life.augmented

AN4539
应用笔记
HRTIM 指南

前言

在 STM32 微控制器中， STM32F334xx 产品的目标市场是需要高度精确计时数字信号、尤其是数字功率转换应用的细分市场。包括：

- 数字电源；
- 照明；
- 不间断电源；
- 太阳能逆变器；
- 无线充电器。

STM32F334xx 微控制器具有高分辨率定时器 （HRTIM）外设，可产生多达 10 个信号，能够处理用于控制、同步或保护的各种不同输入信号。其模块化架构允许对大部分转换拓扑和多并联转换器进行处理，并可在运行中重新配置它们。

通过 STM32F334 参考手册初步了解时，此外设可能看起来比较复杂，这主要是由于它有大量的控制寄存器组。为了补充这份详尽的说明，我们另外提供了一个文档，其中包括快速上手说明以及示例汇总。

在其第一章中，本指南旨在表明 HRTIM 编程很简单。首先说明环境 （就像是有菜谱的同时也需要有厨房）设置，接着给出了若干简单示例，通过实践帮助理解。这些基本案例用来逐步介绍定时器功能，并提供编程指导。不熟悉 HRTIM 的读者应该仔细阅读本章。

第二部分是转换器集合，可在开始新设计时使用，从中选取现成的代码示例，或者从中得到灵感和编程技巧来处理本文档中未描述的拓扑。但是需要注意，本指南不包括转换器设计本身 （控制技术和元件设计），这些内容在专门的应用笔记中有描述。

如有必要，每个示例都提供了简要的转换器说明 （拓扑和到 MCU 的连接）、控制波形和代码段。这些代码片段 （等效代码基于 STM32 HAL 库实现）可从 www.st.com 下载。

表 1. 适用产品

类型	料号和产品类别
微控制器	STM32F334xx

目录

表格索引

图片索引

1 完成环境的准备

本章中，我们会确保在开始前所有必需的要素均准备就绪，因此可以仅关注 HRTIM 编程。

下列文件作为参考：

- STM32F334x4/x6/x8 数据手册
- STM32F334x4/x6/x8 勘误表
- RM0364 参考手册 STM32F334xx 高级 ARM® 为基础的的 32 位 MCU
- UM1733：STM32F334 探索套件入门
- UM1735：STM32F3 系列的探索套件 - 采用 STM32F334C8 MCU
- UM1736：STM32F334 探索软件开发工具入门
- AN4885：使用 STM32F3348 Discovery 进行高亮度 LED 调光

预先阅读 RM0364 中的 HRTIM 章节会有帮助。

1.1 必备条件

在享受 HRTIM 的优势之前，我们列出其前提条件。希望读者具有基本的 C 编程技巧，关于 MCU 和开发环境的少量经验，以及关于开关模式电源的理论背景。控制策略和元件尺寸标注细节不在本应用笔记范围内，它们可在大量文献中获取。

为简单起见，本指南仅考虑逻辑信号或直接由 MCU 处理的模拟电压，这样就是与电平无关的。不过有些参考文献涉及外部元件接口和电源切换影响 （当定时器或 MCU 具有处理它们的功能时）。

最后，需要提醒的是，如果 STM32F334 和 HRTIM 用于具有危险电压的应用中，则应由熟练的技术人员来操作功率应用，以避免电击、烧伤甚或死亡的风险。

1.2 硬件设置

STM32F334 探索板是价格非常实惠的工具，是开始 （以及继续）用 HRTIM 做实验的最佳选择 （订购码：STM32F3348-DISCO）。它包含了编程接口，芯片编程和调试所需的附加材料只是 USB 连接线。所有 I/O 均可在 2.54 mm 间隔的引脚上使用，因此也可连接到穿孔板 / 条状板 / 试验电路板。套件还有两个功率转换器：一个用于 LED 驱动的反向降压转换器和一个低电压降压 / 升压转换器，均具有独立的输入和输出。

示波器是必备的，最后它与逻辑分析仪一起，用于配置对超过 4 个通道的监测。为了显示出细微的高分辨率步长，示波器的采样率必须至少超过 1GS/s，具有交错采样选项，可将时间精度提高超过 217ps 定时器分辨率。

在早期调试阶段，一个或几个函数发生器可帮助仿真来自功率转换器的反馈 （逻辑脉冲或模拟信号）。该发生器必须有触发输入，用于一些特殊用途。如果缺少，也可利用空闲的定时单元，由 HRTIM 本身来仿真反馈信号，这需要多编写一些代码 （或重复利用软件示例）。

1.3 工具设置

必须安装一个编译器 （所有示例均适用 32K）以及支持 ST-LINK-V2 调试接口的 IDE。

下面给出的代码片段与编译器无关：它们将被简单地复制到面向各种工具链的通用 HRTIM 工程模板中。

对于下面的工具链，软件源随工作空间给出：

- IAR (EWARM 7.10.3)；
- KEIL® (MDK-ARM 4.7)。

1.4 STM32F334 和 HRTIM 设置

1.4.1 系统时钟初始化

为了实现高分辨率， HRTIM 需要由 PLL 高频输出直接馈送。有两种选择可供使用：

- 基于晶体的高速外部（HSE）振荡器，由 PLL 倍频后可提供 144MHz 的频率。这种情况下，此高分辨率为 217ps （144MHz 时钟周期的 1/32）；
- 高速内部（HSI）振荡器，能够提供 128MHz 的频率（8MHz 由 PLL 倍频 16 倍）。这种情况下，高分辨率步长为 244ps （128MHz 时钟周期的 1/32）。该选项适用于有限的温度范围，相关情况参见 STM32F334 数据手册。

在 HAL 库初始化 （HAL_Init）后，立即在主程序中使用 SystemClock_Config() 函数，完成时钟初始化。

CPU 时钟也来源于 PLL （除以 2 之后），因此它可达到 PLL 输出频率的一半 （使用 HSE 时为 72MH，使用 HSI 时为 64MHz）。它还可被降低，以减少 MCU 功耗，同时保持高分辨率功能。

可最后在主程序中执行 SystemCoreClockUpdate 函数来验证 CPU 工作频率：该频率在 SystemCoreClock 变量中更新。

1.4.2 HRTIM 初始化

本章逐步地详细介绍如何初始化 HRTIM，包括各个函数调用。实际上，这在 HAL_HRTIM_Init 和 HAL_HRTIM_MspInit 程序中完成。

HRTIM 时钟初始化

当 MCU 上电并开始运行， HRTIM 在编程之前就必须进行时钟控制。这利用复位和时钟控制器 （RCC）来实现，包括 2 个步骤：

1. 为 RCC_CFGR3 寄存器中的 HRTIM 选择高速 PLL 输出：

```
__HAL_RCC_HRTIM1_CONFIG(RCC_HRTIM1CLK_PLLCLK);
```

2. 为在 APB2 总线上映射的寄存器进行时钟使能。

```
__HRTIM1_CLK_ENABLE();
```

1.4.3 HRTIM DLL 初始化

HRTIM 的延迟锁相环 （DLL）可提供细粒度计时，将高频率 （144 或 128MHz）时钟周期分为 32 个均匀分布的间隔。

在使用高分辨率前，此 DLL 必须至少校准一次。如果电压或温度条件发生改变，在 HRTIM 操作过程中可透明地重新进行该校准。也可由硬件使能周期校准。

下面的代码段显示如何完成校准。当 DLLRDY 标志被置位后，高分辨率可用。

```
/* DLL 校准：使能了周期校准，周期设置为 14µs */
 HRTIM1->sCommonRegs.DLLCR = HRTIM_CALIBRATIONRATE_14| HRTIM_DLLCR_CALEN;
 /* 检查 DLL 校准完成的标志位 */
 while(HRTIM1->sCommonRegs.ISR & HRTIM_IT_DLLRDY == RESET);
```

建议使能周期校准，默认情况下采用最小校准周期 （设为 14µs）。

注： *如果 DLL 未锁定 （通常是由于 HSE 振荡器未正常配置），下面的代码将引起执行延迟。HAL 库包含一个函数来实现校准，它具有超时验证，如有必要可重定向至差错处理程序。基于 HAL 的软件示例中使用了此函数。*

1.4.4 HRTIM I/O 初始化

HRTIM 输入和输出映射到标准 I/O 端口，必须像其他 I/O 外设一样进行编程。 HRTIM 端口通道影射为：

- AF13 通道 （面向 HRTIM I/O 端口，位于端口 A 和 B）；
- AF3 通道 （面向 HRTIM I/O 端口，位于端口 C）；

HRTIM I/O 初始化必须在两个阶段中完成。在 HRTIM 寄存器之前，首先在 HAL_HRTIM_MspInit 函数中初始化 HRTIM 输入。

HRTIM 输出必须在 HRTIM 控制寄存器编程 （示例中它在 GPIO_HRTIM_outputs_Config 函数中完成）后且当计数器使能时进行初始化。这是为了保证来自 GPIO 电路的控制信号传输到 HRTIM 定时器之前，在 HRTIM 中能够正确定义输出状态。

1.4.5 其他外设初始化

HRTIM 与多种 MCU 外设交互作用，如下所列。进行 IHRTIM 操作时不强制要求对它们全部进行初始化。下列外设的初始化代码在后面所述的一些示例中提供。

嵌套向量中断控制器 （Nested Vectored Interrupt Controller， NVIC）

HRTIM 中断请求分组为 7 种中断向量。所有错误均分组到一个特殊向量中，可设置为非常高的优先级。

与 HRTIM 相关的 NVIC 部分在 HAL_HRTIM_MspInit 函数中进行编程。

DMA 控制器

大部分中断请求可用作 DMA 请求，分组到 6 个 DMA 通道 （每个定时单元一个，包括主定时器）。

当启动定时器时，基于 DMA 的 HRTIM 操作使能，利用专门的开始 / 停止函数如 *HAL_HRTIM_WaveformCounterStart_DMA*。

更多详细信息，请参见*第 2 节：HRTIM 工作原理基础*。

比较器

3 个内置比较器可用来调节模拟信号：它们必须在输出到达 HRTIM 前进行初始化。

初始化包括模拟输入编程、时钟使能和极性。

运算放大器

内置运算放大器能够放大进入 ADC 或比较器的低电压信号，也可直接充当比较器 （相比普通比较器，它速率较低）。它必须与比较器一样进行初始化。

ADC 转换器

HRTIM 能够触发两个 ADC 转换器中的任意一个。应初始化它们以接收外部触发器信号 （在其常规和 / 或注入转换序列上）。

ADC 的另一个可能用途在于使用模拟看门狗在 HRTIM 上触发外部事件 （用于输出置位 / 复位或计数器复位）。

DAC 转换器

DAC 转换器通常用于定义比较器阈值。它们可利用 HRTIM 的 DAC 触发器，与 HRTIM 操作同步更新。

通用定时器

HRTIM 也可连接到其他片上定时器，有以下用途：

- 作为外部事件；
- 作为突发模式触发器或时钟；
- 用于 HRTIM 寄存器更新触发。

1.4.6 HRTIM 功能检查

当所有初始化完成，可验证 HRTIM 能够利用下面的简单代码来运行。此示例代码（HRTIM_BasicPWM 示例）使能了 HRTIM TD1 输出并通过软件对其进行切换。

```
/* 使用 PLLx2 时钟来实现 HRTIM */
__HAL_RCC_HRTIM1_CONFIG(RCC_HRTIM1CLK_PLLCLK);
```

```
/* 使能 HRTIM 时钟 */
__HRTIM1_CLK_ENABLE();
/* DLL 校准：使能了周期校准，周期设置为 14µs */
HRTIM1->sCommonRegs.DLLCR = HRTIM_CALIBRATIONRATE_14| HRTIM_DLLCR_CALEN;
/* 检查 DLL 校准完成的标志位 */
while(HRTIM1->sCommonRegs.ISR & HRTIM_IT_DLLRDY == RESET);

HRTIM1->sCommonRegs.OENR = HRTIM_OENR_TD1OEN; /* 使能 TD1 输出 */
GPIO_HRTIM_outputs_Config();  /* 初始化 HRTIM 输出 */

while(1)
{
/* 通过软件来置位和复位 TD1 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx1R = HRTIM_SET1R_SST;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx1R = HRTIM_RST1R_SRT;
}
```

此处复制的代码片段可从 HRTIM_Snippets 和 HRTIM_BasicPWM 示例中获取。两种情况下都须通过 *#define HRTIM_CHECK* 语句来选择示例。

对于本文档其余部分，时钟和 DLL 初始化部分将不再重复，而代之以对 *HRTIM_Minimal_Config()* 函数的调用。

2 HRTIM 工作原理基础

HRTIM 因功能数量及其模块结构而导致明显复杂，它主要由 6 个 16 位自动重载递增计数器组成，每个计数器具有 4 个比较寄存器。

周期和比较程序设计

该高分辨率程序是完全透明的，当使用 HSE 振荡器时，类似于 4.6 GHz 时钟控制的定时器（144MHz x 32）。时间 （周期和比较）可以高精度地直接写入一个特殊的 16 位寄存器。利用公式可简单地为计数周期编程：

$$PER = \frac{T_{counting}}{T_{High-res}}$$

例如，通过设置周期寄存器为 10µs / 217ps = 46082d 可得到 10µs 的时间周期，并按以下编程：

```
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].PERxR = 0x0000B400;
```

如果结果溢出 16 位范围，分辨率按照 217ps 的倍数进行调整，这样能够获得 16 位范围内的周期值。

置位 / 复位纵横开关

每个定时单元通过置位 / 复位纵横开关来控制 2 个输出。相比于普通的冻结 PWM 模式 （其中输出在计数周期开始时进行置位，在给定的比较匹配时复位），纵横开关在定义输出如何置位或复位方面提供了更多灵活性。它为利用定时器事件来置位或复位输出提供了可能。

输出级

由置位 / 复位纵横开关产生的波形最终通过输出级来进行 “ 后处理 ”，如

- 产生具有死区时间的补偿信号；
- 增加高频调制；
- 修改信号极性；
- 出于保护目的而切断输出。

考虑这几个功能，现在可以详细阐述第一个基本 PWM 信号。

2.1 单个 PWM 产生

本章我们将讨论：

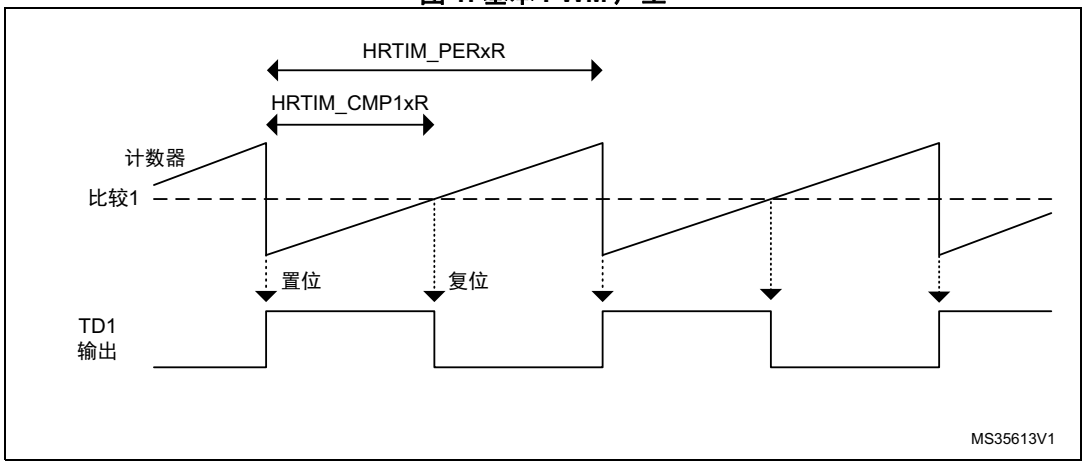
- 定时器连续模式；
- 最简单的纵横开关配置；
- 输出级使能。

PWM 信号是大部分功率转换器的基本模块，有多种用途，如驱动电机、压电蜂鸣器或仿真 DAC 转换器。

本示例表明这能够很容易地通过利用 HRTIM、对有限数量的 HRTIM 寄存器进行编程来实现。

我们考虑一个 100kHz 的 PWM 信号，占空比为 50%，在 TA1 输出上产生，以*图 1* 中为例。

图 1. 基本 PWM 产生



定时器 D 必须配置为连续 （自由运行）模式。 PWM 周期在周期寄存器 HRTIM_PERAR 中利用以下公式编程。

$$PER = \frac{f_{HRCK}}{f_{PWM}}$$

这里 (144MHz x 32)/100kHz = 46080d (0xB400)。

50% 的占空比通过将周期乘以占空比得到：PER x DC。

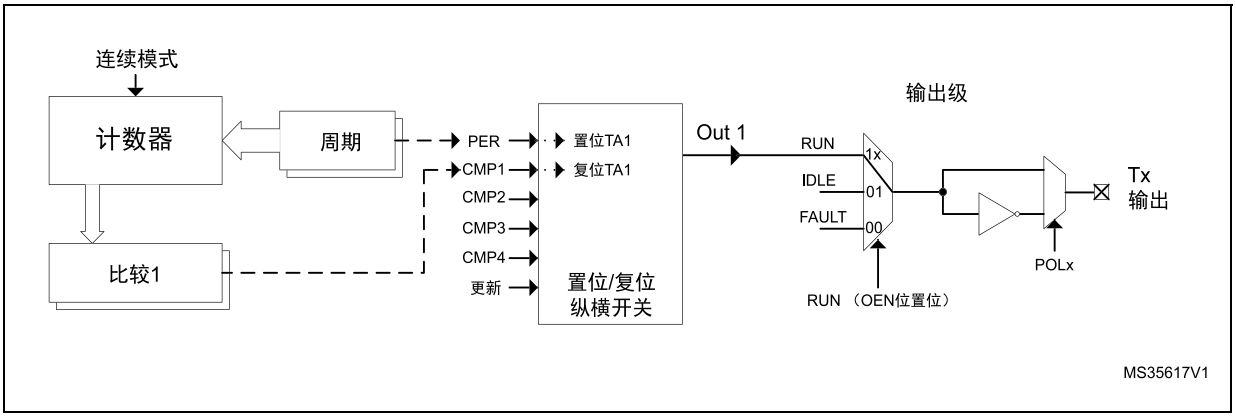
这里 0.5 x 46080d = 23040d (0x5A00)。

其波形在利用寄存器 HRTIM_SETx1R （PER 位置位）和 HRTIM_RSTx1R （CMP1 位置位）置位 / 复位纵横开关中详细说明。

最后，输出由 HRTIM_OENR 寄存器使能。

所述时序给出了简单 PWM 产生所涉及的定时器功能的概览。配置如*图 2* 中所示。

图 2. 产生基本 PWM 信号的 HRTIM 配置



HRTIM_BasicPWM 示例中提供了示例代码，下面复制的代码片段可从 HRTIM_Snippets 示例中获取。两种情况下都须通过 #define SINGLE_PWM 语句来选择示例。

```
/* TIMD 计数器工作于连续模式 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].TIMxCR = HRTIM_TIMCR_CONT;

/* 设置周期为 100kHz，且占空比 （CMP1）为 50% */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].PERxR = 0x0000B400;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP1xR = 0x00005A00;

/* TD1 输出，在 TIMD 周期下置位，在 TIMD CMP1 事件下复位 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx1R = HRTIM_SET1R_PER;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx1R = HRTIM_RST1R_CMP1;

HRTIM1->sMasterRegs.MCR = HRTIM_MCR_TDCEN;  /* 启动定时器 D */
HRTIM1->sCommonRegs.OENR = HRTIM_OENR_TD1OEN; /* 使能 TD1 输出 */
GPIO_HRTIM_outputs_Config();  /* 初始化 HRTIM GPIO 输出 */
```

2.2 产生多个 PWM

本章我们将讨论：

- 使用多个定时单元；
- 寄存器预加载。

HRTIM 能够产生多达 10 个 PWM 信号，具有 5 种独立的频率 （或在使用主定时器时有 6 种频率，参见*第 2.3 节：利用其他定时单元和主定时器来产生 PWM*）。

在下面的示例中，产生了具有 2 种不同时基的 4 个 PWM 信号。

定时器 D 在 TD1 和 TD2 上产生了 2 个移相的 100kHz，具有 25% 的占空比，条件如下：

- TD1：在 TD Period 置位，在 TD CMP1 复位；
- TD2：在 TD CMP2 置位，在 TD 周期复位。

定时器 A 在 TA1 和 TA2 上产生了 2 个移相的 33.333kHz PWM，具有 25% 的占空比，条件如下：

- TA1：在 TA Period 置位，在 TA CMP1 复位
- TA2：在 TA CMP2 置位，在 TA CMP3 复位

定时器 A 周期低于最高分辨率下可获得的最小频率，如*表 2* 中所示：

表 2. 定时器分辨率和最小 PWM 频率，对于 f_{HRTIM} = 144 MHz 时

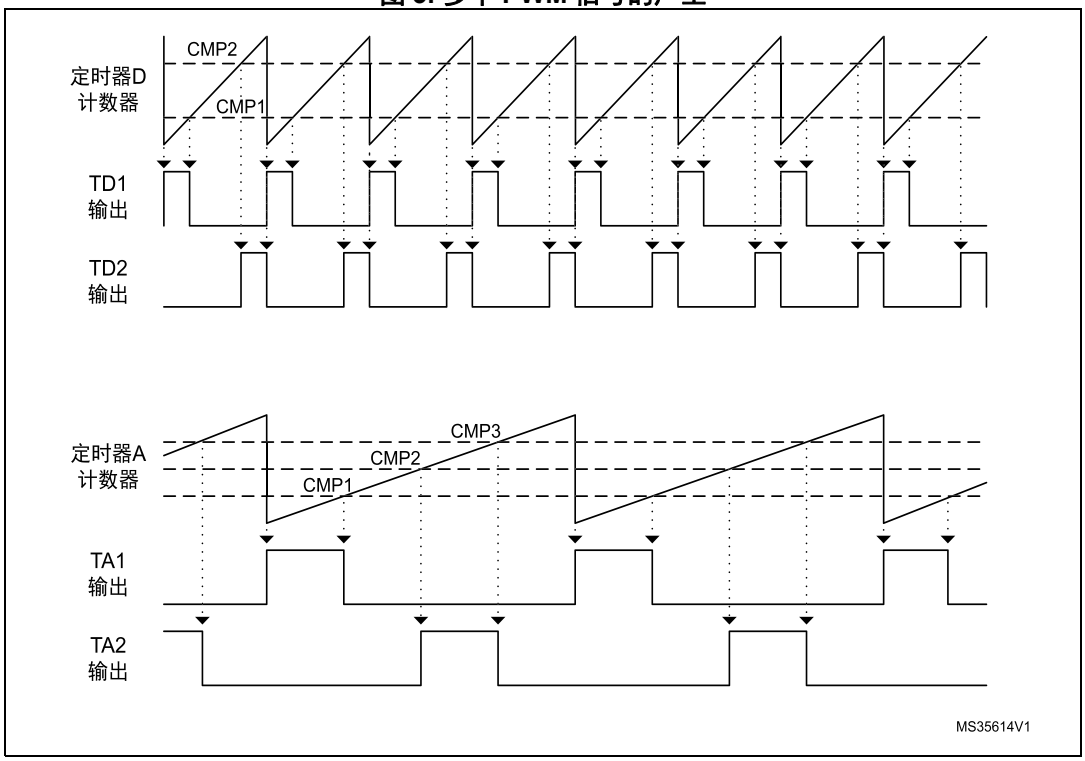
CKPSC[2:0]	预分频比	f_{HRCK} 等效频率	分辨率	最小值 PWM 频率
000	1	144x32 MHz = 4.608 GHz	217 ps	70.3 kHz
001	2	144x16 MHz = 2.304 GHz	434 ps	35.1 kHz
010	4	144x8 MHz = 1.152 GHz	868 ps	17.6 kHz
011	8	144x4 MHz = 576 MHz	1.73 ns	8.8 kHz
100	16	144x2 MHz = 288 MHz	3.47 ns	4.4 kHz
101	32	144MHz	6.95 ns	2.2 kHz
110	64	144/2 MHz = 72 MHz	13.88 ns	1.1 kHz
111	128	144/4 MHz = 36 MHz	27.7 ns	550 Hz

这种情况下，频率预分频器必须设置为 4，周期计算如下：PER = fHRCK / fPWM。这里，通过设置寄存器为 (144MHz * 8) / 33.333kHz) = 34560d (0x8700)，可得到 33.333kHz 的频率。

尽管本例中没有更新占空比，为了使其尽量接近实际用例，可利用 PREEN 位使能寄存器预加载机制。在每个周期开始时， REP （重复）事件触发预加载寄存器的传输。

注： *可以注意到，在 PWM 启动时，在 TA1/TA2 和 TD1/TD2 波形之间有一个延迟。此延迟是正常的，因为仅在第一个计时周期之后就发生了第一个更新事件 （使比较寄存器取出其编程值）。如果波形开始时没有此延迟，可通过软件 （利用 TASWU 和 TDSWU 位）强制更新寄存器，以立即更新所有激活的比较寄存器内容。*

图 3. 多个 PWM 信号的产生



HRTIM_BasicPWM 示例中提供了示例代码，下面复制的代码片段可从 HRTIM_Snippets 示例中获取。两种情况下都须通过 #define MULTIPLE_PWM 语句来选择示例。

```
/* ---------------------- 定时器 D 初始化 ------------------------ */
  /* TIMD 计数器工作于连续模式，发生 REP 事件时使能预加载 */
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].TIMxCR =
HRTIM_TIMCR_CONT + HRTIM_TIMCR_PREEN + HRTIM_TIMCR_TREPU;

    /* 周期设置为 100kHz， CMP1 设置为周期的 25%， CMP2 设置为周期的 75% */
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].PERxR = _100KHz_PERIOD;
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP1xR = _100KHz_PERIOD/4;
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP2xR = (3*_100KHz_PERIOD)/4;

  /* TD1 输出，在 TIMD 周期下置位，在 TIMD CMP1 事件下复位 */
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx1R = HRTIM_SET1R_PER;
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx1R = HRTIM_RST1R_CMP1;
  /* TD2 输出，在 TIMD CMP2 下置位，在 TIMD 周期事件下复位 */
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx2R = HRTIM_SET2R_CMP2;
  HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx2R = HRTIM_RST2R_PER;
```

```
/* ---------------------- 定时器 A 初始化 ------------------------- */
/* TIMA 计数器工作于连续模式，预分频器 = 010b ( 除 以 4) */
/* 在 REP 事件下使能预加载 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].TIMxCR = HRTIM_TIMCR_CONT
+ HRTIM_TIMCR_PREEN + HRTIM_TIMCR_TREPU + HRTIM_TIMCR_CK_PSC_1;

/* 设置周期为 33kHz，且占空比为 25% */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].PERxR = _33KHz_PERIOD;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].CMP1xR = _33KHz_PERIOD/4;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].CMP2xR = _33KHz_PERIOD/2;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].CMP3xR = (3*_33KHz_PERIOD)/4;

/* TA1 输出，在 TIMA 周期下置位，在 TIMA CMP1 事件下复位 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].SETx1R = HRTIM_SET1R_PER;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].RSTx1R = HRTIM_RST1R_CMP1;
/* TA2 输出，在 TIMA CMP2 下置位，在 TIMA 周期事件下复位 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].SETx2R = HRTIM_SET2R_CMP2;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_A].RSTx2R = HRTIM_RST2R_CMP3;

  /* 使能 TA1, TA2, TD1 和 TD2 输出 */
  HRTIM1->sCommonRegs.OENR = HRTIM_OENR_TA1OEN + HRTIM_OENR_TA2OEN +
  HRTIM_OENR_TD1OEN + HRTIM_OENR_TD2OEN;
  GPIO_HRTIM_outputs_Config();  /* 初始化 HRTIM GPIO 输出 */

  /* 启动定时器 A 和定时器 D */
  HRTIM1->sMasterRegs.MCR = HRTIM_MCR_TACEN + HRTIM_MCR_TDCEN;
```

2.3 利用其他定时单元和主定时器来产生 PWM

本章我们将讨论：

- 产生与给定定时器无关的输出信号。

本例表明，由于存在置位 / 复位纵横开关，可利用其他可用定时器在给定输出上产生 PWM 信号 （或其他波形）。这在下列情况下值得关注：

- 利用主定时器产生第 6 个 PWM 独立频率，如下例所示；
- 解决引脚限制，例如即便 TE1 和 TE2 输出不可用时，也可利用定时器 E 来产生波形（通常在引脚数少的封装中）。

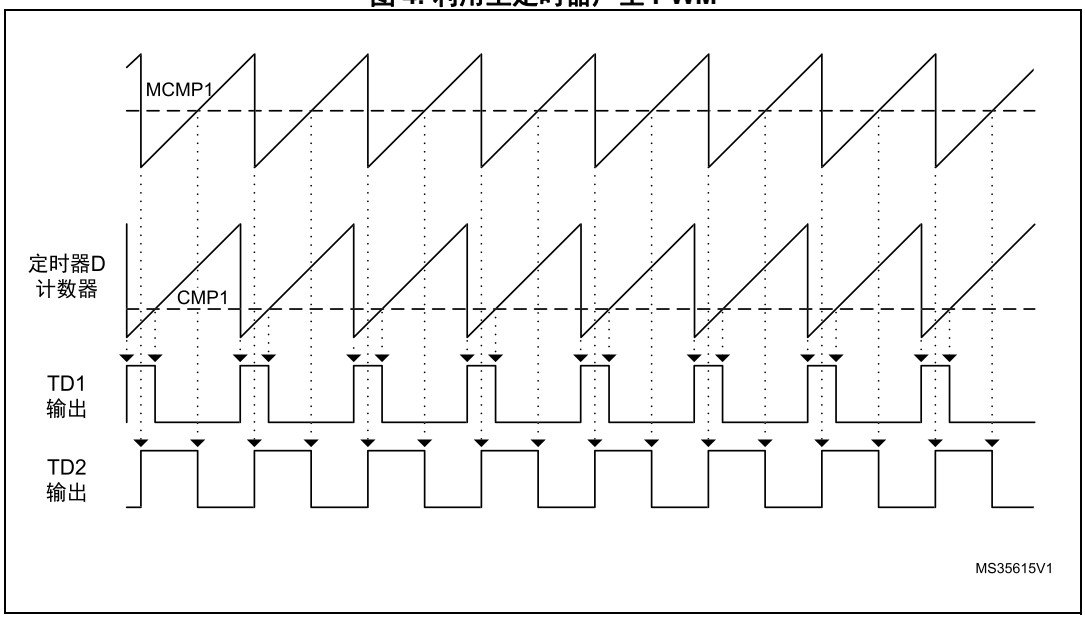
注： *对于共享资源的所有定时器，要求其必须具有相同的预分频比 （例如，如果主定时器控制 TA1 或 TA2 输出，则主定时器和定时器 A 必须具有相同的 CKPSC[2:0] 值）。*

在下面的示例中， TD1 和 TD2 输出上会产生 2 个 PWM 信号 （开关频率略有不同），条件如下：

- TD1：在 TD Period 置位，在 TD CMP1 复位；
- TD2：在 Master Period 置位，在 Master CMP1 复位。

频率被设置为略有不同的值，以便于在示波器上显示 （两个信号具有相对缓慢的相移变化），并可验证两个信号完全异步。

图 4. 利用主定时器产生 PWM



所提供的代码为 HRTIM_BasicPWM 示例，下面复制的代码片段可从 HRTIM_Snippets 示例中获取。两种情况下都须通过 #define PWM_MASTER 语句来选择示例。

```
/* ---------------------- 主定时器初始化 ------------------------ */
 /* 主计数器工作于连续模式， REP 事件下使能预加载 */
 HRTIM1->sMasterRegs.MCR = HRTIM_MCR_CONT + HRTIM_MCR_PREEN +
HRTIM_MCR_MREPU;
 /* 设置周期为 101kHz，且占空比为 50% */
 HRTIM1->sMasterRegs.MPER = _100KHz_Plus_PERIOD;
 HRTIM1->sMasterRegs.MCMP1R = _100KHz_Plus_PERIOD/2;

/* ---------------------- 定时器 D 初始化 ------------------------ */
 /* TIMD 计数器工作于连续模式，发生 REP 事件时使能预加载 */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].TIMxCR =
HRTIM_TIMCR_CONT + HRTIM_TIMCR_PREEN + HRTIM_TIMCR_TREPU;

 /* 设置周期为 100kHz，且占空比 （CMP1）为 50% */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].PERxR = _100KHz_PERIOD;
```

```
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP1xR = _100KHz_PERIOD/4;

/* TD1 输出，在 TIMD 周期下置位，在 TIMD CMP1 事件下复位 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx1R = HRTIM_SET1R_PER;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx1R = HRTIM_RST1R_CMP1;

/* TD2 输出，在 TIMD CMP2 下置位，在 TIMD 周期事件下复位 */
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx2R = HRTIM_SET2R_MSTPER;
HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx2R = HRTIM_RST2R_MSTCMP1;

/* 使能 TD1 和 TD2 输出 */
HRTIM1->sCommonRegs.OENR = HRTIM_OENR_TD1OEN + HRTIM_OENR_TD2OEN;
GPIO_HRTIM_outputs_Config();  /* 初始化 HRTIM GPIO 输出 */

/* 启动主定时器和定时器 D */
HRTIM1->sMasterRegs.MCR |= HRTIM_MCR_MCEN + HRTIM_MCR_TDCEN;
```

2.4 任意波形产生

本章我们将讨论：

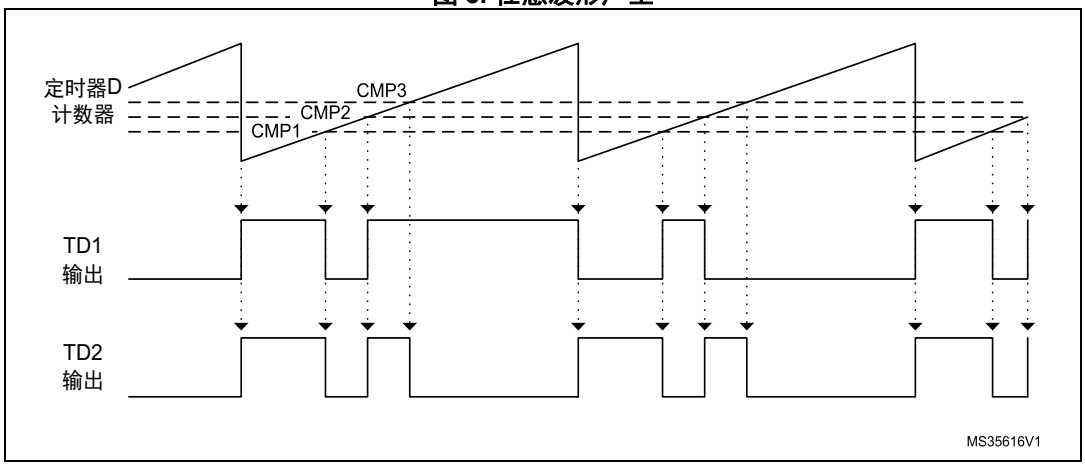
- 每个周期有多个置位 / 复位 / 切换请求的波形。

本例表明，由于纵横开关的 32 个并发置位 / 复位源，能够很容易地产生除 PWM 以外的波形。

在下面的示例中， TD1 和 TD2 输出上会产生 2 个任意波形，条件如下：

- TD1：TD Period 时电平翻转， TD CMP1 时电平翻转， TD CMP2 时电平翻转；
- TD2：TD Period 和 TD CMP3 下置位， TD CMP2 和 TD CMP3 下复位。

图 5. 任意波形产生



所提供的代码为 HRTIM_BasicPWM 示例，下面复制的代码片段可从 HRTIM_Snippets 示例中获取。两种情况下都须通过 #define ARBITRARY_WAVEFORM 语句来选择示例。

```
/* ---------------------- 定时器 D 初始化 ------------------------- */
 /* TIMD 计数器工作于连续模式，发生 REP 事件时使能预加载 */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].TIMxCR = HRTIM_TIMCR_CONT
+ HRTIM_TIMCR_PREEN + HRTIM_TIMCR_TREPU;

 /* 周期设置为 100kHz，且为边缘定时 */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].PERxR = _100KHz_PERIOD;
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP1xR = _100KHz_PERIOD/4;
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP2xR = (3*_100KHz_PERIOD)/8;
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].CMP3xR = _100KHz_PERIOD/2;

 /* TD1 在 TIMD 周期、 CMP1 和 CMP2 事件下切换 */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx1R = HRTIM_SET1R_PER +
HRTIM_SET1R_CMP1 + HRTIM_SET1R_CMP2;
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx1R = HRTIM_RST1R_PER +
HRTIM_RST1R_CMP1 + HRTIM_RST1R_CMP2;

 /* TD2 输出，在 TIMD PER 和 CMP2 下置位，在 TIMD CMP1 和 CMP3 事件下复位 */
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].SETx2R = HRTIM_SET2R_PER +
HRTIM_SET2R_CMP2;
 HRTIM1->sTimerxRegs[HRTIM_TIMERINDEX_TIMER_D].RSTx2R = HRTIM_RST2R_CMP1 +
HRTIM_RST2R_CMP3;

 /* 使能 TD1 和 TD2 输出 */
 HRTIM1->sCommonRegs.OENR = HRTIM_OENR_TD1OEN + HRTIM_OENR_TD2OEN;
 GPIO_HRTIM_outputs_Config();  /* 初始化 HRTIM GPIO 输出 */

 /* 启动定时器 D */
 HRTIM1->sMasterRegs.MCR = HRTIM_MCR_TDCEN;
```

3 电压模式双降压转换器

本章我们将讨论：

- 重复计数器中断；
- ADC 触发；
- 重复事件下寄存器更新。

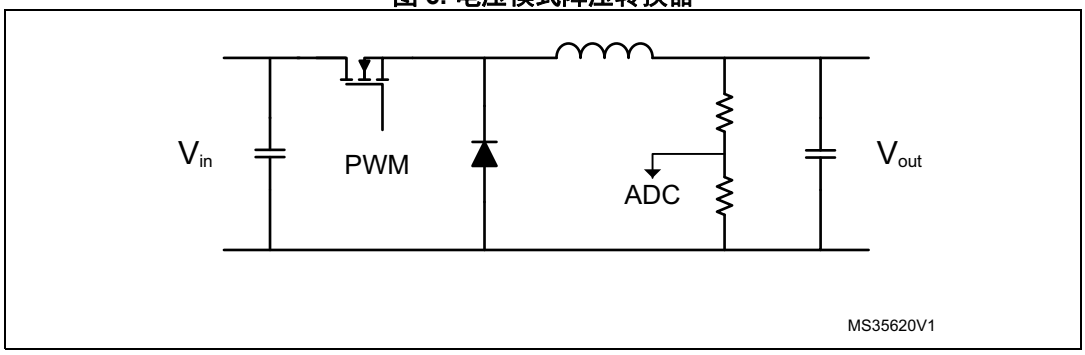
降压转换器通常用来实现步降电压转换。它们通常在固定频率下工作，具有可变的占空比。在理想的转换器中， V_{in}/V_{out} 比仅依赖于施加到电源开关上的占空比 D：

$$V_{out} = D \times V_{in}$$

实际上，占空比可通过控制算法来调节，以维持所需要的输出电压。本例中，考虑电压模式降压转换器：基于转换器输出电压读数来控制占空比。最后可增加输入电压读数来实现前馈补偿。

功率级拓扑如图*图 6* 中所示，其中还显示了由 ADC （用来实现调节）给出的输出电压读数。

图 6. 电压模式降压转换器



本例中给出了软件包（HRTIM_DualBuck），HRTIM 通过编程，实现对两个并行工作的降压转换器 （具有相同的频率和非重叠 ON 时间）的控制。

HRTIM 工作于连续模式， PWM 信号定义如下：

- TD1：在 TD Period 置位，在 TD CMP1 复位；
- TD2：在 TD CMP2 置位，在 TD 周期复位。

TD2 产生如*第 2.2 节：产生多个 PWM* 中所示，下图中不再给出。

本例中应用了另外两个比较事件来触发 ADC。

开关 ON 期间在 TD1 和 TD2 上计算了 CMP3 和 CMP4 时间 。这允许进行平均电压测量，并确保转换中没有电平震荡和由电源开关带来的开关噪声。

图 7 显示了计数器、生成的 PWM 输出和产生的输出电压波纹放大图，以及 TD1 通道的 ADC 采样点。在给定高 PWM 频率时，重复计数器用来减少中断服务程序的数量。*图 7* 中自动重载率设为 1，因此 ISR 速率为 PWM 频率的一半。示例中，重复率设为 127，以便获得极低的变化率。

图 7. VM 降压波形，包括 ADC 采样和中断

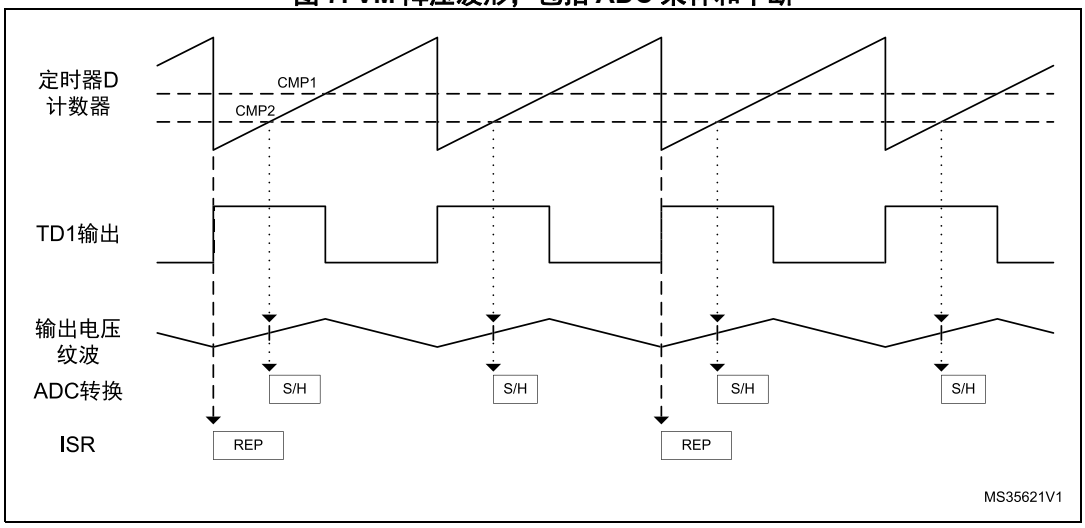
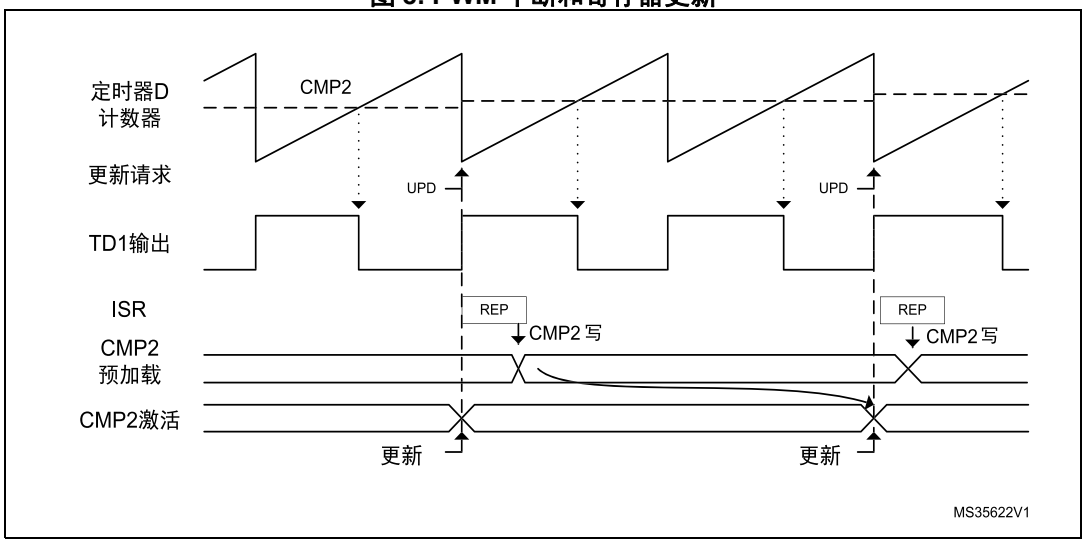


图 8 显示了当中断程序中 TD1 上的工作周期完成时，如何更新占空比。编程定时器，使每次重复事件发生时对寄存器进行更新。

图 8. PWM 中断和寄存器更新



双降压演示概览

TD1 的工作周期在中断服务程序 （在中断事件下产生）中不断变化，来模拟真实的转换器管理。可在 STM32F334 探索套件 TP3 测试点上对一个反映 PWM 占空比的低通滤波信号进行监测。

TD2 上的占空比是恒定的。

配置 ADC，使在每个输出的 ON 时间中触发转换。采用相同的 ADC 注入触发器来完成 2 个转换 （由于 2 个信号不重叠，因此这是可能的）。 ADC 编程为间断注入模式 （1 个转换在 CMP3 下触发，另一个在 CMP4 下触发）。

2 个转换在相同输入 （PA4）下完成：TD1 的低通滤波 PWM 可连接到 PA4，采用调试器来监测 ADC 转换。 对于实际用例，很容易为 ADC 重编程，在每个降压输出电压上实现转换。

FAULT1 输入在 PA12 （低电平激活）上使能，可关断 PWM （详细内容见章节 4）。触发错误时 （PA12 输入连接到 GND），仅 TD1 信号停止。可通过按下用户按钮复位系统功能演示。

LED 指示下列情况：

- 蓝色 LED：正常工作时闪烁；
- 红色 LED：触发 FAULT 时闪烁；
- 橙色 LED：指示 PWM 刷新 ISR 发生和持续。

演示代码

HRTIM_DualBuck 示例中提供了示例代码。

4 同步电压模式降压转换器整流和错误保护

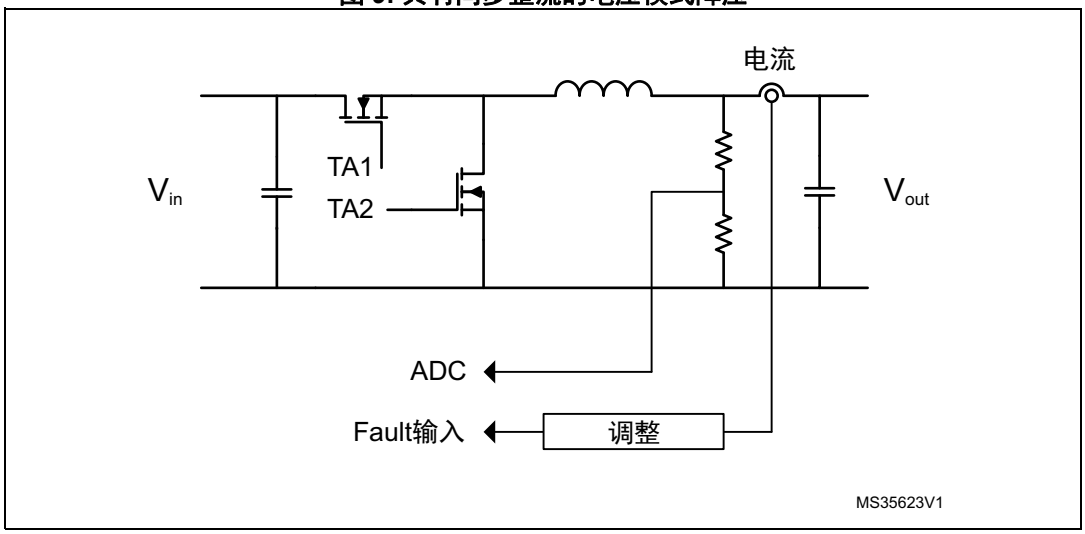
本章我们将讨论：

- 死区时间；
- 数字 FAULT 保护；
- 突发模式控制器。

本例给出了一个电压模式降压控制器，其续流二极管由 MOSFET 代替以实现同步整流。通过降低电感被退磁到输出电容时的损耗，可提高转换器效率。

图 9 显示了功率级拓扑。它包括输出电压读数和过流保护 （利用 FAULT 输入），使在电流超出可编程阈值时关闭转换器。为简单起见，此处不讨论电流传感器和调整电路；预期的 FAULT 反馈 （在 FLT1 输入上）为数字信号 （在 PA12 输入上）。

图 9. 具有同步整流的电压模式降压



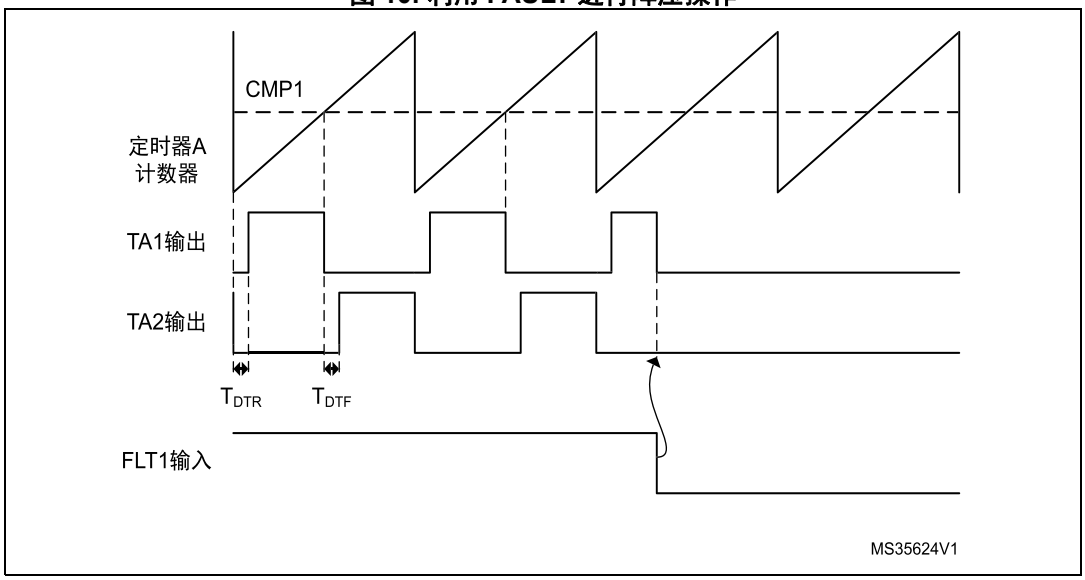
HRTIM 工作于连续模式， PWM 信号定义如下：

- TA1：在 TA Period 置位，在 TA CMP2 复位
- TA2：利用死区时间发生器，与 TA1 互补 （相同的上升沿和下降沿死区时间）

转换器利用 FLT1 数字输入进行过流保护，低电压有效。通过对正输出极性和未激活的错误状态进行编程， TA1 和 TA2 输出在发生 FLT1 事件的情况下被强制为低电平。

图 10 显示了 TA1 和 TA2 输出上的降压控制波形，包括错误发生的情形。

图 10. 利用 FAULT 进行降压操作



同步整流降压演示概览

TA1 的工作周期在中断服务程序 （在中断事件下产生）中不断变化，来模拟真实的转换器管理。对于 TA1 和 TA2， FAULT1 输入在 PA12 （低电平激活）上使能，可关断 PWM （低电平敏感）。

触发错误时 （PA12 输入连接到 GND）， TA1 和 TA2 信号停止。可通过按下用户按钮复位系统功能演示。

LED 指示下列情况：

- 蓝色 LED：正常工作时闪烁；
- 红色 LED：触发 FAULT 时闪烁；
- 橙色 LED：指示 PWM 更新 ISR 发生和持续。

配置 ADC，使在转换器 ON 时间内，在 PA1 （V_{in}）和 PA3 （V_{out}）输入上触发转换。为了运行该演示， BUCK-BOOST 转换器的 V_{out} 输入必须连接到 5V_O 电源。所产生的电压可在 V_{out} 引脚上得到。

注： *为了旁路探索套件的 BOOST 级， PA11 引脚必须强制为 1 （使 T6 PMOS 切换为 ON）。*

演示代码

HRTIM_BuckSyncRect 示例中提供了示例代码。

5 非反相降压 - 升压转换器

本章我们将讨论：

- 4 开关的转换器驱动；
- 0% 和 100% 的 PWM 产生。

本例显示了如何配置 HRTIM 来驱动非反相降压 - 升压转换器，并接通从降压到升压模式的转变。尽管此配置比传统反相降压 - 升压需要更多开关，但它的优势是能提供参考地的正输出电压。

图 11 显示了在 STM32F334 探索板上实现的功率级拓扑。

图 11. 非反相降压 - 升压转换器

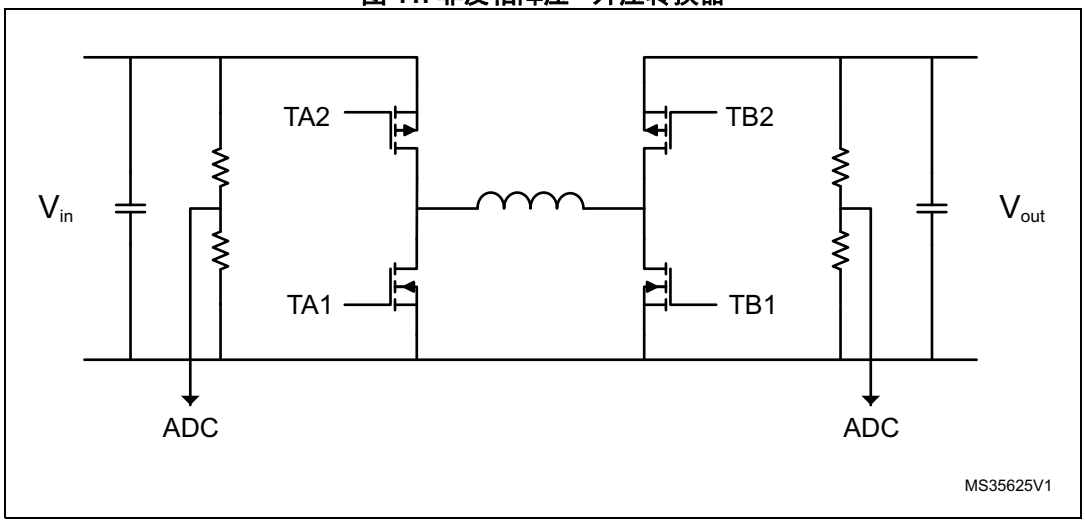
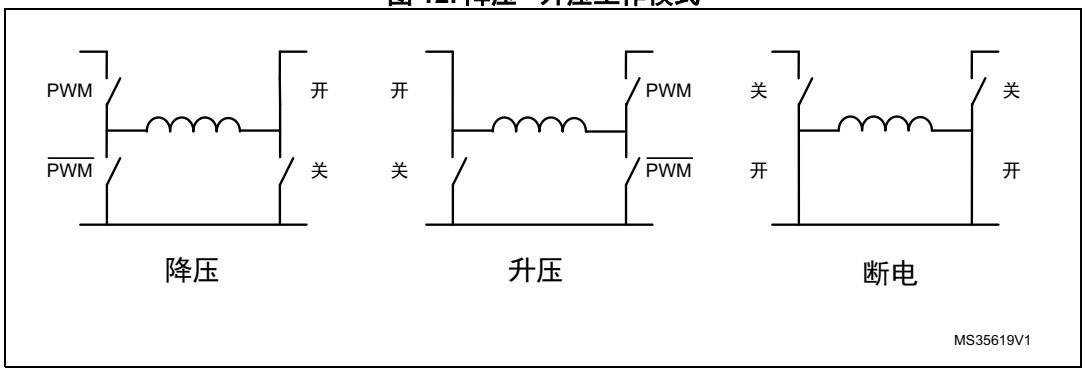


图 12 用简化图总结了演示的 3 个工作模式 （MOSFET 反并行二极管未给出）。在降压和升压模式中都应用了同步整流方案 （驱动互补 MOSFET 来降低其本征二极管中的传导损耗）。第 3 种模式 “ 断电 ” 是必要的，这是为了避免当转换器从升压模式切换为降压模式时，电流流回输入源：MCU 保持此模式，直至输出电容放电至低于输入电压。

通过在驱动半桥的互补输出上强制施加 0% 或 100% 的 PWM，来保持恒定的 ON/OFF 状态。

图 12. 降压 - 升压工作模式



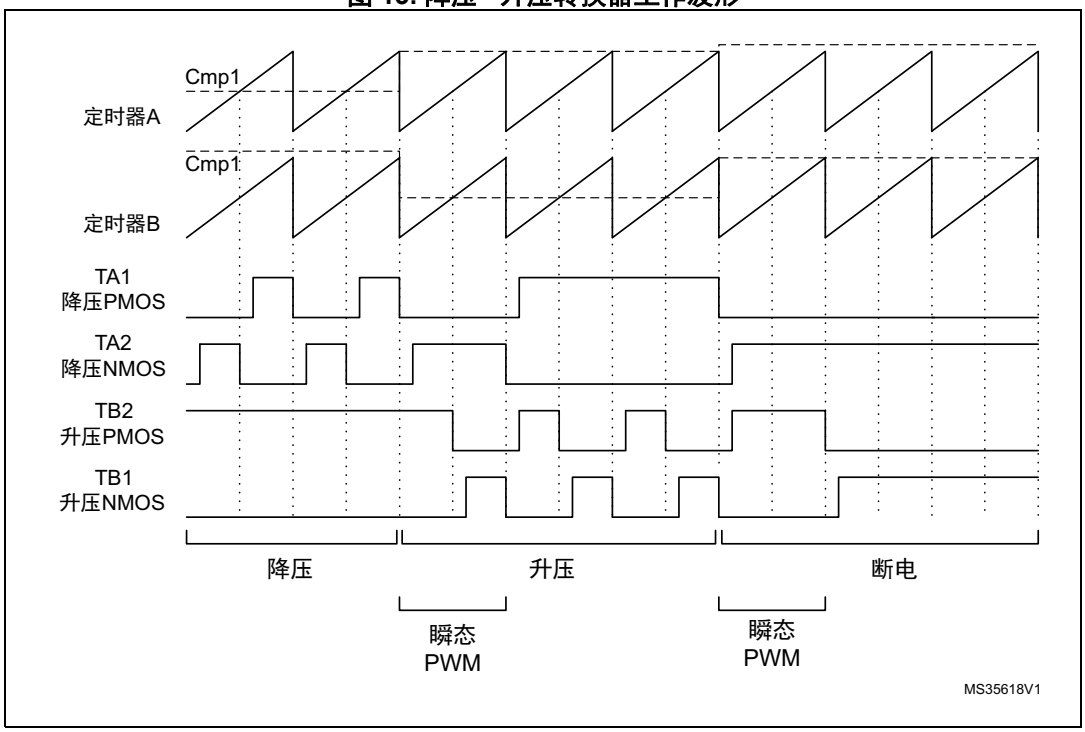
HRTIM 工作于连续模式， PWM 信号定义如下：

- TA1：在 TA CMP1 置位，在 TA Period 复位；
- TA2：利用死区时间发生器，与 TA1 互补 （相同的上升沿和下降沿死区时间）；
- TB1：在 TB CMP1 置位，在 TB Period 复位；
- TB2：利用死区时间发生器，与 TB1 互补 （相同的上升沿和下降沿死区时间）。

利用 FLT1 数字输入来保护转换器，低电压有效。通过对正输出极性和未激活的错误状态进行编程，使所有输出在发生 FLT1 事件的情况下被强制为低电平。

图 13 显示了对于 3 种工作模式下 4 个输出上的降压 - 升压控制波形。

图 13. 降压 - 升压转换器工作波形



通过将比较寄存器值编程为 0 或 100%，得到 0% 和 100% 的占空比，这会使两个互补输出保持为 ON/OFF：

- 当 CMP1 值等于 PER 值时，占空比为 100%（CMP1 置位事件高于 PER 复位事件: 参见参考手册中详述的硬件优先级方案：当两个事件同时发生时，优先级如下：CMP4>CMP3>CMP2>CMP1>PER）；
- 当 CMP1 值大于 PER 值时，占空比为 0% （不再产生置位事件）。

降压 - 升压演示概览

为了运行该演示，BUCK-BOOST 转换器的 VV_{in} 输入必须连接到 STM32F334 探索套件电源的 5V_O 输出引脚。所产生的电压可在 V_{out} 引脚上得到。

演示程序开始处于 BUCK 模式，在 TIMA IRQ 处理器中慢慢调整占空比，使 V_{out} （小于 V_{in} 值）连续变化。如果按下按钮且电压低于 5V，则升压模式使能 （电压检查可防止超出探索套件最大输出电压）。电压增加到大于 V_{in} 值，具有固定的占空比。如果按钮再次按下，在重新使能降压模式之前，输出电容首先断电，降至 4.5V。

配置 ADC，使在转换器 ON 时间内、在 PA1 （Vin）和 PA3 （V_{out}）输入上触发转换。在主程序中这些值转换为 mV，以便在演示中利用调试程序得到直接电压读数 （利用运行时间刷新的钟表）。

对于所有输出， FAULT1 输入在 PA12 （低电平激活）上使能，可关断 PWM （低电平敏感）。触发错误时（PA12 输入连接到 GND）， TA1、TA2、TB1、TB2 信号停止。可通过按下用户按钮复位系统功能演示。

LED 指示下列情况：

- 绿色 LED：降压操作时闪烁
- 蓝色 LED：升压操作时闪烁
- 红色 LED：触发 FAULT 时闪烁
- 橙色 LED：指示 PWM 更新 ISR 发生和持续。

演示代码

HRTIM_BuckBoost 示例中提供了示例代码。

6 过渡模式功率因数控制器

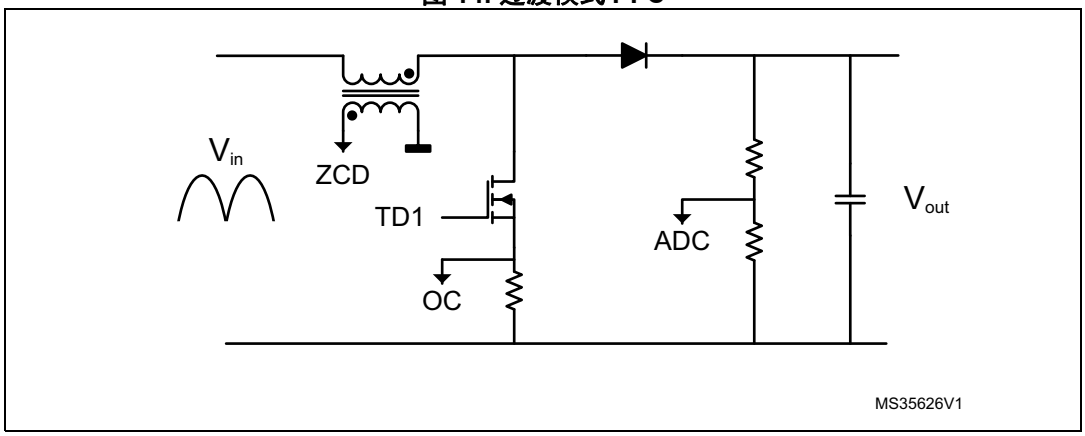
本章我们将讨论：

- 外部事件调节和过滤；
- 定时器计数器复位；
- 事件消隐窗口；
- 捕获单元；
- 恒定 T_{on} 时间转换器。

本例给出了过渡模式 （也称为边界导通模式）功率因数控制器 （PFC）。

图 14 显示了与升压转换器类似的拓扑。它具有一个输出电压读数和退磁检测线圈 （耦合到主电感，用于零电流检测）。

图 14. 过渡模式 PFC



基本工作原理是，在固定的 T_{on} 时间内建立进入电感的电流。

该电流将随后在 T_{off} 时间内消退，当电流为零时重新开始此循环周期。可利用过零检测（ZCD）电路 （实际上由主电感上的辅助线圈构成）对其进行检测。由于有恒定 T_{on}，电感中的电流峰值与整流的交流输入电压直接成比例，这提供了功率因数校正。

此转换器具有恒定的 T_{on}，由于 T_{off} 变化其频率也是变化的 （取决于输入电压）。它还必须包括一些功能，以便在未检测到过零电压时工作，或在过流 （OC）时限制 T_{on}。 OC 反馈通常由内置比较器调节，发送到外部事件通道。

注： *下面的工作原理也可应用于关断时间恒定的转换器。*

图 15 和*图 16* 显示了工作模式变化过程中的波形，其参数定义如下：

- T_{on} min：在此期间，丢弃伪过流 （通常它们是续流二极管恢复电流）。它表示为 OC 消隐并由 CMP3 编程；
- T_{on} max：实际上是转换器置位点。它由 CMP1 定义；
- T_{off} min：在电流极限接近 0 时（退磁非常快）限制频率。自动延迟模式下它由 CMP2 定义；
- T_{off} max：避免在无 ZCD 发生时系统卡死。自动延迟模式下它由 CMP4 定义。

图 15. T_{on} max 和过流时的过渡模式 PFC 操作

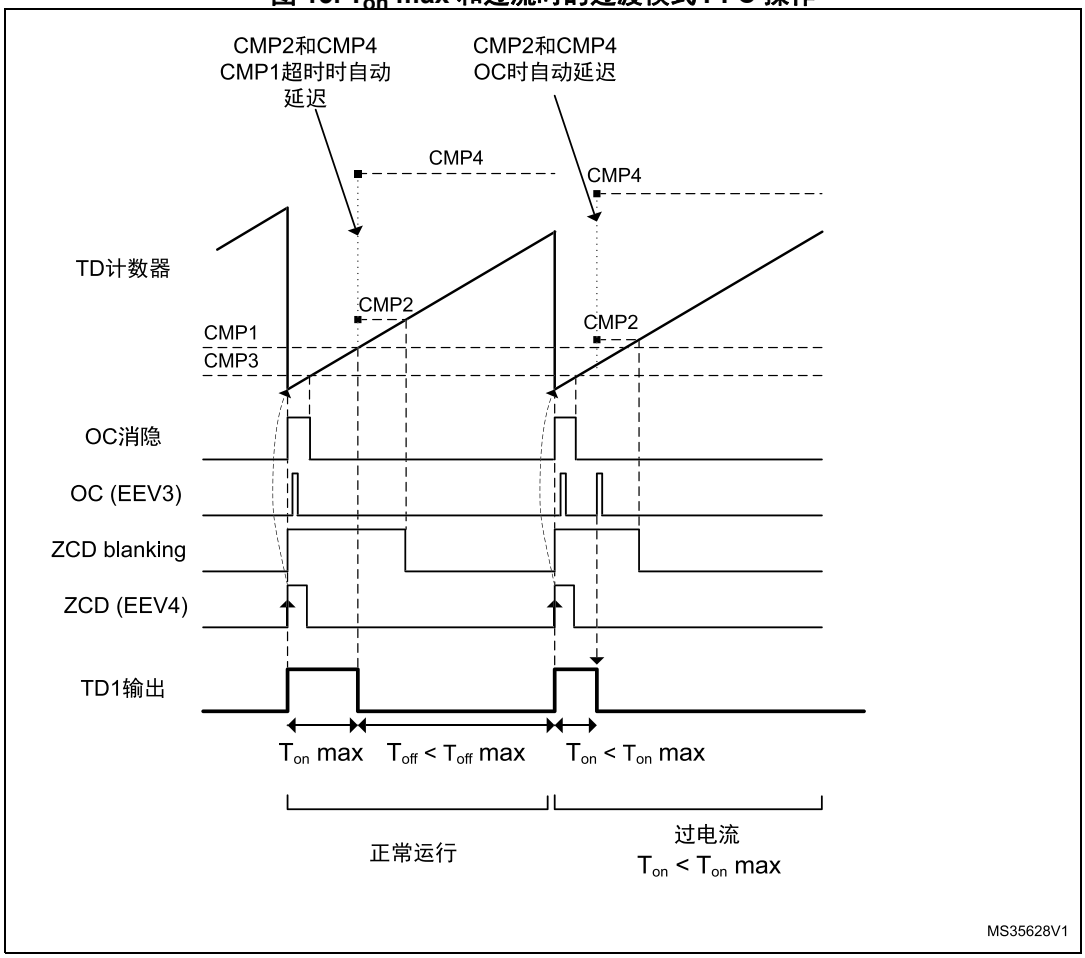
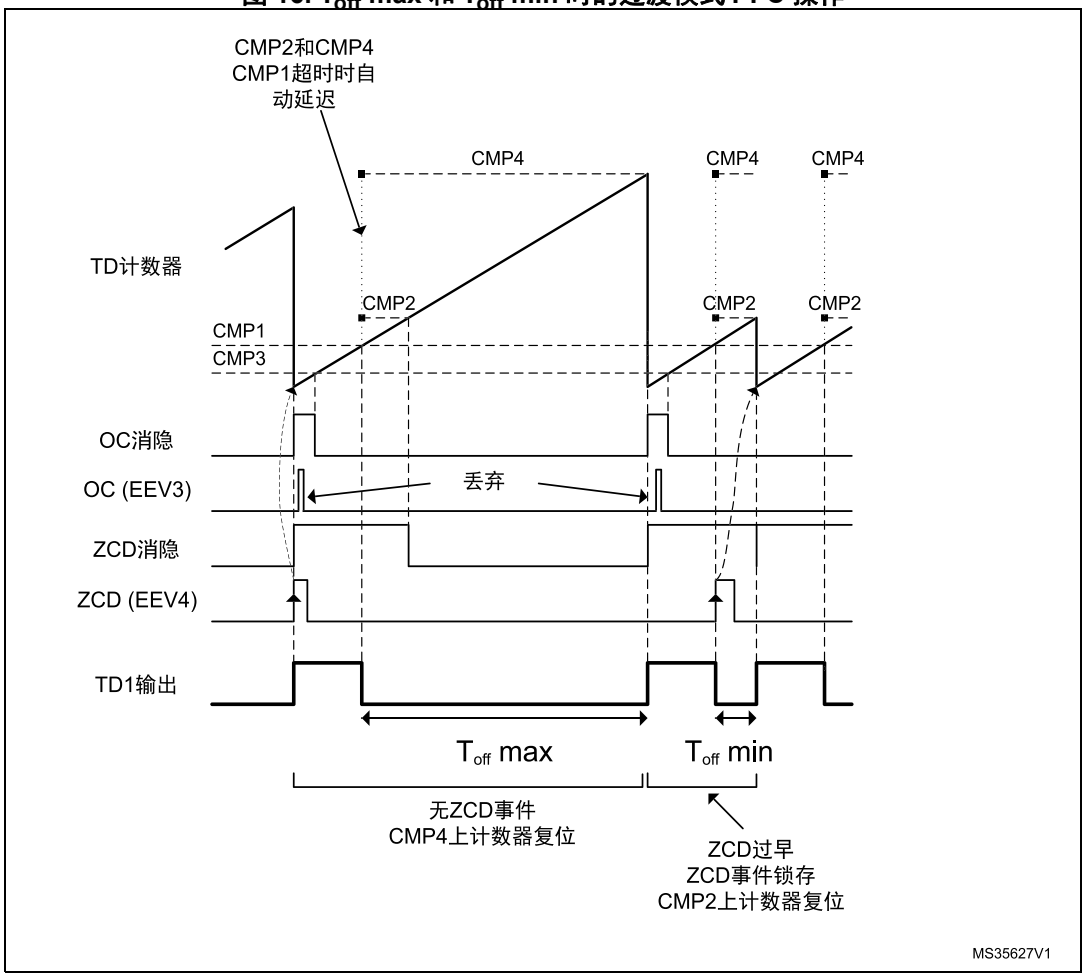


图 16. T_{off} max 和 T_{off} min 时的过渡模式 PFC 操作



HRTIM 工作于连续模式， TD1 信号定义如下：

- 置位，（TD CMP4 或 （ZCD $_{锁存}$。 TDCMP2))
- 复位，当 (TD CMP1 或 (OC. TDCMP3))

ZCD $_{锁存}$。 TDCMP2 指示 ZCD 事件由消隐窗口过滤，从定时器 D 计数器复位开始，至 TD CMP2 匹配结束。 ZCD 事件锁存：如果发生在消隐窗口过程中，则不丢弃，并在消隐周期结束时起作用。 ZCD 信号应用到外部事件 4。

OC. TDCMP3 指示 OC 事件由消隐窗口过滤，从定时器 D 计数器复位开始，至 TD CMP3 匹配结束。 OC 事件不锁存：如果发生在消隐窗口过程中，则丢弃。 OC 信号应用到外部事件 3。

两个 T_{off} 值 （基于 CMP2 和 CMP4）都是自动延迟的：此时间必须与输出信号下降沿相关（CMP1 匹配或 OC 事件下发生）。 CMP2 和 CMP4 事件在自动延迟模式下 CMP1 超时时产生，分别基于捕获 1 和捕获 2 事件。两个捕获单元由 OC 信号 （EEV3）触发。

ZCD 事件或 CMP4 匹配 （超时情况下）时，定时器 D 计数器复位。

利用 FLT1 数字输入来保护转换器，低电压有效。通过对正输出极性和未激活的错误状态进行编程， TD1 输出在发生 FLT1 事件的情况下被强制为低电平。

过渡模式 PFC 演示概览

为了运行演示并测试所有的工作模式，有必要利用函数发生器来仿真反馈 2 输入信号：

- 过流 （OC，在 EEV3/PB7 上）；
- 过零检测 （ZCD，在 EEV4/PB6 上）。

不同工作模式可测试如下：

- 如果 OC 信号在 T_{on} 时间内产生，则脉冲将缩短；
- ZCD 信号复位定时器计数器，并相应地引起开关频率发生变化。

FAULT1 输入在 PA12 （低电平激活）上使能，可关断 PWM （低电平敏感）。触发错误时（PA12 输入连接到 GND）， TD1 信号停止。可通过按下用户按钮复位系统功能演示。

LED 指示下列情况：

- 绿色 LED：正常工作时闪烁；
- 橙色 LED：触发 FAULT 时闪烁；

演示代码

HRTIM_TM_PFC 示例中提供了示例代码。

7 其他示例

本指南将逐步扩展。

将会有更多示例，请在 www.st.com 上查看 STM32F334 页面。

8 版本历史

表 3. 文档版本历史

日期	版本	变更
2014 年 6 月 19 日	1	初始版本。

表 4. 中文文档版本历史

日期	版本	变更
2015 年 12 月 23 日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。