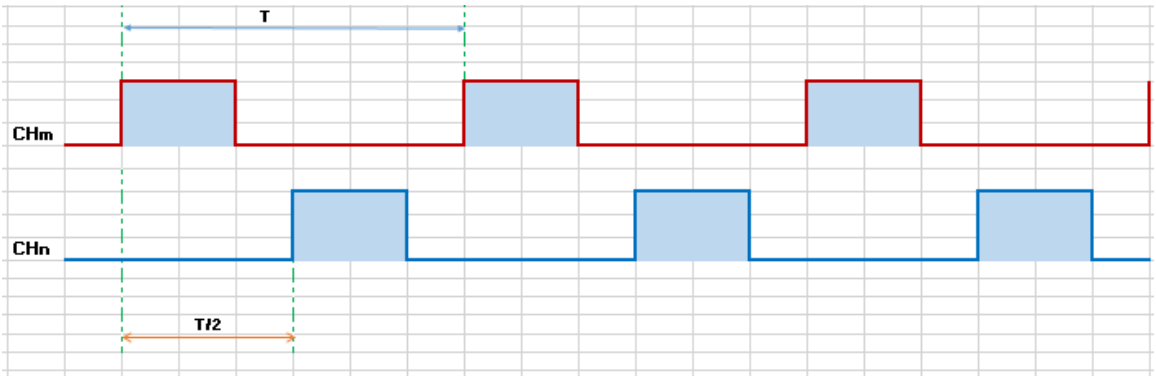
基于 STM32 定时器实现定制波形的示例

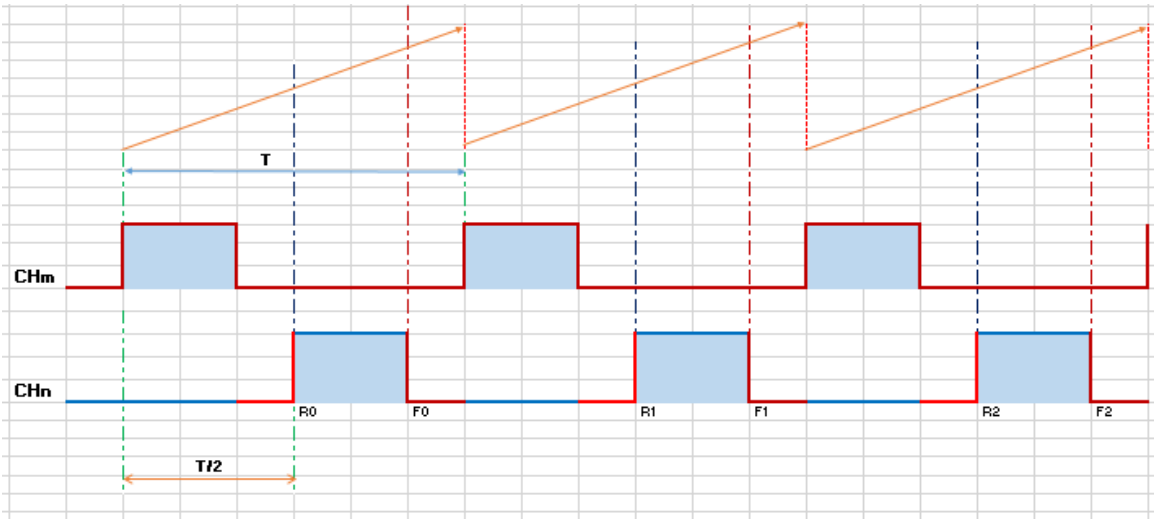
有人想实现如下 PWM 输出波形，周期 T=12us,两路输出波形特征一样，只是第 2 路【下图中的 CHn】波形的输出比第 1 路【下图中的 CHm】滞后 T/2。另外，还要保证二者占空比在 0~50%范围内可以同步调节。



对于这种输出波形，如果使用 2 个定时器来做会相对方便点，通过定时器的主从模式来实现。现在客户希望使用 1 个定时器来完成，那如何实现呢？

当然，用 1 个带多比较通道的定时器也是可以实现的。我们可以使用 PWM 输出模式来实现第 1 路，使用 OC toggle 模式来实现第 2 路。下面以 STM32F334 Nucleo 开发板来着手配置、编程、验证。

这里不妨选用 TIM3 来实现该输出波形。TIM3 的时钟源目前配置为 64Mhz,不做预分频，使用向上计数模式。一个计数周期对应的 ARR 是 （12*64）-1，即 767。TIM3 的通道 1 输出对应于下图中第 1 路,TIM3 的通道 3 输出对应于第 2 路.【图中的黄色斜箭头表示计数器计数变化趋势，从 0 到 ARR，并周期性循环计数。】



因为 TIM3 通道 1 采用 PWM 模式来输出，设置好 CCR1、ARR 的值即可。通道 3 采用 OC 切换输出模式来实现。对于通道 3，在一个周期内有两个翻转点，即上图中第 2 路上升沿的 R 点和下降沿的 F 点。结合本实例需求，不难得知，R 点所对应的比较值【CCR3】为

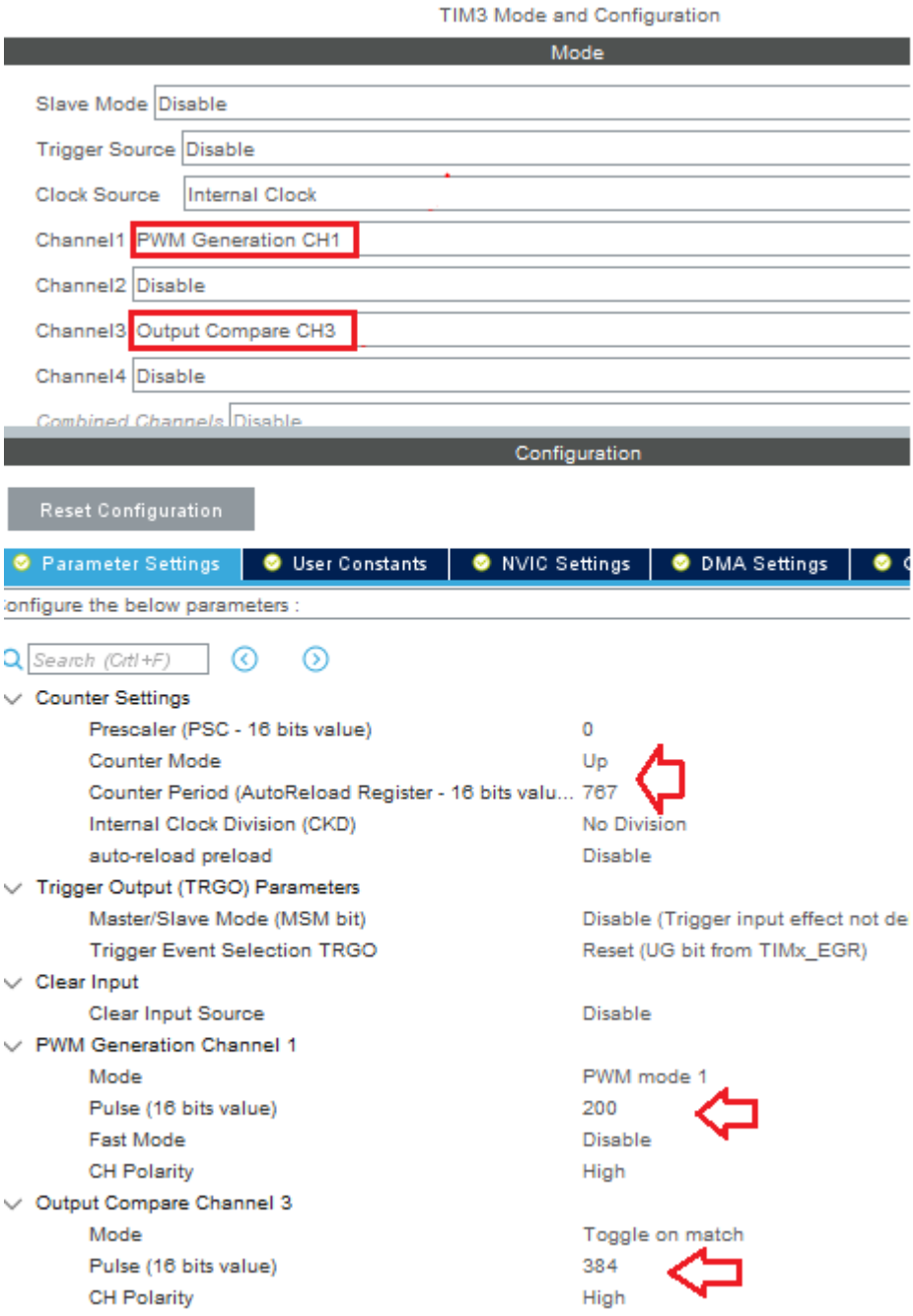
768/2，即 384 并保持不变。F 点的比较值的大小决定其占空比。由于这里实际需求的占空比不超过 50%，正常来讲，则 F 点可以设置的比较值最大不会超过 ARR 的值。

我们可以借助 TIM3 CH3 的比较事件触发 DMA，通过 DMA 来传输存储在内存中 R 点/F 点所对应的比较值以更新 CCR3 寄存器内容。在 R 点发生比较事件时，更新 F 点的比较值，在 F 点发生比较事件时更新 R 点的比较值。刚才前面说过了，这里上升沿 R 点的比较值始终保持不变，即 384，我们是通过改变 F 点的比较值来改变 TIM3 通道 3 的占空比。

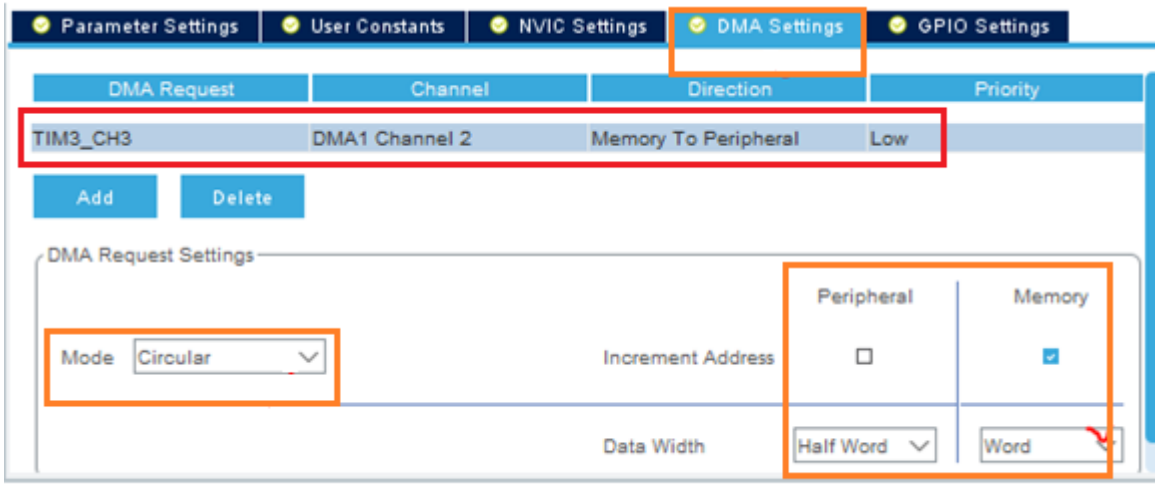
另外，我们在 TIM3 通道 1 的比较中断里修改 CCR1 的值和通道 3 在 F 点所对应的 CCR3 比较值,最终实现 2 路占空比 0~50%灵活可调的输出。

大致原理就介绍到这里。下面描述从基于 STM32CubeMx 进行初始配置开始到实现输出的全过程。下面有关 RCC/SYS 等的初始配置从略，不附加截图了。

1、关于定时器 TIM3 的基本时基参数配置如下：



2、基于定时器事件的 DMA 传输配置如下：



有关 NVIC 的配置可以按需操作，其中 DMA 的传输中断 CubeMx 工具默认帮我们开启。这里开启了 TIM3 相关的 NVIC 配置。将来在 TIM3 的 OC1 比较中断里修改 CCR1 与通道 3 下降沿翻转点（即上文中的 F 点）所对应的比较值,即 CCR3 的值。

3、完成各项配置后，即可生成包含初始化配置代码的工程。

4、在新建的工程里，添加用户代码。代码基于 STM32Cube 库。

4.1 开启 TIM3 CCR1 的预装功能。

```
__HAL_TIM_ENABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_1);
```

4.2 关闭 TIM3 CCR3 的预装功能。

```
__HAL_TIM_DISABLE_OCxPRELOAD(&htim3, TIM_CHANNEL_3);
```

4.3 使能 TIM3 通道 1 的比较输出中断。

```
__HAL_TIM_ENABLE_IT(&htim3, TIM_IT_CC1);
```

4.4 使能 TIM3 通道 1 的比较输出功能。

```
TIM_CCxChannelCmd(TIM3, TIM_CHANNEL_1, TIM_CCx_ENABLE);
```

4.5 使能 TIM3 通道 3 的比较输出及 DMA 传输功能。

```
HAL_TIM_OC_Start_DMA(&htim3, TIM_CHANNEL_3,Data_Pwm, 2);
```

其中，Data_Pwm 是为 DMA 传输定义的一个内存数组名。

初始定义是：uint32_t Data_Pwm[]={584,384};

4.6 启动 TIM3 的计数器工作。

```
__HAL_TIM_ENABLE(&htim3);
```

4.7 在 TIM3 的 OC1 比较中断里修改 CCR1 和通道 3 下降沿翻转点所对应的比较值。

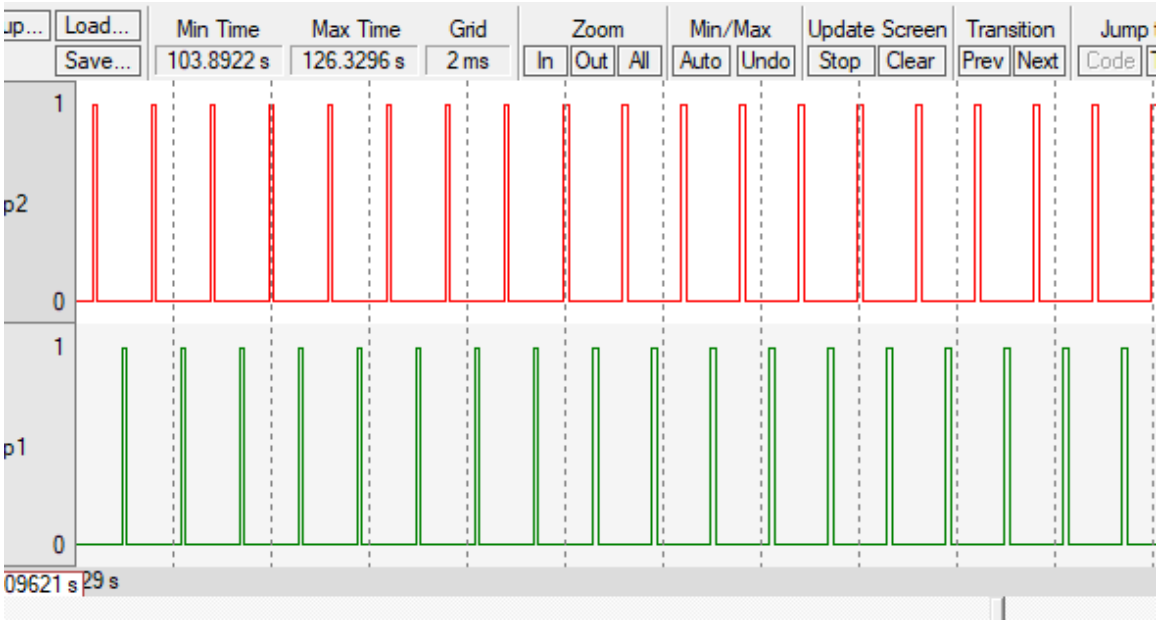
具体代码放在中断里调用的回调函数。

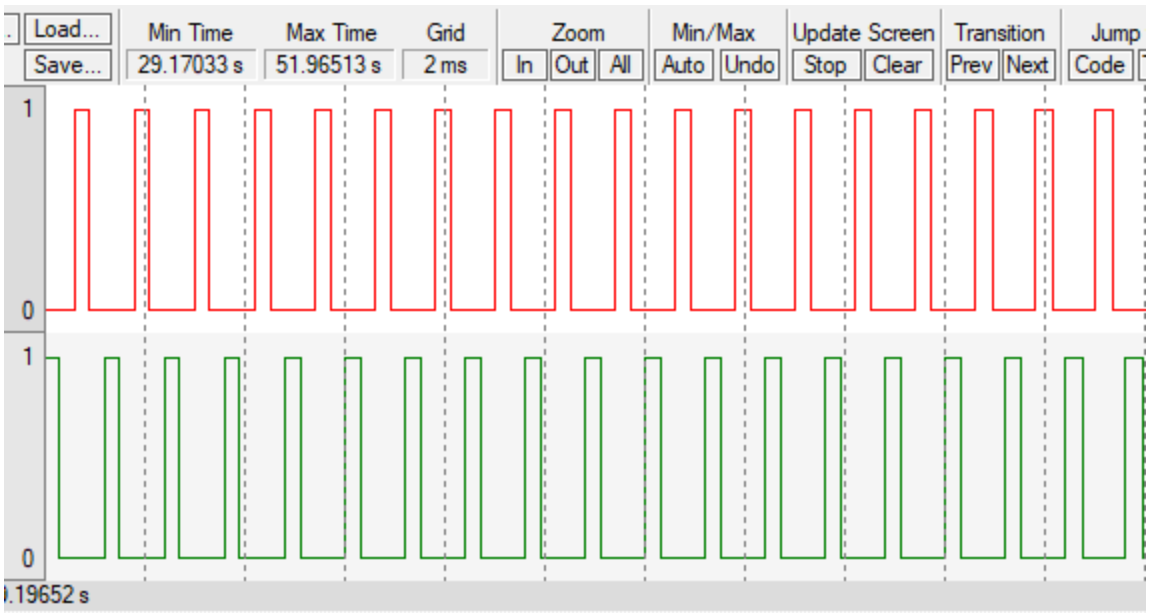
```
void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim)
{
  if ( htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
  {
  if(TIM3->CCR1< 383)
  {
    TIM3->CCR1 = TIM3->CCR1+2;
  }
  else
  {
  TIM3->CCR1 = 0;
  }

  Data_Pwm[0] = 384 + TIM3->CCR1;
 }
}
```

5、编译、运行，验证结果。

我这里借助 MDK-IDE 自带的模拟逻辑分析仪，可以看到预期的结果。两路波形保持固定相差，占空比在 0~50%范围同步变化。

下面是从动态输出过程中截取到的两幅图，供参考。【红色波形是通道 1 使用 PWM 模式实现的，绿色的是通道 3 使用比较切换模式实现的。】





小结下：

利用定时器的比较输出切换模式，结合 DMA 外设，可以灵活地输出各种自定义波形。另外，STM32 常规定时器中的 CCR 寄存器跟 ARR 寄存器一样具备预装功能，什么时候需开启什么时候需要关闭，结合具体应用来正确配置。否则，在更新 ARR/CCR 寄存器内容时，可能输出一些我们不期望的脉冲出来，比方不时产生些类似于干扰的尖而窄的脉冲。

最后，希望通过上面示例能带给大家一些应用上的帮助或启示。