AN12873

eflexPWM 模块 PWM 信号的有效更新

Rev. 0 — 08/2020 Application Note

目录

1 简介

脉冲宽度调制（Pulse width modulation, PWM）作为实时控制应用中的基本控制技术，广泛应用于开关电源、电机控制、音频放大器等各种场合。实时控制应用需要实时变化的 PWM 信号来实现所需的电压和电流，而 PWM 信号的更新机制会因应用场景和控制算法而异。

NXP 提供了各种带 PWM 模块的微控制器，根据其目标应用的不同，各种芯片里的 PWM 模块复杂度和灵活性也不相同。eFlexPWM 模块是 NXP 针对实时控制应用而设计的增强型 PWM 模块，它可以以最少的软件干预实现更灵活、更可靠的 PWM 信号。特殊的寄存器互锁机制可以有效防止意外的 PWM 更新，但这就对 PWM 有效更新提出了一些需求。

本应用笔记旨在说明 eFlexPWM 模块信号有效更新的正确用法。

2 eFlexPWM 模块 PWM 信号更新机制

一个 eFlexPWM 模块包含四个子模块，在每个子模块中，通过比较 PWM 计数器值与相应的 VALx 寄存器的值来控制 PWM 信号的上升沿和下降沿。换句话说，我们可以通过更新相关的 VALx 寄存器来更新 PWM 信号的频率和占空比。

典型配置中，PWM 计数器以 INIT 寄存器值为起点计数到 VAL1 寄存器的值，然后重新复位到 INIT 寄存器值开始计数，因此 INIT 和 VAL1 寄存器决定 PWM 信号的周期。当计数器值与 VAL2 寄存器值相等时，PWM 输出置高，当计数器值与 VAL3 寄存器值相等时，PWM 输出置低，VAL2 和 VAL3 寄存器决定了 PWM 信号的上升沿和下降沿。

2.1 寄存器重载逻辑

eFlexPWM 模块中，CTRL[LDFQ], CTRL[PRSC], INIT, VALx, FRACVALx 以及 PHASEDLYx 寄存器采用缓冲机制，他们的有效寄存器里的值控制当前的 PWM 波形，而缓冲寄存器里的值对硬件动作没有影响，仅用于安全地更新其有效寄存器的值。我们使用 reload 表示将缓冲寄存器里的值重载到相应的有效寄存器里。

图 1 所示为寄存器重载逻辑，用于确定何时将缓冲寄存器里的值重载到其对应的有效寄存中。

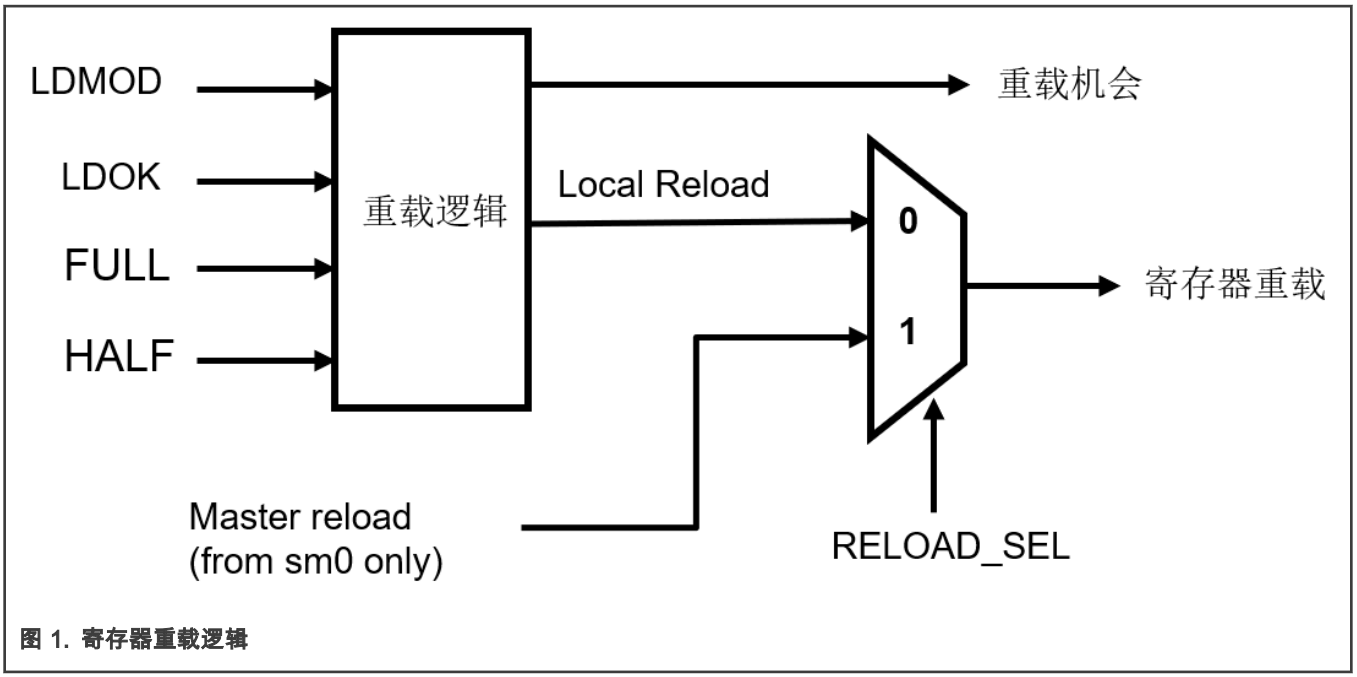

图 1. 寄存器重载逻辑

寄存器重载逻辑由以下寄存器位段控制：

- CTRL[LDMOD]
- CTRL[HALF]
- CTRL[FULL]
- CTRL[LDFQ]
- CTRL2[RELOAD_SEL]
- MCTRL[LDOK]

缓冲寄存器和对应的有效寄存器在内存中地址相同，读寄存器将会得到其缓冲寄存器里的值而不是当前 PWM 生成所使用的有效寄存器的值，写寄存器也是给缓冲寄存器赋值。寄存器重载有两种机制：

- 当 LDMOD 等于 0 时，延时重载，寄存器重载只发生在设定的重载点并且 LDOK 控制位等于 1 时。
- 当 LDMOD 等于 1 时，立即重载，缓冲寄存器里的值将在 LDOK 控制位置 1 的下一个时钟周期立即重载到其对应的有效寄存器。

LDOK 控制位在寄存器有效更新后会自动清除。寄存器重载点由 HALF, FULL 以及 LDFQ 控制位决定，当 HALF 控制位置 1 时，PWM 计数器值与 VAL0 寄存器值相等时会产生一个重载机会，当 FULL 控制位置 1 时，PWM 计数器值与 VAL1 寄存器值相等时产生重载机会。LDFQ 控制位决定重载信号的频率，也就是说多少个重载机会产生一个有效重载信号。当选择延时重载时，确保 HALF 和 FULL 控制位至少有一个置位才能将缓冲寄存器里的值重载到其对应有效寄存器里。当 HALF 和 FULL 控制位都置 1 时，一个 PWM 周期内将有两次重载机会。

如 图 1 所示，子模块 0 的重载信号可以连接到其他子模块作为 Master reload 信号，从而允许子模块 0 的重载逻辑可以同时控制其他子模块的寄存器重载。各个子模块也可以选择由本模块的 HALF 和 FULL 产生的“Local reload”信号作为最终的重载信号。

2.2 重载频率

通过配置 LDFQ 可以实现 n 个 PWM 周期重载一次。无论 LDOK 是否置位，LDFQ 都会在每次重载机会处生效。当 LDMOD = 0 选择延时重载时, 如果 FULL 控制位置位，重载机会在每个 PWM 周期结束时刻（也就是 PWM 计数器值等于 VAL1 寄存器值时），如果 HALF 控制位置位，重载机会在每个 PWM 周期的中间（也就是 PWM 计数器值等于 VAL0 寄存器值时，并不一定是中点）。如果同时设置了 HALF 和 FULL，则每个 PWM 周期有两次重载机会。

图 2 展示了不同重载频率下的寄存器重载，图中示例假设 HALF 和 FULL 控制位都置 1，也就是说每个 PWM 周期有两次重载机会。从图中可以看出 LDFQ 在每次重载机会时都生效，与 LDOK 状态无关，但是有效更新只发生在 LDOK 置位时的重载点。

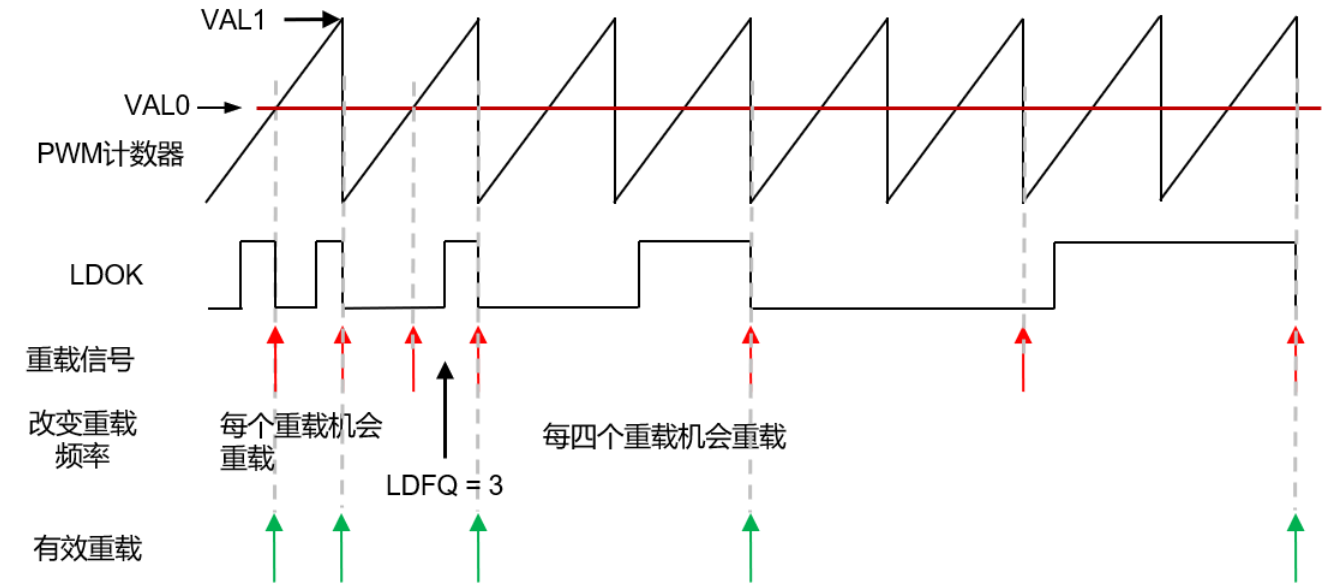

图 2. 不同重载频率下寄存器更新

2.3 寄存器初始化

启动 eFlexPWM 模块，也就是写 MCTRL[RUN] 控制位，对应的子模块将会自动产生一个 Local reload 信号。为了确保 eFlexPWM 模块在启动之后按照预期动作，请按以下步骤进行寄存器初始化：

1. 初始化除了 LDOK 和 MCTRL[RUN] 之外的其他所有寄存器；
2. LDOK 置位；
3. MCTRL[RUN] 置位。 这时 eFlexPWM 模块就会使用初始化的寄存器值来控制 PWM 波形。

注意

置位 MCTRL[RUN] 产生重载信号也就意味着置位重载标志位 STS[RF], 如果初始化时使能了重载中断（INTEN[RIE]=1）并且正确配置了中断控制器（INTC），启动 PWM 后将会立即产生一个重载中断。

3 不同应用中寄存器有效更新的实现

为了在这种重载机制下实现有效更新，得到所需的 PWM 波形，必须保证以下两点：

1. 缓冲寄存器在 LDOK=0 时赋值。
2. 重载信号在 LDOK=1 时产生，并且更新后的寄存器值在期望位置生效。

3.1 缓冲寄存器有效赋值

缓冲寄存器不能在 LDOK=1 时赋值，也就是说，当 LDOK=1 时，写缓冲寄存器将不会带来任何变化，缓冲寄存器依然保持原始值。因此，一般建议在写缓冲寄存器前通过置位 CLDOK 来清除 LDOK，然后在写入完成后再置位 LDOK。下面的示例代码展示了如何给子模块 0（SM0）和子模块 1（SM1）寄存器赋值。

```
PWMA_MCTRL |= 0x30; // clear LDOK of SM0&1 by setting associated CLDOK
PWMA_SM0VAL2 = 20;
PWMA_SM0VAL3 = 399;
PWMA_SM1VAL3 = 299;
PWMA_MCTRL |= 0x3; // set LDOK of SM0&1
```

无论是否有重载信号，缓冲寄存器的值在 LDOK=0 时都不会重载到其对应的有效寄存器，因此清除 LDOK 和置位 LDOK 之间的时间可以称为重载“禁行区”。在很多应用实际应用中，缓冲寄存器的赋值代码要比上面的示例复杂的多，特别是包含了如 if…else 之类的条件判断逻辑，为了缩短上面的重载“禁行区”，请尽可能简化缓冲寄存器赋值代码。

读寄存器得到的是其缓冲寄存器里的值，因此通过在线调试或者借助 FreeMASTER 工具可以方便地实时获取缓冲寄存器里的值，从而确定缓冲寄存器是否按照预期逻辑正确赋值。

3.2 重新加载的寄存器值不生效

eFlexPWM 模块提供了两种寄存器重载机制，延时重载和立即重载，重载机制的选择由 LDMOD 控制位决定。为了防止意外重载，大多数情况下选择延时重载（LDMOD=0），也就是说重载仅发生在预先配置好的重载点。然而，在延时重载机制下，如果控制时序和重载时间安排不当，PWM 信号则可能发生一些不期望的行为，例如，更新后的 PWM 波形与新重载的寄存器值不符。

3.2.1 新重载的寄存器值与重载点计数器值重合

PWM 信号的上升和下降沿发生在 PWM 计数器值等于 VAL2 和 VAL3 寄存器值时的 PWM 时钟上升沿，HALF 重载信号发生在 PWM 计数器值等于 VAL0 寄存器值时的 PWM 时钟上升沿，FULL 重载信号发生在 PWM 计数器值等于 VLA1 寄存器值时的 PWM 时钟上升沿。当新重载的寄存器值与重载点计数器值相等时，PWM 计数器在当前周期已经过了与新重载值相等的点，这意味着 PWM信号将错过这个周期的上升或下降沿。如 图 3 所示，假设 LDOK 在写缓冲寄存器前清除，在写入完成后置位。PWM 初始配置为：VAL0 = INIT = 0，VAL1 = 9，VAL2 = 8， VAL3 = 1，使能半周期重载（HALF 控制位置 1）。

- 在第一个 PWM 周期中间，软件将 VAL3 缓冲寄存器的值重新赋值为 0，但是本周期 PWM 信号依然按照原始的 VAL2 和 VAL3 寄存器值产生，在 PWM 计数器值等于 8 时置高，在 PWM 计数器值等于 1 时拉低。
- 在第二个 PWM 周期起始点，PWM 计数器值在 PWM 时钟上升沿复位到 0。由于 VAL0 也等于 0，计数器复位同时产生一个重载信号，VAL3 有效寄存器的值在这个重载点更新为其缓冲寄存器里的值 0。尽管 PWM 计数器值与 VAL0 匹配迅速产生了重载信号，然后将 VAL3 重载为 0，但是依然错过了 PWM 计数值等于 0 的时钟上升沿，在 PWM 计数器值变为 0 的 PWM 时钟上升沿，VAL3 有效寄存器值还是原始值 1，所以 PWM 信号依然为高电平。本周期内 PWM 计数器值将不会再与新重载的 VAL3 值匹配，所以 PWM 信号始终为高电平。
- 在第三个 PWM 周期起点，当 PWM 计数器值在 PWM 时钟上升沿复位到 0 时，PWM 计数器值与上个周期重载的 VAL3 寄存器值匹配，PWM 输出拉低。

为了防止出现上述问题，一般建议软件中增加条件判断算法，当判断到计算得到的新的 VALx 寄存器值与重载点计数器值相等时，将此寄存器值往后移一个时钟周期，确保重载完成后下一个时钟周期能够立即动作，下面的示例代码可以用于解决上述例子的问题。

```
// avoid rising edge to coincide with reload point
if(PWMA_SM0VAL2 == 0) PWMA_SM0VAL2 = 1;
// avoid falling edge to coincide with reload point
if(PWMA_SM0VAL3 == 0) PWMA_SM0VAL3 = 1;
```

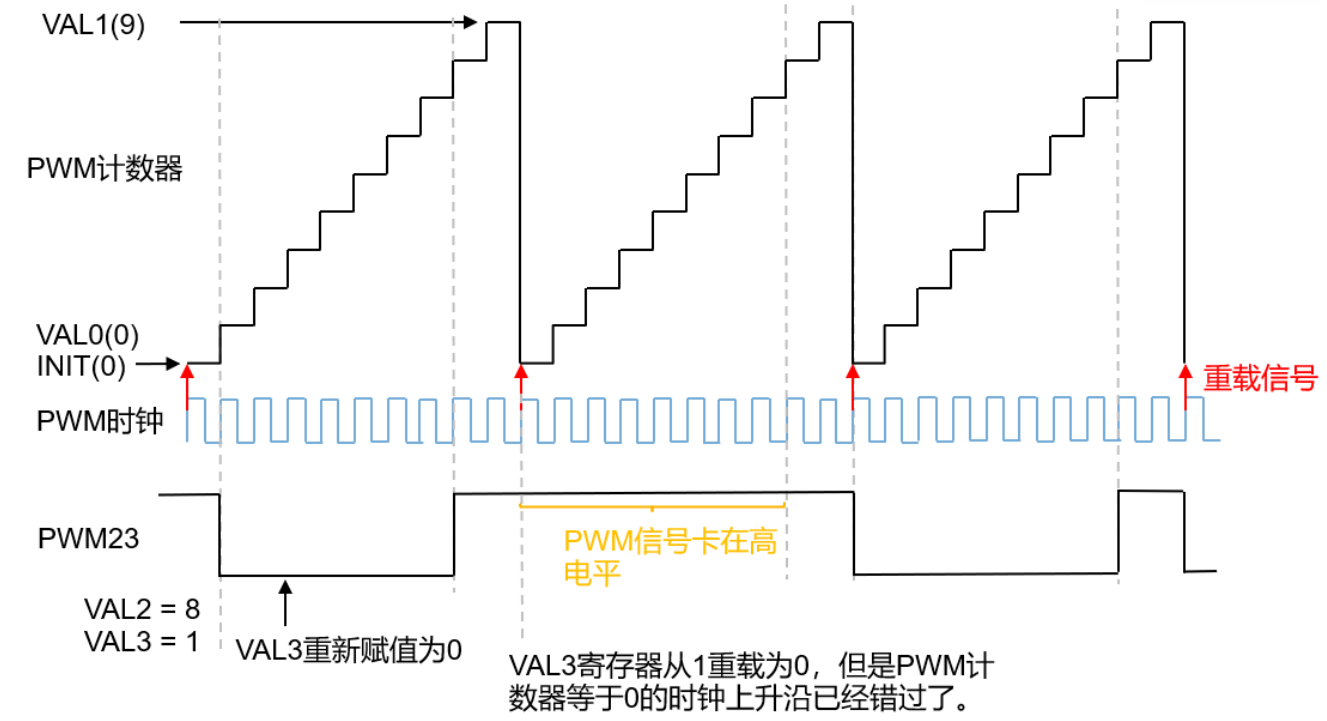

图 3. 新重载值与重载点计数器值重合示例

3.2.2 VAL 寄存器值越位

针对 PWM 移相需求，通常有两种方法：

1. 计数器移相，各子模块使用相同的 VALx 寄存器值；
2. 计数器同步，VALx 寄存器值增加偏置。

不管使用哪种方法，为了节省 CPU 资源并保证输出占空比相同，不同子模块一般在同一时刻对所有寄存器更新，而此时有的子模块的一个 PWM 脉冲的上升沿和下降沿相对重载点的位置可能是变化的。换句话说，以完整 PWM 脉冲而不是 PWM 计数器界定 PWM 周期，有的子模块重载点在周期中间，其 PWM 脉冲的上升沿和下降沿可能在重载点前面也可能在重载点后面，这就有可能出现寄存器值越位问题。所谓寄存器值越位是指对于一个 PWM 脉冲而言，其上升沿或下降沿位置从滞后于重载点变为在重载点之前，如 图 4 所示。

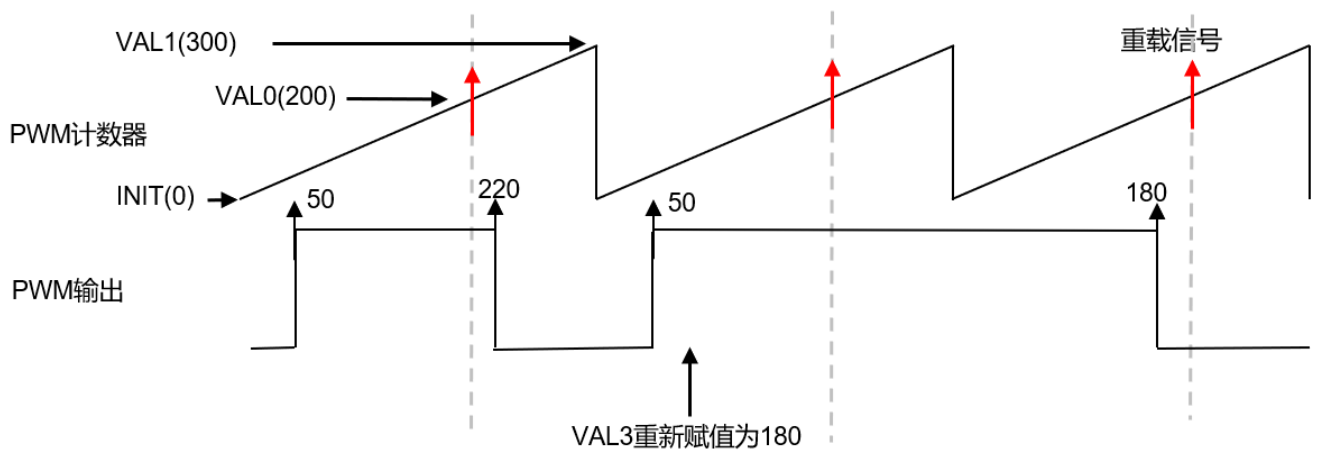

图 4. 寄存器值越位示例

PWM 模块初始配置为：INIT = 0, VAL1 = 300, VAL0 = 200, VAL2 = 50, VAL3 = 220，使能半周期重载（HALF 控制位置 1）。

- 在第一个 PWM 周期，VAL3 寄存器的值 220 大于重载点计数器值 200（VAL0），PWM 脉冲的下降沿滞后于重载点。
- 在第二个周期重载点之前，VAL3 缓冲寄存器值重新赋值为 180，VAL3 缓冲寄存器的值在 PWM 计数器值到达 200 时载入其有效寄存器，也就是说 PWM 脉冲的下降沿从滞后于重载点变为在重载点之前。在 PWM 计数器值小于 200 时，VAL3 有效寄存器值为 220，此时不会发生比较匹配，而在 PWM 计数器值大于 200 后，VAL3 有效寄存器值重载为了 180，小于 200，本 PWM 周期内也不会发生比较匹配了，所以 PWM 输出一直保持高电平。

图 5 是一个典型的移相控制示例，PWM 频率固定为 125kHZ（PWM 时钟为 100MHz，INIT = 0, VAL1 = 799），SM0 和 SM1通过 表 1 所示的配置实现 180°移相。计数器同步，SM1 的 VAL2 和 VAL3 相对 SM0 增加半周期偏置。SM0 的 FULL 控制位置 1，当计数器值与 VAL1 寄存器相等时产生重载信号，SM1 通过选择 Master reload 也在相同位置进行寄存器更新，因而 SM1的 PWM 脉冲下降沿相对重载点位置是不固定的。

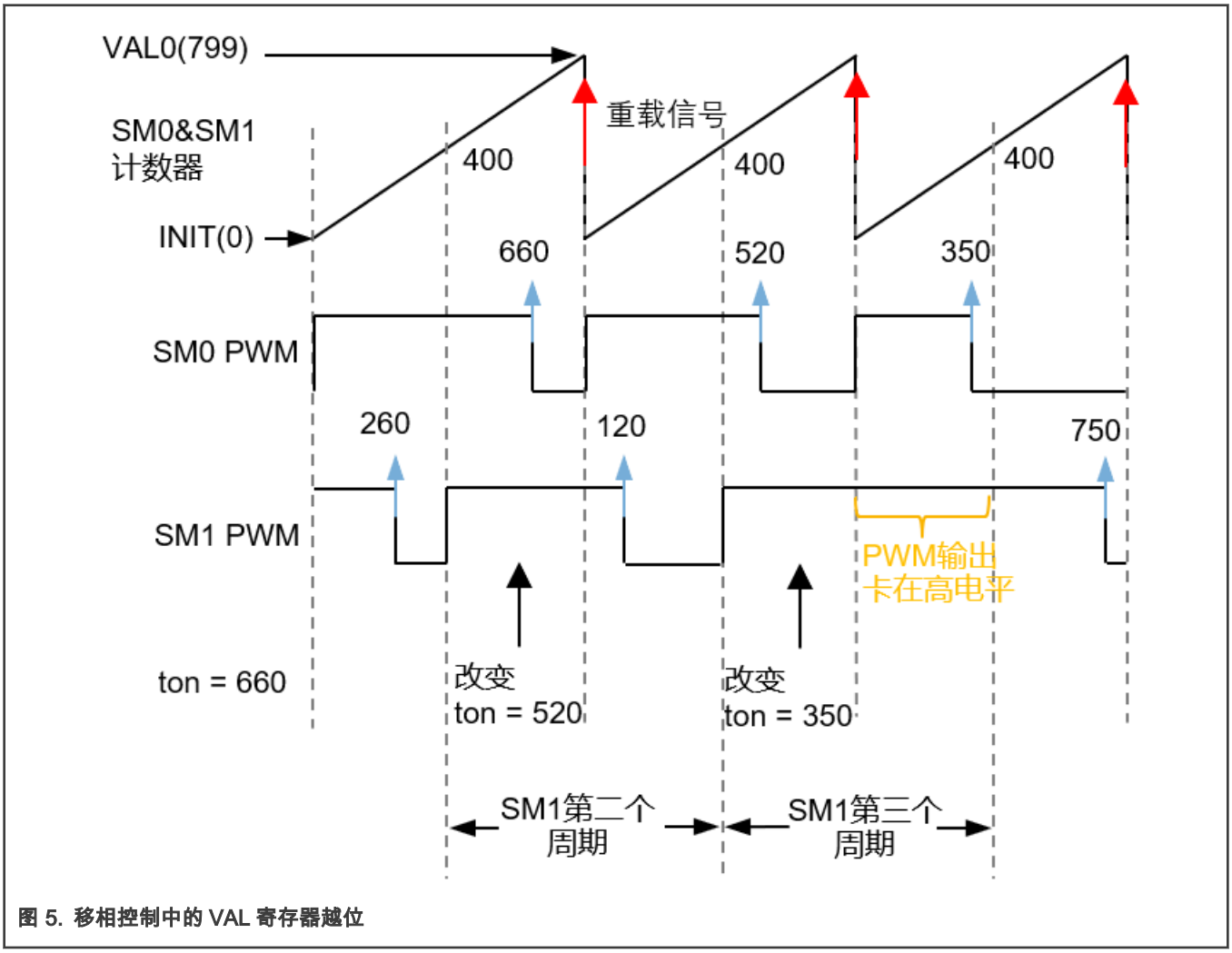

图 5. 移相控制中的 VAL 寄存器越位

图 5 中 ton 是 PWM 信号的导通时间，SM1 的 PWM 输出与 SM0 相移 180°，导通时间相等。

表 1. SM0 和 SM1 的同步配置

配置		SM0	SM1
Reload 配置	重载源	Local reload 信号	Master reload 信号
	HALF	禁止	禁止

Table continues on the next page...

表 1. SM0 和 SM1 的同步配置 (continued)

配置		SM0	SM1
	FULL	使能	禁止
初始化源		Local sync 信号	Master sync 信号
VAL2		固定为 0	固定为 400
VAL3		根据 ton 改变	根据 ton 改变

- 开始时 ton 等于 660，SM0 的 VAL2=0, VAL3=660; SM1 的 VAL2=400, VAL3=260，SM1 的 VAL3 寄存器值在本周期重载点的右边，也就是说 PWM 脉冲下降沿滞后于重载点。
- 在 SM1 第二个 PWM 周期重载点之前，软件计算 ton 值变为 520，SM1 的 VAL3 缓冲寄存器重新赋值为 120。当 PWM 计数器值等于 VAL1 值 799 时，SM1 的 VAL3 有效寄存器重载为 120，新的 PWM 脉冲下降沿依然滞后于重载点，PWM 波形按照预期产生。
- 在 SM1 第三个 PWM 周期重载点之前，软件计算 ton 值变为 350，SM1 的 VAL3 缓冲寄存器重新赋值为 750。在重载点之前，SM1 的 VAL3 有效寄存器值依然为 120，小于 PWM 计数器值不会产生比较匹配。当 PWM 计数器值等于 VAL1 值 799 时，SM1 的 VAL3 有效寄存器重载为 750，而此时计数器值已经超过了 750，PWM 脉冲在本周期保持高电平。由于 VAL3 寄存器值从滞后于重载点更新为在重载点之前，PWM 波形不能按照预期产生，PWM 脉冲错过本周期的下降沿，卡在高电平直到下个 PWM 周期 PWM 计数器值与 VAL3 寄存器值比较匹配。

为了解决寄存器越位带来的问题，可以使用 eFlexPWM 模块的 FORCE init 功能，当软件判断到寄存器值发生越位时，使能 FORCE init 功能将 PWM 输出在重载点强制拉到期望的电平，尽量减小错误电平时间。针对图 5 的例子，配置 SM1CTRL2[FORCE_SEL] = 101b 选择 SM0 的初始化信号为 SM1 的“FORCE_OUT”源，并设置 SM1 的 PWM 输出 init 状态为低电平。如下面示例代码所示，如果计算得到的下个周期 PWM 导通时间小于半个周期，也就是 PWM 脉冲需要在重载点之前变为低电平，配置 SM1CTRL2[FRCEN] = 1 使能 SM1 的 FORCE init 功能，SM0 的初始化信号将强制 SM1 的 PWM 输出到 init 低电平。如果计算得到的下个周期 PWM 导通时间大于等于半个周期，配置 SM1CTRL2[FRCEN] = 0 禁止 SM1 的 FORCE init 功能， SM0 的初始化信号对 PWM 输出没有影响，PWM 波形将完全按照 VALx 值与 PWM 计数器匹配生成。这里 ton 等于 399 时，VAL3 寄存器值将于重载点重合，这时使能 FORCE init 功能也将保证正确的 PWM 输出。图 6 为增加了 FORCE init 功能后的 PWM 输出。

```
if(ton<400)
{
   PWMA_SM1VAL3 = 400 + ton;
   PWMA_SM1CTRL2 |= 0x80; //enable force initialization to force PWM output low
}
else
{
PWMA_SM1VAL3 = ton - 400;
   PWMA_SM1CTRL2 &= ~0x80; //disable force initialization
}
```

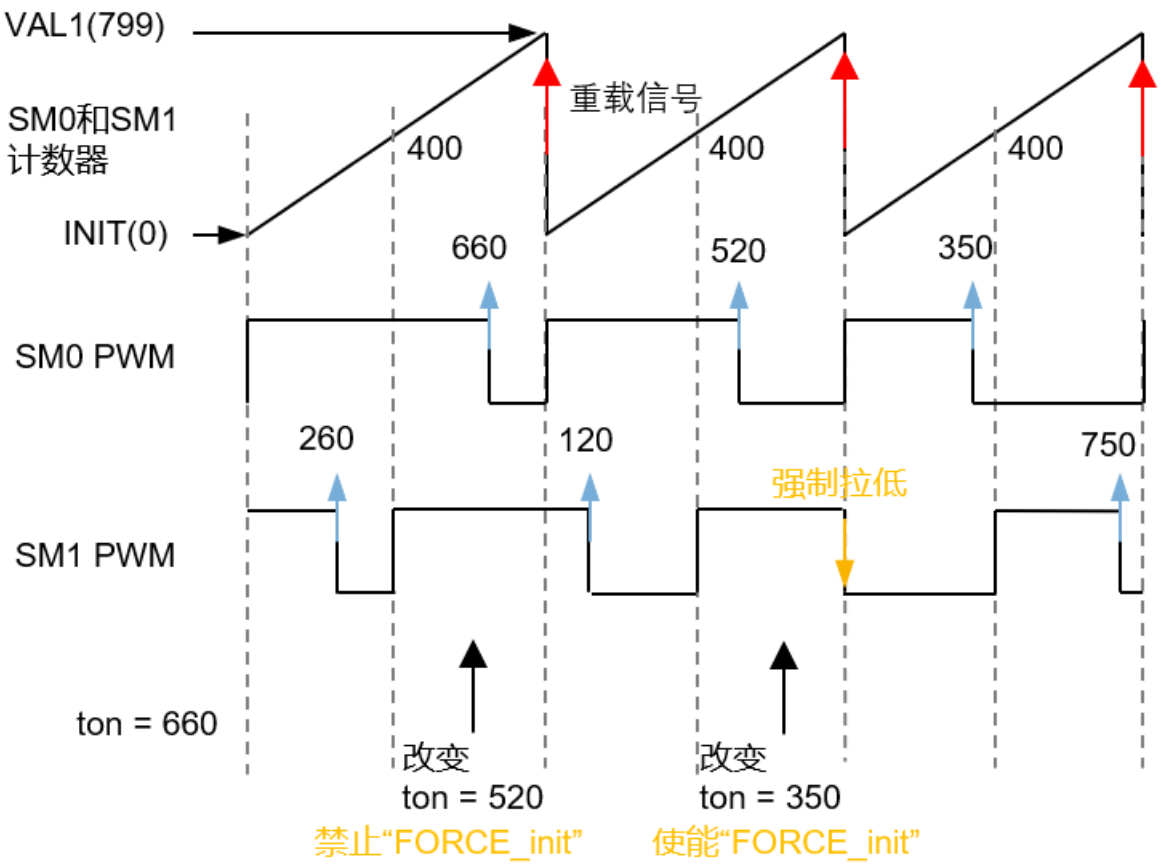

图 6. 增加 FORCE_init 功能后 PWM 波形

3.3 重载点 LDOK 清零

任何重载机制下，LDOK=1 都是有效重载的必要条件。如果重载信号发生在 LDOK 清零的重载“禁行区”，实际将不会发生寄存器重载。如 图 7 所示，配置 CTRL[LDMOD]=0 选择延时重载，在第一个 PWM 周期结束时发生有效重载，而在下一个 PWM 周期结束则不会发生寄存器重载。

基于不同的 PWM 配置，有不同的方法来识别和解决重载点 LDOK 清零的问题。

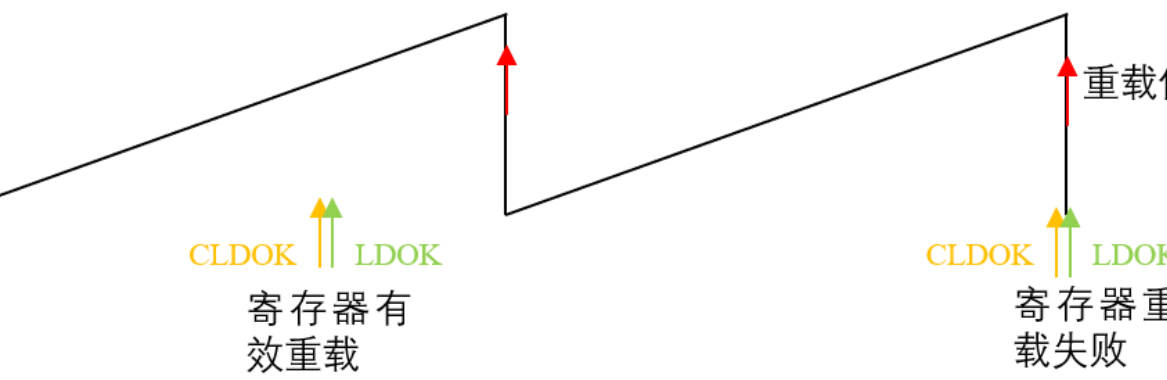

图 7. LDOK 决定重载状态示例

3.3.1 定频 PWM

数字控制的实时性无一例外采用中断（ISR）来保证。当应用中 PWM 频率固定时，我们可以很方便地根据实际测得的 ISR 执行时间判断重载点之前是否有足够的时间用于软件计算并且 LODK 置位。然而，ISR 执行时间会根据不同的条件分支以及流水线影响而变化，并且一些由于流水影响造成的时间差别可能只有几十 ns，我们很难使用示波器精确读取最长 ISR 执行时间。所以我们可能会遇到如 图 8 所示的情况，重载点之前留给 ISR 执行的时间与示波器测量得到的 ISR 最长执行时间非常接近，而实际会出现寄存器有时候重载成功，有时候却重载失败，重载失败可能就是重载信号出现在了 LDOK 清零的区间。

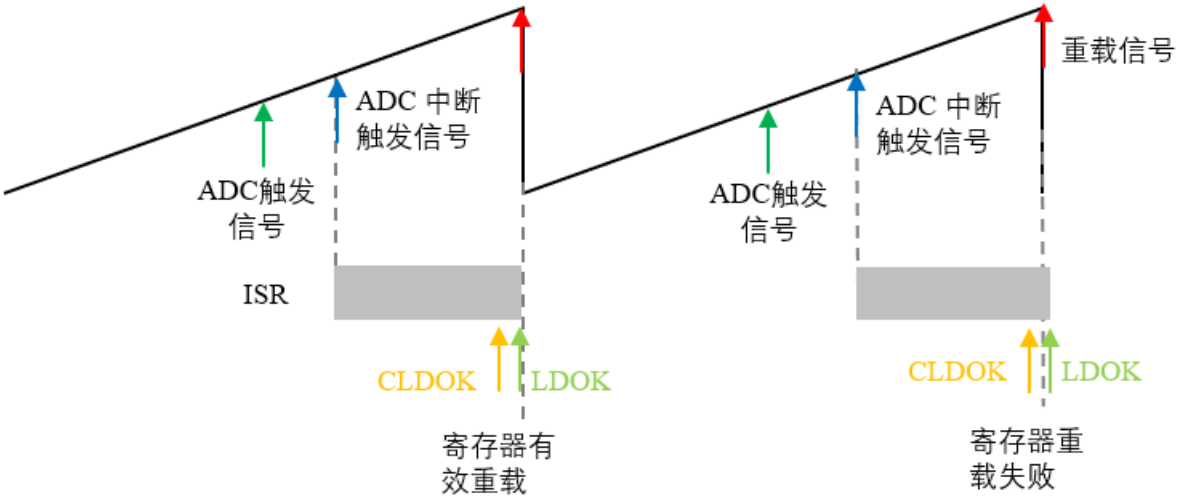

图 8. ISR 执行时间变化导致重载失败

为了避免上面提到问题，一种方法是降低 PWM 频率，从而增加重载点之前留给 ISR 的执行时间。而实际中，PWM 频率一般是不可以移随便改动的，可以通过改变中断触发源或调整 ISR 中的代码执行顺序来保证重载点之前所有缓冲寄存器赋值 ，并且 LDOK置位。例如，图 8 中 ADC 触发信号位置不可以变，可以用一个超前于 ADC 中断的 PWM 比较中断代替图中的 ADC 中断，并且在将不需要 ADC 采样结果的代码放在中断开始的位置执行，如 图 9 所示。

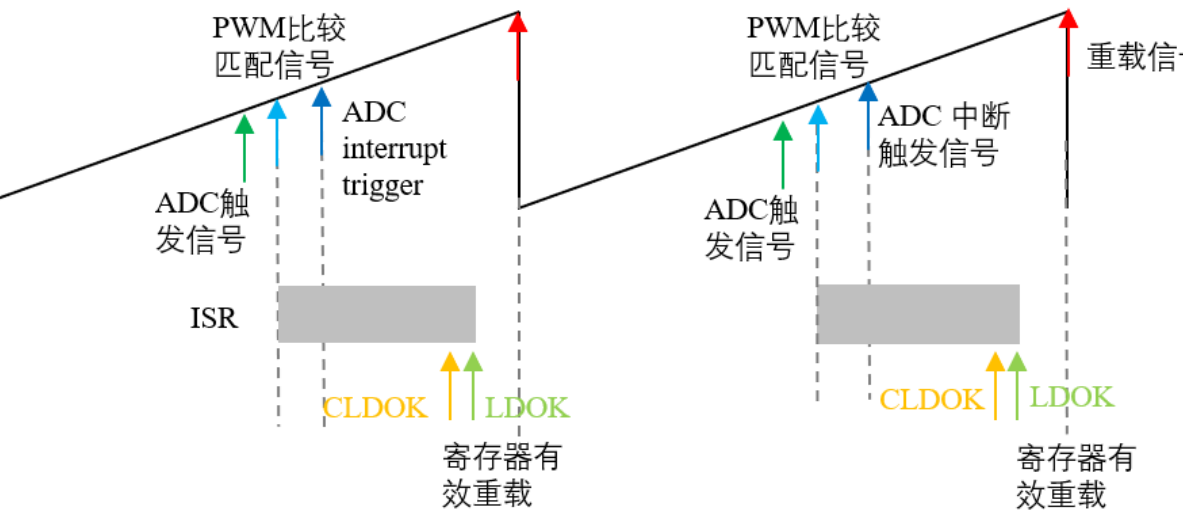

图 9. 使用其他中断源增加 ISR 可用时间

3.3.2 变频 PWM

当采用变频调制（PFM）时，PWM 频率变化，则从中断触发到重载点之间的时间是实时变化的。如果依然每个周期产生一个重载信号，尽管 ISR 执行的最大时间是固定的，重载信号还是有可能出现在 LDOK 清零的区间，导致重载失败。PFM 控制应用中，如果控制时序和 PWM 之间不加入特定的同步或判断机制，重载信号可能出现在任意位置，从而导致冗余计算和重载失败：

- 重载信号在 LDOK 清零之前：寄存器重载上个 ISR 计算的值
- 重载信号在 LDOK 置位之后：寄存器重载最近 ISR 计算的值
- 重载信号在 LDOK 清零与 LODK 置位之间：不发生寄存器重载

3.3.2.1 PWM 频率由软件决定

如果 PWM 频率由软件决定，也就是说下一个 PWM 周期长度和 ISR 执行时间都是已知的，则可以根据计算的下个 PWM 周期 以及 ISR 执行时间调整 PWM 重载频率，确保在重载信号之前 LDOK 置位。重载频率的调整可以通过 CRTL[LDFQ] 控制位实现。如图 10 所示为 PFM 应用中调整 PWM 更新频率示例，假设 ISR 最长执行时间为 8 us，ISR 由重载完成触发。

- 第一个控制周期之前，计算得到的下个 PWM 周期长度大于 8 us，也就是说 ISR 可以在一个 PWM 周期内完成，配置 CTRL[LDFQ] = 0, 每个 PWM 周期重载一次。
- 第一个控制周期中计算得到下个 PWM 周期长度小于 8 us，但是大于 4 us，配置 CTRL[LDFQ] = 1，每两个 PWM 周期重载一次。由于 CTRL[LDFQ] 控制位也采用了缓冲机制，所以新的重载频率要在当前重载点之后才生效。
- 第二个控制周期中计算得到下个 PWM 周期长度小于 4 us，但是大于 2.66 us，配置 CTRL[LDFQ] = 2，每三个 PWM 周期重载一次。

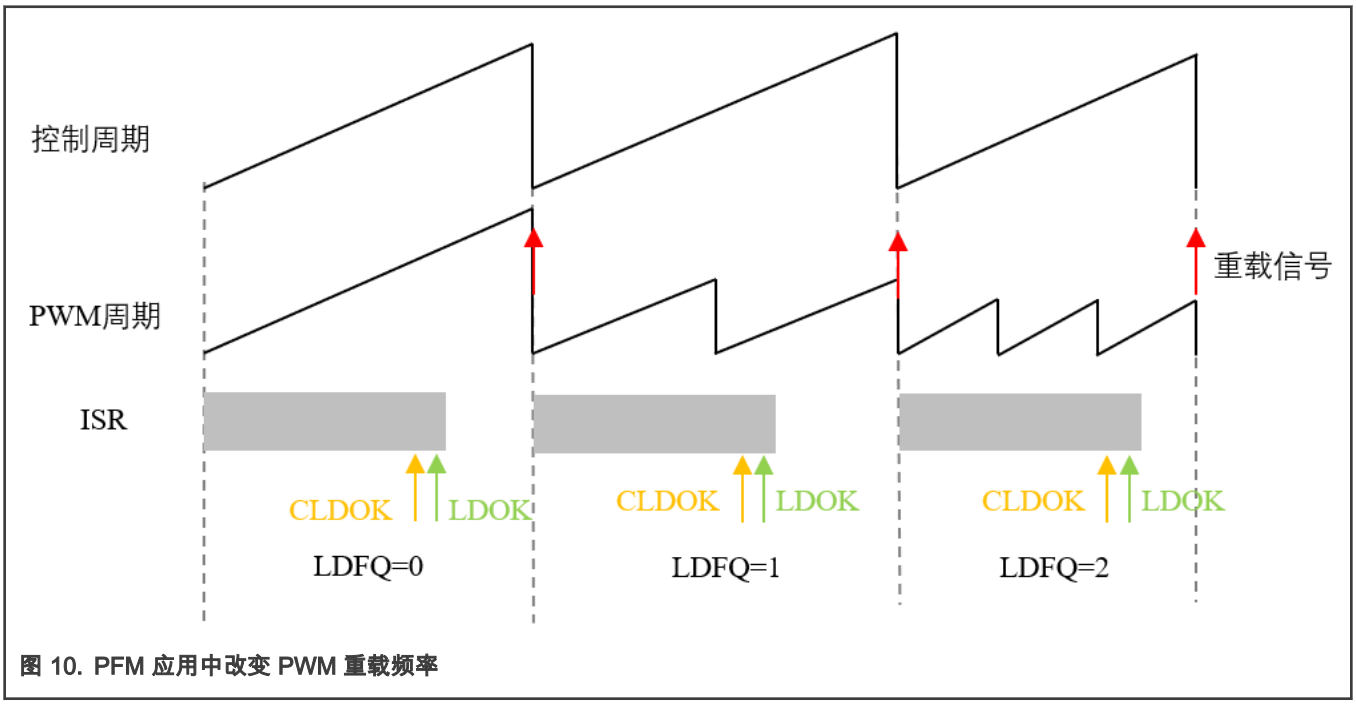

图 10. PFM 应用中改变 PWM 重载频率

4 结语

本应用笔记描述了 eFlexPWM 模块的缓冲寄存器重载机制，分析了不同场景下重载后的寄存器没有立即生效的原因，并提供了解决方案。另外，针对 PWM 定频与变频应用，提供了避免重载信号发生在重载“禁行区”的解决方案。在延时重载机制下(CTRL[LDMOD]=0)，要实现寄存器有效重载必须保证以下两点：在 LDOK=0 时写缓冲寄存器，重载信号发生在 LDOK=1 的位置。

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 08/2020

Document identifier: AN12873

arm