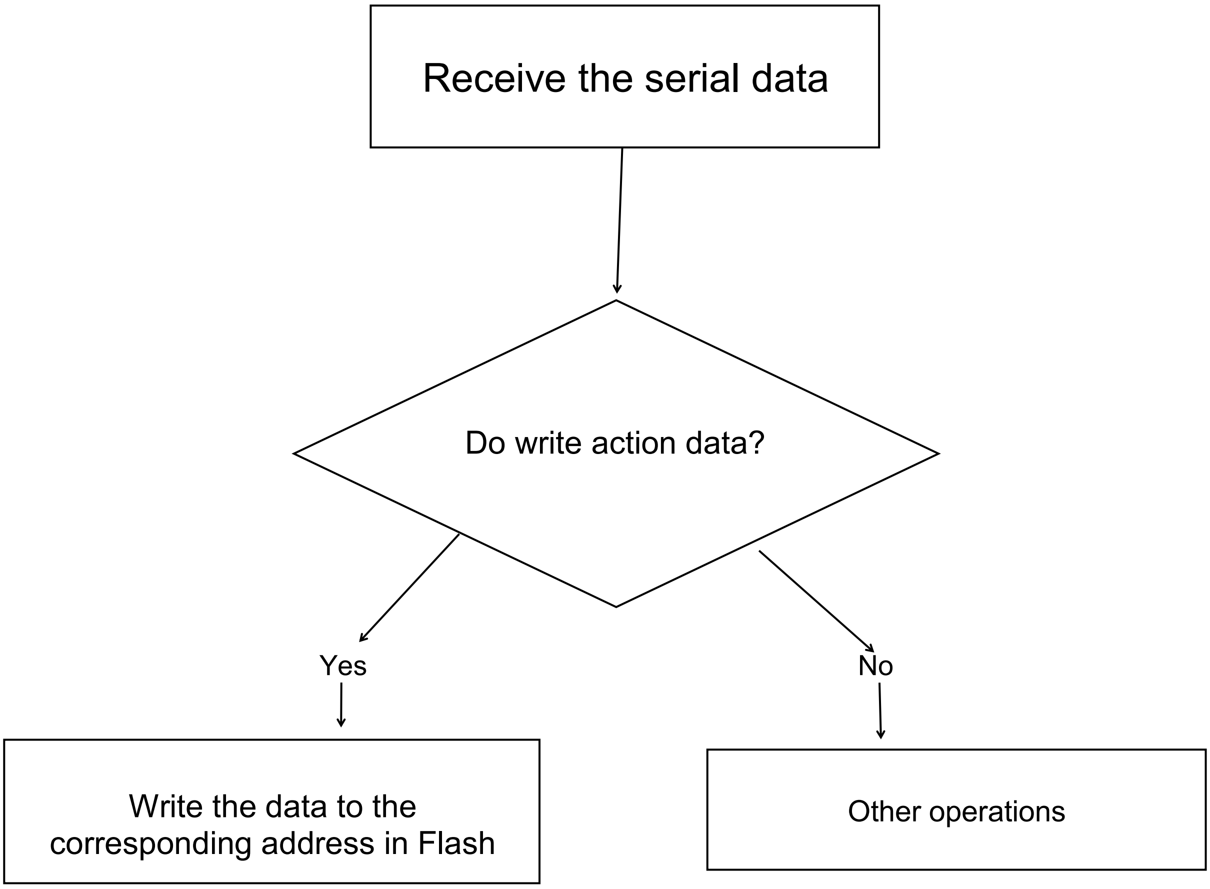
Lesson 10 Save Action to Flash

1. Project Purpose

Write the received action data to Flash through the serial communication.

2. Project Principle

We have learned the writing and reading of SPI Flash. In this section, we are going to save the action data into Flash. The working process is as follow:



Learning from the above, we need to extract the action data from the data received by the serial port, and then write it into Flash correctly. In the previous section, we learned that Flash erasing is sector-based, and at the same time only 1 can be written but not 0. Therefore, in order to facilitate programming and management, storage address is aligned with the sector size of the Flash chip saved by 4096 bytes, which improves the efficiency and reduce the error rate.

3. Program Analyst

```
void TaskBLEMsgHandle(void)
{

    uint16 i;
    uint8 cmd;
    uint8 id;
    uint8 servoCount;
    uint16 time;
    uint16 pos;
    uint16 times;
    uint8 fullActNum;
    if(UartRxOK())
    {
        LED = !LED;
        cmd = UartRxBuffer[3];
        switch(cmd)
        {
            case CMD_MULT_SERVO_MOVE:
                servoCount = UartRxBuffer[4];
                time = UartRxBuffer[5] + (UartRxBuffer[6]<<8);
                for(i = 0; i < servoCount; i++)
                {
                    id =  UartRxBuffer[7 + i * 3];
                    pos = UartRxBuffer[8 + i * 3] + (UartRxBuffer[9 + i * 3]<<8);

                    ServoSetPluseAndTime(id,pos,time);

                }
                break;

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

            case CMD_FULL_ACTION_STOP:
                FullActStop();
                break;

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

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

        }
    }
}
```

1) As you can see from the above figure, the action erase and the action download functions have been added. This two functions are implemented by two functions FlashEraseAll() and SaveAct(). The protocols of the two commands are as follows.

Command name: CMD_FULL_ACTION_ERASE

Command value: 8

Data Length: 3

Instruction: Erase parameter 1 of the action group downloaded to the control board: (reserved).

Return: the control board returns the command without parameters

Command name: CMD_SERVO_ DOWNLOAD

Command value: 25

Date Length: N： 46

Instruction: the action group is downloaded through the serial port, frame by frame, as many times as there are frames in that action group.

Parameter 1: the action group number to be downloaded to

Parameter 2: the total frame of the action group

Parameter 3: which frame of the data

Parameter 4: the number of the servos to be downloaded

Parameter 5: upper-byte of time

Parameter 6: lower-byte of time

Parameter 7: Servo ID number

Parameter 8: lower-byte of angle position

Parameter 9: the upper eight bits of the angle position. Parameter......: The format is the same as that of parameters 7, 8, and 9, the angular position of the different ID. For each frame of data downloaded, the controller returns data with the same command value, but as a command packet without parameters.

2) Before implementing these two functions, the storage location of the data in Flash needs to be arranged first. The following figure shows address distribution of data storage:

The first sector is used to store LOBOT signs; If there is no LOBOT sign, the sector will be considered to be a new Flash chip;

3) The second sector stores the number of actions of the action group. Each action group occupies one byte, that is, each action group can store 255 actions. We only use the first 256 bytes of this sector, that is, up 256 action groups can be stored. All 256 bytes of the data is read during operating. Then erase the data of this sector and rewrite the modified data into.

4) Starting from the this sector is the area for storing action group data. Each action group occupies 16KB, that is, four sectors. The process of writing an action group is as follows:

The first step: erase the storage area 48 of the action group to be written.

The second step: Write the actions of the action group one by one. At the same time, check if the written action is the last action in this action group.

The third step: If it is the last action, we consider that the action has been written completely, and then update the number of the action groups in the second sector. The way to erase an action is to change the number of the actions of the corresponding action group to 0. The way to erase all action groups is to change the first 256 bytes of the second sector to 0. The following figure shows the implementation of this part of the program:

```
void SaveAct(uint8 fullActNum,uint8 frameIndexSum,uint8 frameIndex,uint8* pBuffer)
{
    uint8 i;

    if(frameIndex == 0)//Erase this action group first before downloading
    {// An action group occupies a size of 16k, and erase a sector is 4k, so it needs to be erased 4 times
        for(i = 0;i < 4;i++)//ACT_SUB_FRAME_SIZE/4096 = 4
        {
            FlashEraseSector((MEM_ACT_FULL_BASE) + (fullActNum * ACT_FULL_SIZE) + (i * 4096));
        }
    }

    FlashWrite((MEM_ACT_FULL_BASE) + (fullActNum * ACT_FULL_SIZE) + (frameIndex * ACT_SUB_FRAME_SIZE)
        ,ACT_SUB_FRAME_SIZE,pBuffer);

    if((frameIndex + 1) ==  frameIndexSum)
    {
        FlashRead(MEM_FRAME_INDEX_SUM_BASE,256,frameIndexSumSum);
        frameIndexSumSum[fullActNum] = frameIndexSum;
        FlashEraseSector(MEM_FRAME_INDEX_SUM_BASE);
        FlashWrite(MEM_FRAME_INDEX_SUM_BASE,256,frameIndexSumSum);
    }
}


void FlashEraseAll(void)
{//Set the number of actions of all 255 action groups to 0, which means to erase all action groups
    uint16 i;

    for(i = 0;i <= 255;i++)
    {
        frameIndexSumSum[i] = 0;
    }
    FlashEraseSector(MEM_FRAME_INDEX_SUM_BASE);
    FlashWrite(MEM_FRAME_INDEX_SUM_BASE,256,frameIndexSumSum);
}
```

5) The InitMemory function is used to clear some unused storage in the Flash. Then write a loge to mark that the Flash has been initialized.

```
void InitMemory(void)
{
    uint8 i;
    uint8 logo[] = "LOBOT";
    uint8 datatemp[8];

    FlashRead(MEM_LOBOT_LOGO_BASE,5,datatemp);
    for(i = 0; i < 5; i++)
    {
        if(logo[i] != datatemp[i])
        {
        LED = LED_ON;
            //If it is found to be unequal, it means that it is a new FLASH and needs to be initialized
            FlashEraseSector(MEM_LOBOT_LOGO_BASE);
            FlashWrite(MEM_LOBOT_LOGO_BASE,5,logo);
            FlashEraseAll();
            break;
        }
    }

}
```