Lesson 11 Run the Action stored in Flash

1. Project Purpose

Through the serial command or PS2 handle, control to read the action data in Flash and run the read action group.

2. Project Principle

In this section, we will implement the robotic arm to run the specified action group through the serial command. We need to implement the function of running the action group, and implement the corresponding serial command and call the corresponding action group through the button of PS2 handle. Design the corresponding serial port command protocol is as follow:

Run action group command

Command name: CMD_ACTION_RUN Command valueL:6 Data length:5

Instruction: run action group. If the parameter times are unlimited, the parameter value is 0.

Parameter 1: The number parameter of the action group to be run.

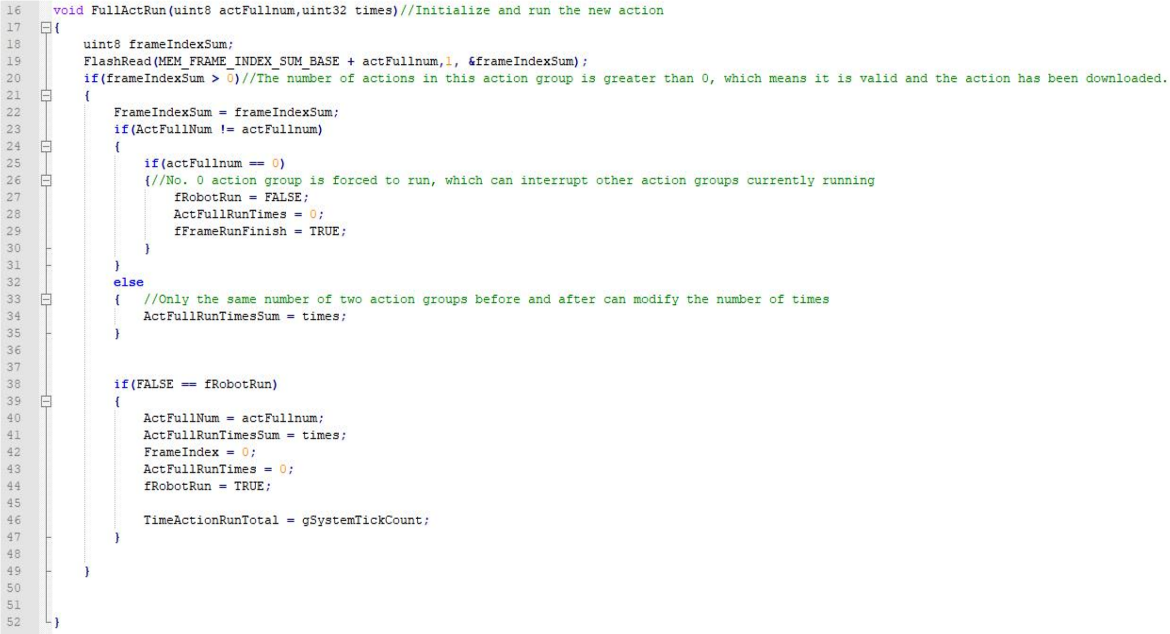
Parameter 2: The lower-byte parameter of the times of the action group to be executed.

Parameter 3: The upper byte parameter of the times of the action group to be executed.

To realize the function of running the action group in SPI Flash, you must understand that the action group is actually a combination of several actions, and the action is the change of the position of the servo.

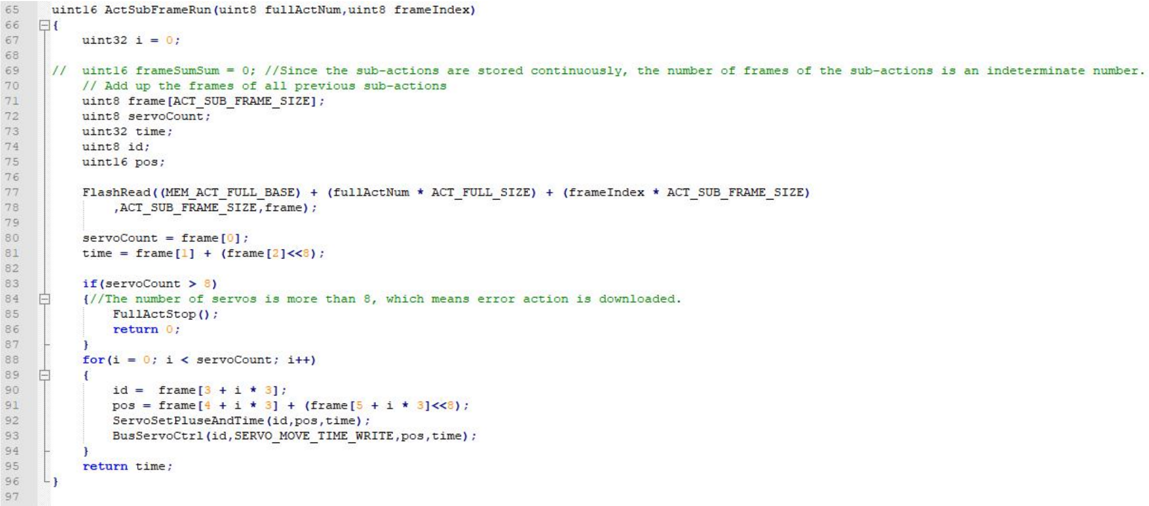
3. Program Analyst

Before running the action group, the environment for running action group need to be set first. For example, whether there are actions in action group, the number of actions it contains and check whether there is an action group currently running. Next do corresponding process according to different situations to prevent errors. Then set the parameters for the action group to be run and set the mark of the action group to be run.



```
void FullActRun(uint8 actFullnum,uint32 times)//Initialize and run the new action
{
    uint8 frameIndexSum;
    FlashRead(MEM_FRAME_INDEX_SUM_BASE + actFullnum,1, &frameIndexSum);
    if(frameIndexSum > 0)//The number of actions in this action group is greater than 0, which means it is valid and the action has been downloaded.
    {
        FrameIndexSum = frameIndexSum;
        if(ActFullNum != actFullnum)
        {
            if(actFullnum == 0)
            {//No. 0 action group is forced to run, which can interrupt other action groups currently running
                fRobotRun = FALSE;
                ActFullRunTimes = 0;
                fFrameRunFinish = TRUE;
            }
        }
        else
        {   //Only the same number of two action groups before and after can modify the number of times
            ActFullRunTimesSum = times;
        }


        if(FALSE == fRobotRun)
        {
            ActFullNum = actFullnum;
            ActFullRunTimesSum = times;
            FrameIndex = 0;
            ActFullRunTimes = 0;
            fRobotRun = TRUE;

            TimeActionRunTotal = gSystemTickCount;
        }

    }


}
```

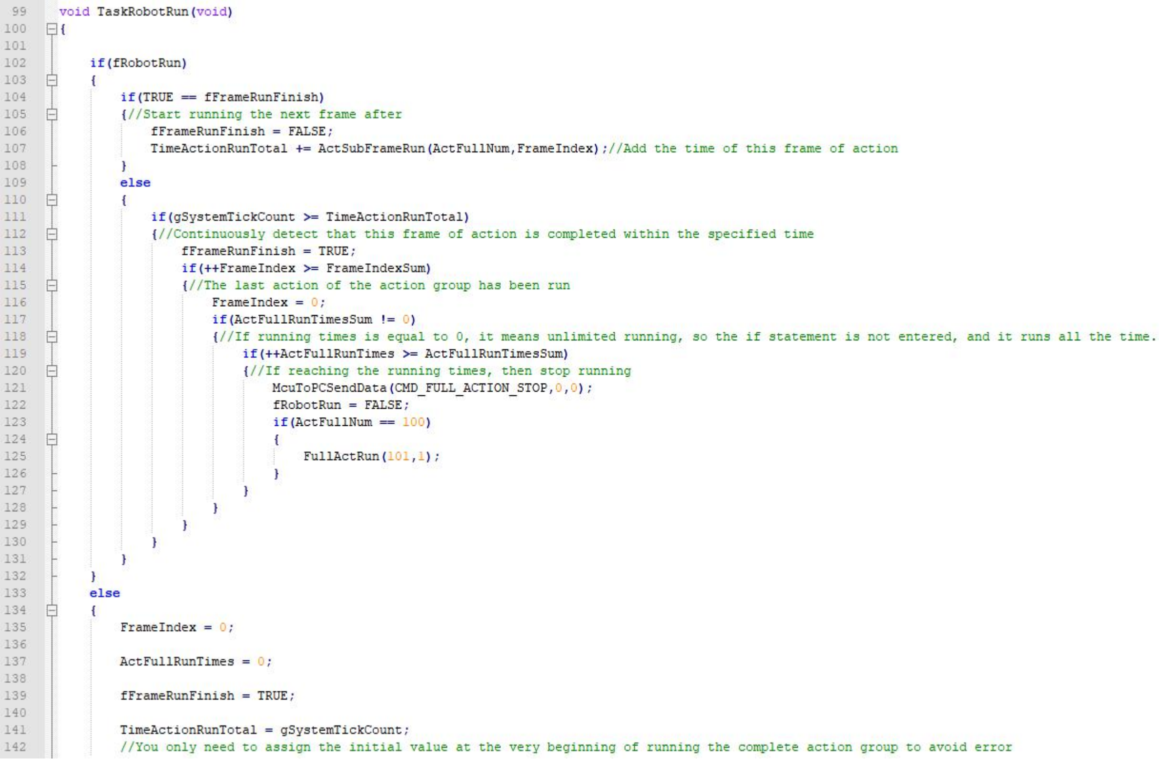
Running an action is reading the running time of the action and the angle of each servo from the corresponding position in Flash. Then control the servo through the previous control servo program to implement the effect of rotating the corresponding angle within a specified time.



```c
uint16 ActSubFrameRun(uint8 fullActNum,uint8 frameIndex)
{
    uint32 i = 0;

//  uint16 frameSumSum = 0; //Since the sub-actions are stored continuously, the number of frames of the sub-actions is an indeterminate number.
    // Add up the frames of all previous sub-actions
    uint8 frame[ACT_SUB_FRAME_SIZE];
    uint8 servoCount;
    uint32 time;
    uint8 id;
    uint16 pos;

    FlashRead((MEM_ACT_FULL_BASE) + (fullActNum * ACT_FULL_SIZE) + (frameIndex * ACT_SUB_FRAME_SIZE)
        ,ACT_SUB_FRAME_SIZE,frame);

    servoCount = frame[0];
    time = frame[1] + (frame[2]<<8);

    if(servoCount > 8)
    {//The number of servos is more than 8, which means error action is downloaded.
        FullActStop();
        return 0;
    }
    for(i = 0; i < servoCount; i++)
    {
        id =  frame[3 + i * 3];
        pos = frame[4 + i * 3] + (frame[5 + i * 3]<<8);
        ServoSetPluseAndTime(id,pos,time);
        BusServoCtrl(id,SERVO_MOVE_TIME_WRITE,pos,time);
    }
    return time;
}
```

1) After an action is executed, the executing time of the action is deferred. When the action time is expired, the action will be judged that the execution of the action is complete and the next new action can be execute.

(At the same time the function will check the action group running flag, the flag is true to execute the action group, for false will not be executed)

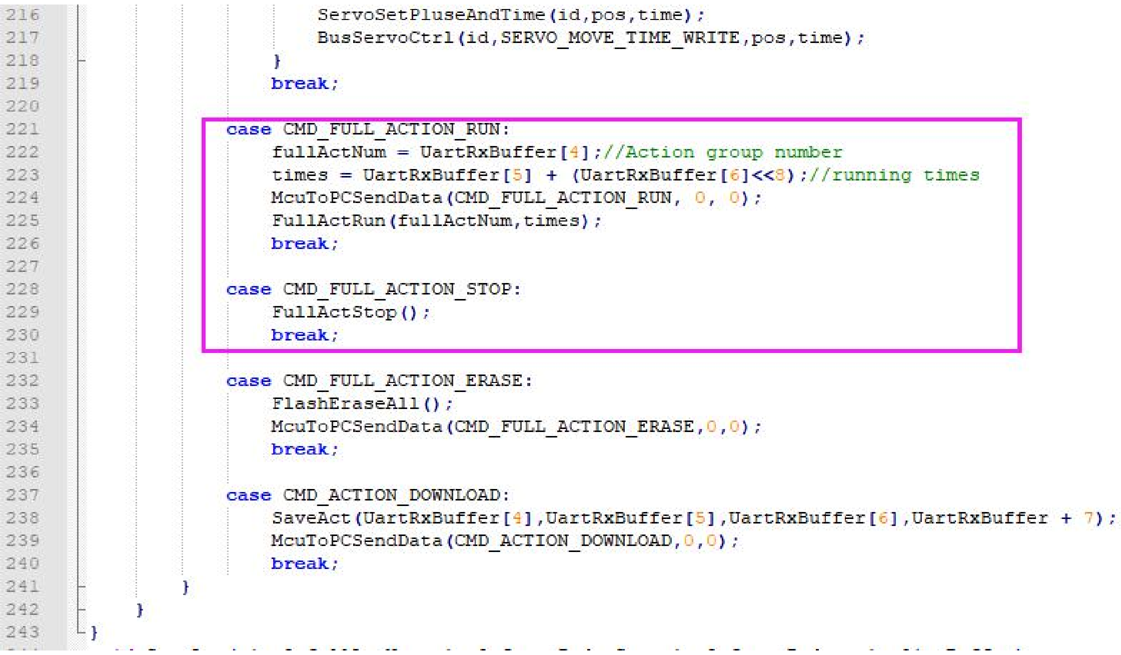


```c
void TaskRobotRun(void)
{

    if(fRobotRun)
    {
        if(TRUE == fFrameRunFinish)
        {//Start running the next frame after
            fFrameRunFinish = FALSE;
            TimeActionRunTotal += ActSubFrameRun(ActFullNum,FrameIndex);//Add the time of this frame of action
        }
        else
        {
            if(gSystemTickCount >= TimeActionRunTotal)
            {//Continuously detect that this frame of action is completed within the specified time
                fFrameRunFinish = TRUE;
                if(++FrameIndex >= FrameIndexSum)
                {//The last action of the action group has been run
                    FrameIndex = 0;
                    if(ActFullRunTimesSum != 0)
                    {//If running times is equal to 0, it means unlimited running, so the if statement is not entered, and it runs all the time.
                        if(++ActFullRunTimes >= ActFullRunTimesSum)
                        {//If reaching the running times, then stop running
                            McuToPCSendData(CMD_FULL_ACTION_STOP,0,0);
                            fRobotRun = FALSE;
                            if(ActFullNum == 100)
                            {
                                FullActRun(101,1);
                            }
                        }
                    }
                }
            }
        }
    }
    else
    {
        FrameIndex = 0;

        ActFullRunTimes = 0;

        fFrameRunFinish = TRUE;

        TimeActionRunTotal = gSystemTickCount;
        //You only need to assign the initial value at the very beginning of running the complete action group to avoid error
```

2) SystemTickCount is the number of milliseconds elapsed from the start of the program to this moment. The number of the milliseconds plus the running time of the action is the time required for the entire program to complete the action. When the number of the milliseconds matches, it is judge that the

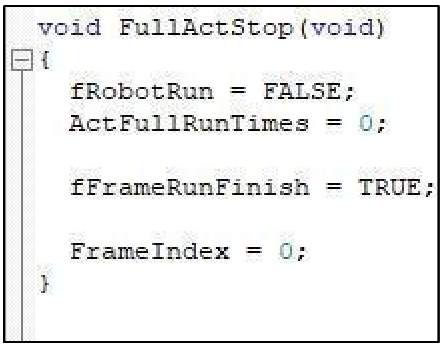
next action will be run or the entire action group has been run.

3) After implementation of the function of running action group, look at the function of the serial data processing: process the command of the movement action group.



```
                ServoSetPluseAndTime(id,pos,time);
                BusServoCtrl(id,SERVO_MOVE_TIME_WRITE,pos,time);
            }
            break;

        case CMD_FULL_ACTION_RUN:
            fullActNum = UartRxBuffer[4];//Action group number
            times = UartRxBuffer[5] + (UartRxBuffer[6]<<8);//running times
            McuToPCSendData(CMD_FULL_ACTION_RUN, 0, 0);
            FullActRun(fullActNum,times);
            break;

        case CMD_FULL_ACTION_STOP:
            FullActStop();
            break;

        case CMD_FULL_ACTION_ERASE:
            FlashEraseAll();
            McuToPCSendData(CMD_FULL_ACTION_ERASE,0,0);
            break;

        case CMD_ACTION_DOWNLOAD:
            SaveAct(UartRxBuffer[4],UartRxBuffer[5],UartRxBuffer[6],UartRxBuffer + 7);
            McuToPCSendData(CMD_ACTION_DOWNLOAD,0,0);
            break;
        }
    }
}
```

4) Stop the action group is to call the stop action group function. This function will set variable and flags, so that Task_RobotRun function will stop running action group.



```
void FullActStop(void)
{
    fRobotRun = FALSE;
    ActFullRunTimes = 0;

    fFrameRunFinish = TRUE;

    FrameIndex = 0;
}
```