ARC

TIM2

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-3-6

目录

第8章 通用定时器 TIM2

8.1 通用定时器简介

通用定时器是由一个通过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。每个定时器都是完全独立的，没有互相共享任何资源。

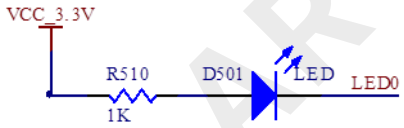
8.2 通用定时器应用实例 ----- TIM2 输出 PWM

8.2.1 实例描述

本实例通过配置 PA1 为 TIM2_CH2，使其输出 PWM 波形，用来驱动连接到 PA1 上的 LED0，通过控制占空比，能够调整 LED0 的亮度，由以下硬件可以看出，占空比越大，LED0 越暗。

8.2.2 硬件设计

LED0	PA1



8.2.3 软件设计

本实例首先使能对应的时钟，包括 TIM2,GPIOA 和 APB2 复用功能时钟。然后配置 PA1 为复用功能推挽输出，最大时钟频率为 50MHz。

通用计数器 TIM2 被配置为时钟频率为 1KHz，占空比为 80%的 PWM 输出，其计算公式如下。

先设置预分频，得到 TIM2 计数器的时钟，该实例中为 1MHz，

Prescaler = (TIM2CLK / TIM2 counter clock) – 1

然后设置 TIM2 的频率，该实例中为 1KHz：

TIM2 Frequency = TIM2 counter clock / (ARR + 1)
= 1MHz / (999 + 1)
= 1KHz

最后设置 TIM2 的占空比：

TIM3 Channel2 duty cycle = TIM2_CCR2 / TIM2_ARR
= 800 / 999
= 80%

该实例的主要代码如下：

文件 TIM_PWM_main.c：

```
/**
  * @brief   Main program, configure TIM2 as PWM output, to drive LED0.
  * @param   None
  * @retval None
  */
int main(void)
{
    ARC_TIM_PWM_Init();
    /* TIM2 enable counter */
    TIM_Cmd(TIM2, ENABLE);

    while (1)
    {
    }
}
```

文件 ARC_TIM_PWM.c

```
/**
  * @brief   Initialize TIM PWM parameters.
  * @param   None
  * @retval None
  */
void ARC_TIM_PWM_PARAM_Init()
{
    /* ------------------------------------------------------------------
    TIM2 Configuration: generate 1 PWM signals:
    The TIM2CLK frequency is set to SystemCoreClock (72MHz), to get TIM2
counter clock at 1 MHz the Prescaler is computed as following:
    - Prescaler = (TIM2CLK / TIM2 counter clock) - 1
    The TIM2 is running at 1 KHz:
    TIM2 Frequency = TIM2 counter clock/(ARR + 1)
                   = 1 MHz / (999 + 1) = 1 KHz

    TIM3 Channel2 duty cycle = TIM2_CCR2 / TIM2_ARR = 800 / 999 = 80%
    ------------------------------------------------------------------ */
    uint16_t PrescalerValue = 0;
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
```

```
    TIM_OCInitTypeDef           TIM_OCInitStructure;

    /* Compute the prescaler value */
    PrescalerValue = (uint16_t) (SystemCoreClock / 1000000) - 1;
    /* Time base configuration */
    TIM_TimeBaseStructure.TIM_Period = 999;
    TIM_TimeBaseStructure.TIM_Prescaler = PrescalerValue;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

    /* PWM1 Mode configuration: Channel2 */
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = 800;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;

    TIM_OC2Init(TIM2, &TIM_OCInitStructure);

    TIM_OC2PreloadConfig(TIM2, TIM_OCPreload_Enable);

    TIM_ARRPreloadConfig(TIM2, ENABLE);
}

/**
  * @brief  Initialize TIM PWM.
  * @param  None
  * @retval None
  */
void ARC_TIM_PWM_Init()
{
    ARC_TIM_PWM_RCC_Init();
    ARC_TIM_PWM_GPIO_Init();
    ARC_TIM_PWM_PARAM_Init();
}
```

文件 ARC_RCC.c

```
/**
  * @brief  Configures TIM PWM clocks.
  * @param  None
  * @retval None
  */
void ARC_TIM_PWM_RCC_Init(void)
```

```
{
    /* TIM3 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

    /* GPIOA clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |
RCC_APB2Periph_AFIO, ENABLE);
}
```

文件 ARC_GPIO.c

```
/**
  * @brief   Configures TIM PWM GPIO ports.
  * @param   None
  * @retval None
  */

/*
          ------------------
          | TIM2 CH2 | PA1 |
          ------------------
*/

void ARC_TIM_PWM_GPIO_Init()
{
    GPIO_InitTypeDef GPIO_InitStructure;

   /* GPIOA.1 Configuration:TIM2 Channel 2 as alternate function push-pull */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```