第14章 IIC控制器

14.1 IIC协议简介

14.2 软件模拟I2C协议程序分析

14.3 模拟I2C总线协议读写AT24CXX系列EEPROM实验

——IIC接口E2PROM存储器AT24C02

电子与通信工程系

华东理工大学信息学院

14.1 IIC协议简介

14.1 IIC协议简介

IIC(Inter-Integrated Circuit BUS) 集成电路总线，该总线NXP(原PHILIPS)公司设计，多用于主控制器和从器件间的一种主从数据交互通信，在小数据量场合使用，传输距离短。

I2C可以支持0kHz～5MHz的设备：普通模式（100kHz）、快速模式（400kHz）、快速模式增强版（1MHz）、高速模式（3.4MHz）和超高速模式（5MHz）。

14.1 IIC协议简介

14.1.1 I2C物理层

在I2C通信总线上，可连接多个I2C通信设备，支持多个通信主机和多个通信从机。I2C通信只需要2条双向总线——一条数据线SDA（Serial Data Line，串行数据线），一条时钟线SCL（Serial Clock Line，串行时钟线）。

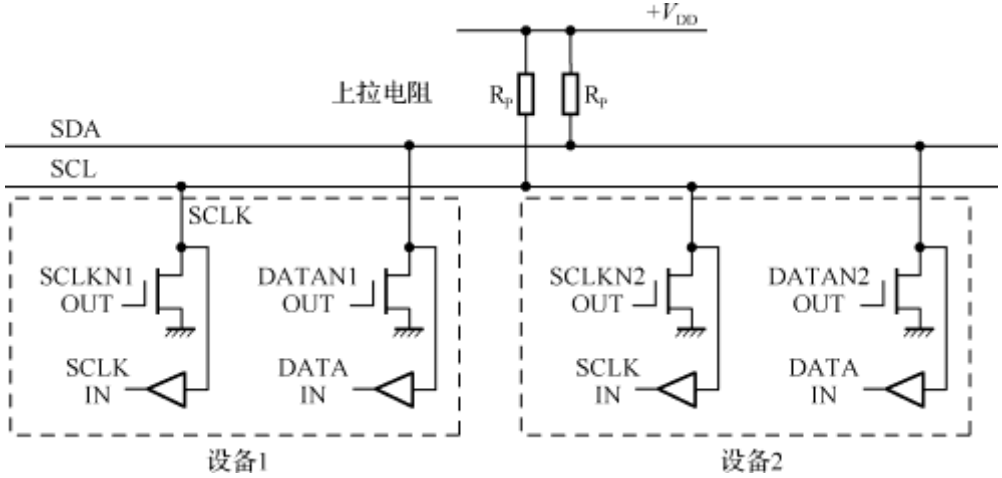
- SDA - 数据信号线(serial DAta)
- SCL - 时钟信号线(serial CLock)

14.1 IIC协议简介

14.1.1 I2C物理层

- 发送器(Transmitter) - 将数据发送到总线的设备。 发射器可以是放置的设备总线上的数据（主发送器）或来自另一个的数据请求的响应设备（从发射器）。
- 接收器(Receiver) - 从总线接收数据的设备。

- 主机(Master) - 初始化传输，生成时钟信号并终止传输的组件传递。 主设备可以是发射器或接收器。
- 从机(Slave) - 主站寻址的设备。 可以是接收器或发射器。



16.1 IIC协议

16.1.2 IIC的物理层

a. 只要求两条总线线路，一条是串行数据线SDA，一条是串行时钟线SCL。（IIC是半双工，而不是全双工）。

b.每个连接到总线的器件都可以通过唯一的地址和其它器件通信，主机/从机角色和地址可配置，主机可以作为主机发送器和主机接收器。

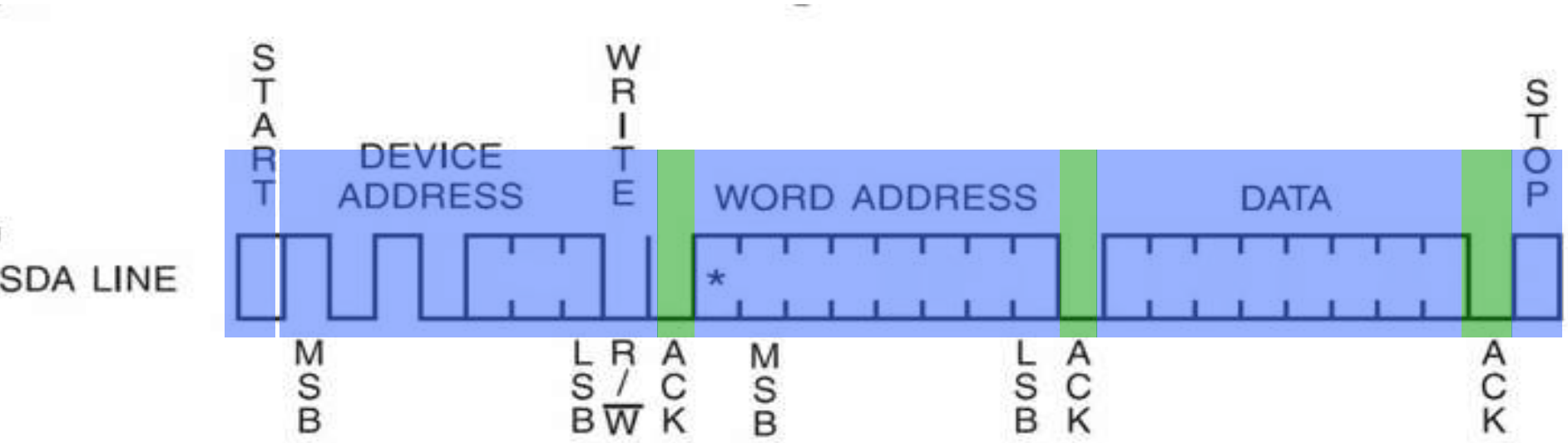
c.IIC是真正的多主机总线，(而类似的SPI总线协议在每次通信前都需要把主机定死，IIC可以在通讯过程中，改变主机)，如果两个或更多的主机同时请求总线，可以通过冲突检测和仲裁防止总线数据被破坏。但是任意时刻只能有一个主机。

d.传输速率在标准模式下可以达到100kb/s,快速模式下可以达到400kb/s。

e.连接到总线的IIC设备数量只是受到总线的最大负载电容400pf限制。

14.1 IIC协议简介

14.2.1 I2C协议层

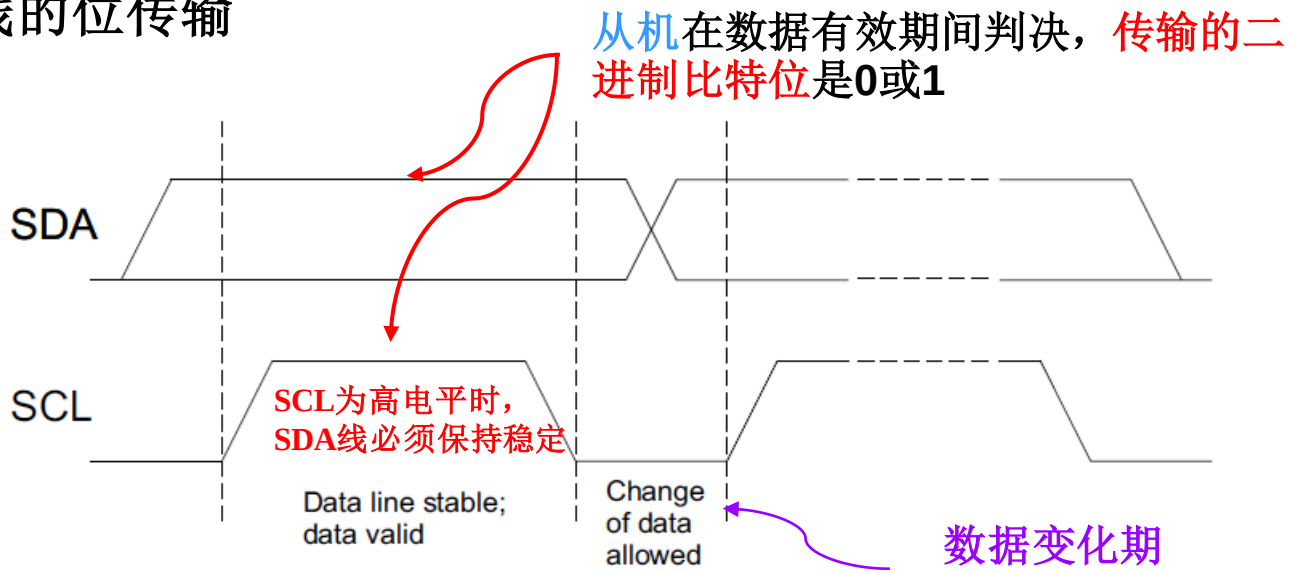


典型时序-写操作

14.1 IIC协议简介

14.1.2 I2C协议层

1. I2C总线的位传输



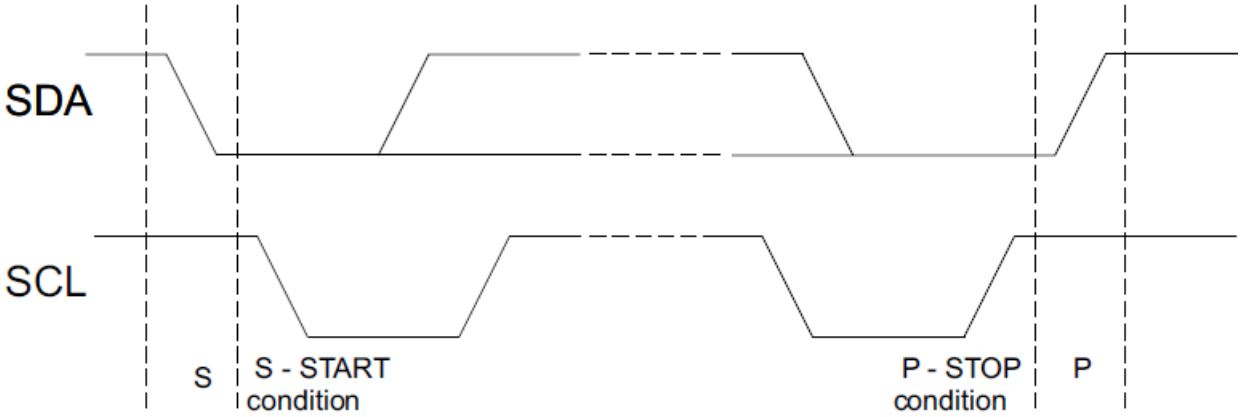
数据传输：SDA的数据在SCL高电平期间被写入从机。所以SDA的数据变化要发生在SCL低电平期间。

在时钟的高电平周期内，SDA线上的数据必须保持稳定，数据线仅可以在时钟SCL为低电平时改变。

14.1 IIC协议简介

14.1.2 I2C协议层

2．I2C总线的开始信号和结束信号



注意：起始和终止信号都是由主机发出的，连接到I2C总线上的器件，若具有I2C总线的硬件接口，则很容易检测到起始和终止信号。总线在起始条件之后，视为忙状态，在停止条件之后被视为空闲状态。

总线空闲状态：

SDA：高电平

SCL：高电平

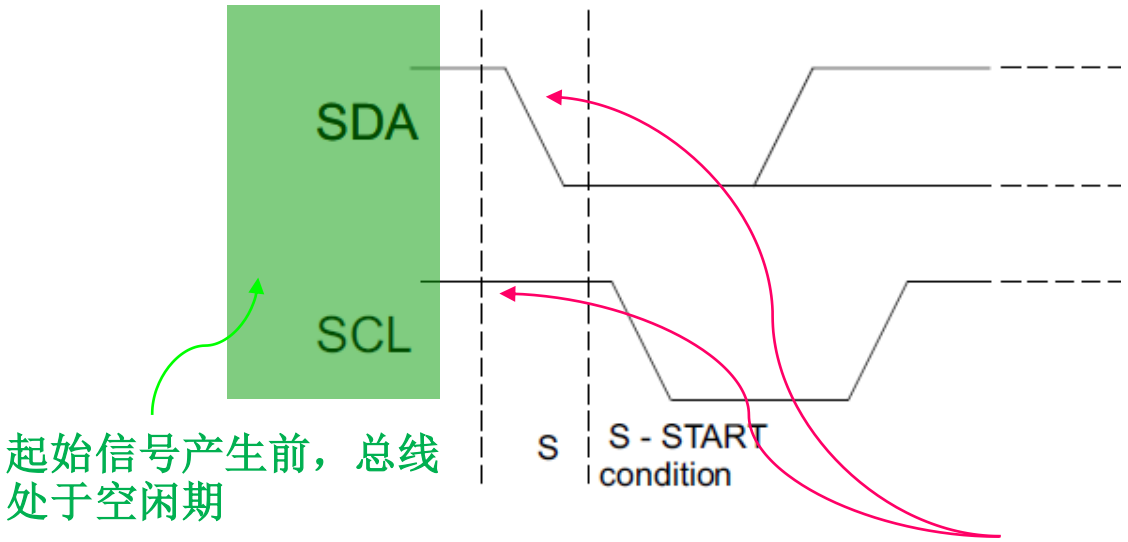
起始条件：当SCL为高电平的时候，SDA线上由高到低的跳变被定义为起始条件。

结束条件：当SCL为高电平的时候，SDA线上由低到高的跳变被定义为停止条件。

14.1 IIC协议简介

14.1.2 I2C协议层

开始信号

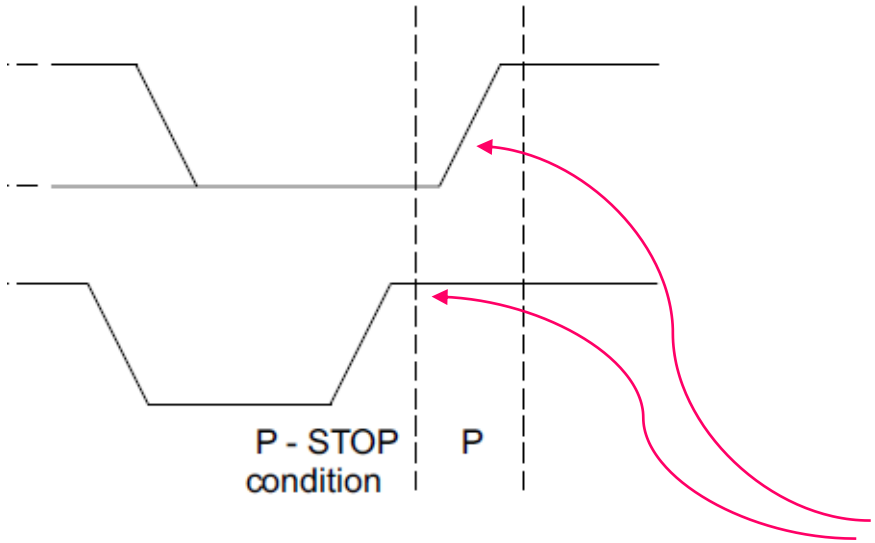


起始位：SCL为高电平期间，SDA出现下降沿

14.1 IIC协议简介

14.1.2 I2C协议层

结束信号

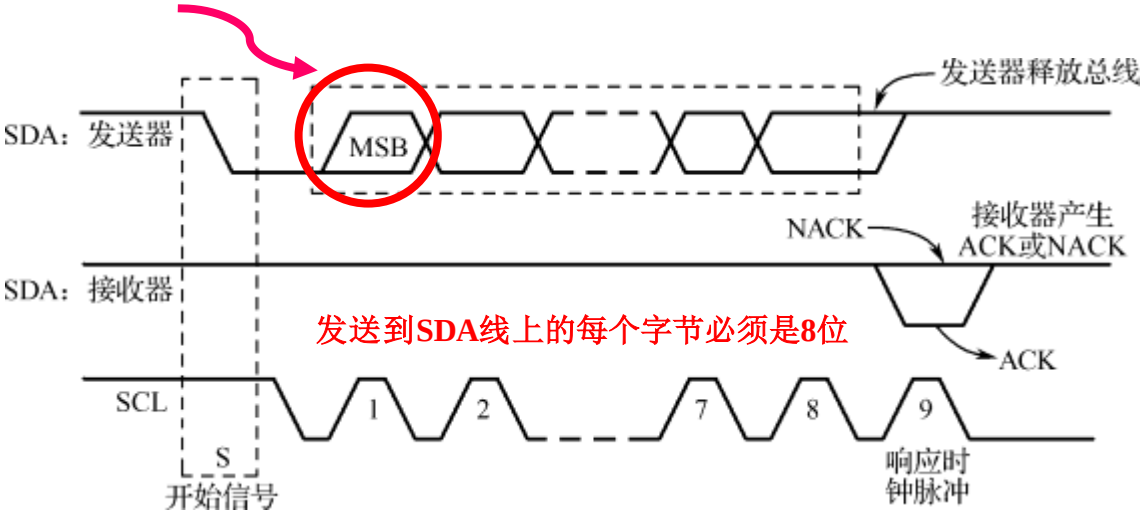


终止位：SCL为高电平期间 SDA出现上升沿

14.1 IIC协议简介

14.1.2 I2C协议层

3. I2C总线的字节格式



当IIC主机（不一定是发送端还是接受端）将8位数据或命令传出后，会将SDA信号设置为输入，等待从机应答(等待SDA由高电平拉为低电平)。

若从机正确应答，表明数据或者命令传输成功，否则传输失败，注意，应答信号是数据接收方发送给数据发送方的。

4. I2C应答信号

应答：

每当主机向从机发送完一个字节的数据，主机总是需要等待从机给出一个应答信号，以确认从机是否成功接收到了数据。

从机应答主机所需要的时钟仍是主机提供的，应答出现在每一次主机完成8个数据位传输后紧跟着的时钟周期，低电平0表示应答，1表示非应答。

14.1 IIC协议简介

14.1.2 I2C协议层

5. I2C总线的仲裁机制

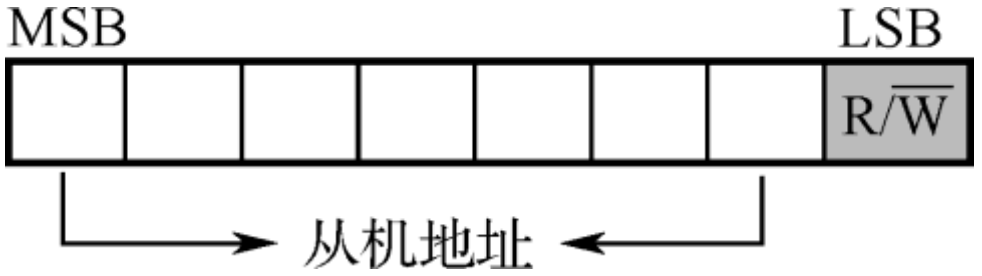
- 多主机 - 多个主机能够在没有冲突下的情况下，同时共存于总线上，能够通过总线仲裁避免碰撞或数据丢失。

- 仲裁 - 预先安排的时序，一次只授权一个主机控制总线。

当SCL线是高电平时，仲裁在SDA线上发生。在其他主机发送低电平时，发送高电平的主机将会断开它的数据传输级，因为总线上的电平与它自己的电平不同（线与连接）。

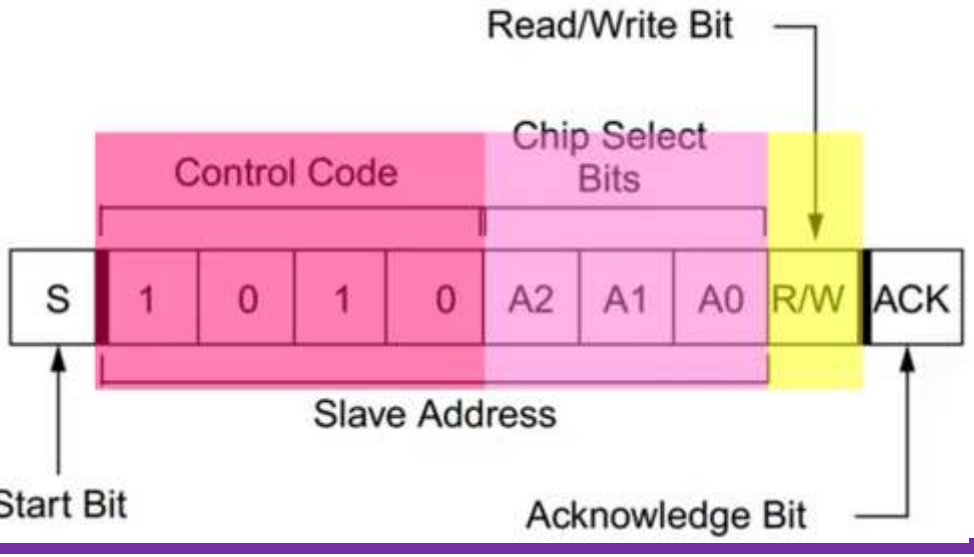
14.1 IIC协议简介

14.1.2 I2C协议层



6．从机地址和子地址

在开始条件（S）后，主机发送一个从机地址（或叫作器件地址），指该器件在I2C总线上被主机寻址的地址，地址共有7bit，紧接着的第8bit是数据的读写标志位（0表示写，1表示读）。



14.1 IIC协议简介

14.1.2 I2C协议层

7. 主机发送数据流程

I2C总线主机发送数据流程（8位从机地址）。

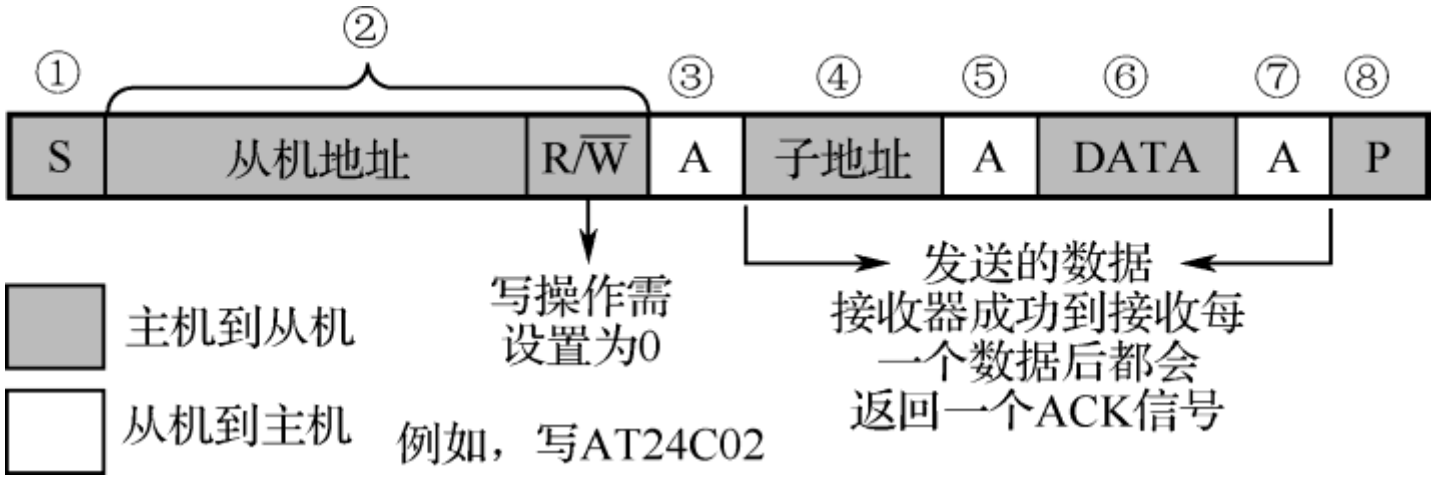


图14-6 I2C总线主机发送数据流程（8位从机地址）

14.1 IIC协议简介

14.1.2 I2C协议层

7. 主机发送数据流程

I2C总线主机发送数据流程（16位从机地址）。

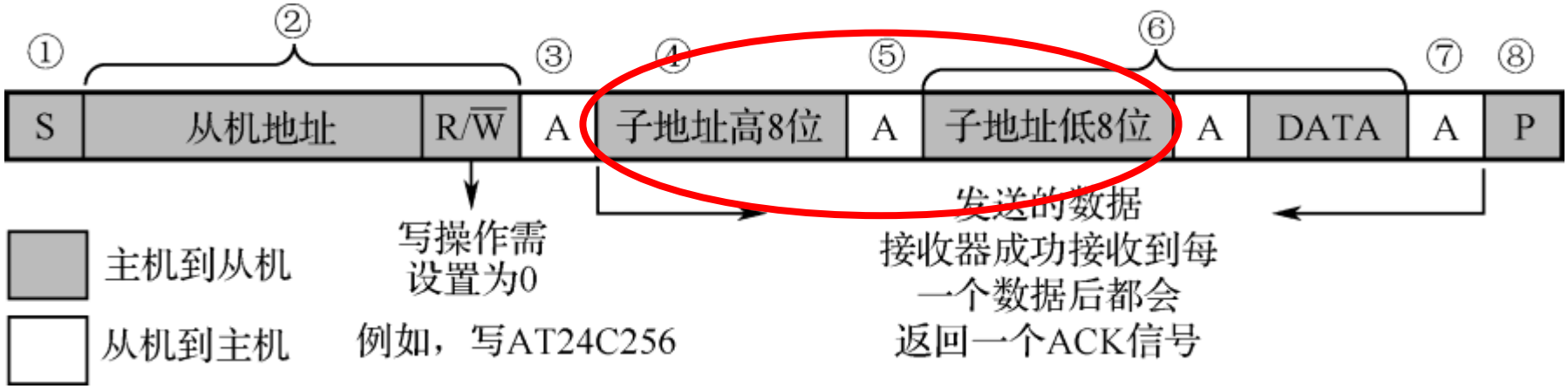


图14-7 I2C总线主机发送数据流程（16位从机地址）

14.1 IIC协议简介

14.1.2 I2C协议层

8. 主机接收数据流程

I2C总线主机接收数据流程。

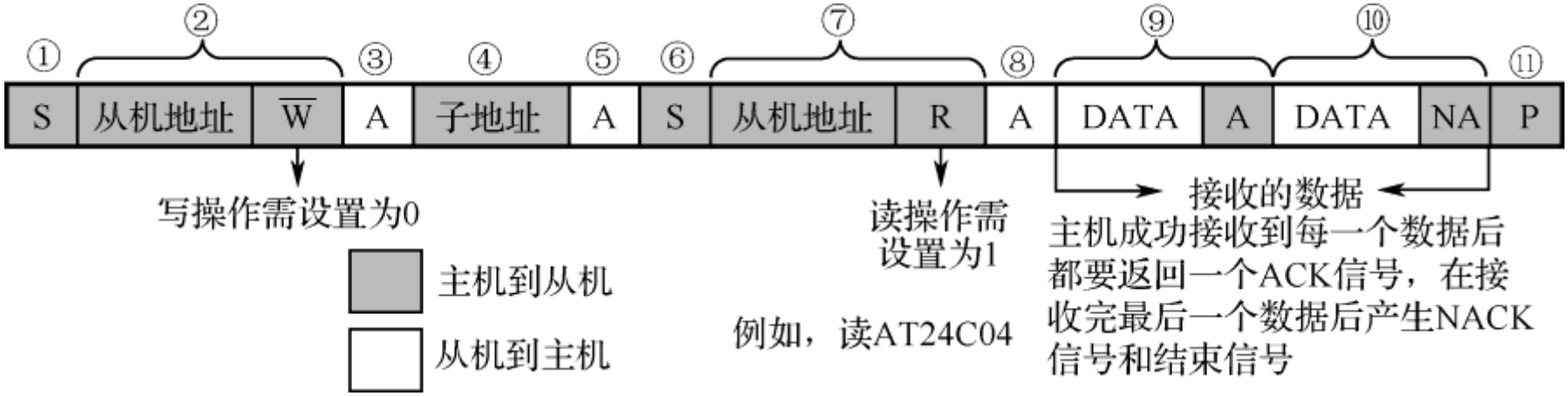


图14-8 I2C总线主机接收数据流程

14.2 软件模拟I2C协议程序分析

14.2 软件模拟I2C协议程序分析

16.3.1 模拟IIC协议概述

由于I2C总线占用的I/O仅需要2根，在很多的实际使用过程中，会使用GPIO引脚来模拟I2C的SDA引脚和SCL引脚，并使用程序来实现I2C协议时序。

软件模拟I2C协议的优点如下。

（1）不需要专门的硬件I2C的控制器。

（2）引脚可以任意分配，方便PCB布线。

（3）软件修改灵活。

缺点：由于采用软件指令会产生时间的延时，不能用于一些时间要求较高的场合。

14.2 软件模拟I2C协议程序分析

14.2.1 I2C引脚配置

1. 引脚工作模式初始化

将SCL设置为输出模式

```
void SCLOUT(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOH,ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
GPIO_Init(GPIOH, &GPIO_InitStructure);
}
```

14.2 软件模拟I2C协议程序分析

14.2.1 I2C引脚配置

1. 引脚工作模式初始化

将SDA设置为输入模式

```
void SDA_IN(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOH,ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOH, &GPIO_InitStructure);
}
```

14.2 软件模拟I2C协议程序分析

14.2.1 I2C引脚配置

1. 引脚工作模式初始化

将SDA设置为输出模式

```
void SDA_OUT(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOH,ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;
GPIO_Init(GPIOH, &GPIO_InitStructure);
}
```

14.2 软件模拟I2C协议程序分析

14.2.1 I2C引脚配置

1. 引脚工作模式初始化

初始化IIC

```
void IIC_Init(void)
{
  SDA_OUT();
  SCL_OUT();

  IIC_SCL_SET;
  IIC_SDA_SET;
}
```

14.2 软件模拟I2C协议程序分析

14.2.1 I2C引脚配置

2．I2C引脚读写控制

IO操作函数

```
#define IIC_SCL_ Hi    GPIO_SetBits(GPIOH,GPIO_Pin_4) //SCL输出高电平
#define IIC_SCL_ Lo    GPIO_ResetBits(GPIOH,GPIO_Pin_4) //SCL输出低电平
#define IIC_SDA_ Hi    GPIO_SetBits(GPIOH,GPIO_Pin_5) //SDA输出高电平
#define IIC_SDA_ Lo    GPIO_ResetBits(GPIOH,GPIO_Pin_5) //SDA输出低电平

#define IIC_SDA_In     GPIO_ReadInputDataBit (GPIOH,GPIO_Pin_5) //输入
SDA状态
```

14.2 软件模拟I2C协议程序分析

IIC所有操作函数

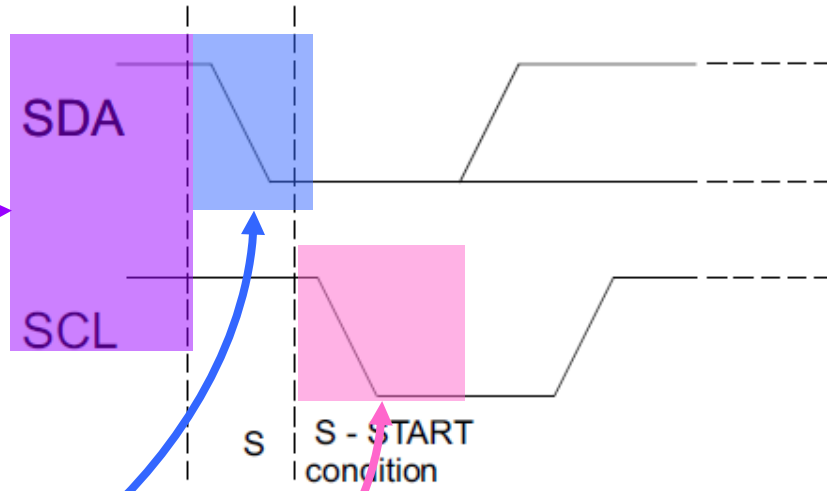
```
void IIC_Init(void                    //初始化IIC的IO口
void IIC_Start(void);                 //发送IIC开始信号
void IIC_Stop(void);                  //发送IIC停止信号
void IIC_Send_Byte(u8 txd);           //IIC发送一个字节
u8 IIC_Read_Byte(unsigned char ack);  //IIC读取一个字节
u8 IIC_Wait_Ack(void);                //IIC等待ACK信号
void IIC_Ack(void);                   //IIC发送ACK信号
void IIC_NAck(void);                  //IIC不发送ACK信号

void IIC_Write_One_Byte(u8 daddr,u8 addr,u8 data);
u8 IIC_Read_One_Byte(u8 daddr,u8 addr);
```

14.2 软件模拟I2C协议程序分析

14.2.2 软件模拟开始信号和结束信号

主机产生IIC起始信号

```
void IIC_Start(void)
{
        SDA_OUT(); //sda线输出

模拟    IIC_SDA_Hi;
总线    IIC_SCL_Hi      ;
空闲

        IIC_CLK_Wait   //延时

        IIC_SDA_Lo;   // 在SCL为高电平时，将SDA由高变低

        IIC_CLK_Wait

        IIC_SCL_Lo     ;   //钳住I2C总线，准备发送或接收数据
}
```



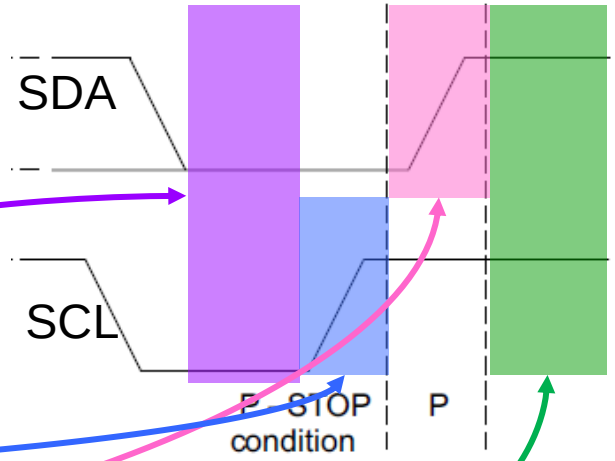
14.2 软件模拟I2C协议程序分析

14.2.2 软件模拟开始信号和结束信号

主机产生IIC停止信号

```
void IIC_Stop(void)
{
        SDA_OUT(); //SDA配置为输出模式
        IIC_SCL_Lo   ;
        IIC_SDA_Lo;
        IIC_CLK_Wait;
        IIC_SCL_Hi;
        IIC_CLK_Wait;
        IIC_SDA_Hi;// 在 SCL为高电平时，将SDA由低变高
        IIC_CLK_Wait


}
```

总线均拉低

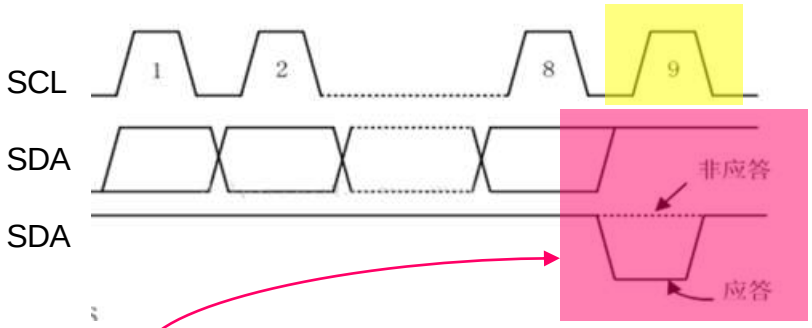


均拉高维持至少4us
表示总线空闲

14.2 软件模拟I2C协议程序分析

14.2.3 模拟检测ACK信号

```
//主机等待应答信号到来
//返回值：  1，接收应答失败
//          0，接收应答成功
u8 IIC_Wait_Ack(void)
{
        u8 ucErrTime=0;
        SDA_IN();       //SDA设置为输入
        IIC_CLK_Wait
        IIC_SCL_Hi      ;
        IIC_CLK_Wait
        while(IIC_SDA_In)
        {
                ucErrTime++;
                if(ucErrTime>250)
                {
                        IIC_Stop();
                        return 1;
                }
        }
        IIC_SCL_Lo      ;   //时钟输出0
        return 0;
}
```

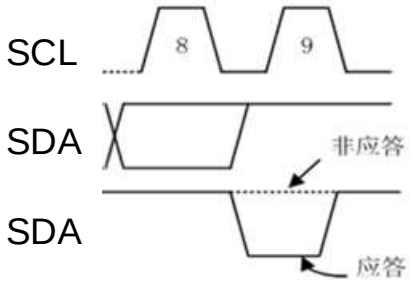


如果等待从机应答信号超时，主机发出STOP信号，结束本次操作

14.2 软件模拟I2C协议程序分析

14.2.4 软件模拟产生ACK信号和NACK信号

```
//主机产生ACK应答
void IIC_Ack(void)
{
    IIC_SCL_RESET;
    SDA_OUT();
    IIC_SDA_RESET;
    delay_us(2);
    IIC_SCL_SET;
    delay_us(2);
    IIC_SCL_RESET;
}
```




```
//主机不产生ACK应答
void IIC_NAck(void)
{
    IIC_SCL_RESET;
    SDA_OUT();
    IIC_SDA_SET;
    delay_us(2);
    IIC_SCL_SET;
    delay_us(2);
    IIC_SCL_RESET;
}
```

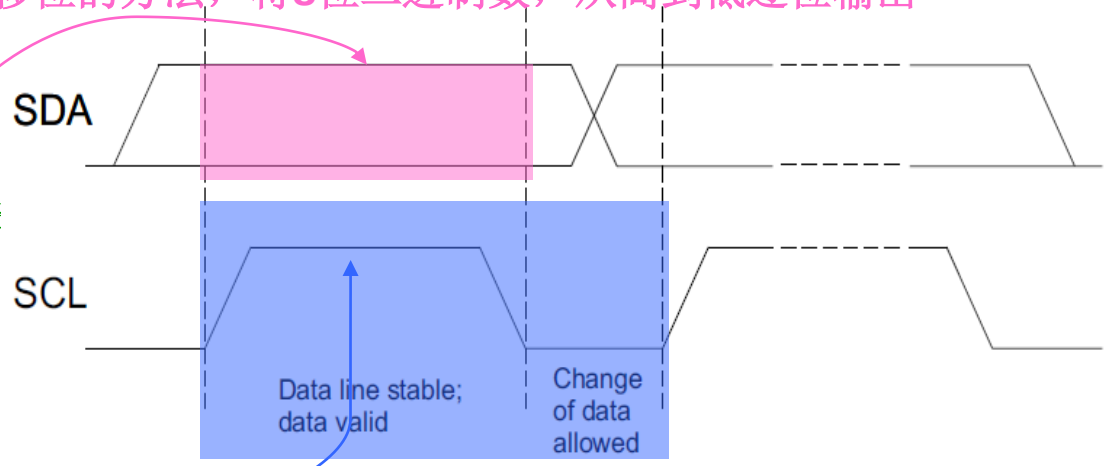
思考：为什么这里的函数是主机产生ACK应答/非应答，难道不应当是从机产生吗？

14.2 软件模拟I2C协议程序分析

14.2.5 软件模拟发送一个字节数据

```
//IIC发送一个字节
void Send_Byte(u8 txd)
{
    u8 t;
    SDA_OUT();
    IIC_SCL_RESET;//拉低时钟开始
    for(t=0;t<8;t++)
    {
        if((txd<<t)&0x80)
            IIC_SDA_SET;
        else
            IIC_SDA_RESET;
        delay_us(2);    //对某些器件这二个延时都是必须的
        IIC_SCL_SET;
        delay_us(2);
        IIC_SCL_RESET;
        delay_us(2);
    }
}
```

通过移位的方法，将8位二进制数，从高到低逐位输出



例如：c2
1100 0010
1000 0000

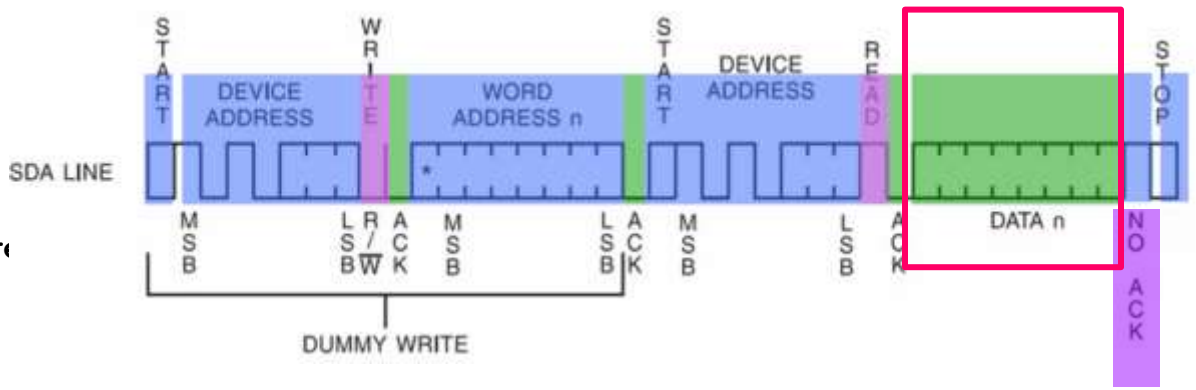
IIC_SDA=最高位MSB

Txd<<1;次高位变到最高位

14.2 软件模拟I2C协议程序分析

14.2.6 软件模拟接收一个字节数据

```
//读1个字节

u8 Read_Byte(unsigned char ack)
{
    unsigned char i,receive=0;
    SDA_IN();  //SDA设置为输入
    for(i=0;i<8;i++ )
    {
        IIC_SCL_RESET;
        delay_us(2);
        IIC_SCL_SET;
        receive<<=1;
        if(READ_SDA)receiv
        delay_us(1);
    }
    return receive;
}
```

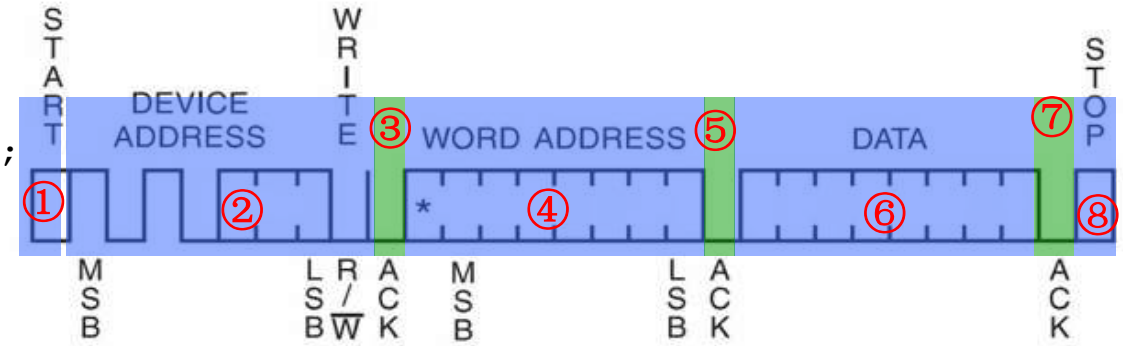


14.2 软件模拟I2C协议程序分析

14.2.7 软件模拟I2C完整写操作

```
uint8_t IIC_SendStr(uint8_t sla,uint8_t
suba,uint8_t *s,uint8_t no)
{
uint8_t i;

① IIC_Start ();      /*启动总线*/
② Send_Byte(sla);    /*发送器件地址*/
③ if(IIC_Wait_Ack())    return(0);
④ Send_Byte(suba);  /*发送器件子地址*/
⑤ if(IIC_Wait_Ack())    return(0);

for(i=0;i<no;i++)
{
⑥Send_Byte(*s);  /*发送数据*/
⑦ if(IIC_Wait_Ack())   return(0);
s++;
}
⑧ IIC_Stop();  /*结束总线*/
return(1);
}
```

```
//IIC总线完整   写多个字节   小于一页
//uint8_t sla    器件地址
//uint8_t suba    器件内部寻址地址
//uint8_t *t      需要写数据指针
//uint8_t no           写数据数量
//返回      1：成功    0：失败
```

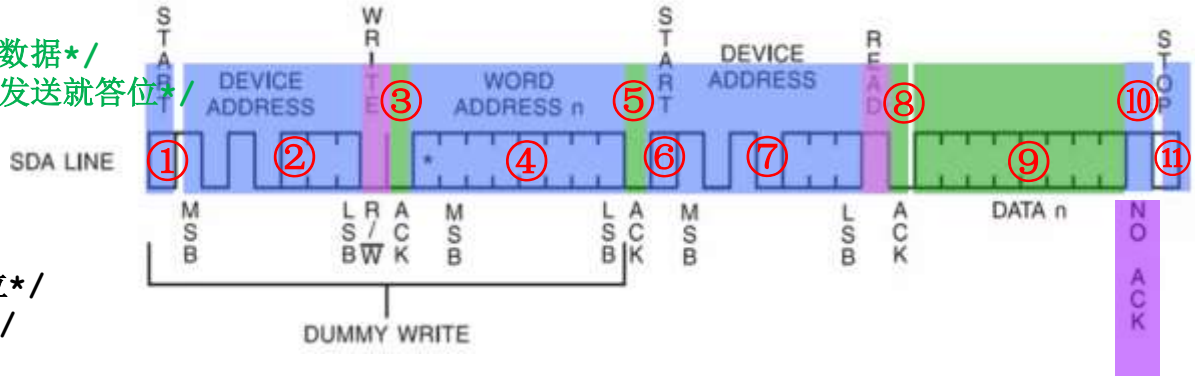


14.2 软件模拟I2C协议程序分析

14.2.8 软件模拟I2C完整读操作

```
uint8_t IIC_RcvStr(uint8_t sla,uint8_t
suba,uint8_t *s,uint8_t no)
{
      uint8_t i;
      ① IIC_Start();                    /*启动总线*/
      ② Send_Byte(sla);                 /*发送器件地址*/
      ③ if(IIC_Wait_Ack())   return(0);
      ④ Send_Byte(suba);                 /*发送器件子地址/
      ⑤ if(IIC_Wait_Ack())   return(0);

      ⑥ IIC_Start();
      ⑦ Send_Byte(sla+1);
      ⑧ if(IIC_Wait_Ack())   return(0);
      for(i=0;i<no-1;i++)
      {
⑨         *s=Read_Byte(); /*接收数据*/
          IIC_Ack(); /*连续读时，发送就答位*/
          s++;
      }
      *s=Read_Byte();

      ⑩ IIC_NAck(); /*发送非应位*/
      ⑪ IIC_Stop(); /*结束总线*/
      return(1);
}
```

```
//IIC总线完整   读多个字节   时序
//uint8_t sla     器件地址

//uint8_t suba     器件内部寻址地址

//uint8_t *t        接收数据空间指针

//uint8_t no              读数据数量
//返回       1：成功     0：失败
```



14.3 模拟I2C总线协议读写AT24CXX系列EEPROM实验

14.3模拟I2C总线协议读写AT24CXX系列EEPROM实验

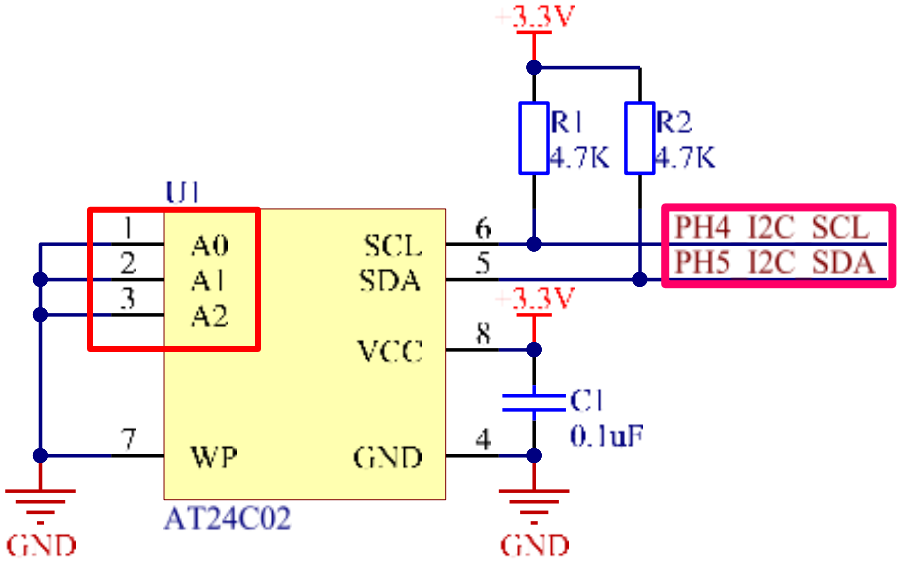
14.3.1 AT24Cxx概述

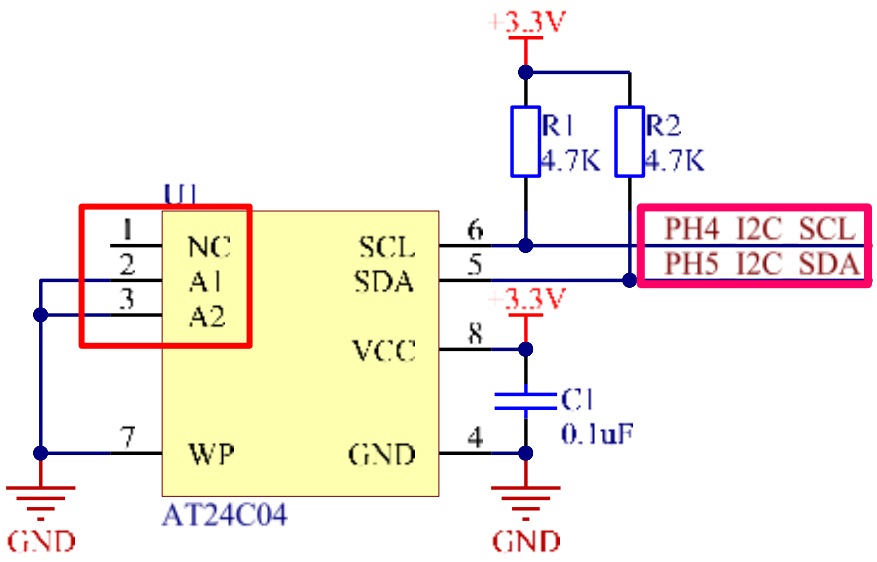
AT24C02/04/08/16是串行CMOS E2PROM， 内部含有。该器件通过IIC总线进行操作，并有一个专门的写保护功能。

02-2K bit=256字节, 04-7K bit=512字节,

08-8K bit=1K字节, 16-16K bit=2K字节。

管脚名称	功能
A0、A1、A2	器件地址选择
SDA	串行数据、地址
SCL	串行时钟
WP	写保护
VCC	+1.8V~6.0V工作电压
VSS	地





14.3模拟I2C总线协议读写AT24CXX系列EEPROM实验

14.3.1 AT24Cxx概述

•**SCL 串行时钟**　AT24Cxx串行时钟输入管脚用于产生器件所有数据发送或接收的时钟，这是一个输入管脚。

•**SDA 串行数据/地址**　AT24Cxx 双向串行数据/地址管脚用于器件所有数据的发送或接收，SDA 是一个开漏输出管脚，可与其它开漏输出或集电极开路输出进行线与。

•**A0、A1、A2 器件地址输入端**　这些输入脚用于多个器件级联时设置器件地址，当这些脚悬空时默认值为0。当使用AT24Cxx 时最大可级联8个器件。如果只有一个AT24C02被总线寻址，这三个地址输入脚（A0、A1、A2 ）可悬空或连接到Vss or GND。AT24C02具备全部三个引脚。AT24C04的A0引脚悬空不用。AT24C08的A1、A0引脚悬空不用。AT24C16的A2、A1、A0引脚悬空不用。

•**WP** 写保护　如果WP管脚连接到Vcc，所有的内容都被写保护只能读。当WP管脚连接到Vss or GND 或悬空允许器件进行正常的读/写操作。