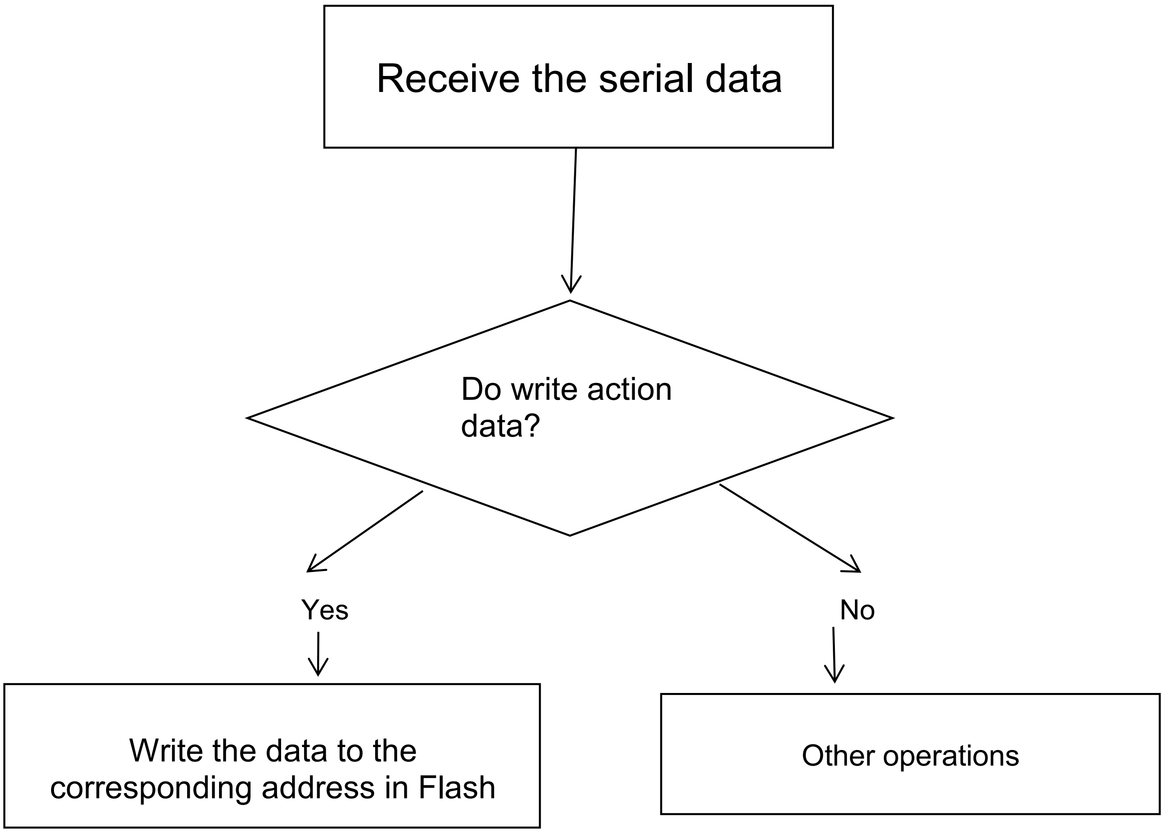
Lesson 9 Save Action to Flash

1. Project Purpose

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

2. Project Principle

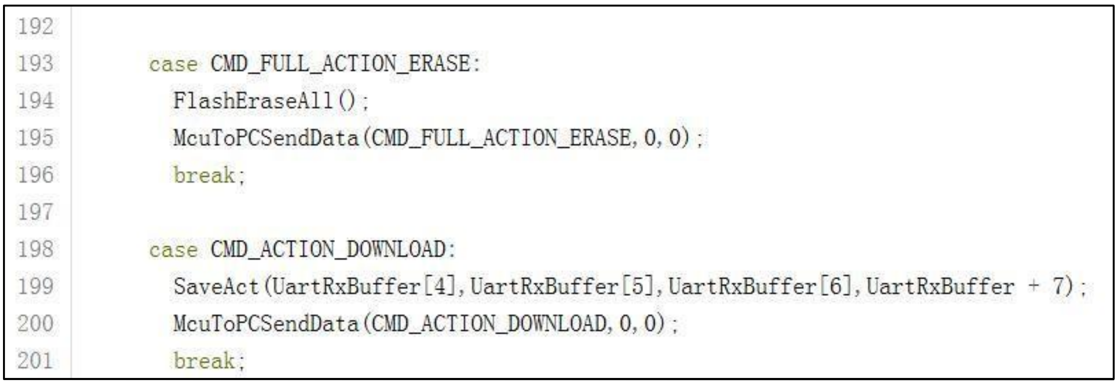
We have learned the writing and reading of SPI Flash. In this section, we are going to save the action data into Flash. The working process is as follow:



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



```
        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;
```

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.

Action Erase

Command name: CMD_FULL_ACTION_ERASE

Command value: 8

Date Length: 3

Instruction: erase the action group downloaded in the controller.

Parameter 1: (Reserved)

Return: the control board returns the command without parameters

Action download command

Command name CMD_ACTION_ 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.

Data Length: N=the number of the downloaded servos×3+8

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: the number of the servos to be downloaded

Parameter 5: upper-byte of time

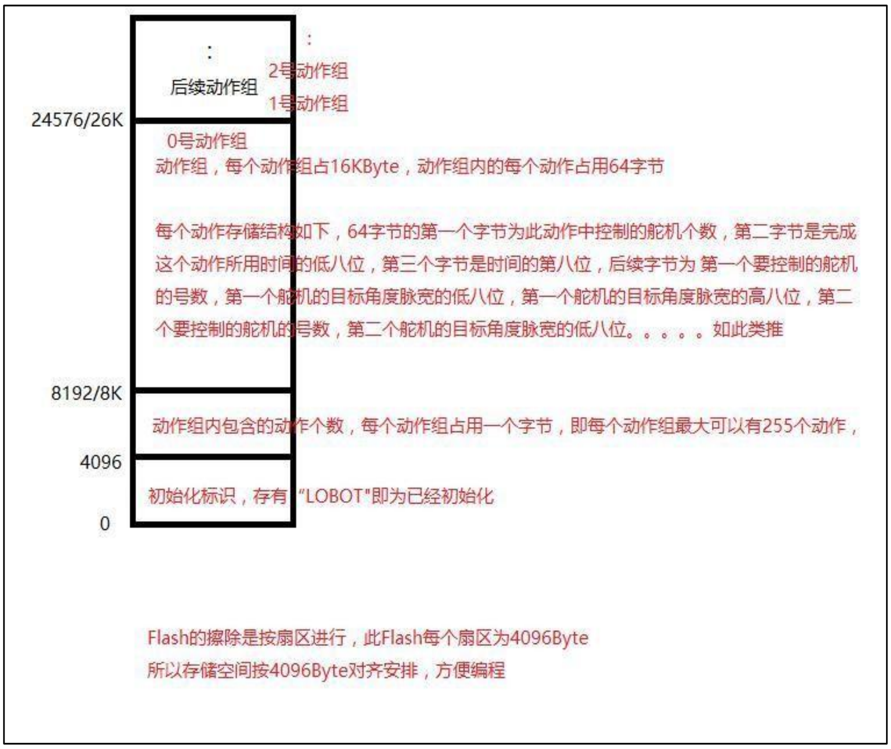
Parameter 6: lower-byte of time

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.

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:



动作组，每个动作组占16KByte，动作组内的每个动作占用64字节

每个动作存储结构如下，64字节的第一个字节为此动作中控制的舵机个数，第二字节是完成这个动作所用时间的低八位，第三个字节是时间的第八位，后续字节为 第一个要控制的舵机的号数，第一个舵机的目标角度脉宽的低八位，第一个舵机的目标角度脉宽的高八位，第二个要控制的舵机的号数，第二个舵机的目标角度脉宽的低八位。。。。。如此类推

动作组内包含的动作个数，每个动作组占用一个字节，即每个动作组最大可以有255个动作，

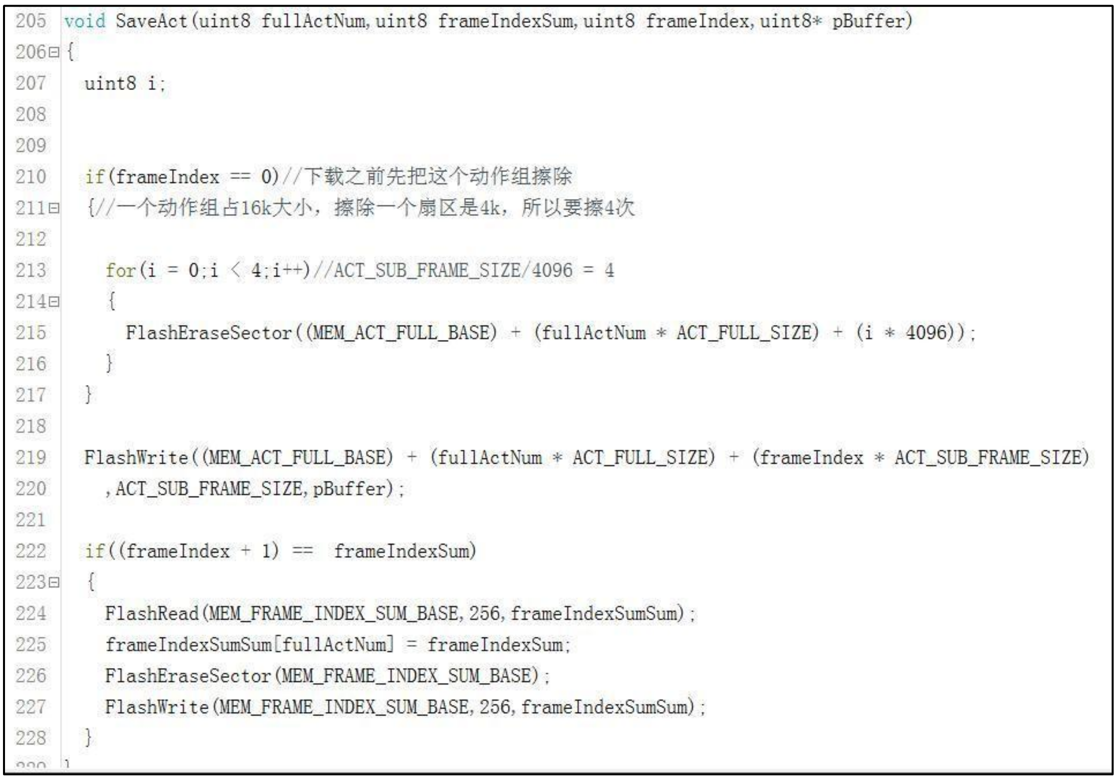
初始化标识，存有"LOBOT"即为已经初始化

Flash的擦除是按扇区进行，此Flash每个扇区为4096Byte
所以存储空间按4096Byte对齐安排，方便编程

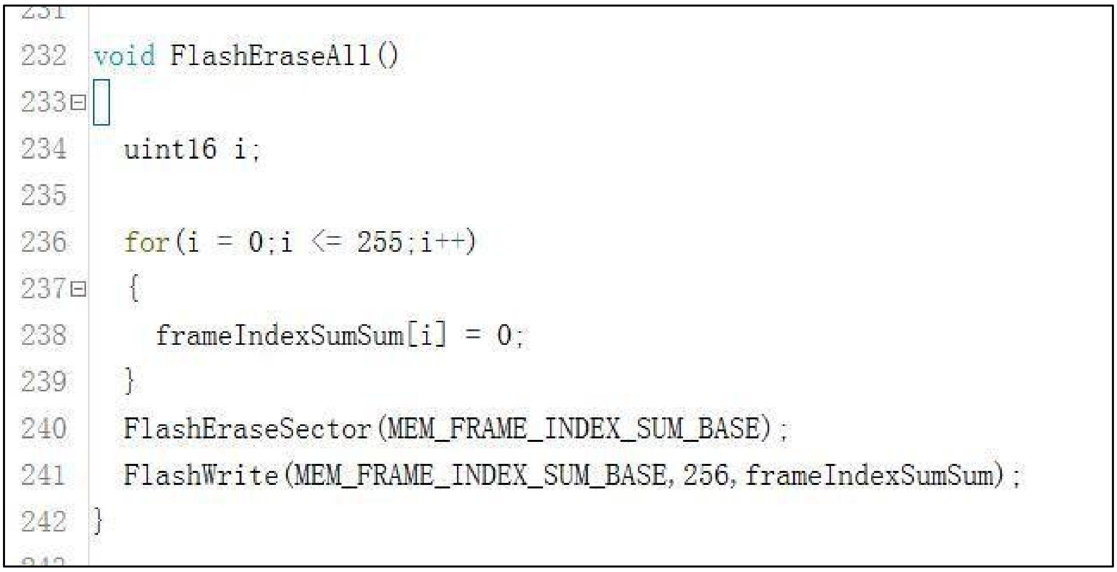
1) The first sector is used to store LOBOT signs; If there is no LOBOT sign, the sector will be considered to be a new Flash chip;

2) 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.

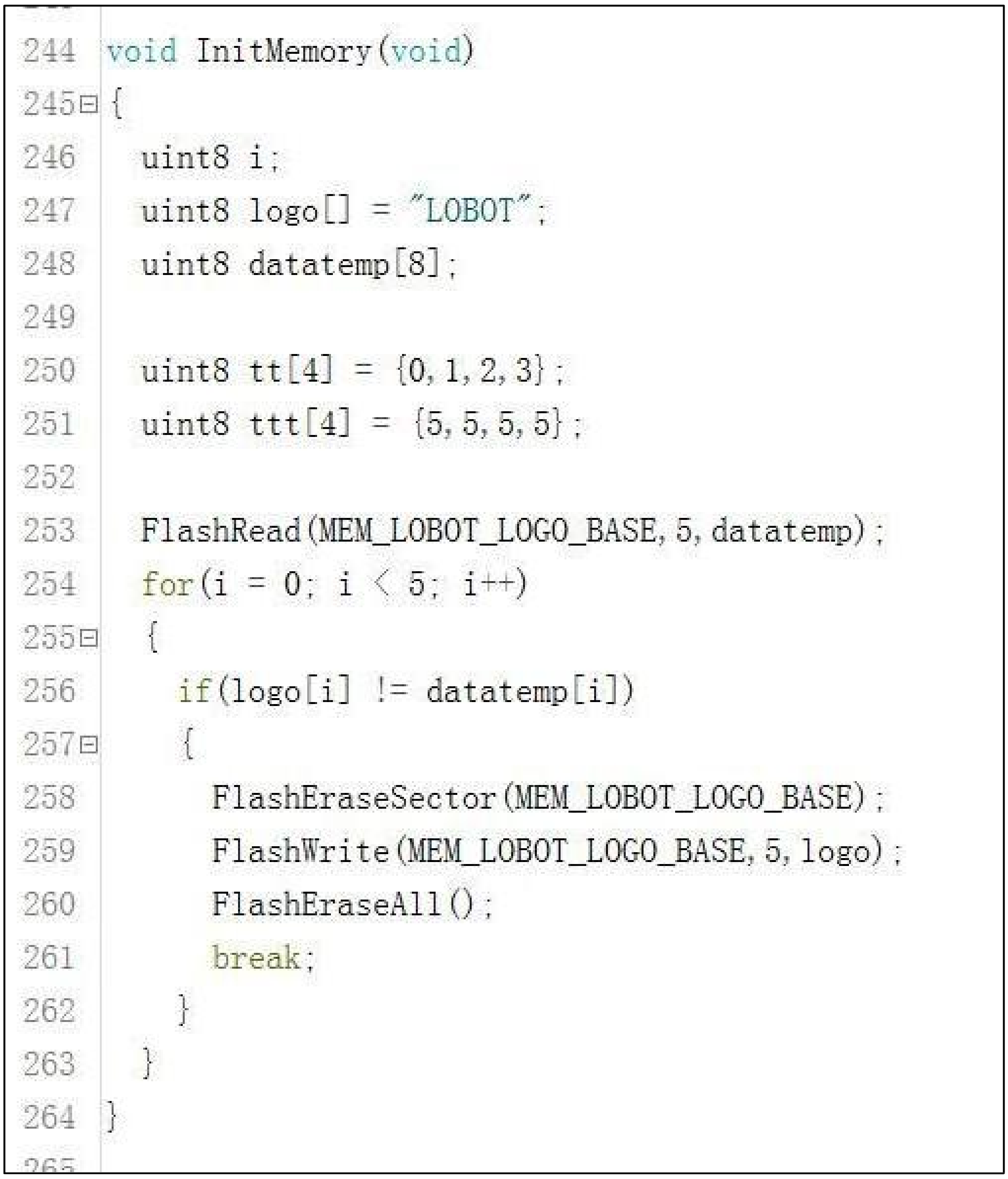
3) All 256 bytes of the data is read during operating. Then erase the data of this sector and rewrite the modified data into. Starting from the this sector is the area for storing action group data. Each action group occupies 16KB, that is, four sectors. The process of writing an action group is as follows:
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)//下载之前先把这个动作组擦除
  {//一个动作组占16k大小，擦除一个扇区是4k，所以要擦4次

    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()
{
  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);
}
```



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

  uint8 tt[4] = {0,1,2,3};
  uint8 ttt[4] = {5,5,5,5};

  FlashRead(MEM_LOBOT_LOGO_BASE,5,datatemp);
  for(i = 0; i < 5; i++)
  {
    if(logo[i] != datatemp[i])
    {
      FlashEraseSector(MEM_LOBOT_LOGO_BASE);
      FlashWrite(MEM_LOBOT_LOGO_BASE,5,logo);
      FlashEraseAll();
      break;
    }
  }
}
```