AN14196

灵活脉冲宽度调制器（FlexPWM）在MCX N系列上的使用

第1.0版—2024年5月2日 **应用笔记**

文档信息

信息	内容
关键词	AN14196、FlexPWM、MCX Nx4x、MCX N23x、电机控制、PWM
摘要	本文档描述了如何在MCX N系列芯片上使用FlexPWM模块，介绍了几种操作模式及其相应的实现方法，为不同的应用提供参考。

1 介绍

灵活脉冲宽度调制器（FlexPWM）模块包含多个PWM子模块，每个子模块都可用于控制单个半桥功率平台。该模块能够生成各种开关模式，包括高度复杂的波形，特别适合用于控制电机。

FlexPWM模块具有以下主要特性：

- 具有 16 位分辨率的中心对齐、边沿对齐和不非对称 PWM
- 通过抖动技术模拟增强分辨率，适用于无法进行精细边沿调整时
- PWM输出可作为一对互补通道或独立通道运行
- 能够接受有符号数进行PWM生成
- 可独立控制每个PWM输出的上升沿和下降沿
- 支持与外部硬件或其他PWM同步
- 双缓冲PWM寄存器
  – 整数倍重载率为1至16
  – 半周期重载能力
- 每个PWM周期内可通过硬件生成多个输出触发事件
- 支持双开关PWM输出
- 故障输入可被指定用于控制多个PWM输出
- 可编程的故障输入滤波器
- 可独立编程的PWM输出极性
- 可独立插入顶部死区时间和底部死区时间
- 每对互补输出可以使用其自己的PWM频率和死区时间值运行
- 可对每个PWM输出进行单独的软件控制
- 所有输出均可通过FORCE_OUT事件编程为同时改变
- PWM_X引脚可选择从每个子模块输出第三个PWM信号
- 未用于PWM生成的通道可用于缓冲输出比较功能和输入捕获功能
- 增强型双边沿捕获功能

本文档描述了如何在MCX N系列芯片上使用FlexPWM模块，介绍了几种操作模式及其相应的实现方法，为不同的应用提供参考。同时，PWM操作逻辑，如寄存器重载逻辑和分数延迟逻辑，也可以帮助用户进一步了解FlexPWM模块。

2 框图

图1展示了FlexPWM的框图：

- 该器件配备两个PWM模块实体：PWM0和PWM1。
- 每个模块包含四个子模块（每个子模块都有其自己的时基），并配备四个故障通道。每个通道可容纳四个不同的故障输入。
- 每个FAULTx引脚可任意映射，以控制任意组合的PWM输出。默认情况下，子模块0被视为内部同步控制的主模块。
- 只有子模块0输出控制信号，包括主重载、主强制、主同步（sync）和辅助时钟，而其他子模块（1/2/3）或外部组件则接收这些信号。
- 另外，外部信号PWM_EXT_SYNC、EXT_FORCE和EXT_CLK可以共同控制和同步这四个子模块。
- 此外，每个子模块可以生成独立的输出触发信号，以触发其他组件中的事件，并生成一个中断信号以供CPU进行中断响应。

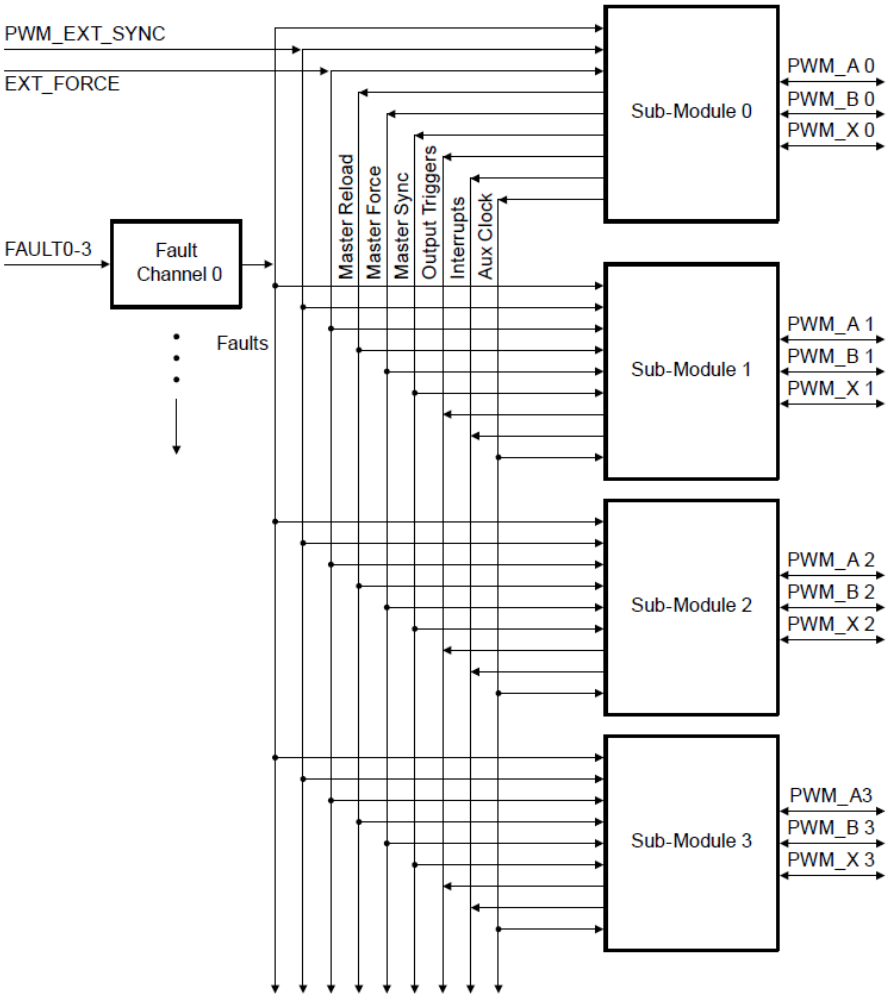

图1. FlexPWM模块框图

图2展示了PWM子模块的详细信息：

- 在每种情况下，每个PWM输出信号都使用两个比较器和相关的VALx寄存器。
- 一个比较器和VALx寄存器控制开启边沿，而另一个比较器和VALx寄存器控制关闭边沿。
- 本地同步信号的生成方式与子模块中的其他PWM信号相同。
- 比较器0导致本地同步信号生成上升沿，比较器1则生成下降沿。
- 比较器1还与重载逻辑硬连接，以生成全周期重载指示信号。

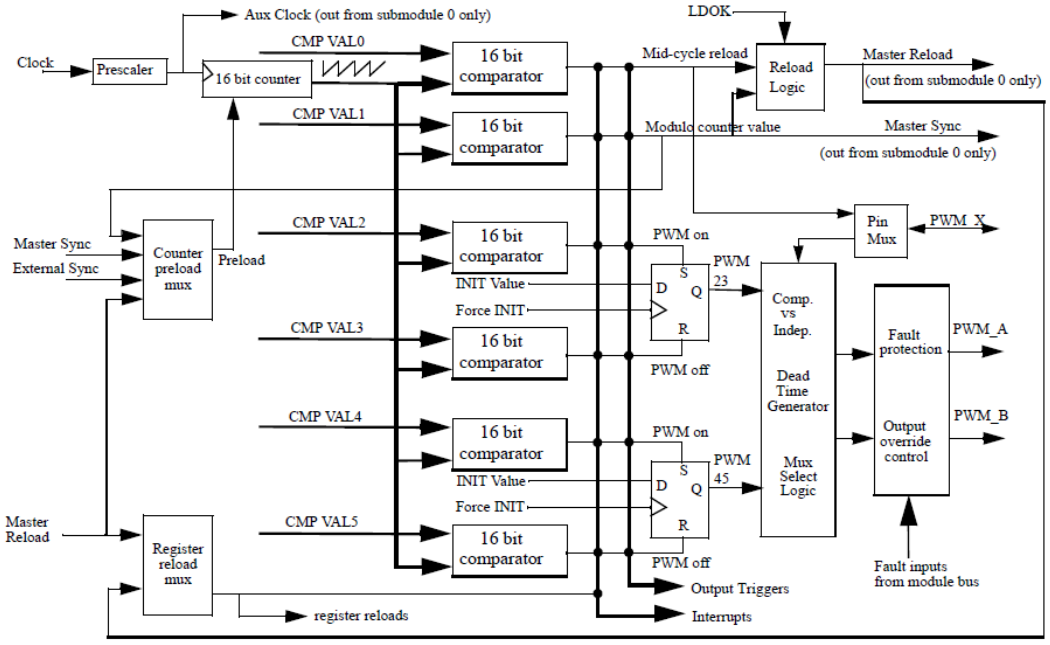

图2. PWM 子模块框图

- 如果 VAL1 控制计数器的模数，而 VAL0 为 VAL1 寄存器值减去 INIT 值的一半，那么半周期重载脉冲在定时器计数周期的中点发生。此时，本地同步信号的占空比为 50%。
- 另一方面，如果 VAL1 和 VAL0 寄存器不需要用于寄存器重载或计数器初始化，它们可用于调制本地同步信号的占空比，从而有效将其转换为辅助 PWM 信号（PWM_X）。这里假设 PWM_X 引脚未用于其他功能，如输入捕获或死区时间失真校正等。
- 包括本地同步信号在内，每个子模块生成三个 PWM 信号，软件可以完全控制每个信号的每个边沿。
- 如果比较器和边沿值寄存器不需要用于 PWM 生成，可将其用于其他功能。这些功能可以是输出比较、生成输出触发或按时间间隔生成中断。
- 图 2 中所示的 16 位比较器是“大于等于”比较器，而不仅仅是“等于”比较器。
- 此外，如果双稳态触发器的置位和复位都被置于有效，则其输出变为0。

3 功能描述

FlexPWM是一个功能强大的模块，主要用于生成PWM信号并实现多种不同的PWM功能，如本节所述。

3.1 PWM的能力

MCX N系列的FlexPWM模块包含四个子模块，每个子模块都有自己的时基和PWM生成能力。这些子模块可以独立工作，也可以同步运行。在本节中，仅以一个子模块为例进行说明。

3.1.1 中心对齐PWM

中心对齐PWM提供单一的PWM信号。在这种模式下，PWM周期的一半出现在中心点之前，另一半出现在中心点之后，如图3所示。这种模式会减少1位占空比分辨率，因为PWM占空比在每个PWM时钟周期内要改变两次。

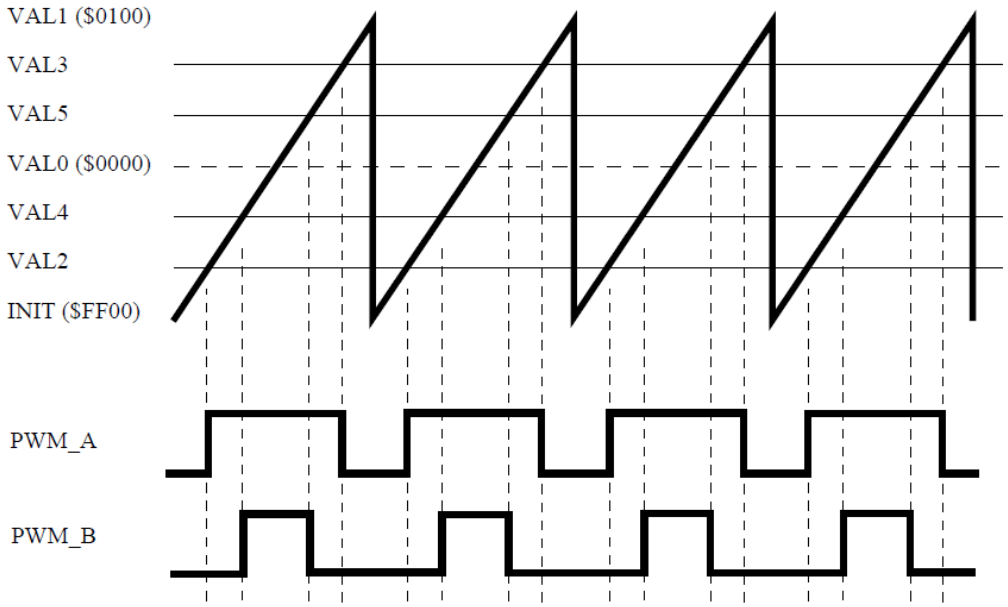

图3. 中心对齐PWM示例

对于MCX N系列的中心对齐PWM，中心点被定义为INIT值和VAL1值之和的一半。因此，每个脉冲的开启边沿值（VAL2、VAL4）必须更新，使中心点与开启边沿之间的差值等于占空比的一半。同样，应更新关闭边沿值（VAL3、VAL5），使关闭边沿与中心点之间的差值等于占空比的一半。

如果所有PWM信号边沿的计算都遵循这一约定，那么这些信号就会相对于彼此中心对齐。信号之间的中心对齐并不局限于围绕零计数值的对称性，因为任何其他数值也可以作为中心点。不过，以零为中心可以在有符号模式下提供最大的范围，并简化了计算。

中心对齐PWM用于多种功率开关或多相转换器，例如半桥和全桥逆变器。使用这种PWM可以显著改善系统的电磁干扰（EMI）和总谐波失真（THD）性能。

3.1.2 边沿对齐PWM

边沿对齐PWM提供单一的PWM信号。在这种模式下，PWM信号的两个边沿中的一个与周期边界对齐,而占空比则决定另一个边沿的位置，如图4所示。使用边沿对齐PWM能够实现最大的占空比分辨率，因为PWM占空比可以在整个PWM时钟周期内改变。

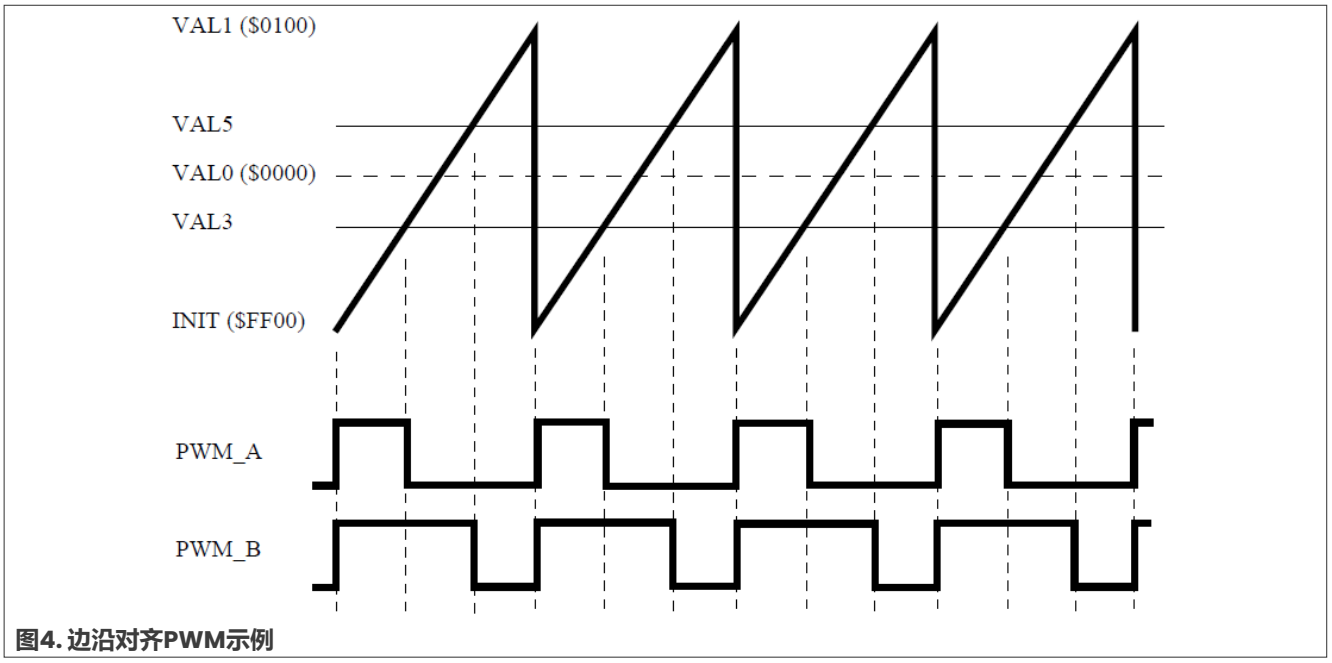

图4. 边沿对齐PWM示例

对于MCX N系列的边沿对齐PWM,每个脉冲的开启边沿值（VAL2、VAL4）被设定为INIT值。因此,只需更新关断边沿值（VAL3、VAL5）来改变占空比。

3.1.3 相移PWM

相移PWM提供的是相互同步但相位相对偏移的PWM信号,如图5所示。相移不影响占空比 。因此，平均负载电压不会受到影响。

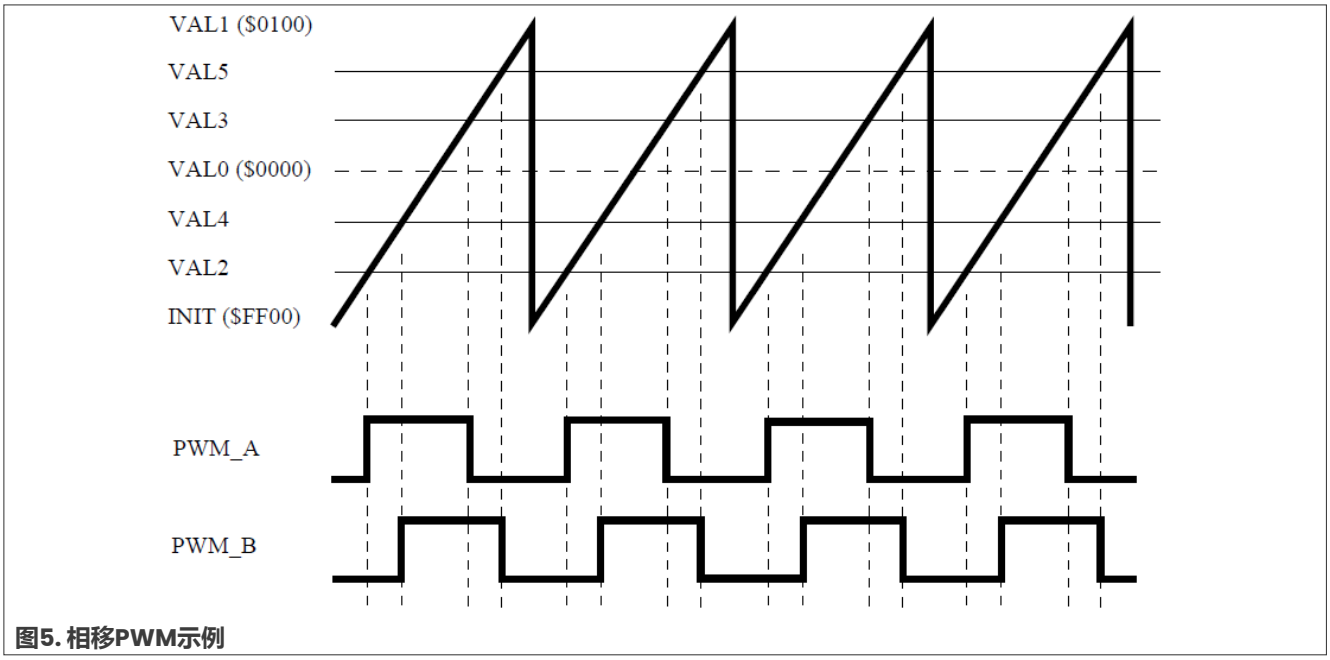

图5. 相移PWM示例

对于MCX N系列的相移PWM，PWM_A和PWM_B的占空比相同。但是，PWM_B相对于PWM_A存在相位延迟。

VAL2和VAL3定义PWM_A的输出。要得到VAL4和VAL5的值，需要将相移值分别加到VAL2和VAL3。

注：*当需要在子模块0和其他子模块之间实现相移时,应使用PHASEDLY寄存器,而不是直接在开启和关断边沿加偏移量。*

3.1.4 双开关PWM

支持双开关PWM（DBLPWM）输出功能以辅助进行单相电流测量和三相重构。这种方法支持在每个PWM周期内有两个独立的上升沿和两个独立的下降沿。VAL2和VAL3寄存器用于生成PWM_A信号，而VAL4和VAL5用于生成PWM_B信号。强制输出逻辑的两个通道PWM_A和PWM_B，使用XOR逻辑（强制输出逻辑）进行组合，如图6所示。DBLPWM信号可以加入死区时间插入逻辑运行。

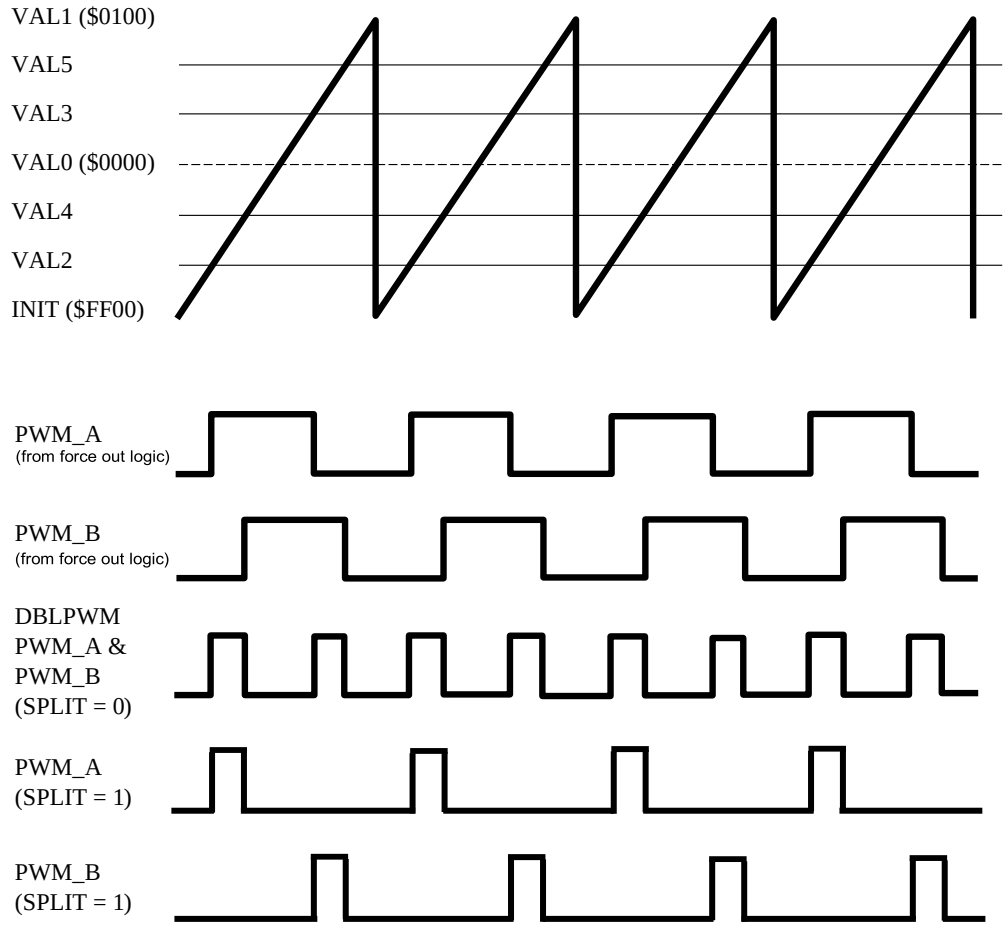

图6. 双开关PWM示例

3.1.5 ADC触发

在ADC触发时机至关重要的情况下，ADC触发必须由硬件事件调度，而不是由软件激活。使用该PWM模块，每个PWM周期内可以在硬件中生成多个ADC触发，而无需另一个定时器模块，如图7所示。当指定互补工作模式时，对于给定的子模块，只需要两个边沿比较器就能生成输出PWM信号。这意味着其他比较器可以执行其他功能。在此示例中，软件无需在第一次转换后快速响应来设置必须在同一PWM周期内发生的其他转换。

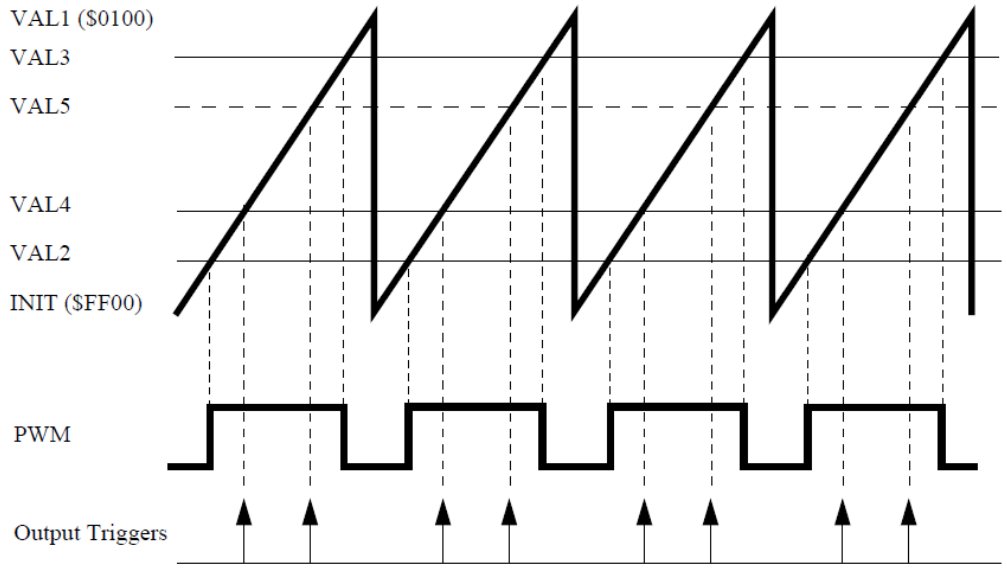

图7. 硬件中多输出触发生成

每个子模块都有自己的定时器，因此每个子模块都有可能以不同的频率运行。其中一个可能的选项是让一个或多个子模块以较低的频率运行，但仍与子模块0中的定时器保持同步。

图8展示了如何使用这一功能在多个PWM周期内调度ADC触发。使用较低频率的子模块来控制软件控制算法的采样频率，在整个采样周期内调度了多个ADC触发。图8显示了所有子模块比较器都用于ADC触发的生成。

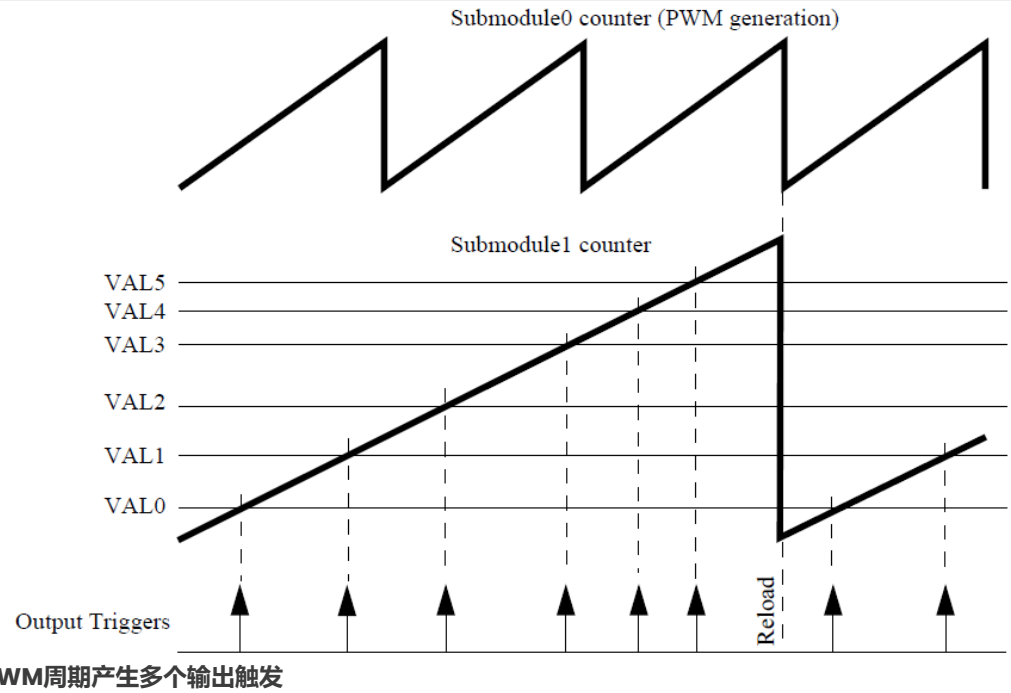

图8. 跨多个PWM周期产生多个输出触发

3.1.6 增强型捕获功能（E-capture）

当PWM引脚不用于PWM生成时，可以用来执行输入捕获。对于PWM生成，比较寄存器的值指定PWM信号的两个边沿。当PWM引脚被编程为输入捕获时，两个寄存器可在同一个引脚上工作以捕获多个边沿。捕获在CVAL0和CVAL1两个值之间交替进行，可以是自由运行的方式或单次触发的方式。通过编程每个捕获电路的期望边沿，可以轻松测量输入信号的周期和脉宽，无需重新配置电路。此外，输入信号的每个边沿都可以为一个8位计数器提供时钟，而此计数器的输出与用户指定的值（EDGCMP）进行比较。当计数器输出等于EDGCMP时，子模块定时器的值被捕获，计数器自动复位。此功能可以计数指定数量的边沿事件，然后执行捕获并触发中断。图9展示了E-capture电路的部分功能。

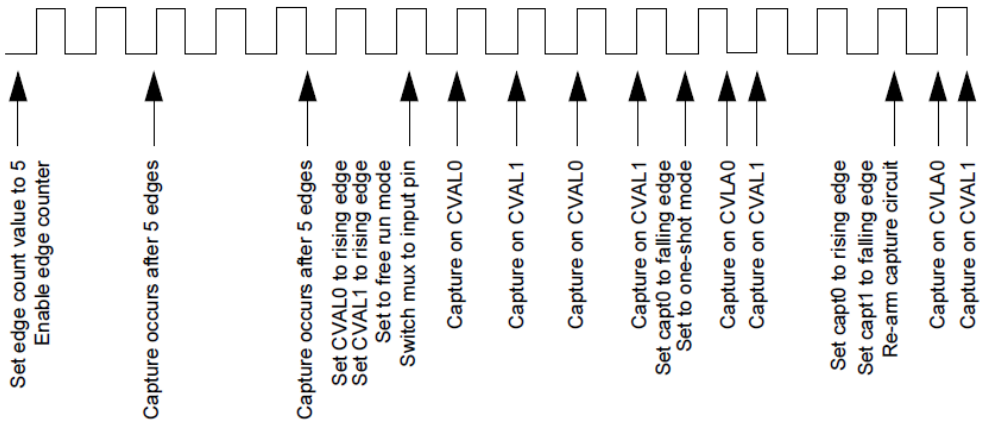

图9. E-capture电路的捕获能力

当一个子模块用于PWM生成时，其定时器计数到用于指定PWM频率的模数，并重新初始化。这个定时器不会计数所有数字，定时器的重置代表在16位数字范围存在不连续性。因此，将这个定时器用于其他引脚（如PWM_X）的输入捕获时，应用场景会受到限制。不过，当测量与PWM频率同步的信号时，定时器的模数范围就非常适合这种应用。

如图10所示，PWM功率平台的输出连接到配置为自由运行输入捕获模式的PWM_X引脚。CVAL0捕获电路被设置为捕获上升沿，CVAL1捕获电路被设置为捕获下降沿。因此，每个PWM周期都会获取新的负载脉宽数据。要计算脉宽，用CVAL1寄存器的值减去CVAL0寄存器的值。这种测量方法在对驱动感性负载的半桥电路进行死区时间失真校正时非常有用。此外，这些捕获值可以直接与负责生成PWM输出的VALx寄存器进行比较，以获得系统传播延迟的测量值。

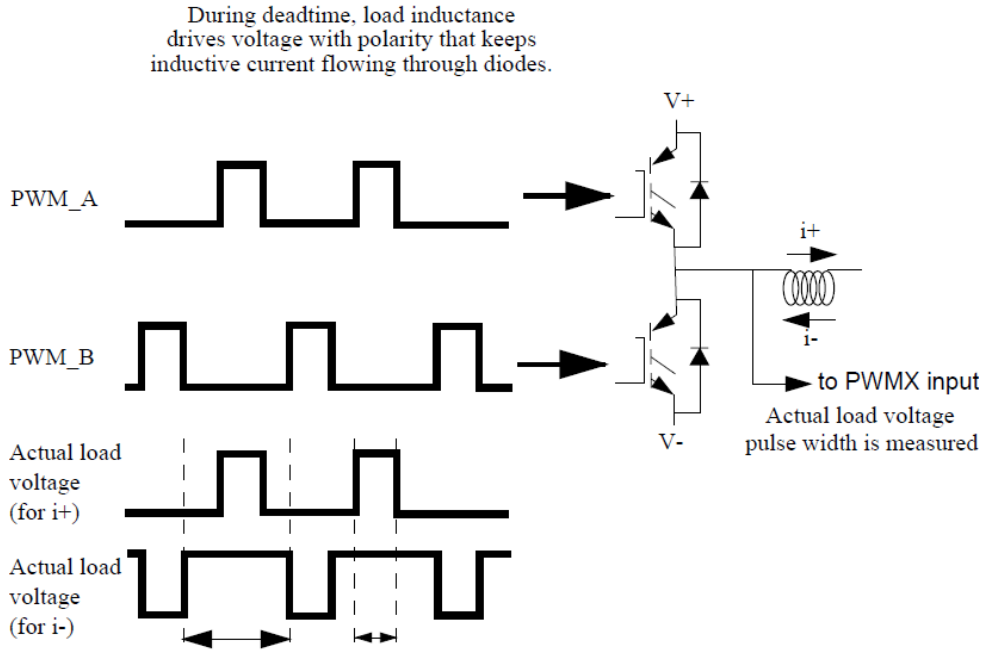

图10. 使用E-Capture电路测量输出脉冲宽度

3.2 操作

本节详细描述了PWM的操作逻辑。

图11是一个生成输出PWM的概览框图。

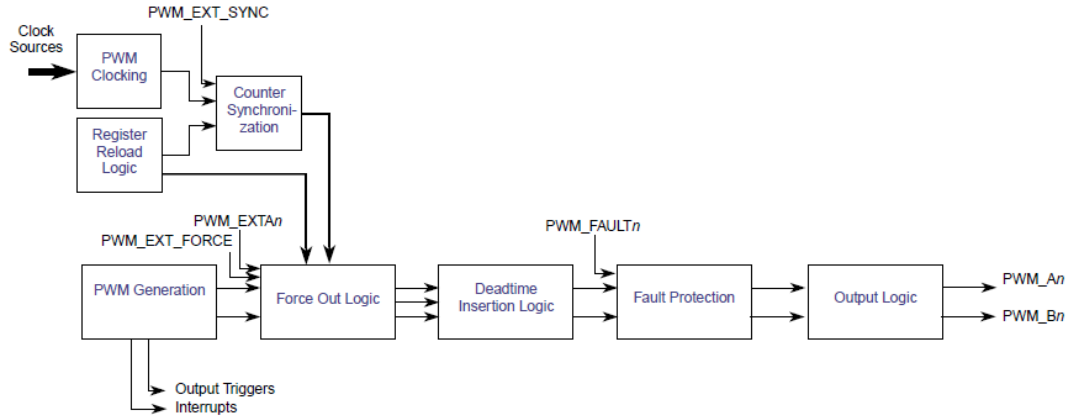

图11. 输出PWM生成的概览框图

3.2.1 寄存器重载逻辑

寄存器重载逻辑决定了所有双缓冲寄存器对的外部寄存器组何时必须转移到内部寄存器组。通过使用CTRL[LDFQ]和CTRL[FULL]（由VAL1寄存器定义），可以将寄存器重载事件调度为每“n”个PWM周期发生一次。系统还支持半周期重载选项（CTRL[HALF]），此时重载可以发生在一个PWM周期的中间。该半周期点由VAL0寄存器定义，并且不必恰好在PWM周期的中点。

如图12所示，子模块0的重载信号可以作为主重载信号进行广播。据此，允许子模块0的重载逻辑控制其他子模块中寄存器的重载。

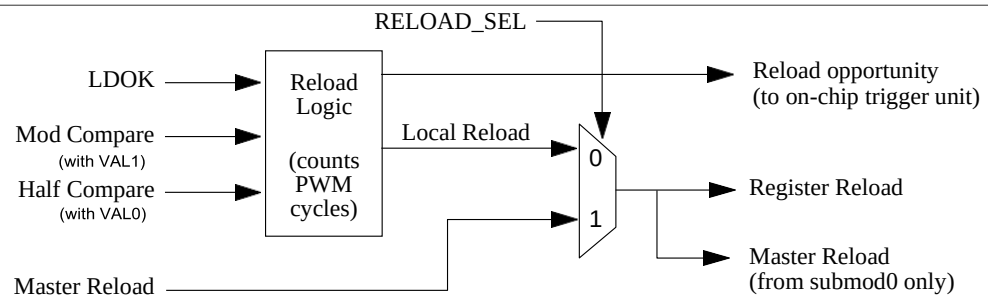

图12. 寄存器重载逻辑

3.2.2 计数器同步

图13中所示的16位计数器可以通过以下四个可能的源使用INIT的值进行初始化：

- 本地同步
- 主重载
- 主同步
- PWM_EXT_SYNC

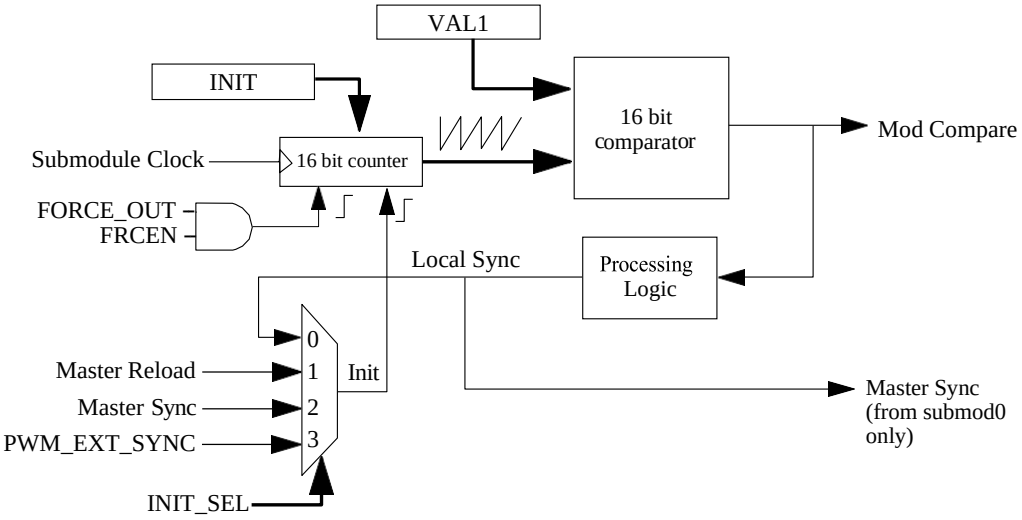

图13. 子模块计数器同步

如果选择本地同步作为计数器初始化信号，计数器会一直计数，直到其输出等于VAL1。VAL1用于指定计数器的模数。因此，子模块内的VAL1有效地控制定时器周期（从而控制了该子模块生成的PWM的频率），并且所有操作都在本地层面进行。要初始化计数器，可以配置主同步信号（源自子模块0的本地同步）。因此，任何子模块的计数器周期都可以与子模块0中的计数器周期锁定。其他子模块的VAL1寄存器及相关比较器可以自由地用于其他功能，如PWM生成、输入捕获、输出比较或输出触发等。

如果选择主重载信号作为计数器初始化的源，那么任何子模块的计数器周期都会与子模块0的重载频率锁定。

***注：**此主重载信号仅源自子模块0。*

由于重载频率是可选的，并且可以在1到16之间变化，因此支持生成同步的多频率PWM信号。

如果选择PWM_EXT_SYNC信号作为计数器初始化的源，外部源可以控制所有子模块中的计数器的周期。

***注：**此PWM_EXT_SYNC信号可能源自芯片内部或外部，具体取决于系统架构。*

因此，实现FlexPWM模块与外部信号之间的同步变得更加方便。

此外，如果设置了CTRL2[FRCEN]，计数器还可以在FORCE_OUT信号置位时选择性地初始化。FORCE_OUT信号主要用于多个PWM输出的同步切换。如图13所示，它构成了计数器的第二个初始化输入。

因此，无论选择哪个信号作为计数器初始化信号，计数器都会进行初始化。

总之，如果需要同步多个相同频率的PWM，当所有PWM信号都来自内部子模块时，推荐使用主同步信号（子模块0选择本地同步进行计数器初始化）来初始化从属子模块（子模块1/2/3）的计数器。否则，如果某些PWM信号来自另一个PWM模块或芯片外部，推荐使用PWM_EXT_SYNC信号来初始化所有子模块的计数器。

如果需要同步多个不同频率的PWM，推荐以下两种方法：

- 方法1：
  1. 配置子模块0的PWM信号为最高频率。该子模块使用本地同步初始化计数器来实现内部PWM同步，或使用PWM_EXT_SYNC来实现外部PWM同步。
  2. 选择主重载（重载率取决于不同PWM的频率比）信号来进行从属子模块（子模块1/2/3）中计数器的初始化。
- 方法2：
  1. 初始化生成较高频率PWM信号的子模块的计数器时，选择本地同步。
  2. 将它们的FORCE源配置为主同步信号或EXT_FORCE信号。
     ***注：**对于主同步信号，子模块0被配置为生成最低频率的PWM。对于EXT_FORCE信号，另一个从属子模块被配置为生成最低频率的PWM。这个子模块需要输出一个本地同步触发，作为其他子模块的EXT_FORCE信号输入。*
  3. 配置子模块（生成最低频率PWM的子模块），选择本地同步初始化计数器来实现内部PWM同步，或选择EXT_SYNC来实现外部PWM同步。

3.2.3 强制输出逻辑

对于每个子模块，软件可以根据芯片架构为FORCE_OUT信号选择以下八个信号源之一：

1. 本地CTRL2[FORCE]
2. 子模块0的主强制信号
3. 本地重载信号
4. 子模块0的主重载信号
5. 本地同步信号
6. 子模块0的主同步信号
7. 片上或片外的EXT_SYNC信号
8. 片上或片外的EXT_FORCE信号

本地信号会改变子模块输出引脚上的信号，而不考虑与其他子模块的同步。然而，如果所有子模块输出上的所有信号需要同时改变，则必须选择主信号、EXT_SYNC或EXT_FORCE信号。

图14展示了强制输出逻辑。SEL23和SEL45字段各自从以下四个信号中选择一个，这些信号可以提供给子模块的输出：

- PWM信号
- 反相PWM信号
- 由软件通过OUT23和OUT45位指定的二进制电平
- PWM_EXTA或PWM_EXTB备用外部控制信号

此选择可以提前确定。当发生FORCE_OUT事件时，这些值会被呈现给信号选择多路复用器。该多路复用器会立即将请求的信号切换到其输出端，以供下游进一步处理。

3.2.4 独立或互补通道运行

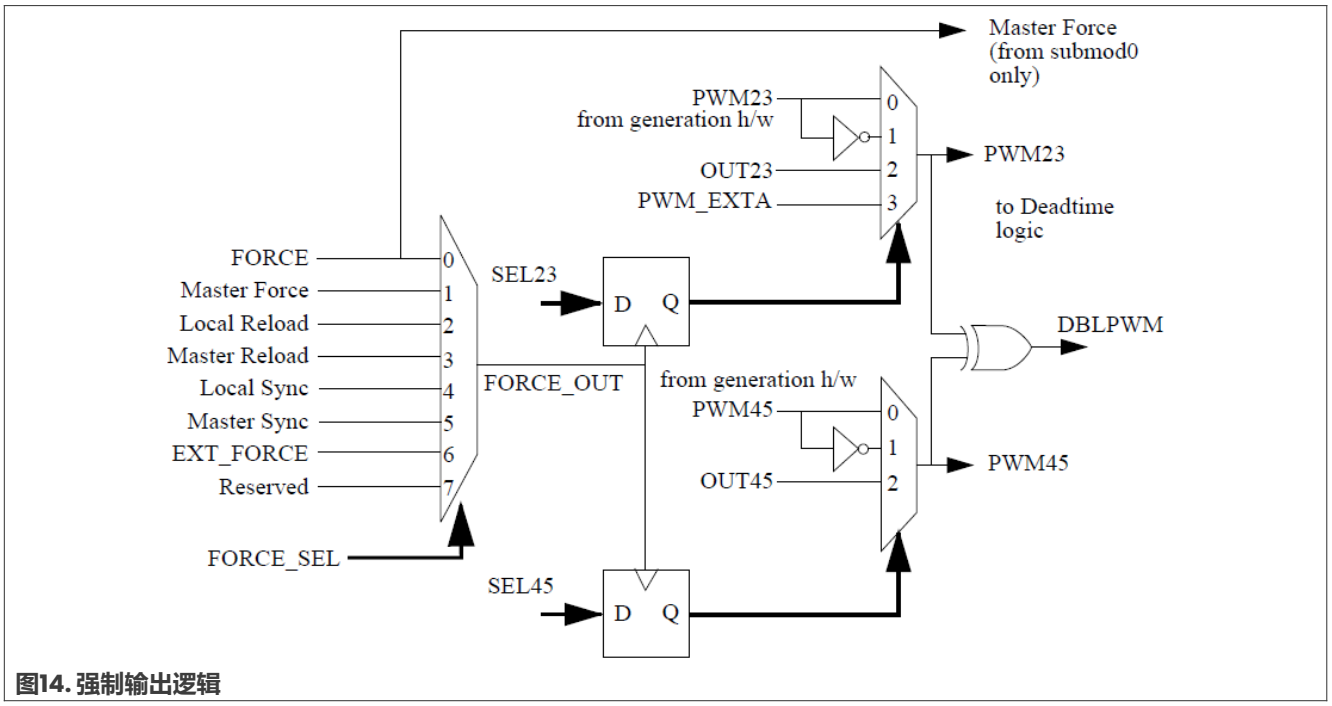

图14. 强制输出逻辑

向CTRL2[INDEP]写入逻辑1，可将一对PWM输出配置为两个独立的PWM通道。在这种模式下，每个PWM输出由各自的VALx对独立控制，互不影响。

向CTRL2[INDEP]写入逻辑0，可将PWM输出配置为一对互补通道。在互补通道操作模式下，PWM引脚是成对的，如图15所示。MCTRL[IPOL]决定将哪个信号连接到输出引脚（PWM23或PWM45）。

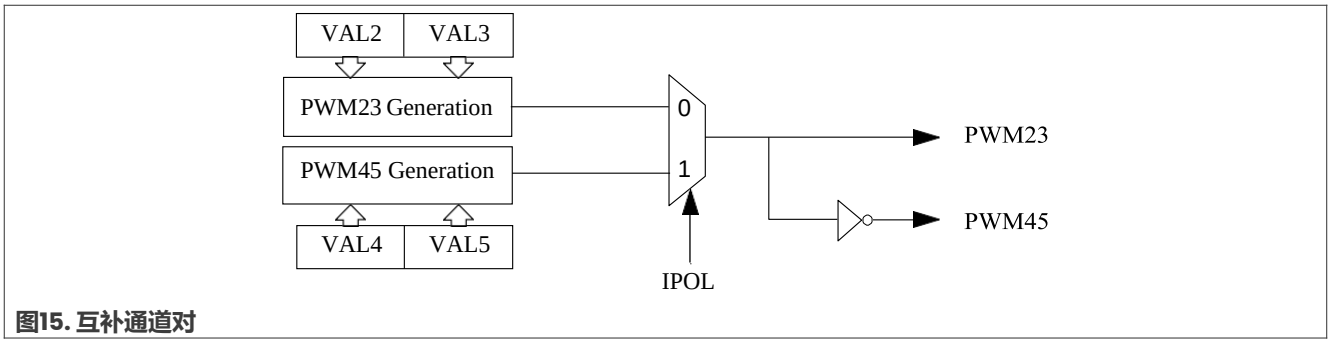

图15. 互补通道对

互补通道操作主要用于驱动电机驱动电路中的上下晶体管，如图16所示。

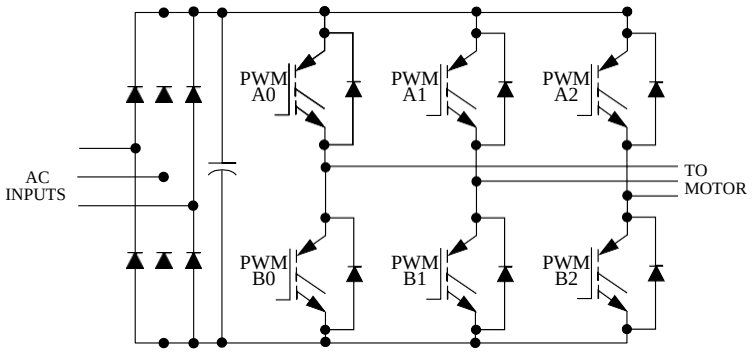

图16. 典型三相交流电机驱动

这种互补操作模式允许使用死区时间插入功能。

3.2.5 死区时间插入逻辑

图17展示了每个子模块的死区时间插入逻辑，该逻辑用于在非独立模式下创建互不重叠的互补信号。

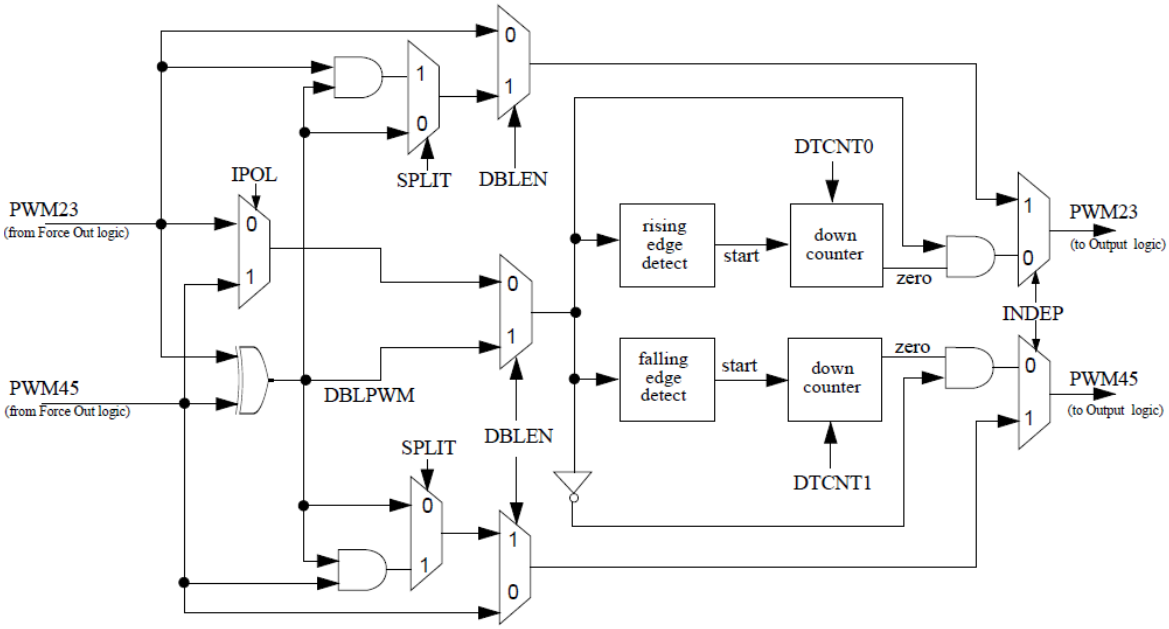

图17. 死区时间插入逻辑

在互补模式下，一对PWM信号可用于驱动上下晶体管，如图17所示。当顶部PWM通道处于激活状态时，底部PWM通道处于非激活状态，反之亦然。

***注：**为了避免直流母线短路和损坏晶体管，必须确保上下晶体管的导通区间不重叠。然而，晶体管的特性导致其关断时间比开通时间长。为了防止上下晶体管出现导通重叠，必须在开关周期内插入死区时间，如图18所示。*

死区时间生成器会自动在一对PWM输出中插入可通过软件选择的激活延迟。死区时间寄存器（DTCNT0和DTCNT1）指定用于死区时间延迟的IPBus时钟周期数。每当检测到上升沿和下降沿时，倒数计数器就会启动，并插入死区时间。死区时间会将PWM输出对中的信号强制置为非激活状态。

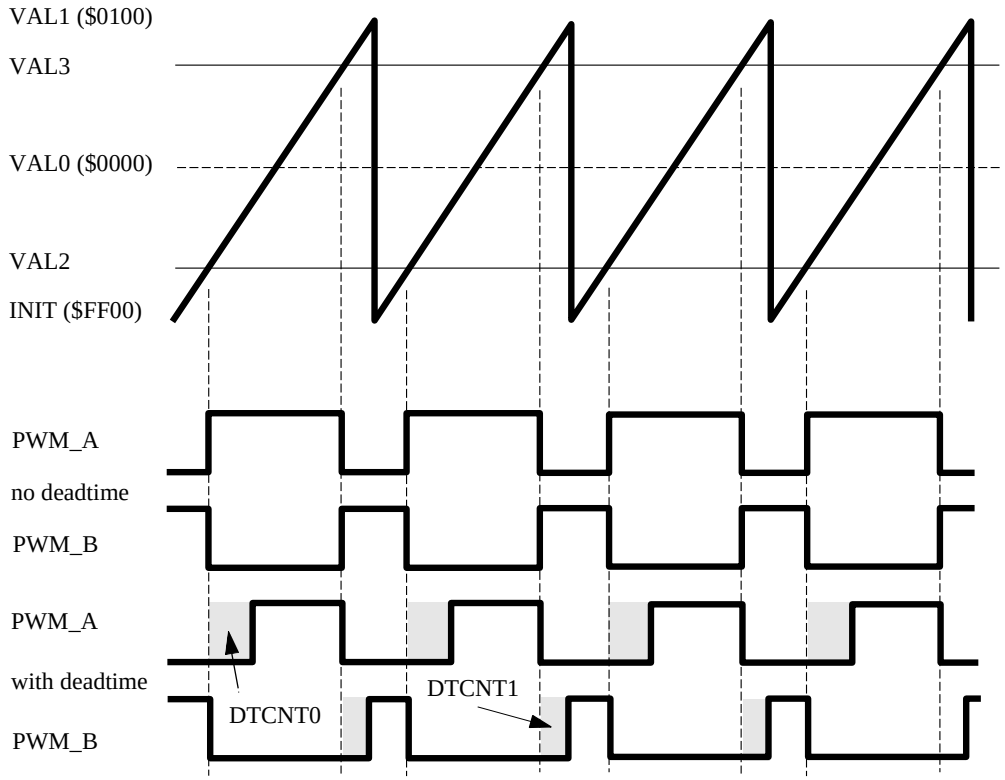

图18. 死区时间插入

3.2.6 分数延迟逻辑

对于需要比单个IPBus时钟周期更高分辨率的应用，可以使用分数延迟逻辑。这种分数延迟逻辑有助于在PWM_A和PWM_B输出的上升沿和下降沿上实现精细分辨率。通过设置FRCTRL[FRACx_EN]来启用分数延迟逻辑。除了由VAL2、VAL3、VAL4或VAL5寄存器指定的开启和关闭计数外，FRACVALx寄存器可用作分数时钟周期。FRACVAL1寄存器用作由VAL1定义的PWM周期的分数增量。如果FRACVAL1被编程为非零值，那么VAL1寄存器的最大值在无符号使用时为0xFFFE，在有符号使用时为0x7FFE。这个限制是为了避免在累积更多分数周期时发生计数器溢出。

分数使能（1、23、45）和分数值（1至5）都是双缓冲的，并与数值寄存器同时重新加载。

对于每个PWM周期，比较点的值会加1。此比较值会增大到当分数寄存器的双缓冲值加上5位累积分数值发生溢出时。累积分数值从零开始，因此第一个PWM周期不可能发生溢出。

在每个PWM周期结束时，如果相应的分数使能被设置，那么累积分数值会按分数值（双缓冲值）递增。累积分数值为5位，因此在溢出情况下只保留余数。如果相应的分数使能被清除，则累积分数值会被重置。这是重置累积分数值的唯一方法。

软件无法访问累积分数值。

例如，假设以下条件：

- INIT = 0x0000
- VAL1 = 0x000F
- VAL2 = 0x0000
- VAL3 = 0x0007
- FRACVAL3 = 0x00

这会导致PWM输出具有50%的占空比。当计数值为从0x0到0x7时，PWM输出为高电平；到达该点后，PWM输出变为低电平，直到计数器达到最大值0xF。

如果FRACVAL3的值变为0x17，则PWM输出的有效时间 = 8个周期 + (23 / 32)个周期 = 8.719个周期。因此，高电平占空比 = (8.719个周期 / 16个周期) x 100% = 54.49%。

另一种情况涉及使用FRACVAL1寄存器微调PWM周期。如果用户想要100.25个时钟周期的长度，可以将VAL1编程为0x0064，FRACVAL1编程为0x4000。分数值会累积，这样每4个PWM周期会比原来多1个时钟周期（即101而不是100）。PWM输出的上升沿和下降沿也会使用累积的分数来延迟其边沿，并在连续的周期对应边沿之间保持一致的100.25个周期间隔。

3.2.6.1 不使用NanoEdge植入模块的分数延迟逻辑

对于不支持NanoEdge（纳秒级边沿）植入功能的子模块，PWM可以使用抖动技术来模拟精细边沿控制。通过设置FRCTRL[FRAC1_EN]、FRCTRL[FRAC23_EN]和FRCTRL[FRAC45_EN]位来启用此功能。PWM周期或PWM边沿会在最接近的整数值附近进行抖动，以达到与编程的分数值相等的平均值。增加的周期是基于分数部分的累积。例如，如果用户希望PWM周期为50.25个时钟周期，可以将VAL1编程为0x0032，FRACVAL1编程为0x4000。大多数情况下，PWM周期将为50个时钟周期，但偶尔会出现51个时钟周期，从而实现50.25个时钟周期的长期平均值。

对于不支持NanoEdge植入功能的子模块，时钟频率不需要任何特定值即可正确运行。

3.2.7 输出逻辑

图19展示了每个子模块的输出逻辑，包括每个PWM输出如何具有独立的故障禁用、极性控制和输出使能。这在与外部电路对接时允许有最大的灵活性。

从死区时间逻辑输出的PWM23和PWM45信号（如图19所示）是正的真信号。换句话说，高电平信号必须导致PWM逆变器中相应的晶体管导通。PWM输出引脚开启或关闭晶体管所需的电压水平是该引脚和晶体管之间的逻辑函数。因此，在启用输出引脚之前，用户必须对OCTRL[POLA]和OCTRL[POLB]进行编程。故障条件可能导致PWM输出处于高阻态、强制为逻辑1状态或逻辑0状态。这取决于OCTRL[PWMxFS]字段中编程的值。

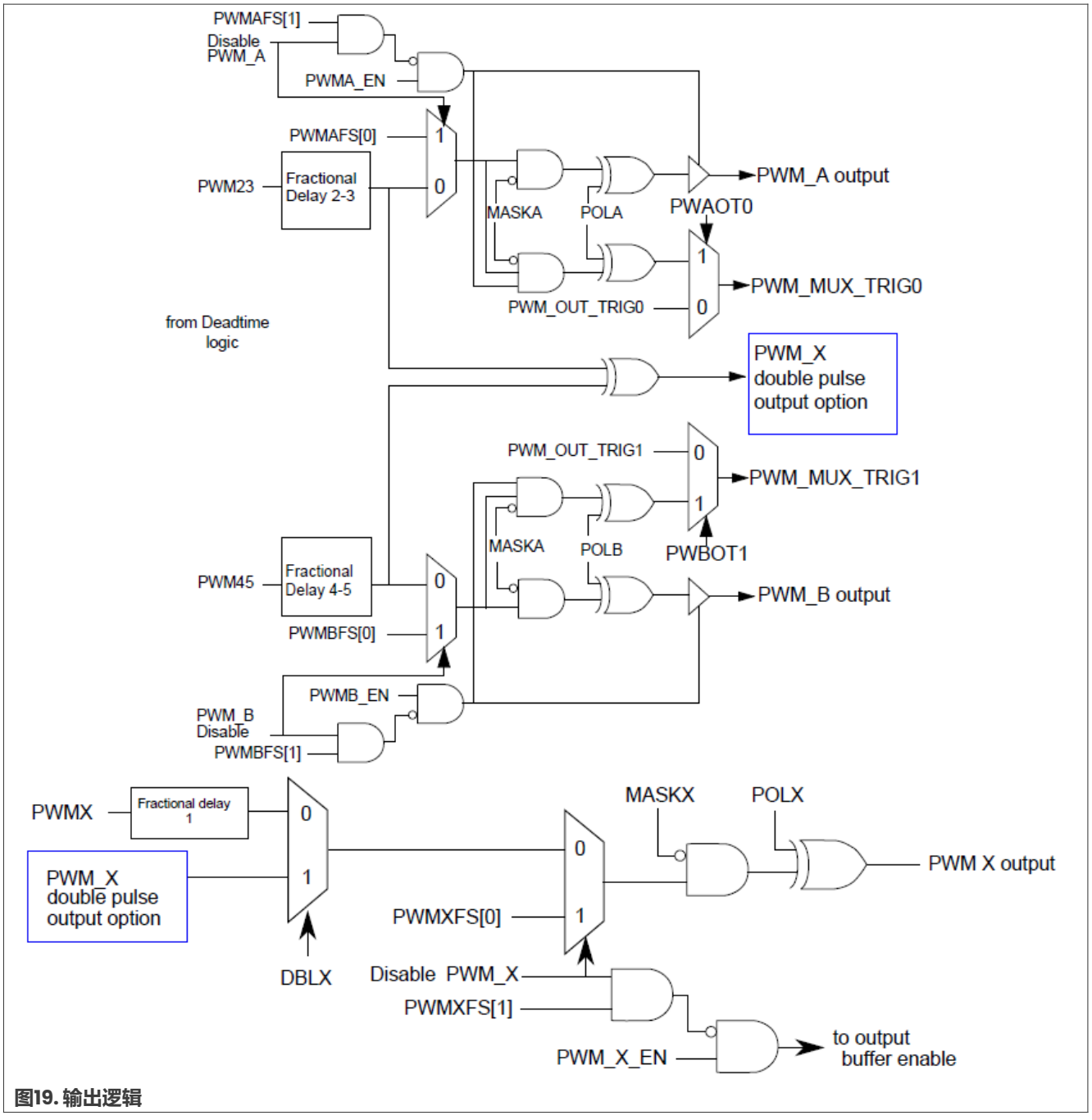

图19. 输出逻辑

3.2.8 E-capture逻辑

图20是E-capture电路的框图。信号进入引脚输入后，分为两条路径：

- 一条路径直接通往多路复用器输入，软件可以将信号直接传递给捕获逻辑进行处理。
- 另一条路径将信号连接到一个8位计数器，该计数器对输入信号的上升沿和下降沿进行计数。

这个计数器的输出与用户指定的一个8位数值（EDGCMPx）进行比较。如果这两个值相等，比较器会生成一个脉冲来重置计数器。为了让捕获逻辑处理这个脉冲，它也被提供给多路复用器的输入，软件可以在此处选择它。这个特性允许该模块计数指定数量的边沿事件，然后执行捕获并触发中断。

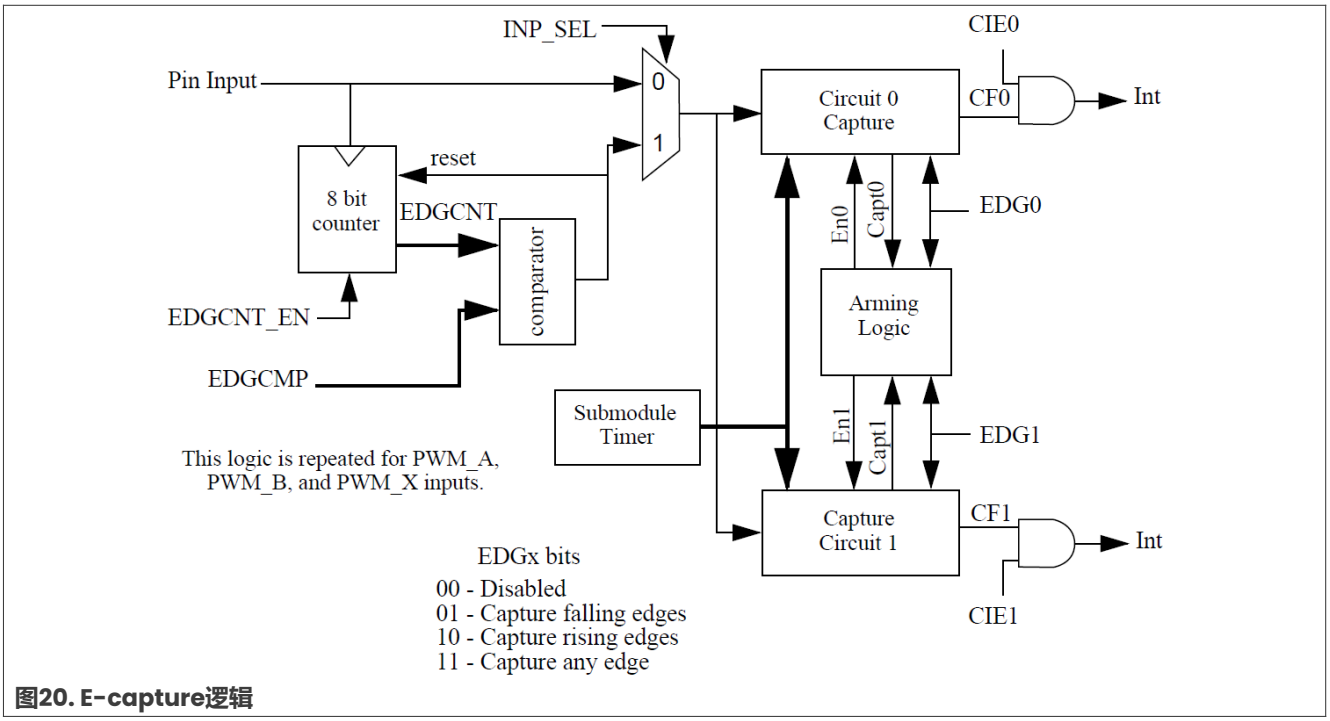

图20. E-capture逻辑

基于模式的选择，多路复用器会选择引脚输入或计数器/比较器电路的比较器输出，而该输出已由捕获逻辑处理。被选中的信号会被路由到两个独立的捕获电路，这两个电路协同工作以捕获信号的连续边沿。CAPTCTRLx[EDGx1]和CAPTCTRLx[EDGx0]确定每个电路捕获的边沿类型，其功能如图20所列。此外，一个装弹逻辑控制捕获电路的操作，允许在自由运行（连续）模式或单发模式下执行捕获。在自由运行模式下，捕获序列无限期地执行。如果两个捕获电路都被启用，它们会以乒乓方式协同工作，其中一个电路的捕获事件会导致另一个电路装弹，反之亦然。在单发模式下，只执行一个捕获序列。如果两个捕获电路都被启用，捕获电路0首先被装弹，当捕获事件发生时，捕获电路1再被装弹。一旦发生第二次捕获，进一步的捕获被禁用，直到另一个捕获序列被启动。两个捕获电路都还能够向CPU生成中断。

3.2.9 故障保护

故障保护可以控制任意组合的PWM输出引脚。逻辑1会在任何FAULTx引脚上生成故障。其极性可以通过FCTRL[FLVL]进行更改。每个FAULTx引脚可以任意地映射到任何一个PWM输出。当故障保护硬件禁用PWM输出时，PWM生成器仍会继续运行。只有输出引脚会被强制置为逻辑0、逻辑1或高阻态，具体取决于OCTRL[PWMxFS]的值。

故障解码器会禁用由故障逻辑和禁用映射（DISMAPn）寄存器选定的PWM引脚。图21展示了故障禁用逻辑的一个示例。DISMAPn中每一组的位控制单个PWM引脚的映射，详见表1。

即使PWM模块未启用，故障保护也会启用。因此，故障会被锁存，必须将其清除以防止在启用PWM时产生中断。

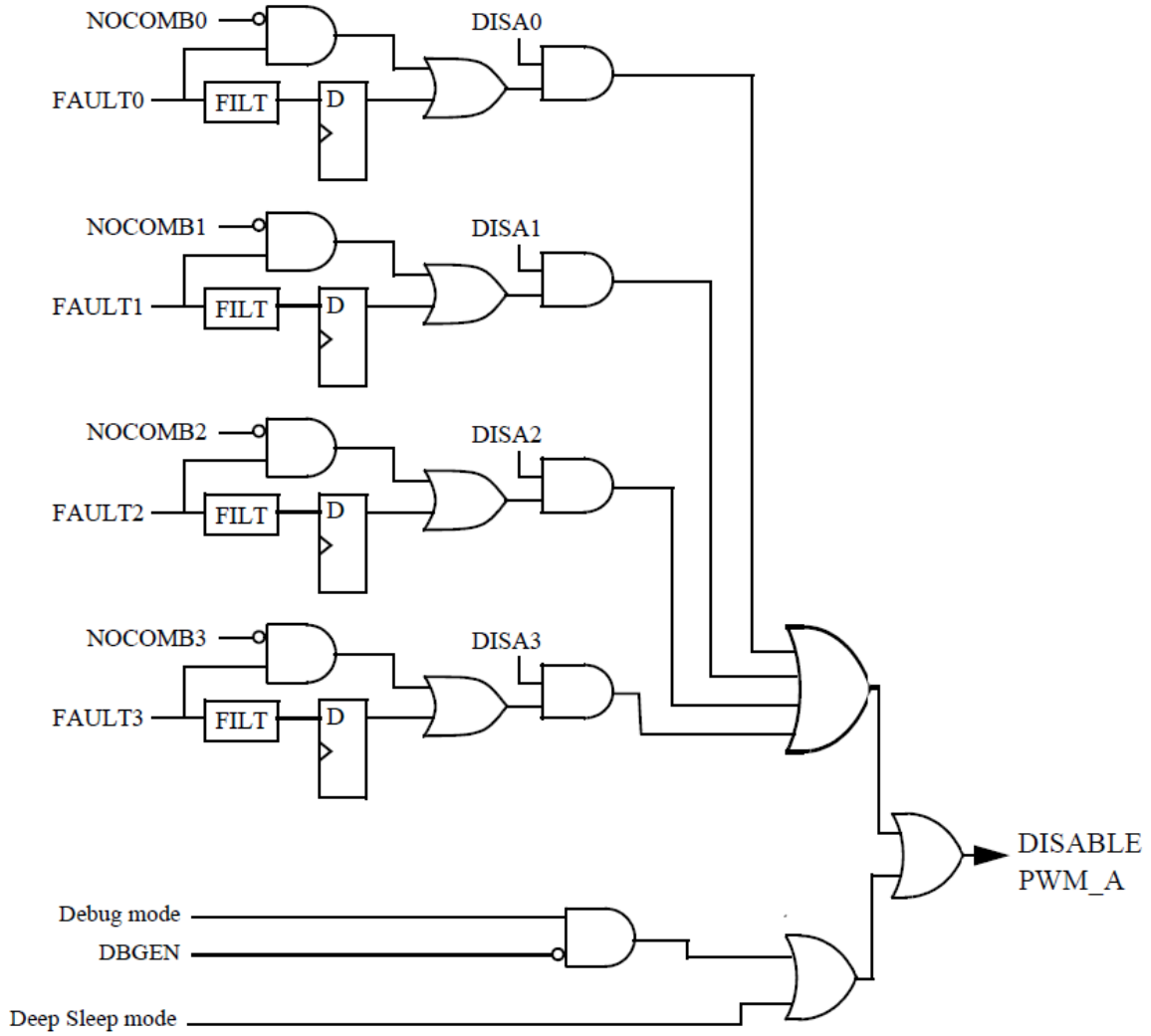

图21. PWM_A的故障解码器

表1. 故障映射

PWM子模块引脚	控制寄存器位
PWM_A	DISMAP0[DIS0A]
PWM_B	DISMAP0[DIS0B]
PWM_C	DISMAP0[DIS0X]

3.2.9.1 故障引脚滤波器

每个故障引脚都配备了一个可编程滤波器，该滤波器可以被旁路。滤波器的采样周期可通过FFILT[FILT_PER]进行调整。在输入变换被识别之前必须连续一致的采样次数可使用FFILT[FILT_CNT]进行调整。将FFILT[FILT_PER]设置为0会禁用给定FAULTx引脚的输入滤波器。

当在滤波后的FAULTx引脚上检测到逻辑0（如果FCTRL[FLVLx]已设置则为逻辑1）时，相应的FSTS[FFPINx]和FSTS[FFLAGx]位会被设置。如果滤波后的FAULTx引脚保持为0，FSTS[FFPINx]也会保持设置状态。要清除FSTS[FFLAGx]，需要向该位写入逻辑1。

如果FIEx位和FAULTx引脚中断使能位已设置,那么FSTS[FFLAGx]会生成一个CPU中断请求。中断请求锁存会一直保持设置状态，直到：

- 软件通过向FSTS[FFLAGx]写入逻辑1来清除该位
- 软件通过向FIEx位写入逻辑0来清除该位
- 发生复位

即使启用了滤波器，从FAULTx输入到PWM引脚之间仍存在一条组合路径。当FCTRL20[NOCOMBx] = 0时，该路径会旁路滤波器。此逻辑还能在PWM模块时钟丢失的情况下保持故障状态。

3.2.9.2 自动故障清除

设置自动清除模式位FCTRL[FAUTOx]可以将FAULTx引脚的故障配置为自动清除。

图22显示了以下内容：

- 当FCTRL[FAUTOx]已设置时，一旦FAULTx引脚返回到逻辑1，并且一个新的PWM完整周期或半周期开始，被禁用的PWM引脚将被启用。
- 如果FSTS[FFULLx]已设置，且FAULTx上的故障条件消失，被禁用的PWM引脚将在下一个完整周期的开始时被启用。
- 如果FSTS[FHALFx]已设置，被禁用的PWM引脚将在半周期的开始时被启用。当FCTRL[FAUTOx]已设置时，清除FSTS[FFLAGx]不会影响被禁用的PWM引脚。

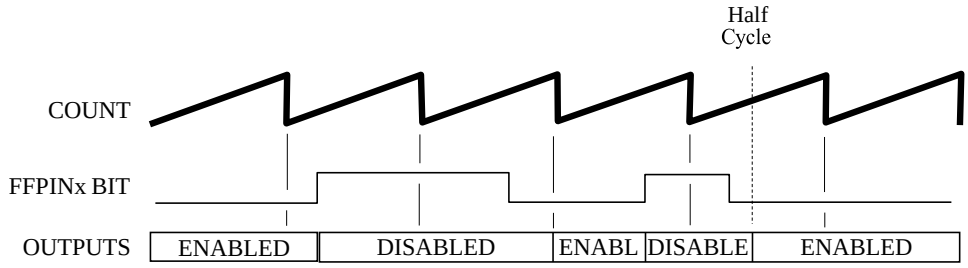

图22. 自动故障清除

3.2.9.3 手动故障清除

设置自动清除模式位FCTRL[FAUTOx]可以将FAULTx引脚的故障配置为自动清除：

清除自动清除模式位FCTRL[FSAFEx]可以将FAULTx引脚的故障配置为手动清除：

- 如果故障安全模式位FCTRL[FSAFEx]未设置，则由FAULTx引脚禁用的PWM引脚会在以下情况下启用：
  - 软件清除相应的FSTS[FFLAGx]标志。
  - 无论FAULTx引脚上滤波器检测到的逻辑电平如何，下一个PWM完整周期或半周期开始时引脚将被启用。如图23所示：
    - 如果FSTS[FFULLx]已设置，被禁用的PWM引脚将在完整周期的开始时被启用。
    - 如果FSTS[FHALFx]已设置，被禁用的PWM引脚将在半周期的开始时被启用。

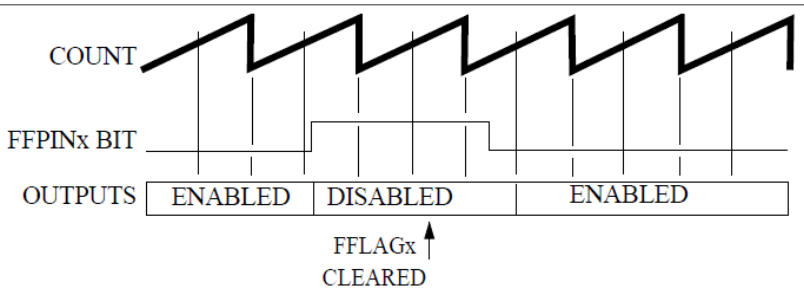

图23. 手动故障清除（FCTRL[FSAFEx] = 0, FSTS[FFULLx] = 1）

- 如果故障安全模式位FCTRL[FSAFEx]已设置，则由FAULTx引脚禁用的PWM引脚会在以下情况下被启用：
  – 软件清除相应的FSTS[FFLAGx]标志。
  – 在下一个PWM完整周期或半周期开始时，滤波器在FAULTx引脚上检测到逻辑零。如图24所示：
    – 如果FSTS[FFULLx]已设置，被禁用的PWM引脚将在完整周期开始时被启用。
    – 如果FSTS[FHALFx]已设置，被禁用的PWM引脚将在半周期开始时被启用。

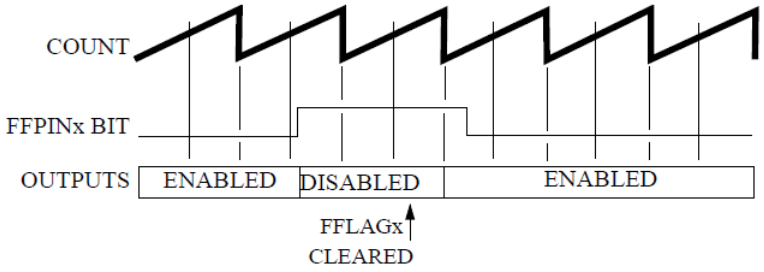

图24. 手动故障清除（FCTRL[FSAFEx] = 1, FSTS[FHALFx] = 1）

3.2.9.4 故障测试

FTST[FTEST]用于模拟该故障通道内每个故障输入上的故障条件。

4 板载实验

MCX N系列的示例代码有助于理解不同PWM功能的实现。

***注：**本应用笔记仅描述了MCX Nx4x的实验。关于使用MCX N23x进行实验的板卡连接信息，请参考软件包中的readme文件。*

要测试PWM功能，使用FRDM-MCXN947板，如图25所示。示例代码基于SDK示例：`\SDK_2_14_0_FRDM-MCXN947\boards\frdmmcxn947\driver_examples\pwm`

用户必须熟悉上述示例及相关硬件。

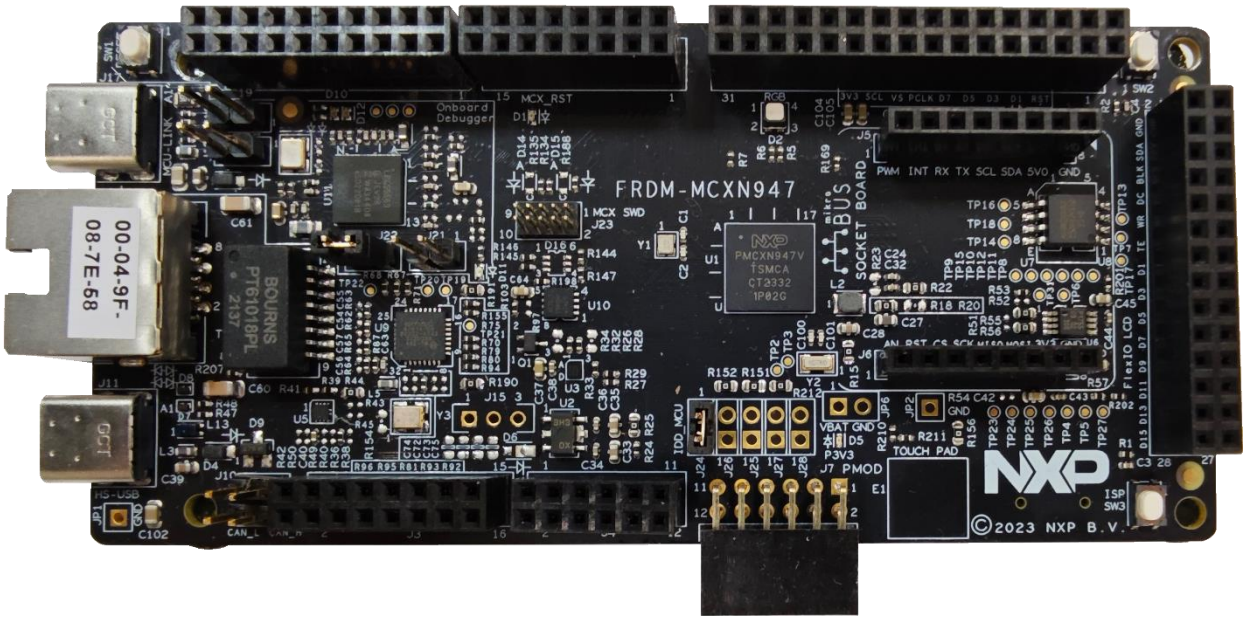

图25. FRDM-MCXN947板

4.1 板卡设置

FDRM-MCXN947板自带板载调试器。使用USB-C线缆通过J17接口连接到板卡，用于下载和调试。

为了捕获输出波形，用户必须使用示波器或逻辑分析仪探测PWM信号。如图26所示：

- PWM1_A0 (P2_6) 连接到 J3-15。
- PWM1_B0 (P2_7) 连接到 J3-13。
- TEST_PIN (P2_4) 连接到 J3-11。
- ADC0_A2 (P4_23) 连接到 J8-28。

***注：** P2_4引脚被用作TEST_PIN，用于指示ADC触发和E-Capture的动作。*

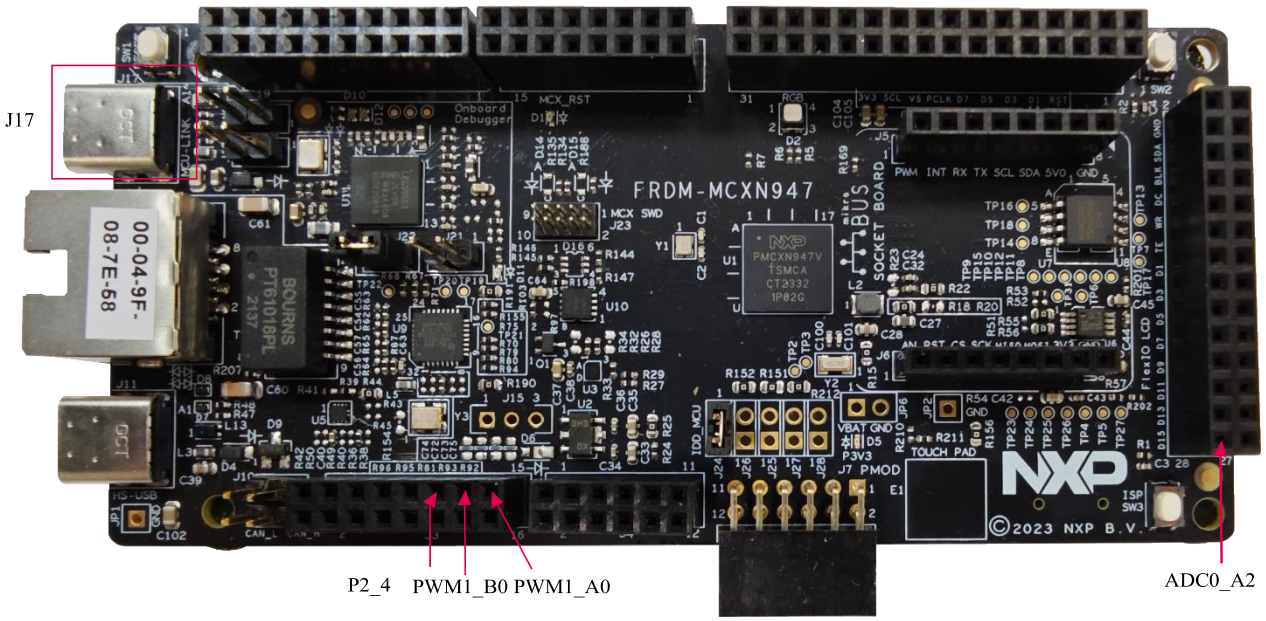

图26. FRDM-MCXN947板连接

4.2 FlexPWM功能测试

要实现FlexPWM的不同功能，需要在代码中进行多项配置。

注：在本测试中，以子模块0为例。如果使用其他子模块，请查找相应的寄存器。

一些主要配置如下：

- 初始化时钟，将FRO 12M连接到FLEXCOMM4，用于调试控制台。
- 初始化必须使用的引脚，包括P1_8 (FC4_P0)、P1_9 (FC4_P1)、P2_6 (PWM1_A0)、P2_7 (PWM1_B0)、P4_23 (ADC0_A2)和P2_4 (TEST_PIN)。
- 初始化并启用FlexPWM模块：
  - 使用IP总线时钟作为PWM子模块的源时钟。
  - 使用本地重载信号来重载寄存器。
  - 使用全周期重载作为重载模式。
  - 为PWM_A和PWM_B设置工作模式，包括独立模式和互补模式，具体取决于PWM的工作模式。
  - 为子模块0的PWM_A和PWM_B设置PWM故障禁用映射。
  - 参数设置如下：
    - SM0INIT：指示PWM周期的开始
    - SM0VAL0：指示中心值
    - SM0VAL1：指示PWM周期的结束
    - SM0DTCNT0和SM0DTCNT1：设置死区时间值
  - 根据PWM模式配置其他相关寄存器，并启用PWM输出。
- 初始化并启用LPADC模块：
  - 使用FRO HF作为ADC模块的源时钟。
  - 使用VREF模块驱动的VREF_OUT作为参考电压。
  - 在启用前进行校准。
  - 设置转换命令（CMD）配置。

– 设置触发配置；配置触发器0和触发器1用于ADC触发。
– 启用LPADC中断。

在示例代码中，用户可以使用键盘输入来设置不同的PWM模式。

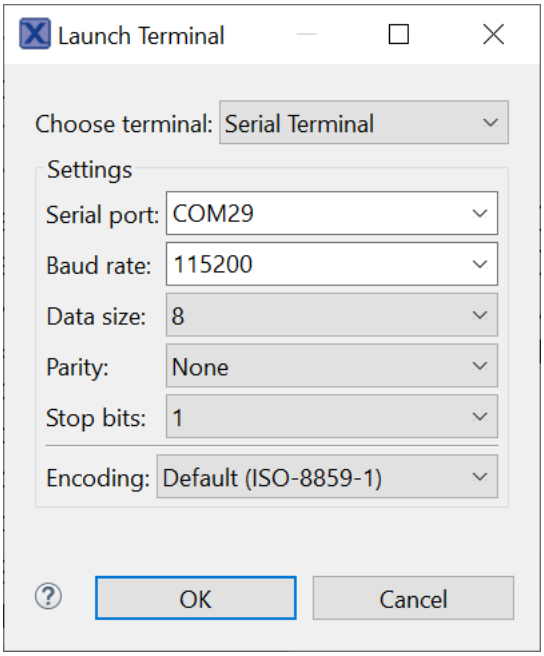

图27. UART调试终端配置

1. 打开一个UART调试终端，并按图27所示配置设置：
   - 波特率：115200
   - 数据位：8位
   - 奇偶校验位：无
   - 停止位：1
2. 按下复位按钮。终端显示提示信息，如图28所示。要选择PWM模式，用户可以从PC键盘输入'0'、'1'、'2'等。

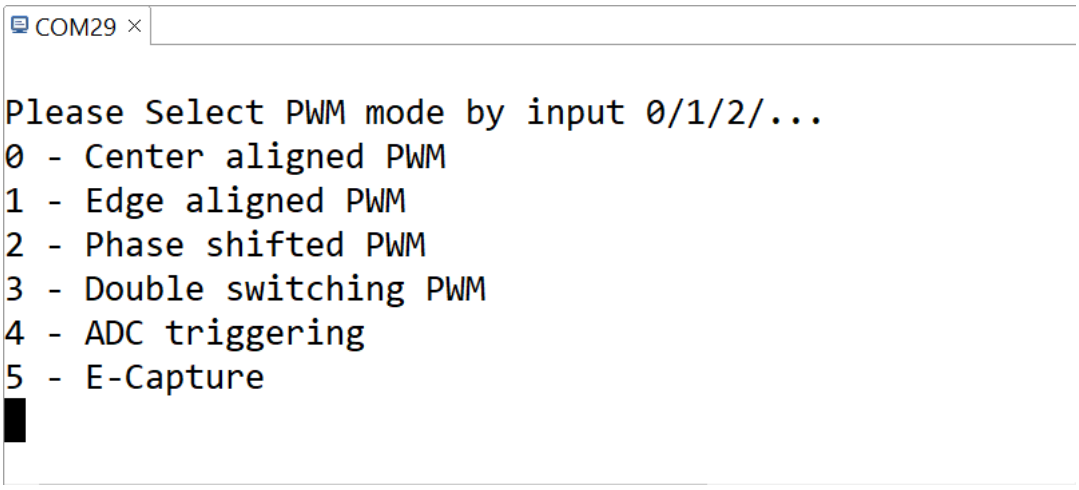

图28. PWM模式选择菜单

4.2.1 中心对齐PWM

在中心对齐PWM模式下：

- PWM_A和PWM_B以独立模式运行。
- SM0VAL2/SM0VAL3（SM0VAL4/SM0VAL5）分别定义PWM_A（PWM_B）的上升沿和下降沿。
- PWM_MODULO表示一个PWM周期内的总计数值。它等于PWM源时钟频率（本例中为150MHz）除以PWM频率（本例中为10kHz）。
- 将0设置为对称中心。SM0VAL2/SM0VAL3（SM0VAL4/SM0VAL5）的值关于0对称。
  中心对齐PWM的核心代码如下：

```
pwmHighPulse[0] = PWM_MODULO * dutyCyclePercent[0] / 100UL;     // PWM counter value for PWM_A
pwmHighPulse[1] = PWM_MODULO * dutyCyclePercent[1] / 100UL;     // PWM counter value for PWM_B

/* Center aligned PWM */
if(g_pwmMode == 0)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(pwmHighPulse[0] / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)( (pwmHighPulse[0] / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL4 = PWM_VAL4_VAL4((uint16_t)(-(pwmHighPulse[1] / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL5 = PWM_VAL5_VAL5((uint16_t)( (pwmHighPulse[1] / 2)));
}
```

- PWM_A和PWM_B分别以60%和40%的占空比在中心对齐PWM模式下工作，如图29所示。

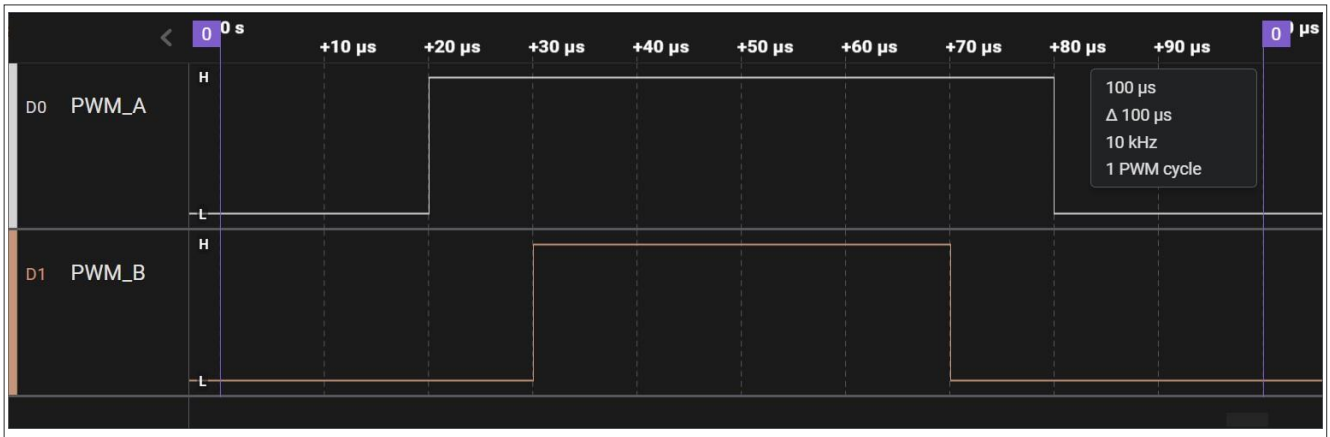

图29. 中心对齐PWM波形

4.2.2 边沿对齐PWM

在边沿对齐PWM模式下：

- PWM_A和PWM_B以独立模式运行。
- SM0VAL2/SM0VAL3（SM0VAL4/SM0VAL5）分别定义PWM_A（PWM_B）的上升沿和下降沿。
- 将SM0VAL2和SM0VAL4都设置为初始计数值。
- PWM_A和PWM_B同时生成上升沿，并通过设置SM0VAL3和SM0VAL5来实现不同占空比的输出。
  边沿对齐PWM的核心代码如下：

```
pwmHighPulse[0] = PWM_MODULO * dutyCyclePercent[0] / 100UL;     // PWM counter value for PWM_A
pwmHighPulse[1] = PWM_MODULO * dutyCyclePercent[1] / 100UL;     // PWM counter value for PWM_B

/* Edge aligned PWM */
else if(g_pwmMode == 1)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(PWM_MODULO / 2)));
```

```
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)(-(PWM_MODULO / 2) + pwmHighPulse[0]));
    BOARD_PWM_BASEADDR -> SM[0].VAL4 = PWM_VAL4_VAL4((uint16_t)(-(PWM_MODULO / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL5 = PWM_VAL5_VAL5((uint16_t)(-(PWM_MODULO / 2) + pwmHighPulse[1]));
}
```

- PWM_A和PWM_B分别以60%和40%的占空比在边沿对齐PWM模式下工作，如图30所示。

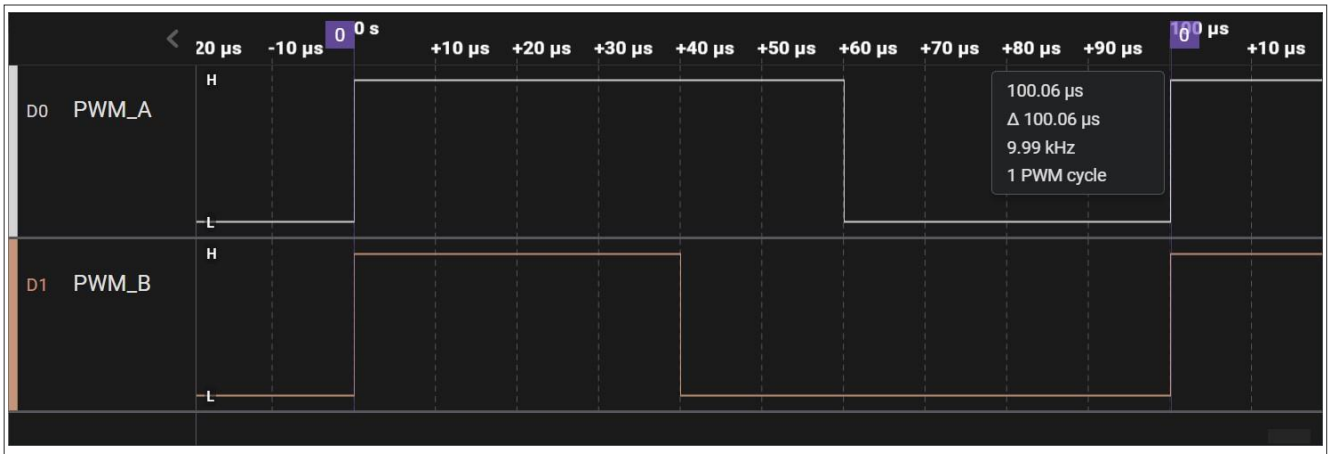

图30. 边沿对齐PWM波形

4.2.3 相移PWM

在相移PWM模式下：

- PWM_A和PWM_B以独立模式运行。
- SM0VAL2/SM0VAL3（SM0VAL4/SM0VAL5）分别定义PWM_A（PWM_B）的上升沿和下降沿。
- 通过SM0VAL2和SM0VAL3设置PWM_A的相位和占空比信息。
- 然后，使用SM0VAL4和SM0VAL5设置PWM_B相对于PWM_A的相移。
  相移PWM的核心代码如下：

```
pwmHighPulse[0] = PWM_MODULO * dutyCyclePercent[0] / 100UL;     // PWM counter value for PWM_A
phaseA = PWMA_PHASE * PWM_MODULO / 360UL;                       // Phase for PWM_A
phaseShift = PWM_PHASESHIFT * PWM_MODULO / 360UL;               // Phase shift for PWM_B
/* Phase shifted PWM */
else if(g_pwmMode == 2)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(PWM_MODULO / 2) + phaseA));
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)(-(PWM_MODULO / 2) + phaseA + pwmHighPulse[0]));
    BOARD_PWM_BASEADDR -> SM[0].VAL4 = PWM_VAL4_VAL4((uint16_t)(-(PWM_MODULO / 2) + phaseA + phaseShift));
    BOARD_PWM_BASEADDR -> SM[0].VAL5 = PWM_VAL5_VAL5((uint16_t)(-(PWM_MODULO / 2) + phaseA + phaseShift+
 pwmHighPulse[0]));
}
```

- PWM_A和PWM_B在相移PWM模式下工作：
  – PWM_A的相位为36度（1/10 PWM周期）。
  – PWM_B相对于PWM_A的相移为36度（1/10 PWM周期），如图31所示。

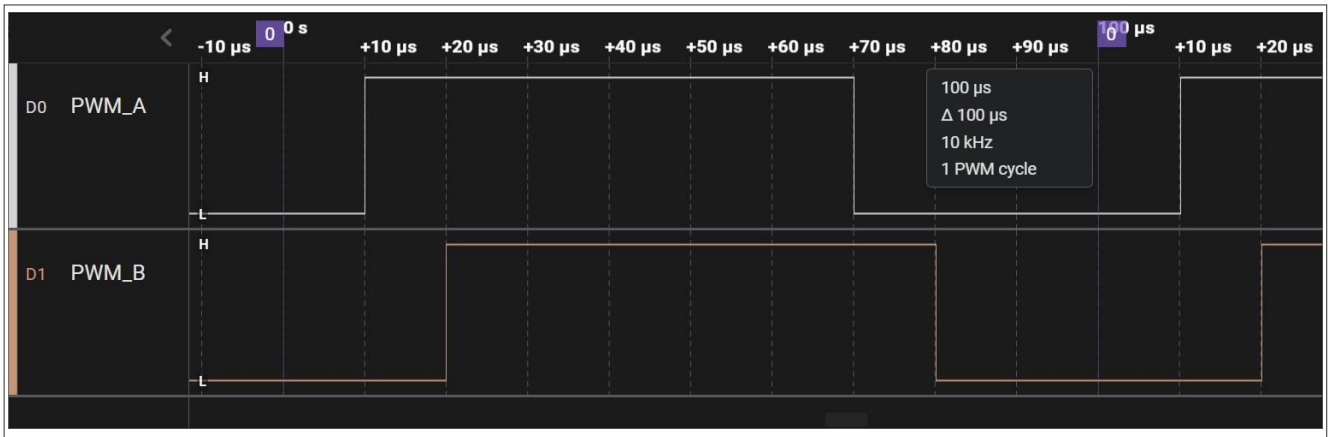

图31. 相移PWM波形

4.2.4 双开关PWM

在双开关PWM模式下：

- PWM_A和PWM_B以独立模式运行。
- SM0VAL2/SM0VAL3（SM0VAL4/SM0VAL5）分别定义PWM_A（PWM_B）的上升沿和下降沿。
- 用户可以灵活地设置PWM_A和PWM_B的上升沿和下降沿。
- 使用SM0VAL2和SM0VAL4分别设置PWM_A和PWM_B的上升沿。
- 使用SM0VAL3和SM0VAL5分别设置PWM_A和PWM_B的下降沿。图32显示了PWM_A源和PWM_B源。
- 当SM0CTRL[DBLEN] = 1且SM0CTRL[SPLIT] = 0时，PWM_A和PWM_B的输出为PWM_A源和PWM_B源的异或（XOR）
- 当SM0CTRL[DBLEN] = 1且SM0CTRL[SPLIT] = 1时，PWM_A的输出是当PWM_A源为1且PWM_B源为0时产生的脉冲。PWM_B的输出是当PWM_B源为1且PWM_A源为0时产生的脉冲。
  双开关PWM的核心代码如下：

```
risingA = PWMA_RISING * PWM_MODULO / 100UL;        // Rising value for PWM_A
fallingA = PWMA_FALLING * PWM_MODULO / 100UL;       // Falling value for PWM_A
risingB = PWMB_RISING * PWM_MODULO / 100UL;      // Rising  value for PWM_B
fallingB = PWMB_FALLING * PWM_MODULO / 100UL;      // Falling value for PWM B

/* Double switching PWM */
else if(g_pwmMode == 3)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(PWM_MODULO / 2) + risingA));
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)(-(PWM_MODULO / 2) + fallingA));
    BOARD_PWM_BASEADDR -> SM[0].VAL4 = PWM_VAL4_VAL4((uint16_t)(-(PWM_MODULO / 2) + risingB));
    BOARD_PWM_BASEADDR -> SM[0].VAL5 = PWM_VAL5_VAL5((uint16_t)(-(PWM_MODULO / 2) + fallingB));
    if(g_dblen_bit == 1)
    {
        BOARD_PWM_BASEADDR -> SM[0].CTRL |= PWM_CTRL_DBLEN_MASK;
    }
    if(g_split_bit == 1)
    {
        BOARD_PWM_BASEADDR -> SM[0].CTRL |= PWM_CTRL_SPLIT_MASK;
    }
}
```

- PWM_A的上升沿和下降沿分别设置为20和60（范围从0到100）。
- PWM_B的上升沿和下降沿分别设置为40和80（范围从0到100），如图32所示。

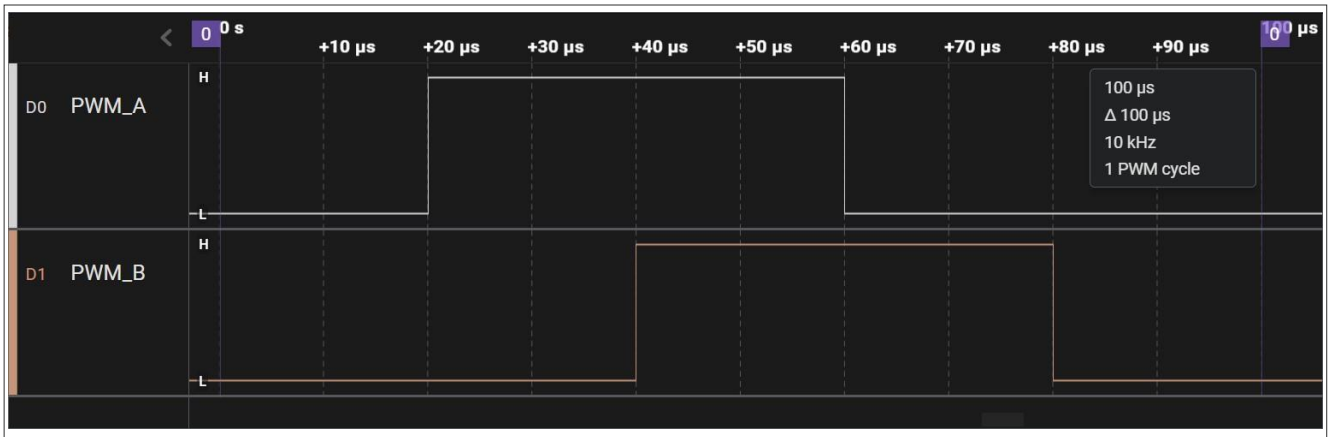

图32. 双开关PWM波形（DBLEN = 0, SPLIT = 0）

- 当SM0CTRL[DBLEN] = 1时，启用双开关PWM操作，如图33所示。

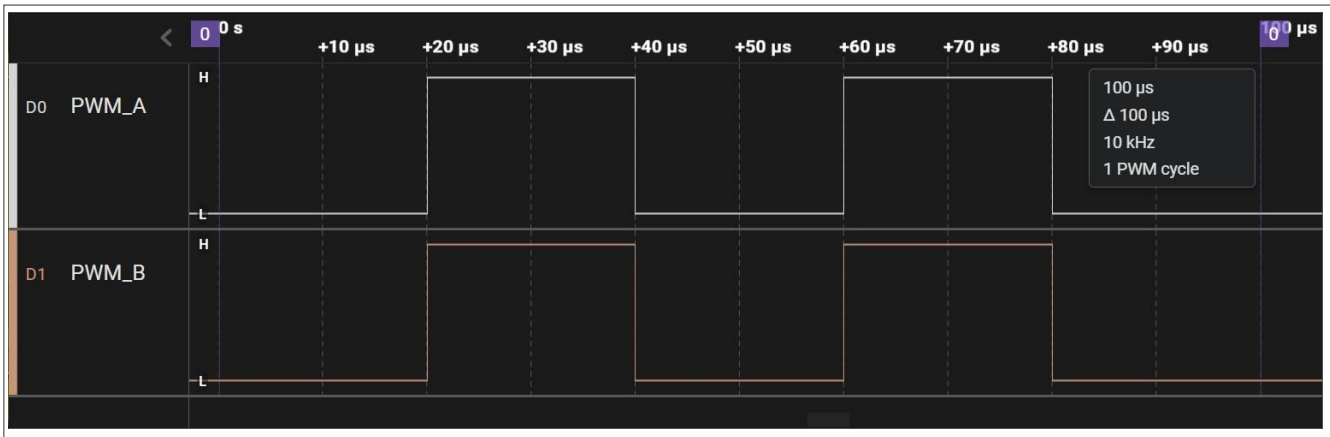

图33. 双开关PWM波形（DBLEN = 1, SPLIT = 0）

- 图34分别表示在SM0CTRL[SPLIT] = 0且SM0CTRL[SPLIT] = 1条件下PWM_A和PWM_B的输出。

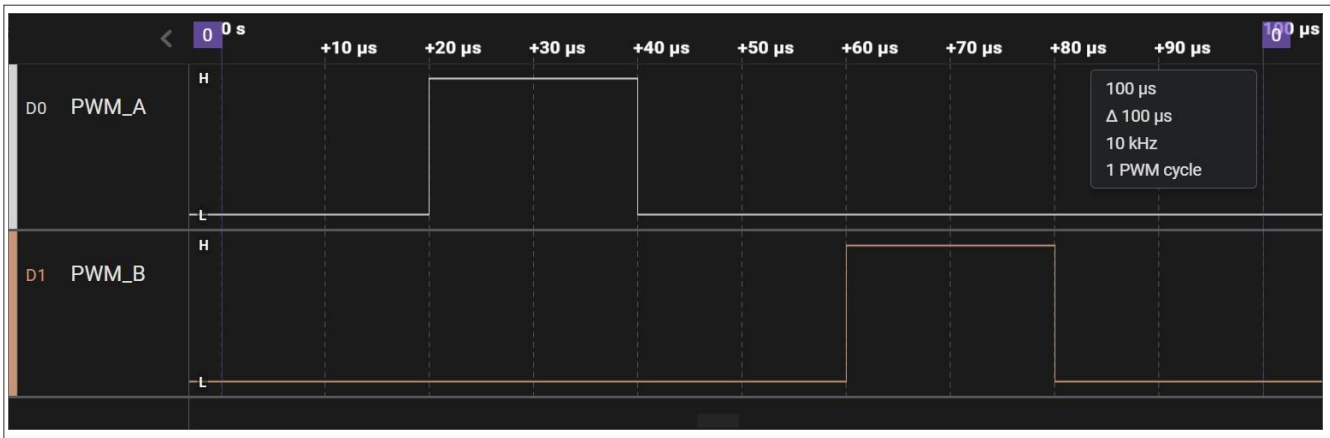

图34. 双开关PWM波形（DBLEN = 1, SPLIT = 1）

4.2.5 ADC触发

在ADC触发模式下：

- PWM_A和PWM_B以互补模式运行。

- SM0VAL2和SM0VAL3分别定义PWM_A的上升沿和下降沿。
- PWM_B与PWM_A互补，因此SM0VAL4和SM0VAL5可自由配置为ADC触发源。
- 要在SM0VAL4和SM0VAL5上启用相应的触发源（PWM_OUT_TRIG0和PWM_OUT_TRIG1），需设置SM0TCTRL[OUT_TRIG_EN]的第4位和第5位。
- 配置ADC0的触发器0和触发器1。
- 将PWM1_SM0_MUX_TRIG0链接到ADC0_TRIG0，将PWM1_SM0_MUX_TRIG1链接到ADC0_TRIG1。
- 启用触发器0和触发器1中断后，ADC触发功能就绪。当计数器值与SM0VAL4和SM0VAL5的值匹配时，会生成ADC0触发器0和ADC0触发器1中断。
- 在互补模式下，支持死区时间操作。SM0DTCNT0[DTCNT0]和SM0DTCNT1[DTCNT1]分别用于在PWM_A的上升沿和PWM_B的下降沿插入死区时间。
- P2_4用作TEST_PIN，来显示LPADC中断。
  ADC触发的核心代码如下：

```
pwmHighPulse[0] = PWM_MODULO * dutyCyclePercent[0] / 100UL;    // PWM counter value for PWM_A
pwmTrigger[0] = PWM_TRIG0 * PWM_MODULO / 100UL;                  // Trigger0 value set
pwmTrigger[1] = PWM_TRIG1 * PWM_MODULO / 100UL;                  // Trigger1 value set
deadTimeVal = PWM_DEADTIME * (PWM_SRC_CLK_FREQ / 1000000UL) / 1000UL; // Deadtime value set

/* ADC triggering */
else if(g_pwmMode == 4)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(pwmHighPulse[0] / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)( (pwmHighPulse[0] / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL4 = PWM_VAL4_VAL4((uint16_t)(-(PWM_MODULO / 2) + pwmTrigger[0]));;
    BOARD_PWM_BASEADDR -> SM[0].VAL5 = PWM_VAL5_VAL5((uint16_t)(-(PWM_MODULO / 2) + pwmTrigger[1]));
    BOARD_PWM_BASEADDR -> SM[0].DTCNT0 = deadTimeVal;
    BOARD_PWM_BASEADDR -> SM[0].DTCNT1 = deadTimeVal;
    /* PWM_OUT_TRIG0 will set when the counter value matches the VAL4 value*/
    BOARD_PWM_BASEADDR -> SM[0].TCTRL |= PWM_TCTRL_OUT_TRIG_EN(1 << 4);
    /* PWM_OUT_TRIG1 will set when the counter value matches the VAL5 value*/
    BOARD_PWM_BASEADDR -> SM[0].TCTRL |= PWM_TCTRL_OUT_TRIG_EN(1 << 5);
}

void DEMO_LPADC_IRQ_HANDLER_FUNC(void)
{
   GPIO_PortToggle(GPIO2, 1U << 4U);               // Indicate entry interrupt
   g_LpadcInterruptCounter++;
#if (defined(FSL_FEATURE_LPADC_FIFO_COUNT) && (FSL_FEATURE_LPADC_FIFO_COUNT == 2U))
   if (LPADC_GetConvResult(DEMO_LPADC_BASE, &g_LpadcResultConfigStruct, 0U))
#else
   if (LPADC_GetConvResult(DEMO_LPADC_BASE, &g_LpadcResultConfigStruct))
#endif   /* FSL_FEATURE_LPADC_FIFO_COUNT */
   {
         g_LpadcConversionCompletedFlag = true;
   }

   DEMO_LPADC_BASE -> STAT |= ADC_STAT_TCOMP_INT_MASK; // Clear trigger interrupt flag

   SDK_ISR_EXIT_BARRIER;
}
```

- PWM_A和PWM_B以60%的占空比在互补模式下工作。PWM_A和PWM_B在40和60（范围从0到100）处实现ADC触发，如图35所示。

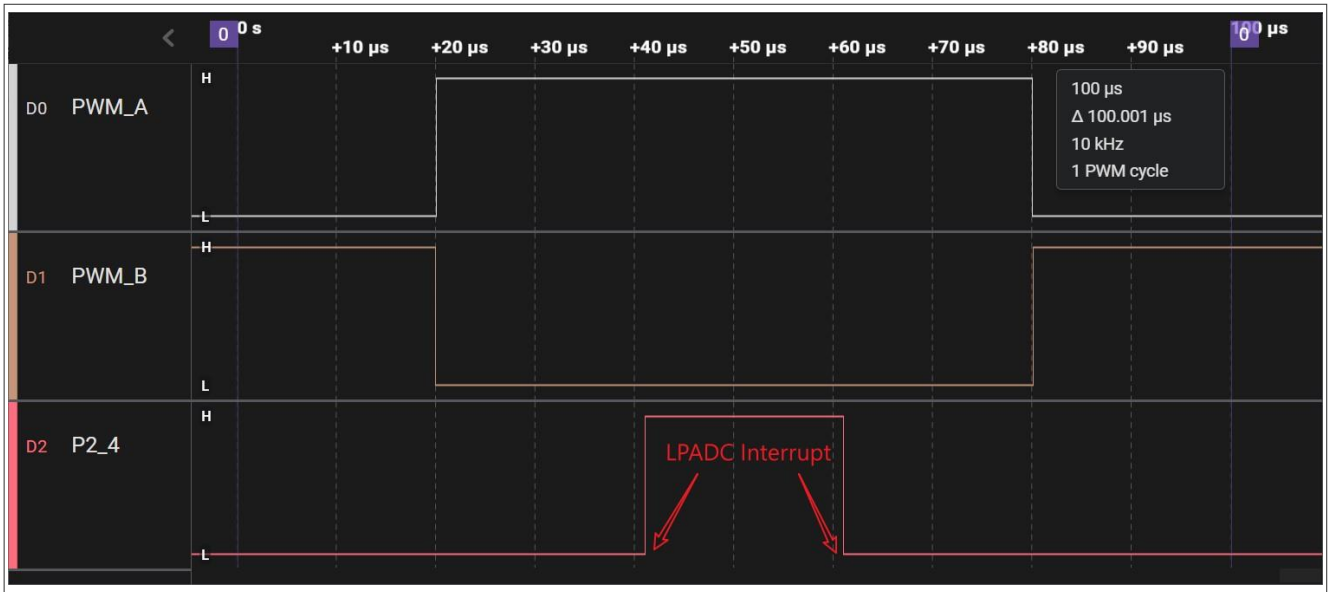

图35. ADC触发波形

- 图36显示了ADC采样的结果，当输入为3.3V时，采样值接近4095。

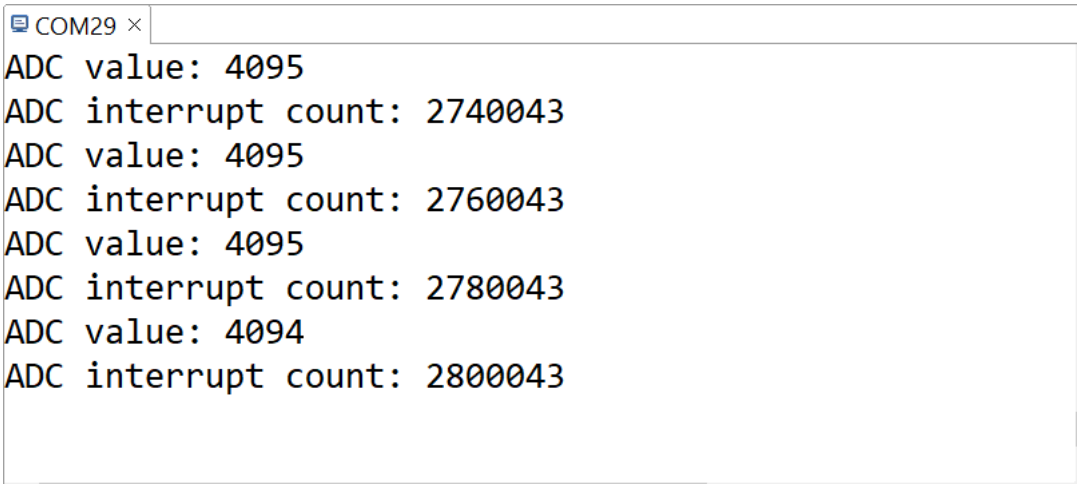

图36. ADC采样值

4.2.6 E-capture

在E-capture模式下：

- PWM_A通常作为PWM输出，而PWM_B则作为E-capture的输入。
- SM0VAL2和SM0VAL3分别定义PWM_A的上升沿和下降沿。
- 对于不同的应用，用户可以选择不同的信号源作为E-Capture的输入。
- 当SM0CAPTCTRLB[INP_SELB] = 0时，选择原始PWM_B输入信号作为源。
- 当SM0CAPTCTRLB[INP_SELB] = 1时，选择边沿计数器/比较器的输出作为源。
- 此外，当SM0CAPTCTRLB[INP_SELB] = 0时，用户可以通过设置SM0CAPTCTRLB[EDGB0]/[EDGB1]来设置要捕获的边沿类型。
- 当SM0CAPTCTRLB[INP_SELB] = 1时，用户可以通过设置SM0CAPTCOMPB[EDGCMPB]来设置边沿计数器的值。

- 如果用户想使用单发模式，可以通过SM0CAPTCTRLB[ONESHOTB]启用。P2_4引脚用作TEST_PIN，来显示捕获动作。
  E-capture的核心代码如下：

```
pwmHighPulse[0] = PWM_MODULO * dutyCyclePercent[0] / 100U;     // PWM counter value for PWM_A

/* E-Capture */
else if(g_pwmMode == 5)
{
    BOARD_PWM_BASEADDR -> SM[0].VAL2 = PWM_VAL2_VAL2((uint16_t)(-(PWM_MODULO / 2)));
    BOARD_PWM_BASEADDR -> SM[0].VAL3 = PWM_VAL3_VAL3((uint16_t)(-(PWM_MODULO / 2) + pwmHighPulse[0]));

    /* 0: Raw PWM_B input signal selected as source, 1: Set the output of the edge counter/compare circuitry as
 the source for the input capture */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_INP_SELB(0U);
    /* Capture 0 capture rising edges, only works when INP_SELB = 0 */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_EDGB0(2U);
    /* Capture 1 capture falling edges, only works when INP_SELB = 0 */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_EDGB1(1U);
    /* Edge Counter B Enabled */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_EDGCNTB_EN(1U);
    /* Enabled one shot mode */
    // BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_ONESHOTB(1U);
    /* Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_ARMB(1U);
    /* Set the compare value associated with the edge counter for the PWM_B input capture, works when INP_SELB =
 1 */
    BOARD_PWM_BASEADDR -> SM[0].CAPTCOMPB = PWM_CAPTCOMPB_EDGCMPB(5U);

    BOARD_PWM_BASEADDR -> SM[0].INTEN     |= PWM_INTEN_CB0IE(1U);      // Counter B0 interrupt Enabled
    BOARD_PWM_BASEADDR -> SM[0].INTEN     |= PWM_INTEN_CB1IE(1U);      // Counter B1 interrupt Enabled

    EnableIRQ(FLEXPWM1_SUBMODULE0_IRQn);
}

void FLEXPWM1_SUBMODULE0_IRQHandler(void)
{
    GPIO_PortToggle(GPIO2, 1U << 4U);                                  // Indicate entry interrupt
    BOARD_PWM_BASEADDR -> SM[0].STS |= PWM_STS_CFB0(1U);               // Clear Capture B0 interrupt flag
    BOARD_PWM_BASEADDR -> SM[0].STS |= PWM_STS_CFB1(1U);               // Clear Capture B1 interrupt flag
    // BOARD_PWM_BASEADDR -> SM[0].CAPTCTRLB |= PWM_CAPTCTRLB_ARMB(1U); // Re-arm Capture circuit

    SDK_ISR_EXIT_BARRIER;
}
```

- 在此模式下，PWM_B被设置为输入，并以60%的占空比采样PWM_A的信号。
- 当SM0CAPTCTRLB[INP_SELB] = 0时，捕获0被设置为捕获上升沿，捕获1被设置为捕获下降沿，如图37所示。

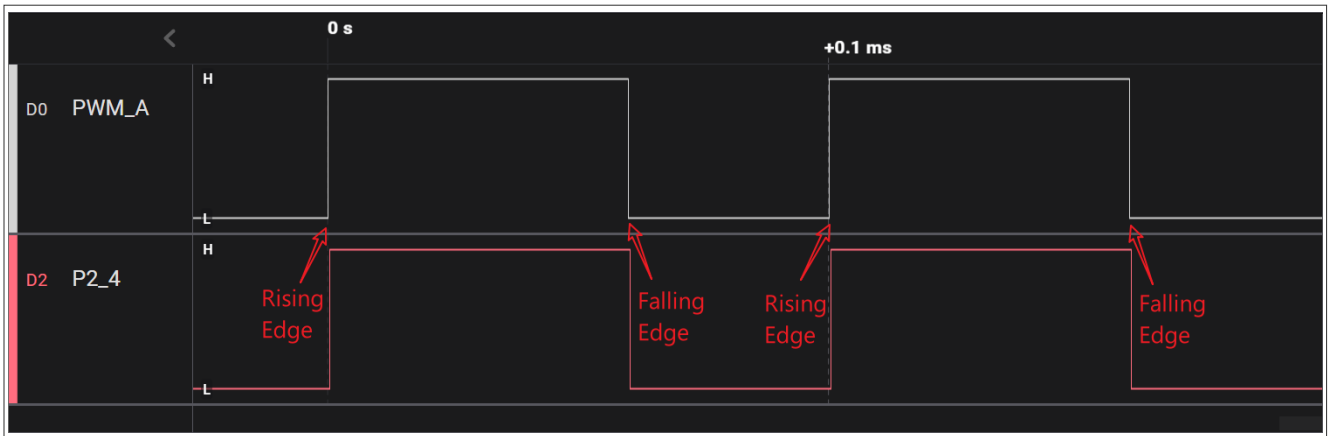

图37. E-Capture波形（INP_SELB = 0）

- 当SM0CAPTCTRLB[INP_SELB] = 1时，边沿计数器的比较值设为5，并且每5个边沿触发一次中断，如图38所示。

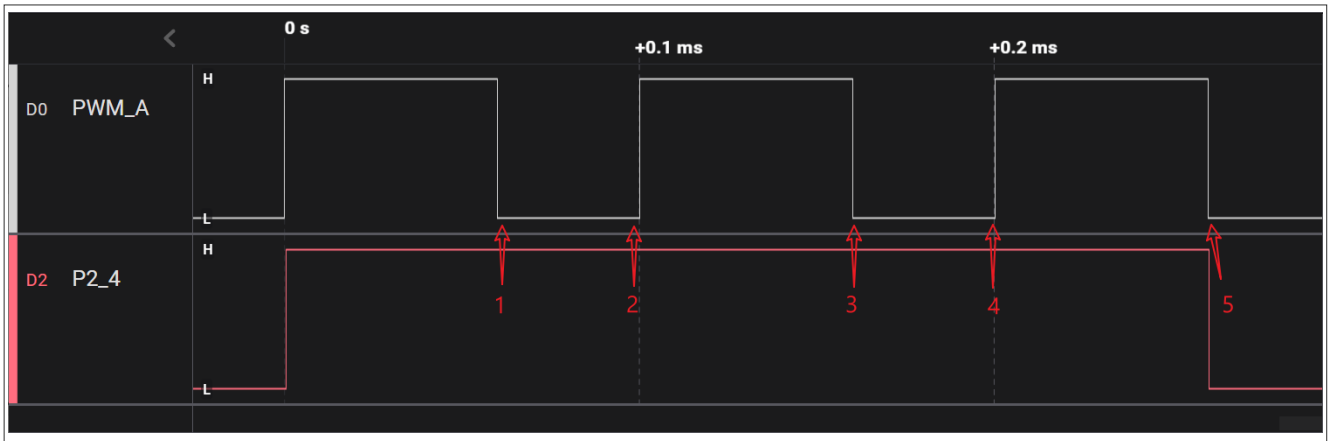

图38. E-Capture波形（INP_SELB = 1）

5 设计注意事项

为确保MCX N系列上的FlexPWM模块正确运行，请考虑以下事项：

- PWM输出引脚需要靠近引脚连接一个强外部下拉或上拉电阻（1kΩ到10kΩ）。这是为了确保在不确定条件下的安全状况。
- 要手动禁用PWM输出（无论占空比和时钟设置如何），可以采用以下两种方法：
  - 通过清除PWM_OUTEN的相应位来禁用PWM输出，这会导致PWM引脚输出三态。
  - 通过设置PWM_MASK的相应位来禁用PWM输出，然后触发一个FORCE_OUT事件，这会导致PWM引脚上在输出极性之前输出逻辑零。
- 请确保满足以下边界条件以正确生成 PWM 信号：INIT ≤ VAL2 (VAL4), VAL3 (VAL5) ≤ VAL1
- 当需要可变频率的PWM时，保持INIT不变，仅通过改变VAL1来改变频率。

6 结语

本应用笔记总结了MCX N系列上FlexPWM模块的结构和功能。它提供了示例代码，以便用户更好地理解这些功能的实现，使用户能够轻松地在自己的项目中正确使用该模块。最后，本文还分享了一些设计注意事项，以帮助用户有效地使用该模块。

7 参考资料

用于补充本文档的参考资料如下：

- 《使用MC56F82xx DSC中的eFlexPWM模块》（文档：AN4485）
- 《MCX Nx4x参考手册》（文档：MCXNX4XRM）

8 关于本文中源代码的说明

本文中所示的示例代码具有以下版权和BSD-3-Clause许可：

9 修订历史

表2汇总了本文档的修订内容。

表2. 修订历史

文档ID	发布日期	描述
AN14196 v.1.0	2024年5月2日	初始公开发布

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com.cn/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

目录

For more information, please visit: https://www.nxp.com.cn

Date of release: 2 May 2024
Document identifier: AN14196