Lesson 10 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. Look at the corresponding serial command protocol as follows:

The command of running action group

Command name CMD _ACTION_GROUP_RUN

Command value: 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.

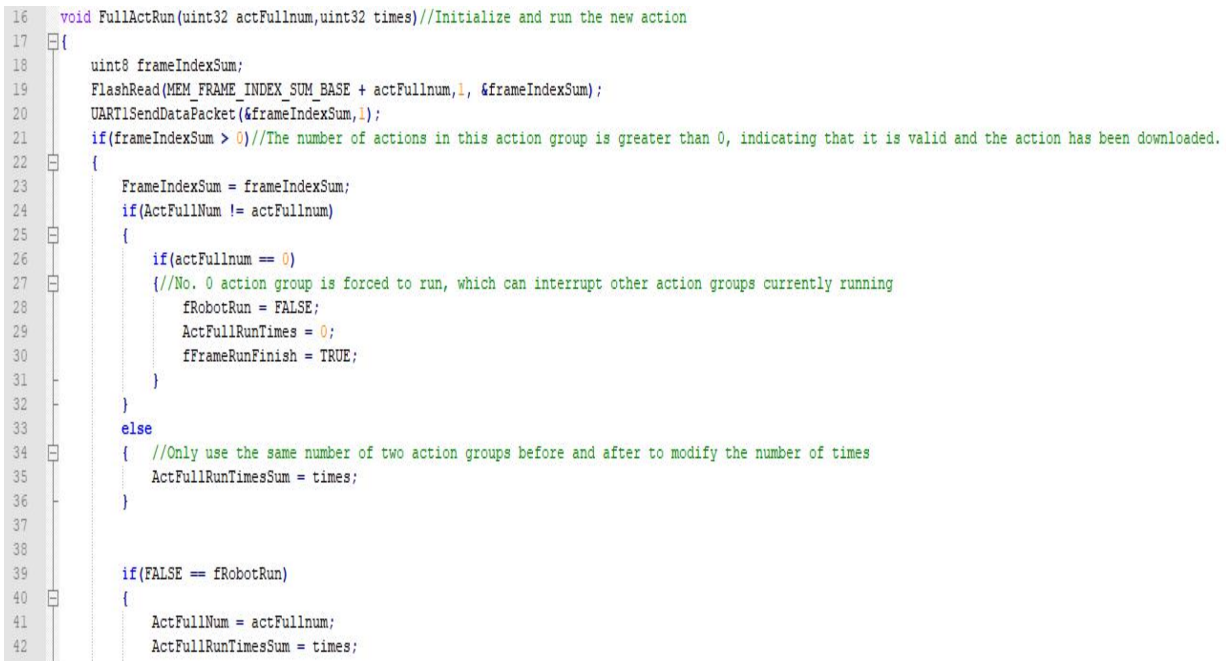
Parameter 3: The upper byte parameter of the times of the action group to be executed.

For example, control No.1 servo to rotate to 2000 position on 1000ms. The command is as follow:

Frame header	Data length	Command	Parameter
0x55 0x55	0x08	0x03	0x01 0xE8 0x03 0x01 0xE8 0x03 0x01 0xD0 0x07

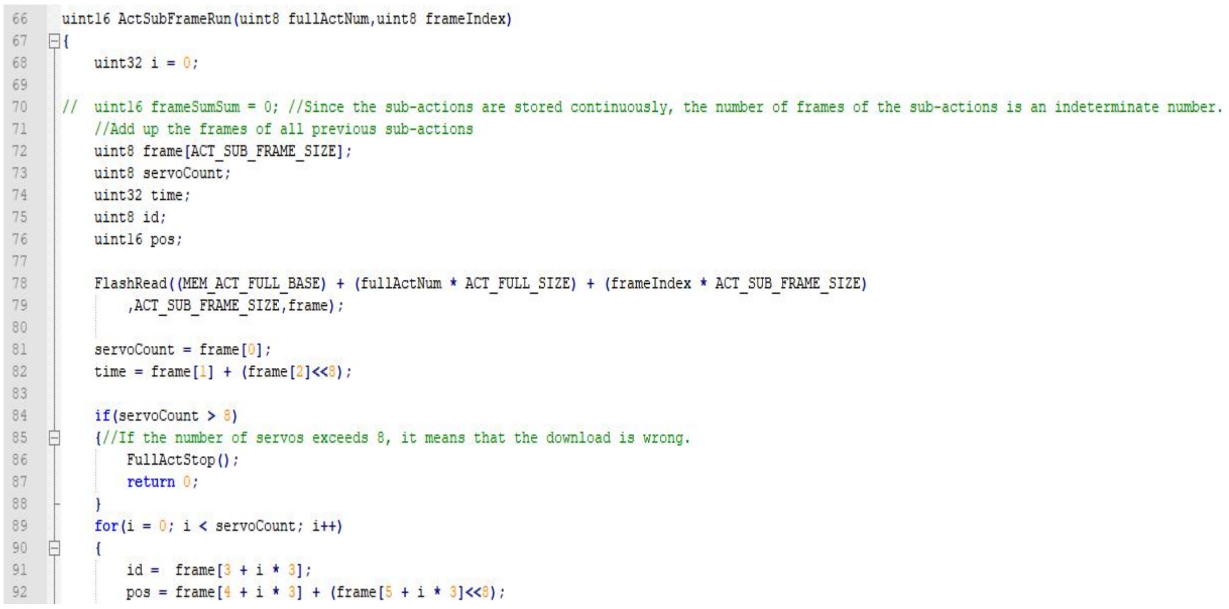
3. Program Analyst

1) Firstly, set the environment for running action group. For example, whether there are actions in the action group, the number of action contained in the action group, and check whether there is any action group currently running. Then do the corresponding treatment according to different situations to prevent errors. Then set the parameters for running the action group and set the sign for running the action group.

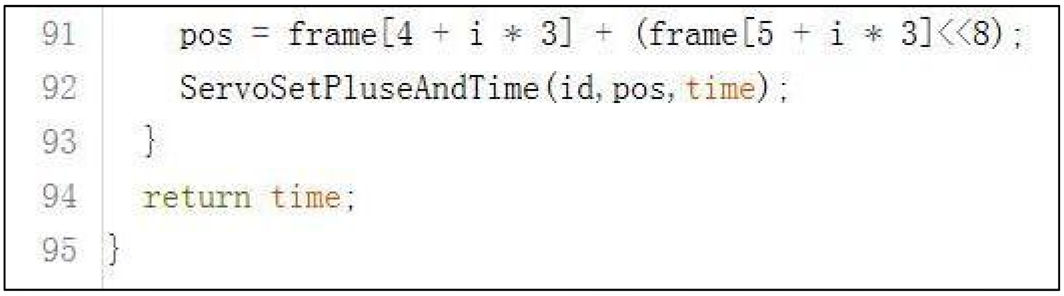


```
void FullActRun(uint32 actFullnum,uint32 times)//Initialize and run the new action
{
    uint8 frameIndexSum;
    FlashRead(MEM_FRAME_INDEX_SUM_BASE + actFullnum,1, &frameIndexSum);
    UART1SendDataPacket(&frameIndexSum,1);
    if(frameIndexSum > 0)//The number of actions in this action group is greater than 0, indicating that 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 use the same number of two action groups before and after to modify the number of times
            ActFullRunTimesSum = times;
        }


        if(FALSE == fRobotRun)
        {
            ActFullNum = actFullnum;
            ActFullRunTimesSum = times;
```

2) Running an action is to read the running time of this action and and the angle information of each servo from the corresponding location in Flash. Then control the servo to rotate to the corresponding angle at the specified time through the program written before to control the servo.



```
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)
    {//If the number of servos exceeds 8, it means that the download is wrong.
        FullActStop();
        return 0;
    }
    for(i = 0; i < servoCount; i++)
    {
        id =  frame[3 + i * 3];
        pos = frame[4 + i * 3] + (frame[5 + i * 3]<<8);
```



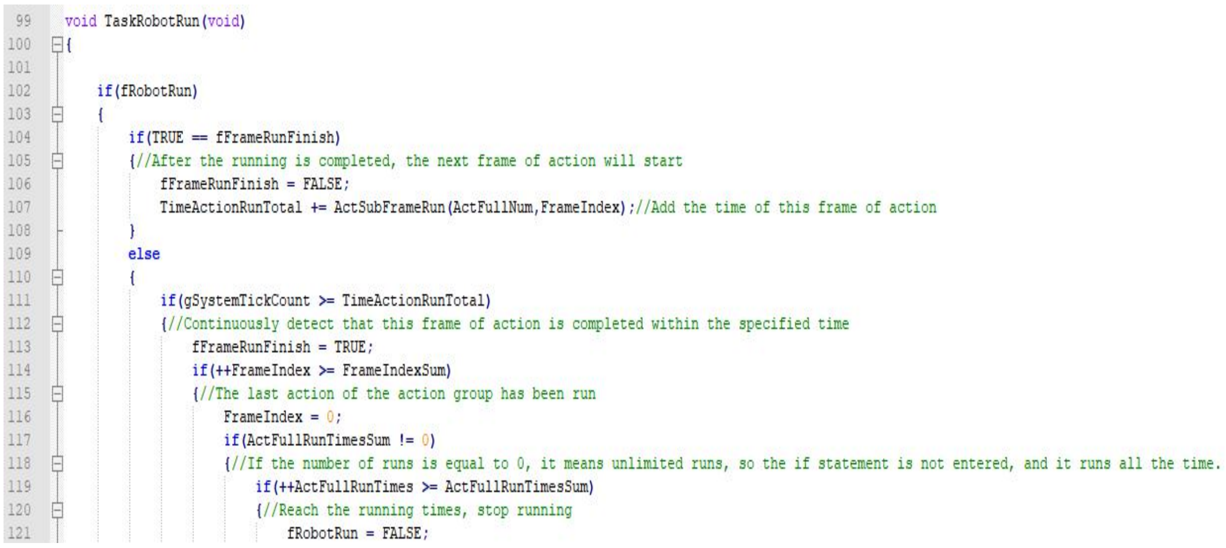
```
        pos = frame[4 + i * 3] + (frame[5 + i * 3]<<8);
        ServoSetPluseAndTime(id, pos, time);
    }
    return time;
}
```

3) Implement the running action is to read the action data from the corresponding position in the Flash. Then call ServoSetPluseAndTime function and add the treatment that can judge whether the data is correct to prevent the wrong action after the data error.

4) To execute an action group, it needs to be able to automatically execute one action and then execute the next action until all the actions in the action group have been executed.

5) We want the program to defer the execution time of an action after it has been executed. When the execution time has expired, the action will be considered

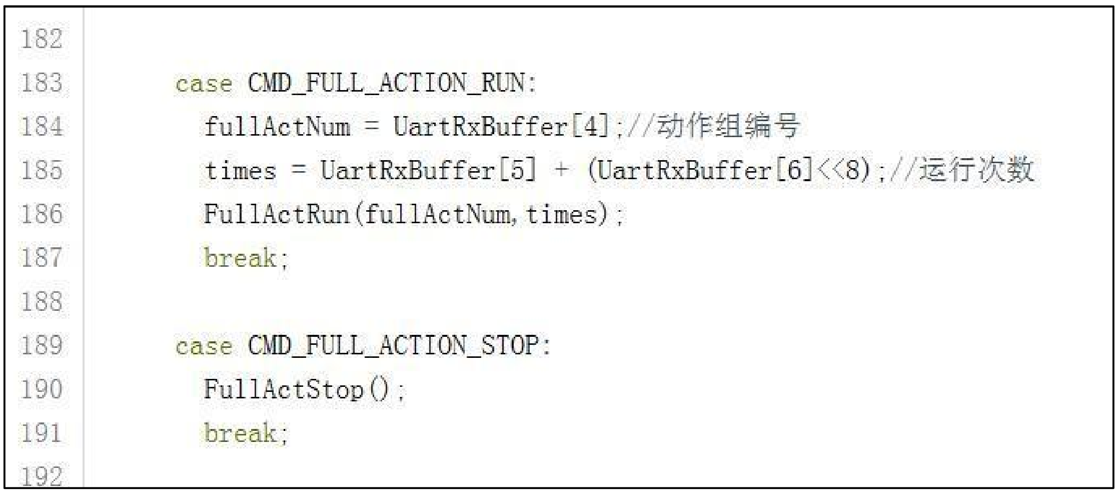
finished and a new action will be executed. At the same time, this function can check the running sign of action group. When the sign is true, the action group will be executed, and if the sign is false, it will not be executed.



```c
void TaskRobotRun(void)
{

	if(fRobotRun)
	{
		if(TRUE == fFrameRunFinish)
		{//After the running is completed, the next frame of action will start
			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 the number of runs is equal to 0, it means unlimited runs, so the if statement is not entered, and it runs all the time.
						if(++ActFullRunTimes >= ActFullRunTimesSum)
						{//Reach the running times, stop running
							fRobotRun = FALSE;
```

6) gSystemTickCount is the number of milliseconds that have elapsed since the returned program started to run. The number of milliseconds at this moment plus the running time of the action is the number of milliseconds elapsed for the whole program when the action run is completed.

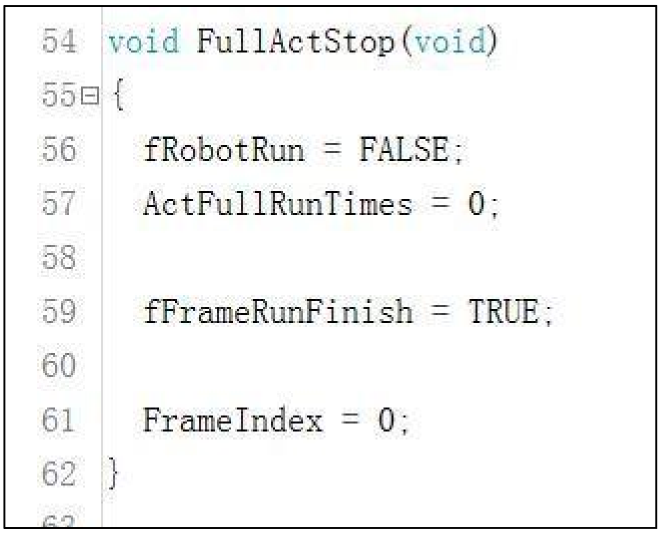
7) When the 54 milliseconds are reached, it can be considered that the next action is started or the entire action group has been completed. After implementing the function of running action group, we add the code for processing the action group command in the serial data processing function.



```c

		case CMD_FULL_ACTION_RUN:
			fullActNum = UartRxBuffer[4];//动作组编号
			times = UartRxBuffer[5] + (UartRxBuffer[6]<<8);//运行次数
			FullActRun(fullActNum,times);
			break;

		case CMD_FULL_ACTION_STOP:
			FullActStop();
			break;

```

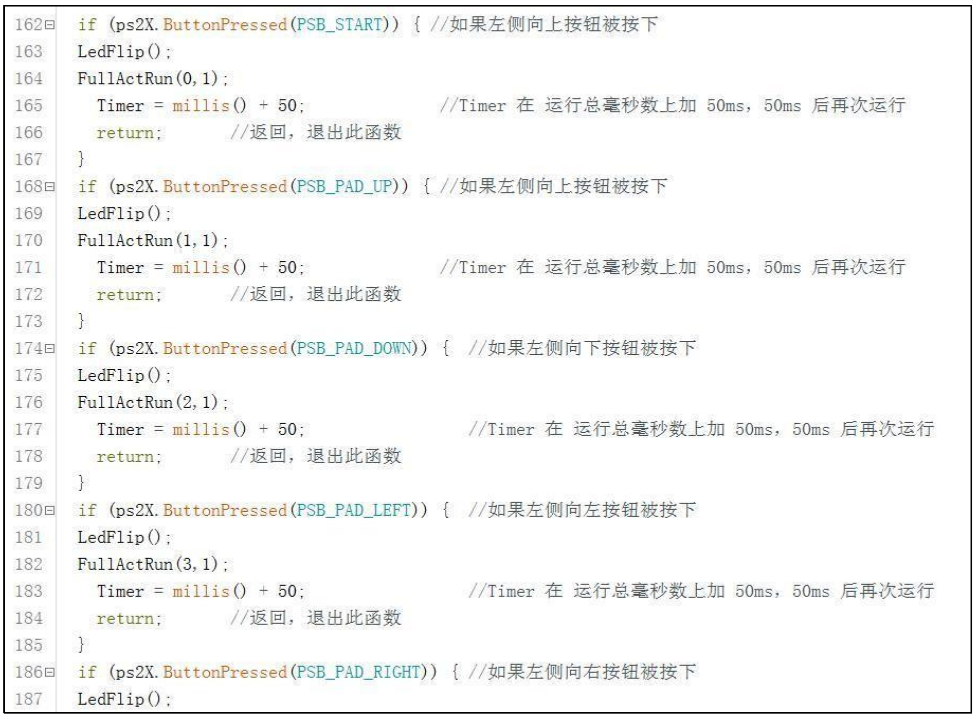
8) Stopping action group is to call the stop action group function. This function is to set several variables and signs so that the Task_RobotRun funciton will no

longer run the action group.



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

  fFrameRunFinish = TRUE;

  FrameIndex = 0;
}
```

9) Then read the button state of the PS2 handle in loop and perform different operations according to different button states. For example, when the up button is pressed, No.1 action group will be executed, and when the START button is pressed, No.0 action group is executed.



```
if (ps2X.ButtonPressed(PSB_START)) { //如果左侧向上按钮被按下
LedFlip();
FullActRun(0,1);
  Timer = millis() + 50;          //Timer 在 运行总毫秒数上加 50ms, 50ms 后再次运行
  return;       //返回, 退出此函数
}
if (ps2X.ButtonPressed(PSB_PAD_UP)) { //如果左侧向上按钮被按下
LedFlip();
FullActRun(1,1);
  Timer = millis() + 50;          //Timer 在 运行总毫秒数上加 50ms, 50ms 后再次运行
  return;       //返回, 退出此函数
}
if (ps2X.ButtonPressed(PSB_PAD_DOWN)) {  //如果左侧向下按钮被按下
LedFlip();
FullActRun(2,1);
  Timer = millis() + 50;             //Timer 在 运行总毫秒数上加 50ms, 50ms 后再次运行
  return;       //返回, 退出此函数
}
if (ps2X.ButtonPressed(PSB_PAD_LEFT)) {  //如果左侧向左按钮被按下
LedFlip();
FullActRun(3,1);
  Timer = millis() + 50;             //Timer 在 运行总毫秒数上加 50ms, 50ms 后再次运行
  return;       //返回, 退出此函数
}
if (ps2X.ButtonPressed(PSB_PAD_RIGHT)) { //如果左侧向右按钮被按下
LedFlip();
```

10) So far, the functions of uHand2.0 have been basically implemented.