51单片机

一：

编写程序：1.建立工程文件2.新建一个.c的文件3.添加新建的C文件到工程中。

点亮LED小灯：关键字：sbit

功能：位定义

一般格式： sbit 标识符 = 地址值;

例如：sbit LED1 = P1^0;

注意：地址值中P1的“P”必须为大写的P

二：

延时：1.定时器延时

1. 软件延时（while(a--)）

三：

宏定义： #define uchar unsigned char

Typedef 已有数据类型 代替数据类型；

自定义函数：延时函数delay(毫秒级)

void delay(unsigned int z) //延时程序

{ //参数 ：z 延时毫秒设定，取值范围0-65535

unsigned int x,y; //返回值 ：无

for(x=z;x>0;x--) //描述 ：12T/Fosc11.0592M毫秒级延时

for(y=114;y>0;y--);

}

另一种延时程序实例： void DELAY\_MS (unsigned int a)

{//延时程序

unsigned int i;

while( a-- != 0)

{

for(i = 0; i < 600; i++);

}

}

循环移位函数：标准库函数—intrins.h

内部函数：字符型循环左移：\_crol\_

字符型循环右移：\_cror\_(与左移右移相比，循环移位会补位)

#include <intrins.h>

void test\_crol (void)

{

unsigned char a;

unsigned char b;

a = 0xFE; //1111 1110

b = \_crol\_(a,1); // b now is 0xFD 二进制为1111 1101

}

四：

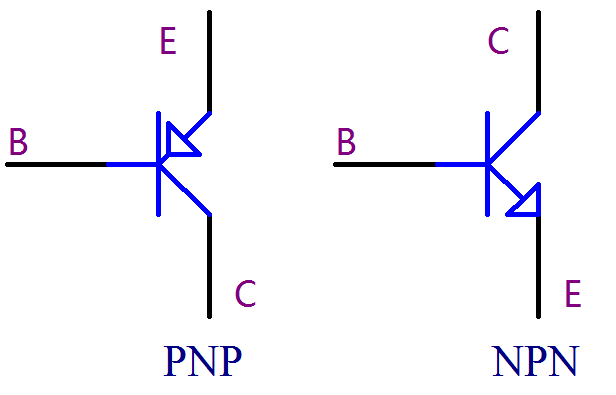
1.三极管做开关应用：

BASE(基极) EMITTER（发射极） COLLECTOR（集电极）

三极管8550是一种常用的普通三极管。 它是一种低电压，大电流，小信号的PNP型硅三极管。

主要用途：开关应用、射频放大





蜂鸣器程序： 1.sbit beep（名称）= P2^3；(引脚)

2.beep = ~beep;(改 变蜂鸣器状态)

五：数码管静态显示



1.

数码管根据内部连接不同分为共阴极和共阳极。八段数码管内部有8颗LED组成，如果想要显示特定字形只需控制相应的LED亮起，其他LED熄灭即可。（当前使用为共阴极）。故给要点亮的LED段1;

3.共阴极数码码表

0x77, //"A"

0x7C, //"B"

0x39, //"C"

0x5E, //"D"

0x79, //"E"

0x71, //"F"

0x76, //"H"

0x38, //"L"

0x40, //"-"

0x00, //熄灭

0x3F, //"0"

0x06, //"1"

0x5B, //"2"

0x4F, //"3"

0x66, //"4"

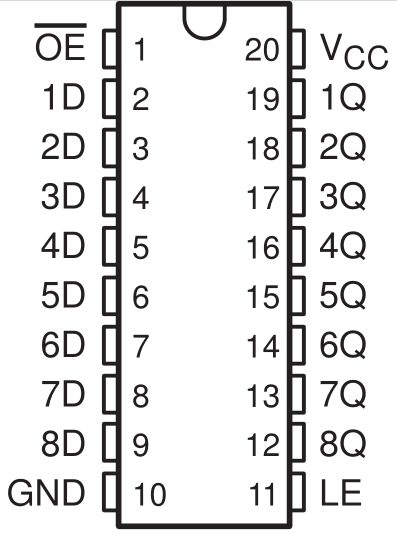
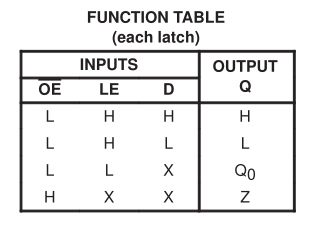
0x6D, //"5"

0x7D, //"6"

0x07, //"7"

0x7F, //"8"

0x6F, //"9"

4.锁存器：

输出口Q要想输出高低电平OE脚必须接GND。

LE脚为高时，输出端Q随输入端D的数据而变化。

LE脚为低时，输出端Q数据保持不变，输入端D数据变化不会改变Q的数据。（1为打开，0为关闭）

位选与段选即为两个LE。LE=1时，左右相同，锁存器等于开启。

先位选（要亮的位给0），再段选（要亮的段给1）。

1. 动态显示：动态显示的特点是将所有位数码管的段选线并联在一起，由位选线控制是哪一位数码管有效。选亮数码管采用动态扫描显示。所谓动态扫描显示即轮流向各位数码管送出字形码和相应的位选，利用发光管的余辉和人眼视觉暂留作用，使人的感觉好像各位数码管同时都在显示。

——数组的一般格式定义如下：**数据类型 数组名[常量表达式] = {元素表}；**

例如：

unsigned char tabel[3] ={0x3F, 0x06, 0x5B};

P0 = tabel[0]; //P0此时的值为0x3F

注意清除断码，防止出现数码管错乱。

1. 四位数码管显示程序

#include <reg52.h>

#include <intrins.h>

sbit WE = P2^7;//定义数码管位选

sbit DU = P2^6;//定义数码管段选

unsigned int code tabel[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

//共阴数码管段选表

unsigned int code tabel[22]= {

//0 1 2 3 4 5 6 7 8

0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,

//9 A B C D E F H L

0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x76, 0x38,

//n u - 熄灭

0x37, 0x3E, 0x40, 0x00 };

void delay(unsigned int z)

{

unsigned int x,y;

for(x=z;x>0;x--)

for(y=114;y>0;y--);

} // z的赋值为延时的秒数（毫秒）

void displayl(unsigned int i)

{

unsigned int ge,shi,bai,qian;

qian = i/1000;

bai = (i%1000)/100;

shi = i%100/10;

ge = i%10; //给各位数赋值。

//第一位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XFE; // 0XC3 1111 1110 选择1

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[qian];

DU = 0;//锁存段选数据

delay(2) ;

//第二位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XFD; // 0XC3 1111 1101 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[bai];

DU = 0;//锁存段选数据

delay(2);

//第三位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XFB; // 0XC3 1111 1011 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[shi];

DU = 0;//锁存段选数据

delay(2);

//第四位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XF7; // 0XC3 1111 0111 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[ge];

DU = 0;//锁存段选数据

delay(2);

}

void displayr(unsigned int i)

{

unsigned int ge,shi,bai,qian;

qian = i/1000;

bai = (i%1000)/100;

shi = i%100/10;

ge = i%10; //给各位数赋值。

//第五位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XEF; // 0XC3 1110 1111 选择1

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[qian];

DU = 0;//锁存段选数据

delay(2) ;

//第六位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XDF; // 0XC3 1101 1111 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[bai];

DU = 0;//锁存段选数据

delay(2);

//第七位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0XBF; // 0XC3 1011 1111 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[shi];

DU = 0;//锁存段选数据

delay(2);

//第八位数码管显示

P0 = 0xFF;//清除断码

WE = 1;//打开位选锁存器

P0 = 0X7F; // 0XC3 0111 1111 0为亮

WE = 0;//锁存位选数据

DU = 1;//打开段选锁存器

P0 = tabel[ge];

DU = 0;//锁存段选数据

delay(2);

}

void main()

{

while(1)

{

displayl(8484);displayr(8484);

}

}

六：键盘

1. 键盘类型

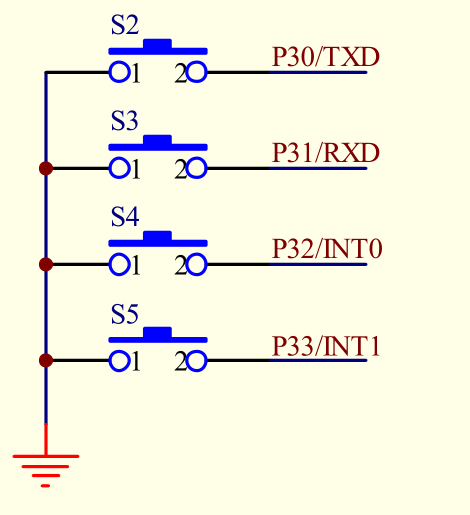
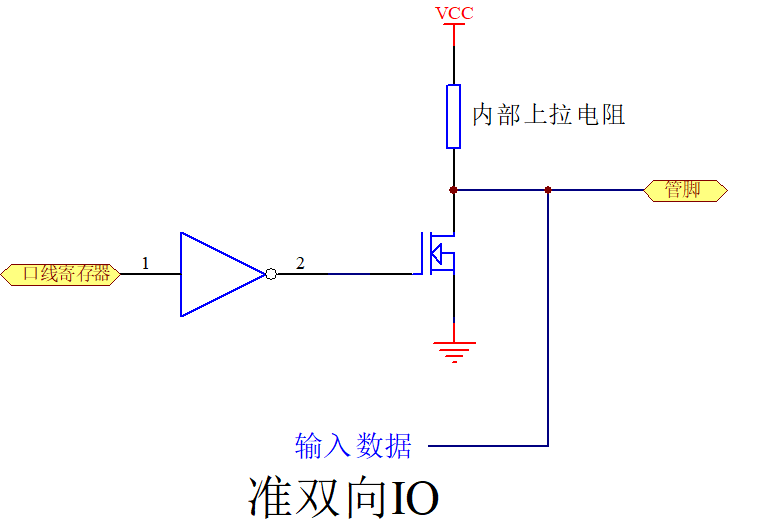
通常键盘有编码键盘和非编码键盘两种。编码键盘通过硬件电路产生被按按键的键值码，这种键盘使用方便，所需程序简单，但硬件电路复杂，

如计算机的键盘，单片机则通常不采用编码键盘。

而软件编程来识别的称为非编码键盘，非编码键盘硬件电路简单。

在单片机组成的各种系统中，最常用的是非编码键盘。

1. 独立键盘与矩阵键盘



按键一端与IO口连接，另外一端接地。通过控制IO口输出高电平即可检测

按键是否按下，当按下时IO口会被拉低，松开后IO口回到高电平。

注意按键消抖与松手检测；

if(key\_s2 == 0)

{

delay(20);//按键消抖

if(key\_s2 == 0)

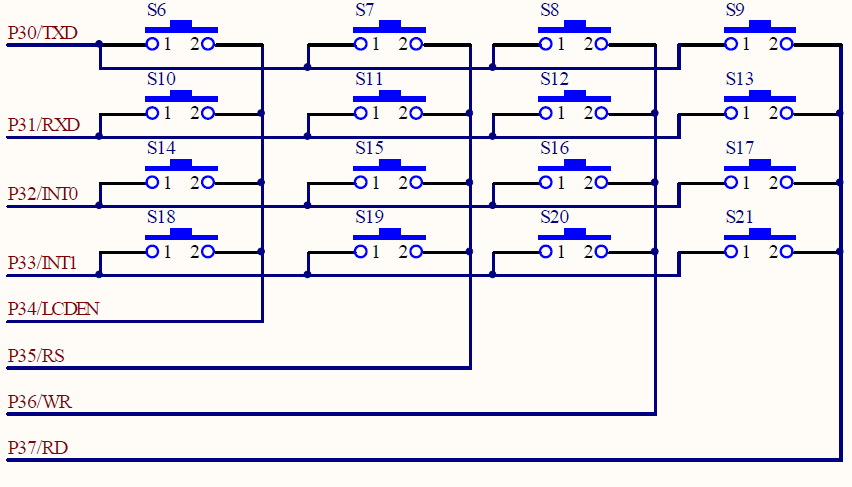
{

if(number != 9) //注意while 与 if 的使用

number++;

while (!key\_s2);//松手检测

}

 }

矩阵键盘识别相对于独立键盘要复杂一些。右图矩阵键盘一共有4行和4列一共16个按键组成。

确定矩阵键盘上哪一个按键被按下可以采用列扫描和行扫描。

列扫描时先把接在列上面的所有IO口拉高，接在行上的所有IO置低。当其中有一列内任何一个按键按下那么整条列线都会被拉低。

行扫描时相反。

void keyscan()

{

P3 = 0XF0;// 1111 0000 准备列扫描

if(P3 != 0XF0)//判断按键按下

{

delay(10);//按键消抖

if(P3 != 0XF0)

{

switch(P3)

{

case 0XE0: KeyValue =0;break;//1110 0000

case 0XD0: KeyValue =1;break;//1101 0000

case 0XB0: KeyValue =2;break;//1011 0000

case 0X70: KeyValue =3;break;//0111 0000

}

P3 = 0X0F; //0000 1111 准备行扫描此时手仍在按键上

switch(P3)

{

case 0X0E: KeyValue = KeyValue;break;//0000 1110

case 0X0D: KeyValue = KeyValue+4;break;//0000 1101

case 0X0B: KeyValue =KeyValue+8;break;//0000 1011

case 0X07: KeyValue =KeyValue+12;break;//0000 0111

}

while(!P3) ;//松手检测

}

}

P3 = 0XFF;//准备独立键盘扫描

if(P3 != 0XFF)//判断按键按下

{

delay(10);//按键消抖

if(P3 != 0XFF)

{

switch(P3)

{

case 0xfe: KeyValue = 16; break;//S2被按下

case 0xfd: KeyValue = 17; break;//S3被按下

case 0xfb: KeyValue = 18; break;//S4被按下

case 0xf7: KeyValue = 19; break;//S5被按?

}

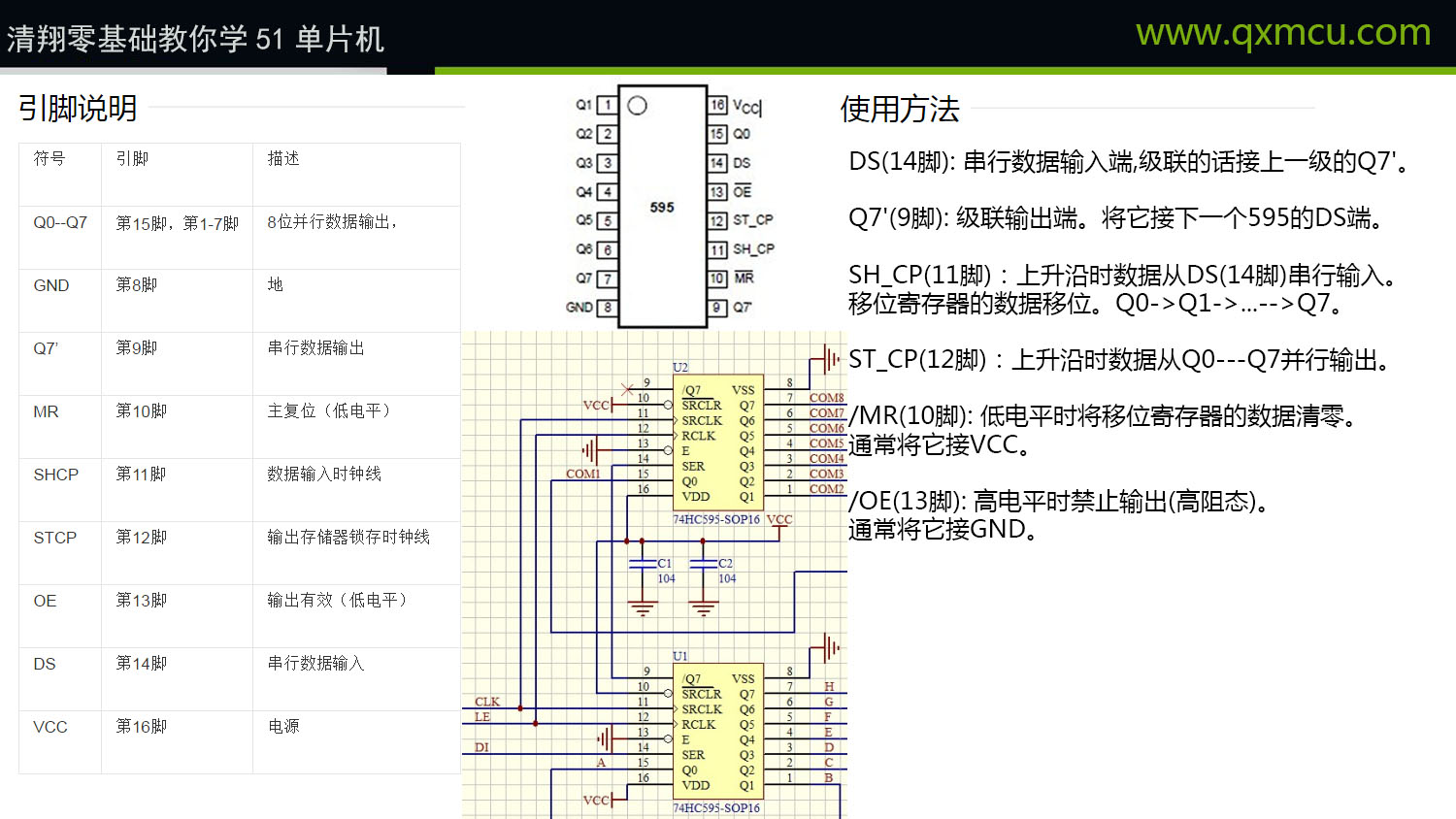
while(!P3) ;//松手检测

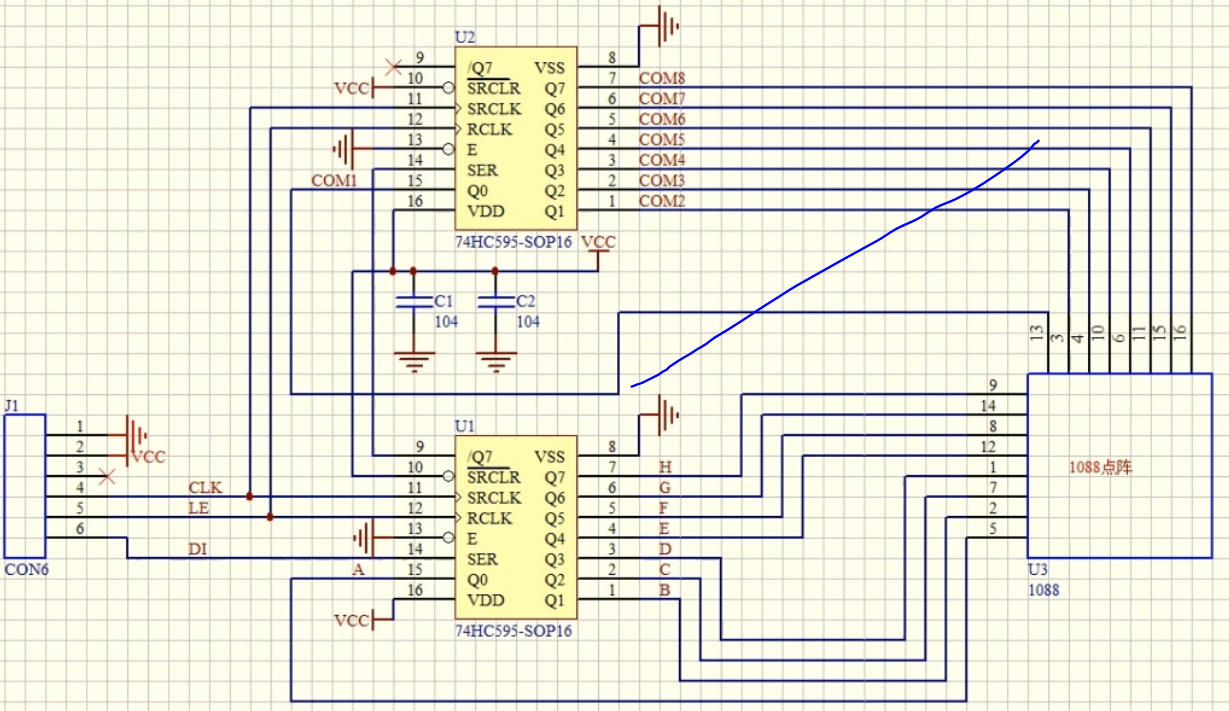
}

}

}

1. 点阵

1.原理

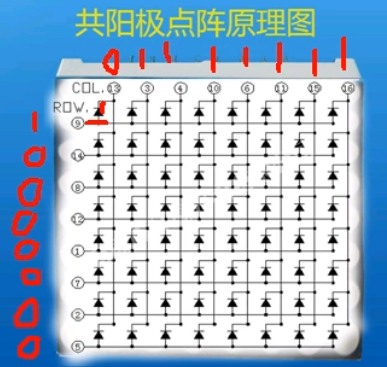
 void main()

{

//ROW 行 1000 0000 0X80 COL 列1111 1110 0XFE

unsigned char i,dat;

dat = 0XFE;//发送列值（横着的）



for(i=0;i<8;i++)//循环8次

{

S\_CLK = 0;//拉低移位储存器时钟

R\_CLK = 0;//拉低输出锁存器时钟

if(dat&0x01)// 最低位为1则为真

DIO = 1; //输出1

else

DIO = 0;//输出0

S\_CLK = 1;//产生上升沿，移位

dat >>= 1;//右移一位

}

dat = 0x80;//发送行值

for(i=0;i<8;i++)

{

S\_CLK = 0;//拉低移位储存器时钟

R\_CLK = 0;//拉低输出锁存器时钟

if(dat&0x01)// 最低位为1则为真

DIO = 1; //输出1

else

DIO = 0;//输出0

S\_CLK = 1;//产生上升沿，移位

dat >>= 1;//右移一位

}

R\_CLK = 1;//产生上升沿，输出

}

1. 中断
2. I/O口引脚

P0三态IO口，P1口内部有上拉普通准双向IO口，P2口内部有上拉普通准双向

IO口，P3口内部有上拉普通准双向IO口（有第二功能，可以通过配置特殊功能寄存

来实现）。

P3第二功能各引脚功能定义：

P3.0：RXD串行口输入

P3.1：TXD串行口输出

P3.2：INT0外部中断0输入

P3.3：INT1外部中断1输入

P3.4：T0定时器0外部输入

P3.5：T1定时器1外部输入

P3.6：WR外部写控制

P3.7：RD外部读控制

2.数据的输入/输出传送方式

无条件传送方式：

一方对另一方来说总是准备好的。

查询传送方式(LOOK UP)：

传送前一方先查询另一方的状态，若已经准备好就传送，否则就继续查询/等待。

中断传送方式(IRQ)：

一方通过申请中断的方式与另一方进行数据传送。

直接存储器存取方式(DMA)：

双方直接通过总线传送数据, 不经CPU中转。适用于数据量大高速通讯的设备不占用CPU时间。

3.51子系列允许5个中断源：

外部中断源（2个）：

INT0——由P3.2端口线引入，低电平或下降沿引起。

INT1——由P3.3端口线引入，低电平或下降沿引起。

这两个外部中断源标志和它们的触发方式控制位由特殊功能寄存器TCON的低4位控制。

内部中断源（3个）：

T0——定时器/计数器0中断，由T0回零溢出引起。

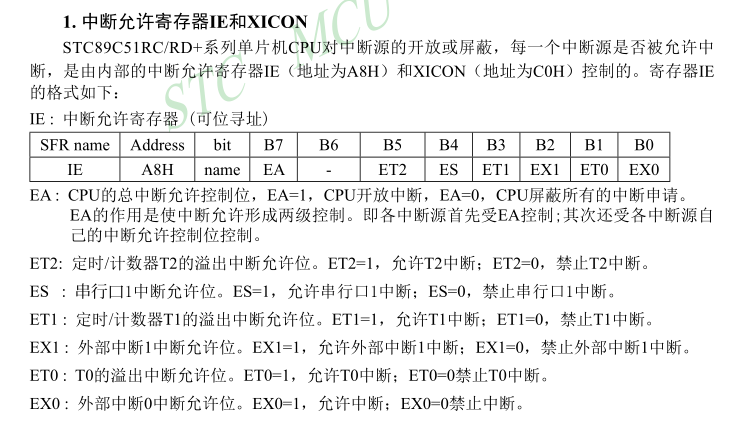
T1——定时器/计数器1中断，由T1回零溢出引起。

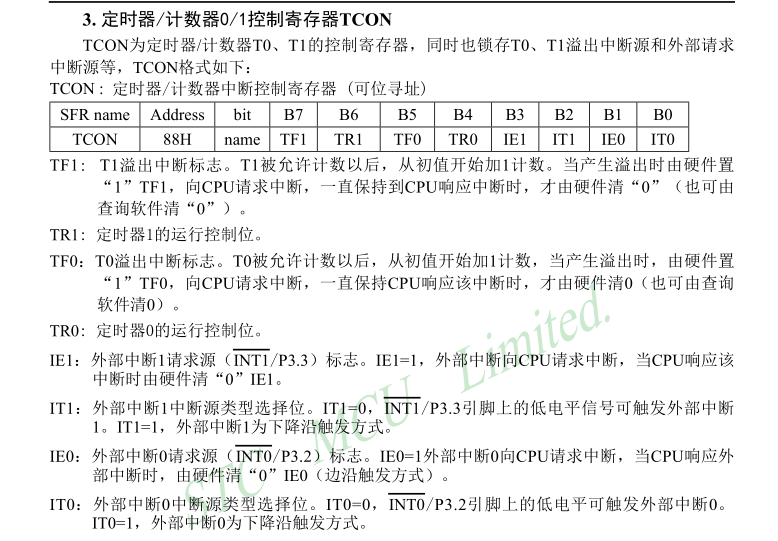
TI/RI——串行I/O中断，串行端口完成一帧字符发送/接收后引起。

这3个内部中断源的控制位分别锁存在特殊功能寄存器TCON和SCON中。

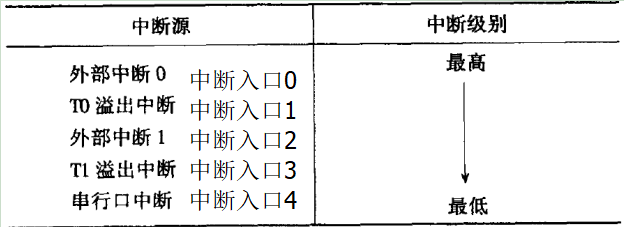
4.如何去使用单片机中断系统

1、允许中断 2、配置中断方式 3、编写中断处理函数





5.单片机同优先级中内部查询顺序



**void int1() interrupt 2 //中断处理函数，加关键字interrupt和入口号**

**{**

**中断处理语句**

**}**

**中断响应条件**

此中断源的中断允许位为1。

总中断CPU中断打开（EA = 1）。

中断源有中断请求。

**外部中断1中断处理函数**

void int1() interrupt 2

{

中断处理程序

}

**外部中断1初始化例：**

EA = 1; //开总中断

EX1 = 1; //开外部中断1

IT1 = 1; //跳变沿触发

#include<reg52.h>

sbit key\_S2 = P3^0;//独立键盘S2

sbit flag = P3^7;//外部中断产生脚 杜邦线连P33与P36

void delay(unsigned int z)//延时程序

{

unsigned int x,y;

for(x=z;x>0;x--)

for(y=114;y>0;y--);

}

void int1Init() //外部中断1初始化

{

EA = 1; //开总中断

EX1 = 1; //开外部中断1

IT1 =1 ; //外部中断1下降沿触发 0为低电平触发

}

void main()

{

while(1)

{

int1Init();//外部中断1初始化

if( key\_S2 == 0)

{

delay(20);//按键消抖

if( key\_S2 == 0)

{

flag = 1;

flag = 0;//产生下降沿

while(!key\_S2);//松手检测

}

}

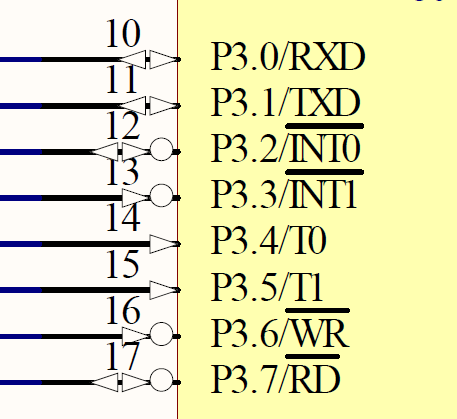
}

}

void int1() interrupt 2 //中断服务程序 不用在在前面声明

{

P1 = ~P1;

}

1. 定时/计数器

1.51单片机有2个16位定时器/计数器：

定时器0（T0为P3.4）和定时器1（T1为P3.5）

这里所说的16位是指定时/计数器内部分别有16位的计数寄存器。

**当工作在定时模式时，每经过一个机器周期内部的16位计数寄存器的值就**

**会加1，当这个寄存器装满时溢出。**

我们可以算出工作在定时模式时最高单次定时时间为

65535\*1.085us=时间（单位us）

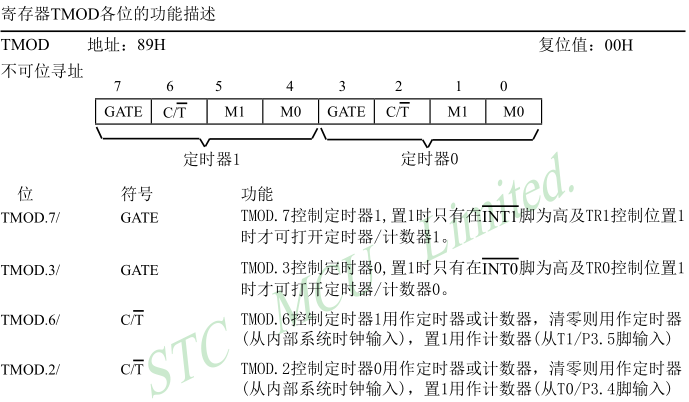
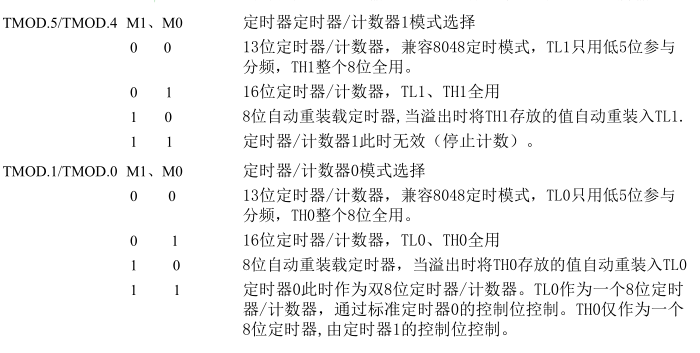
当工作在计数器模式时，T0(P3.4引脚),T1（P3.5引脚）每来一个脉冲计数寄存器加1

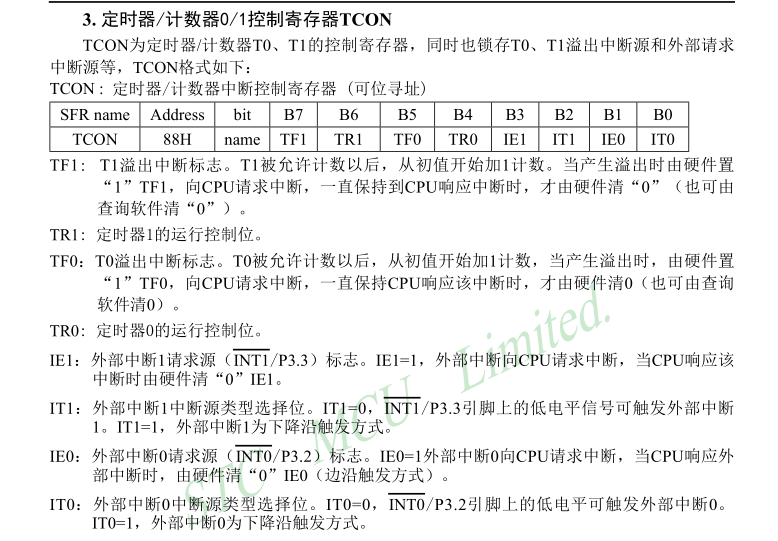
1. 使用定时计数器步骤

启动定时/计数器（通过TCON控制器）

设置定时/计数器工作模式（通过TMOD控制器）

查询定时/计数器是否溢出（读TCON内TF位）





程序：

//定时器0初始化

void timer0Init()

{

TR0 = 1; //启动定时器0

TMOD = 0X01; //定时器工作模式1，16位定时器模式 0000 0001

TH0 = 0x4b;// （65535-50000/1.085）/256

TL0 = 0xfd; // （65535-50000/1.085）%256 定时50mS EDFF 5ms

}

void main()//main函数自身会循环

{

unsigned int mSec, Sec;//毫秒和秒储存变量

timer0Init();//定时器0初始化

while(1)

{

if(TF0 == 1)//判断是否溢出

{

TF0 = 0;//软件清零溢出标志位

TH0 = 0x4b;

TL0 = 0xfd; //定时50ms

mSec++;//50ms到

if(mSec == 20)

{

mSec = 0;

Sec++;//1秒时间到

}

}

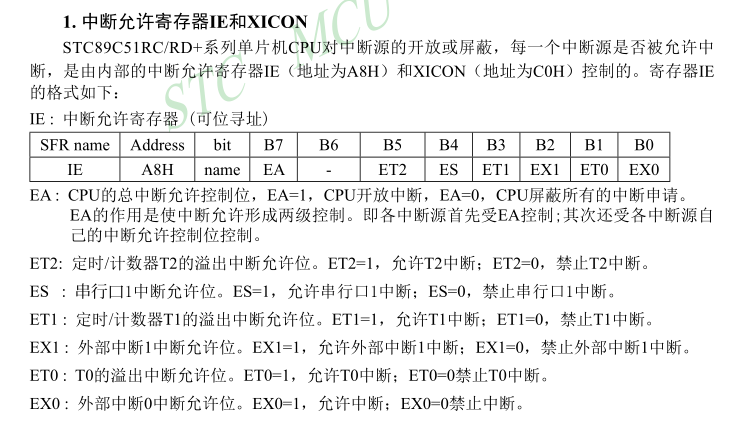
display(Sec); //数码管显示函数

if(Sec > 1000)

Sec = 0;//秒清零

}

}



**运用定时器中断进行的一种动态扫描：**

//共阴数码管段选表0-9

uchar code SMGduan[]= {

//0 1 2 3 4 5 6 7 8

0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,

//9 A B C D E F H L

0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x76, 0x38,

//n u - 熄灭

0x37, 0x3E, 0x40, 0x00 };

//数码管位选码

uchar code SMGwei[] = { 0xfe, 0xfd, 0xfb,0xf7};

void delay(unsigned int z) //延时程序

{ //参数 ：z 延时毫秒设定，取值范围0-65535

unsigned int x,y; //返回值 ：无

for(x=z;x>0;x--) //描述 ：12T/Fosc11.0592M毫秒级延时

for(y=114;y>0;y--);

}

void displayl(uchar i)

{

static uchar wei;

P0 = 0XFF;//清除断码

WE = 1;//打开位选锁存器

P0 = SMGwei[wei];

WE = 0;//锁存位选数据

switch(wei)

{

case 0: DU = 1; P0 = SMGduan[i / 1000]; DU = 0; break;

case 1: DU = 1; P0 = SMGduan[(i%1000)/100]; DU = 0; break;

case 2: DU = 1; P0 = SMGduan[i%100/10]; DU = 0; break;

case 3: DU = 1; P0 = SMGduan[i%10]; DU = 0; break;

}

wei++;

if(wei == 4)

wei = 0;

}

//定时器0初始化

void timer0Init()

{

EA = 1; //打开总中断

ET0 = 1;//打开定时器0中断

TR0 = 1; //启动定时器0

TMOD = 0X01; //定时器工作模式1，16位定时模式

TH0 = 0xED;

TL0 = 0xFF; //定时5ms

}

**-----直流小风扇程序**

#include <reg52.h>

unsigned char pwm\_motor\_val = 130;//左右电机占空比值 取值范围0-170，0最快

unsigned char pwm\_t;//周期

sbit \_4IN = P1^0;//

sbit key\_s2 = P3^0;

sbit key\_s3 = P3^1;

void delay(unsigned int z)//毫秒级延时

{

unsigned int x,y;

for(x = z; x > 0; x--)

for(y = 114; y > 0 ; y--);

}

//定时器0中断

void timer0() interrupt 1

{

pwm\_t++;

if(pwm\_t == 255)

\_4IN = 0;

if(pwm\_motor\_val == pwm\_t)

\_4IN = 1;

}

void main()

{

TMOD |= 0x02;//8位自动重装模块

TH0 = 220;

TL0 = 220;//11.0592M晶振下占空比最大比值是256,输出100HZ

TR0 = 1;//启动定时器0

ET0 = 1;//允许定时器0中断

EA = 1;//总中断允许

while(1)

{

if(key\_s2 == 0) //S2按下LED变暗

{

delay(5);

if(key\_s2 == 0)

{

if(pwm\_motor\_val < 170)

{

pwm\_motor\_val++;

}

}

}

if(key\_s3 == 0) //S3按键LED变亮

{

delay(5);

if(key\_s3 == 0)

{

if(pwm\_motor\_val > 0)

{

pwm\_motor\_val--;

}

}

}

}

}

应用PWM至SG90舵机

#include <reg51.h>

unsigned char count; //0.5ms次数标识

sbit pwm =P1^0 ; //PWM信号输出

sbit jia =P3^0; //角度增加按键检测IO口

sbit jan =P3^1; //角度减少按键检测IO口

unsigned char jd; //角度标识

void delay(unsigned int z) //延时程序

{ //参数 ：z 延时毫秒设定，取值范围0-65535

unsigned int x,y; //返回值 ：无

for(x=z;x>0;x--) //描述 ：12T/Fosc11.0592M毫秒级延时

for(y=114;y>0;y--);

}

void Time0\_Init() //定时器初始化

{

EA=1;

ET0=1;

TMOD = 0x01; //定时器工作模式1，16位定时器模式 0000 0001

TH0 = 0xFE;

TL0 = 0x33; //11.0592MZ晶振，0.5ms

TR0=1; //定时器启动

}

void Time0\_Int() interrupt 1 //中断程序

{

TH0 = 0xFE; //重新赋值

TL0 = 0x33;

if(count<jd) //判断0.5ms次数是否小于角度标识

pwm=1; //确实小于，PWM输出高电平

else

pwm=0; //大于则输出低电平

count++; //0.5ms次数加1

count=count%40; //次数始终保持为40 即保持周期为20ms

} //高电平保持 0.5ms到 2.5ms 及jd标识变量范围为1-5

void keyscan() //按键扫描

{

if(jia==0) //角度增加按键是否按下

{

delay(10); //按下延时，消抖

if(jia==0) //确实按下

{

jd++; //角度标识加1

count=0; //按键按下 则20ms周期从新开始

if(jd==6)

jd=5; //已经是180度，则保持

while(jia==0); //等待按键放开

}

}

if(jan==0) //角度减小按键是否按下

{

delay(10);

if(jan==0)

{

jd--; //角度标识减1

count=0;

if(jd==0)

jd=1; //已经是0度，则保持

while(jan==0);

}

}

}

void main()

{

jd=1;

count=0;

Time0\_Init();

while(1)

{

keyscan(); //按键扫描

}

}

1. **串口通信**

通信有并行通信和串行通信两种方式。在多微机系统以及现代测控系统中信息的交换多采用串行通信方式。

并行通信通常是将数据字节的各位用多条数据线同时进行传送，并行通信控制简单、传输速度快；由于传输线较多，长距离传送时成本高且接收方的各位同时接收存在困难。

串行通信是指 使用一条数据线，将数据一位一位地依次传输，每一位数据占据一个固定的时间长度。其只需要少数几条线就可以在系统间交换信息，特别适用于计算机与计算机、计算机与外设之间的远距离通信；传输线少，长距离传送时成本低，但数据的传送控制比并行通信复杂。

串行通信又可分为异步通信和同步通信。

异步通信是指通信的发送与接收设备使用各自的时钟控制数据的发送和接收过程。为使双方的收发协调，要求发送和接收设备的时钟尽可能一致。

例如：同步就是你叫我去吃饭，我听到了就和你去吃饭，如果没有听到，你就不停的叫，直到我告诉你听到了，才一起去吃饭。

异步就是你叫我，然后自己去吃饭，我得到消息后可能立即走，也可能等到下班后才去吃饭。

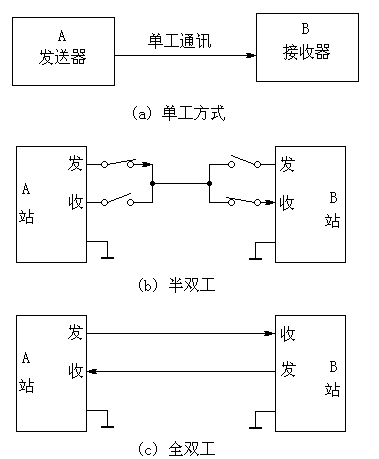
——异步通信是以字符（构成的帧）为单位进行传输，字符与字符之间的间隙（时间间隔）是任意的，但每个字符中的各位是以固定的时间传送的，即字符之间不一定有“位间隔”的整数倍的关系，但同一字符内的各位之间的距离均为“位间隔”的整数倍。



异步通信是一种很常用的通信方式（效率较低）异步通信在发送字符时，发送端可以在任意时刻开始发送字符，因此必须在每一个字符的开始和结束的地方加上标志，即加上开始位和停止位，以便使接收端能够正确地将每一个字符接收下来。所传送的数据以字节为单位。每个字节前加上一位起始位，每个字节的后面加上停止位。好处：异步通信的好处是通信设备简单、便宜，但传输效率较低。

——同步通信

同步通信时要建立发送方时钟对接收方时钟的直接控制，使双方达到完全同步。此时，传输数据的位之间的距离均为“位间隔”的整数倍，同时传送的字符间不留间隙，即保持位同步关系，也保持字符同步关系。发送方对接收方的同步可以通过两种方法实现。



串行通信的传输方向；

串行通信常见的错误校验：

1、奇偶校验

在发送数据时，数据位尾随的1位为奇偶校验位（1或0）。奇校验时，数据中“1”的个数与校验位“1”的个数之和应为奇数；偶校验时，数据中“1”的个数与校验位“1”的个数之和应为偶数。接收字符时，对“1”的个数进行校验，若发现不一致，则说明传输数据过程中出现了差错。

2、代码和校验

代码和校验是发送方将所发数据块求和（或各字节异或），产生一个字节的校验字符（校验和）附加到数据块末尾。接收方接收数据同时对数据块（除校验字节外）求和（或各字节异或），将所得的结果与发送方的“校验和”进行比较，相符则无差错，否则即认为传送过程中出现了差错。

3、循环冗余校验

这种校验是通过某种数学运算实现有效信息与校验位之间的循环校验，常用于对磁盘信息的传输、存储区的完整性校验等。这种校验方法纠错能力强，广泛应用于同步通信中。

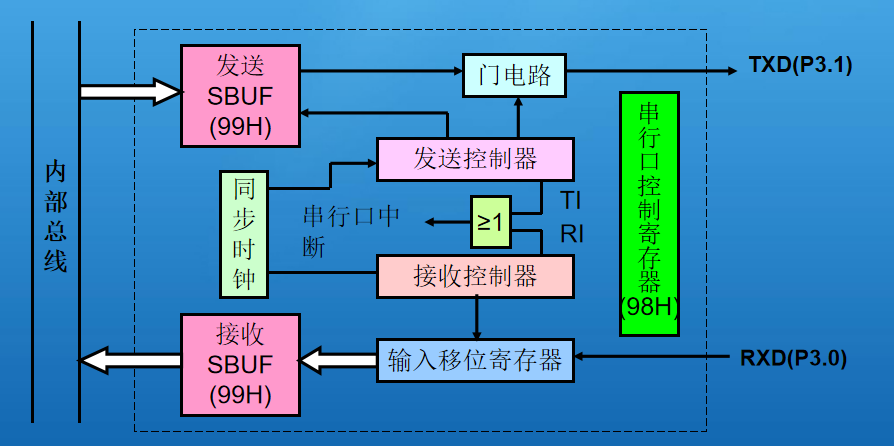
传输速率：

比特率是每秒钟传输二进制代码的位数，单位是：位／秒（bps）。如每秒钟传送240个字符，而每个字符格式包含10位(1个起始位、1个停止位、8个数据位)，这时的比特率为：

10位×240个/秒 = 2400 bps

串行通信接口标准：略；

——串行接口的结构



有两个物理上独立的接收、发送缓冲器SBUF，它们占用同一地址99H， 在逻辑上，SBUF只有一个，它既表示发送寄存器，又表示接收寄存器，具有同一个单元地址99H。但在物理结构上，则有两个完全独立的SBUF，一个是发送缓冲寄存器SBUF，另一个是接收缓冲寄存器SBUF。如果CPU写SBUF，数据就会被送入发送寄存器准备发送；如果CPU读SBUF，则读入的数据一定来自接收缓冲器。即CPU对SBUF的读写，实际上是分别访问上述两个不同的寄存器。 a = SBUF; SBUF = a;

**串行口工作之前需对相关寄存器进行配置，设定其工作模式**。

**1.设置T1的工作方式（编程TMOD寄存器）；**

**2.计算T1的初值，装载TH1、TL1；**

**3.启动T1（编程TCON中的TR1位）；**

**4.确定串行口控制（编程SCON寄存器）；**

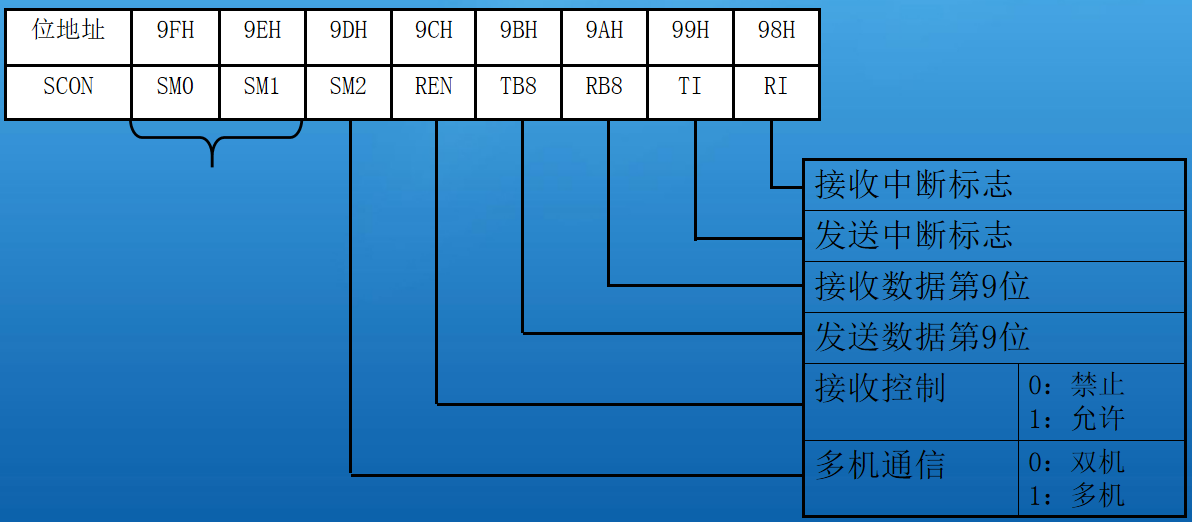
**5.如需串行口在中断方式工作时，**

**要进行中断设置编程IE寄存器。**

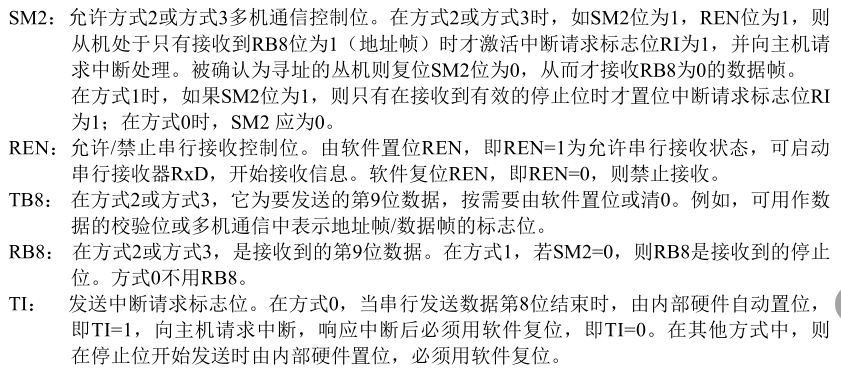


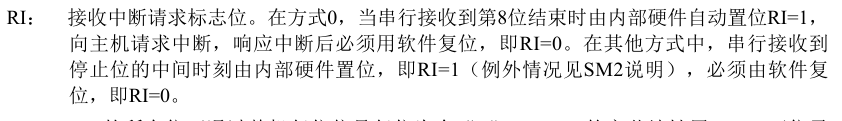
**与串行通信相关的寄存器**

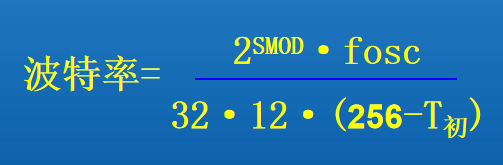
串行口控制寄存器SCON(可位寻址)









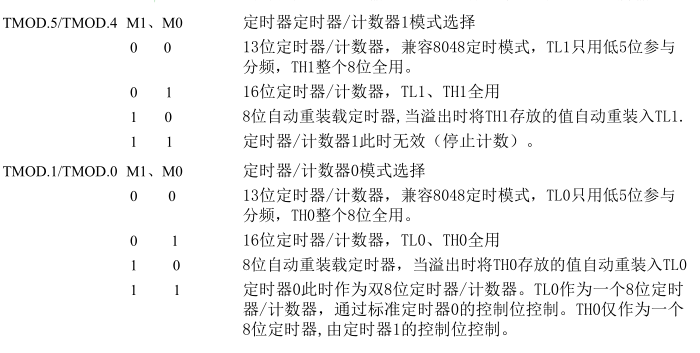
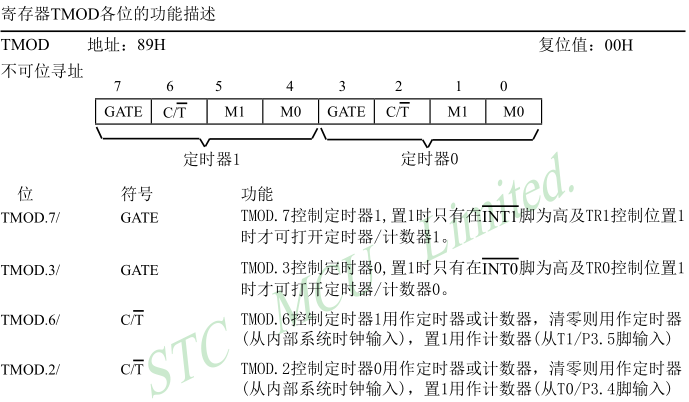
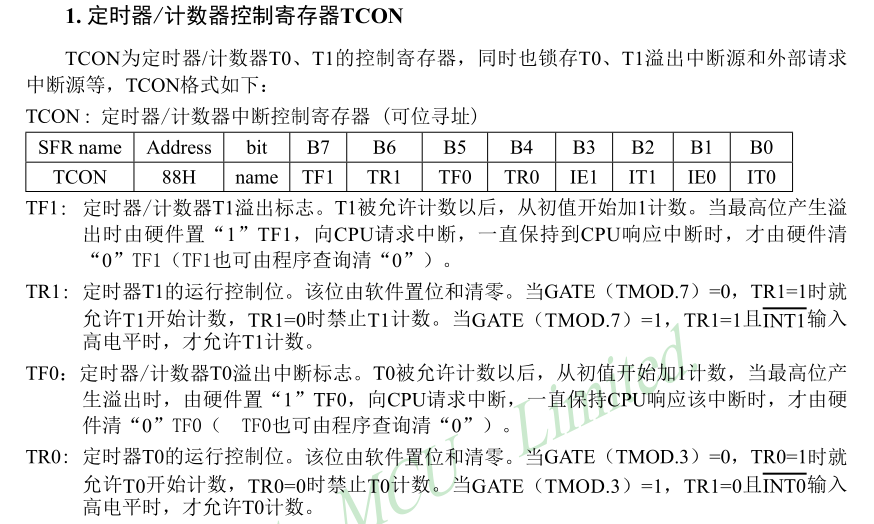
当T1作为波特率发生器时，最典型的用法是使T1工作在自动再装入的8位定时器方式（即方式2，且TCON的TR1=1，以启动定时器）。这时溢出率取决于TH1中的计数值。

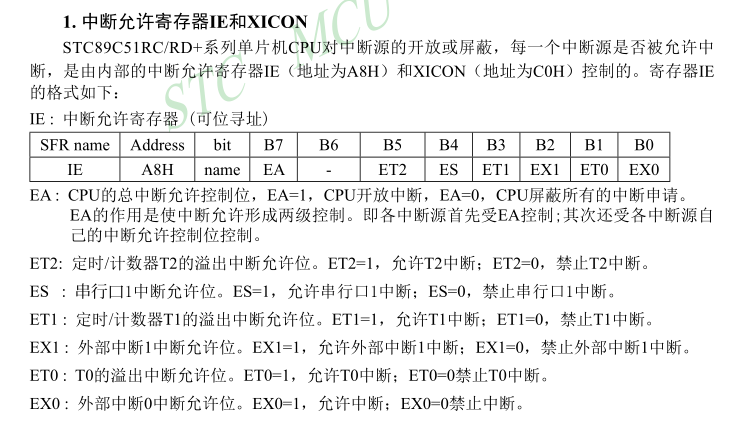
T1 溢出率 = fosc /{12×[256 －（TH1）]}

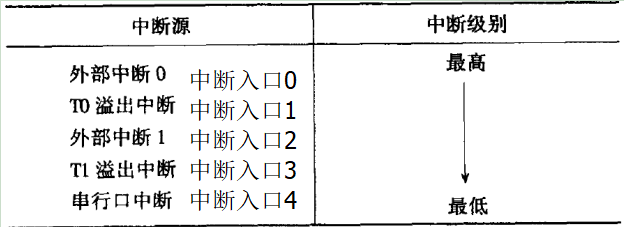
——使用辅助软件计算初值 11.0592MHZ

常用串口波特率：

300、600、1200、2400、4800、9600、19200 ……115200；





单片机同优先级中内部查询顺序

void UART() interrupt 4

//串口中断处理函数，

//加关键字interrupt和入口号

{

中断处理语句

}



#include <reg52.h>

#include <intrins.h>

#define uint unsigned int

#define uchar unsigned char

sbit DU = P2^6;//数码管段选

sbit WE = P2^7;//数码管段选

uchar num;//数码管显示的值

//共阴数码管段选表0-9

uchar code SMGduan[]= {

//0 1 2 3 4 5 6 7 8

0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,

//9 A B C D E F H L

0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x76, 0x38,

//n u - 熄灭

0x37, 0x3E, 0x40, 0x00 };

//数码管位选码

uchar code SMGwei[] = { 0xfe, 0xfd, 0xfb,0xf7};

void displayl(uchar i)

{

static uchar wei;

P0 = 0XFF;//清除断码

WE = 1;//打开位选锁存器

P0 = SMGwei[wei];

WE = 0;//锁存位选数据

switch(wei)

{

case 0: DU = 1; P0 = SMGduan[i / 1000]; DU = 0; break;

case 1: DU = 1; P0 = SMGduan[(i%1000)/100]; DU = 0; break;

case 2: DU = 1; P0 = SMGduan[i%100/10]; DU = 0; break;

case 3: DU = 1; P0 = SMGduan[i%10]; DU = 0; break;

}

wei++;

if(wei == 4)

wei = 0;

}

//定时器0初始化

void timer0Init()

{

EA = 1; //打开总中断

ET0 = 1;//打开定时器0中断

TR0 = 1; //启动定时器0

REN = 1;//允许串口接收

TMOD |= 0X01; //定时器工作模式1，16位定时模式

TH0 = 0xED;

TL0 = 0xFF; //定时5ms

}

//串口初始化

void UARTInit()

{

EA = 1; //打开总中断

ES = 1; //打开串口中断

SM0 = 0; SM1 = 1;//串口工作方式1,8位UART波特率可变

REN = 1;//串口允许接收

TR1 = 1;//启动定时器1

TMOD |= 0x20;//定时器1，工作模式2 8位自动重装

TH1 = 0xfd;

TL1 = 0xfd;//设置比特率9600

}

void main()//main函数自身会循环

{

timer0Init();//定时器0初始化

UARTInit();//串口初始化

while(1);

}

//定时器0中断函数

void timer0() interrupt 1

{

TH0 = 0xED;

TL0 = 0xFF; //定时5ms

displayl(num); //数码管显示函数

}

//串口中断函数

void UART() interrupt 4

{

uchar temp;

if(RI)//判断接收是否完成

{

num = SBUF;//读SBUF，读出串口接收到的数据

RI = 0;//软件清零接收标志位

temp = num;//

SBUF = ++temp;//写SBUF，把要发送的数据送给发送缓存器

}

if(TI)//判断是否发送完成

TI = 0;//清零发送完成标志位

}

volatile unsigned char sending;

void send(unsigned char d) //发送一个字节的数据，形参d即为待发送数据。

{

SBUF=d; //将数据写入到串口缓冲

sending=1; //设置发送标志

while(sending); //等待发送完?

}

void sendc(unsigned char \* pd)

{

while((\*pd)!='\0') //发送字符串，直到遇到0才结束

{

send(\*pd); //发送一个字符

pd++; //移动到下一个字符

}

}

void UART() interrupt 4 //串口发送中断

{

if(RI) //收到数据

{

RI=0; //清中断请求

}

else //发送完一字节数据

{

TI=0;

sending=0; //清正在发送标志

} }

1. **IIC总线EEPROM**

1.常用的串行总线协议：

目前常用的微机与外设之间进行数据传输的串行总线主要有UART、1-wire、I2C和SPI总线。

UART:是以异步方式进行通信（一条数据输入线，一条数据输出线）。

1-wire：即单线总线，又叫单总线(只有一条线)。

I2C：同步串行2线方式进行通信（一条时钟线，一条数据线）。

SPI：同步串行3线方式进行通信（一条时钟线，一条数据输入线，一条数据输出线）。

UART总线异步串口

1-wire

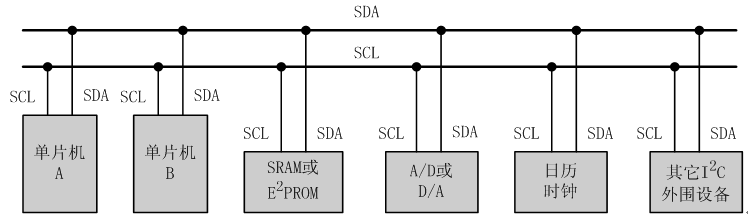
I2C总线 （本节课主讲内容）

SPI总线

2.I2C串行总线的组成及工作原理：

I2C总线是PHLIPS公司推出的一种串行总线，它只有两根双向信号线。一根是数据线SDA（serial data I/O），另一根是时钟线SCL（serial clock）。

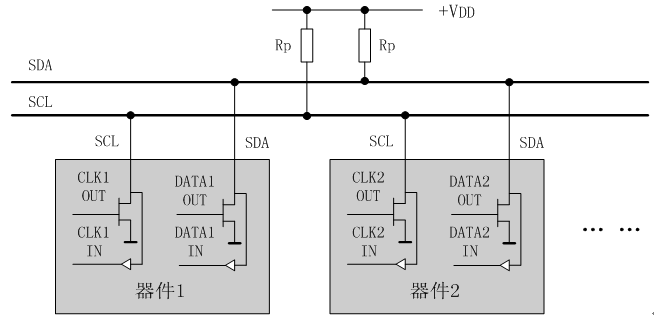
如下图所示，IIC总线上可以挂多个器件，而每个器件都有唯一的地址，这样可以标识通信目标。数据的通信的方式采用主从方式，主机负责主动联系从机，而从机则被动回应数据。



在多主机系统中，可能同时有几个主机企图启动总线传送数据。为了避免混乱， I2C总线要通过总线仲裁，以决定由哪一台主机控制总线。

在80C51单片机应用系统的串行总线扩展中，我们经常遇到的是以80C51单片机为主机，其它接口器件为从机的单主机情况。

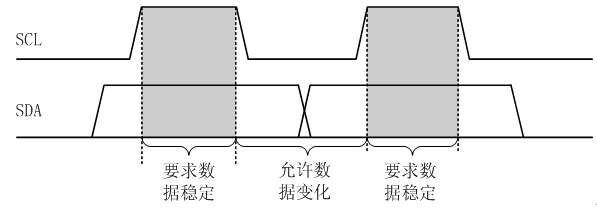
I2C总线通过上拉电阻接正电源。当总线空闲时，两根线均为高电平。连到总线上的任一器件输出的低电平，都将使总线的信号变低，即各器件的SDA及SCL都是线“与”关系。



1. I2C总线传输协议

——数据位的有效性规定

SCL为高电平期间，数据线上的数据必须保持稳定，只有SCL信号为低电平期间，SDA状态才允许变化。



4.I2C的起始和终止信号

SCL线为高电平期间，SDA线由高电平向低电平的变化表示起始信号；SCL线为高电平期间，SDA线由低电平向高电平的变化表示终止信号。

起始和终止信号都是由主机发出的，在起始信号产生后，总线就处于被占用的状态；在终止信号产生后，总线就处于空闲状态。

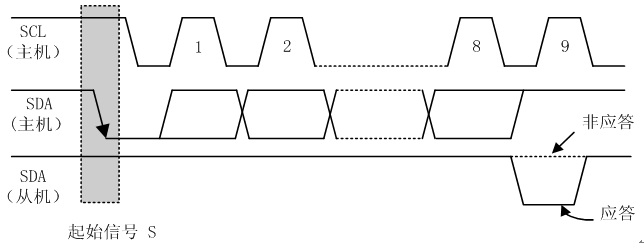
连接到I2C总线上的器件，若具有I2C总线的硬件接口，则很容易检测到起始和终止信号。

接收器件收到一个完整的数据字节后，有可能需要完成一些其它工作，如处理内部中断服务等，可能无法立刻接收下一个字节，这时接收器件可以将SCL线拉成低电平，从而使主机处于等待状态。直到接收器件准备好接收下一个字节时，再释放SCL线使之为高电平，从而使

数据传送可以继续进行。

5.I2C字节的传送与应答

每一个字节必须保证是8位长度。数据传送时，先传送最高位（MSB），每一个被传送的字节后面都必须跟随一位应答位（即一帧共有9位）。



——应答位的作用

主机在发送数据时，每次发送一字节数据，都需要读取从机应答位，当从机空闲可以接收该字节数据时，从机会发出应答（一帧数据的第9位为“0”），当从机正忙于其他工作的处理来不及接收主机发送的数据时，从机会发出非应答（一帧数据的第9位为“1”）主机则应发出终止信号以结束数据的继续传送，主机通过从机发出的应答位来判断从机是否成功接收数据。

当主机接收数据时，它收到最后一个数据字节后，必须向从机发出一个结束传送的信号。这个信号是由对从机的“非应答”来实现的。然后，从机释放SDA线，以允许主机产生终止信号。

5.数据帧格式

I2C总线上传送的数据信号是广义的，既包括地址信号，又包括真正的数据信号。

在起始信号后必须传送一个从机的地址（7位），第8位是数据的传送方向位（R/T），用“0”表示主机发送数据（T），“1”表示主机接收数据（R）。每次数据传送总是由主机产生的终止信号结束。但是，若主机希望继续占用总线进行新的数据传送，则可以不产生终止信号，马上再次发出起始信号对另一从机进行寻址。

在总线的一次数据传送过程中，可以有以下几种组合方式：

I.主机向从机发送数据，数据传送方向在整个传送过程中不变：



注：有阴影部分表示数据由主机向从机传送，无阴影部分则表示数据由从机向主机传送。

A表示应答， A非表示非应答（高电平）。S表示起始信号，P表示终止信号。

Ii.主机在第一个字节后，立即从从机读数据



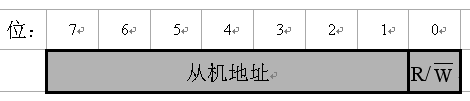
iii.在传送过程中，当需要改变传送方向时，起始信号和从机地址都被重复产生一次，但两次读/写方向位正好反向。



6.总线的寻址

I2C总线协议有明确的规定：采用7位的寻址字节（寻址字节是起始信号后的第一个字节）。

（1）寻址字节的位定义



D7～D1位组成从机的地址。D0位是数据传送方向位，为“0”时表示主机向从机写数据，为“1”时表示主机由从机读数据。

主机发送地址时，总线上的每个从机都将这7位地址码与自己的地址进行比较，如果相同，则认为自己正被主机寻址，根据R/T位将自己确定为发送器或接收器。

从机的地址由固定部分和可编程部分组成。在一个系统中可能希望接入多个相同的从机，从机地址中可编程部分决定了可接入总线该类器件的最大数目。如一个从机的7位寻址位有4位是固定位，3位是可编程位，这时仅能寻址8个同样的器件，即可以有8个同样的器件接入到该I2C总线系统中。

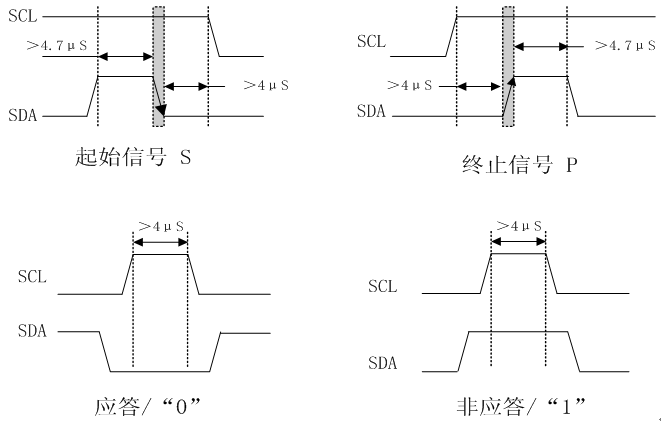
7.80C51单片机I2C串行总线器件的接口

总线数据传送的模拟

主机可以采用不带I2C总线接口的单片机，如80C51、STC89C52等单片机，利用软件实现I2C总线的数据传送，即软件与硬件结合的信号模拟。

——总线数据传送的模拟

为保证数据的可靠性，I2C总线的数据传送有严格的时序要求。I2C总线的起始信号、终止信号、发送“0”及发送“1”的模拟时序 ：



实例：

#include <reg52.h>

#include <intrins.h>

#define uint unsigned int

#define uchar unsigned char

#define At24c02ADDR 0XA0

#define I2cRead 1

#define I2cWrite 0

bit AckFlag;

sbit SCL =P2^1;

sbit SDA =P2^0;

sbit DU = P2^6;//数码管段选

sbit WE = P2^7;//数码管段选

uchar num;//数码管显示的值

//共阴数码管段选表0-9

uchar code SMGduan[]= {

//0 1 2 3 4 5 6 7 8

0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,

//9 A B C D E F H L

0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x76, 0x38,

//n u - 熄灭

0x37, 0x3E, 0x40, 0x00 };

//数码管位选码

uchar code SMGwei[] = { 0xfe, 0xfd, 0xfb,0xf7};

void delay(unsigned int z) //延时程序

{ //参数 ：z 延时毫秒设定，取值范围0-65535

unsigned int x,y; //返回值 ：无

for(x=z;x>0;x--) //描述 ：12T/Fosc11.0592M毫秒级延时

for(y=114;y>0;y--);

}

void displayl(uchar i)

{

static uchar wei;

P0 = 0XFF;//清除断码

WE = 1;//打开位选锁存器

P0 = SMGwei[wei];

WE = 0;//锁存位选数据

switch(wei)

{

case 0: DU = 1; P0 = SMGduan[i / 1000]; DU = 0; break;

case 1: DU = 1; P0 = SMGduan[(i%1000)/100]; DU = 0; break;

case 2: DU = 1; P0 = SMGduan[i%100/10]; DU = 0; break;

case 3: DU = 1; P0 = SMGduan[i%10]; DU = 0; break;

}

wei++;

if(wei == 4)

wei = 0;

}

//定时器0初始化

void timer0Init()

{

EA = 1; //打开总中断

ET0 = 1;//打开定时器0中断

TR0 = 1; //启动定时器0

TMOD = 0X01; //定时器工作模式1，16位定时模式

TH0 = 0xED;

TL0 = 0xFF; //定时5ms

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ICC代码

void delay5us()

{

\_nop\_();

}

void I2cStart() //I2C总线起始信号

{//时钟总线为高电平期间数据总线又高变低产生起始信号

SCL = 1;

SDA = 1;

delay5us();

SDA = 0;

delay5us();

}

void IIC\_Init()

{

SDA=1;

delay5us();

SCL=1;

delay5us();

}

void I2cStop() //I2C总线停止信号

{//时钟总线为高电平期间，数据总线从高变低产生终止信号

SCL = 0;

SDA = 0;

SCL = 1;

delay5us();

SDA = 1;

delay5us();

}

bit ReadACK() //I2C总线读从机应答信号

{

SCL = 0; //拉低时钟总线，允许从机控制SDA

SCL = 1; //拉高，读SDA

delay5us();

if(SDA)

{ //NOACK

SCL = 0;

return(1); //非应答返回1

}

else //ACK 即应答

{

SCL = 0;

return(0); //应答返回0

}

}

void SendACK(bit i) //主机发送应答信号

{

SCL = 0; //拉低时钟总线，允许主机控制SDA

if(i) //发非应答

SDA = 1;

else //发应答

SDA = 0;

SCL = 1; //拉高总线，让从机读SDA

delay5us();//保持5us

SCL = 0; //拉低时钟总线，允许SDA释放

SDA = 1;//释放数据总线

}

void I2cSendByte(uchar DAT)//I2C发送一个字节数据

{

uchar i;

for(i=0;i<8;i++) //分别写8次，每次写1位

{

SCL = 0; //拉低时钟总线，允许SDA变化

if(DAT & 0X80)//先写数据最高位

SDA = 1; //写1

else

SDA = 0; //写0

SCL = 1; //拉高时钟，让从机读SDA

DAT <<= 1; //左移一位

}

SCL = 0;//拉低时钟总线，允许SDA释放

SDA = 1;//释放数据总线

}

void At24c02Write(uchar ADDR,uchar DAT) //At24c02指定单元写入一个字节数据

{

I2cStart();

I2cSendByte(At24c02ADDR + I2cWrite);

// 0xA0 + 0 发送器件地址加读写方向位

if(ReadACK()) //读从机应答

AckFlag = 0;//NOACK

else

AckFlag = 1;//ACK

I2cSendByte(ADDR);//发送储存单元地址字节

if(ReadACK()) //读从机应答

AckFlag = 0;//NOACK

else

AckFlag = 1;//ACK

I2cSendByte(DAT);//发送一字节数据

if(ReadACK())//读从机应答

AckFlag = 1; //NOACK

else

AckFlag = 0; //ACK

I2cStop(); //I2C停止信号

}

uchar I2cReadByte() //I2C总线读一字节数据

{

uchar i, DAT;

for(i=0; i<8; i++)//分别读8次，每次读一位

{

DAT <<= 1; //数据左移1位，准备接收一位

SCL = 0; //拉低时钟总线，允许从机控制SDA变化

SCL = 1; //拉高时钟总线，读取SDA上的数据

if(SDA)

DAT |= 0X01;//为1则写1，否则不行执行写1，通过左移补0

}

return(DAT); //返回读出的数据

}

uchar At24c02Read(uchar ADDR)//读AT24C02指定单元内数据

{

uchar DAT;

I2cStart();//I2C起始信号

I2cSendByte(At24c02ADDR + I2cWrite);//发送器件地址加读写方向位

if(ReadACK())//读从机应答

AckFlag = 1; //NOACK

else

AckFlag = 0; //ACK

I2cSendByte(ADDR);//I2C发送一个字节

ReadACK();//读从机应答

I2cStart();//再次产生I2C起始信号

I2cSendByte(At24c02ADDR + I2cRead);//发送器件地址加读写方向位 读

if(ReadACK())//读从机应答

AckFlag = 1; //NOACK

else

AckFlag = 0; //ACK

DAT = I2cReadByte();//读一字节

SendACK(1);//主机发送非应答

I2cStop(); //I2C停止信号

return(DAT);//返回读出数据

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

timer0Init();//定时器0初始化

EA = 0;//屏蔽中断

At24c02Write(3, 84);//给第3单元写入数据“188”

delay(5);//延时等待AT24C02处理

num = At24c02Read(3);//读出第3单元内数据送给显示变量

if(AckFlag)//当从机非应答

P1 = 0;//亮P1所有灯

else

P1 = 0XFF;//灭P1所有灯

EA = 1;//开中断

while(1);

}

void timer0() interrupt 1

{

TH0 = 0xED;

TL0 = 0xFF; //定时5ms

displayl(num);

}