初级挑战任务2——裸机下驱动舵机转动90°

题目

驱动舵机转动并在OLED屏幕上显示当前转动角度

- 输出PWM由哪几部分构成？
- ARR/CNT是什么？时基单元的工作流程是什么？（信号数据，内部处理，传出信号）
- 如何依据CCR/CNT/ARR信号输出PWM?

> *下文中粗体是对问题的回答*

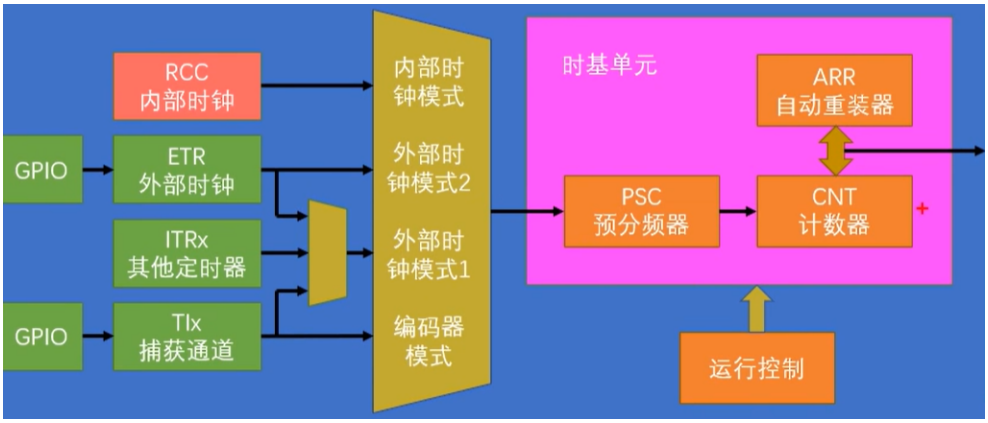
知识笔记

1. PWM （Pulse Width Modulation）脉冲宽度调制：调制脉冲宽度，通过快速变化用数字量等效输出模拟量

2. PWM参数：频率 = 1 / （Ton+Toff），占空比 = Ton / （Ton+Toff），分辨率 = 占空比变化步距，分辨率越大，波形变化越细腻。

占空比越大，模拟电压更趋近于高电平，占空比越低，模拟电压更趋近于低电平。

3. 与配置输出PWM相关的定时器知识
定时器的简化结构图：



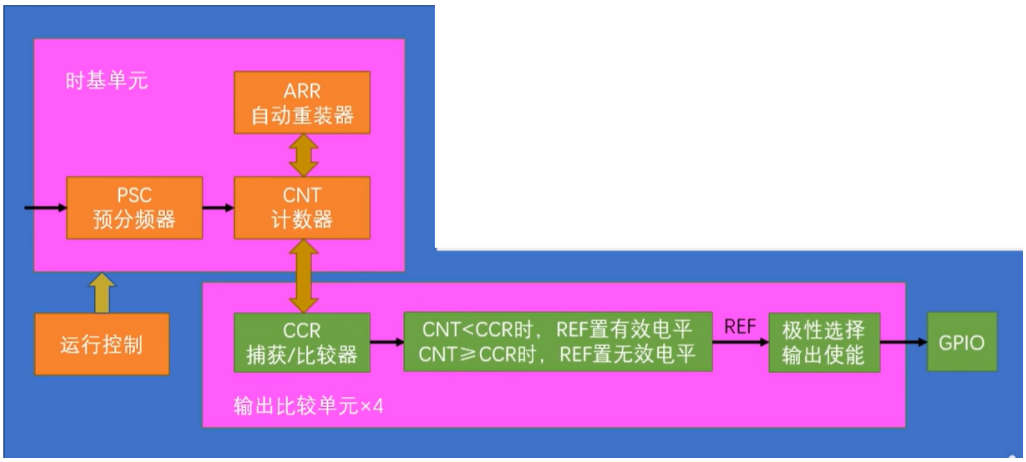


图1中：

- **时基单元（内部处理）**
  - **ARR（Auto-reload register）自动重装寄存器： 计数器到这个值就触发中断或输出比较等操作**
  - **CNT（Counter）时基单元里的计数器： 由内部或外部信号引起，通过预分频器分频进行数据自增或自减**
  - **PSC (Prescaler) 预分频器：其值表示将传入信号的频率减半的次数，也就是每一个脉冲的时间放大的次数。**
  - **运行控制：负责启动停止、计数器向上或向下计数等功能**
- **时基单元左边（信号数据的生成和处理）**
  - **为时基单元提供时钟。要时钟就要频率源，在电路中最常用的频率源就是晶振，可以是在STM32芯片的内部，也可以是外部。在实际中经常用8MHZ的晶振，由外部晶振(8MHZ)，作为系统的频率源，经过倍频处理，将频率提高到72MHZ，将72MHZ输入到系统时钟，在经由APB总线到定时器RCC内部时钟，再选择内部时钟模式，信号数据即可进入时基单元。**

图2中：

- **时基单元右边（传出信号）**
  - **CCR（Capture/Compare Register）捕获/比较寄存器**
  - **输出比较（Output Compare）： 只有通用定时器和高级定时器各有4个输出比较通道，可同时输出4路PWM波形，基本定时器没有。输出比较可以通过比较CNT与CCR寄存器值的关系，来对输出电平 进行置1、置0或翻转的操作，用于输出一定频率和占空比的PWM波形，当CNT大于CCR或CNT小于CCR 时，输出置0或1，具体对应关系与输出模式配置有关。**
  - **REF（Reference）： 频率，占空比可调的PWM波形。**

4. PWM参数计算（依据CCR/CNT/ARR信号输出PWM）
   PWM频率 = CK_PSC / (PSC + 1)/ (ARR + 1)
   PWM占空比 = CCR/(ARR + 1)
   PWM分辨率 = 1/(ARR + 1)

5. SG90舵机

- 舵机的控制需要一个20ms 左右的时基脉冲，该脉冲的高电平部分一般为0.5ms~2.5ms 范围内的角度控制脉冲部分。以180 度角度伺服为例，对应的控制关系是： 0.5ms -- 0 度；
  1.0ms -- 45 度；
  1.5ms -- 90 度；

2.0ms -- 135 度；
2.5ms -- 180 度；

- 硬件电路：三条线：VCC，GND，PWM信号线

实现步骤

与输出PWM有关重要库函数：

```
void TIM_OC1Init(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct);  //
初始化输出比较单元
void TIM_SetCompare2(TIM_TypeDef* TIMx, uint16_t Compare1);   //单独设置CCR值
```

1. RCC开启时钟：打开TIM外设和连接舵机的GPIO外设的时钟。

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);//开启TIM时钟
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//开启GPIO时钟
```

2. 配置定时器：时钟源选择，时基单元。

```
TIM_InternalClockConfig(TIM2);//选择内部时钟
//初始化时基单元
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInitStructure.TIM_Period = 100 - 1;   //ARR自动重装值
TIM_TimeBaseInitStructure.TIM_Prescaler = 720 - 1;   //PSC预分频值
TIM_TimeBaseInitStructure.TIM_RepetitionCounter = 0;
TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStructure);
```

3. 配置输出比较单元：CCR的值输出比较模式，极性选择，输出使能。
   极性选择常用PWM模式1：
   向上计数：CNT<CCR时，REF置有效电平，CNT≥CCR时，REF置无效电平;
   向下计数：CNT>CCR时，REF置无效电平，CNT≤CCR时，REF置有效电平

```
TIM_OCInitTypeDef TIM_OCInitStructure;
TIM_OCStructInit(&TIM_OCInitStructure);   //给结构体赋初始值，之后就直接改要用就
行
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;   //PWM模式1
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;   //高极性，极性不转，
REF波形直接输出
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;  //输出使能
TIM_OCInitStructure.TIM_Pulse = 0;      //设置CCR值
TIM_OC1Init(TIM2, &TIM_OCInitStructure);
```

4. 配置GPIO：将与PWM对应的GPIO口配置为复用推挽输出。

```
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

5. 配置运行控制：启动计数器

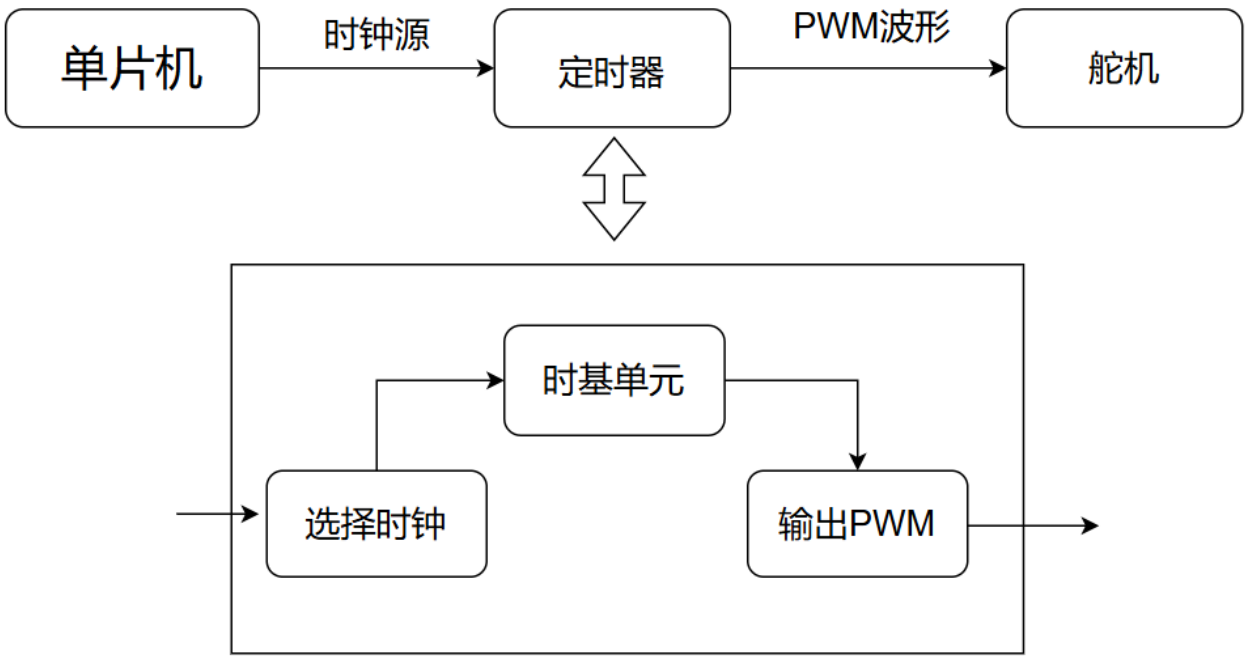
```
TIM_Cmd(TIM2, ENABLE);
```

6. 连接舵机，控制舵机旋转：按照PWM波占空比和角度对应关系配置舵机函数

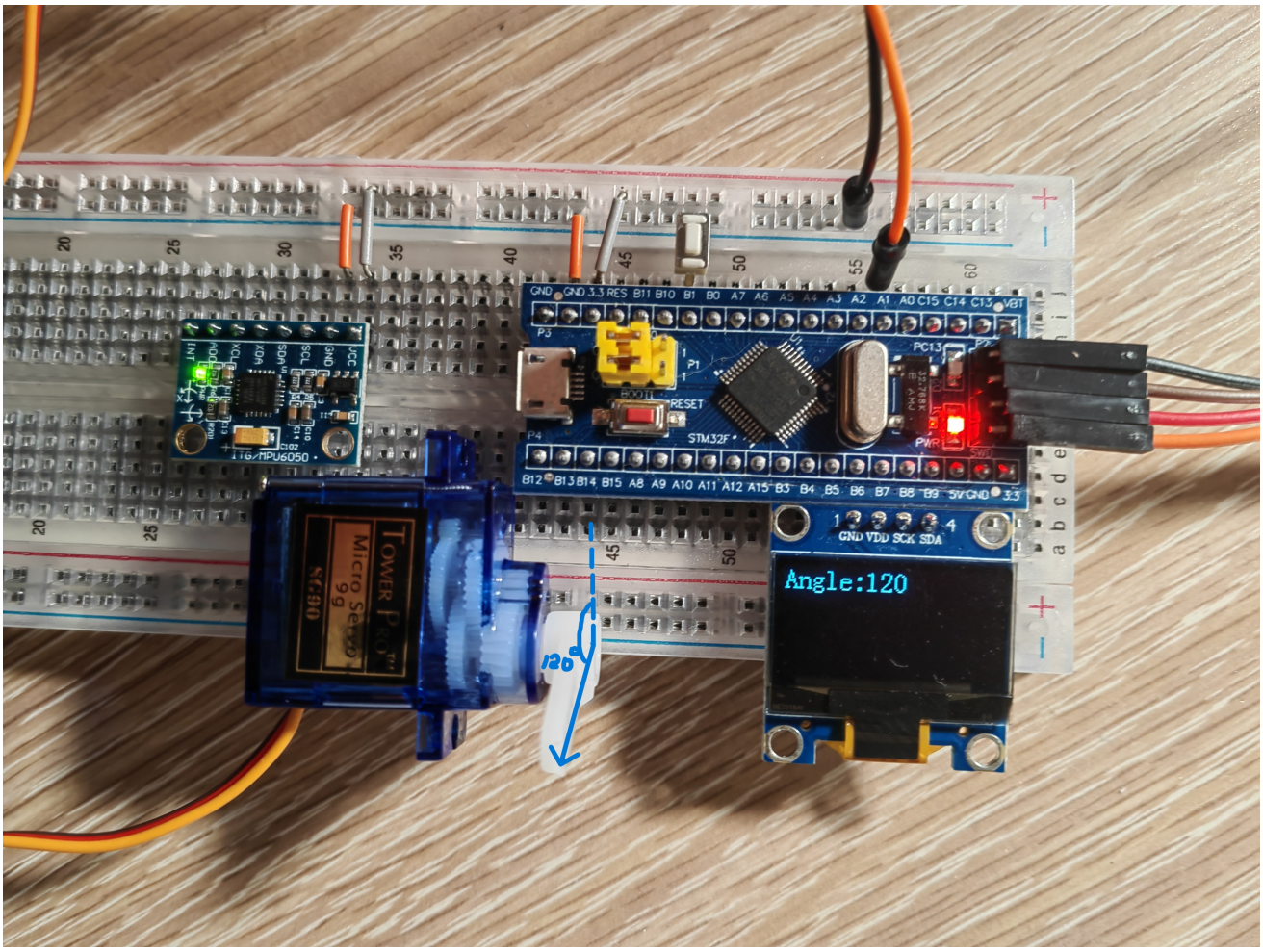
```
void Servo_Init(void)
{
    PWM_Init();
}

void Servo_SetAngle(float Angle)
{
    PWM_SetCompare2(Angle / 180 * 2000 + 500);
}
```

7. 主函数调用PWM_SetCompare(Angle)来控制舵机角度

框图



照片


Angle:120
120
Micro Servo 9g
SG90
GND VDD SCK SDA
STM32F
RESET
PWR
MPU6050