STM32F10xxx 正交编码器接口应用笔记

Abstract --- The quadrature encoder which is very popular in the motor servo control application, known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring the number of pulses and the relative phase of the two signals you can track the position, the direction of rotation and speed. In ddition, a third channel, or index signal, can be used to reset the position counter.
STM3210x which is the MCU based on ARM latest core - Cortex-M3, integrates the quadrature encoder interface. Accordingly STM3210x can handles the encoder signal without any CPU overhead which it is possible for CPU to focus on the vector control.

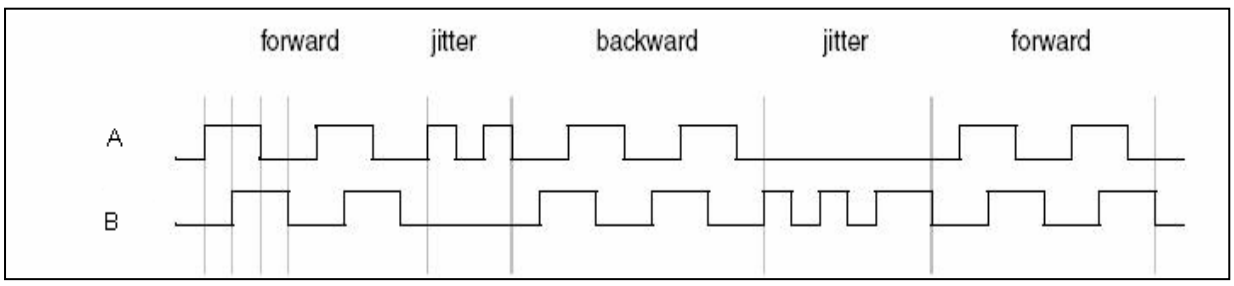
在马达控制类应用中，正交编码器可以反馈马达的转子位置及转速信号。TM32F10x 系列 MCU 集成了正交编码器接口，增量编码器可与 MCU 直接连接而无需外部接口电路。该应用笔记详细介绍了 STM32F10x 与正交编码器的接口，并附有相应的例程，使用户可以很快地掌握其使用方法。

1 正交编码器原理

正交编码器实际上就是光电编码器，分为增量式和绝对式，较其它检测元件有直接输出数字量信号，惯量低，低噪声，高精度，高分辨率，制作简便，成本低等优点。增量式编码器结构简单，制作容易，一般在码盘上刻 A、B、Z 三道均匀分布的刻线。由于其给出的位置信息是增量式的，当应用于伺服领域时需要初始定位。格雷码绝对式编码器一般都做成循环二进制代码，码道道数与二进制位数相同。格雷码绝对式编码器可直接输出转子的绝对位置，不需要测定初始位置。但其工艺复杂、成本高，实现高分辨率、高精度较为困难。

本文主要针对增量式正交编码器，它产生两个方波信号 A 和 B，它们相差+/- 90°，其符号由转动方向决定。如下图所示：

图 1：增量式正交编码器输出信号波形

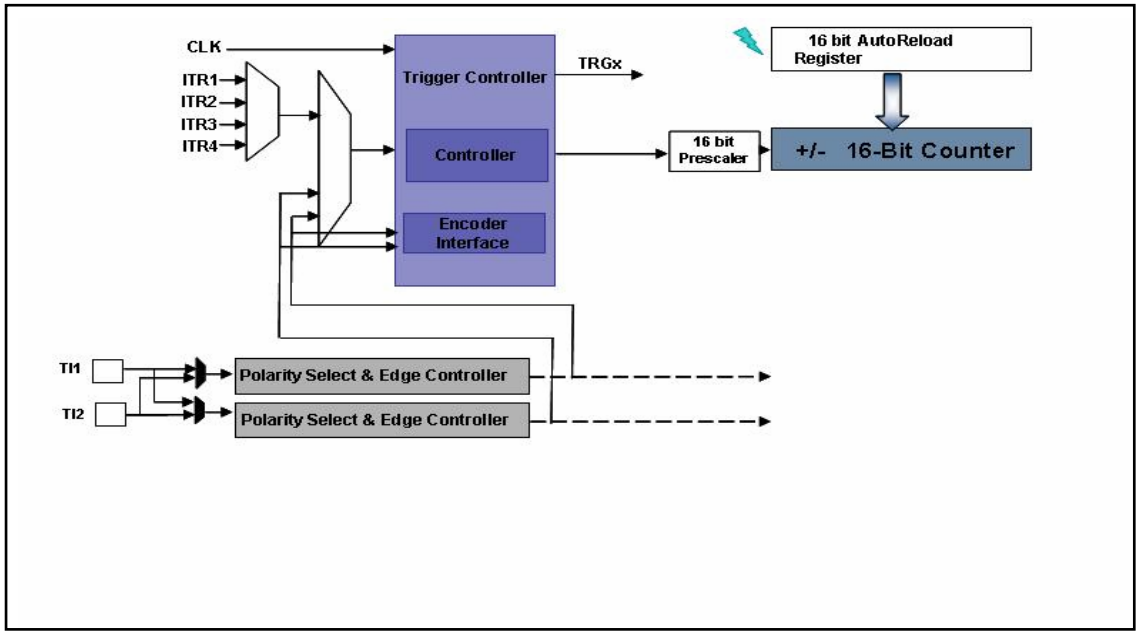


2 STM32F10x 正交编码器接口详述

STM32F10x 的所有通用定时器及高级定时器都集成了正交编码器接口。定时器的两个输入 TI1 和 TI2 直接与增量式正交编码器接口。当定时器设为正交编码器模式时，这两个信号的边沿作为计数器的时钟。而正交编码器的第三个输出（机械零位），可连接外部中断口来触发定时器的计数器复位。

2.1 定时器正交编码器接口框图

图 2：定时器正交编码器接口



2.2 功能描述

选择编码器接口模式的方法是：如果计数器只在 TI2 的边沿计数，则置 TIM1_SMCR 寄存器中的 SMS=001；如果只在 TI1 边沿计数，则置 SMS=010；如果计数器同时在 TI1 和 TI2 边沿计数，则置 SMS=011。

通过设置 TIM1_CCER 寄存器中的 CC1P 和 CC2P 位，可以选择 TI1 和 TI2 极性；如果需要，还可以对输入滤波器编程。

两个输入 TI1 和 TI2 被用来作为增量编码器的接口。参看表 1，假定计数器已经启动(TIM1_CR1 寄存器中的 CEN=1)，则 TI1FP1 或 TI2FP2 上的有效跳变作为计数器的时钟信号。TI1FP1 和 TI2FP2 是 TI1 和 TI2 在通过输入滤波器和极性控制后的信号；如果没有滤波和变相，则 TI1FP1=TI1，TI2FP2=TI2。根据两个输入信号的跳变顺序，产生了计数脉冲和方向信号。依据两个输入信号的跳变顺序，计数器向上或向下计数，因此 TIM1_CR1 寄存器的 DIR 位由硬件进行相应的设置。不管计数器是对 TI1 计数、对 TI2 计数或者同时对 TI1 和 TI2 计数。在任一输入(TI1 或者 TI2)跳变时都会重新计算 DIR 位。

编码器接口模式基本上相当于使用了一个带有方向选择的外部时钟。这意味着计数器只在 0 到 TIM1_ARR 寄存器中自动装载值之间连续计数(根据方向，或是 0 到 ARR 计数，或是 ARR 到 0 计数)。所以在开始计数之前必须配置 TIM1_ARR；同样，捕获器、比较器、预分频器、周期计数器、触发输出特性等仍工作如常。编码模式和外部时钟模式 2 不兼容，因此不能同时操作。

在这个模式下，计数器依照增量编码器的速度和方向被自动的修改，因此，它的内容始终指示着编码器的位置。计数方向与相连的传感器旋转的方向对应。表 1 列出了所有可能的可能的组合，假设 TI1 和 TI2 不同时变换。

表 1：计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

一个外部的增量编码器直接和 MCU 连接不需要外部接口逻辑。但是，一般使用比较器将编码器的差动输出转换到数字信号，这大大增加了抗噪声干扰能力。编码器输出的第三个信号表示机械零点，可以连接到一个外部中断输入，触发一个计数器复位。

图 2 给出一个计数器操作的实例，显示了计数信号的产生和方向控制。它还显示了当选择了双边沿时，输入抖动是如何被抑制的；这种情况可能会在传感器的位置靠近一个方向转换点时发生。在这个例子中，我们假定配置如下：

- CC1S=’01’ (TIMx_CCMR1 寄存器，IC1FP1 映射到 TI1)
- CC2S=’01’ (TIMx_CCMR2 寄存器，IC2FP2 映射到 TI2)
- CC1P=’0’ (TIMx_CCER 寄存器，IC1FP1 不反相，IC1FP1=TI1)
- CC2P=’0’ (TIMx_CCER 寄存器，IC2FP2 不反相，IC2FP2=TI2)
- SMS=’011’ (TIMx_SMCR 寄存器，所有的输入均在上升沿和下降沿有效).
- CEN=’1’ (TIMx_CR1 寄存器，计数器使能)

图 **3** 编码器模式下的计数器操作实例

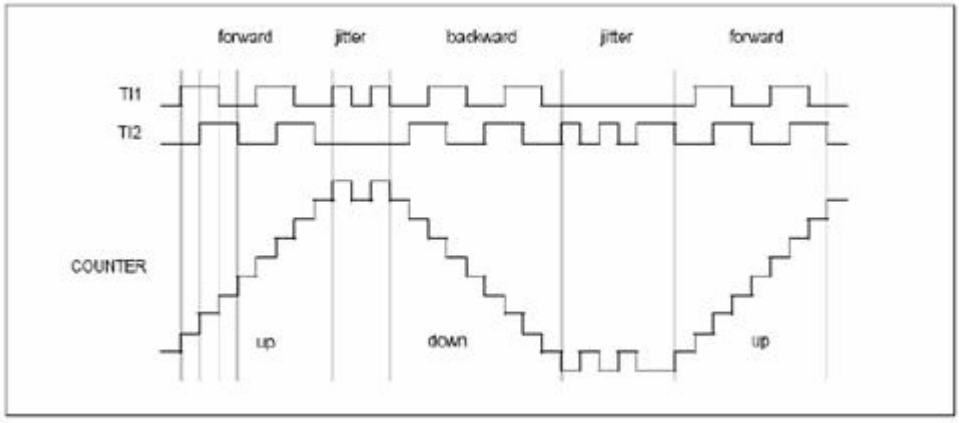
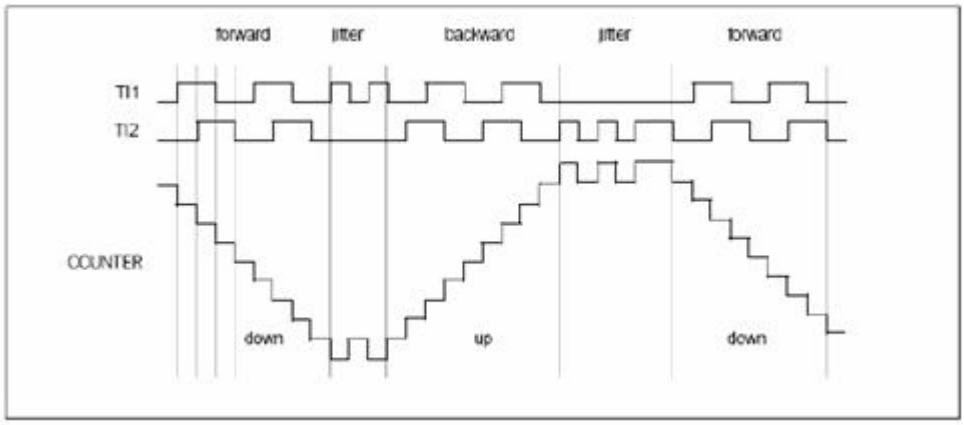


图 3 为当 IC1FP1 极性反相时计数器的操作实例(CC1P='1'，其他配置与上例相同)

图 **4** **IC1FP1** 反相的编码器接口模式实例



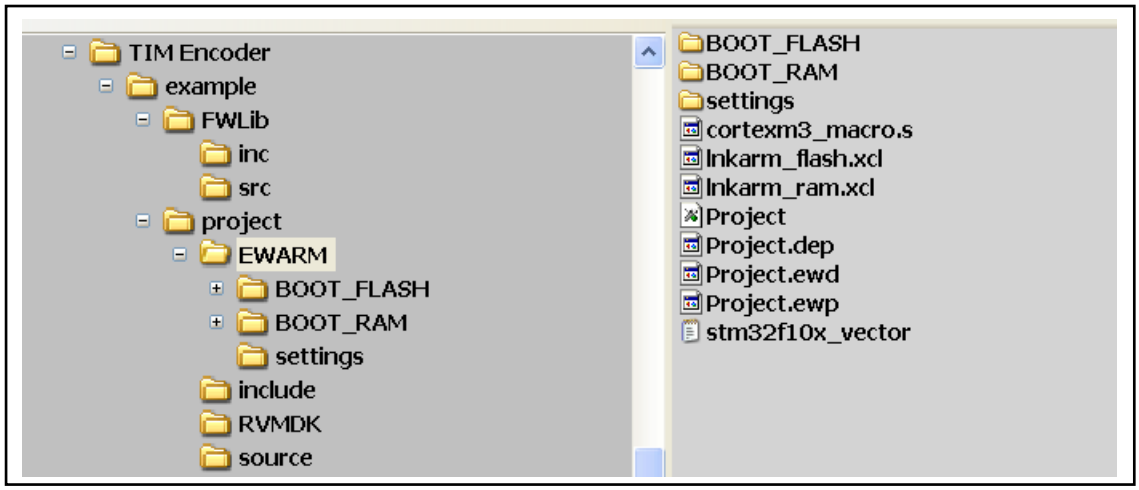
当定时器配置成编码器接口模式时，提供传感器当前位置的信息。如果使用另一个配置在捕获模式的定时器，测量两个编码器事件的间隔，可以获得动态的信息(速度，加速度，减速度)。指示机械零点的编码器输出可被用作此目的。根据两个事件间的间隔，可以按照固定的时间读出计数器值。如果可能的话，你可以把计数器的值锁存到第三个输入捕获寄存器(捕获信号必须是周期的并且可以由另一个定时器产生)。它也可以通过一个由实时时钟产生的 DMA 请求来读取它的值。

3 例程

为了使用户更快捷地掌握和使用 STM32F10x 的正交编码器接口，该应用笔记提供了一个应用例程。该例程基于学习板 EK_STM32F，开发环境为 IAR EWARM 4.42A。

3.1 文件目录

例程的文件目录如下图所示。在安装完 IAR EWARM 4.42A 后，直接双击以下目录 \example\project\EWARM\下的 project Workspace 文件，打开该例程。



在上图中：

- FWLib 目录包含了 STM32F10x 的标准库文件；
- Project 目录中，source 目录包含了例程的源文件，而头文件在 include 目录中。

3.2 功能描述

正交编码器的输出信号 A 和 B 分别连接 MCU 的 PA6 和 PA7 管脚，这两个管脚复用为 TIM3_CH1 及 TIM3_CH2。LCD 可用于显示三类信息，分别为：TIM3 的计数器值、马达转子的电角度值（即马达转子的位置，取值范围为-180 度 ~ +180 度，其单位为 1 度）及马达转动的电频率（其单位为 0.1Hz）。由于学习板 EK_STM32F 的 LCD 一次只能显示四个数值，因此该例程使用了 KEY2 键来切换 LCD 的显示内容。

3.3 例程函数描述

该例程共有五个源文件，分别为：

- main.c：主程序
- lcd.c：LCD 驱动程序
- stm32f10x_encoder.c：正交编码器的接口驱动程序
- stm32f10x_it.c：包含了定时器 2 的中断程序
- stm32f10x_Timebase.c：SysTick 的时基程序

每个源文件包含的函数如下表所示。

表 2 main.c 包含的函数

函数名	函数功能描述
RCC_Configuration	配置系统的时钟
GPIO_Configuration	配置 GPIO
NVIC_Configuration	配置 NVIC
LcdShow_Init	初始化 LCD 驱动，Timer2 驱动 LCD
KEYS_Init	键盘接口初始化
KEYS_Read	键值的读取

表 3 lcd.c 包含的函数

函数名	函数功能描述
Convert	ASCII 码字符转换成 LCD 的驱动缓存值
write_char	在 LCD 上显示一个字符
write_string	在 LCD 上显示一个字符串
int2char	无符号整数值转换成一个字符串

表 4 stm32f10x_encoder.c 包含的函数

函数名	函数功能描述
ENC_Init	正交编码器接口的初始化，Timer3 驱动编码器
ENC_Get_Electrical_Angle	计算马达转子的位置电角度
ENC_Clear_Speed_Buffer	初始化马达速度缓存
ENC_Calc_Rot_Speed	计算马达转动的电频率
ENC_Calc_Average_Speed	计算马达转动的平均电频率
LCD_Display	LCD 显示管理
TIM3_IRQHandler	Timer3 的 Update 中断处理程序

表 5 stm32f10x_it.c 包含的函数

函数名	函数功能描述
TIM2_IRQHandler	Timer2 的 OC 中断处理程序，实现 LCD 的动态驱动

表 6 stm32f10x_Timebase.c 包含的函数

函数名	函数功能描述
TB_Init	时基定时器（SysTick）的初始化
TB_Set_DisplayDelay_500us	LCD 显示刷新的时基配置
TB_DisplayDelay_IsElapsed	判断 LCD 显示刷新的时基是否已经结束
SysTickHandler	SysTick 的中断处理程序

3.4 例程性能分析

增量编码器由 TM32F10x 的定时器硬件处理，查看该例程，我们可以发现，定时器有一个 Update 中断。该中断响应定时器的计数器溢出事件。其中断处理程序如下：

```
void TIM3_IRQHandler(void)
{
  /* Clear the interrupt pending flag */
  TIM_ClearFlag(ENCODER_TIMER, TIM_FLAG_Update);

  if (hEncoder_Timer_Overflow != U16_MAX)
  {
    hEncoder_Timer_Overflow++;
  }
}
```

上述函数是 CPU 唯一要处理的，它只有几条语句，因此几乎不需要 CPU 花多少时间去处理它。

另外，在例程中，我们假设编码器的分辨率为 400 pulse/rev，因此，它反馈的马达转子的机械角度的精度为：

$360^{o}/(4*400) = 0.225^{o}$

很明显，编码器的分辨率越高，马达转子的机械角度的精度也越高。

4 小结

综上所述，STM32F10x 的所有通用定时器及高级定时器都集成了正交编码器接口，它与正交编码器的连接非常便捷，其配置非常灵活：

- 可编程的计数率
  - x4：标准模式，所有边沿有效
    - 1000 线的正交编码器每转产生 4000 个计数脉冲
  - x2：只对 A (或 B)计数，但仍可确定方向
  - “转速模式”:正交编码器输入可分频
- 当自动重载寄存器的值配置为正交编码器每转产生的计数脉冲时，则计数器的值直接为马达转子的角度/位置
- 编码器每转一周可发出一个或多个中断
  - 一个，每 360°；
  - 多个，每 60°，90°,...（依赖于自动重载寄存器的配置）