例程八 TIM1

这一节，我们将向大家介绍如何使用 STM8 的定时器中的基本定时功能，STM8 的定时器功能十分强大，有 TIM1 高级定时器，也有 TIM2、TIM3 等通用定时器，还有 TIM4 基本定时器。在 STM8S 参考手册里面，定时器的介绍占了 1/3 的篇幅，足见其重要性。这一节，我们分别介绍 TIM1 的基本定时功能

16 位高级控制定时器(TIM1)

简介：

TIM1 由一个 16 位的自动装载计数器组成，它由一个可编程的预分频器驱动。TIM1 有 4 个通道，分别是 1 到 4。分别对应于四个不同的捕获/比较通道。

高级控制定时器适用于许多不同的用途：

基本的定时

测量输入信号的脉冲宽度(输入捕获)

产生输出波形(输出比较，PWM 和单脉冲模式)

对应与不同事件(捕获，比较，溢出，刹车，触发)的中断

与 TIM5/TIM6 或者外部信号(外部时钟，复位信号，触发和使能信号)同步

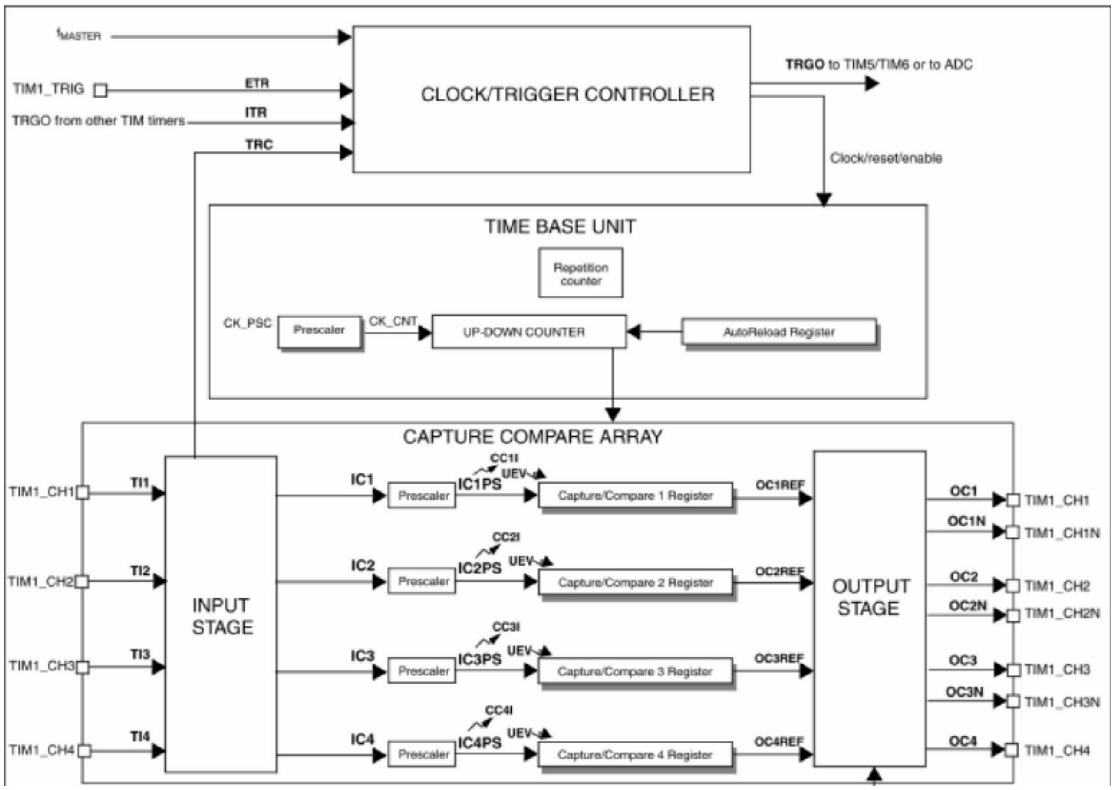
高级控制定时器广泛的适用于各种控制应用中，包括那些需要中间对齐模式 PWM 的应用，该模式支持互补输出和死区时间控制。

高级控制定时器的时钟源可以是内部时钟，也可以是外部的信号，可以通过配置寄存器来进行选择。

TIM1的特性包括：

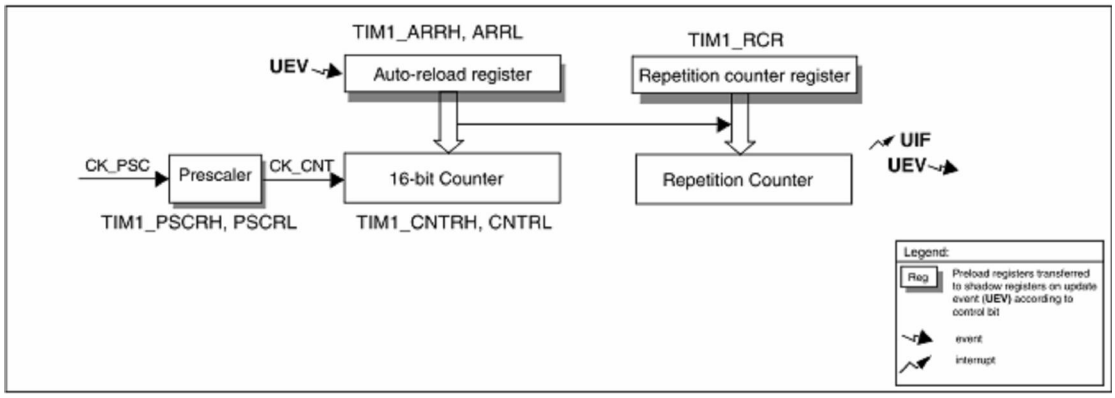
- 16位向上、向下、向上/下自动装载计数器
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 16位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为1～65535之间的任意数值
- 同步电路，用于使用外部信号控制定时器以及定时器互联 (某些型号的芯片没有定时器互联功能)
- 多达4个独立通道可以配置成：
  - 输入捕获
  - 输出比较
  - PWM 生成(边缘或中间对齐模式)
  - 六步 PWM 输出
  - 单脉冲模式输出
  - 三个支持带互补输出，并且死区时间可编程的通道
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 产生中断的事件包括：
  - 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发)
  - 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数)
  - 输入捕获
  - 输出比较
  - 刹车信号输入

图28 TIM1框图



TIM1 的时基单元包括，如下图所示：

- 16 位向上/向下计数器
- 16 位自动重载寄存器
- 重复计数器
- 预分频器



16 位计数器，预分频器，自动重载寄存器和重复计数器寄存器都可以通过软件进行读写操作。

自动重载寄存器由预装载寄存器和影子寄存器组成。

可在在两种模式下写自动重载寄存器：

● 自动预装载已使能(TIM1_CR1 寄存器的 ARPE 位置位)。在此模式下，写入自动重载寄存器的数据将被保存在预装载寄存器中，并在下一个更新事件(UEV)时传送到影子寄存器。

● 自动预装载已禁止(TIM1_CR1 寄存器的 ARPE 位清除)。在此模式下，写入自动重载寄存器的数据将立即写入影子寄存器。

更新事件的产生条件：

●计数器向上或向下溢出。

●软件置位了 TIM1_EGR 寄存器的 UG 位。

●时钟/触发控制器产生了触发事件。

在预装载使能时(ARPE=1)，如果发生了更新事件，预装载寄存器中的数值(TIM1_ARR)将写入影子寄存器中，并且 TIM1_PSCR 寄存器中的值将写入预分频器中。

置位 TIM1_CR1 寄存器的 UDIS 位将禁止更新事件(UEV)。

计数器由预分频器的输出 CK_CNT 驱动，而 CK_CNT 仅在 IM1_CR1 寄存器的计数器使能位(CEN)被置位时才有效。

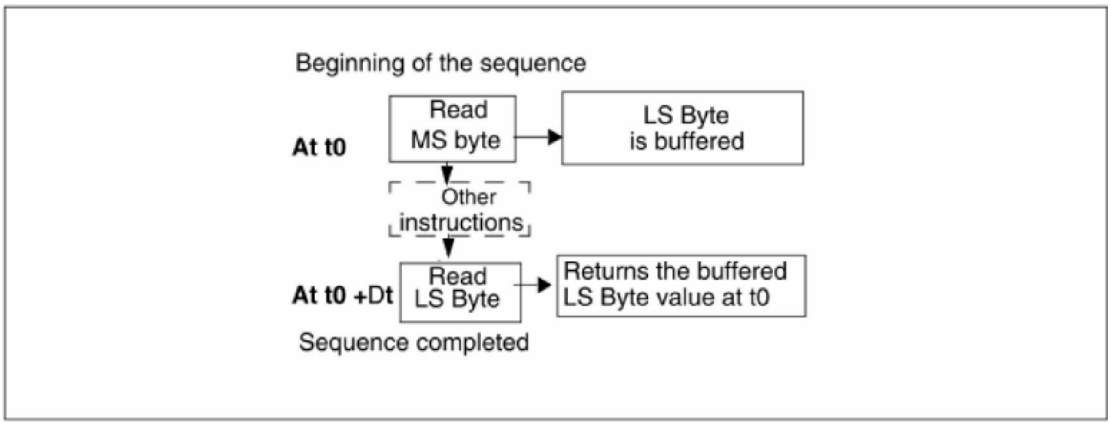
读写 16 位计数器

写计数器的操作没有缓存，并且可以在任何时候写TIM1_CNTRH和TIM1_CNTRL寄存器，因此我们建议不要在计数器运行时写入新的数值，以免写入了错误的数值。

读计数器的操作带有8位的缓存。在用户读了高位(MS)字节后，低位(LS)字节将被自动缓存，缓存的数据在16位的读操作完成之前不会有变化，*图30*解释了这一过程。

注意：不要使用LDW指令来读取16位计数器的值，因为此指令先读低位(LS)字节，这样读出的数值是错误的。

图30 读16位计数器的过程(TIM1_CNTR)



16 位TIM1_ARR寄存器的写操作

预装载寄存器中的值将写入16位的TIM1_ARR寄存器中，此操作由两条指令完成，每条指令写入1个字节，高位(MS)字节是先写入的。

影子寄存器在高位(MS)字节写入时被锁定，并保持到低位(LS)字节写完。不要使用LDW指令，因为此指令先写低位(LS)字节，这将导致写入的数值错误。

预分频器

预分频器的实现：

- TIM1的预分频器基于一个由16位寄存器(TIM1_PSCR)控制的16位计数器。由于这个控制寄存器带有缓冲器，因此它能够在运行时被改变。预分频器可以将计数器的时钟频率按1到65536之间的任意值分频。

  计数器的频率可以由下式计算：

  $f_{CK\_CNT} = f_{CK\_PSC} / (PSCR[15:0] + 1)$

  预分频器的值由预装载寄存器写入，保存了当前使用值的影子寄存器在低位(LS)写入时被载入。

  需两次单独的写操作来写16位寄存器，高位(MS)先写。不要使用先写低位(LS)的LDW指令。

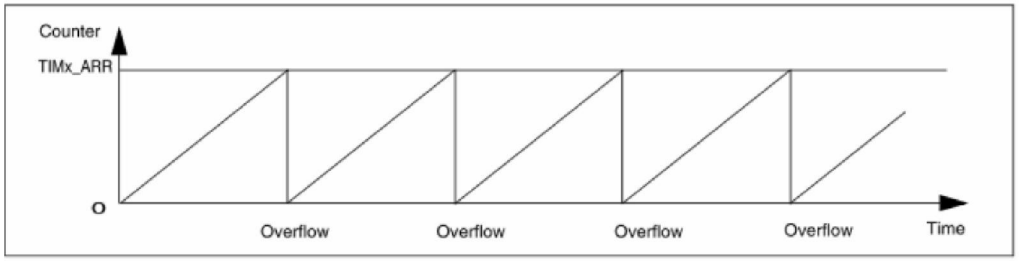
  新的预分频器的值在下一次更新事件到来时被采用。

  对TIM1_PSCR寄存器的读操作通过预装载寄存器完成，因此不需要特别的关注。

向上计数模式

在向上计数模式中，计数器从0计数到用户定义的比较值(TIMx_ARR寄存器的值)，然后重新从0开始计数并产生一个计数器溢出事件，同时，如果TIM1_CR1寄存器的UDIS位是0，将会产生一个更新事件(UEV)。*0*描述了向上计数模式。

图31　向上计数模式的计数器



置位TIMx_EGR寄存器的UG位(通过软件方式或者使用从模式控制器)也同样可以产生一个更新事件。

使用软件置位TIMx_CR1寄存器的UDIS位，可以禁止更新事件，这样可以避免在更新预装载寄存器时更新影子寄存器。在UDIS位被清除之前，将不产生更新事件。但是在应该产生更新事件时，计数器仍会被清0，同时预分频器的计数也被清0(但预分频器的数值不变)。此外，如果设置了TIMx_CR1寄存器中的URS位(选择更新请求)，设置UG位将产生一个更新事件UEV，但硬件不设置UIF标志(即不产生中断请求)。这是为了避免在捕获模式下清除计数器时，同时产生更新和捕获中断。

当发生一个更新事件时，所有的寄存器都被更新，硬件同时(依据URS位)设置更新标志位(TIMx_SR寄存器的UIF位)：

自动装载影子寄存器被重新置入预装载寄存器的值(TIMx_ARR)。

预分频器的缓存器被置入预装载寄存器的值(TIMx_PSC寄存器的内容)。

下图给出一些例子，说明当TIMx_ARR=0x36时，计数器在不同时钟频率下的动作。

*图32*的预分频为2，因此计数器的时钟(CK_CNT)频率是预分频时钟(CK_PSC)频率的一半。

*图32*禁止了自动装载功能(ARPE=0)，所以在计数器达到0x36时，计数器溢出，影子寄存器立刻被更新，同时产生一个更新事件。

图32　　当ARPE=0(ARR不预装载)，预分频为2时的计数器更新。

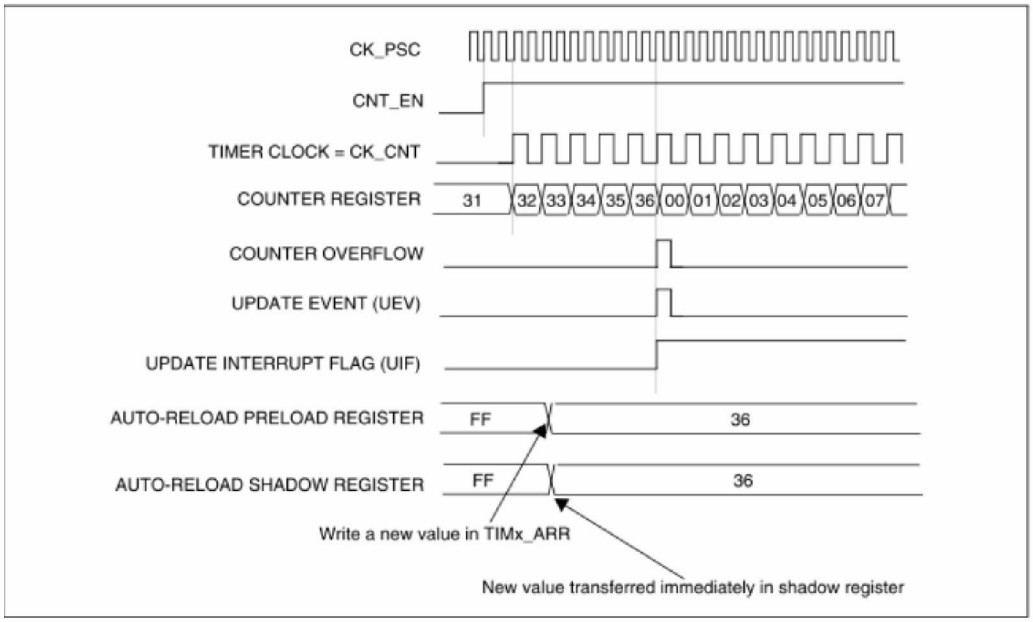


*图33*的预分频为1，因此CK_CNT的频率与CK_PSC一致。

*图33*使能了自动重载(ARPE=1)，所以在计数器达到0xFF产生溢出。0x36将在溢出时被写入，同时产生一个更新事件。

向下计数模式

在向下模式中，计数器从自动装载的值(TIMx_ARR寄存器的值)开始向下计数到0，然后再从自动装载的值重新开始计数，并产生一个计数器向下溢出事件。如果TIM1_CR1寄存器的UDIS位被清除，还会产生一个更新事件(UEV)。*图34*描述了向下计数模式的计数器。

中央对齐模式(向上/向下计数)

在中央对齐模式，计数器从0开始计数到自动加载的值(TIMx_ARR寄存器)-1，产生一个计数器溢出事件，然后向下计数到0并且产生一个计数器下溢事件；然后再从0开始重新计数。

在此模式下，不能写入TIMx_CR1中的DIR方向位。它由硬件更新并指示当前的计数方向。

下图给出一个中央对齐模式的例子。

预分频时钟(CK_PSC)

时基单元的预分频时钟(CK_PSC)可以由以下源提供：

- 内部时钟(f_{MASTER})
- 外部时钟模式1：外部时钟输入(TIx)
- 外部时钟模式2：外部触发输入ETR
- 内部触发输入(ITRx)：使用一个定时器做为另一个定时器的预分频时钟。更多信息请参考*图52*的例子。

简要说明：

CK_PSC 的时钟来源于 f_master，我们使用 16M 内部时钟源 HIS

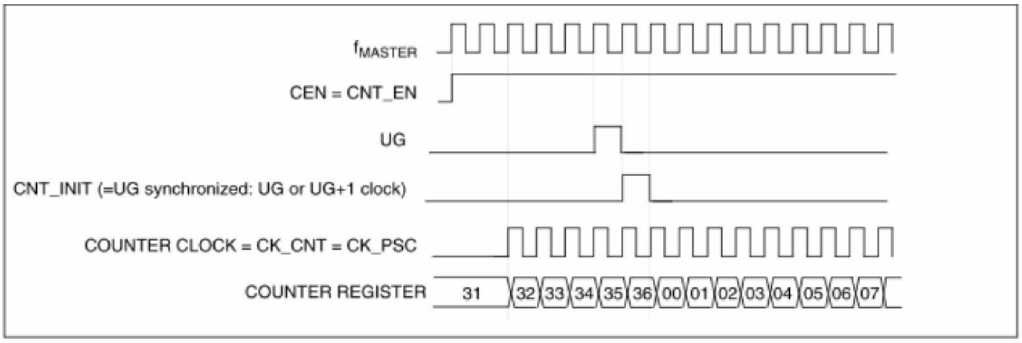
然后可以通过 PSCR 这个寄存器设置 CK_CNT，PSCR 是 2 个 8 位寄存器组成的 16 位寄存器，可以在 0~65535 之间任务分频，分频后的频率提供给 CK_CNT

内部时钟源(f_{MASTER})

如果同时禁止了触发模式控制器和外部触发输入(TIM1_SMCR寄存器的SMS=000，TIM1_ETR寄存器的ECE＝0)，则CEN、DIR和UG位是实际上的控制位，并且只能被软件修改(UG位仍被自动清除)。一旦CEN位被写成1，预分频器的时钟就由内部时钟提供。

下图描述了控制电路和向上计数器在普通模式下，不带预分频器时的操作。

图41　普通模式下的控制电路，f_{MASTER}分频因子为1



TIM1 的功能很强大的，在这里也是分析部分而已，以上的理论部分摘自 STM8 寄存器.pdf 文档，至于其他方面的分析，大家可以参考 STM8 寄存器.pdf 文档。TIM1 也可以做 PWM 功能，由于 STM8 开发板没有连接到这几个管脚的外设，调起程序比较困难，在 TIM1 这个模块里只做定时器的重点分析，只有 PWM 功能 TIM2 里面有很详细的分析，大家可以参考 TIM2 的例程和教程文档，初始话都是差不多的，只要修改一下参数就可以满足大家的要求。下面看看 TIM1 的是怎样初始化当一个定时器用的。

```
int main(void)
{
  /* Infinite loop */

  /*设置内部高速时钟16M为主时钟*/
   CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);

  /*!<Set High speed internal clock */
   LED_Init();
   Tim1_Init();
   SetLedOFF(); /* 让所有灯灭 */
    __enable_interrupt();
  while (1)
  {
    /*  添加你的代码  */


    LED_Display();

  }
}
```

里面最重要的还是 TIM1 的初始化。

```
void Tim1_Init(void)
{
  TIM1_TimeBaseInit(16,TIM1_COUNTERMODE_UP,1000,0);
  TIM1_ARRPreloadConfig(ENABLE);
  TIM1_ITConfig(TIM1_IT_UPDATE , ENABLE);
  TIM1_Cmd(ENABLE);
}
```

这个初始化函数定义了 TIM2 的计数频率为主时钟的 16 分频，就是 1M,当从 0 计数到 999 就是 1ms,但计数到 999 是马上又自动重装系数，又从 0 开始计数，当计数到 999，会产生一个中断。

__enable_interrupt();这个就是开总中断

下面看看中断服务子程序

```
#pragma vector=0xD
__interrupt void TIM1_UPD_OVF_TRG_BRK_IRQHandler(void)
{
  TimingDelay_Decrement();
  TIM1_ClearITPendingBit(TIM1_IT_UPDATE);

}
```

这个 TIM1 计数溢出中断的中断号为 0xD，里面的函数是放中断后要处理的语句，最后要清楚标志位，TIM1_ClearITPendingBit(TIM1_IT_UPDATE);否则就跳不出中断。这里主要是实现 1ms 的延迟函数,所以只调用了 TimingDelay_Decrement();这个函数。看看它的函数原型

```
void TimingDelay_Decrement(void)
{
  if (TimingDelay != 0x00)
     {
        TimingDelay--;
      }
}
```

就是每近一次中断 TimingDelay 就减 1，TimingDelay 定义为静态全局变量，这样的话就可以实现 1ms 精准延迟。下面看看延迟函数的原型

```
void delay_ms( u16 nTime)
{
  TimingDelay = nTime;

  while(TimingDelay != 0);
}
```

整个延迟思想是 TimingDelay 不等于 0 的话，就在这等到 TimingDelay 为 0，TimingDelay 减 1 是进去中断子程序，TimingDelay 是全局静态变量

当大家下载这个例程进去风驰电子 STM8 开发板，就会见到 LED 流起来了。

风驰电子祝您学习愉快～～～!!!!!