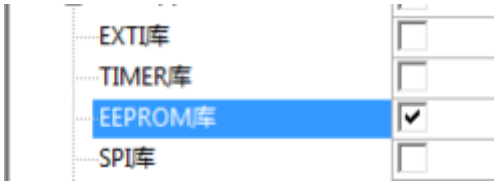
EEPROM库

STC的eeprom用法和平时常见的以AT24C02为代表的eeprom不一样。从名字上来说它们都可以叫“电可擦编程只读存储器(Electrically Erasable Programmable Read-Only Memory)”。但AT24C02能以字节为单位进行读写，而STC的eeprom是用flash模拟的，所以写入可以写单个字节、而擦除只能擦除整个扇区。

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。



API

eeprom_init

函数原型：void eeprom_init(void);

描述

eeprom初始化函数。

输入

无

输出

无

返回值

无

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。
    while(1){
    }
}
```

注意事项

无

eeprom_erase

函数原型：void eeprom_erase(u16 addr);

描述

eeprom擦除函数。

输入

- addr：要擦除的地址。

输出

无

返回值

无

小科普

STC的eeprom是用flash的模拟的，flash的特点有：

1. 读可以一个字节一个字节读。
2. 写可以一个字节一个字节写，但是写操作只能把数据里的1写成0（“1写成1”和“0写成0”这两种因为数据不变所以算不上写入）。

   比如：地址0原来的数据是0x09（0000 1001），写入0x01（0000 0001）是可以的，因为写入0x01会把D7位从0写成0、D6位从0写成0、D5位从0写成0、D4位从0写成0、D3位从1写成0、D2位从0写成0、D1位从0写成0、D0位从1写成1。本次操作中，除了D3位是1写0外，其他位数据不变。

   如果地址0原来的数据是0x09（0000 1001），写入0x03（0000 0011）是不可以的，因为写入0x03会把D7位从0写成0、D6位从0写成0、D5位从0写成0、D4位从0写成0、D3位从1写成0、D2位从0写成0、D1位从0写成1、D0位从1写成1。本次操作中，只有两位改变了，分别是D3位的1写0和D1的0写1。其中D3位能写成功，D1位的0写1不成功，所以D1还是0。于是源数据0x09（00001001）只有D3位1写0，最终结果就是0x01（0000 0001）。

   这就是为什么STC的eeprom每次写入之前都要擦除的原因。
3. 擦除是按扇区擦除，擦除是flash能把数据里0写成1的唯一操作。擦除操作把一个扇区共512字节的数据全变成0xFF，之后就可以任意写入任意值了。

调用例程

```
eeprom_erase(20);//擦除地址20所在的扇区。
```

注意事项

1. STC8的扇区是512字节。所以本函数只要一执行，参数地址所在的扇区都会被擦除。比如例程中擦除地址为20，地址20所在的扇区0的范围是0~511。所以0~511地址的数据都会被擦除掉。
2. 本库中的eeprom操作是通过IAP寄存器的，STC的硬件在运行时会在IAP寄存器里自动加入flash空间偏移，因此无论是哪个型号的STC8单片机，eeprom的地址都是从0开始的！STC手册上给的地址偏移仅针对MOVC法，IAP法永远都是从地址0开始！

eeprom_write

函数原型：void eeprom_write(u16 addr,u8 dat);

描述

eeprom写入函数。

输入

- addr：要写入的地址。
- dat：要写入该地址的数据。

输出

无

返回值

无

调用例程

上电后的第一次写入：

```
//单片机flash在下载的时候都擦除一遍了，因此默认就是0xFF，所以第一次不用擦除。
eeprom_write(20,125);//向地址20写入数据125。
```

上电后的第N次写入：

```
eeprom_erase(20);//第N次写入的时候，由于其地址内容不一定是0xFF，所以要先擦除地址20所在
的扇区。
eeprom_write(20,15);//向地址20写入数据15。
```

注意事项

1. 单片机电压低的时候，会导致写入异常。尽量保证单片机在3V以上电压工作吧。
2. 本库中的eeprom操作是通过IAP寄存器的，STC的硬件在运行时会在IAP寄存器里自动加入flash空间偏移，因此无论是哪个型号的STC8单片机，eeprom的地址都是从0开始的！STC手册上给的地址偏移仅针对MOVC法，IAP法永远都是从地址0开始！

eeprom_read

函数原型：u8 eeprom_read(u16 addr);

描述

eeprom读取函数。

输入

- addr：要读取的地址。

输出

无

返回值

- 该地址的数据。

调用例程

```
val=eeprom_read(20);//读取地址20的数据赋予变量val。
```

注意事项

1. 有人反馈说eeprom写入数据立刻读取会读取到异常数据，可以适当在写入函数和读取函数之间加入几毫秒延时。
2. 本库中的eeprom操作是通过IAP寄存器的，STC的硬件在运行时会在IAP寄存器里自动加入flash空间偏移，因此无论是哪个型号的STC8单片机，eeprom的地址都是从0开始的！STC手册上给的地址偏移仅针对MOVC法，IAP法永远都是从地址0开始！

eeprom_read_ex

函数原型：void eeprom_read_ex(u16 addr,u8 * dat,u16 num);

描述

eeprom批量读取函数。

输入

- addr：要读取的地址。
- num：要读取的数量。

输出

- dat：读取到的数据。

返回值

无

调用例程

单个字节读取：

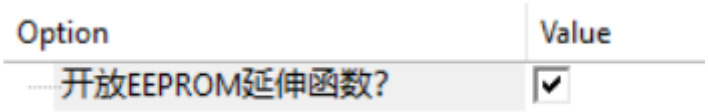
```
u8 val;
eeprom_read_ex(0,&val,1);//从地址0读取一个数据到变量val。
```

多个字节读取：

```
u8 val[10];
eeprom_read_ex(0,val,10);//从地址0开始连续读取10个数据到数组val。
```

注意事项

1. 本函数需要在图形化配置界面使能【开放EEPROM延伸函数】才能使用。



2. 有人反馈说eeprom写入数据立刻读取会读取到异常数据，可以适当在写入函数和读取函数之间加入几毫秒延时。
3. 本库中的eeprom操作是通过IAP寄存器的，STC的硬件在运行时会在IAP寄存器里自动加入flash空间偏移，因此无论是哪个型号的STC8单片机，eeprom的地址都是从0开始的！STC手册上给的地址偏移仅针对MOVC法，IAP法永远都是从地址0开始！

eeprom_write_ex

函数原型：void eeprom_write_ex(u16 addr,u8 * dat,u16 num);

描述

eeprom批量写入函数。

输入

- addr：要写入的地址。
- dat：要写入的数据。
- num：要写入的数量。

输出

无

返回值

无

调用例程

单个字节写入：

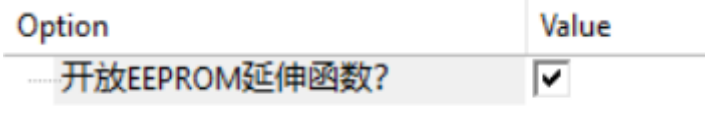
```
u8 val;
val=100;
eeprom_write_ex(0,&val,1);//把变量val的值写到地址0。
```

多个字节写入：

```
u8 val[10]={10,32,43,4,65,96,17,38,49,20};//10个数据
eeprom_write_ex(0,val,10);//将以上10个数据写入地址0~9。
```

注意事项

1. 本函数需要在图形化配置界面使能【开放EEPROM延伸函数】才能使用。



2. 由于擦除操作一次会擦除512字节的数据，所以本函数会在xdata区开设512字节的缓存用于缓存没有被修改的数据。如果单片机xdata空间紧张，可以不用这个函数。
3. 本库中的eeprom操作是通过IAP寄存器的，STC的硬件在运行时会在IAP寄存器里自动加入flash空间偏移，因此无论是哪个型号的STC8单片机，eeprom的地址都是从0开始的！STC手册上给的地址偏移仅针对MOVC法，IAP法永远都是从地址0开始！

优化建议

如果存入的数据很少，或者是不需要频繁改变。可以把扩展函数的使能取消掉，不仅可以节省FLASH空间还能节省512字节的XDATA空间。