# mini2440裸机试炼之——IIC控制EEPROM数据传输

## **内容**:

向EEPROM(AT24C02)内部地址0x00—0xff，依次写入0x00—0xff，然后再读出数据。

## IIC总线介绍：

IIC（Inter-Integrated Circuit，I2C）总线是一种由PHILIPS公司开发的两线式串行总线，用于连接微处理器及其外围设备。在iic总线上，只需要两条线：串行数据线SDA和串行时钟线SCL,便可完成通信

## IIC要点

1. 清IIC中断标志语句rIICCON &= ~0x10;一定要在读写寄存器IICDS的后面，中断是读写寄存器后发生的；
2. 由于EEPROM的读取速度并不快，所以每次读写中断都需要短暂的延时函数；
3. 在对AT24C02A进行读取数据时，**在发送带有读命令的从设备地址后，AT24C02A会再返回一个从设备地址信息或从设备内存地址信息作为应答，所以一定要把该字节读取后抛弃，因为它不是我们所要读取的信息；**
4. 按照AT24C02A的时序，在发送从设备地址字节时，它的最低位是0表示写，1表示读。但对于s3c2440来说，不用人为设置这一位，即是0是1都无所谓，因为这一位是由s3c2440根据是主设备发送模式还是主设备接收模式来**自动设置**。（所以都默认使用0xa0）；

## EEPROM介绍

AT24CXX系列是带有iic总线接口的EEPROM，其中主要包括AT24C01/02/08/16等，其容量（bits x页）分别为128 x 8/256 x 8/1024 x 8/2048 x 8/；对于AT24C02A的三位地址线都是写死的，因为在进行读写操作时使用8位地址已经足够，所以三位地址线写死作为片选，对于AT24C02A的三位地址线第一位必须写死，后两位可以作为内部页地址。因为AT24C02A的大小超过了256byte，8为寻址，已经没法使用到芯片内所有的空间。因此对于后面两位也就可以由程序决定了。

### 写EEPROM中的数据

在写数据的时候，AMR9作为主设备，EEPROM是从设备

第一种写byte方式（我的程序中是使用这个方式）：

写一个byte实际上需要发送三次数据。在这个过程中，主设备为发送状态。第一个数据——设备地址。第二数据——ARM9想写的EEPROM中相对于数据存储的首地址的偏移地址。第三个数据——想写入到EEPROM中的具体数据。最后停止。

第二种页写方式：

其实写页与写byte应该是一致的，第一个数据——设备地址。第二数据——ARM9想写的EEPROM中相对于数据存储的首地址的偏移地址（但是这个地址是首地址。AT24C02一页是8byte，AT24C04/08一页是16byte。所以在写页的时候最多写一页的大小，如果写太多就会重新又从首地址开始，以前写的会被覆盖掉。）。第三、四、······数据——就是你想写入到EEPROM中的数据。最后停止

### 读EEPROM中的数据

主设备仍然是ARM9，从设备是EEPROM，是发送状态还是接收状态，这个iic会自动设置；

第一种读当前地址数据：

第一个数据——（主设备现在处于发生状态）发送从设备地址，

第二个数据——（主设备处于接收状态）ARM9接收数据，注意此时是NO ACK。再停止。（要在产生NO ACK后在读取数据这时数据会是稳定的。网上有问为什么在读IIC最后需要读两次，我自己实验了，只需要最后一次就行，）

第二种随机读数据方式：（我的程序中是使用这个方式）

第一个数据——（主设备处于发生状态），发送一个从设备地址。第一个设备地址是用来从设备匹配的，也在文档中被称为a “dummy” byte write sequence

第二个数据——（主设备处于发生状态），发送一个想读取数据在EEPROM中相对于数据存储的首地址的偏移地址。

第三个数据——（主设备处于发生状态），发送一个从设备地址。这是特定要求这样发送的。。（在这里主设备会被配置为接收状态），这此发送设备地址是用来同时调整主设备状态的。

第四个数据——（主设备处于接收状态）需要读的数据。也是一个NO ACK，与读当前地址类似。最后再停止。

第三种读序列地址（页读）：

第一个数据——（主设备处于发送状态），发出设备地址，并配置主设备为接收状态。为后面接收数据准备

　　第二、三···个数据——（主设备处于接收状态），前面每个数据都会发送ACK，最后一个数据是一个NO ACK。

　　再停止。

以上这些，主要要注意主设备状态的调整，以及为NO ACK时的处理，后面有事例程序，能够比较清楚的看到怎么进行处理的

## 代码块

|  |
| --- |
| #include "def.h"  #include "2440addr.h"  #include "2440lib.h"  static unsigned int i**,**j**,**save\_E**,**save\_PE**;**  static U8 data**[**256**];**  static volatile int flag**;** //用于标识是否收到应答信号，改标识在终端处理程序中被清0  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  IIC 初始化函数  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void iic\_init**(**void**){**  rGPECON **|=** 0xa0000000**;** //GPE15:IICSDA , GPE14:IICSCL (rGPEUP[14:15]没有上拉选项)    //[7]=1允许应答发生 [6]=0 IICCLK = fPCLK /16 PCLK 50MHz, IICCLK = 3.17MHz, Tx Clock = 0.198MHz  // 允许IIC总线Tx/Rx中断使能 IICCON[3:0]=16  rIICCON **=** **(**1**<<**7**)** **|** **(**0**<<**6**)** **|** **(**1**<<**5**)** **|** **(**0xf**);**    rIICADD **=** 0x10**;** //从地址 表示2440作为从设备的时候的地址，在这里2440是作为一个主设备存在的，所以没有作用。  rIICSTAT **=** 0x10**;** //IIC总线数据输出使能读写  rIICLC **=** **(**1**<<**2**)|(**1**);** //使能滤波器 延时五个时钟  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  IIC中断函数  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void \_\_irq Iic\_isr**(**void**)**  **{**  flag **=** 0**;**  rSRCPND **=** 0x1**<<**27**;** //清除中断  rINTPND **=** 0x1**<<**27**;**  **}**  void iic\_isr**(**void**){**  rINTMSK **&=** **~(**0x1**<<**27**);**  pISR\_IIC **=** **(**unsigned**)**Iic\_isr**;**//中断入口  **}**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  写EEPROM中的数据  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Wr24C080**(**U32 slvAddr**,** U32 addr**,** U8 data**)**  **{**  flag**=**1**;** //以下都是应答标志 标志位置1  rIICDS **=** slvAddr**;** //\*\*设备地址（EEPROM的地址为1010）  rIICSTAT **=** 0xf0**;** // 主设备发送模式，写）产生起始信号  rIICCON **&=** **~**0x10**;** //以下都是清中断标志  **while(**flag **==** 1**)** //当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**      flag **=**1 **;**  rIICDS **=** addr**;** //偏移地址  rIICCON **&=** **~**0x10**;**  **while(**flag **==** 1**)** //当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**    flag **=**1 **;**  rIICDS **=** data**;** //要写入的具体数据  rIICCON **&=** **~**0x10**;**  **while(**flag **==** 1**)** //当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**    rIICSTAT **=** 0xd0**;** //Stop MasTx condition  rIICCON **=** 0xaf**;** //Resumes IIC operation.  Delay**(**1**);**    **}**    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  读EEPROM中的数据  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void Rd24C080**(**U32 slvAddr**,** U32 addr**,** U8 **\***data**)**  **{**  unsigned char temp**;**  flag**=**1**;**  rIICDS **=** slvAddr**;** //设备地址（EEPROM的地址为1010）  rIICSTAT **=** 0xf0**;** // 主设备发送模式用来发送slvAddr和addr,，启动  rIICCON **&=** **~**0x10**;**  **while(**flag **==** 1**)** // 当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**    flag **=**1 **;**  rIICDS **=** addr**;** //偏移地址  rIICCON **&=** **~**0x10**;**  **while(**flag **==** 1**)** // 当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**    flag**=**1**;**  rIICDS **=** slvAddr**;** //从设备地址。这是特定要求这样发送的。。（在这里主设备会被配置为接收状态），  //这此发送设备地址是用来同时调整主设备状态的。  rIICSTAT **=** 0xb0**;** // 主设备接收模式用来接收数据，启动  rIICCON **&=** **~**0x10**;**  **while(**flag **==** 1**)** // 当发送从地址完成之后会收到ACK信号，在中断处理函数中将该标志置为0  Delay**(**1**);**    // 注意：读取下面这个字节必须进行，因为在发送带有读命令的从设备地址后，  // AT24C02A会再返回一个从设备地址信息或从设备内存地址信息作为应答，所以一定要把该字节读取后抛弃，因为它不是我们所要读取的信息；  flag **=**1 **;** // 标志位置1  temp **=** rIICDS**;** // 抛弃第一自己  rIICCON **&=** **~**0x10**;** // 清中断标志  **while(**flag**)**  Delay**(**1**);**    **if(**i**==**256**-**1**)** rIICCON **&=** **~**0x80**;**//如果是最后一个数据 不再响应  rIICCON **=** 0x2f**;** //Resumes IIC operation with NOACK.  **\***data **=** rIICDS**;**  Delay**(**1**);**  rIICSTAT **=** 0x90**;** //Stop MasTx condition  rIICCON **=** 0xaf**;** //Resumes IIC operation.    Delay**(**1**);**    **}**    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  IIC 子main函数  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void IIC**(**void**)**  **{**    Uart\_Printf**(**"\n \*\*\* IIC TEST \*\*\* \n"**);** // Delay(1000); //1    //保护现场  save\_E **=** rGPECON**;**  save\_PE **=** rGPEUP**;**    iic\_init**();** //IIC初始化  iic\_isr**();** //IIC中断    Uart\_Printf**(**"写入数据到EEPROM\n"**);** //向EEPROM写数据i  **for(**i**=**0**;**i**<**256**;**i**++){** Wr24C080**(**0xa0**,(**U8**)**i**,**i**);** Delay**(**1**);}** //注意这个延时不能少，否则出现有些数据无法写入的问题    **for(**i**=**0**;**i**<**256**;**i**++)** data**[**0**]** **=** 0**;** //初始化data[i]    Uart\_Printf**(**"读EEPROM中的数据\n"**);** //EEPROM读取数据到data[i]  **for(**i**=**0**;**i**<**256**;**i**++){** Rd24C080**(**0xa0**,(**U8**)**i**,&(**data**[**i**]));** Delay**(**1**);}**    **for(**i**=**0**;**i**<**16**;**i**++)** //打印data[i]数组  **{** Uart\_Printf**(**"\n"**);**  **for(**j**=**0**;**j**<**16**;**j**++)** Uart\_Printf**(**" %2x"**,**data**[**i**\***16**+**j**]);** **}**    //还原现场  rINTMSK **|=** 0x1**<<**27**;**  rGPEUP **=** save\_PE**;**  rGPECON **=** save\_E**;**    **while(**1**){**  **}**  **}** |

## 截图:

