Technology Innovator
Puya

AN1125
应用笔记

使用 PY32F005
通用定时器方波捕获应用注意事项

前言

PY32F005 系列微控制器通用定时器由可编程预分频器驱动的 16 位自动装载计数器构成。它适用于多种场合，包括测量输入信号的脉冲长度(输入捕获)或者产生输出波形(输出比较和 PWM)。使用定时器预分频器和 RCC 时钟控制器预分频器，脉冲长度和波形周期可以在几个微秒到几个毫秒间调整。

本应用笔记将帮助用户了解 PY32F005 通用定时器模块侦测方波方法，并快速着手应用。

表 1. 适用产品

类型	产品系列
微型控制器系列	PY32F005

目录

1 定时器捕获方波应用注意事项

1.1 应用方案说明

本方案采用：

- 通用定时器（TIM13/TIM14）单通道输入捕获方波
- 高级定时器 TIM1 周期性中断轮询
- 主循环计算 Duty 与 Frequency

适用于：

- 单路 PWM 信号测量
- 占空比范围：5% ~ 95%
- 中低频信号测量：约 1kHz ~ 10kHz

2 通用定时器配置(TIM13/TIM14)

2.1 配置要点

- GPIO 复用为 TIMx_CH1
- 输入捕获滤波 IC1F
- 单通道双边沿捕获
- 16bit 自动重装
- 计数频率 = 48MHz

代码示例

```
void App_GPIO_Config(void)
{
    RCC->IOPENR          |=   (RCC_IOPENR_GPIOAEN);
    GPIOA->MODER     &= ~(|GPIO_MODER_MODE3_Msk);
    GPIOA->MODER       |= (2<<GPIO_MODER_MODE3_Pos);
    GPIOA->AFR           |= (5<<GPIO_AFR_AFSEL3_Pos);        //PA3->TIM13_Ch1
}
void App_TIM13_Config(void)
{
     RCC->APBENR2 |= (RCC_APBENR2_TIM13EN);
      TIM13->CR1        = 0;
      TIM13->CR1        |= (0<<TIM_CR1_CKD_Pos);
      TIM13->CCMR1    = 0;
      TIM13->CCMR1    |= (3<<TIM_CCMR1_IC1F_Pos);   // 滤波
      TIM13->CCMR1    |= (0<<TIM_CCMR1_IC1PSC_Pos);
      TIM13->CCMR1    |= (1<<TIM_CCMR1_CC1S_Pos); // IC1 映射到 TI1
      TIM13->ARR        = 0xFFFF;
      TIM13->PSC        = 1-1;
      TIM13->TISEL       = 0;
```

```
    TIM13->CCER   |= (1<<TIM_CCER_CC1NP_Pos);//配置下降沿捕获
    TIM13->CCER   |= (1<<TIM_CCER_CC1P_Pos);//配置上升沿捕获
    TIM13->CCER   |= (1<<TIM_CCER_CC1E_Pos);//捕获使能
    TIM13->CR1    |= (TIM_CR1_CEN);//计数使能

}
```

3 高级定时器配置(TIM1)

3.1 设计目的

TIM1 用于：

- 固定周期（20kHz）中断
- 轮询 TIM13->SR
- 调用捕获解析函数

TIM1 不参与捕获，仅用于调度。

3.2 配置说明

- 中央对齐模式
- 20kHz 更新中断
- PSC = 0
- ARR = SystemCoreClock / (2 * 20000)

代码示例

```
void App_TIM1_Config(void)
{
    /* Configure TIM1 */
  LL_TIM_InitTypeDef TIM1CountInit = {0};

  /* Enable TIM1 clock */
  LL_APB1_GRP2_EnableClock(LL_APB1_GRP2_PERIPH_TIM1);

  TIM1CountInit.ClockDivision       =  LL_TIM_CLOCKDIVISION_DIV1;     /*  tDTS=tCK_IN
*/
  TIM1CountInit.CounterMode         =        LL_TIM_COUNTERMODE_CENTER_UP_DOWN;
TIM1CountInit.Prescaler             = 1-1;
  TIM1CountInit.Autoreload          = SystemCoreClock /(2*20000) + 1;
  TIM1CountInit.RepetitionCounter   = 0;                /* Repetition counter value：0       */
  /* Initialize TIM1*/
  LL_TIM_Init(TIM1,&TIM1CountInit);

  /* Clear the update interrupt flag (UIF).  */
  LL_TIM_ClearFlag_UPDATE(TIM1);

  /* Enable update interrupt (UIE) */
  LL_TIM_EnableIT_UPDATE(TIM1);

  /* Enable UPDATE interrupt */
  NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
  NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn,0);

  /* Enable TIM1 */
  LL_TIM_EnableCounter(TIM1);

}
```

4 ISR 中断处理逻辑

4.1 捕获解析函数说明

功能:

- 读取 TIM13->SR
- 处理 CC1OF 溢出
- 读取 CCR1
- 判断上升/下降沿
- 计算周期与 Ton
- 设置 duty_ready 标志

代码示例

```
void App_TIM13_DutyDecode(void)
{
    uint32_t sr = TIM13->SR;

    if (sr & TIM_SR_CC1OF)
    {
        TIM13->SR &= ~TIM_SR_CC1OF;
        gsUser.u16LastUpEdgeCaptureValue = 0;
        return;
    }

     gsUser.u16CaptureValue = TIM13->CCR1;

    /* --------  下降沿  -------- */
    if (sr & TIM_SR_IC1IF)
    {
        TIM13->SR &= ~TIM_SR_IC1IF;
        gsUser.u16DownEdgeCaptureValue = gsUser.u16CaptureValue;
    }

    /* --------  上升沿  -------- */
    if (sr & TIM_SR_IC1IR)
    {
        TIM13->SR &= ~TIM_SR_IC1IR;

        gsUser.u16LastUpEdgeCaptureValue= gsUser.u16UpEdgeCaptureValue;
        gsUser.u16UpEdgeCaptureValue =gsUser.u16CaptureValue;
        if (gsUser.u16LastUpEdgeCaptureValue != 0)
        {
            uint16_t period = (uint16_t)(gsUser.u16UpEdgeCaptureValue -
gsUser.u16LastUpEdgeCaptureValue);
            uint16_t ton = (uint16_t)(gsUser.u16DownEdgeCaptureValue -
gsUser.u16LastUpEdgeCaptureValue);

             /* 关键判断：下降沿必须落在当前周期内 */
             if (ton < period && period > 500)
```

```
            {
                gsUser.u16PeriodCaptureValue = period;
                gsUser.u16CaptureTon = ton;
                duty_ready = 1;
            }
            /* 如果 ton >= period，直接丢弃这个周期 */
        }
    }
}
```

5 Main Loop 解码方波

5.1 计算流程

- duty_ready == 1
- 一次性读取 Period 与 Ton
- 立即清零标志
- 进行整数计算

代码示例

```
void App_CaptureDutyCalc(void)
{
    if (duty_ready)
    {
        duty_ready = 0;

        gsUser.u32CaptureDuty = ((uint32_t)gsUser.u16CaptureTon * 1000) / gsUser.u16PeriodCaptureValue;
        gsUser.u32CaptureFreq = (uint32_t)((48000000UL) / gsUser.u16PeriodCaptureValue);
        gsUser.u32CaptureDuty_Acc = gsUser.u32CaptureDuty + gsUser.u32CaptureDuty_Acc;
        gsUser.u32CaptureFreq_Acc = gsUser.u32CaptureFreq + gsUser.u32CaptureFreq_Acc ;
        u16FilterCounter++;
        if(u16FilterCounter>=128)
        {
            u16FilterCounter = 0;
            gsUser.u32CaptureDuty_Filt = gsUser.u32CaptureDuty_Acc>>7;
            gsUser.u32CaptureFreq_Filt = gsUser.u32CaptureFreq_Acc>>7;
            gsUser.u32CaptureDuty_Acc  = 0;
            gsUser.u32CaptureFreq_Acc  = 0;

        }
}
```

6 结论

本应用在中低频 PWM 测量场景下可稳定运行，通过状态机与溢出处理机制，避免跨周期计算错误。

7 版本历史

版本	日期	更新记录
V1.0	2026.03.04	初版

Puya Semiconductor Co., Ltd.