YC31xx TIMER 应用说明

V1.0

Yichip Microelectronics

Revision History

Version	Date	Author	Description
V1.0	2020-2-25	Dengzhiqian	Initial version

Confidentiality Level:

confidential

目录

1.文档说明

1.1 编写目的

为使用 Timer 相关 demo 及 demo 中相关 API 提供指南

1.2 适用范围

31xx 系列芯片

1.3 文件说明

Timer Demo 路径为 ModuleDemo\Timer

Timer 库文件为如下图 yc_timer.c 与 yc_timer.h,路径为 Librarier\sdk

2. 结构体及枚举说明

2.1. 计时器模式枚举定义（TIM_ModeTypeDef）

元素名称	说明
TIM_Mode_PWM	PWM 模式
TIM_Mode_TIMER	定时器模式

2.2. 计时器编号枚举定义（TIM_NumTypeDef）

元素名称	说明
TIM0	计时器 0

TIM1	计时器 1
TIM2	计时器 2
TIM3	计时器 3
TIM4	计时器 4
TIM5	计时器 5
TIM6	计时器 6
TIM7	计时器 7
TIM8	计时器 8

2.3. PWM 初始化结构体（PWM_InitTypeDef）

元素名称	类型	说明
TIMx	TIM_NumTypeDef	计时器编号
LowLevelPeriod	uint32_t	低电平时间重装载值
HighLevelPeriod	uint32_t	高电平时间重装载值
SatrtLevel	GPIO_OutputTypeDef	初始电平

2.4. 计时器初始化结构体（TIM_InitTypeDef）

元素名称	类型	说明
TIMx	TIM_NumTypeDef	计时器编号
period	uint32_t	计时器重装载值

3. 函数说明

3.1.delay_us

函数原型：void delay_us(int us)；说明：
以微秒为单位的阻塞延迟函数。

参数	方向	说明
int us	IN	延迟时间，微秒单位

返回值	说明
None	None

3.2. delay_ms

函数原型：void delay_ms(int ms);
说明：　　以毫秒为单位的阻塞延迟函数。

参数	方向	说明
int ms	IN	延迟时间，毫秒单位

返回值	说明
None	None

3.3. TIM_Init

函数原型：void TIM_Init(TIM_InitTypeDef* TIM_init_struct)；说明：

计时器初始化函数。

参数	方向	说明

TIM_InitTypeDef* TIM_init_struct	IN	参考计时器初始化结构体说明

返回值	说明
None	None

3.4. TIM_DeInit

函数原型: void TIM_DeInit(void); 说明: 去计时器初始化函数。

参数	方向	说明
None		None

返回值	说明
None	None

3.5. TIM_Cmd

函数原型: void TIM_Cmd(TIM_NumTypeDef TIMx, FunctionalState NewState) 说明: 计时器开启或关闭函数。

参数	方向	说明
TIM_NumTypeDef TIMx	IN	计时器编号（TIM0~TIM5）
FunctionalState	IN	计时器状态:
NewState		1: 使能 0: 不使能

返回值	说明
None	None

3.6. TIM_SetPeriod

函数原型：void TIM_SetPeriod(TIM_NumTypeDef TIMx, uint32_t Period)；说明：

配置计时器周期函数。

参数	方向	说明
TIM_NumTypeDef TIMx	IN	计时器编号（TIM0~TIM5）
uint32_t Period	IN	计时器重载值

返回值	说明
None	None

3.7. TIM_ModeConfig

函数原型：void TIM_ModeConfig(TIM_NumTypeDef TIMx, TIM_ModeTypeDef TIM_Mode)；说明：

配置计时器模式函数。

参数	方向	说明
TIM_NumTypeDef TIMx	IN	计时器编号（TIM0~TIM5）
TIM_ModeTypeDef TIM_Mode	IN	计时器模式（参考 TIM_ModeTypeDef 枚举说明）

返回值	说明
None	None

3.8. TIM_PWMInit

函数原型：void TIM_PWMInit(PWM_InitTypeDef* PWM_init_struct)；说明：

PWM 初始化函数。

参数	方向	说明
PWM_InitTypeDef* PWM_init_struct	IN	参考 PWM 初始化结构体

返回值	说明
None	None

3.9. TIM_SetPWMPeriod

函数原型：void TIM_SetPWMPeriod(TIM_NumTypeDef TIMx,
uint32_t LowLevelPeriod,
uint32_t HighLevelPeriod);

说明： 配置 PWM 周期函数。

参数	方向	说明
TIM_NumTypeDef TIMx	IN	计时器编号（TIM0~TIM5）
uint32_t LowLevelPeriod	IN	低电平周期
uint32_t HighLevelPeriod	IN	高电平周期

返回值	说明
None	None

3.10. TIM_PWMDifferential

函数原型：void TIM_PWMDifferential(TIM_NumTypeDef TIMx,
TIM_NumTypeDef TIMy,
uint32_t LowLevelPeriod, uint32_t
HighLevelPeriod);

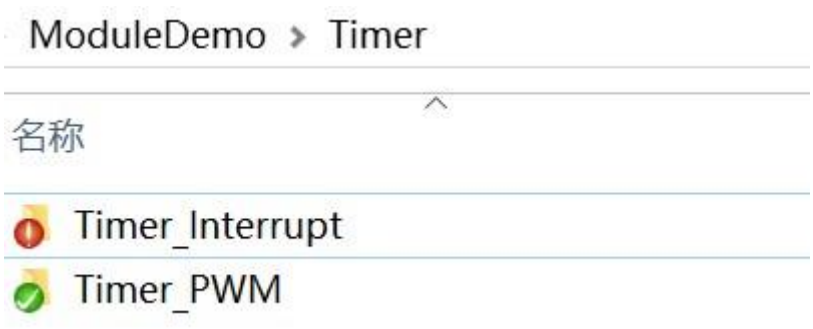
说明： PWM 差分模式配置函数。

参数	方向	说明
TIM_NumTypeDef TIMx	IN	计时器编号（TIM0~TIM5）
TIM_NumTypeDef TIMy	IN	计时器编号（TIM0~TIM5）
uint32_t LowLevelPeriod	IN	低电平周期
uint32_t HighLevelPeriod	IN	高电平周期

返回值	说明
None	None

4. 示例代码及说明

示例代码存放在 ModuleDemo\Timer 目录下（如下图）



① Timer_Interrupt ： 配置为 1ms 进一次中断示例

② Timer_PWM ： 配置输出 5KHz PWM 示例

4.1. Timer_Interrupt 部分示例代码

```
int main(void)
{
    NVIC_Configuration(); //中断配置，使能各定时器中断
    UART_Configuration(); //串口配置
    TIMER_Configuration(); // 计时器配置

    MyPrintf("YiChip Yc3121 Timer Demo V1.0.\r\n");

    while (1)
    {
        TIMM0_Mdelay(TIM0, 1000); // 定时器 0，设置进 1000 次中断
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM0, TIMM0_GetTick(TIM0));
        TIMM0_Mdelay(TIM1, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM1, TIMM0_GetTick(TIM1));
        TIMM0_Mdelay(TIM2, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM2, TIMM0_GetTick(TIM2));
        TIMM0_Mdelay(TIM3, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM3, TIMM0_GetTick(TIM3));
        TIMM0_Mdelay(TIM4, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM4, TIMM0_GetTick(TIM4));
TIMM0_Mdelay(TIM5, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM5, TIMM0_GetTick(TIM5));
        TIMM0_Mdelay(TIM6, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM6, TIMM0_GetTick(TIM6));
        TIMM0_Mdelay(TIM7, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM7, TIMM0_GetTick(TIM7));
        TIMM0_Mdelay(TIM8, 1000);
        MyPrintf("current Timer_%d_GetTick tick = %d \n", TIM8, TIMM0_GetTick(TIM8));
    }
```

```
}
uint32_t TIMM0_GetTick(uint32_t TIMx) //获取某个定时器进中断次数
{
    return tick_Timer[TIMx];
}

void TIMM0_Mdelay(uint32_t TIMx, uint32_t delay)
{
    uint32_t tick = tick_Timer[TIMx];
    TIM_Cmd((TIM_NumTypeDef)TIMx, ENABLE); //  开定时器
    if((tick + delay) < tick_Timer[TIMx])
    {
        while((tick_Timer[TIMx] - delay) < tick);
    }
    else
    {
        while((tick + delay) > tick_Timer[TIMx]);
    }
  TIM_Cmd((TIM_NumTypeDef)TIMx, DISABLE); //  关定时器
}
```

4.2. Timer_PWM 部分示例代码

```
void PWM_Configuration(void)
{
    PWM_InitTypeDef PWM_init_struct;

    PWM_init_struct.TIMx = TIM0;
    PWM_init_struct.LowLevelPeriod = 4800; //设置低电平周期
    PWM_init_struct.HighLevelPeriod = 4800; //  设置高电平周期
    PWM_init_struct.SatrtLevel = OutputLow; //  设置初始电平
    GPIO_Config(GPIOA, GPIO_Pin_11, GPCFG_PWM_OUT0);
    TIM_PWMInit(&PWM_init_struct);

    /* Configure PWM for counting mode */



    TIM_ModeConfig(TIM0, TIM_Mode_PWM);

    /* The last step must be enabled */
```

```
    TIM_Cmd(TIM0, ENABLE);
}
```