Lesson 5 Multi-channel Servo Control

1. Project Purpose

Learn the principle of the servo control and control several servos to rotate.

2. Project Principle

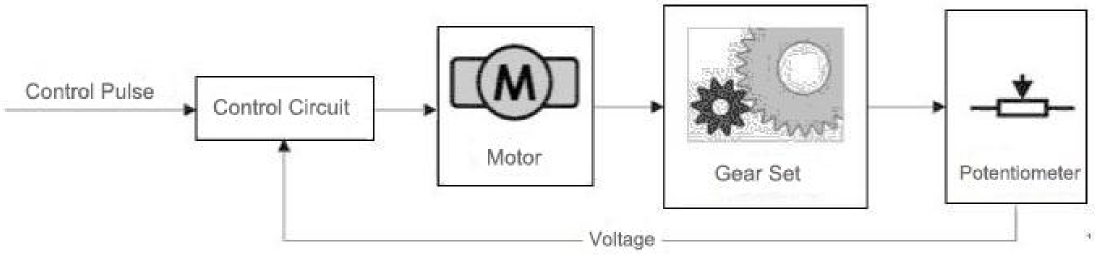
2.1 Servo Internal Structure

Servo consists of several parts namely small DC motor, a set of change gears, a linear feedback potentiometer, and a control circuit.

Of these, the high-speed DC motor provides the raw power for the servo and drives the reduction gear set to produce the high torque output. The greater the gear ratio, the greater the output torque of the servo, which means that it can drive a heavier load (limited by the gear strength), but the lower the output speed (response speed).

2.2 Servo Working Principle

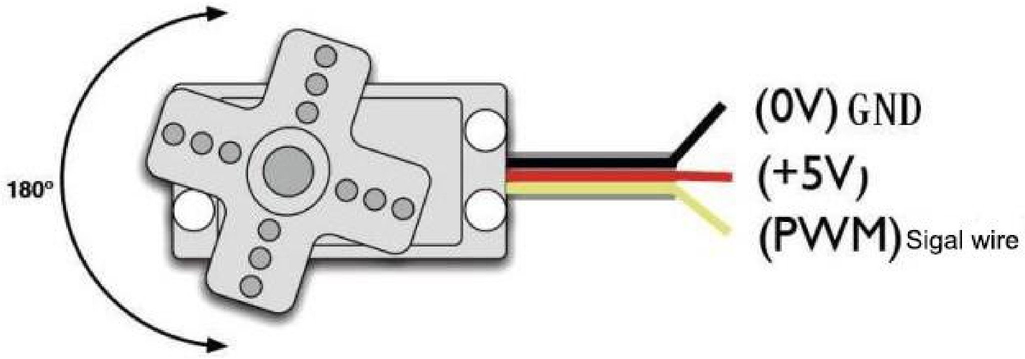
Servo is a typical closed loop feedback system. Its principle can refer to the figure below.



The reduction gear is driven by motors and its output terminal drives a linear potentiometer for positional detection. This potentiometer converts the angle into a proportional voltage feedback to the control circuit. Then the control circuit compares the proportional voltage with the angle corresponding to the input control signal and drives the motor to rotate clockwise or counterclockwise so as to make the potentiometer feedback angle to approach to the anticipated angle of the control signal, which achieves the accurate the purpose of accurate positioning of the servo motor.

2.3 How to control servo

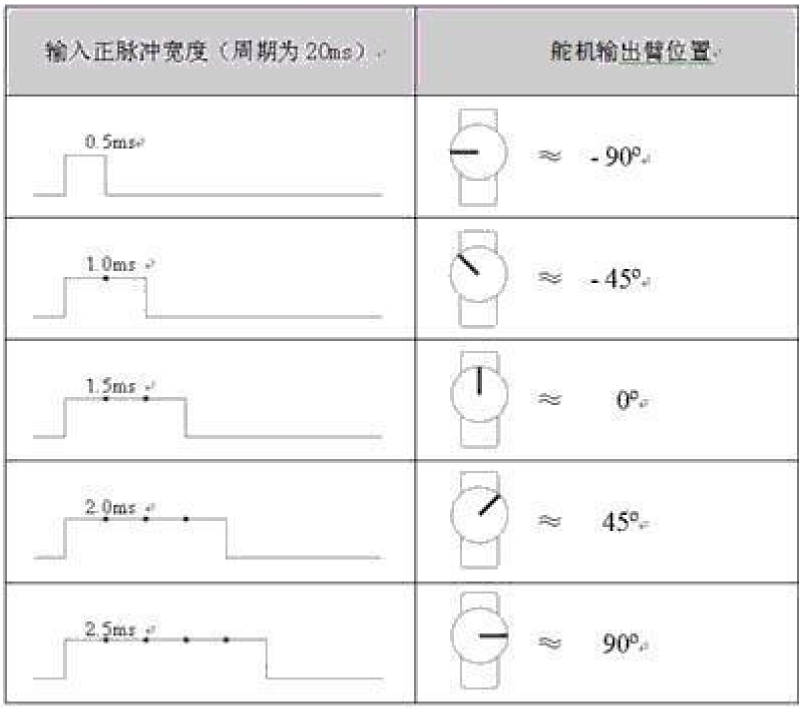
Servo motors have three wires: power, ground, and signal.



The power and ground wires are used to provide energy required for the internal DC motor and the control circuit. The voltage usually ranges between 5V and 8V, and the power supply should be isolated from the power supply of the processing system as much as possible. (because it will generate noise.)

Input a periodic positive pulse signal. The high level of this periodic pulse signal is usually between 1ms-2ms, and the low level time should be between 5ms and 20ms.

Analog servo is required to maintain periodic signal all the time to keep the servo angle. When the signal is lost, servo will no longer output power. The digital servo is used in uHand2.0 As long as the correct high-level signal is sent once, the locked angle can be maintained, and no strict requirement for the low level time.



输入正脉冲宽度（周期为 20ms）	舵机输出臂位置
0.5ms	≈ -90°
1.0ms	≈ -45°
1.5ms	≈ 0°
2.0ms	≈ 45°
2.5ms	≈ 90°

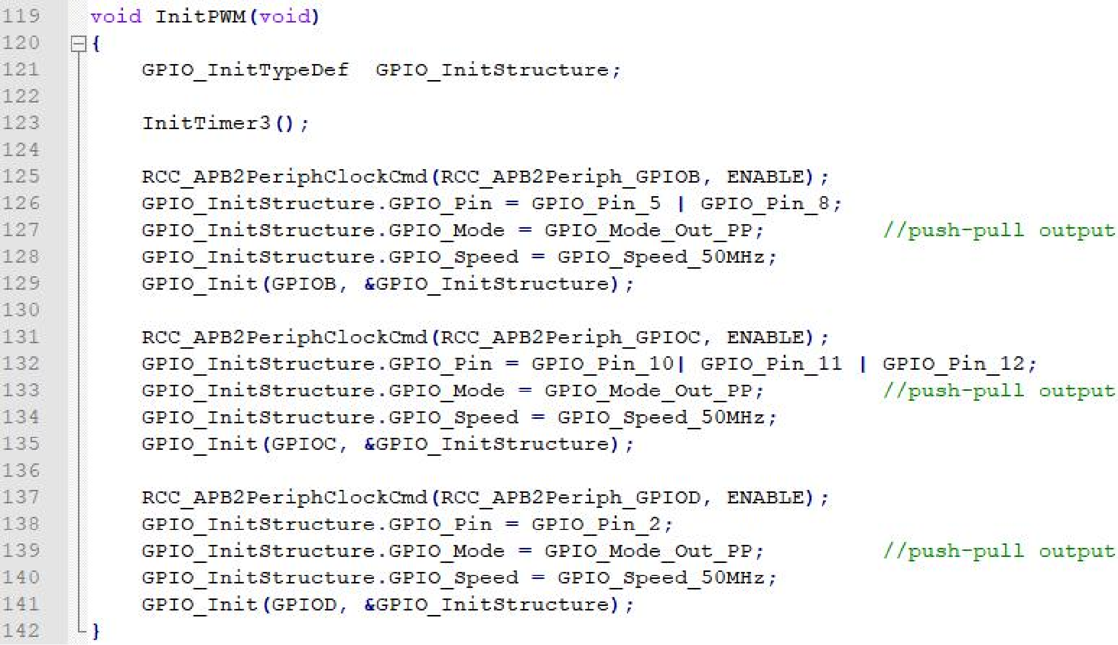
3. Program Analyst

1) The pulse width of the signal for controlling ranges 500us-2500us. 2500us–20000us is low level.

2) There is a way for you: take 500us high level as an example, that is, it is 500us high level + 2000 low level + 17500us low level; 1000us high level is 1000us high level + 1500us low level + 17500us low level. Therefore, we actually only need to control the high and low level division of 2500us and keep the low level at other times.

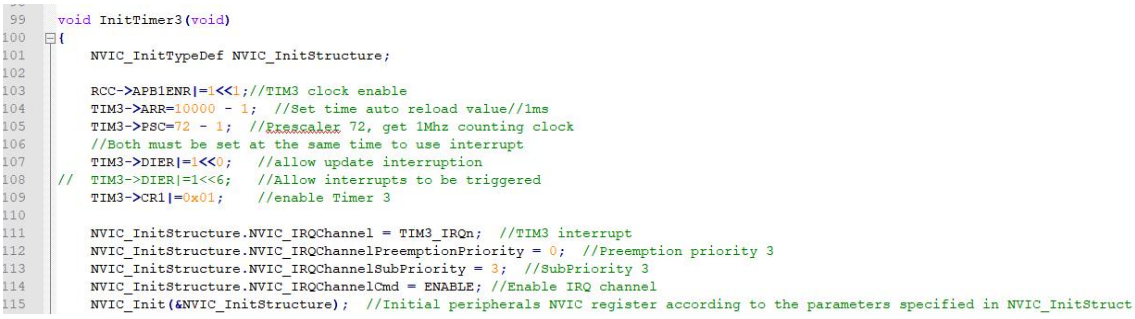
3) We know that s whole signal period is 20ms, that is, 20000us, which means eight 2500us. Therefore, cyclic control of the 2500is high and low level division of eight servos in a 20ms period can realize the operation of controlling eight servos with a timer.

4) If want to control the servo, the pin for controlling the servo needs to be configured to Push-Pull output mode.



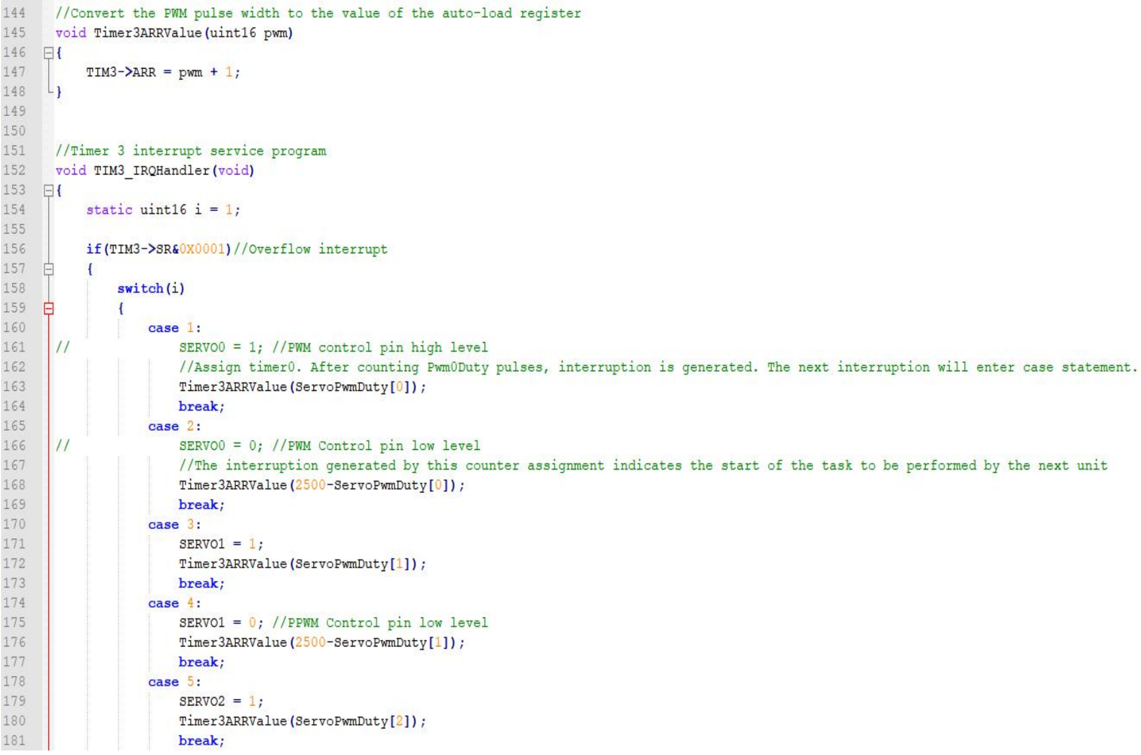
```
void InitPWM(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;

    InitTimer3();

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10| GPIO_Pin_11 | GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;        //push-pull output
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

5) The minimum resolution of the servo control signal is 1us so the time base of the timer can be configured to 1us. Turn on the timer interrupt, and then modify the output status of I/O port and the time of next interrupt in timer interrupt.



```
void InitTimer3(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC->APB1ENR|=1<<1;//TIM3 clock enable
    TIM3->ARR=10000 - 1;  //Set time auto reload value//1ms
    TIM3->PSC=72 - 1;  //Prescaler 72, get 1Mhz counting clock
    //Both must be set at the same time to use interrupt
    TIM3->DIER|=1<<0;   //allow update interruption
//  TIM3->DIER|=1<<6;   //Allow interrupts to be triggered
    TIM3->CR1|=0x01;    //enable Timer 3

    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;  //TIM3 interrupt
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;  //Preemption priority 3
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;  //SubPriority 3
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //Enable IRQ channel
    NVIC_Init(&NVIC_InitStructure);  //Initial peripherals NVIC register according to the parameters specified in NVIC_InitStruct
```

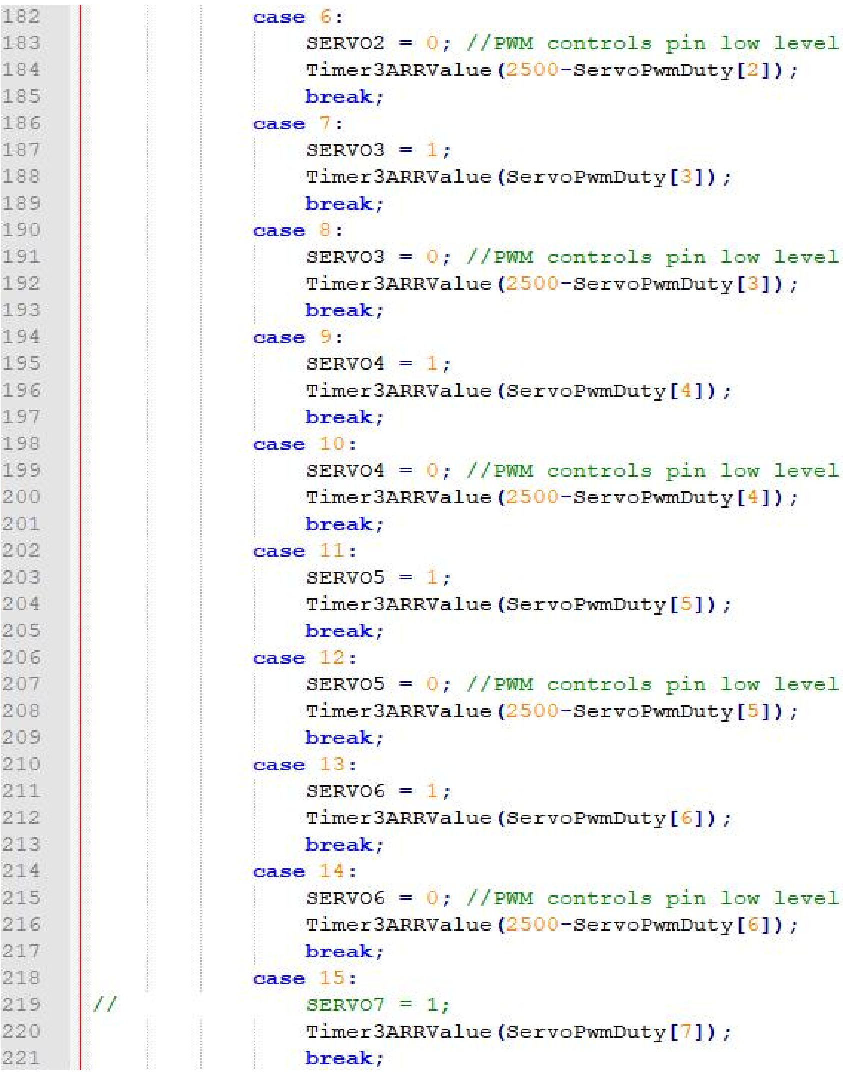
6) The following is the code for timer interrupt.



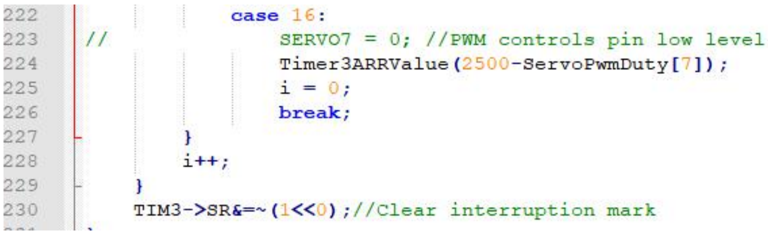
```c
//Convert the PWM pulse width to the value of the auto-load register
void Timer3ARRValue(uint16 pwm)
{
    TIM3->ARR = pwm + 1;
}


//Timer 3 interrupt service program
void TIM3_IRQHandler(void)
{
    static uint16 i = 1;

    if(TIM3->SR&0X0001)//Overflow interrupt
    {
        switch(i)
        {
            case 1:
//              SERVO0 = 1; //PWM control pin high level
                //Assign timer0. After counting Pwm0Duty pulses, interruption is generated. The next interruption will enter case statement.
                Timer3ARRValue(ServoPwmDuty[0]);
                break;
            case 2:
//              SERVO0 = 0; //PWM Control pin low level
                //The interruption generated by this counter assignment indicates the start of the task to be performed by the next unit
                Timer3ARRValue(2500-ServoPwmDuty[0]);
                break;
            case 3:
                SERVO1 = 1;
                Timer3ARRValue(ServoPwmDuty[1]);
                break;
            case 4:
                SERVO1 = 0; //PPWM Control pin low level
                Timer3ARRValue(2500-ServoPwmDuty[1]);
                break;
            case 5:
                SERVO2 = 1;
                Timer3ARRValue(ServoPwmDuty[2]);
                break;
```

7) Let’s briefly analyze the switch statement. Suppose the interrupt is entered for the first time now. After entering the interrupt, execute case 1 ,and then set the I/O port of SERVO0 to high level, and set the time of timer interrupt to the high level time of the servo signal.

8) When entering the next interrupt, execute case 2, and set the I/O port of SERVO0 to low level, and then set the interrupt time of the timer to the remaining time in 2500us.By repeating this process 8 times, a period of 20ms can be realized.



```
            case 6:
                SERVO2 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[2]);
                break;
            case 7:
                SERVO3 = 1;
                Timer3ARRValue(ServoPwmDuty[3]);
                break;
            case 8:
                SERVO3 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[3]);
                break;
            case 9:
                SERVO4 = 1;
                Timer3ARRValue(ServoPwmDuty[4]);
                break;
            case 10:
                SERVO4 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[4]);
                break;
            case 11:
                SERVO5 = 1;
                Timer3ARRValue(ServoPwmDuty[5]);
                break;
            case 12:
                SERVO5 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[5]);
                break;
            case 13:
                SERVO6 = 1;
                Timer3ARRValue(ServoPwmDuty[6]);
                break;
            case 14:
                SERVO6 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[6]);
                break;
            case 15:
//              SERVO7 = 1;
                Timer3ARRValue(ServoPwmDuty[7]);
                break;
```



```
            case 16:
//              SERVO7 = 0; //PWM controls pin low level
                Timer3ARRValue(2500-ServoPwmDuty[7]);
                i = 0;
                break;
        }
        i++;
    }
    TIM3->SR&=~(1<<0);//Clear interruption mark
```

9) Because there are only 6-ch servo ports on the palm controller, comment out SERVO0 and SERVO7. In this way, when operating the array, the subscript of the array is the servo corresponding to the servo number.