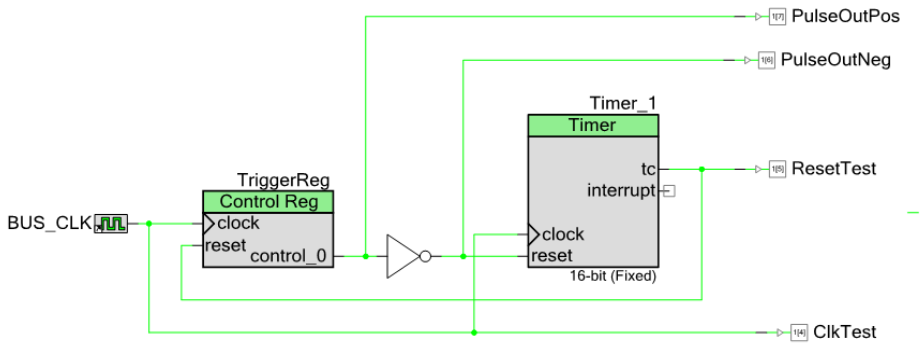
在单次触发模式下配置的 PWM 可以与控制寄存器组件一起用于实现此设计。在单次触发模式下，PWM 应在其达到周期值之后复位，以确保其正确运行。可以通过将 TC 输出连接到其复位输入来实现此目的。下面的原理图提供了 UDB 实现的 PWM 组件实现同样功能的电路图。



前面的设计是 UDB 资源密集型。可以选择固定功能实现，如下面的原理图所示。此设计使用一个附加 DFF 组件，并且（与前面的实现不同）不使用任何数据路径元素。



在 PSoC 3 Production 中，可以按不同方式配置控制寄存器组件，以便去掉 DFF。在此配置中，只需调用控制寄存器写入函数一次以写入 1。最终使用 PSoC3 实现时序脉冲发生器的电路如图所示。

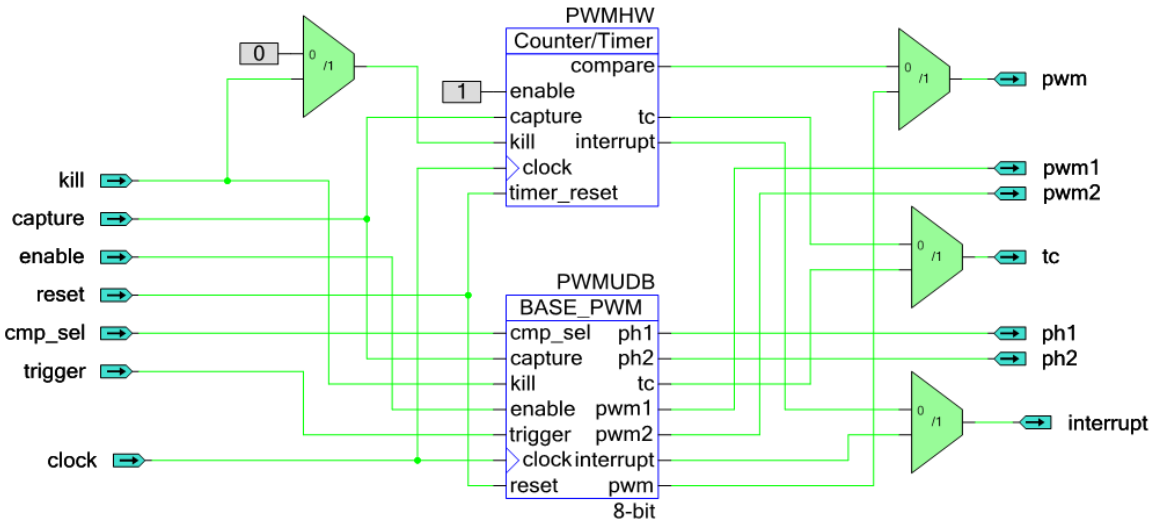


框图和配置

可以使用固定功能模块或 UDB 组件实现 PWM。通过高级参数 **Implementation**（实现），您可以指定期望将此组件放入的模块。固定功能实现将使用计时器/计数器/PWM 模块之一。在固定功能或 UDB 配置中，所有寄存器和 API 都进行了合并，以给出一个整体的外观。上面已描述了 API，此处将描述寄存器以定义 PWM 的整体实现。

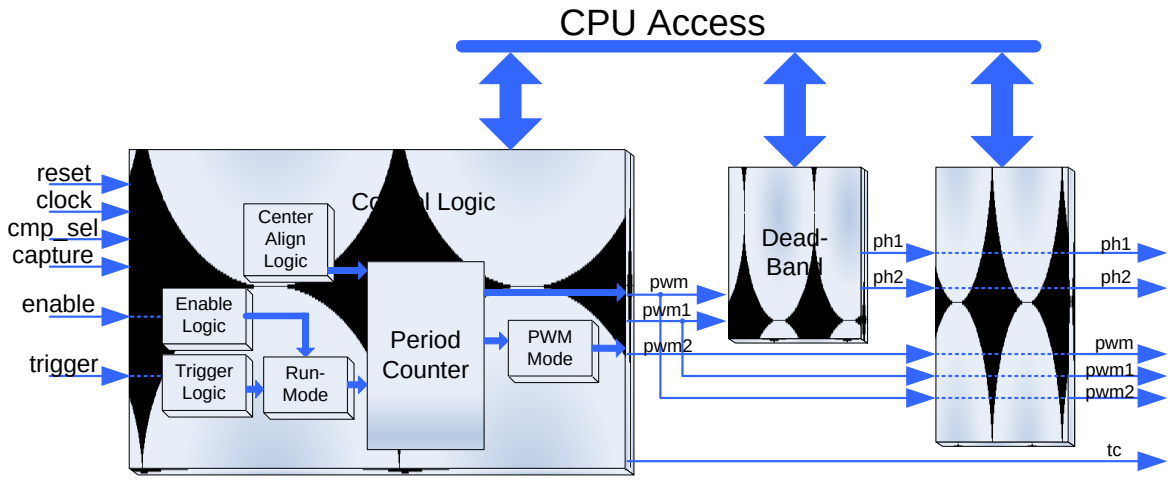
选择的两个硬件实现选自顶层原理图，如图 4 所示。

图 4. 顶层原理图



此配置允许选择固定功能模块或 UDB 实现。将在后台处理 I/O 的路由以给出此单一组件外观。图 5 中介绍了 UDB 实现。

图 5. UDB 实现



寄存器

状态

状态寄存器是只读寄存器，包含为 PWM 定义的各种状态位。调用 PWM_ReadStatusRegister() 函数为您提供此寄存器的值。中断输出信号（中断）是从此寄存器内的掩码位域的 OR 运算生成的。可以使用 PWM_SetInterruptMode() 函数设置掩码，在收到中断时，可以通过使用 PWM_ReadStatusRegister() 函数读取状态寄存器来检索中断源。状态寄存器是“读取后清除”寄存器，因此中断源保留到调用 PWM_ReadStatusRegister() 函数为止。PWM_ReadStatusRegister() API 将记录哪些中断被使能，以提供有关实际中断源的准确报告。状态寄存器中的所有操作必须将以下定义用于位字段，因为这些位字段可能在放置和路由期间在状态寄存器内移动。

您可以通过在配置编辑器的 **Interrupts**（中断）部分设置 **None**（无）选项，选择从硬件完全删除状态寄存器。如果设置了此选项，则 API 将不支持访问状态寄存器。构建的设计中使用 API 访问状态寄存器将产生错误，指出 PWM_1_PWMUDB_sSTSReg_stsreg__STATUS_REG 是未定义的标识符。此错误可以通过删除 API 并在配置编辑器中为中断取消选择 **None**（无）选项，来进行更正。

状态数据在计数器的输入时钟沿处寄存，使得所有配置为 Mode = 1 的位具有了计数器的时序分辨率。这些位是粘滞的，将在读取状态寄存器后清除。所有其他配置为 Mode = 0 的位都是透明的，直接从输入读取到状态寄存器中，它们不是粘滞的，因此不在读取后清除。在下面列表的定义中，所有配置为 Mode = 1 的位都使用星号 (*) 进行指示。

状态寄存器中定义了一些位域掩码。这些位字段中的任何一个可能作为中断源包括在内。#定义在如下已生成的头文件 (.h) 中可用：

PWM_STATUS_TC *

终端计数输出的状态。当终端计数输出为高电平时，此位变为高电平。

PWM_STATUS_CMP1 *

pwm1 比较值与周期计数器相关时的状态。对于固定功能实现，在比较输出为高电平时，此位变为高电平。对于 UDB 实现，使用比较输出的寄存版本激活此位；因此，在比较输出为高电平后的两个时钟激活此位。

PWM_STATUS_CMP2 *

pwm2 比较值与周期计数器相关时的状态。对于固定功能实现，在比较输出为高电平时，此位变为高电平。对于 UDB 实现，使用比较输出的寄存版本激活此位；因此，在比较输出为高电平后的两个时钟激活此位。

PWM_STATUS_ KILL

输出非同步停止输入的状态。如果它当前有效，则输出将为高电平。

PWM_STATUS_FIFOFULL

捕获 FIFO 级别的状态。此位是 FIFO 级别的实时状态，表示 FIFO 当前已满。状态寄存器的此位中的“0”表示 FIFO 未满，但是不表示 FIFO 中没有数据。

控制

控制寄存器允许您控制 PWM 的常规操作。使用 PWM_WriteControlRegister() 函数写入此寄存器，使用 PWM_ReadControlRegister() 读取此寄存器。当读取或写入控制寄存器时，必须使用头 (.h) 文件中定义的位字段定义。控制寄存器的 #defines 如下：

PWM_CTRL_ENABLE

使能位控制对 PWM 操作的软件使能。PWM 具有在构建时定义的可配置使能模式。如果 **Enable Mode**（启用模式）参数设置为 **Hardware Only**（仅硬件），则此位不起作用。不过在任一其他模式下，如果此位未设置为高电平，PWM 将不会递减。正常操作要求设置此位且此位必须在 PWM 的所有操作中保持高电平。

PWM_CTRL_CMPMODE1_MASK

比较模式控制是一个三位字段，用于定义 pwm1 输出的预计比较输出操作。此位字段是控制寄存器中的三个连续位。此位字段中的所有操作必须使用与可用比较模式关联的 #defines。它们是：

- PWM_1_B_PWM_CM_LESSTHAN
- PWM_1_B_PWM_CM_LESSTHANOREQUAL
- PWM_1_B_PWM_CM_EQUAL
- PWM_1_B_PWM_CM_GREATERTHAN
- PWM_1_B_PWM_CM_GREATERTHANOREQUAL

此位字段是在初始化时使用 CompareMode1 参数中定义的比较模式配置的，可以通过 PWM_SetCompareMode() 或 PWM_SetCompareMode1() API 来修改。

PWM_CTRL_CMPMODE2_MASK

比较模式控制是一个三位字段，用于定义 pwm2 输出的预计比较输出操作。此位字段是控制寄存器中的三个连续位。此位字段中的所有操作必须使用与可用比较模式关联的 #defines。它们是：

- PWM_1_B_PWM_CM_LESSTHAN

- PWM_1_B_PWM_CM_LESSTHANOREQUAL
- PWM_1_B_PWM_CM_EQUAL
- PWM_1_B_PWM_CM_GREATERTHAN
- PWM_1_B_PWM_CM_GREATERTHANOREQUAL

此位域是在初始化时使用 CompareMode2 参数中定义的比较模式配置的，可以通过 PWM_SetCompareMode2() API 来修改。

周期（8 或 16 位，根据分辨率）

周期寄存器包含用户通过 PWM_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。PWM_ReadPeriod() 函数可用于查找此寄存器的当前值。周期寄存器不对 PWM 产生任何影响，直到达到终端计数为止，此时将使用此值重新加载周期计数器。

Compare1/Compare2（8 或 16 位，根据分辨率）

比较寄存器包含用于确定 pwm 或 pwm1 和 pwm2 输出状态的比较值（具体取决于 **PWM Mode**（PWM 模式）参数的设置）。pwm/pwm1 和 pwm2 输出由这些寄存器与周期计数器的比较值以及设置的比较模式决定。

周期计数器（8 或 16 位，根据分辨率）

周期计数器寄存器包含整个 PWM 操作中的计数器值。在基本操作中，当启用 PWM 时，此寄存器在时钟输入的每个上升沿按 1 递减。可以使用 PWM_ReadCounter() 函数调用随时读取此寄存器的内容。当达到终端计数时，计数值将重新加载初始化值或 PWM_WritePeriod() 函数写入的计数值。

pwm、pwm1 和 pwm2 输出基于此寄存器中保存的值之间的关系，以及通过 PWM_WriteCompare() 函数调用或使用 CompareValue 参数初始化期间在比较寄存器中定义的值。

有条件编译信息

PWM API 需要一些有条件编译定义来处理它必须支持的多个配置。API 必须在选择的分辨率上、固定功能模块或 UDB 模块之间选择的实现上、在死区模式、非同步停止输入模式和 PWM 模式上有条件地进行编译。定义的条件基于 FixedFunction、Resolution、DeadBand、KillMode 和 PWMMode 参数。API 永远不应当直接使用这些参数，但是应当使用下列定义。

PWM_Resolution

在构建时为分辨率值分配了分辨率定义。它在整个 API 中使用，用于根据此信息的 API 函数的正确数据宽度类型中进行编译。

PWM_UsingFixedFunction

“使用固定功能”定义通常用于在头文件中进行正确的寄存器分配。这是必需的，因为固定功能模块中提供的寄存器不同于当在 UDB 中实现 PWM 时使用的寄存器。

PWM_DeadBandMode

死区模式定义用于在 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() API 中进行有条件地编译。

PWM_KillModeMinTime

非同步停止输入模式最小时间定义用于在 PWM_WriteKillTime() 和 PWM_ReadKillTime() API 中进行有条件地编译。

PWM_KillMode

非同步停止输入模式定义用于定义“非同步停止输入模式最小时间”寄存器的寄存器访问点（如果 **Kill Mode**（非同步停止输入模式）为 **Minimum Time**（最小时间））。

PWM_PWMMode

PWM 模式定义用于根据需要为选择的模式调用相关的 PWM_WriteCompare() 和 PWM_ReadCompare() API 函数。

PWM_PWMModeIsCenterAligned

PWM 模式是中心对齐的定义，用于重新定义周期寄存器地址。中心对齐与实现中的其他模式之间不同，它需要将不同的寄存器用于必须在头文件中处理的操作。

PWM_DeadBandUsed

“使用的死区”定义用于控制 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() API 的有条件编译。

PWM_DeadBand2_4

死区 2-4 定义用于控制在 PWM_WriteDeadTime() 和 PWM_ReadDeadTime() API 中有条件地编译实现。

PWM_UseStatus

“使用状态”定义用于在仿真模型代码中删除状态寄存器（如果设计需要）以及在头文件和 C 文件中有条件地编译出状态寄存器定义和 API。

PWM_UseControl

“使用控制”定义用于在仿真模型代码中删除控制寄存器（如果设计需要）以及在头文件和 C 文件中有条件地编译出控制寄存器定义和 API。

PWM_UseOneCompareMode

“使用单比较模式”用于有条件地编译进和编译出单或双比较模式 PWM 模式函数的预计所需 API 调用。

PWM_MinimumKillTime

当 **Kill Mode**（非同步停止输入模式）设置为 **Minimum Kill Time**（最小非同步停止输入时间）时，提供编程到最小时间数据路径的初始最小非同步停止输入时间。

PWM_EnableMode

允许进行条件编译以删除为特定启用模式提供的 API。

常量

有一些为状态寄存器和控制寄存器以及某些枚举类型定义的常量。在本数据手册的前面已介绍了控制和状态寄存器的大多数常量。不过，头文件中需要更多常量来进行所有操作。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。由于编译器具有多个字节顺序，因此必须将 CY_GET_REGX 和 CY_SET_REGX 宏用于大于八位的寄存器访问。这些宏需要对每个寄存器使用_PTR 定义。

还需要允许适当的引擎放置和路由控制寄存器和状态寄存器位，因为组件必须具有用于定义位放置的常量。对于每个状态寄存器和控制寄存器位，有一个相关的_SHIFT 值，该值用于定义该位在寄存器中的偏移。它们在头文件中用于将最终位掩码定义为_MASK 定义。（_MASK 扩展仅添加到大于一位的位字段中，所有一位值不需要 _MASK 扩展。）

与 UDB 实现相比，固定功能模块具有一些限制，因为它是使用有限可配置性设计的。

PSoC 3 的直流电和交流电电气特性（FF 实现）

下面的值表示了预计性能，它们基于初始特性数据。

PWM 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 PWM，在所列的输入时钟频率下	-	-	-	µA
	3 MHz		-	15	-	µA
	12 MHz		-	30	-	µA
	48 MHz		-	260	-	µA
	67 MHz		-	350	-	µA

PWM 交流电规范

参数	说明	条件	最小值	典型值	最大值	单位
	工作频率		DC	-	67	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

PSoC 5 的直流电和交流电电气特性（FF 实现）

下面的值表示了预计性能，它们基于初始特性数据。

PWM 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 PWM，在所列的输入时钟频率下	-	-	-	µA
	3 MHz		-	65	-	µA
	12 MHz		-	170	-	µA
	48 MHz		-	650	-	µA
	67 MHz		-	900	-	µA

PWM 交流电规范

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
	脉冲宽度		13	-	-	ns
	脉冲宽度（外部）		30	-	-	ns
	非同步停止输入脉冲宽度		13	-	-	ns
	非同步停止输入脉冲宽度（外部）		30	-	-	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）

下面的值表示了预计性能，它们基于初始特性数据。

时序特性“额定路由的最大值”

参数	说明	配置 [4]	最小值	典型值	最大值	单位
f_{CLOCK}	组件时钟频率	配置 1	–	–	50	MHz
		配置 2	–	–	45	MHz
		配置 3	–	–	40	MHz
		配置 4	–	–	50	MHz
		配置 5	–	–	50	MHz
		配置 6	–	–	40	MHz
		配置 7	–	–	35	MHz
		配置 8	–	–	30	MHz
		配置 9	–	–	35	MHz
		配置 10	–	–	35	MHz
t_{CLOCKH}	输入时钟高电平时间 [5]	不可用	–	0.5	–	$1/f_{CLOCK}$
t_{CLOCKL}	输入时钟低电平时间 5	不可用	–	0.5	–	$1/f_{CLOCK}$

[4] PWM 组件的配置
- 配置 1:
  - 分辨率 : 8 位
  - PWM 类型 : 连续运行模式下的单输出
- 配置 2:
  - 分辨率 : 8 位
  - PWM 类型: 带有颤振的连续运行模式下的单输出
- 配置 3:
  - 分辨率 : 8 位
  - PWM 类型 : 中心对齐输出
- 配置 4:
  - 分辨率 : 8 位
  - PWM 类型 : 带有死区的单输出
- 配置 5:
  - 分辨率 : 8 位
  - PWM 类型 : 带有非同步停止输入模式的连续运行模式。
- 配置 6:
  - 分辨率 : 16 位
  - PWM 类型 : 连续运行模式下的单输出
- 配置 7:
  - 分辨率 : 16 位
  - PWM 类型: 带有颤振的连续运行模式下的单输出
- 配置 8:
  - 分辨率 : 16 位
  - PWM 类型 : 中心对齐输出
- 配置 9:
  - 分辨率: 16 位
  - PWM 类型: 带有死区的单输出
- 配置 10:
  - 分辨率 : 16 位
  - PWM 类型 : 带有非同步停止输入模式的连续运行模式。

[5] $t_{CY_clock} = 1/f_{CLOCK}$。这是时间是一个时钟周期。

参数	说明	配置 [4]	最小值	典型值	最大值	单位
输入						
t_{PD_ps}	输入路径延迟，引脚到同步模块 [6]	1	–	–	STA [7]	ns
t_{PD_ps}	输入路径延迟，引脚到同步模块 [8]	2	–	–	8.5	ns
t_{PD_si}	同步输出到输入路径的延时（路由延时）	1,2,3,4	–	–	STA 7	ns
t_{l_clk}	clockX 与时钟的对齐	1,2,3,4	0	–	1	t_{CY_clock}
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	1,2	t_{PD_ps} + t_{SYNC} + t_{PD_si}	–	t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{l_clk}	ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	3,4	t_{SYNC} + t_{PD_si}	–	t_{SYNC} + t_{PD_si} + t_{l_clk}	ns
t_{IH}	输入高电平时间	1,2,3,4	t_{CY_clock}	–	–	ns
t_{IL}	输入低电平时间	1,2,3,4	t_{CY_clock}	–	–	Ns

[6] 可以在后面所述的“静态时序分析结果”中找到 t_{PD_ps}。此处列出的数字是基于许多输入的 STA 分析的额定值。

[7] t_{PD_ps} 和 t_{PD_si} 是路由路径延迟。由于路由是动态的，这些值可以更改，且将直接影响组件的时钟组件和同步时钟的最高频率。可在静态时序分析结果中找到这些值。

[8] 配置 2 中的 t_{PD_ps} 是为器件的每个引脚定义的固定值。此处列出的数字是器件上可用的所有引脚的额定值。

时序特性“所有路由的最大值”

参数	说明	配置 [9]	最小值	典型值	最大值 [10]	单位
f_{CLOCK}	组件时钟频率	配置 1			25	MHz
		配置 2			23	MHz
		配置 3			20	MHz
		配置 4			25	MHz
		配置 5			25	MHz
		配置 6			20	MHz
		配置 7			18	MHz
		配置 8			15	MHz
		配置 9			18	MHz
		配置 10			18	MHz
t_{CLOCKH}	输入时钟高电平时间 [11]	不可用		0.5		$1/f_{clock}$
t_{CLOCKL}	输入时钟低电平时间 [11]	不可用		0.5		$1/f_{clock}$

[9] PWM 组件的配置

配置 1:
- 分辨率 : 8 位
- PWM 类型 : 连续运行模式下的单输出

配置 2:
- 分辨率 : 8 位
- PWM 类型 : 带有颤振的连续运行模式下的单输出

配置 3:
- 分辨率 : 8 位
- PWM 类型 : 中心对齐输出

配置 4:
- 分辨率 : 8 位
- PWM 类型 : 带有死区的单输出

配置 5:
- 分辨率 : 8 位
- PWM 类型 : 带有非同步停止输入模式的连续运行模式。

配置 6:
- 分辨率 : 16 位
- PWM 类型 : 连续运行模式下的单输出

配置 7:
- 分辨率 : 16 位
- PWM 类型 : 带有颤振的连续运行模式下的单输出

配置 8:
- 分辨率 : 16 位
- PWM 类型 : 中心对齐输出

配置 9:
- 分辨率: 16 位
- PWM 类型: 带有死区的单输出

配置 10:
- 分辨率 : 16 位
- PWM 类型 : 带有非同步停止输入模式的连续运行模式。

[10] “所有路由的最大值”时序数字通过将“额定路由”时序数字按系数 2 降额来进行计算。如果组件实例以这些速度或更低速度运行，则对于此组件不应遇到时序问题。

[11] $t_{CY_clock} = 1/f_{CLOCK}$。这是一个时钟时钟周期。

参数	说明	配置 [9]	最小值	典型值	最大值 [10]	单位
输入						
t_{PD_ps}	输入路径延迟，引脚到同步模块 [12]	1			STA [13]	ns
t_{PD_ps}	输入路径延迟，引脚到同步模块 [14]	2			8.5	ns
t_{PD_si}	同步输出到输入路径的延时（路由延时）	1,2,3,4			STA [13]	ns
t_{I_clk}	clockX 与时钟的对齐	1,2,3,4	0		1	t_{CY_clock}
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	1,2	t_{PD_ps} + t_{SYNC} + t_{PD_si}		t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{PD_IE}	组件时钟的输入路径延迟（边沿敏感输入）	3,4	t_{SYNC} + t_{PD_si}		t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	输入高电平时间	1,2,3,4	t_{CY_clock}			ns
t_{IL}	输入低电平时间	1,2,3,4	t_{CY_clock}			ns

如何将 STA 结果用于特性数据

额定路由最大值是通过使用静态时序分析 (STA) 进行多次测试而收集的。您可以用下列方法，使用 STA 结果计算设计的最大值：

f_{CLOCK} 最大组件时钟频率显示在命名外部时钟的时钟汇总中的时序结果中。下图演示了 *_timing.html* 中的时钟限制示例：

-Clock Summary

Clock	Actual Freq	Max Freq	Violation
BUS_CLK	24.000 MHz	118.683 MHz	
clock	24.000 MHz	56.967 MHz	

[12] 可以在后面所述的“静态时序结果”中找到 t_{PD_ps}。此处列出的数字是基于许多输入的 STA 分析的额定值。

[13] t_{PD_ps} 和 t_{PD_si} 是路由路径延迟。由于路由是动态的，这些值可以更改，且将直接影响时钟组件和同步时钟的最高频率。可在静态时序分析结果中找到这些值。

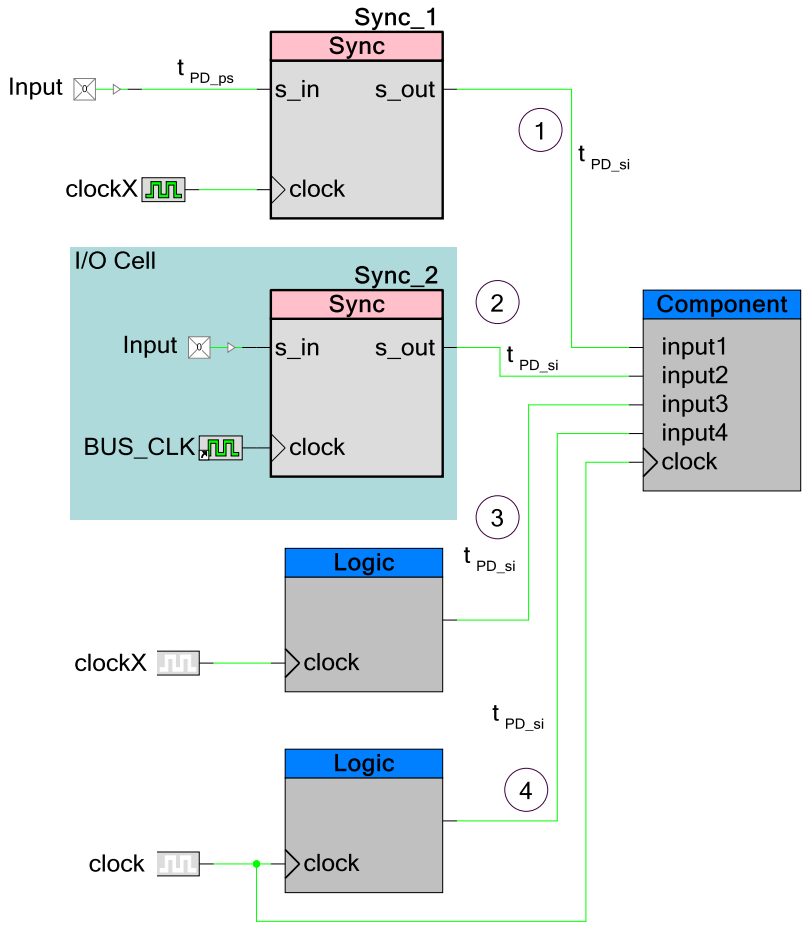
[14] 配置 2 中的 t_{PD_ps} 是为器件的每个引脚定义的固定值。此处列出的数字是器件上可用的所有引脚的额定值。

输入路径延迟和脉冲宽度

当表现输入功能的特征时，所有输入（无论您如何配置它们）看上去都类似于四种可能配置之一，如图 6 所示。

必须同步所有输入。同步机制取决于组件输入源。为了完全解析您的系统如何工作，您必须了解已为每个输入设置哪个输入配置以及系统的时钟配置。本节介绍如何使用静态时序分析 (STA) 结果确定系统的特性。

图 6. 组件时序规范的输入配置



配置	组件时钟	同步器时钟（频率）	图形
1	master_clock	master_clock	图11
1	计时器	master_clock	图9
1	计时器	clockX = 时钟 [15]	图7

[15] 时钟频率相等，但是不保证上升沿的对齐。

配置	组件时钟	同步器时钟（频率）	图形
1	计时器	clockX > 时钟	图8
1	计时器	clockX < 时钟	图10
2	master_clock	master_clock	图11
2	计时器	master_clock	图9
3	master_clock	master_clock	图16
3	计时器	master_clock	图14
3	计时器	clockX = 时钟 [15]	图12
3	计时器	clockX > 时钟	图13
3	计时器	clockX < 时钟	图15
4	master_clock	master_clock	图16
4	计时器	计时器	图12

1. 输入由器件引脚驱动，并在内部与“同步”组件同步。此组件的时钟采用与组件所使用时钟不同的内部时钟（所有内部时钟派生自 master_clock）。

   当表现按此方法配置的输入的特性时，clockX 可以快于、等于或慢于组件时钟。它还可以等于 master_clock，该时钟生成如图 7、图 8、图 10 和图 11 所示的特性参数。

2. 输入由器件引脚驱动，并使用 master_clock 在引脚同步。

   当按此方法配置输入时，master_clock 快于或等于组件时钟（不会慢于组件时钟）。这会生成如图 8 和图 11 所示的特性参数。

图 7. 输入配置 1 和 2：同步时钟频率 = 同步时钟频率（不保证时钟和 clockX 的边沿对齐）

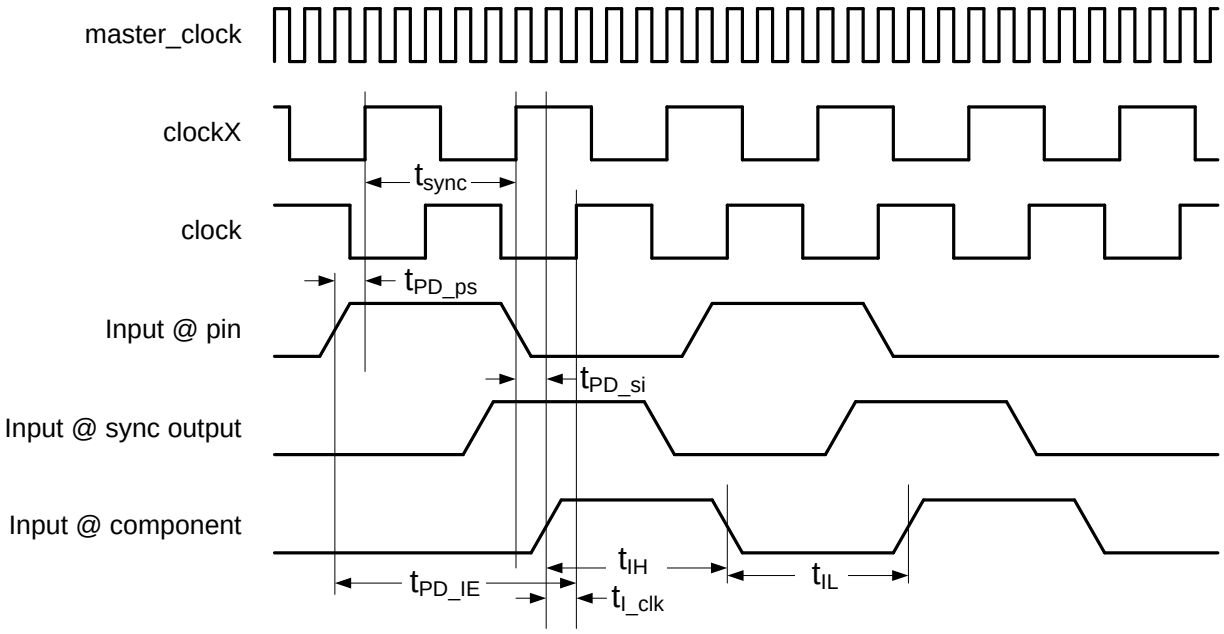


图 8. 输入配置 1 和 2：同步器时钟频率 > 组件时钟频率

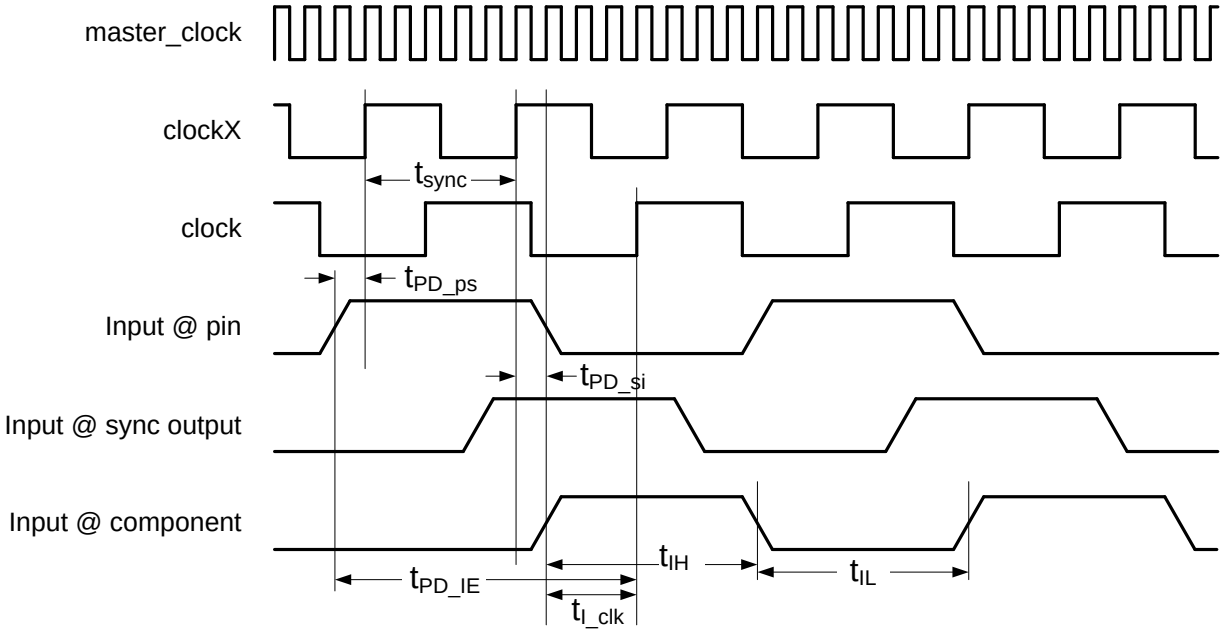


图 9. 输入配置 1 和 2：[同步器时钟频率 = master_clock] > 组件时钟频率

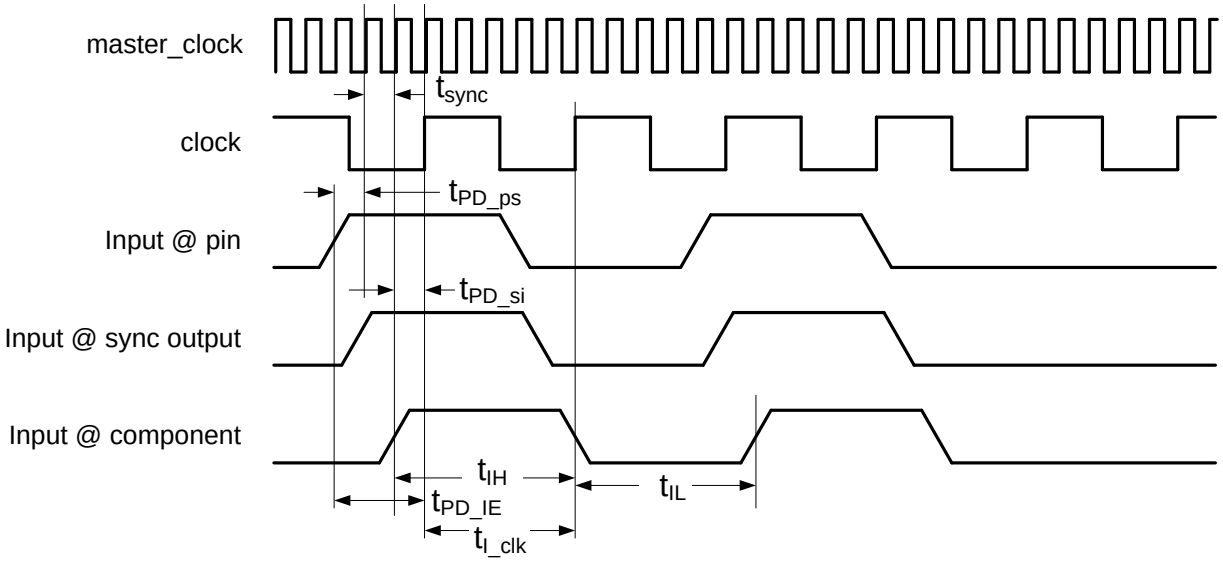


图 10. 输入配置 1：同步器时钟频率 < 组件时钟频率

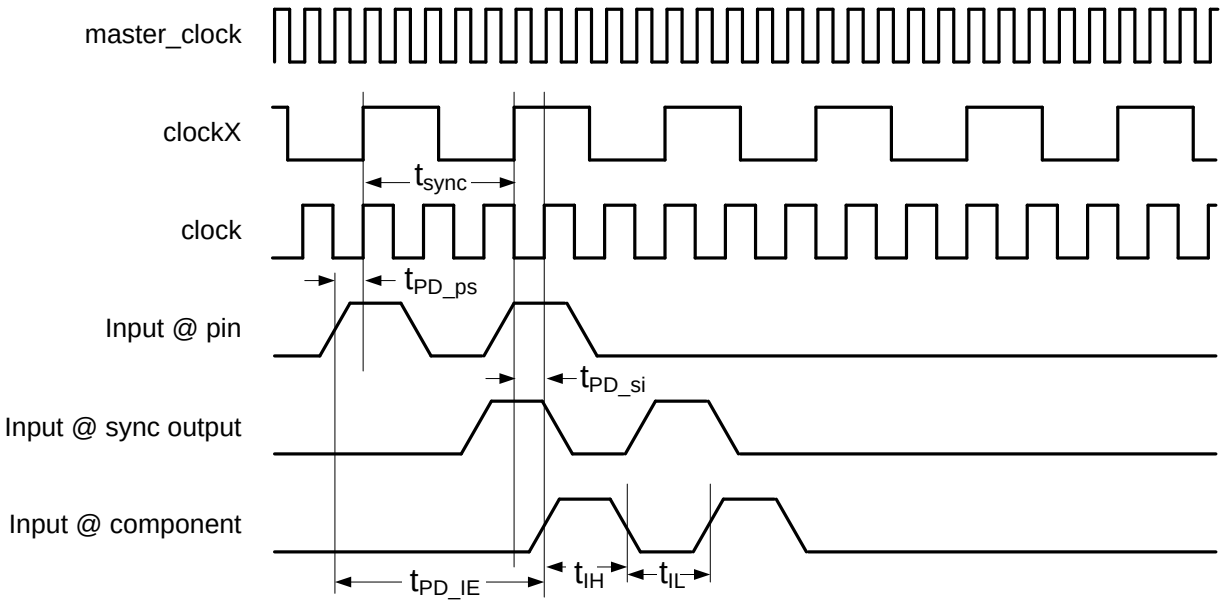
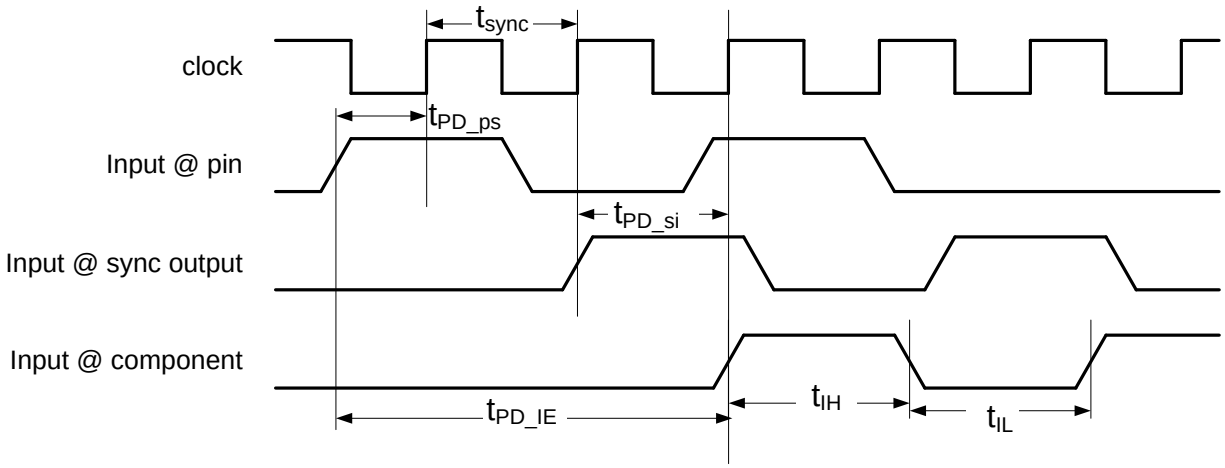


图 11. 输入配置 1 和 2：同步器时钟 = 组件时钟 = master_clock



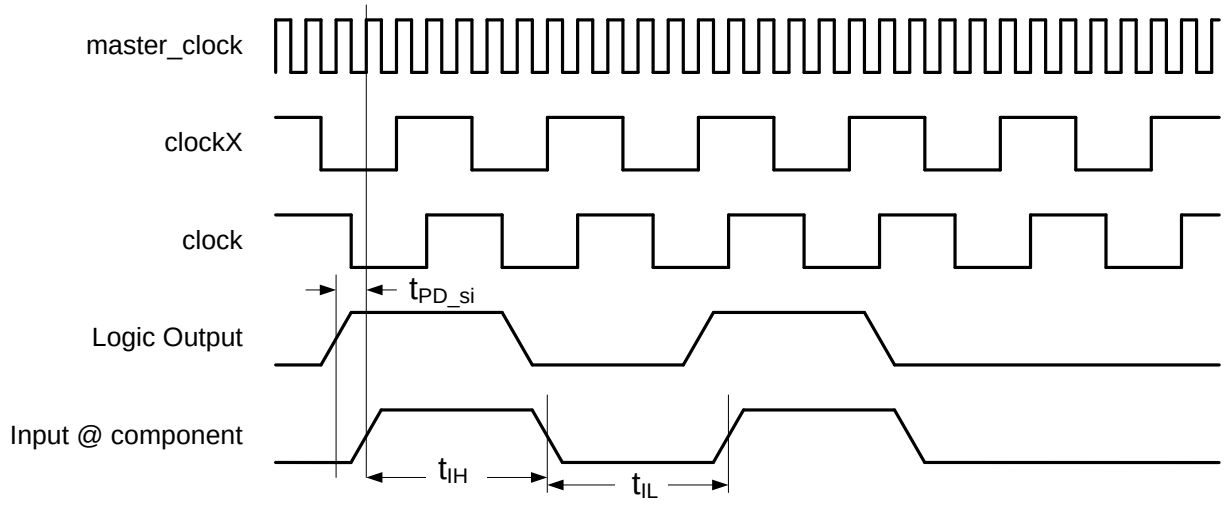
3. 输入由 PSoC 内部逻辑驱动，它基于与组件所使用的时钟不同的时钟同步（所有内部时钟都派生自 master_clock）。

   当按此方法配置输入时，同步器时钟快于、慢于或等于组件时钟。这会生成如图 12、图 13 和图 15 所示的特性参数。

4. 输入由 PSoC 内部逻辑驱动，它基于与组件所使用的时钟同步。

   当按此方法配置输入时，同步器时钟等于组件时钟。这会生成如图 16 所示的特性参数。

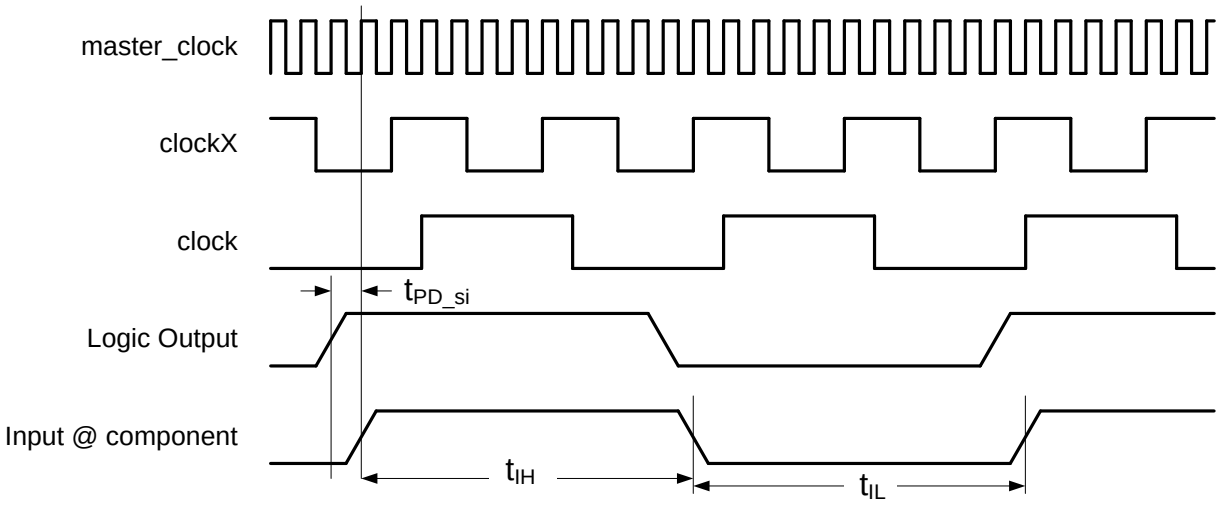
图 12. 仅输入配置 3：同步时钟频率 = 同步时钟频率（不保证时钟和 clockX 的边沿对齐）



此图表明静态时序分析保持时钟。数字时钟域中的所有时钟与 master_clock 同步。不过，具有相同频率的两个时钟的上升沿很可能不对齐。因此，静态时序分析工具不了解时钟同步到哪个边沿，

必须假设最小值为一个 master_clock 循环。这意味着 t_{PD_si} 现在对系统的 master_clock 的影响有限。如果此路径延迟太长，则 master_clock 设置时间会出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 13. 输入配置 3：同步器时钟频率 > 组件时钟频率



与图 12 中的方法几乎相同，所有时钟都派生自 master_clock。STA 在此配置中指明了对一个 master_clock 周期的 master_clock 的 t_{PD_si} 限制。如果此路径延迟太长，则 master_clock 设置时间出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 14. 输入配置 3：同步器时钟频率 = master_clock > 组件时钟频率

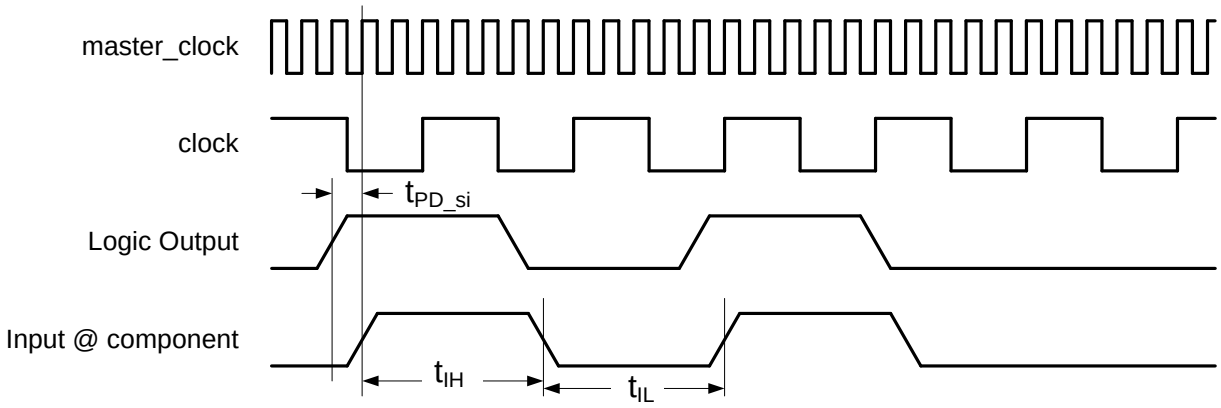
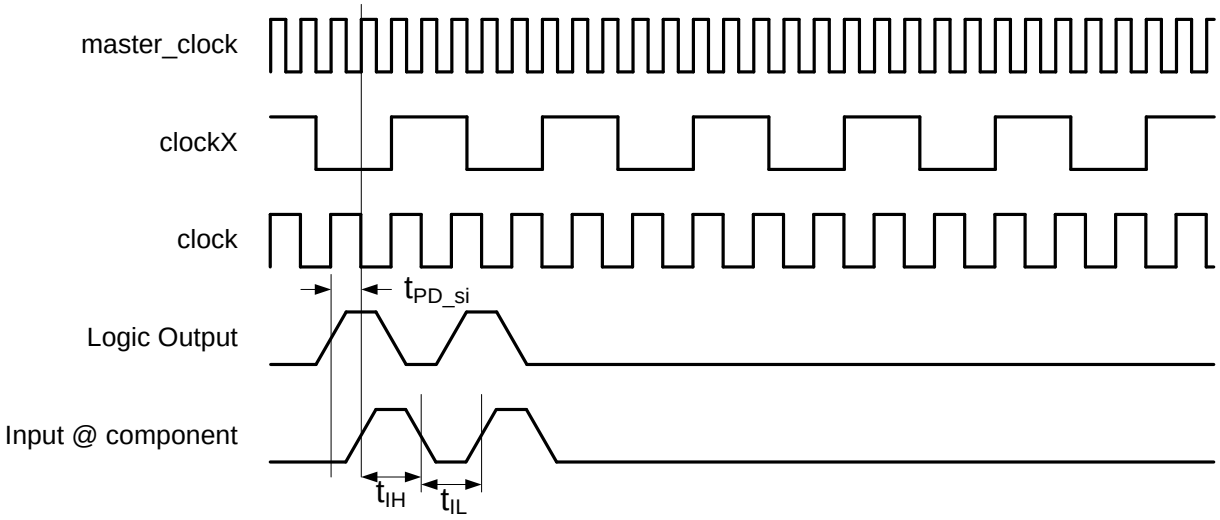
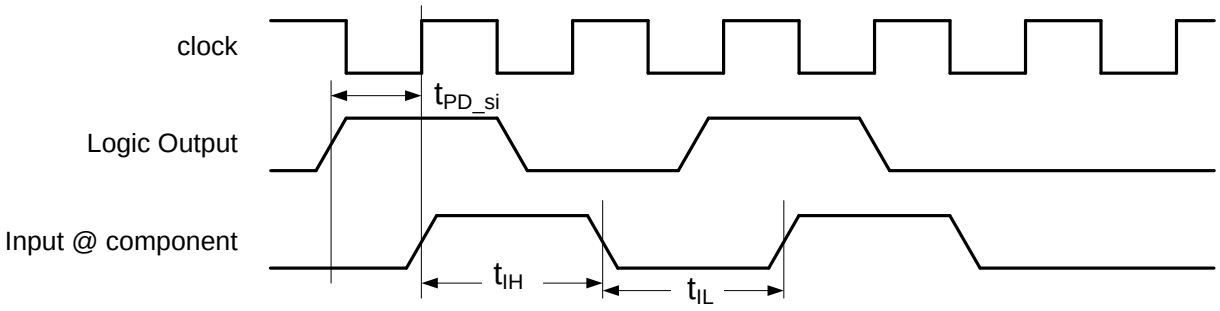


图 15. 输入配置 3：同步器时钟频率 < 组件时钟频率



与图 12 中的方法几乎相同，所有时钟都派生自 master_clock。STA 在此配置中指明了对一个 master_clock 周期的 master_clock 的 t_{PD_si} 限制。如果此路径延迟太长，则 master_clock 设置时间出现冲突。您必须更改系统的同步时钟，或者以较慢的频率运行 master_clock。

图 16. 仅输入配置 4：同步器时钟 = 组件时钟



在本节的所有上述图形中，在了解实现时使用的最关键参数是 f_{CLOCK} 和 t_{PD_IE}。t_{PD_IE} 由 t_{PD_ps} 和 t_{SYNC}（仅针对配置 1 和 2）、t_{PD_si} 和 t_{I_Clk} 定义。重要的是记住 t_{PD_si} 定义最大组件时钟频率。t_{I_Clk} 不源自 STA 结果，但是用于表示何时寄存 t_{PD_IE}。这是同步器与组件时钟之间的路由之后余留的余量。

t_{PD_ps} 和 t_{PD_si} 包括在 STA 结果中。

要查找 t_{PD_ps}，请查看 *_timing.html* 文件中定义的输入设置时间。此输入的输出端可以大于 1，因此您需要计算这些路径的最大值。

-Setup times

-Setup times to clock BUS_CLK

Start	Register	Clock	Delay (ns)
input1(0):iocell.pad_in	input1(0):iocell.ind	BUS_CLK	16.500

t_{PD_si} 是在“寄存器至寄存器”时间中定义的。您需要知道使用 *_timing.html* 文件的网络的名称。此路径的输出端可以大于 1，因此您需要计算这些路径的最大值。

-Register-to-register times

-Destination clock clock

Destination clock clock (Actual freq: 24.000 MHz)

+Source clock clock

-Source clock clock_1

Source clock clock_1 (Actual freq: 24.000 MHz)
Affected clock: BUS_CLK (Actual freq: 24.000 MHz)

Start	End	Period (ns)	Max Freq	Frequency	Violation
\Sync_1:genblk1[0]:INST\:synccell.syncq	\PWM_1:PWMUDB:runmode_enable\:macrocell.mc_d	7.843	127.508 MHz	24.000 MHz	

输出路径延迟

当表现输出路径延迟的特性时，必须考虑输出的去向，以了解在 STA 结果中何处可以找到数据。对于此组件，所有输出同步到组件时钟。输出可以是下列两类之一。输出到器件中的另一个组件，或输出到器件外的引脚。在第一种情况下，必须查看为上面“逻辑至输入”说明显示的“寄存器至寄存器”时间（源时钟是组件时钟）。对于第二种情况，可以在*_timing.html* STA 结果中查看“时钟至输出”时间。

组件更改记录

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
2.10	与自定义程序相关的更新	用于修复少量 GUI 相关问题。
	更新了 PWM_RestoreConfig() API	用于修复与在从低功耗模式唤醒后的中断触发器有关的问题。
	更新了 PWM_Stop() and PWM_Enable() API	FF PWM 可以在交替激活功耗模式使用
	更新了 PWM_SaveConfig() 和 PWM_RestoreConfig()updates	用于在唤醒之后恢复 PWM 周期寄存器。
	向数据手册添加了 PSoC 5 FF 直流和交流电气特性	
2.0.a	数据手册更新	
2.0	同步的输入	所有输入是在 “固定功能” 实现的 PWM模块都是同步的。
	PWM_GetInterruptSource() 函数转换为宏	PWM_GetInterruptSource() 函数与 PWM_ReadStatusRegister() 函数功能完全一致。为了节省代码空间，该函数转换为 PWM_ReadStatusRegister() 函数的宏替换。
	输出现在注册到组件时钟	为了避免组件输出中出现短时脉冲，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。
	实现了写入辅助控制寄存器时使用的关键区域。	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。
	向数据手册中添加了特性数据	
	对数据表进行了少量编辑和更新	

© 赛普拉斯半导体公司，2012。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任，也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC® 是赛普拉斯半导体公司的注册商标，PSoC Creator™ 和 Programmable System-on-Chip™ 是赛普拉斯半导体公司的商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途之外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用的赛普拉斯软件许可协议限制。