C51 单片机模拟 I2C 总线驱动程序设计

/********** I2C 总线驱动 **********

模块名：I2C 总线驱动　　型号：I2C

功能描述：

此模块包括发送数据及接收数据，应答位发送，并提供了几个直接面对器件的操作函数，能很方便的与用户程序进行连接并扩展。需要注意的是，函数是采用延时方法产生 SCL 脉冲，对高晶振频率要做一定的修改！！在写 E2PROM 的时候一定要延时！！！

说明：

1us 机器周期，晶振频率要小于 12MHz
返回 1 则操作成功，返回 0 则操作失败。
sla 为器件从地址，suba 为器件子地址。

****

```
#i nclude "AT89X52.h"
#i nclude <intrins.h>

#define SomeNOP(); {_nop_();_nop_();_nop_();_nop_();_nop_();} //定义空指令

sbit    SDA = P1^3;      //模拟 I2C 数据传输位
sbit    SCL = P1^2;      //模拟 I2C 时钟控制位

bit    bdata I2C_Ack;     //应答标志位
```

/********** I2C_Start **********

函数名：void I2C_Start()
入口：
出口：

功能描述：启动 I2C 总线，即发送 I2C 初始条件

解释：在 I2C 总线协议中规定的起始位格式是:在 SCL 高电平期间,SDA 发生从高到低的电平跳变.它与其它数据格式的区别在于,协议中规定有效的数据必须在 SCL 的高电平期间保持不变,只有在 SCL 的低电平期间才能发生跳变.所以这一有别与其它格式的数据才能做为起始位.

调用函数：
全局变量：

****

```
void I2C_Start()
{
SDA = 1;       //发送起始条件的数据信号
_Nop();
SCL = 1;
SomeNOP();    //起始条件建立时间大于 4.7us,延时
   SDA = 0;       //发送起始信号
SomeNOP(); //起始条件建立时间大于 4us,延时
SCL = 0;       //钳住 I2C 总线准备发送或接收数据

/******解释:I2C 总线在空闲状态下都是被上拉为高电平的,所以当它们处于低电平时就表示忙的状态.***/
_nop_();
_nop_();
}
```

/********** I2C_Stop **********

函数名：void I2C_Stop()
入口：
出口：

功能描述：结束 I2C 总线，即发送 I2C 结束条件

解释:同起始条件的格式类似,结束条件的格式是在 SCL 高电平期间,SDA 由低电平向高电平跳变.
调用函数：
全局变量：

****

```
void I2C_Stop()
{
SDA = 0;       //发送结束条件的数据信号
_Nop();
SCL = 1;       //发送结束条件的时钟信号
SomeNOP();//结束条件建立时间大于 4us,延时
```

```
SDA = 1;        //发送 I2C 总线结束信号
SomeNOP();

}

/********************************** I2C_CheckAck ****************************
```

函数名：bit I2C_CheckAck(void)
入口：
出口：0（无应答），1（有应答）

功能描述：
检验 I2C 总线应答信号，有应答则返回 1，否则返回 0，超时值取 255.

解释:I2C 总线协议中规定传输的每个字节之后必须跟一个应答位,所以从器件在接收到每个字节之后必须反馈一个应答信号给主控制器,而主控制器就需要检测从器件回传的应答信号,根据其信息做出相应的处理.另外,主从之别是相对的,接收数据的即为从,发送数据的及为主.

再看看应答信号的格式:在由发送器产生的时钟响应周期里,发送器先释放 SDA(置高),然后由接受器将 SDA 拉低,并在这个时钟脉冲周期的高电平期间保持稳定的低电平.即表示从器件做出了应答.

调用函数：void I2C_Stop()
全局变量：

```
*******************************************************************************

bit I2C_CheckAck(void)
{
uchar errtime = 255;      // 因故障接收方无 Ack,超时值为 255
SDA = 1;                   //发送器先释放 SDA
SomeNOP();
SCL = 1;
SomeNOP();          //时钟电平周期大于 4 us
   while(SDA)       //判断 SDA 是否被拉低
{
   errtime--;
   if(errtime==0)
   {
    I2C_Stop();
    return(0);
   }
}
SCL = 0;
```

```
_nop_();
return(1);
}

/************************************ I2C_SendB ********************************
```

函数名：void I2C_SendB(uchar c)
入口：uchar 型数据
出口：

功能描述：
字节数据传送函数，将数据 c 发送出去，可以是地址，也可以是数据，发完后等待应答，并对
此状态位进行操作

注意:在传送数据时,数据(SDA)的改变只能发生在SCL的低电平期间,在SCL的高电平期间保持不变

调用函数：bit I2C_CheckAck()
全局变量：I2C_Ack

```
*******************************************************************************

void I2C_SendB(uchar c)
{
uchar BitCnt;

for (BitCnt=0; BitCnt<8; BitCnt++)   //要传送的数据长度为 8 位
{
   if((c<<BitCnt)&0x80)      //判断发送位(从高位起发送)
   {
    SDA = 1;
   }
   else
   {
    SDA = 0;
   }

   _nop_();
   _nop_();
   SCL = 1;         //置时钟线为高通知被控器开始接收数据位
SomeNOP(); //保证时钟高电平周期大于 4us
   SCL = 0;
}
```

```
_nop_();
_nop_();

I2C_Ack = I2C_CheckAck();      //检验应答信号,作为发送方,所以要检测接收器反馈的应答信号.
_nop_();
_nop_();
}


/************************************ I2C_RcvB *********************************

函数名：uchar I2C_RcvB()
入口：
出口：uchar 型数据

功能描述：
接收从器件传来的数据，并判断总线错误（不发应答信号），收完后需要调用应答函数。

调用函数：
全局变量：

*******************************************************************************

uchar I2C_RcvB()
{
uchar retc;
uchar BitCnt;          //位

retc = 0;
SDA = 1;            //置数据总线为输入方式,作为接收方要释放 SDA.
for(BitCnt=0;BitCnt<8;BitCnt++)
{
   _nop_();
   SCL = 0;          //置时钟线为低准备接收数据位
SomeNOP();           //时钟低电平周期大于 4.7us
   SCL = 1;          //置时钟线为高使数据有效
   _nop_();
   _nop_();
   retc = retc<<1;
   if(SDA==1)
   {
    retc = retc + 1;       //读数据位，接收的数据放入 retc 中
   }
   _nop_();
```

```
    _nop_();
}

SCL = 0;
_nop_();
_nop_();

return(retc);
}
```

/********** I2C_Ackn **********

函数名：void I2C_Ackn(bit a)
入口：0 或 1
出口：

功能描述：主控制器进行应答信号（可以是应答或非应答信号）

说明:作为接收方的时候,必须根据当前自己的状态向发送器反馈应答信号
调用函数：
全局变量：

****

```
void I2C_Ackn(bit a)
{
if(a==0)      //在此发送应答或非应答信号
{
    SDA = 0;
}
else
{
    SDA = 1;
}
SomeNOP();
SCL = 1;
SomeNOP();      //时钟电平周期大于 4 us
SCL = 0;      //清时钟线钳住 I2C 总线以便继续接收
_nop_();
_nop_();
}
```

/********** I2C_ISendB **********

函数名：bit I2C_ISendB(uchar sla, uchar suba, uchar c)
入口：从器件地址 sla，子地址 suba，发送字节 c
出口：0（操作有误），1（操作成功）

功能描述：从启动总线到发送地址、数据，结束总线的全过程，
　　如果返回 1，表示操作成功，否则操作有误。

调用函数：I2C_Start(),I2C_SendB(uchar c),I2C_Stop()
全局变量：I2C_Ack

```
*******************************************************************************

bit I2C_ISendB(uchar sla, uchar suba, uchar c)
{
I2C_Start();      //启动总线

I2C_SendB(sla);      //发送器件地址
if(!I2C_Ack)
{
   return(0);
}

I2C_SendB(suba);     //发送器件子地址
if(!I2C_Ack)
{
   return(0);
}

I2C_SendB(c);      //发送数据
if(!I2C_Ack)
{
   return(0);
}

I2C_Stop();       //结束总线
return(1);
}

/********************************** I2C_IRcvB ***********************************

函数名：bit I2C_IRcvB(uchar sla, uchar suba, uchar *c)
入口：从器件地址 sla，子地址 suba，收到的数据在 c
出口：1（操作成功），0（操作有误）

功能描述：从启动总线到发送地址、读数据，结束总线的全过程。
```

调用函数： I2CS_tart(),
I2C_SendB(uchar c),
I2C_RcvB(),
I2C_Ackn(bit a),
I2C_Stop()

全局变量：I2C_Ack

创建者：陈曦　日期：2005-5-15
修改者：　　日期：

****

```
bit I2C_IRcvB(uchar sla, uchar suba, uchar *c)
{
I2C_Start();      //启动总线

I2C_SendB(sla);
if(!I2C_Ack)
{
   return(0);
}

I2C_SendB(suba);     //发送器件子地址
if(!I2C_Ack)
{
   return(0);
}

I2C_Start();      //重复起始条件

I2C_SendB(sla+1);     //发送读操作的地址
if(!I2C_Ack)
{
   return(0);
}

*c = I2C_RcvB();     //读取数据
I2C_Ackn(1);      //发送非应答位
I2C_Stop();       //结束总线
return(1);
}
```