Lesson 12 Single Servo Mode

The program defaults single servo mode. This section is only for reference.

1. Project Purpose

Add the mode of controlling a single servo by the handle, so that the handle button can directly control a certain servo on the robot.

2. Project Principle

In the previous program, the function of PS2 handle to call the corresponding action group has been implemented, but sometimes running the pre-programmed action group can not meet the actual needs, for example, directly controlling the rotation of a certain servo. In order to implement this function, we add a single servo mode.

The handle principle can be reviewed in Lesson 8 PS2 Handle Control.

3. Program Analyst

1) Create a variable mode to record the working mode: 0 is the action group mode. 1 is single servo mode. Because the variable is initialized to 0, it defaults to single servo mode.

```
350         if(mode == 1)
351         {
352             if( PS2_Button( PSB_SELECT ) & PS2_ButtonPressed( PSB_START ) )
353             {
354                 mode = 0;
355                 Ps2State = 0;
356                 manual = TRUE;
357                 BuzzerState = 1;
358                 LED=~LED;
359                 DelayMs(80);
360                 manual = FALSE;
361                 DelayMs(50);
362                 manual = TRUE;
363                 BuzzerState = 1;
364                 DelayMs(80);
365                 manual = FALSE;
366                 LED=~LED;
367             }
368             else
369             {
370                 if(PS2KeyValue && !PS2_Button(PSB_SELECT))
371             {
372                 LED=~LED;
373             }
374 
375                 switch( PS2KeyValue )
376                 {
377                     //Control the servo to rotate according to the button pressed
378                     case PSB_PAD_LEFT:
379                         ServoSetPluseAndTime( 6, ServoPwmDutySet[6] + 20, 50 );
380                         BusServoPwmDutySet[6] = BusServoPwmDutySet[6] + 10;
381                         if (BusServoPwmDutySet[6] > 2500)
382                             BusServoPwmDutySet[6] = 2500;
383                         BusServoCtrl(6,SERVO_MOVE_TIME_WRITE,BusServoPwmDutySet[6],50);
384                         break;
385                     case PSB_PAD_RIGHT:
386                         ServoSetPluseAndTime( 6, ServoPwmDutySet[6] - 20, 50 );
387                         BusServoPwmDutySet[6] = BusServoPwmDutySet[6] - 10;
388                         if (BusServoPwmDutySet[6] < 500)
389                             BusServoPwmDutySet[6] = 500;
390                         BusServoCtrl(6,SERVO_MOVE_TIME_WRITE,BusServoPwmDutySet[6],50);
391                         break;
392                     case PSB_PAD_UP:
393                         ServoSetPluseAndTime( 5, ServoPwmDutySet[5] - 20, 50 );
394                         BusServoPwmDutySet[5] = BusServoPwmDutySet[5] - 10;
395                         if (BusServoPwmDutySet[5] < 900)
396                             BusServoPwmDutySet[5] = 900;
397                         BusServoCtrl(5,SERVO_MOVE_TIME_WRITE,BusServoPwmDutySet[5],50);
398                         break;
399                     case PSB_PAD_DOWN:
400                         ServoSetPluseAndTime( 5, ServoPwmDutySet[5] + 20, 50 );
401                         BusServoPwmDutySet[5] = BusServoPwmDutySet[5] + 10;
402                         if (BusServoPwmDutySet[5] > 2200)
403                             BusServoPwmDutySet[5] = 2200;
404                         BusServoCtrl(5,SERVO_MOVE_TIME_WRITE,BusServoPwmDutySet[5],50);
405                         break;
406                     case PSB_L1:
407                         ServoSetPluseAndTime( 2, ServoPwmDutySet[2] + 20, 50 );
```

The program above is the code of mode 1 (single servo mode), which detects the status of PSB_SELECT and PSB_START by calling PS2_Button and PS2_ButtonPressed.

When PSB_SELECT has been pressed, and then PSB_START is pressed, it can switch to mode 0 (action group mode).

```
bool PS2_NewButtonState( u16 button )
{
  button = 0x0001u << ( button - 1 );  //The value of the input button is the value of the button in the data bit + 1, for example, the value of the PSB_SELECT macro is 1, the bit in the data is 0, and so on.
  return ( ( ( LastHandkey ^ Handkey ) & button ) > 0 );  //The final key data and the key data of this time are XORed, and the result is that the two different parts are 1, and the key whose state has changed is obtained.
                                                          //Then perform the AND operation with the button we want to detect. If the button changes, the result is 1, which is greater than 0, so the return is true
}

bool PS2_Button( u16 button )
{
  button = 0x0001u << ( button - 1 );  //The value of the input button is the value of the button in the data bit + 1, for example, the value of the PSB_SELECT macro is 1, the bit in the data is 0, and so on.
  return ( ( (~Handkey) & button ) > 0 );  //When the button is pressed, the corresponding bit is 0, if it is not pressed, it is 1. After the key data is inverted, it becomes the button as 1, and if it is not pressed, it is 0.
                                           //Then do the AND operation with the button we want to detect. If this button is pressed, the corresponding bit is 1, and if it is not pressed, it is 0. The result of the comparison with 0 is returned.
}

bool PS2_ButtonPressed( u16 button )
{
  return (PS2_NewButtonState( button ) && PS2_Button( button ));  //The button is pressed, and this is a new state of the button, then the button has just been pressed
}

bool PS2_ButtonReleased( u16 button )
{
  return ( PS2_NewButtonState( button ) && !PS2_Button( button )); //The button is not pressed, and this is a new state of the button, then the button has just been released
```

2) As the figure shown above, in PS2_ButtonPressed and PS_Button functions, Handkey is the latest status of the button read, while LastHandkey is the status of the button read in the last time. The difference between the two states can be used to determine whether the button is continuously pressed or just pressed. Similarly, the code of mode 0 (action group mode) is shown in the following figure.

```
    if( PS2_Button( PSB_SELECT ) && PS2_ButtonPressed( PSB_START ) )  //Check if the SELECT button is held down, then press the START button, if yes, switch the mode
{
  mode = 1; //Change the mode to 1, the single servo mode
  FullActStop();  //Stop running action group
  Ps2State = 0;  //Clear the flags used in the action group mode.
  ServoSetPluseAndTime( 1, 1500, 1000 );  //Turn all the servos of the robotic arm to the 1500 position
  ServoSetPluseAndTime( 2, 1500, 1000 );
  ServoSetPluseAndTime( 3, 1500, 1000 );
  ServoSetPluseAndTime( 4, 1500, 1000 );
  ServoSetPluseAndTime( 5, 1500, 1000 );
  ServoSetPluseAndTime( 6, 1500, 1000 );
        manual = TRUE;
      BuzzerState = 1;
      LED=~LED;
      DelayMs(80);
      manual = FALSE;
      DelayMs(50);
      manual = TRUE;
      BuzzerState = 1;
      DelayMs(80);
      manual = FALSE;
      LED=~LED;
}
      else
      {
  if(PS2KeyValue && !Ps2State && !PS2_Button(PSB_SELECT))
  {
      LED=~LED;
  }

  switch(PS2KeyValue)
  {
      case 0:
          if(Ps2State)
          {
              Ps2State = FALSE;
          }
          break;

      case PSB_START:
          if(!Ps2State)
          {
              FullActRun(0,1);
          }
          Ps2State = TRUE;
          break;

      case PSB_PAD_UP:
          if(!Ps2State)
          {
              FullActRun(1,1);
          }
```

3) We can find that the status of PSB_SELECT and PSB_STAR is judged like mode 1 (single servo mode), and then whether the mode has been switched is judged according to the status. If the mode does not be switched, the palm will execute different action groups according to the different pressed buttons.