I²C接口

12

- I2C接口结构和特点
- Exynos 4412内部的I2C控制器
- I2C应用实例

12.1 I²C接口结构和特点

1. I²C总线简介

I²C(Inter Intergrated Circuit，也称I²C)总线是由PHILIPS公司开发的两线式串行总线，用来连接微控制器及其外围设备，是微电子通信控制领域广泛采用的一种总线标准。具有如下的特点：

- 具有两条总线线路，即一条串行数据线SDA和一条串行时钟线SCL。
- 每个连接到总线上的器件都可以通过唯一的地址联系主机。
- 它是一个真正的多主机总线，数据传输通过冲突检测和仲裁防止数据被破坏。
- 串行的8位双向数据传输位速率更高。
- 连接到相同总线的IC数量只受到总线最大电容400 pF的限制。

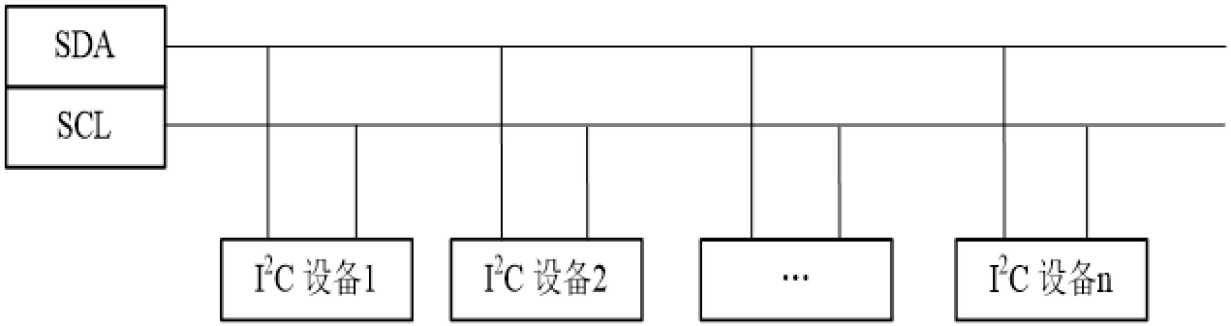

图12.1 I²C总线上多个设备互连示意图

2. I²C总线相关术语

术语名称	功 能 描 述
发送器	发送数据到总线的器件
接收器	从总线接收数据的器件
主机	发起/停止数据传输，提供时钟信号的器件
从机	被主机寻址的器件
多主机	可以有多个主机试图去控制总线，但是不会破坏数据
仲裁	当多个主机试图去控制总线时，通过仲裁可以使得只有一个主机获得总线的控制权

表12.1 I²C总线相关术语

3. I²C总线的信号类型

I²C总线在传送数据的过程中共有三种类型的信号：启动信号、结束信号和响应信号。

(1) 启动信号(S)：SCL为高电平时，SDA由高电平向低电平跳变，开始传输数据。

(2) 停止信号(P)：SCL为低电平时，SDA由低电平向高电平跳变，结束传输数据。

(3) 响应信号(ACK)：接收器在接收到8位数据后，在第9个时钟周期时，拉低SDA电平。

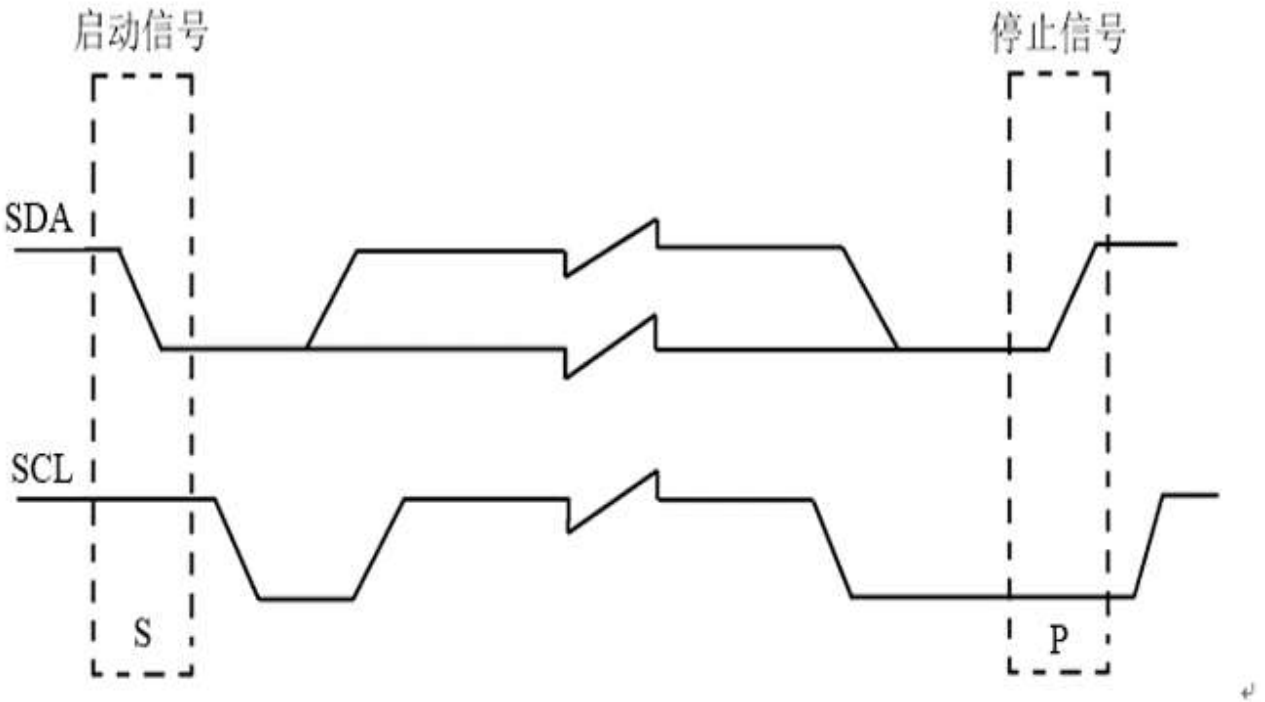

图12.2 启动信号和停止信号示意图

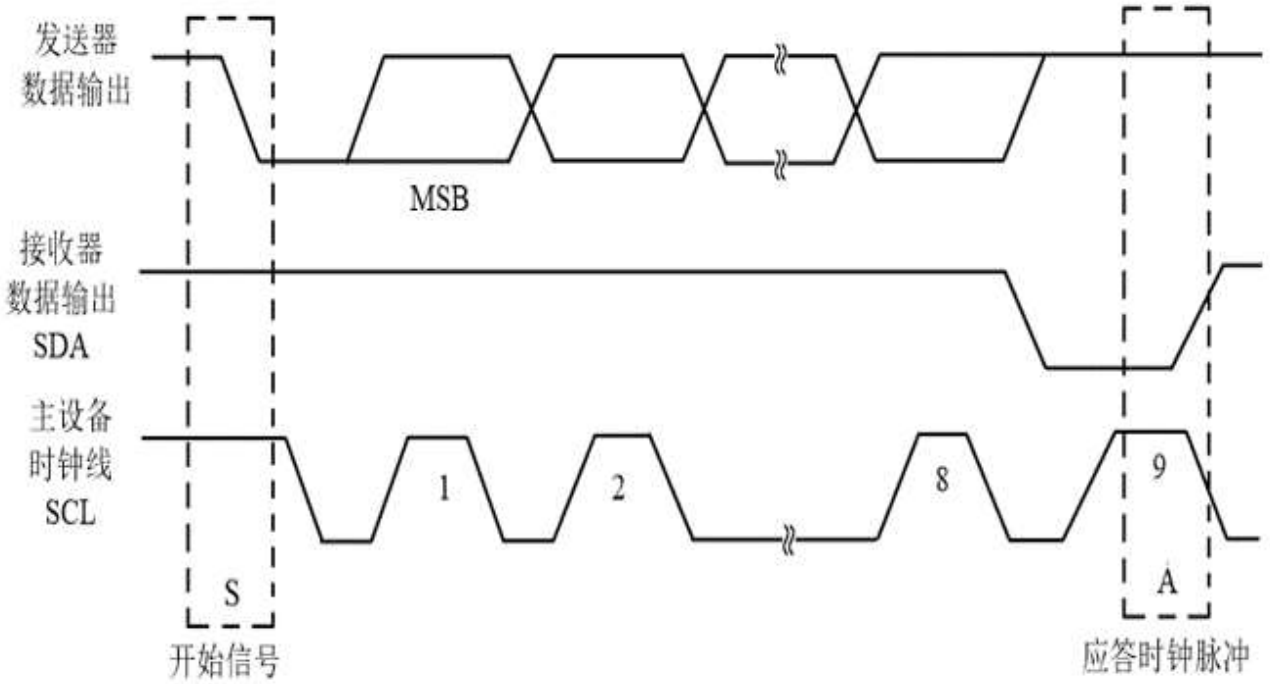

图12.3 应答信号(ACK)

4. I²C总线的数据传输格式

在一次传输的过程中，主机先发出S信号，然后发出8位数据。这8位数据中前7位为从机的地址，第8位表示传输的方向(0表示写操作，1表示读操作)。被选中的从机发出应答信号。紧接着传输一系列字节及其响应位。最后，主机发出P信号，结束本次传输。图12.5所示为I²C总线数据传输格式。

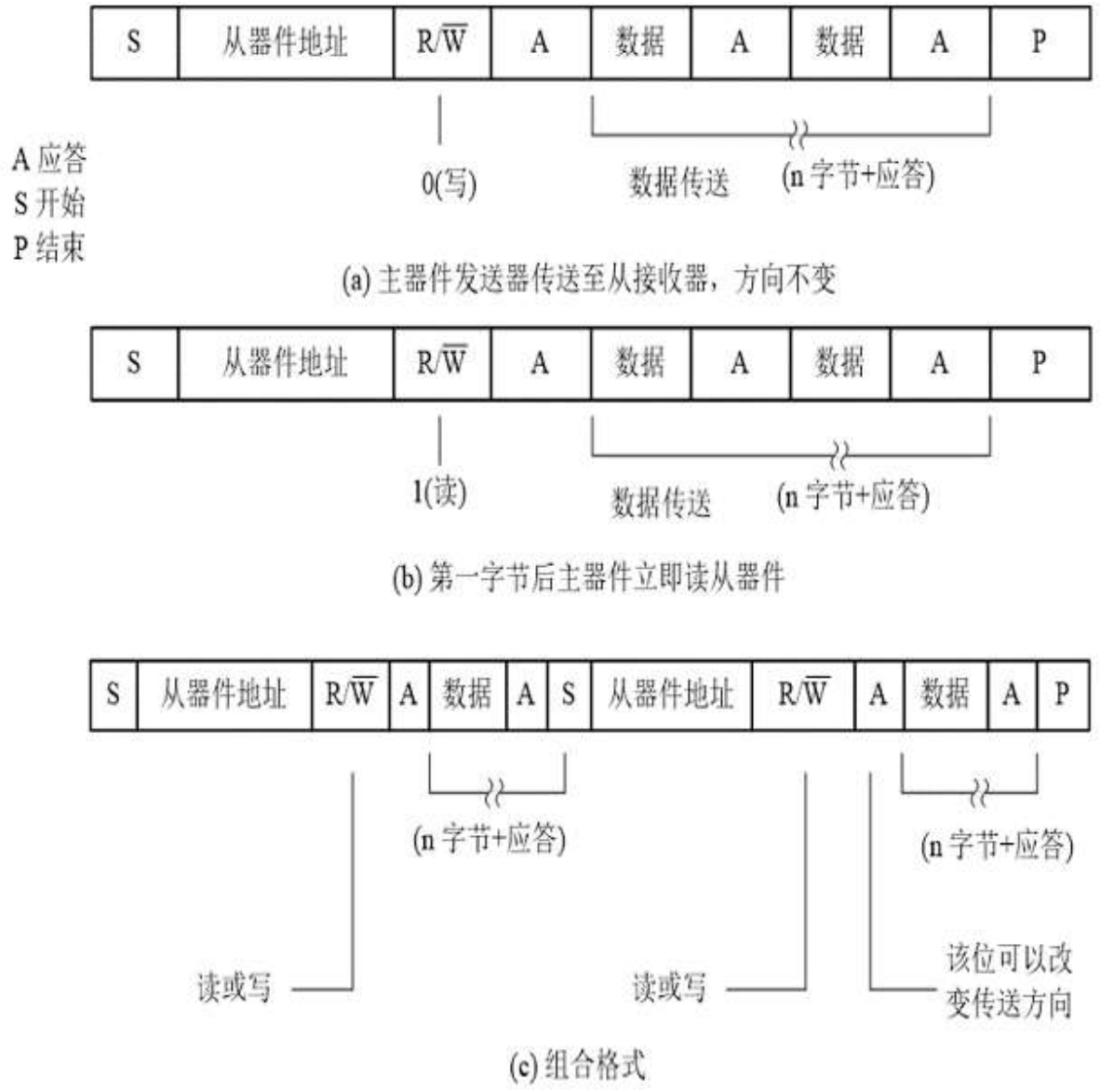

图12.5 I²C总线数据传输格式

注意：以下的三种情况，在传输完**8**位数据之后不会有**ACK**信号的产生。

1. 当从机不能响应从机地址时(例如它正忙于其他事而无法响应I²C总线的操作，或者找到的地址没有对应的从机)，在第9个SCL周期内SDA线没有被拉低，即没有ACK信号。

(2) 如果从机接收器在传输过程中不能接收更多的数据，它也不会发出ACK信号。

(3) 主机接收器在接收到最后一个字节后，也不会发出ACK信号。

5. I²C总线的寻址方式

1. 7位寻址

第一个字节的开始7位组成从机地址，最低位(LSB)是第8位，它决定了普通的和带重复开始条件的7位地址格式的方向。第一个字节的最低位是“0”时，表示主机写信息到被选中的从机；第一个字节的最低位是“1”时，表示主机从从机读信息。当发送了一个地址后，系统中的每个器件都在起始条件后将前7位与自己的地址比较，如果一样，器件会判定它被主机寻址。至于是从机接收器还是从机发送器，都由R/W位决定。

2) 10位寻址

10位从机地址由在起始条件或者重复起始条件后的前两个字节组成。第一个字节的前7位是1110xx的组合，其中最后两位xx是10位地址的两个最高位(MSB)。第一个字节的第8位是R/W位，决定了传输的方向，第一个字节的最低位是“0”时，表示主机写信息到被选中的从机；第一个字节的最低位是“1”时，表示主机从从机读信息。如果R/W位是“0”，则第二个字节是10位从机地址剩下的8位；如果R/W位是“1”，则下一个字节是从机发送给主机的数据。

12.2 Exynos 4412内部的I²C控制器

12.2.1 I²C简介

Exynos 4412支持多主机的I²C串行总线接口，支持主机发送、主机接收、从机发送和从机接收四种模式。其特性如下：

- 9通道I²C接口，其中8通道为通用的，1通道专用于高清多媒体接口(HDMI)。
- 7位寻址模式。
- 支持串行8位双向传输。
- 支持高达100 Kb/s的标准传输模式和400 Kb/s的快速传输模式。
- 支持中断和查询事件。

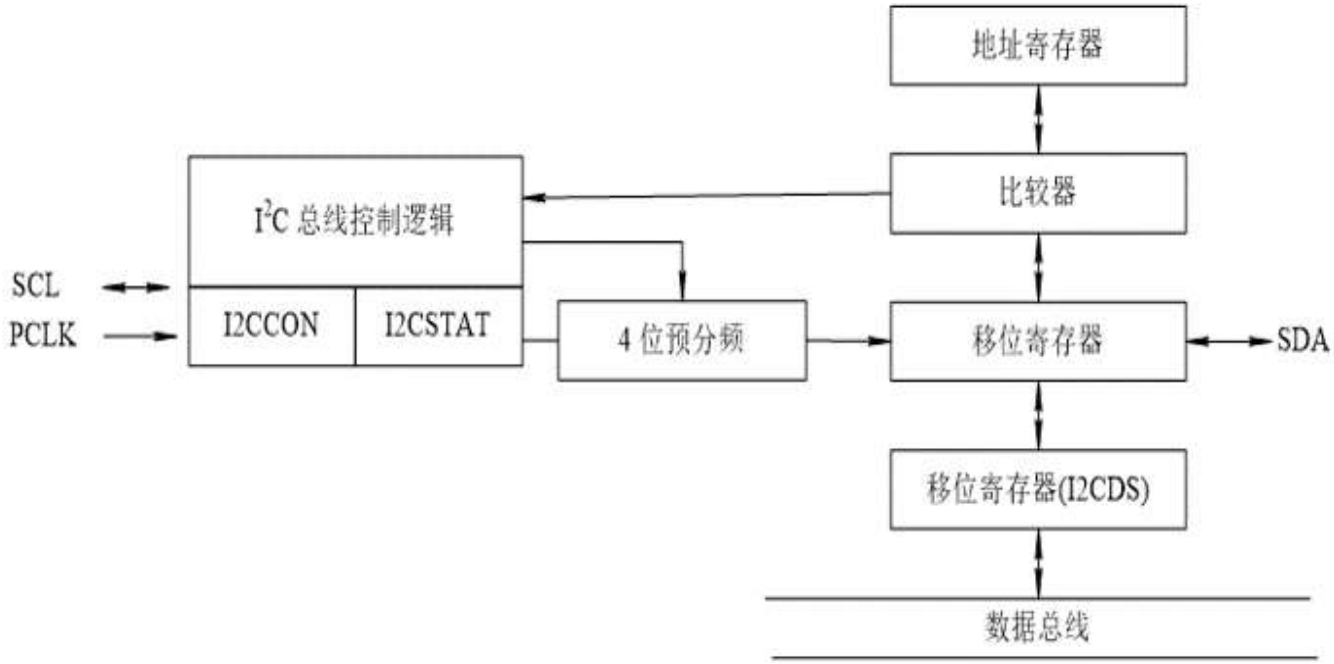

图12.6 I²C串行总线接口内部结构

Exynos 4412提供了4个寄存器进行配置和控制I²C操作。SDA线上的数据从I2CDS寄存器发出，或者传入I2CDS寄存器。I2CADD寄存器中保存Exynos 4412作为从机时的地址。I2CCON、I2CSTAT两个寄存器用来控制和标识各种状态，比如选择工作模式，发送S信号、P信号，决定是否发出ACK信号，检测是否收到ACK信号等。

12.2.2 I²C相关寄存器

Exynos 4412 I²C总线控制寄存器主要有控制寄存器I2CCONn、状态寄存器I2CSTATn、地址寄存器I2CADDn、收/发数据移位寄存器I2CDSn和多主机线控寄存器I2CLCn。每类寄存器针对8个通道各有一个与之对应，即上述寄存器中的n，其取值为0～7。

1. 控制寄存器I2CCONn(n = 0～7)

该类寄存器用于配置8个I²C通道的时钟和使能等功能，如表12.2所示。

名称	位域	类型	功 能 描 述	复位值
应答产生	[7]	RW	应答使能位。0=禁止；1=使能	0
Tx 时钟源选择	[6]	RW	传输时钟预分频选择位。 0=I2CCLK=f_{PCLK}/16；1=I2CCLK=f_{PCLK}/512	0
Tx/Rx 中断	[5]	RW	Tx/Rx 中断使能位。0=禁止；1=使能	0
中断挂起标志位	[4]	S	读：0=未产生中断；1=产生中断 写：0=无效；1=恢复操作	0x01
发送时钟值	[3:0]	RW	Tx 时钟=I2CCLK/(I2CCON[3:0]+1)	—

表12.2 控制寄存器I2CCONn(n = 0～7)

2. 状态寄存器I2CSTATn (n = 0～7)

该类寄存器用于标示I²C的运行状态等，如表12.3所示。

名 称	位域	类型	功 能 描 述	复位值
主从收发模式选择	[7:6]	RWX	00=从接收模式；01=从发送模式；10=主接收模式；11=主发送模式	00
忙信号状态/开始停止条件	[5]	S	读：0=准备；1=忙　写：0=产生停止信号；1=产生开始信号	0
串行输出使能	[4]	S	0=禁止 Rx/Tx；1=使能 Rx/Tx	0
仲裁状态标识位	[3]	RO	0=总线仲裁成功；1=总线仲裁失败	0
从地址状态标识位	[2]	RO	0=当侦测到开启/停止条件时清除；1=接收与 I2CADD 匹配的从地址	0
地址 0 状态标志	[1]	RO	0=当侦测到开启/停止条件时清除；1=接收到从地址值为 00000000b	0
最后接收位状态标志	[0]	RO	0=最后接收位被清零(接收 ACK)；1=最后接收位置 1(不接收 ACK)	0

表12.3 状态寄存器I2CSTATn (n = 0～7)

3. 从机地址寄存器I2CADDn (n = 0～7)

该类寄存器用于配置I²C的7位从机地址，如表12.4所示。

名称	位域	类型	功 能 描 述	复位值
从机地址	[7:0]	RWX	7 位从机地址[7:1]，位[0]不可用。该寄存器随时可读。当 I2CSTAT 串行输出使能时可写	–

表12.4 从机地址寄存器I2CADDn (n = 0～7)

4. 数据收/发移位寄存器I2CDSn (n = 0～7)

该类寄存器用于存储I²C收/发的数据，如表12.5所示。

名称	位域	类型	功 能 描 述	复位值
数据移位	[7:0]	RWX	8 位数据移位寄存器。如果串行输出使能，该寄存器可写；任何时候可读	–

表12.5 数据收/发移位寄存器I2CDSn (n = 0～7)

12.3 I²C应用实例

本节通过Exynos 4412的I²C对E²PROM芯片AT24C02的操作，介绍I²C的编程方法。AT24C02A/04A/08A/16A是存储容量为2K/4K/8K/16K位(bit)的E²PROM，支持电擦除、电烧写，支持I²C总线接口协议等。

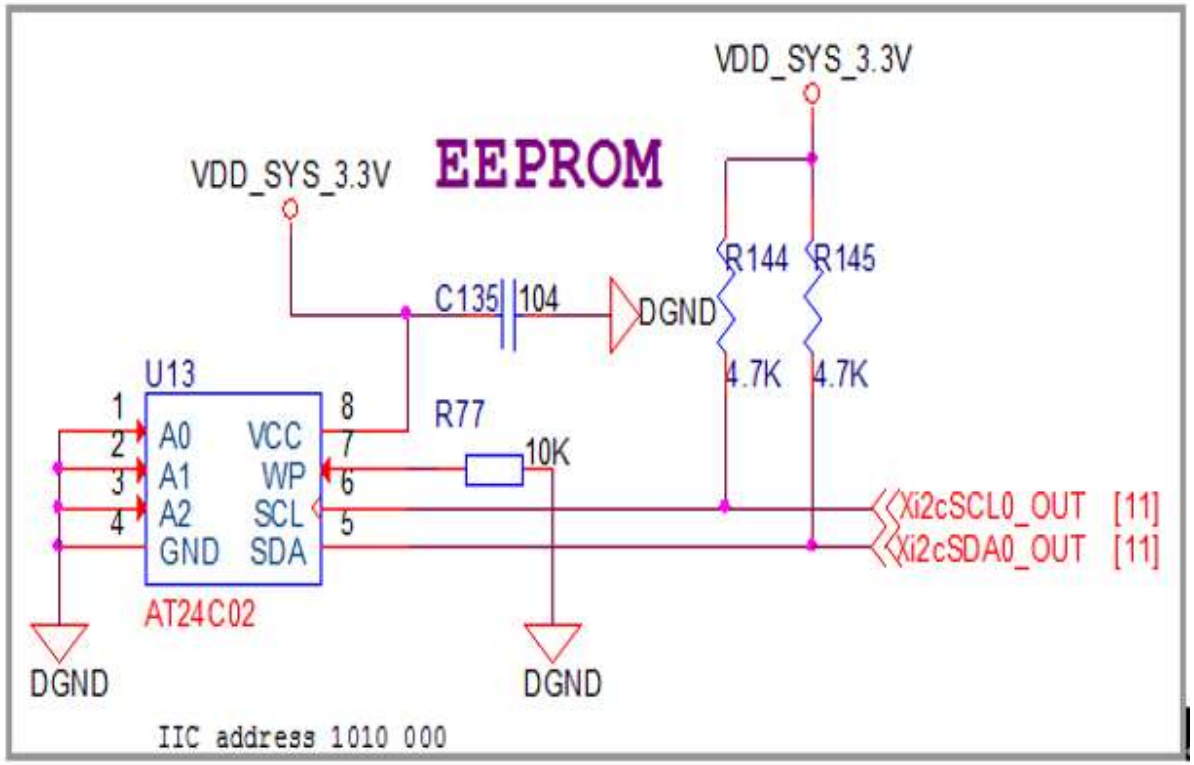

图12.7 Exynos 4412的I²C与E²PROM芯片AT24C02电路接线图

12.3.1 基本操作

1. 单字节写操作

字节写时序依次要发送器件的地址(包括LSB用于读/写标识，0标识为写，1标识为读)、器件片内数据写入地址和写入的8位数据。AT24C02接收到上面的每一次地址或数据，都会返回一个应答信号ACK。当AT24C02接收完成最后一个数据并返回ACK应答信号时，I²C主机必须产生停止位来结束写时序。

接下来的这个函数的功能是写一个字节到24C02。

```
void iic_write_24c02(uchar SlaveAddr, uchar Addr, uchar Data)
    {
        I2C0.I2CDS0 = SlaveAddr; //装载要写入的目的地地址

        //使能ACK；时钟源PCLK/512；中断使能；清中断挂起标志；
Tx时钟              I2CCLK/(1 + 1)
        I2C0.I2CCON0 = 0xe1;
        I2C0.I2CSTAT0 |= 0xf0; //主发送模式；开始传送；输出使能
        while(!(I2C0.I2CCON0 & (1<<4))); //等待发送结束
        I2C0.I2CCON0 &= ~(1<<4)); //清除中断挂起标志，恢复传送
        I2C0.I2CDS0 = Addr; //装载从机总线地址
        while(!(I2C0.I2CCON0 & (1<<4))); //等待发送结束
        I2C0.I2CCON0 &= ~(1<<4)); //清除中断挂起标志，恢复传送
        I2C0.I2CDS0 = Data; //装载数据
        while(!(I2C0.I2CCON0 & (1<<4))); //等待发送结束
        I2C0.I2CSTAT0 = &= ~(1<<5);     //停止信号产生，释放总
线
        I2C0.I2CCON0 & ~(1<<4); //清除中断挂起标志，恢复传送
    }
```

2. 多字节写操作

AT24C02具有页写功能(多字节写)，其存储大小为2 KB，支持8字节的连续写入。芯片根据型号不同，其他型号支持不同大小的页写入。页写入时序和字节写入时序大致一样，只是在写入第一个数据并接收到E²PROM的ACK应答时，主机不再发送停止位，而是继续写入数据。

3. 当前地址读操作

由于在E²PROM中维护了一个最新的器件片内数据写入地址，当要读取E²PROM当前地址的内部数据时，只需要传送器件地址(包括LSB用于读/写，此时为1，标示读方向)，E²PROM会将当前地址的内部数据发送到总线上。此后，主机不需要回复应答信号，但要发送一个停止位。

4. 随机地址读操作

随机地址读也就是指定地址读。一旦有读/写操作，这个内部地址将发生改变。也就是说，其维护的是最新的内部访问地址，而且这个自动更新操作是由内部数据地址计数器来完成的。内部数据地址自动更新的过程具有“回滚”特性。按照其操作时序，依次发送器件地址、要访问的片内数据地址。

接下来的这个函数的功能是从24C02读一个字节数据。

```
void iic_ read_24c02(uchar SlaveAddr, uchar Addr, uchar *Data)
    {    I2C0.I2CDS0 = SlaveAddr;
         I2C0.I2CCON0 = 0xe1;
         I2C0.I2CSTAT0 = 0xf0; //主发送模式；开始传送；输出使能
         while(!(I2C0.I2CCON0 & (1<<4))); //等待发送结束
         I2C0.I2CCON0 &= ~(1<<4)));    //清除中断挂起标志，恢复传送
         I2C0.I2CDS0 = Addr; //装载从机总线地址
         while(!(I2C0.I2CCON0 & (1<<4)));//等待发送结束
         I2C0.I2CCON0 &= ~(1<<4); //清除中断挂起标志，恢复传送
         I2C0.I2CDS0 = SlaveAddr|0x01;
         I2C0.I2CSTAT0 = 0xb0;
         I2C0.I2CCON0 &= ~((1<<7)|(1<<4));
         while(!(I2C0.I2CCON0 & (1<<4)));//等待发送结束
         *data = I2C0.I2CDS0;//读取一个字节数据
         I2C0.I2CSTAT0 = &= ~(1<<5);//停止信号产生，释放总线
         I2C0.I2CCON0 & ~(1<<4); //清除中断挂起标志，恢复传送
    }
```

12.3.2 编程实例

通过编写程序，将GPD1的GPD1[0]和GPD1[1]分别配置成I2C_0_SDA、I2C_0_SCL。采用字节写的方式，向E²PROM的地址0循环写入256 B数据，然后采用随机读的方式，循环读出0地址开始的256个刚写入的数据，最后将其以串口的形式在终端打印出来。

代码如下所示：

```
#define N 256
int main( )
{
  uchar src[N],dst[N];
  volatile int i,sum;
  for(i = 0;i<N;i++)
  {
    src[i] = i;
    dst[i] = 0;
  }
  //GPD1的GPD1[0]和GPD1[1]分别配置成I2C_0_SDA、
I2C_0_SCL
  GPD1.GPD1CON = (GPD1.GPD1CON &(~0xff)) | 0x22;

  uart0_init();
```

```
for(i = 0;i<N;i++)
  {
    iic_write_24c02(0xa0,i,src[i]);
    for(sum = 10000;sum!=0;sum--);      //延时等待写操作
  }
  for(sum = 100000;sum!=0;sum--); //延时等待
  for(i = 0;i<N;i++)
    iic_read_24c02(0xa0,i,&(dst[i]));
  printf("read from AT24C02: ");
  for(i = 0;i<N;i++)
   printf("  %d",dst[i]);
  while(1);
  return 0;
}
```

问题与思考：

1. 试述I²C总线的优、缺点。

2. 编程实现I²C总线操作E²PROM芯片AT24C02。