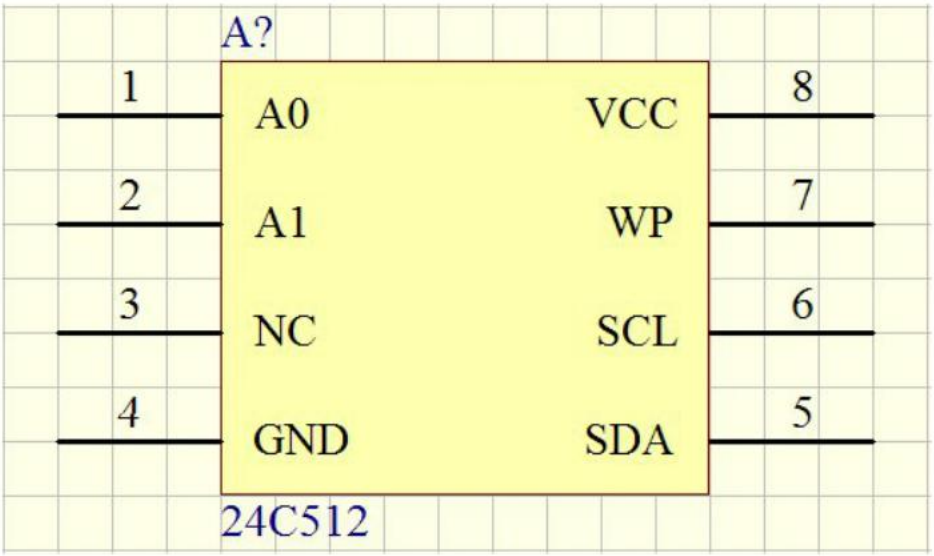
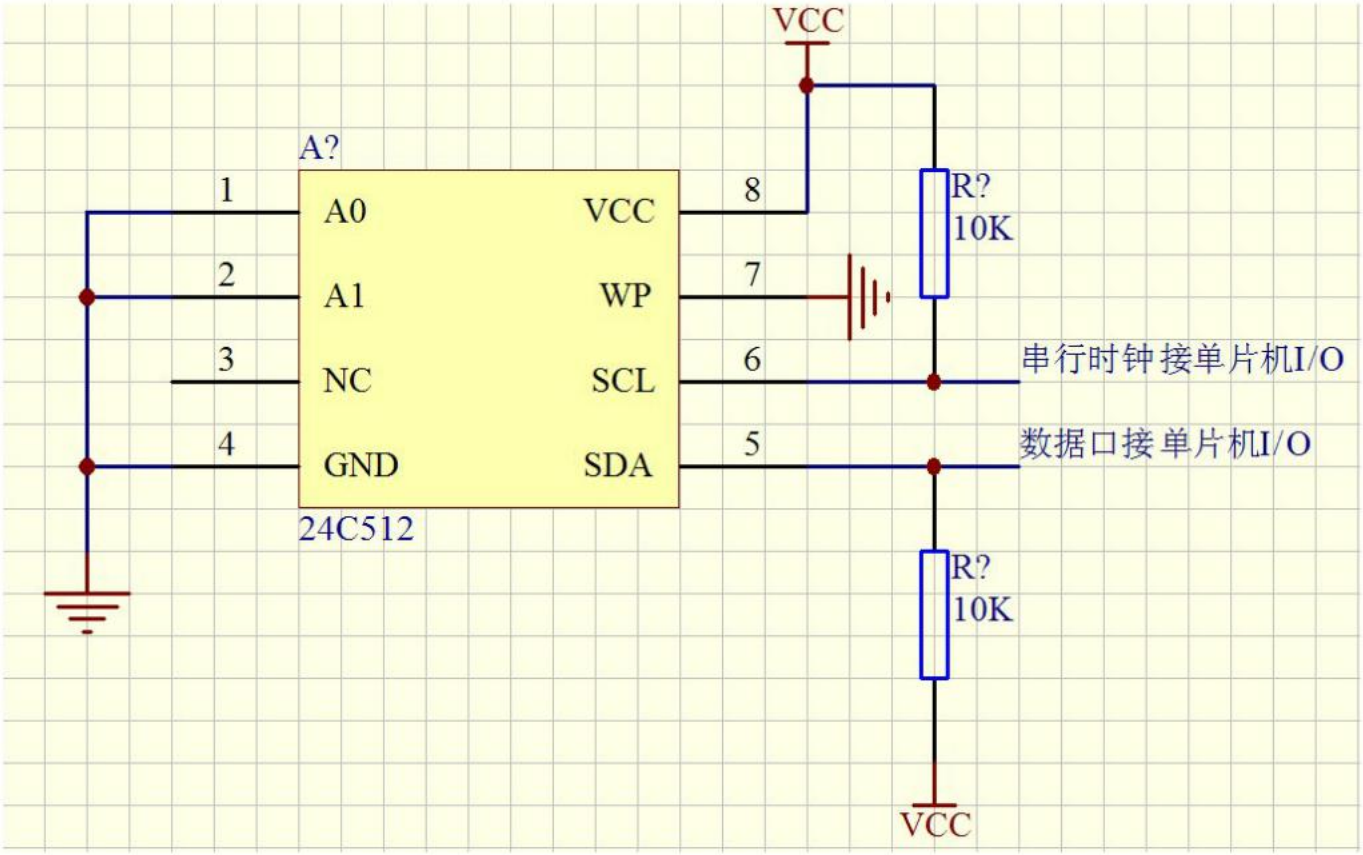
EEPROM:AT24C512

24C512是Atmel公司生产的64KB串行电可擦除可编程存储器（EEPROM）。24C512内部有512K位（bit）共64K字节，分为512页（Page），每页128字节，地址为16位，范围为0000~FFFFH。采用8引脚DIP封装，通过2个地址引脚可以在总线上并联4片24C512。



编号	引脚	功能
1	A0	地址输入，用来区分芯片，一般接地即可，悬空为地址0
2	A1	
3	NC	未使用
4	GND	电源地
5	SDA	串行数据输入输出
6	SCL	串行时钟输入
7	WP	写保护，高电平写保护，悬空接地
8	VCC	可接5v、2.7v、1.8v三种电压

24C512典型连接方式



A0,A1接地：地址为0。多片24C512需要通过高低电平设置不同的地址。
WP接地为可读写，当然，WP也可以通过跳线或软件控。
VCC为5.0v、2.7v、1.8v三种之一，芯片自适应。电压越高，工作频率越高。
SCL,SDA一般接单片机I/O口，由于SDA是漏极开路的，需要接上拉电阻，当然你的I/O带上拉就不用接了。

24C512的驱动方式（一）

24C512使用I2C总线驱动方式。因此，它遵守I2C的协议规范。I2C中有7种基本操作：　　1、发送起始信号；2、发送结束信号；3、发送应答信号；4、总线无应答；
　　5、接收并检测应答信号；6、发送1字节数据；7、接收1字节数据
关于I2C的具体原理及说明详见《I2C总结》。
以下是每种I2C操作的具体代码：I2C.c

0-1、准备工作：包含头文件和数据类型定义

```
#ifndef __msp430x14x            //包含msp430x14x.h
#include "msp430x14x.h"
#endif

#define __i2c                   //文件标志符

#define I2C_DIR P5DIR           //I2C芯片的输入输出切换，这里使用P5端口，用户可根据具体实际情况修改端口寄存器即可
#define I2C_OUT P5OUT           //I2C芯片的输出
#define I2C_IN P5IN             //I2C芯片的输入

#define SDA BIT1                //数据位，P5.1
#define SCL BIT2                //时钟位，P5.2

#ifndef uchar                   //定义数据类型
#define uchar unsigned char     //uchar
#endif

#ifndef uint                    //uint
#define uint unsigned int
#endif
```

0-2、准备工作：延迟函数

```
void I2C_delay(void)
{
  _NOP();        //本函数在MSP430F149，8MHz/8分频主频下通过验证
  _NOP();        //用户可以根据所用430单片机具体的性能适当调整延迟时长
}

//毫秒级延迟，stop信号后的长延迟
void I2C_delay_ms(uint ms)
{
  uint i,j;

  for(i=0;i<ms;i++)
    for(j=0;j<8000;j++);
}
```

0-3、准备工作：初始化I2C

```
void I2C_init(void)                //设置端口方向和初始电平
{
  I2C_DIR |= SCL;                  //SCL设置为输出
  I2C_DIR &= ~SDA;                 //SDA设置为输入

  I2C_OUT &= ~SCL;                 //SCL=0，占用总线
  I2C_delay();

  I2C_stop();                      //I2C_stop() 函数释放总线
}
```

1、start信号：启动I2C

```
void I2C_start(void)
{
  I2C_DIR |= SDA;                 //SDA设置为输出

  I2C_OUT |= SDA;                 //SDA=1
  I2C_delay();

  I2C_OUT |= SCL;                 //SCL=1
  I2C_delay();

  I2C_OUT &= ~SDA;                //SDA=0，SDA完成负跳，启动I2C
  I2C_delay();

  I2C_OUT &= ~SCL;                //SCL=0，SCL钳位在0上，持续占用总线
  I2C_delay();
}
```

2、stop信号：释放I2C

```
void I2C_stop(void)
{
  I2C_DIR |= SDA;                 //SDA设置为输出

  I2C_OUT &= ~SCL;                //SCL=0
  I2C_delay();

  I2C_OUT &= ~SDA;                //SDA=0
  I2C_delay();

  I2C_OUT |= SCL;                 //SCL=1
  I2C_delay();

  I2C_OUT |= SDA;                 //SDA=1，SDA完成正跳，释放I2C总线
  I2C_delay();
}
```

3、ack信号：发送【应答】信号

```
void I2C_ack(void)
{
  I2C_DIR |= SDA;                 //SDA设置为输出

  I2C_OUT &= ~SDA;                //SDA=0，正确应答
  I2C_delay();

  I2C_OUT |= SCL;                 //SCL=1，发送应答信号
  I2C_delay();

  I2C_OUT &= ~SCL;                //SCL=0，SCL拉回低电平，防止误操作
  I2C_delay();

  I2C_OUT |= SDA;                  //SDA=1，SDA拉回高电平，防止误操作
  I2C_delay();
}
```

4、NoAck信号：发送【无应答】信号

```
void I2C_NoAck(void)
{
  I2C_DIR |= SDA;                 //SDA设置为输出

  I2C_OUT |= SDA;                 //SDA=1，错误或无应答标志
  I2C_delay();

  I2C_OUT |= SCL;                 //SCL=1，发送无应答信号
  I2C_delay();

  I2C_OUT &= ~SCL;                //SCL=0，SCL拉回低电平，防止误操作
  I2C_delay();
}
```

5、接收并检测ACK信号

```
uchar I2C_TestAck(void)
{
  uchar ack_;                    //保存接收到的ACK信号

  I2C_OUT |= SCL;                //SCL=1
  I2C_delay();

  I2C_DIR &= ~SDA;               //SDA设置为输入

  ack_ = I2C_IN & SDA;           //获取ACK信号
  I2C_delay();

  I2C_OUT &= ~SCL;               //SCL=0
  I2C_delay();

  if(ack_==0x00)                  //应答有效返回0，应答无效返回1
    return 0;
  else
    return 1;
}
//单片机每发送1个字节的数据、指令或地址，都要检测应答。
```

6、发送1个字节

```
void I2C_SendByte(uchar data_)
{
  uchar i,temp_;                    //temp_用于保存临时数据

  I2C_DIR |= SDA;                   //SDA设置为输出

  for(i=0;i<8;i++)                  //从最高位开始，一位一位发送
  {
    temp_ = data_ & 0x80;

    if(temp_==0x80)
      I2C_OUT |= SDA;               //SDA=1，发送位=1
    else
      I2C_OUT &= ~SDA;              //SDA=0，发送位=0

    I2C_delay();

    I2C_OUT |= SCL;                 //SCL=1，把SDA送到I2C总线上
    I2C_delay();

    I2C_OUT &= ~SCL;                //SCL=0，拉回SCL
    I2C_delay();

    data_ = data_<<1;               //准备下一位数据
  }
}
//使用进位标志位CY来取data_的每一位数据也可以，而且效率还高一些。
```

7、接收1个字节

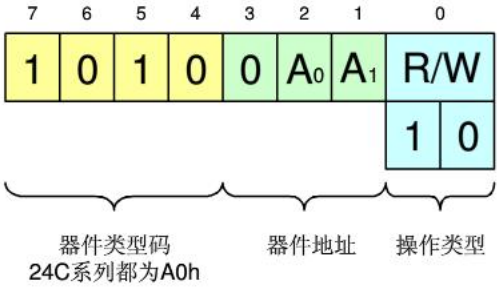
```
uchar I2C_ReceiveByte(void)
{
  uchar i,temp_;                  //temp_用于保存临时数据
  uchar data_=0x00;               //data_用于保存接收数据

  I2C_DIR &= ~SDA;                //SDA设置为输入

  for(i=0;i<8;i++)
  {
    I2C_OUT |= SCL;               //SCL=1
    I2C_delay();

    data_ = data_<<1;             //从最高位开始接收

    temp_ = (I2C_IN & SDA);       //接收数据
    if(temp_==SDA)
      data_ = data_ | 0x01;

    I2C_delay();

    I2C_OUT &= ~SCL;              //SCL=0
    I2C_delay();
  }

  return(data_);                  //返回接收到的数据
}
```

以上内容可参见附录1：I2C.c文件

24C512的驱动方式（二）

24C512的操作时序。
单片机对24C512的操作无非两种：写入和读取，根据读写地址和数据量又可细分为：单个字节写入、页写入、读当前地址数据、读任意地址数据、
连续读取（连续读取当前地址、连续读取任意地址）。
下面配合函数逐一介绍：

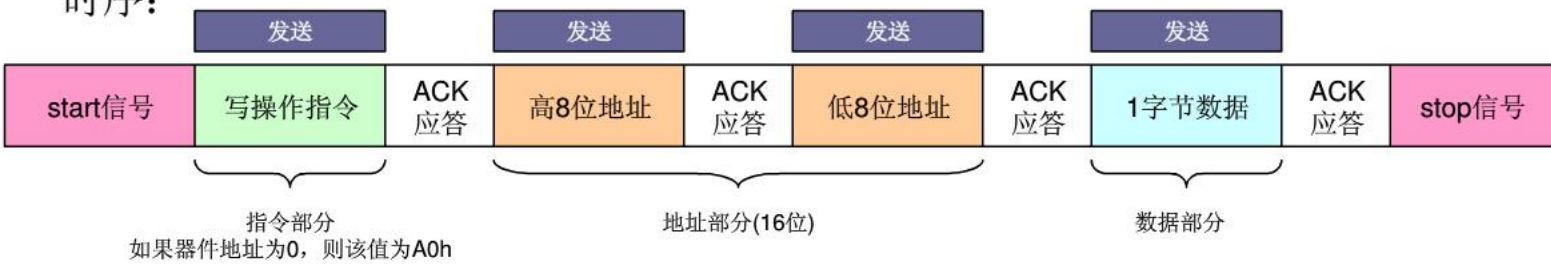
0、准备工作：24C512的指令格式



无论读还是写，操作24C512的第一个部分都是指令，而且24C512只有这么一条指令。

1、写入单个字节

时序：

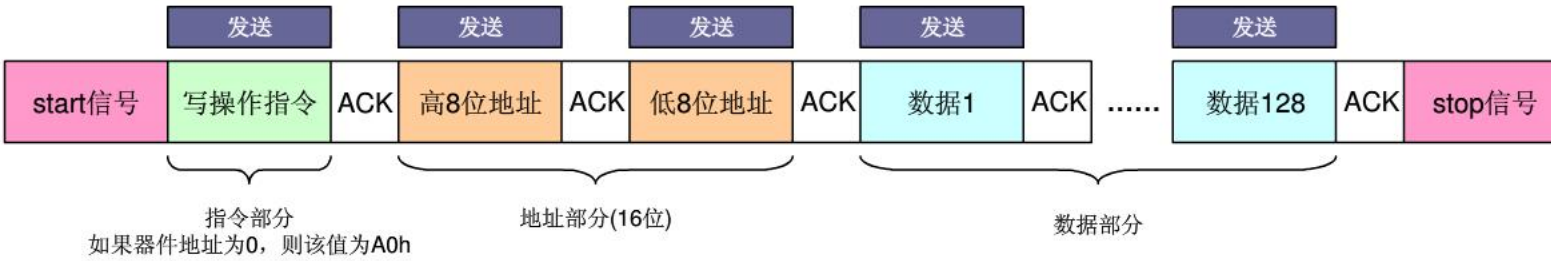


操作开始，单片机需要发送start信号来占用I2C总线，随后发送写操作指令A0h，判断应答信号为有效后再分2次发送写入地址，每次发送完事都要判断24C512返回的应答信号。数据跟在地址后面发送给器件，同样，也要判断ACK应答。
最后发送stop信号来释放I2C总线。

```
uchar _24C_Send_1B(uchar data_,uint addr_)    //写入单个字节
{
  I2C_start();                                //启动总线

  I2C_SendByte(_24C_CMD_W);                   //发送写操作指令
  if(I2C_TestAck()==1)return 1;               //检测ACK，失败返回错误代码

  I2C_SendByte((uchar)(addr_/0x100));         //发送写高8位地址
  if(I2C_TestAck()==1)return 2;               //检测ACK

  I2C_SendByte((uchar)(addr_%0x100));         //发送写低8位地址
  if(I2C_TestAck()==1)return 3;               //检测ACK

  I2C_SendByte(data_);                        //发送数据
  if(I2C_TestAck()==1)return 4;               //检测ACK

  I2C_stop();                                 //释放总线
  I2C_delay_ms(10);

  return 0;
}
```

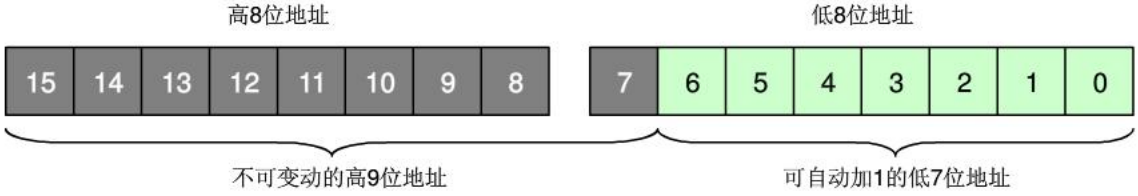
2、页写入

时序：



24C512可以进行每次长达128个字节的页写入操作。页写入时，直到写入完第一个字节前，所有的操作过程都与字节写入完全一样，不同的是单片机不会在写入第一个字节后发送STOP信号，而是继续写入，最多可以再写入127个字节（算上第1次写入的总共不超过128个字节）。在写入全部数据后，单片机必须通过发送STOP信号来停止页写入操作。

页写入时，写入地址的低7位会在接收字节后自动加1，但高9位不会（一共16位地址），因此一次页写入最多只能写128个字节，2^7=128。如果一次写入的数据超过128字节，则会发生地址发生溢出（低7位地址会归零），后续写入的字节回覆盖最前边的数据。



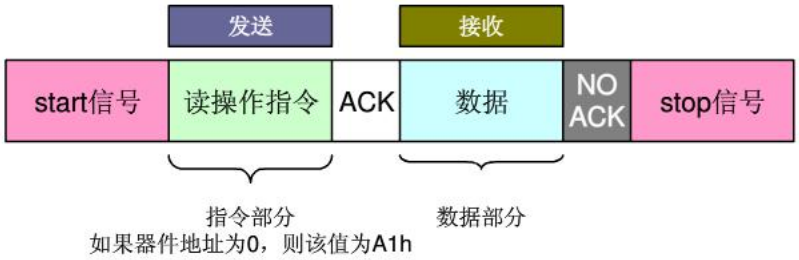
页的起始位置是任意的，但最好设置为128的整数倍，这样可以避免数据覆盖。

页写入函数

```
uchar _24C_Send_128B(uchar data_[],uint addr_)
{
  uchar i=0;

  I2C_start();                                  //启动总线

  I2C_SendByte(_24C_CMD_W);                     //发送写芯片指令
  if(I2C_TestAck()==1)return 1;                 //检验应答，失败返回错误代码

  I2C_SendByte((uchar)(addr_/0x100));           //发送写高8位地址
  if(I2C_TestAck()==1)return 2;

  I2C_SendByte((uchar)(addr_%0x100));           //发送写低8位地址
  if(I2C_TestAck()==1)return 3;

  while((i<128)||(data_[i]!=0x00))              //循环写入数据，可以少于128字节但不能超过128字节
  {
    I2C_SendByte(data_[i]);                     //发送数据
    if(I2C_TestAck()==1)return 4;
    i++;
  }

  I2C_stop();                                   //释放总线
  I2C_delay_ms(10);

  return 0;
}
```

3、从当前地址读取1个字节

时序：

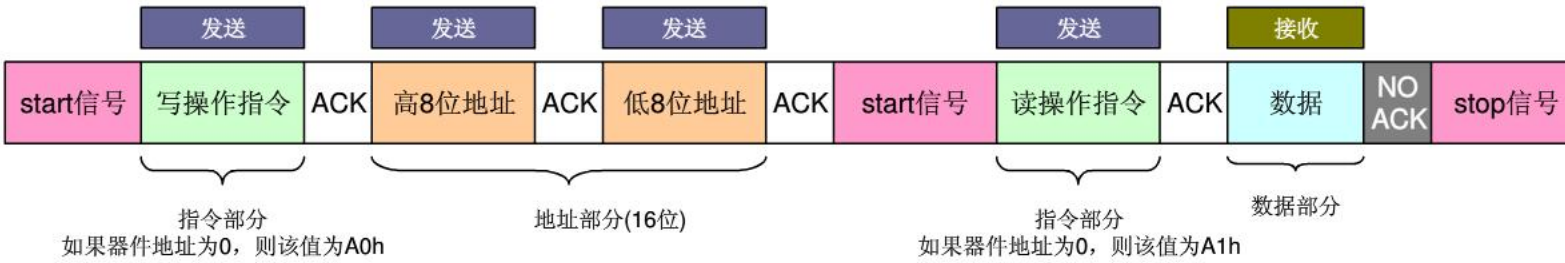


24C512中有一个地址计数器，读写操作会改变指针的值，【读当前地址的数据】指令的地址就是最后一次读写操作的地址值作为数据接收方的单片机在读取完数据后直接发送STOP信号，不回复ACK信号。
由于当前地址是不确定的，所以这个操作读取的数据也不确定是哪个地址上的数据。

```
uchar _24C_ReceiveCurrent_1B(uchar *p)
{
  *p = 0x00;

  I2C_start();                                  //启动总线

  I2C_SendByte(_24C_CMD_R);                     //发送读芯片指令
  if(I2C_TestAck()==1)return 4;

  *p = I2C_ReceiveByte();                       //从24C接收数据

  I2C_NoAck();                                  //无需应答操作
  I2C_stop();                                   //释放总线
  I2C_delay_ms(10);

  return 0;

}
```

4、从指定地址读取1个字节

时序：



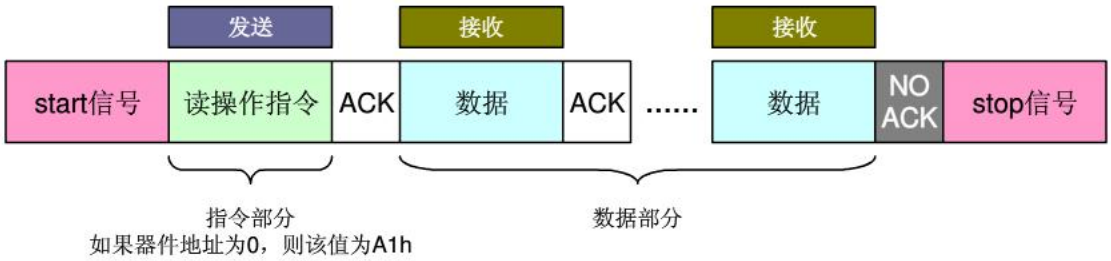
读指定地址的数据，先要写入地址，再重复start信号和操作指令，最后才能读取数据。

从指定地址读取1个字节

```
uchar _24C_Receive_1B(uchar *p,uint addr_)
{
  *p = 0x00;

  I2C_start();                                  //启动总线

  I2C_SendByte(_24C_CMD_W);                     //发送写芯片指令
  if(I2C_TestAck()==1)return 1;                 //检验应答，失败返回错误代码

  I2C_SendByte((uchar)(addr_/0x100));           //发送写高8位地址
  if(I2C_TestAck()==1)return 2;

  I2C_SendByte((uchar)(addr_%0x100));           //发送写低8位地址
  if(I2C_TestAck()==1)return 3;

  I2C_start();                                  //再次启动总线

  I2C_SendByte(_24C_CMD_R);                     //发送读芯片指令
  if(I2C_TestAck()==1)return 4;

  *p = I2C_ReceiveByte();                       //从24C接收数据

  I2C_NoAck();                                  //无需应答操作
  I2C_stop();                                   //释放总线
  I2C_delay_ms(10);

  return 0;

}
```

5、从当前地址连续读取N个字节

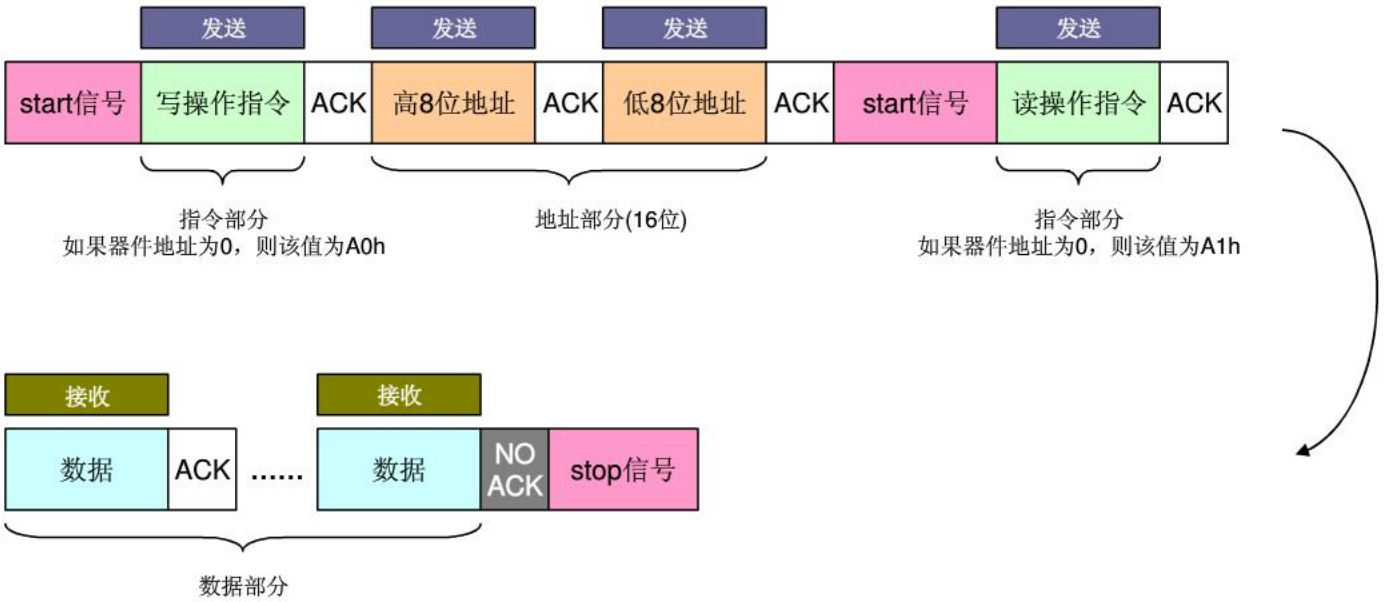
时序：



只要在读取数据以后单片机向24C512发送应答信号就可以继续读取下面的数据，24C512中的地址计数器会自动加1。
单片机必须通过发送STOP信号来终止数据读取，如果读取数据的个数超过了地址计数器的最大值，读到得数据不可靠。

```
uchar _24C_ReceiveCurrent_(uchar data_[],uint count_)
{
  uint i;

  I2C_start();                              //启动总线

  I2C_SendByte(_24C_CMD_R);                 //发送读芯片指令
  if(I2C_TestAck()==1)return 4;

  for(i=0;i<count_;i++)                     //连续读取数据
  {
    data_[i] = I2C_ReceiveByte();           //从24C接收数据
    I2C_ack();                              //应答操作
  }

  I2C_NoAck();                              //无需应答操作
  I2C_stop();                               //释放总线
  I2C_delay_ms(10);

  return 0;

}
```

6、从指定地址连续读取N个字节
时序：



和上一个类似，单片机只要在读取数据后向24C512发送了ACK应答，就可以读下一个数据。读取结束后，单片机要发送NoAck信号和STOP信号，释放总线。

从指定地址连续读取N个字节

```
uchar _24C_Receive_(uchar data_[],uint count_,uint addr_)
{
  uint i;

  I2C_start();                                  //启动总线

  I2C_SendByte(_24C_CMD_W);                     //发送写芯片指令
  if(I2C_TestAck()==1)return 1;                 //检验应答，失败返回错误代码

  I2C_SendByte((uchar)(addr_/0x100));           //发送写高8位地址
  if(I2C_TestAck()==1)return 2;

  I2C_SendByte((uchar)(addr_%0x100));           //发送写低8位地址
  if(I2C_TestAck()==1)return 3;

  I2C_start();                                  //再次启动总线

  I2C_SendByte(_24C_CMD_R);                     //发送读芯片指令
  if(I2C_TestAck()==1)return 4;

  for(i=0;i<count_;i++)
  {
    data_[i] = I2C_ReceiveByte();               //从24C接收数据
    I2C_ack();                                  //应答操作
  }

  I2C_NoAck();                                  //无需应答操作
  I2C_stop();                                   //释放总线
  I2C_delay_ms(10);

  return 0;

}
```

以上就是6种读取操作，EEPROM_24C.c文件的其他部分可以参考附录2中的内容。

分享知识是一种精神！是知识传承的唯一途径！

附录说明：
整个工程由6个文件组成：
主程序文件：main.c
初始化系统时钟文件：InitSys.c
1602液晶驱动文件：LCD_1602.c
24C512文件：EEPROM_24C.c
I2C文件：I2C.c
延迟函数文件：delay.c
都分别收录在附录1-6中，如果需要整个工程文件夹可以向buythem@163.com邮箱索取。

KiFi　　2012-08-11

//附录1：I2C.c文件

```
#ifndef __msp430x14x
#include "msp430x14x.h"
#endif

#define __i2c                  //文件包含标志符

#define I2C_DIR P5DIR          //I2C芯片的输入输出切换
#define I2C_OUT P5OUT          //I2C芯片的输出
#define I2C_IN P5IN            //I2C芯片的输入

#define SDA BIT1               //数据位
#define SCL BIT2               //时钟位

#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

//-------------------------------------------------------------------------------
//延迟函数
//-------------------------------------------------------------------------------
void I2C_delay(void)
{
  _NOP();
  _NOP();
}

void I2C_delay_ms(uint ms)
{
  uint i,j;

  for(i=0;i<ms;i++)
    for(j=0;j<8000;j++);
}

//-------------------------------------------------------------------------------
//Start信号，启动总线
//-------------------------------------------------------------------------------
void I2C_start(void)
```

//附录2：EEPROM_24C.c

```
#ifndef __msp430x14x
#include "msp430x14x.h"
#endif

/*包含I2C*/
#ifndef __i2c
#include "I2C.c"
#endif

#define __eeprom_24c

#define _24C_CMD_W 0xA0          //写指令，器件地址为0
#define _24C_CMD_R 0xA1          //读指令，器件地址为0

#ifndef uchar
#define uchar unsigned char
#endif
#ifndef uint
#define uint unsigned int
#endif

//-------------------------------------------------------------------------------
//向24C发送1个字符
//-------------------------------------------------------------------------------
uchar _24C_Send_1B(uchar data_,uint addr_)
{
  I2C_start();                          //启动总线

  I2C_SendByte(_24C_CMD_W);               //发送写芯片指令
  if(I2C_TestAck()==1)return 1;           //检验应答，失败返回错误代码

  I2C_SendByte((uchar)(addr_/0x100));     //发送写高8位地址
  if(I2C_TestAck()==1)return 2;

  I2C_SendByte((uchar)(addr_%0x100));     //发送写低8位地址
  if(I2C_TestAck()==1)return 3;

  I2C_SendByte(data_);                    //发送数据
  if(I2C_TestAck()==1)return 4;

  I2C_stop();                           //释放总线
  I2C_delay_ms(10);
```

//附录3： LCD_1602.c

```
#ifndef __msp430x14x
#include msp430x14x.h
#endif

#ifndef __delay
#include "delay.c"
#endif

#define __lcd1602

#define LCD_CTL_DIR P6DIR                    //控制线IO方向
#define LCD_CTL_SEL P6SEL                    //控制线IO功能选择

#define LCD_DATA_DIR P4DIR                   //数据线IO方向
#define LCD_DATA_SEL P4SEL                   //数据线IO功能选择
#define LCD_DATA_OUT P4OUT                   //数据线IO
#define LCD_DATA_IN P4IN                     //数据线返回值

#define LCD_RS_0 P6OUT &= ~BIT6              //指令/数据切换位RS=0
#define LCD_RS_1 P6OUT |= BIT6

#define LCD_RW_0 P6OUT &= ~BIT5              //读/写切换位RW=0
#define LCD_RW_1 P6OUT |= BIT5

#define LCD_EN_0 P6OUT &= ~BIT4              //LCD使能EN=0
#define LCD_EN_1 P6OUT |= BIT4

#define LCD_BUSY 0x80                        //忙标志

#ifndef uchar
#define uchar unsigned char
#endif
#ifndef uint
#define uint unsigned int
#endif


//检测忙信号并遇忙等待
void chk_Busy_LCD(void)
{
  LCD_DATA_DIR = 0x00;                       //数据线IO端口方向为输入
```

//附录4： InitSys.c

```
#ifndef __msp430x14x
#include msp430x14x.h
#endif

#define __InitSys

//初始化XT2时钟
void init_XT2(void)
{
  unsigned int i;

  BCSCTL1 &= ~XT2OFF;          //起震

  do
  {
    IFG1 &= ~OFIFG;            //设置起震标志，如果起震成功则设置成功

    for(i=0;i<0xFF;i++);       //延迟，等待起震成功

  }while((IFG1&OFIFG)!=0);     //判断是否起震

  BCSCTL2 = SELM1 + SELS;      //MCLK,SMCLK时钟为XT2
}
```

//附录5： delay.c

```
#define __delay

#ifndef uint
#define uint unsigned int
#endif

//毫秒延迟
void Delay_ms(uint ms)
{
  uint i,j;

  for(i=0;i<ms;i++)
    for(j=0;j<8000;j++);

}

//普通延迟
void Delay(uint n)
{
  uint i;

  for(i=0;i<n;i++);
}
```

//附录6： main.c

```
#ifndef __msp430x14x
#include "msp430x14x.h"
#endif

/*系统初始化函数*/
#ifndef __InitSys
#include "InitSys.c"              //msp430初始化函数
#endif

/*LCD-1602*/
#ifndef __lcd1602
#include "LCD_1602.c"             //LCD,1602
#endif

/*24C*/
#ifndef __eeprom_24c
#include "EEPROM_24C.c"
#endif

/*数据类型定义*/
#ifndef uchar
#define uchar unsigned char
#endif
#ifndef uint
#define uint unsigned int
#endif


/******************************************************************************
**main函数
******************************************************************************/
void main( void )
{
  uchar str[]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p',
  'q','r','s','t','u','v','w','x','y','z',0x00};

  uchar data_,r,i;


  /*初始化*/
  WDTCTL = WDTPW + WDTHOLD;              //关狗
```