IIC库

IIC是集成电路总线接口（Inter-Integrated Circuit）的缩写，IIC库就是关于单片机的IIC的操作库。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IIC库。

API

iic_master_init

函数原型：void iic_master_init(void);

描述

IIC主机初始化函数。

输入

无

输出

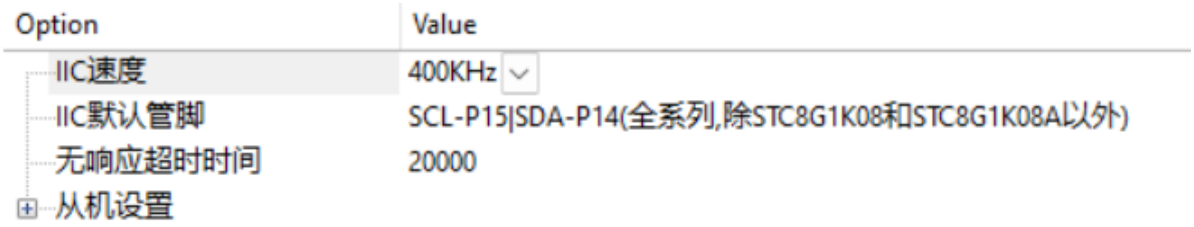
无

返回值

无

参数配置

本函数初始化的参数都是由图形化配置界面来设置，双击打开iic.h文件，进入图形化配置界面。



设置说明如下：

- IIC速度： 有100KHz和400KHz可选。虽然可以设置成其他速度，但是这两个是比较常用的。
- IIC默认管脚：按照选项内容和实际需求选择即可。注意选项括号里的提示，有些型号的引脚可能会不同。
- 无响应超时时间：没有单位，数值越大，时间就越久。不过可以确定就是uS级别。

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。
    iic_master_init();//初始化iic为主机模式。
    while(1){
    }
}
```

注意事项

1. 调用本函数会将IIC设置为主机模式，如果想作为从机使用，就不能用这个函数。

iic_set_pin

函数原型：void iic_set_pin(u8 group);

描述

IIC的引脚设置函数。

输入

- group：引脚所在的分组。

输出

无

返回值

无

分组定义

基本宏定义的名字就说明了IIC将会用到哪些IO了：

- IIC_PIN_P32_P33
- IIC_PIN_P54_P55
- IIC_PIN_P15_P14
- IIC_PIN_P25_P24
- IIC_PIN_P77_P76

在调用之前，请确认当前的型号确实有这些脚。前期确认一遍，不会耽误太多时间。

调用例程

```
if(run_mode==1){//当运行模式为1的时候，
    iic_set_pin(IIC_PIN_P32_P33);//控制P3连接的IIC器件。
}else{//在其他模式下，
    iic_set_pin(IIC_PIN_P54_P55);//控制P5连接的IIC器件。
}
```

注意事项

1. 在执行本函数的时候会自动设置相应的管脚为开漏输出，并使能内部上拉电阻。
2. iic_master_init函数已经在内部调用了本函数，如果不切换管脚的话，没必要使用本函数。

iic_reset_pin

函数原型：void iic_reset_pin(u8 group);

描述

IIC的还原引脚设置函数。

输入

- group：引脚所在的分组。

输出

无

返回值

无

分组定义

基本宏定义的名字就说明了IIC将会用到哪些IO了：

- IIC_PIN_P32_P33
- IIC_PIN_P54_P55
- IIC_PIN_P15_P14
- IIC_PIN_P25_P24
- IIC_PIN_P77_P76

在调用之前，请确认当前的型号确实有这些脚。前期确认一遍，不会耽误太多时间。

调用例程

```
if(run_mode==1){//当运行模式为1的时候，
    iic_set_pin(IIC_PIN_P32_P33);//控制P3连接的IIC器件。
    iic_reset_pin(IIC_PIN_P54_P55);//还原P54和P55。
}else{//在其他模式下，
    iic_set_pin(IIC_PIN_P54_P55);//控制P5连接的IIC器件。
    iic_reset_pin(IIC_PIN_P32_P33);//还原P32和P33。
}
```

注意事项

1. 在执行本函数的时候会自动设置相应的管脚为弱上拉，并关闭内部上拉电阻。
2. 一般在引脚复用的情况下，才需要调用这个函数来还原。否则可以一直保持开漏模式。

操作函数

函数原型：

- void iic_start(void);
- void iic_stop(void);
- void iic_write(u8 dat);
- void iic_write_ack(void);
- void iic_write_nack(void);
- u8 iic_read(void);
- bit iic_read_ack(void);

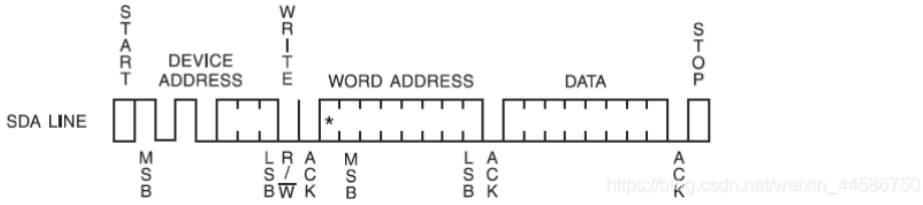
描述

IIC的操作相关函数，需要根据实际时序去调用。

调用例程

比如下图是AT24C02芯片的写一个字节的时序：

Figure 8. Byte Write



可以看出来时序动作是：【开始】【写器件地址和写入位】【等ACK】【写片内地址】【等ACK】【写数据】【等ACK】【结束】。所以可以得到如下代码。

```
void at24c02_write_byte(u8 addr,u8 dat){
    iic_start(); //开始。
    iic_write(0xA0);//写器件地址和写入位。
    iic_read_ack(); //等从机ACK。
    iic_write(addr);//写片内地址。
    iic_read_ack(); //等从机ACK。
    iic_write(dat); //写数据。
    iic_read_ack(); //等从机ACK。
    iic_stop(); //结束。
}
```

注意事项

1. 根据时序写就行，但目前STC的硬件IIC似乎有点问题，如果驱动不成功，可以换软件IIC试试。

iic_slave_init

函数原型：void iic_slave_init(void);

描述

IIC从机初始化函数。

输入

无

输出

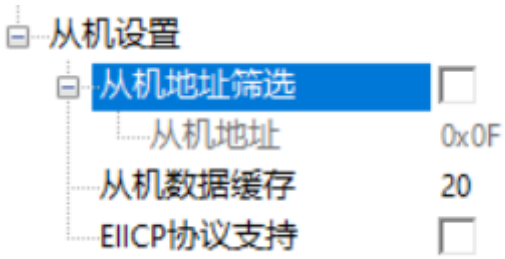
无

返回值

无

参数配置

本函数初始化的参数都是由图形化配置界面来设置，双击打开iic.h文件，进入图形化配置界面，找到从机设置。



设置说明如下：

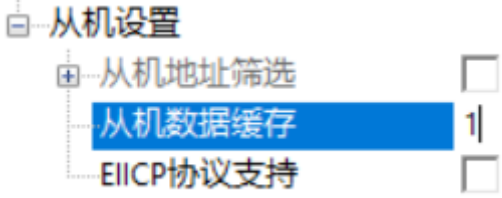
- 从机地址筛选：使能本选项之后，如果IIC总线上的地址不是本机的地址的话，将不会触发IIC中断。
- 从机地址：当使能了筛选功能之后，就得设置一个本机地址。
- 从机数据缓存：虽然这个缓存主要是为了EIICP协议建立的，但是在不使用EIICP协议的情况下，也能记录IIC从机模式下接收的数据。可以通过读取数组eiicp_data_buf_gu8a来读取该缓存。
- EIICP协议支持：由串口的ECP修改而来，是一种一问一答的协议。和市面上的IIC协议没有任何共同点，更加类似于modbus那种，只不过不是基于485总线，而是IIC总线罢了。因此只推荐用于多片STC8互相通信的情况。

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。
    iic_slave_init();//初始化iic为从机模式。
    while(1){
        //虽然是为了EIICP开发的，但普通的接收工作依然可以借用这些标志位和缓存。
        if(eiicp_trig_gb){//当这个标志位置位时，说明收到了一个完整的数据帧。
            eiicp_trig_gb=0;//清零标志位。
            if(eiicp_data_buf_gu8a[0]==0xEC){//读取缓存数组eiicp_data_buf_gu8a就能知道收到什么数据了。
                ...//剩下的自由发挥。
            }
            ...//剩下的自由发挥。
        }
    }
}
```

注意事项

1. 调用本函数会将IIC设置为从机模式，如果想作为主机使用，就不能用这个函数。
2. 缓存eiicp_data_buf_gu8a是使能IIC库之后就会定义的，所以如果只用到了主机，可以在这里将缓存设置为1，尽量减少占用。



优化建议

本库比较简单，所以没有可优化的地方。