

优秀不够，你是否无可替代

知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 821, 文章 - 0, 评论 - 327, 阅读 - 202万

导航

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[订阅](#)
[管理](#)

公告

渡我不渡她 -
Not available
00:00 / 03:41

- 渡我不渡她
- 小镇姑娘
- PDD洪荒之力

加入QQ群

昵称：杨奉武
 园龄：6年2个月
 粉丝：693
 关注：1

搜索

我的标签

[8266\(88\)](#)
[MQTT\(50\)](#)
[GPRS\(33\)](#)
[SDK\(29\)](#)
[Air202\(28\)](#)
[云服务器\(21\)](#)
[ESP8266\(21\)](#)
[Lua\(18\)](#)
[小程序\(17\)](#)
[STM32\(16\)](#)
[更多](#)

随笔分类

[Air724UG学习开发\(8\)](#)
[Android\(22\)](#)
[Android 开发\(8\)](#)
[C# 开发\(4\)](#)
[CH395Q学习开发\(17\)](#)
[CH573F学习开发\(1\)](#)
[CH579M物联网开发\(12\)](#)
[CH579M学习开发\(8\)](#)
[ESP32学习开发\(30\)](#)
[ESP8266 AT指令开发\(基于STC89C52单片机\)\(3\)](#)
[ESP8266 AT指令开发\(基于STM32\)\(1\)](#)
[ESP8266 AT指令开发基础入门篇备份\(12\)](#)
[ESP8266 LUA脚本语言开发\(13\)](#)

17-3-华大单片机HC32F460系列flash存储方案

<p>
 <iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/单片机知识点总结/directory.html" frameborder="0" scrolling="auto" width="100%" height="1500">
 </iframe>
 </p>

数据处理思想和程序架构

资料源

码:<https://gitee.com/yang456/OpenProgrammingModule>

点击加入群聊【单片机,物联网,上位机】： 加入QQ群

说明1:知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系！

说明2:学的是思想，而非程序！此代码思路适用于所有的单片机。

说明3:学会以后,下面的代码可能会跟你一辈子！

说明4:这一系列文章是为大幅度裁剪本人博客文章！使博客文章更有条理。便于推其它教程！

目录:

- [01-来看下我的程序架子吧](#)
- [02-看看是不是你想要的按键处理](#)
- [03-单片机接收数据之空闲中断](#)
- [04_1-关于环形队列](#)
- [04_2-单片机接收数据之环形队列](#)
- [05-单片机接收数据之缓存管理,DMA](#)
- [06-单片机发送数据之中断发送](#)
- [07-单片机发送数据之环形队列](#)
- [08-单片机发送数据之缓存管理,DMA](#)
- [09-μCOS-II中内存管理程序使用说明](#)
- [10-数据缓存封装-内存管理实现](#)
- [11-给单片机写个回调函数怎么样](#)
- [12-单片机AT指令配置模块程序模板\(阻塞版\)](#)

ESP8266 LUA开发基础入门篇
备份(22)
ESP8266 SDK开发(33)
ESP8266 SDK开发基础入门篇
备份(30)
GPRS Air202 LUA开发(11)
HC32F460(华大单片机)物联网
开发(10)
HC32F460(华大单片机)学习开
发(8)
NB-IOT Air302 AT指令和LUA
脚本语言开发(27)
PLC(三菱PLC)基础入门篇(2)
STM32+Air724UG(4G模组)
物联网开发(43)
STM32+BC26/260Y物联网开
发(37)
STM32+CH395Q(以太网)物
联网开发(24)
STM32+ESP8266(ZLESP826
6A)物联网开发(1)
STM32+ESP8266+AIR202/3
02远程升级方案(16)
STM32+ESP8266+AIR202/3
02终端管理方案(6)
STM32+ESP8266+Air302物
联网开发(65)
STM32+W5500+AIR202/30
2基本控制方案(25)
STM32+W5500+AIR202/30
2远程升级方案(6)
UCOSii操作系统(1)
W5500 学习开发(8)
编程语言C#(11)
编程语言Lua脚本语言基础入
门篇(6)
编程语言Python(1)
单片机(LPC1778)LPC1778(2)
单片机(MSP430)开发基础入门
篇(4)
单片机(STC89C51)单片机开发
板学习入门篇(3)
单片机(STM32)基础入门篇(3)
单片机(STM32)综合应用系列
(16)
更多

阅读排行榜

1. ESP8266使用详解(AT,LUA, SDK)(174403)
2. 1-安装MQTT服务器(Windo ws),并连接测试(105491)
3. 用ESP8266+android,制作 自己的WIFI小车(ESP8266篇) (68025)
4. ESP8266刷AT固件与node mcu固件(67039)
5. 有人WIFI模块使用详解(394 98)
6. (一)基于阿里云的MQTT远 程控制(Android 连接MQTT服 务器,ESP8266连接MQTT服 务器实现远程通信控制----简单 的连接通信)(37155)
7. C#中public与private与stat ic(36173)
8. 关于TCP和MQTT之间的转 换(35432)
9. android 之TCP客户端编程 (33102)
10. android客服端+eps8266 +单片机+路由器之远程控制系 统(31724)

推荐排行榜

- [13-单片机AT指令配置模块程序模板 \(非阻塞版\)](#)
- [14-单片机加入JSON是个不错的选择](#)
- [15-IEEE754规约,浮点数和16进制之 间的转换](#)
- [16-CRC校验](#)
- [17-1-单片机stm32的flash保存数据 优化方案\(让擦写次数达到上百万至上 千万次\)](#)
- [17-2-单片机 STM32F407xx,F405xx,F415xx,417xx 系列flash存储方案](#)
- [17-3-华大单片机,HC32F460系列 flash存储方案](#)
- [18-关于SSL](#)
- [19-单片机移植Mbedtls](#)
- [20-使用Mbedtls包中的SSL,和服务器 进行网络加密通信](#)

说明

因为用到了HC32F460系列的单片机的flash存储数据,所以写了这套程序.

目的是为了在便于存储.

关于407的Flash

- 1. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(9)
- 2. C#委托+回调详解(9)
- 3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
- 4. 我的大学四年(6)
- 5. ESP8266使用详解(AT,LUA, SDK)(6)

最新评论

- 1. Re:2-6-1-视频传输,监控,直播方案-手机连接ESP32的热点,使用微信小程序查看摄像头图像(WiFi视频小车,局域网视频监控)
赞赞赞，感谢大佬无私奉献
--SJA2C2A
- 2. Re:中移动M5311模块使用手册(TCP,MQTT)
请问你用的usb转ttl是哪一种呢，我用的ch340可是开机串口助手没有SIM识别显示
--夏洛的网娅

FLASH 具有以下主要特性：

- 容量高达 512 KBytes(其中有 32bytes 为功能保留位)
分为 64 个扇区，每个扇区为 8KBytes。
- OTP(One Time Program)区域共 1020Bytes，分为 960Bytes 数据区，并配有 60Bytes 的锁存区。
- 128 位宽数据读取
- 编程单位为 4bytes，擦除单位为 8Kbytes

在 512KB 产品中，FLASH 地址结构如下图所示。

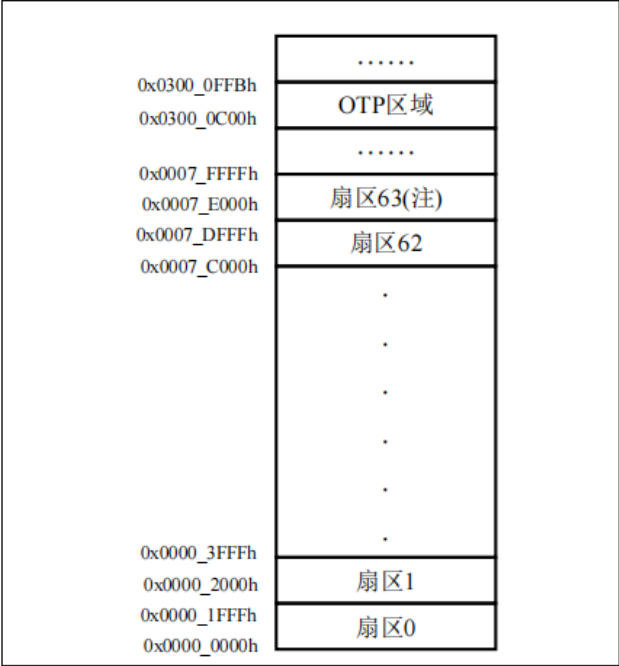


图 9-1 FLASH 地址结构（512KB 产品）

注意：

- 扇区 63 中地址 0x0007_FFE0~0x0007_FFFF 共 32Bytes 为功能保留地址；对这 32Bytes 地址进行编程、扇区擦除、全擦除，FLASH 数据不会改变，对这些地址读，读到数据为全 1。

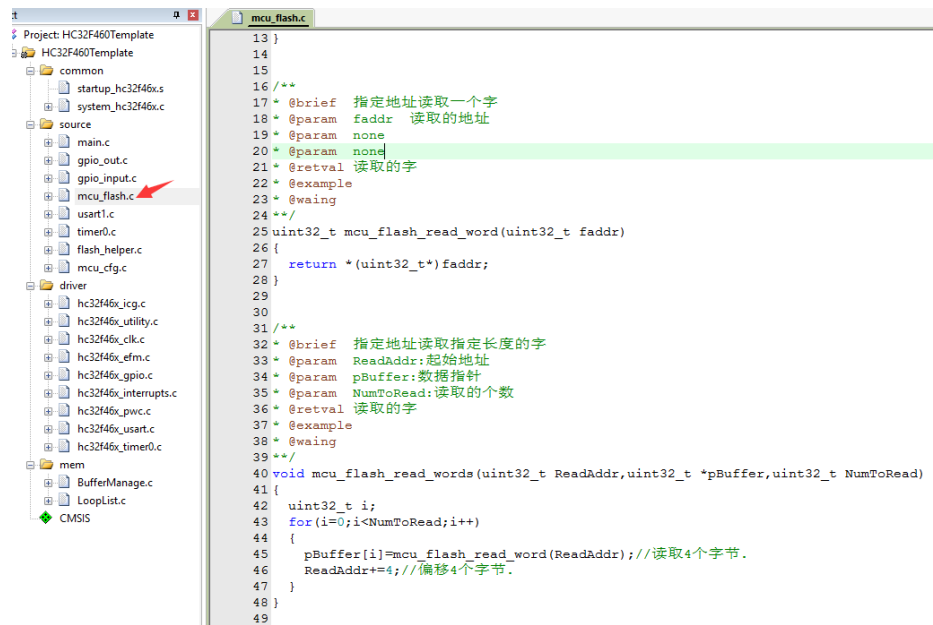
写数据时最小写的单位是 4字节

擦除的时候,最小擦除的单位是1个扇区(8KB)

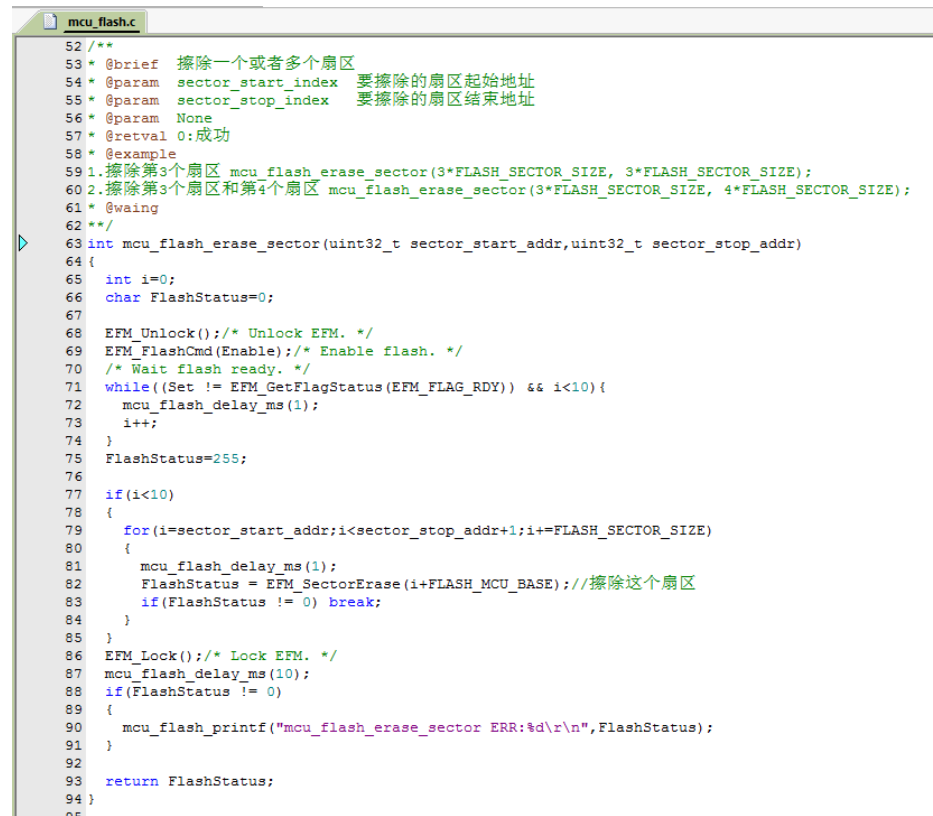
读取数据的时候最大读取的位宽是128位(16字节)

关于提供的基本的flash操作函数

1.读取



2.擦除



3.写入

mcu_flash.c

```

97
98
99 /**
100 * @brief 指定地址写入一个字(u32)(需要事先擦除才可使用此函数)
101 * @param WriteAddr 开始写入的地址
102 * @param data 写入的数据
103 * @param none
104 * @retval 0:成功
105 * @example
106 * @waing
107 */
108 int mcu_flash_write_word(uint32_t WriteAddr,uint32_t data)
109 {
110     int i=0;
111     char FlashStatus = 0;
112     mcu_flash_delay_us(5);
113     EFM_Unlock();/* Unlock EFM. */
114     EFM_FlashCmd(Enable);/* Enable flash. */
115     i=0;
116     while((Set != EFM_GetFlagStatus(EFM_FLAG_RDY)) && i<5000){
117         mcu_flash_delay_us(1);
118         i++;
119     }
120     FlashStatus=255;
121
122     if(i<5000)
123     {
124         FlashStatus = EFM_SingleProgram(WriteAddr,data);/*写入数据
125     }
126     EFM_Lock();/* Lock EFM. */
127     mcu_flash_delay_us(5);
128
129     if(FlashStatus != 0) {
130         mcu_flash_printf("mcu_flash_write_word ERR:%d\r\n",FlashStatus);
131     }
132
133     return FlashStatus;
134 }
135

```

mcu_flash.c

```

136 /**
137 * @brief 指定地址写入指定长度的字(需要事先擦除才可使用此函数)
138 * @param WriteAddr 开始写入的地址
139 * @param pBuffer 写入的数据
140 * @param NumToWrite 写入的数据个数
141 * @retval 0:成功
142 * @example
143 写数据到0x0007C000地址
144 uint32_t data[255];
145 mcu_flash_write_words(0x0007C000,data,255);
146 * @waing
147 */
148 int mcu_flash_write_words(uint32_t WriteAddr,uint32_t *pBuffer,uint32_t NumToWrite)
149 {
150     int i=0;
151     char FlashStatus = 0;
152     mcu_flash_delay_us(5);
153     EFM_Unlock();/* Unlock EFM. */
154     EFM_FlashCmd(Enable);/* Enable flash. */
155     i=0;
156     while((Set != EFM_GetFlagStatus(EFM_FLAG_RDY)) && i<5000){
157         mcu_flash_delay_us(1);
158         i++;
159     }
160     FlashStatus=255;
161     if(i<5000)
162     {
163         for(i=0;i<NumToWrite;i++)
164         {
165             FlashStatus = EFM_SingleProgram(WriteAddr,pBuffer[i]);/*写入数据
166             if(FlashStatus != 0) break;
167             WriteAddr = WriteAddr + 4;
168         }
169     }
170
171     EFM_Lock();/* Lock EFM. */
172     mcu_flash_delay_us(5);
173     if(FlashStatus != 0) {
174         mcu_flash_printf("mcu_flash_write_words ERR:%d\r\n",FlashStatus);
175     }
176     return FlashStatus;
177 }
178

```

关于本节封装的Flash写读函数思路

1.使用其中一个扇区存储数据,假设使用11扇区,准备一个固定大小(假设是512)的u32类型的数组用来存储数据

数组



整个11号扇区

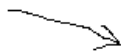
记住一句话:只要擦除过了的地方,不需要再次擦除就可以写入数据

在下载程序的时候,默认所有的flash都是已经擦除过的.

2.第一次存储

把整个数组全部写到扇区的最前面

数组



整个11号扇区

3.第二次存储(不需要擦除,直接写入就可以,因为后面都已经擦除过了)

把整个数组紧接着写到后面

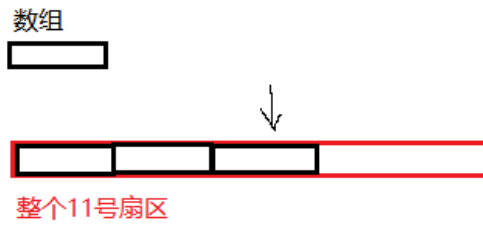
数组



整个11号扇区

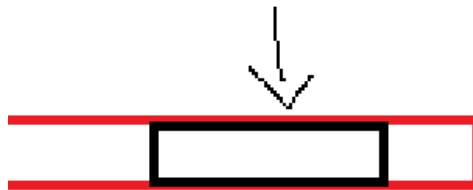
4.第三次存储(不需要擦除,直接写入就可以,因为后面都已经擦除过了)

把整个数组紧接着写到后面



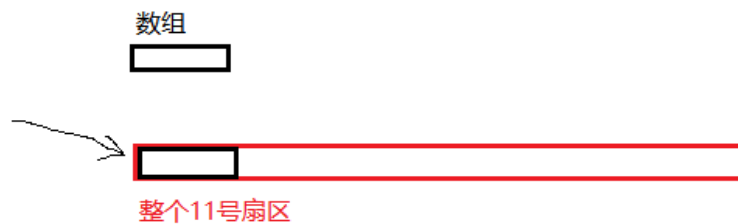
5.假设写到最后了

我后面空了一点是因为定义的数组大小不一样,不一定正好占满.



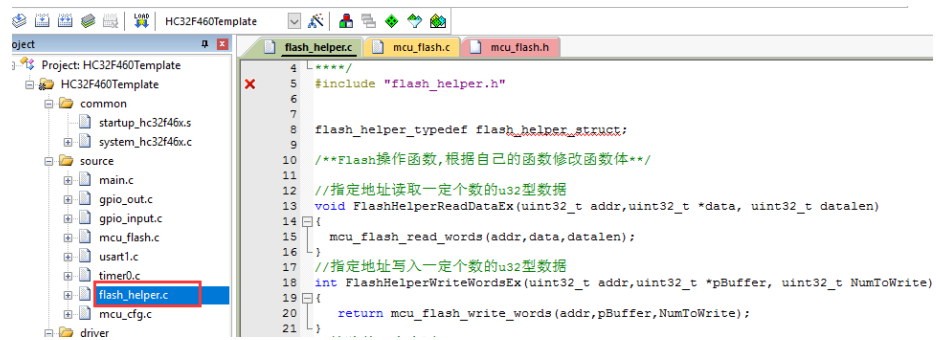
6.接着再写

检测不够了以后,擦除下这个扇区,把数据从头开始存储.

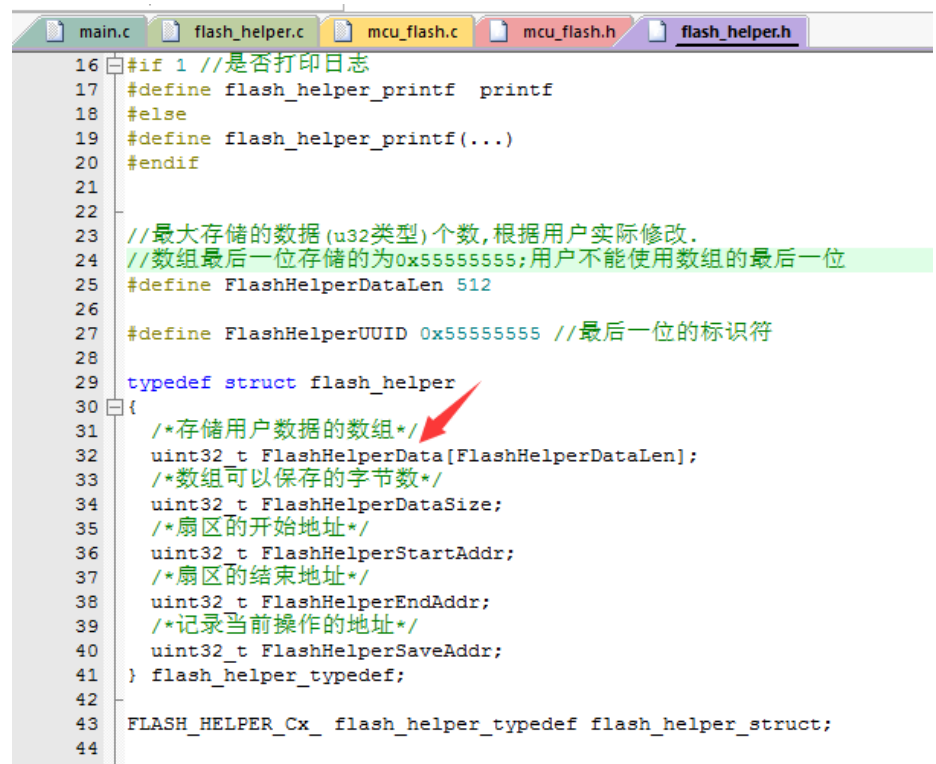


源码使用

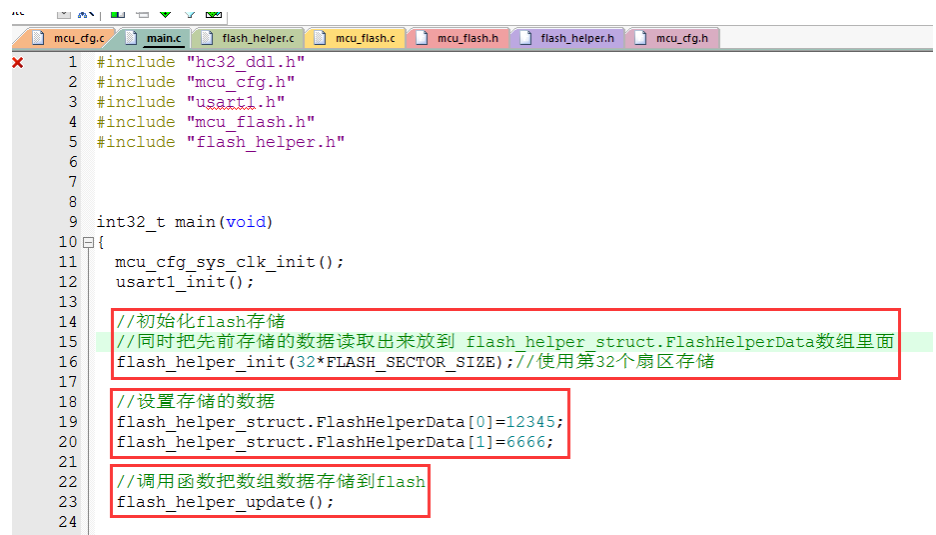
1.我先说一下flash_helper 的使用



2.提示:存储数据的时候是把数据放到这个数组里面



3.使用flash_helper存储数据





```
//初始化flash存储
//同时把先前存储的数据读取出来放到 flash_helper_struct.FlashHelperData数组里面
flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储

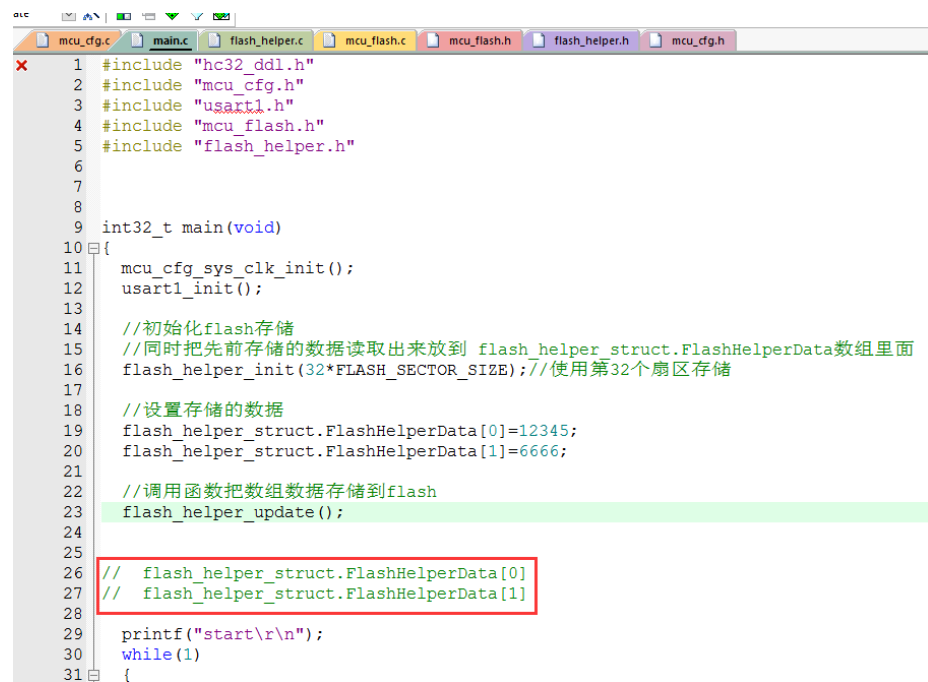
//设置存储的数据
flash_helper_struct.FlashHelperData[0]=12345;
flash_helper_struct.FlashHelperData[1]=6666;

//调用函数把数组数据存储到flash
flash_helper_update();
```



4. 读取存储的数据(直接从数组获取) 推荐

因为数组里面的值就是存储在flash里面的,所以数组里面的值就是存储在flash里面的值



```
1 #include "hc32_ddl.h"
2 #include "mcu_cfg.h"
3 #include "usart1.h"
4 #include "mcu_flash.h"
5 #include "flash_helper.h"
6
7
8
9 int32_t main(void)
10 {
11     mcu_cfg_sys_clk_init();
12     usart1_init();
13
14     //初始化flash存储
15     //同时把先前存储的数据读取出来放到 flash_helper_struct.FlashHelperData数组里面
16     flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储
17
18     //设置存储的数据
19     flash_helper_struct.FlashHelperData[0]=12345;
20     flash_helper_struct.FlashHelperData[1]=6666;
21
22     //调用函数把数组数据存储到flash
23     flash_helper_update();
24
25
26     // flash_helper_struct.FlashHelperData[0]
27     // flash_helper_struct.FlashHelperData[1]
28
29     printf("start\r\n");
30     while(1)
31     {
```

```
flash_helper_struct.FlashHelperData[0]
```

```
flash_helper_struct.FlashHelperData[1]
```

5. 读取存储的数据(从flash里面获取)

```
mcu_cfg.c main.c flash_helper.c mcu_flash.c mcu_flash.h flash_helper.h mcu_cfg.h
147 {
148     flash_helper_printf("flash_helper_update_WriteWords ERR\r\n");
149     return 2;
150 }
151 return 0;
152 }
153
154
155 /**
156  * @brief  获取缓存的数据
157  * @param  index 和数组的下标对应
158  * @param  data 返回的数据
159  * @param  datalen 获取的个数
160  * @retval 0:成功
161  * @warning
162  * @example
163  */
164 int flash_helper_get(uint32_t index, uint32_t *data, uint32_t datalen)
165 {
166     //最大获取的个数不能超出实际最大能返回的个数
167     if(datalen > FlashHelperDataLen - index) datalen = FlashHelperDataLen - index;
168     FlashHelperReadDataEx(flash_helper_struct.FlashHelperSaveAddr+(index<<2), data, datalen);
169     return 0;
170 }
171
```

这个函数一般用于查看下存储的对不对, 不过需要是调用完 flash_helper_update函数以后, 然后接着使用

例如, 获取刚刚数组的第一位存储在flash里面的值

```
ite
mcu_cfg.c main.c flash_helper.c mcu_flash.c mcu_flash.h flash_helper.h mcu_cfg.h
4 #include "mcu_flash.h"
5 #include "flash_helper.h"
6
7
8 uint32_t data;
9
10 int32_t main(void)
11 {
12     mcu_cfg_sys_clk_init();
13     usart1_init();
14
15     //初始化flash存储
16     //同时把先前存储的数据读取出来放到 flash_helper_struct.FlashHelperData数组里面
17     flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储
18
19     //设置存储的数据
20     flash_helper_struct.FlashHelperData[0]=12345;
21     flash_helper_struct.FlashHelperData[1]=6666;
22
23     //调用函数把数组数据存储到flash
24     flash_helper_update();
25
26     // flash_helper_struct.FlashHelperData[0]
27     // flash_helper_struct.FlashHelperData[1]
28     flash_helper_get(0, &data, 1);
29     printf("get0=%d\r\n", data);
30
31     printf("start\r\n");
32     while(1)
33     {
34
35

```

```
uint32_t data;

flash_helper_get(0, &data, 1);

printf("get0=%d\r\n", data);
```

```
sectors_addr_start: 0x40000
sectors_addr_stop: 0x42000
data:0xFFFFFFFF addr: 0x040000
flash_helper_update FlashHelperSaveAddr:40800
get0=12345
start
```

例如,获取多个(下面的例子是获取前两位)

```
mcu_cfg.c main.c flash_helper.c mcu_flash.c mcu_flash.h flash_helper.h mcu_cfg.h
4 #include "mcu_flash.h"
5 #include "flash_helper.h"
6
7
8 uint32_t data_temp[2];
9
10 int32_t main(void)
11 {
12     mcu_cfg_sys_clk_init();
13     usart1_init();
14
15     //初始化flash存储
16     //同时把先前存储的数据读取出来放到 flash_helper_struct.Flash
17     flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储
18
19     //设置存储的数据
20     flash_helper_struct.FlashHelperData[0]=12345;
21     flash_helper_struct.FlashHelperData[1]=6666;
22
23     //调用函数把数组数据存储到flash
24     flash_helper_update();
25
26     // flash_helper_struct.FlashHelperData[0]
27     // flash_helper_struct.FlashHelperData[1]
28
29     flash_helper_get(0,data_temp,2);
30     printf("data=%d %d \r\n",data_temp[0], data_temp[1]);
31
32     printf("start\r\n");
33     while(1)
```

```
sectors_addr_start: 0x40000
sectors_addr_stop: 0x42000
data:0xFFFFFFFF addr: 0x040000
flash_helper_update FlashHelperSaveAddr:40800
data=12345 6666
start
```

推荐使用方式(新建一个falsh_data文件)

1. 存储数据如果一个u32就存储一个数据显得有些浪费, 咱们应该使用共用体来操作一下

```
flash_helper.c  flash_data.c  main.c  flash_data.h
1 #ifndef flash_data_h_
2 #define flash_data_h_
3
4 #ifndef flash_data_c_ //如果没有定义
5 #define flash_data_cx_ extern
6 #else
7 #define flash_data_cx_
8 #endif
9
10
11 #include <string.h>
12 #include <stdio.h>
13 #include "flash_helper.h"
14
15 typedef union union_flash_data
16 {
17     uint32_t uint32_data;
18     char char_temp[4];
19 } union_flash_data_value;
20
21 flash_data_cx_ union_flash_data_value union_flash_data_value1;
22
23 #endif
```

2. 然后举个例子(让一个u32可以存储多个数据)

```
flash_helper.c  flash_data.c  main.c  flash_data.h
3 documents:
4 ****/
5 #include "flash_data.h"
6
7 //假设 flash_helper_struct.FlashHelperData[0] ;的高8位, 中16位 低8位各使用来存储数据
8
9 union_flash_data_value union_flash_data_value1;
10
11 /**演示存储char*/
12 void flash_data_set_char_data(char data){
13     //获取先前存储的数据
14     union_flash_data_value1.uint32_data = flash_helper_struct.FlashHelperData[0];
15     //更新下那一位数据
16     union_flash_data_value1.char_temp[0] = data;
17     //把更新过的数据重新赋值
18     flash_helper_struct.FlashHelperData[0] = union_flash_data_value1.uint32_data;
19     //刷新
20     flash_helper_update();
21 }
22 /**演示存储u16*/
23 void flash_data_set_u16_data(uint16_t data){
24     //获取先前存储的数据
25     union_flash_data_value1.uint32_data = flash_helper_struct.FlashHelperData[0];
26     //更新下那一位数据
27     union_flash_data_value1.char_temp[1] = (data>>8)&0xff;
28     union_flash_data_value1.char_temp[2] = data&0xff;
29
30     //把更新过的数据重新赋值
31     flash_helper_struct.FlashHelperData[0] = union_flash_data_value1.uint32_data;
32     //刷新
33     flash_helper_update();
34 }
35 /**演示获取char*/
36 char flash_data_get_char_data(){
37     //获取先前存储的数据
38     union_flash_data_value1.uint32_data = flash_helper_struct.FlashHelperData[0];
39     //返回数据
40     return union_flash_data_value1.char_temp[0];
41 }
42 /**演示获取u16*/
43 uint16_t flash_data_get_u16_data(){
44     uint16_t data;
45     //获取先前存储的数据
46     union_flash_data_value1.uint32_data = flash_helper_struct.FlashHelperData[0];
47     //返回数据
48     data = union_flash_data_value1.char_temp[1];
49     data = data<<8;
50     data = data + union_flash_data_value1.char_temp[2];
51     return data;
52 }
53
```

```
ste
flash_helper.c flash_data.c main.c flash_data.h flash_helper.h
1 #include "hc32_ddl.h"
2 #include "mcu_cfg.h"
3 #include "usart1.h"
4 #include "mcu_flash.h"
5 #include "flash_helper.h"
6 #include "flash_data.h"
7
8 uint32_t data_temp[2];
9
10 int32_t main(void)
11 {
12     mcu_cfg_sys_clk_init();
13     usart1_init();
14
15     //初始化flash存储
16     //同时把先前存储的数据读取出来放到 flash_helper_struct.FlashHelperData数组里面
17     flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储
18
19     flash_data_set_char_data(25); //存储数据
20     //获取数据
21     printf("char_data=%d\r\n", flash_data_get_char_data());
22
23
24     flash_data_set_u16_data(5666);
25     //获取数据
26     printf("u16_data=%d\r\n", flash_data_get_u16_data());
27
28     printf("start\r\n");
29     while(1)
30     {
31
32     }
33 }
34
```

```
ATK XCOM V2.0
char_data=25
u16_data=5666
start
```

3.如果存储字符串

假设这个字符串最大不到1024字节, 咱可以先建一个1024的数组,然后

```
flash_helper.c flash_data.c main.c flash_data.h flash_helper.h
1 #define flash_data_c_
2 /**
3 documents:
4 ****/
5 #include "flash_data.h"
6
7 //假设 flash_helper_struct.FlashHelperData[0] ;的高8位, 中16位 低8位各使用来存储数据
8
9 union_flash_data_value union_flash_data_valuel;
10
11
12 //假设 flash_helper_struct.FlashHelperData[1] 到 flash_helper_struct.FlashHelperData[256] 存储数据
13 char test_string[1024];
14 /*存储string*/
15 void flash_data_set_str_data(void){
16     memcpy(&flash_helper_struct.FlashHelperData[1], test_string, 1024);
17     //刷新
18     flash_helper_update();
19 }
20 /*获取string*/
21 void flash_data_get_str_data(void){
22     memcpy(test_string, &flash_helper_struct.FlashHelperData[1], 1024);
23 }
24
```

```

11 {
12     mcu_cfg_sys_clk_init();
13     usart1_init();
14
15     //初始化flash存储
16     //同时把先前存储的数据读取出来放到 flash_helper_struct.FlashHelperData数组里
17     flash_helper_init(32*FLASH_SECTOR_SIZE); //使用第32个扇区存储
18
19     flash_data_set_char_data(25); //存储数据
20     //获取数据
21     printf("char_data=%d\r\n", flash_data_get_char_data());
22
23
24     flash_data_set_u16_data(5666);
25     //获取数据
26     printf("u16_data=%d\r\n", flash_data_get_u16_data());
27
28     /*设置字符串数据*/
29     memset(test_string, 0, 1024);
30     sprintf(test_string, "%s", "string0000000000");
31     //把数据存储到flash
32     flash_data_set_str_data();
33
34     //获取数据
35     flash_data_get_str_data();
36     printf("string_data=%s\r\n", test_string);
37
38     printf("start\r\n");
39     while(1)
40     {
41     }
42 }

```

ATX XCOM V2.0

```

char_data=25
u16_data=5666
string_data=string0000000000
start

```

移植使用

把下面的文件放到自己的工程里面即可

HC32F460Template

- flash_helper.c
- flash_helper.h
- mcu_flash.c
- mcu_flash.h

结语

代码写多了呢其实写的是思路 and 思想,当然好的思路 and 思想需要建立在(不会表达了.....)

[好文要顶](#)[关注我](#)[收藏该文](#)

杨奉武

关注 - 1

粉丝 - 693

0

0

« 上一篇：[001-HC32F460\(华大\)+Air724UG\(4G GPRS\)基本控制篇\(阿里云物联网平台\)-C#,网页,android,微信小程序,单片机等使用MQTT接入阿里云物联网平台](#)

posted on 2021-12-19 19:13 杨奉武 阅读(0) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

编辑

预览

B



支持 Markdown

自动补全

提交评论

退出

[Ctrl+Enter快捷键提交]

- 【推荐】华为开发者专区，与开发者一起构建万物互联的智能世界
- 【推荐】云资源需求调查，百度智能云准备给园友们发专属优惠福利
- 【推荐】跨平台组态\工控\仿真\CAD 50万行C++源码全开放免费下载！
- 【推荐】华为 HMS Core 线上 Codelabs 挑战赛第4期，探索“智”感生活

编辑推荐：

- .NET内存性能分析指南
- 在腾讯这一年，坚守初心持续单纯 | 2021年终总结
- [WPF] 用 OpacityMask 模仿 UWP 的 Text Shimmer 动画
- 了解 C# 的Expression
- [.NET 与树莓派] 控制彩色灯带（WS28XX）

最新新闻：

- 阿里全力押注海外：阿里云是“矛” 蒋凡是“掌舵人”（2021-12-19 13:34）
- 贝壳强力回击浑水做空！中信证券力挺：廊坊数据不能代表全国（2021-12-19 13:25）
- 12306再辟谣，称消费者使用的加速包并不能拥有优先购票权（2021-12-19 13:17）
- 花200元找人做PPT、保存领导表扬截图，打工人为年终总结有多拼？（2021-12-19 13:05）

· 一线爱奇艺：被迫“裁员”，被逼“涨价”，六年烧掉400亿后春天在哪里？（2021-12-19 12:51）

» 更多新闻...

历史上的今天：

2019-12-19 ESA2GJK1DH1K升级篇: STM32远程乒乓升级,基于Wi-Fi模块AT指令TCP透传方...

Powered by:

博客园

Copyright © 2021 杨奉武

Powered by .NET 6 on Kubernetes



单片机,物联网,上位机,...

扫一扫二维码，入群聊。