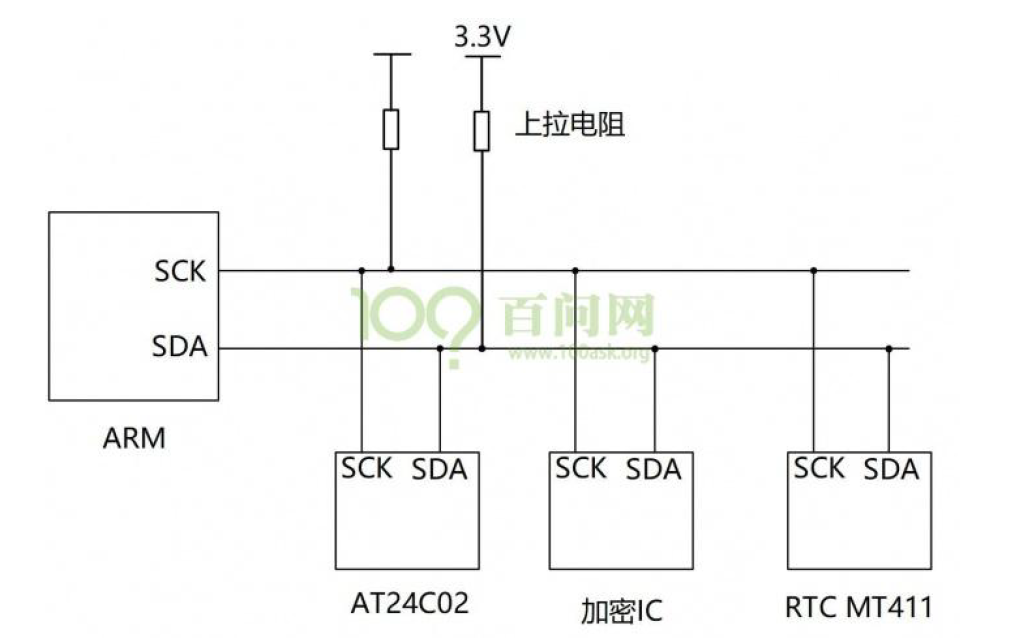
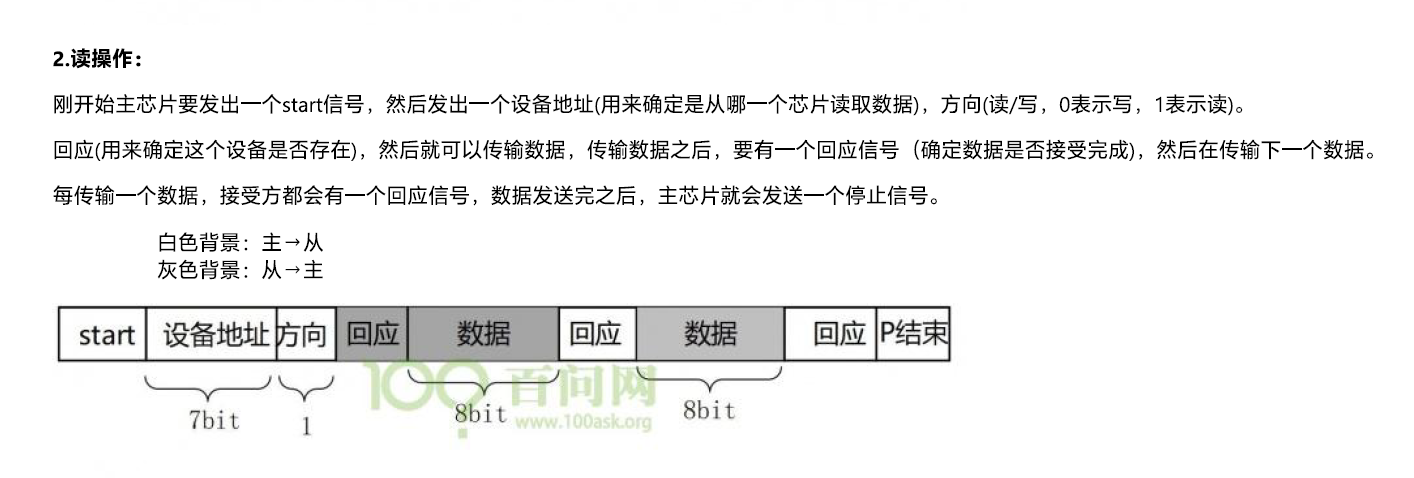
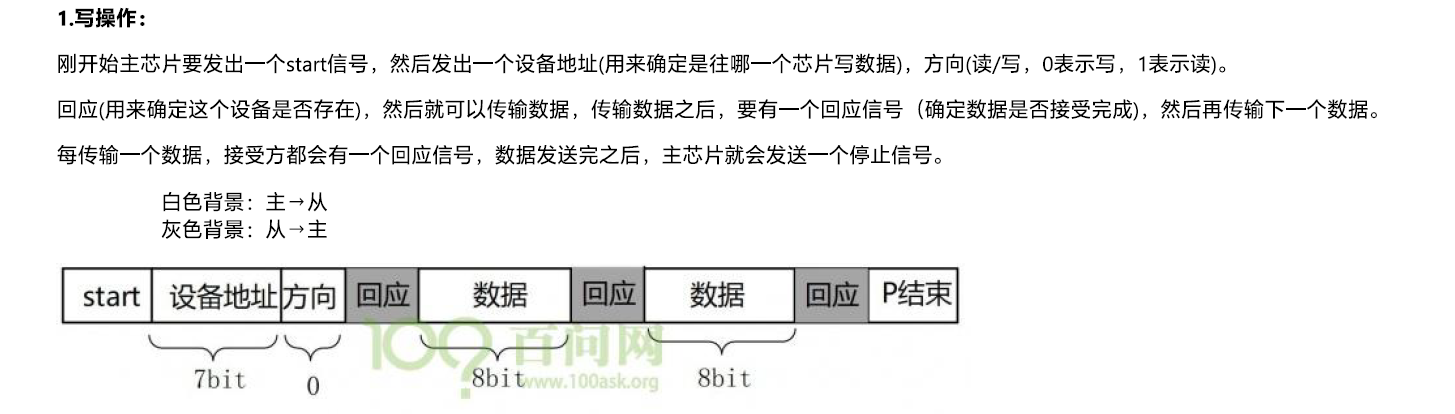
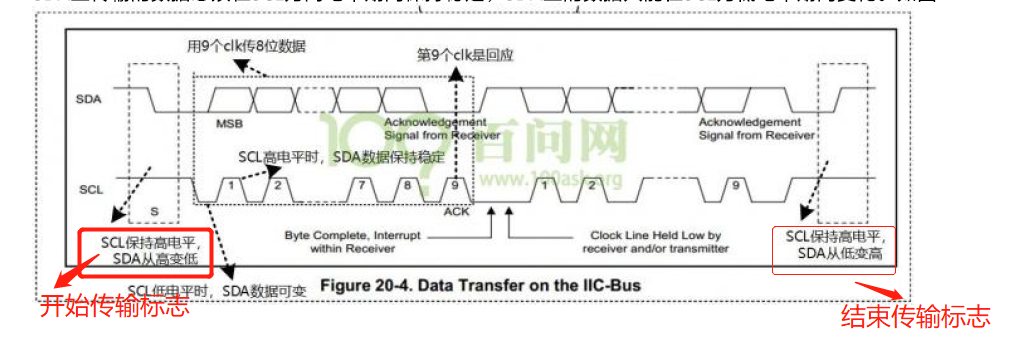
## i2c协议与eeprom







SDA上传输的数据必须在SCL为高电平器件保持稳定，SDA上的数据只能在SCL为低电平期间变化

主芯片通过一根SDA先既可以把数据发给从设备，也可以从SDA上读取数据，连接SDA线的必有引脚（发送引脚，接收引脚）





当一个芯片不想影响SDA线时，那就不驱动这个三极管

想输出高电平是，都不驱动（高电平有上拉电阻决定）

想输出低电平，驱动

比如：住着被发送给从设备

前8clk：

从设备不影响发送，就不驱动三极管

主设备决定数据

第9clk：

主设备不驱动三极管

从设备决定数据

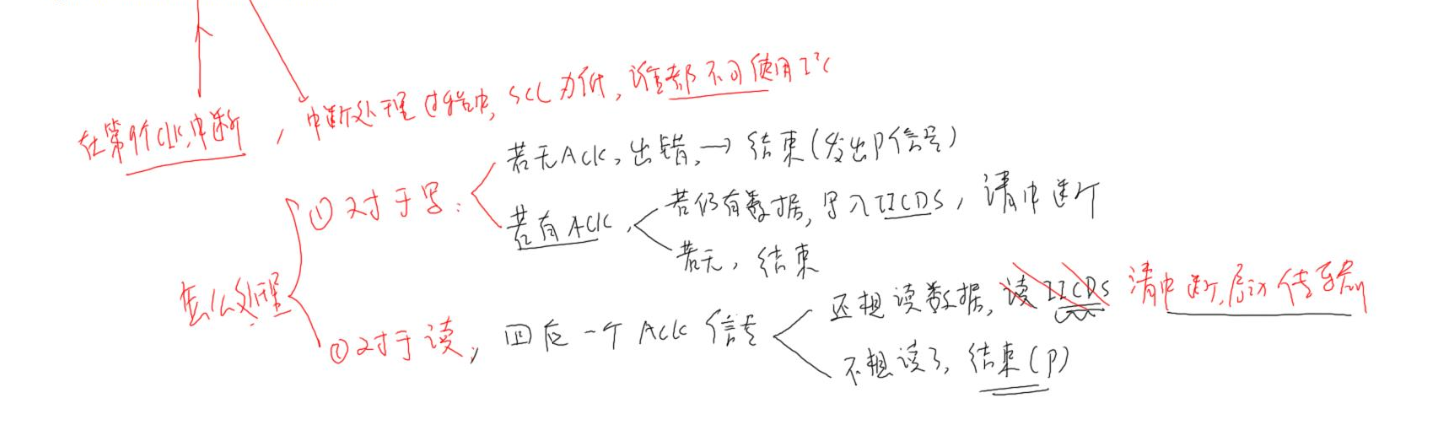
第九个时钟之后，如果有一方处于繁忙状态，它可以把SCL拉低,SCL为低电平时候，两方都不使用iic总线，只有当SCL由低变高才能使用

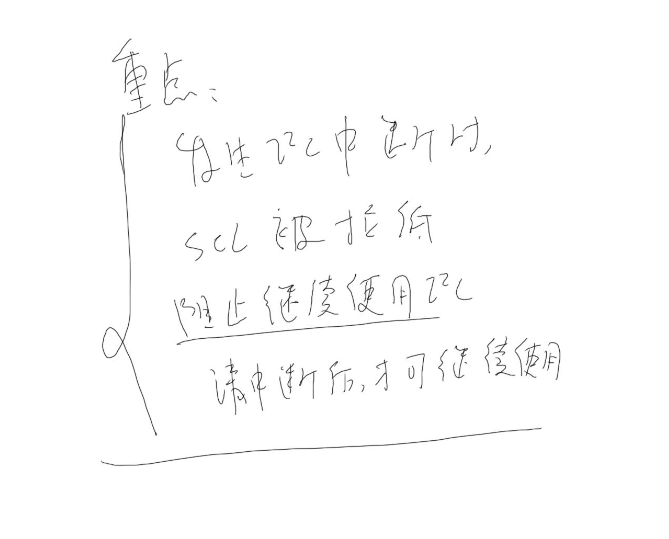
## s3c2440的iic控制器





根据上图，首先设置IICCON来设置时钟分频，原PCLK太快了，把时钟降低，接着设置寄存器发出start信号，之后发出数据，程序可以把数据写入到IICDS寄存器，一写入就会自动发出时钟，并且把这8位数据从SDA发送给设备，数据发送之后，在第九个时钟会受到回应信号，可以查询IICSTAT是否有ACK，有就表示数据发送成功了。可以继续发送，等发送完再设置IICSTAT让它发出P信号，在第九个CLK，就会产生一个中断，在中断处理过程中SCL被拉低电平，谁都不能再使用iic总线，等待中断处理完成。





处理中断

写操作：

若无ACK，出错然后发出P信号，

若有ACK，表示上一个字节发送成功

若仍有数据，写入IICDS寄存器，然后清中断，一清中断就会释放SCL信号，继续发出时钟，把数据再次发送出去

若没有数据了，发出P信号结束

读操作：

读到8位数时，回应一个ACK信号

还想读数据，清中断，启动传输，等它再次发生中断时，再来读取IICDS寄存器，得到数据，不想读取数据，发出P信号结束

发生中断时，iic控制器会把scl拉低，阻止任何设备再使用iic总选，清中断之后才能继续使用，这种记住就给服务中断程序执行提供了时间

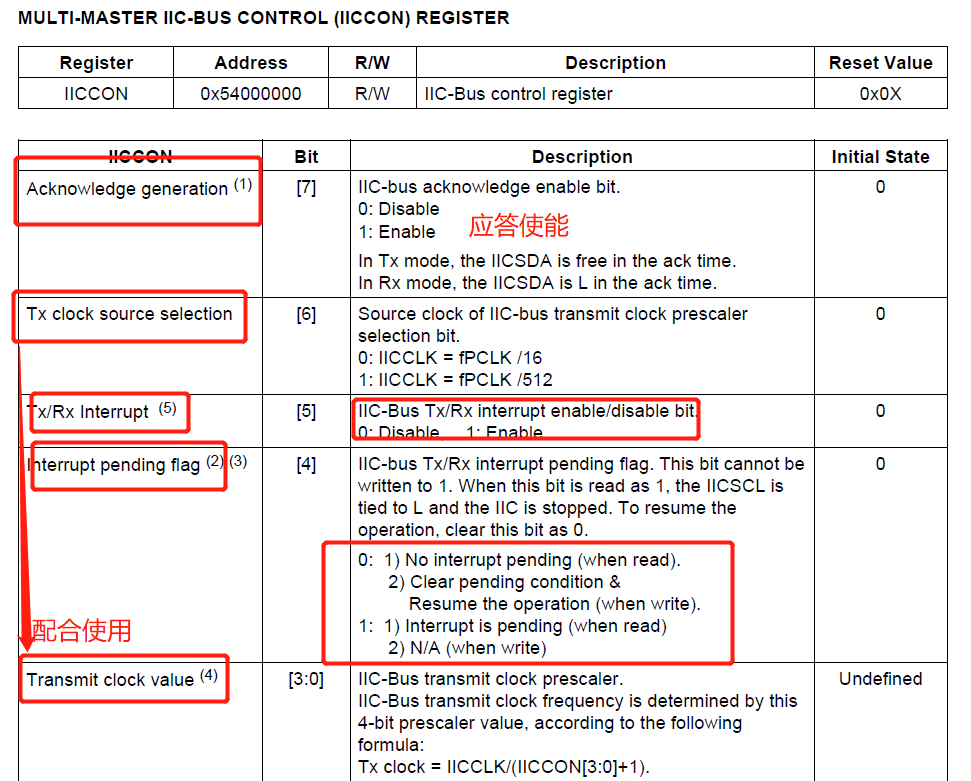
读写操作：

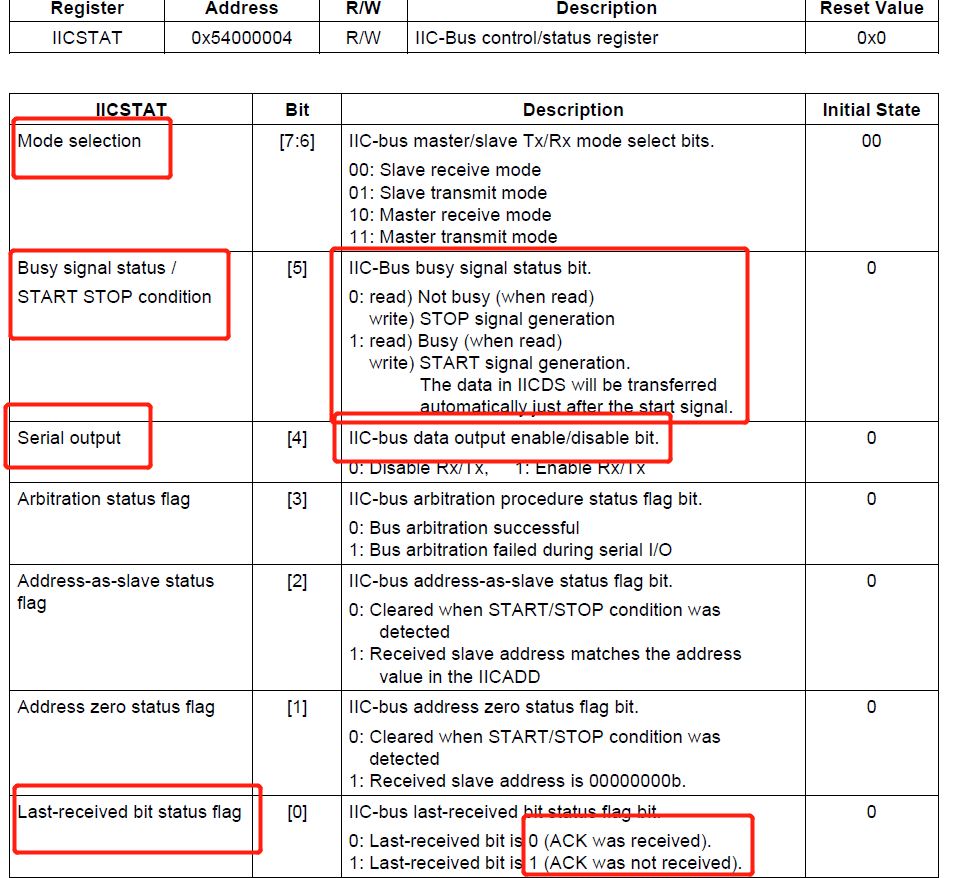
发送模式：

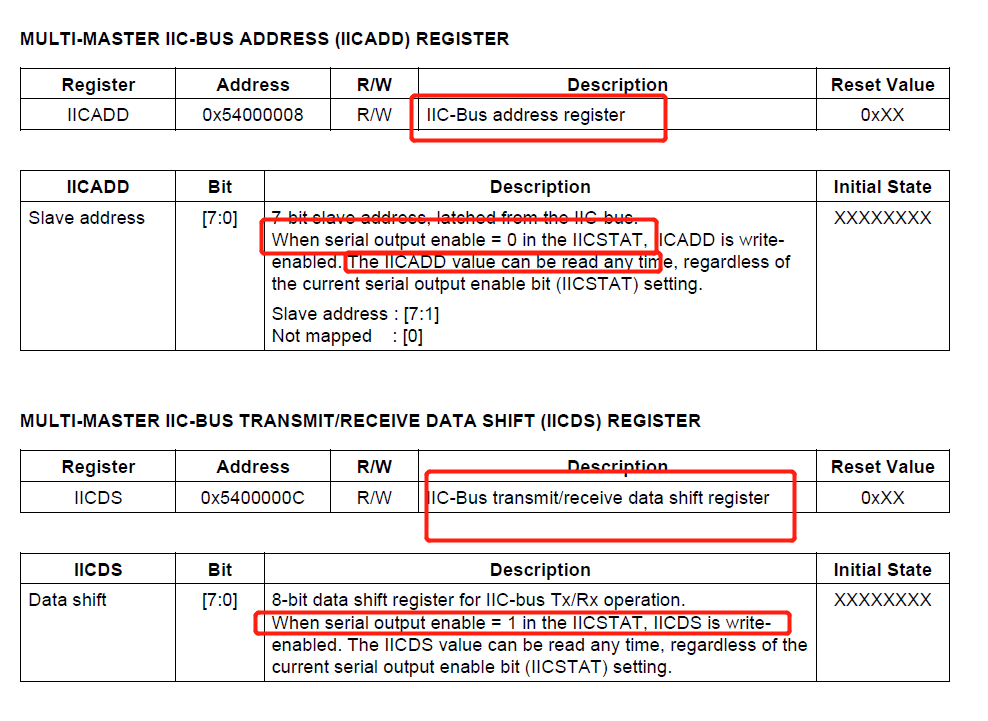
1. 往寄存器IICDS寄存器放入val值
2. 发完，产生中断，并且会把SCL拉低
3. 在中断程序里，判断状态，然后往IICDS里写入下一个数据，一旦写入下一个数据iic继续操作，若再次发完，就会再次产生中断

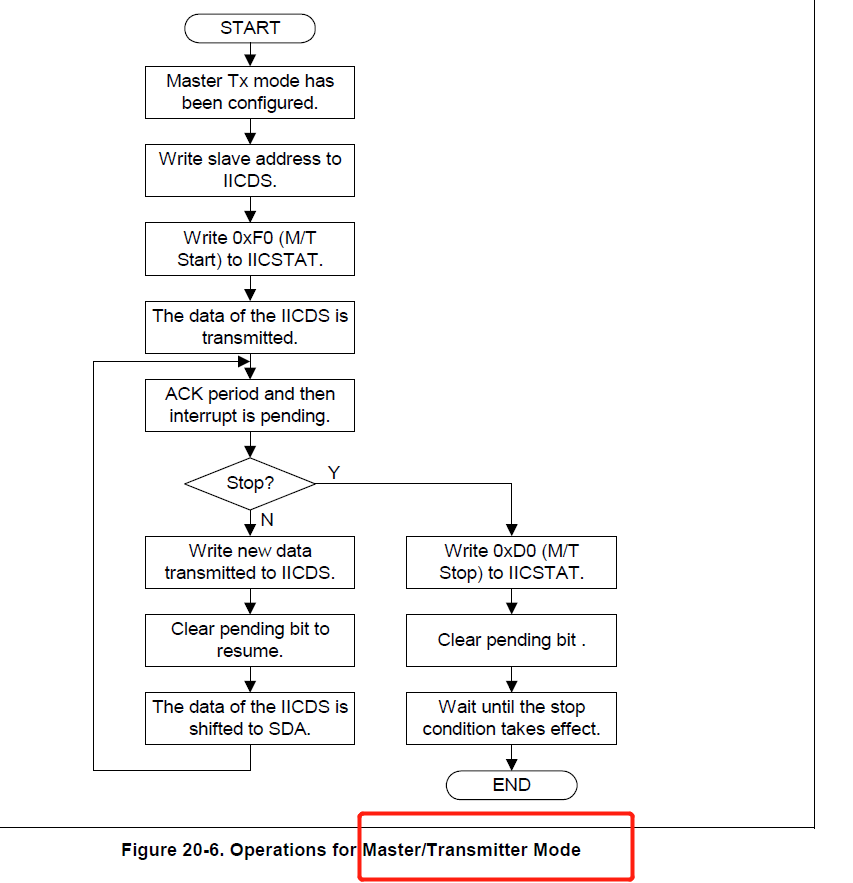
接收模式：

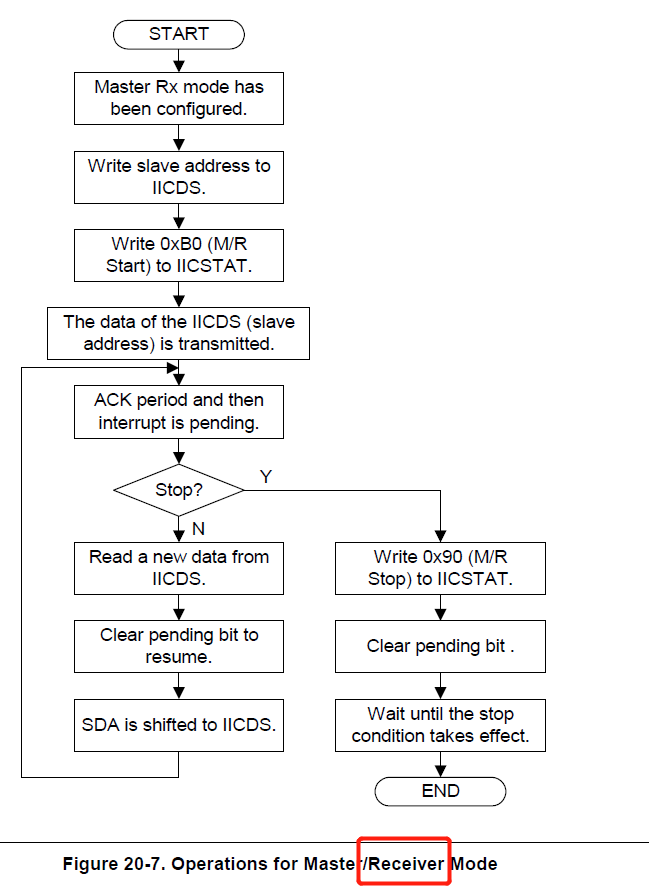
1. 程序发起传输，接收数据
2. 接收到数据之后，产生中断，SCL被拉低
3. 中断程序里，判断数据是否要继续就收，若要，再次设置，读取IICDS寄存器，一但读出来IIC继续接收下一个数据，收到新数据之后，又会产生一个中断



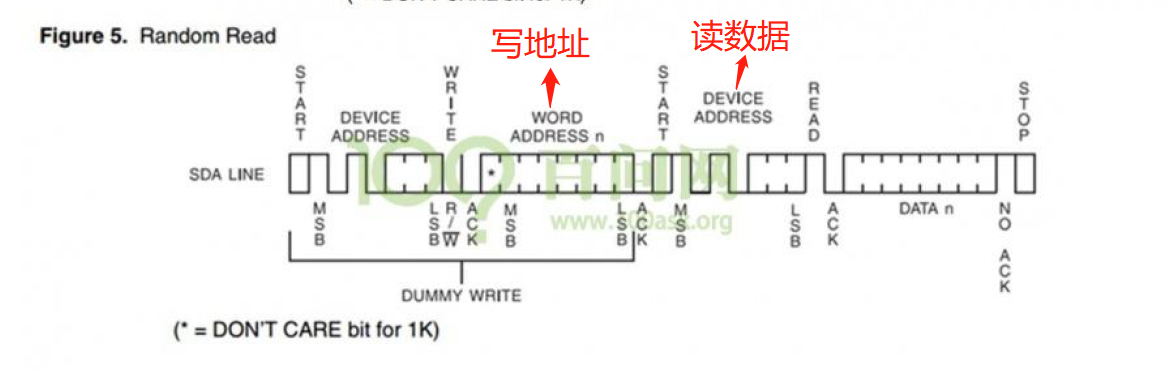


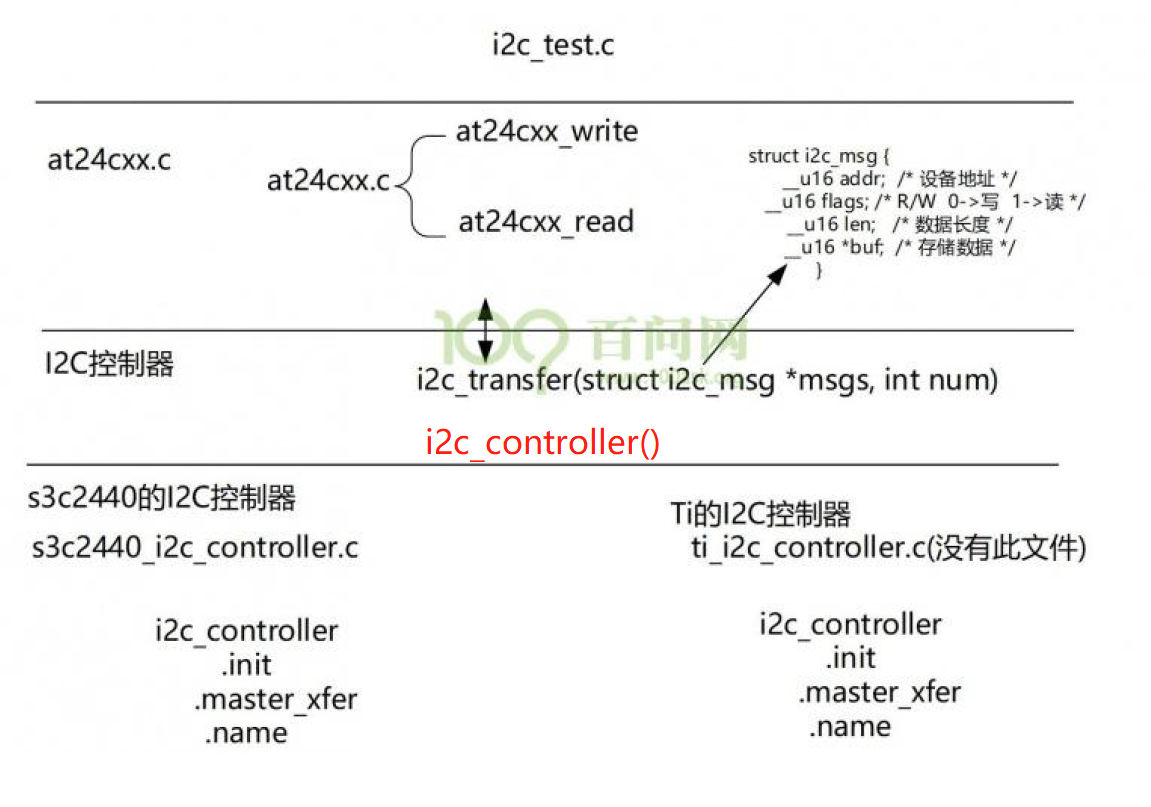






## 程序框架





提供一个统一的接口i2c\_transfer，不管使用哪个芯片，他最终会调用i2c\_transfer，来选择某款I2c控制器，把数据发送出去，或者从i2c设备读到数据，对于没一次传输的数据都可以用一个i2c\_msg结构体来表示，但读取某个地址的数据时，要用两个i2c\_msg来描述，因为一个i2c\_msg结构体只能描述一个传输方向（读或写），读取ac24cxx某个地址上的数据时，要先写出要读取的地址，然后来读取设备地址上的数据。

设计出一个结构体比较容易拓展的框架，对于i2c控制器抽象出一个结构日i2c\_controller，然后把这个结构体告诉上层，上层有个管理者i2c\_controller文件

在s3c2440\_i2c\_controller这个文件中构造出一个i2c\_controller结构体，放入上层文件的数组中，以后就根据结构体的名字，把这个结构体取出来用

假设有一个ti开发板，那么杂it\_i2c\_controller文件中，同样要构造出一个i2c\_controller，同样放入上层，根据名字使用

对于设备层中的at24cxx芯片，在这个at24cxx中实现读写函数：

at24cxx\_write at24cxx\_read

读写函数都会调用i2c\_transfer发起传输，

写程序主要涉及三个文件

at24cxx.c s3c2440\_i2c\_controller i2c\_controller

最上层写i2c\_test

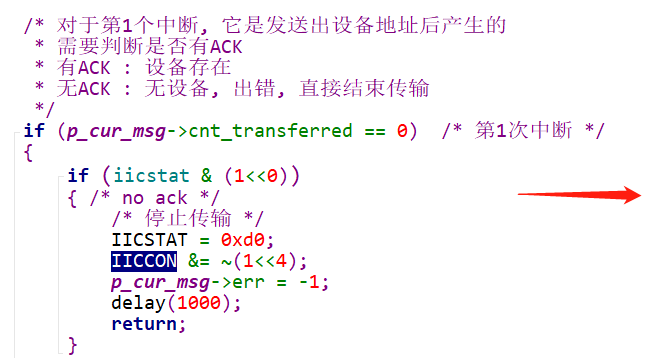
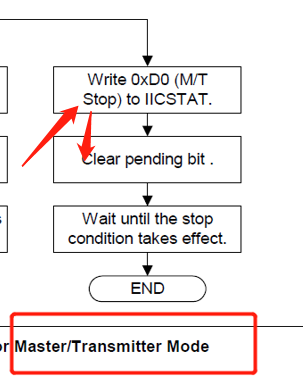
框架写完见001

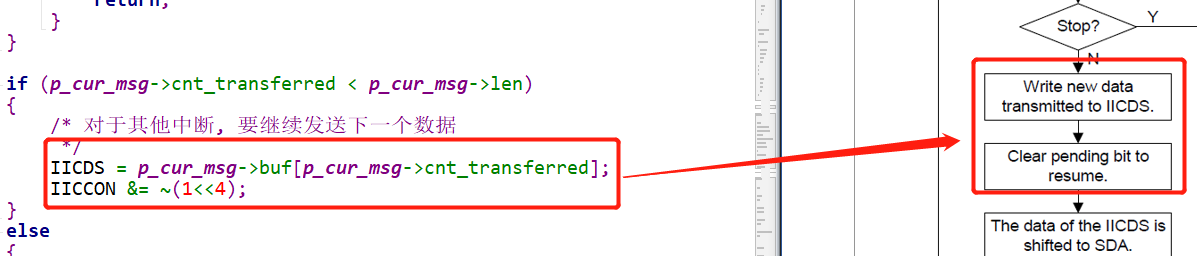
接下来是完善中断等

## 中断

start信号之后，发出设备地址，在第九个时钟就会产生一个中断，根据i2c流程图来编写中断程序

每传输完一个数据将产生一个中断，i2c操作的主体在中断服务程序，分为读写操作



见002

最后测试

1. 中断没产生：未配置gpio用于iic
2. 只产生了一次中断并且出错：tx err，no ack
3. 第一次读ok，再次写卡死，复位写仍卡死，上电写ok：读最后一个数据时，不要回应ack给at24cxx

