EDS库

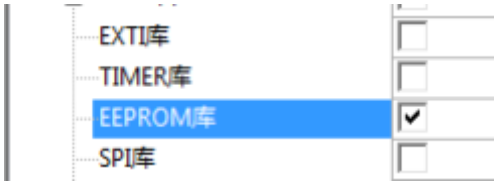
EDS全称Eeprom-Data-System，是基于eeprom库的数据管理系统。它的出现是为了解决一个问题，就是STC的eeprom是flash模拟的，那么会有：

1. 每次修改数据都得先擦除，耗时太长。
2. 如果同扇区还有别的不用修改的数据，还得建立缓存保护，修改完还得写回去，耗时更多了。
3. 如果同扇区只有这一个频繁修改的数据，虽然可以不用缓存省点时间，但是剩余空间的擦除寿命依然是被白白消耗的。

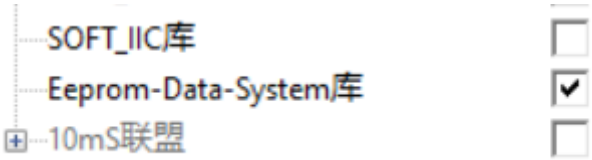
使用EDS库之后，可以有以下几点好处：

1. 不需要每次修改前擦除，省时。
2. 对同一扇区进行擦写均衡管理，大大延长了flash的寿命。
3. 有数据完整性检查，在遇到断电，跑飞重启等情况后，可对保存的数据进行回滚。

在使用本库之前先到ecbm_core.h里使能。双击打开ecbm_core.h文件，然后进入图形化配置界面，使能EEPROM库。



然后在下面的【单片机软件库的选用】里使能EDS库。



原理说明

为了让您更合理的使用EDS库，避免异常操作造成数据损失，请先阅读它的原理并结合原理来使用！

首先，EDS的管理范围以数据包的形式作为分区。每一个数据包可占用一个扇区或者多个扇区。EDS会在众多管理的扇区中，挑选第一个扇区用于存放记录标志和数据包信息，大约占用64字节。剩下的区域都是用来存放数据包的。举个例子：

- 当前有菜单设置项和上位机校准参数两个数据包需要经常修改。菜单设置项占用20字节，于是只分配扇区0一个扇区。上位机校准参数占用320字节，可以分配多一些即扇区1、扇区2和扇区3共三个扇区。此时扇区0存放着菜单设置项的记录标志，扇区1存放着上位机校准参数的记录标志。两个区域互不干扰，当扇区0存满之后，会擦除扇区0再写入，不会动到扇区1的数据。同理当扇区1、2、3都写满之后，将会清除扇区1、2、3，不会影响到扇区0。
- 数据包采用顺序写入，拿上面的例子来说，菜单设置项的数据包第一次存在地址64~地址83。加上2字节校验之后，下次写入会将数据包放到地址86~地址105。依此类推直至扇区0剩余的空间小于22字节时（数据包占用20字节、校验占用2字节）触发擦除动作，然后从地址64重新开始存。

然后，EDS最初的设计理念只有“**把flash当eeprom用，每次写入不需要先擦除**”这一条，擦写均衡是附带的效果。由于不是以“擦写均衡”为主要目的设计的，所以在某些极端的情况下，是达不到擦写均衡的效果的：

1. 管理扇区设置过小。由于第一扇区的前64字节用来存放标志信息，所以可用的空间只有448字节。如果数据包大小超过224，在存放第一次之后，剩下的空间不足以再放一次。那么再次存放的时

候，就一定会触发擦除。这样一来就没有擦写均衡的效果了。**解决办法就是将管理扇区设置大一些。**
2. 管理扇区没有覆盖完全部eeprom空间。除了个别型号可以调整eeprom的大小外，剩下的绝大部分型号是不可改大小的。比如说一个拥有32K大小eeprom的STC8A8K32S4A12，只使用EDS管理扇区0、1、2、3、4，那么后面的扇区就是空闲状态。等到前面的寿命结束后，后面的eeprom还没有被使用过，那自然就没有“均衡”的效果了。**解决办法也是把管理扇区设置大一些，尽量覆盖全部eeprom空间。**
3. EEPROM除了EDS之外还有别的擦写操作。对EDS所管理的区域进行操作，会增加EDS在校验方面的工作量。假如代码里不小心修改了信息区，那么每次运行EDS相关函数都会初始化擦除一次，那么EEPROM寿命会极速下降。**解决办法就是搜索整个工程，将里面能修改到EDS管理扇区的代码进行改写。**

在最近的一次升级中，为了解决掉电速度过快导致得写入数据不全问题，引入了校验算法。虽然为了节省flash空间，只做了最简单的校验和。而后在思考校验出错之后能干些什么，推出了“**数据回滚**”功能。该功能能在最新数据包的校验出错时，自动读取上一次正确的数据包返回。这样可以保证数据丢失得不多，但和上面一样，EDS的框架最初不是为了这个功能设计的，所以多多少少还是有些限制：

1. 数据回滚的前提当然还是有旧数据的存在。目前在数据包存满管理扇区之后，会有擦除的动作。说具体点，假如数据包大小为100字节，EDS管理扇区数量为1个扇区即512字节，去掉信息区和标志区之后可用的空间总数就是448字节。在存第五次的时候，由于剩下空间只有48字节不够存，那么就会触发擦除。该扇区的所有信息和数据都会被清理掉，然后重建信息区和初始化标志区，旧数据自然也就没了。这种情况是没有办法进行数据回滚的，也没有好的解决方法。

以上的种种，都是极端条件下的问题，如果为了解决小概率问题而加各种防护代码又会使得EDS库的体积过大（顺便一提，此次升级进行了大幅度的优化，在功能不变的情况下，测试工程的编译大小从7769字节优化到6804字节。毕竟库体积过大是广大用户经常吐槽的点。）。如果对数据管理有需求的人很多，那么以后可能会推出新的框架。

API

eds_init

函数原型：u8 eds_init(u8 sector_start,u8 sector_szie,u16 pack_len)

描述

eds初始化函数。

输入

- sector_start：开始扇区号。
- sector_szie：管理扇区个数。
- pack_len：每次写入的数据包长度。

输出

无

返回值

- EDS_ERR_SECTOR：说明参数的开始扇区号或者管理扇区个数没设置对。
- EDS_DATA_VOID：当前扇区在本函数运行之前没有EDS管理的痕迹，本次运行中将该区域清空了，所以目前内部没有数据。
- EDS_OK：一切正常。

调用例程

```
#include "ecbm_core.h"//加载库函数的头文件。
void main(){//main函数，必须的。
    system_init();//系统初始化函数。
    eeprom_init();//初始化eeprom。EDS是基于eeprom的上层操作，所以要先初始化eeprom。
    eds_init(0,1,20);//从扇区0开始，划分1个扇区用于管理一个大小为20字节的数据包。
    eds_init(1,3,320);//从扇区1开始，划分3个扇区用于管理一个大小为320字节的数据包。
    while(1){
    }
}
```

注意事项

1. 在设置开始扇区号和管理扇区个数之前，要搞清楚当前使用的单片机型号有多大的eeprom空间！提前合理规划好哪些扇区分配给哪些数据存放。
2. 函数返回EDS_DATA_VOID通常就是用来提醒你，当前扇区内没有数据，一般这个时候可以做一些数据的初始化什么的。在下面写函数的例子里会提到。

eds_write

函数原型：u8 eds_write(u8 sector_start,u8 * buf)

描述

eds写入函数。

输入

- sector_start：开始扇区号。
- buf：要写入该地址的数据缓存。

输出

无

返回值

- EDS_ERR_NULL：当前扇区的EDS标识错误。可能是还没初始化或者初始化之后被破坏了。需要重新调用eds_init初始化。
- EDS_ERR_ID：同上，也属于标识错误。这个错误可能是由于当前的数据是由其他扇区复制得到的，数据完整性有问题。
- EDS_DATA_EXISTS：当前的数据已存在。如果要写入的数据和之前存的数据一模一样，就会返回这个，代表本次数据没有写入。
- EDS_ERR_DATA：数据写入出错。flash的特点就是只能写0不能写1，假如写入区域内不全是0xff，就会出这个错误。建议重写一次避开这个区域。
- EDS_ERR_SUM：校验出错。和上面数据写入出错差不多，只不过这里出错的是校验部分。建议重写一遍。
- EDS_OK：一切正常。

调用例程

```
#include "ecbm_core.h"//加载库函数的头文件。
typedef struct{//示意代码，语法上肯定是错的，不要直接复制。
    u8 id;
    u8 option;
```

```
    ...
}menu;//定义菜单设置项。
u8 uart_buf[320];//用于存放校准数据。
void main(){//main函数，必须的。
    u8 info;//用于存放函数返回结果。
    system_init();//系统初始化函数，因为开了自动下载，所以这个函数里是包含uart_init的。
    eeprom_init();//初始化eeprom。EDS是基于eeprom的上层操作，所以要先初始化eeprom。
    info=eds_init(0,1,20);//从扇区0开始，划分1个扇区用于管理一个大小为20字节的数据包。
    if(info==EDS_DATA_VOID){//如果是空的，通常烧录完第一次上电就是空的。
        menu.id=1;
        menu.option='A';
        ...//示意代码。就是在初始化结构体menu。
        eds_write(0,(u8 *)&menu);//将初始值写入。
    }
    info=eds_init(1,3,320);//从扇区1开始，划分3个扇区用于管理一个大小为320字节的数据包。
    if(info==EDS_DATA_VOID){//如果是空的，通常烧录完第一次上电就是空的。
        uart_printf(1,"VOID");//通知上位机发送校准数据。
    }
    while(1){
        if(...){//遇到某个条件后，比如按下了菜单的保存键。
            info=eds_write(0,(u8 *)&menu);//将菜单设置项存入扇区0。
            if((info!=EDS_OK)&&(info!=EDS_DATA_EXISTS)){//如果不是写入正常，或者数据
已存在，
                eds_write(0,(u8 *)&menu);//就再写一遍。
            }
        }
        if(...){//遇到某个条件后，比如上位机发送了校准数据。
            info=eds_write(1,uart_buf);//将上位机发送的数据存入扇区1.2.3。
            if((info!=EDS_OK)&&(info!=EDS_DATA_EXISTS)){//如果不是写入正常，或者数据
已存在，
                eds_write(1,uart_buf);//就再写一遍。
            }
        }
    }
}
```

注意事项

1. 如果数据比较重要，还是推荐判断函数返回值，返回值不正常的话该重写就要重写不要怕flash寿命问题。但是也要注意不要在死循环里执行写入操作，寿命再长的flash也禁不起这样消耗。

eds_read

函数原型：u8 eds_read(u8 sector_start,u8 * buf)

描述

eds读取函数。

输入

- sector_start：扇区开始号。
- buf：缓存，用于存放从该地址读出的数据。

输出

无

返回值

- EDS_ERR_NULL：当前扇区的EDS标识错误。可能是还没初始化或者初始化之后被破坏了。需要重新调用eds_init初始化。
- EDS_ERR_ID：同上，也属于标识错误。这个错误可能是由于当前的数据是由其他扇区复制得到的，数据完整性有问题。
- EDS_DATA_VOID：数据为空。出现这个代表着在储存区没有找到数据，有可能数据被其他代码清理或者刚初始化EDS还没来得及写入数据。
- EDS_WAR_DATA：数据警告。出现这个警告说明最新的数据有问题，但是上一次的数据没问题，因此数据回滚了。
- EDS_ERR_SUM：校验错误。当读到的数据和校验对比不成功的时候返回这个。可能的原因有写入时断电，存好的数据被其他代码改写等。
- EDS_OK：一切正常。

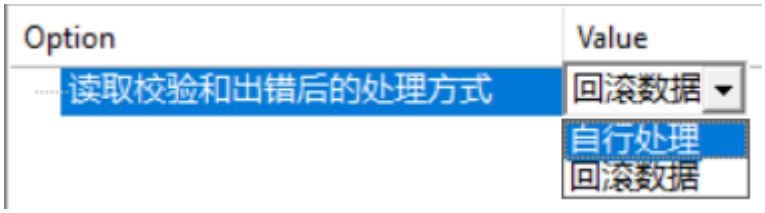
调用例程

```
#include "ecbm_core.h"//加载库函数的头文件。
typedef struct{//示意代码，语法上肯定是错的，不要直接复制。
    u8 id;
    u8 option;
    ...
}menu;//定义菜单设置项。
u8 uart_buf[320];//用于存放校准数据。
void main(){//main函数，必须的。
    u8 info;//用于存放函数返回结果。
    system_init();//系统初始化函数，因为开了自动下载，所以这个函数里是包含uart_init的。
    eeprom_init();//初始化eeprom。EDS是基于eeprom的上层操作，所以要先初始化eeprom。
    info=eds_init(0,1,20);//从扇区0开始，划分1个扇区用于管理一个大小为20字节的数据包。
    if(info!=EDS_OK){//只要不OK，都要初始化。
        menu.id=1;
        menu.option='A';
        ...//示意代码。就是在初始化结构体menu。
        eds_write(0,(u8 *)&menu);//将初始值写入。
    }else{//如果数据OK，就读出来使用。
        eds_read(0,(u8 *)&menu);
    }
    info=eds_init(1,3,320);//从扇区1开始，划分3个扇区用于管理一个大小为320字节的数据包。
    if(info!=EDS_OK){//只要不OK，
        uart_printf(1,"VOID");//都要通知上位机发送校准数据。
    }else{//如果数据OK，就读出来使用。
        eds_read(1,uart_buf);
    }
    while(1){
        if(...){//遇到某个条件后，比如按下了菜单的保存键。
            info=eds_write(0,(u8 *)&menu);//将菜单设置项存入扇区0。
            if((info!=EDS_OK)&&(info!=EDS_DATA_EXISTS)){//如果不是写入正常，或者数据
已存在，
                eds_write(0,(u8 *)&menu);//就再写一遍。
            }
        }
        if(...){//遇到某个条件后，比如上位机发送了校准数据。
            info=eds_write(1,uart_buf);//将上位机发送的数据存入扇区1.2.3。
```

```
            if((info!=EDS_OK)&&(info!=EDS_DATA_EXISTS)){//如果不是写入正常，或者数据已存在，
                eds_write(1,uart_buf);//就再写一遍。
            }
        }
    }
}
```

注意事项

1. 在eds.h有选项，可以在数据出错的时候，选择自己处理或是数据回滚。



2. 回滚的最大次数是5次，如果超过了5次还没有正确的数据，那么说明该扇区被破坏得很严重了。建议核查代码看看是什么操作在搞破坏。

优化建议

无，当前功能已经是最精简的（只有3个函数供调用）了。如果还是用不到这样的功能，可以自行在eeprom库上建立自定义操作。