电子元件

排阻

有两种：

- n
- n+1

二极管

几个概念

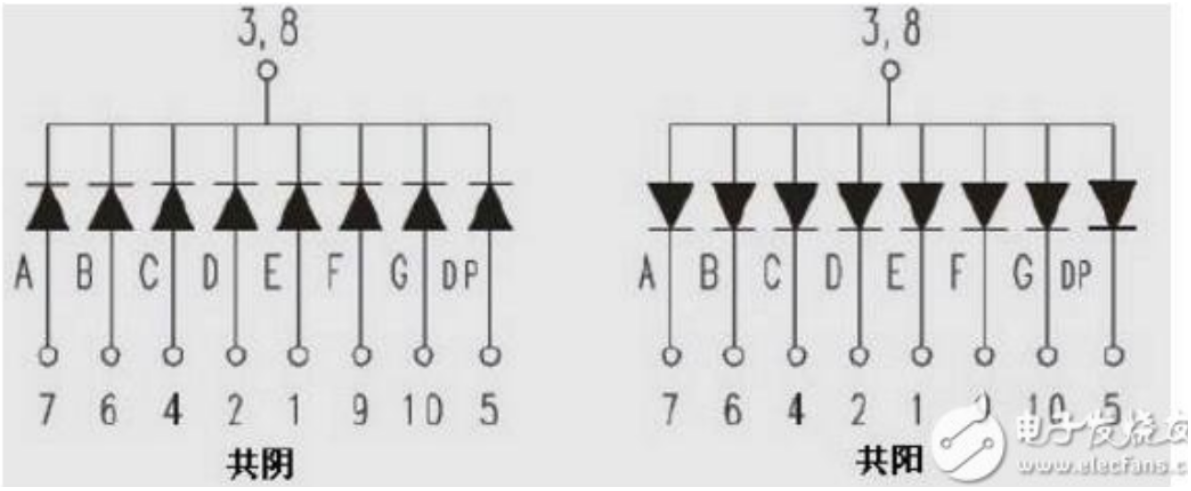
1. 限流电阻：与二极管串联，防止电流过大烧毁二极管
2. 导通压降：二极管亮起的最小电压

贴片式绿色为阴极，插入式短脚为阴极

数码管

共阴极：对应段选信号置1亮

共阳极：对应段选信号置0亮



数码管位选一般是低电平有效

段选：数码管的哪一段（节）亮

位选：哪个数码管亮

多个数码管可利用LED余晖和视觉暂留效应，辅以“消影”，实现“同时”显示不同数字

```
#include <reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit dula=P2^6;
sbit wela=P2^7;
uchar num;
uchar code number[]=
{
    //0-9   A-F,??????????
    0x3f,0x06,0x5b,0x4f,
```

```
    0x66,0x6d,0x7d,0x07,
    0x7f,0x6f,0x77,0x7c,
    0x39,0x5e,0x79,0x71
};
uchar code dig[]=
{   //第1个-第6个
    0xfe,0xfd,0xfb,0xf7,
    0xef,0xdf
};
void delayms(uint);
void main()
{
    while(1)
    {
        for(num=0;num<6;num++)
        {
            dula=1;
            P0=number[num];
            dula=0;
            P0=0xff;
            wela=1;
            P0=dig[num];
            wela=0;
            delayms(50);
        }
    }
}
void delayms(uint xms)
{
    uint i,j;
    for(i=xms;i>0;i--)
    for(j-110;j>0;j--)
    ;
}
```

74HC573锁存器

OE为低电平时工作，否则输出高阻态

LE为高电平时输出Q跟随输入D变化，否则锁定为之前状态

独立按键

一端直接与低电平相连，另一端与单片机引脚相连时可以通过低电平而直接检测按键是否被按下

缺点：一个独立按键需要一个独立IO口，浪费资源

矩阵键盘

通常是多个按键以矩阵形式组成键盘，通过逐行扫描的方法来检测按键被按下的情况

矩阵键盘扫描显示代码

```
#include<reg52.h>
#define uchar unsigned char
#define uint unsigned int
sbit dula=P2^6;
sbit wela=P2^7;
uchar code table[]={
0x3f,0x06,0x5b,0x4f,
0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,
0x39,0x5e,0x79,0x71
};
//uchar code segnum[6]={0x00,0x00,0x00,0x00,0x00,0x00};
//uint index=0;
void delayms(uint xms)
{
    uint i,j;
    for(i=xms;i>0;i--)
        for(j=110;j>0;j--);
}
void display(uchar num)
{
    P0=table[num];//data first
    dula=1;//SS followed
    dula=0;
}
void clear()
{
    P0=0x00;
    dula=1;
    dula=0;
}
uchar matrixkeyscan()
{
    uchar tmp,key;

    P3=0xfe;//the first row
    tmp=P3;
    tmp=tmp&0xf0;//how and why
    if(tmp!=0xf0)
    {
        delayms(10);
        tmp=P3;
        tmp=tmp&0xf0;
        if(tmp!=0xf0)
        {
            tmp=P3;
            switch(tmp)
            {
                case 0xee:
```

```
                    key=0;
                break;
                case 0xde:
                    key=1;
                break;
                case 0xbe:
                    key=2;
                break;
                case 0x7e:
                    key=3;
                break;
            }
            display(key);
            while(tmp!=0xf0)
            {
                tmp=P3;
                tmp=tmp&0xf0;
            }
            return key;
            //display(key);
        }
    }

    P3=0xfd;//the second row
    tmp=P3;
    tmp=tmp&0xf0;//how and why
    if(tmp!=0xf0)
    {
        delayms(10);
        tmp=P3;
        tmp=tmp&0xf0;
        if(tmp!=0xf0)
        {
            tmp=P3;
            switch(tmp)
            {
                case 0xed:
                    key=4;
                break;
                case 0xdd:
                    key=5;
                break;
                case 0xbd:
                    key=6;
                break;
                case 0x7d:
                    key=7;
                break;
            }
            display(key);
            while(tmp!=0xf0)
            {
                tmp=P3;
                tmp=tmp&0xf0;
            }
```

```
            return key;
            //display(key);
        }
    }

    P3=0xfb;//the third row
    tmp=P3;
    tmp=tmp&0xf0;//how and why
    if(tmp!=0xf0)
    {
        delayms(10);
        tmp=P3;
        tmp=tmp&0xf0;
        if(tmp!=0xf0)
        {
            tmp=P3;
            switch(tmp)
            {
                case 0xeb:
                    key=8;
                break;
                case 0xdb:
                    key=9;
                break;
                case 0xbb:
                    key=10;
                break;
                case 0x7b:
                    key=11;
                break;
            }
            display(key);
            while(tmp!=0xf0)
            {
                tmp=P3;
                tmp=tmp&0xf0;
            }
            return key;
            //display(key);
        }
    }

    P3=0xf7;//the fourth row
    tmp=P3;
    tmp=tmp&0xf0;//how and why
    if(tmp!=0xf0)
    {
        delayms(10);
        tmp=P3;
        tmp=tmp&0xf0;
        if(tmp!=0xf0)
        {
            tmp=P3;
            switch(tmp)
            {
```

```
                case 0xe7:
                    key=12;
                break;
                case 0xd7:
                    key=13;
                break;
                case 0xb7:
                    key=14;
                break;
                case 0x77:
                    key=15;
                break;
            }
            display(key);
            while(tmp!=0xf0)
            {
                tmp=P3;
                tmp=tmp&0xf0;
            }
            return key;
        }
    }
    return 0;
}

void main()
{
    P0=0;
    dula=1;
    dula=0;//clear led
    P0=0xc0;//low enable
    wela=1;
    wela=0;
    while(1)
    {
        matrixkeyscan();
        clear();
    }
}
```

元件作用

上拉电阻

与单片机内部电阻并联，减小工作元件之前的压降，提高工作电压

单片机

周期

脉冲信号之间的时间间隔称为周期

时钟周期

单片机最小的时间单位

状态周期（暂时不懂干嘛的）

时钟周期的两倍

机器周期

单片机一个基本操作的周期，比如取指令、读写寄存器

指令周期

一条指令的完整执行周期，比如`mov ax,2`

寄存器

PSW（program status word）

CY	AC	F0	RS1	RS0	OV	- -	P
进位标志	辅助进位标志	用户用于测试程序	与RS0用来在4组工作寄存器里做选择		溢出标志位		奇偶标志位

ROM（read only memmory）

ROM的数据在程序运行的时候是不容改变的，除非你再次烧写程序，他就会改变，就像我们的书本，印上去就改不了了，除非再次印刷，这个就是ROM的原理

RAM（random access memory）

RAM就是在程序运行中，数据会随时改变的，就像我们的黑板，写上了可以擦，擦完再写上去，相当于程序运行的时候，调用ROM里面的数据进行各种运算

sfr（special function register）

RAM中有特殊功能的寄存器，比如`sfr P1 = 0x90;`.sfr是8位，sfr16是16位

sbit

一个可用于位寻址空间的位地址`sbit led=P1^1`，与bit相比，sbit代表的是单片机地址，而bit仅仅可以用来当作变量

code

```
数组名前code关键字表示这个表格会存放在代码区代码区的内容，也就是程序，最后会被烧写到只读存储器中，运行中不可改变如果不写
code，一般放在内部存储区
```

中断

中断步骤

中断请求（**中断源**提出）

中断响应

中断服务

中断返回

中断嵌套

执行中断服务时遇到新的中断请求

中断允许和中断优先级

中断程序的重要程度，有多个中断请求时，利用中断优先级判断是否响应中断（是否可屏蔽）以及先响应哪个中断，是否响应中断存储于**中断允许寄存器IE**中，优先级内容存储于**优先级寄存器IP**中

eg.52单片机所有中断源及中断级别

中断源	默认中断级别	序号（C语言用）	入口地址
INT0-外部中断0	top	0	03h
T0-计数器0中断	2		0bh
INT1	3		13h
T1	4		1Bh
T1/R1-串行口中断	5		23h
T2	bottom	5	2bh

IE：

IE——中断允许寄存器（可位寻址）

B7	B6	B5	B4	B3	B2	B1	B0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA：中断总开关

ET2：定时器2的中断允许

ES：串行口中断允许

ET1：定时器1中断允许

EX1：外器中断1中断允许

ET0：定时器0中断允许

EX0：外部中断0的中断允许

IP：

IP——中断优先级控制寄存器（可位寻址）

B7	B6	B5	B4	B3	B2	B1	B0
-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2：定时器2中断优先

PS ：串行口中断优先

PT1：定时器1中断优先

PX1：外部中断1中断优先

PT0：定时器0中断优先

PX0：外部中断0中断优先

为1是是高优先级，为0时是低优先级，用于中断嵌套执行决策，

若同时有多个同级中断请求，按照默认顺序响应

串口通信

并行通信

成本较高，同时接收有困难

串行通信

发送时：并->串

接收时：串->并

成本低，适合远距离传输

异步串行通信

以字符（帧）的形式传输，字符间间隙不定，但字符内间隙一定，是位间隔的整数倍。

由其定义可知每一帧需要4部分：起始位、数据位、奇偶校验位、停止位

同步串行通信

字符间无间隙，字符内间隙一定，是位间隔的整数倍

串行通信制式

1. 单工
2. 半双工：不能同时互传
3. 全双工

串行通信的错误校验

1. 奇偶校验
2. 代码和校验（类似于互联网运输层EDC）
3. 循环冗余校验

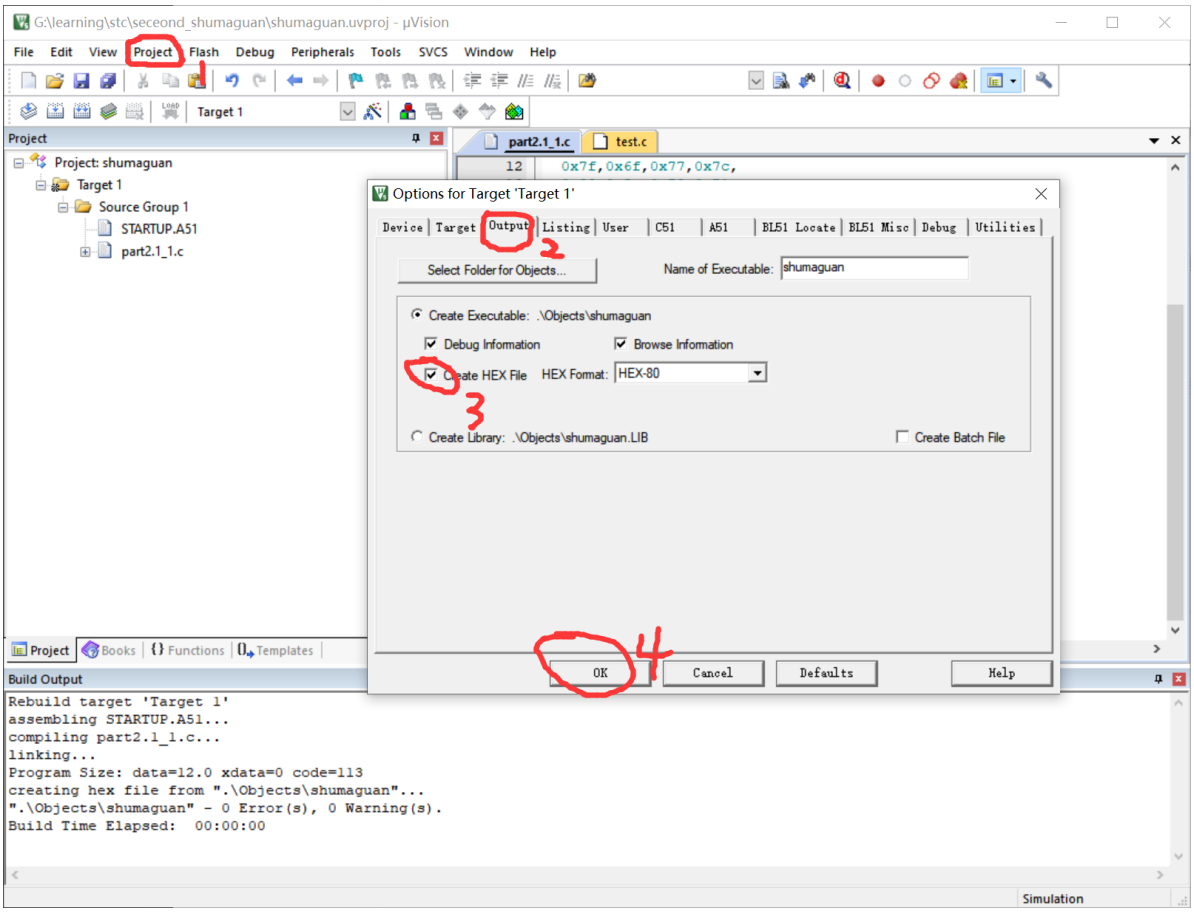
keil使用

include ""与<>的区别

<>：编译器进入软件所在文件夹搜索该头文件，keil是Keil\C51\INC里寻找

""：编译器先在工程文件夹下找该头文件，找不到在去软件的文件夹寻找

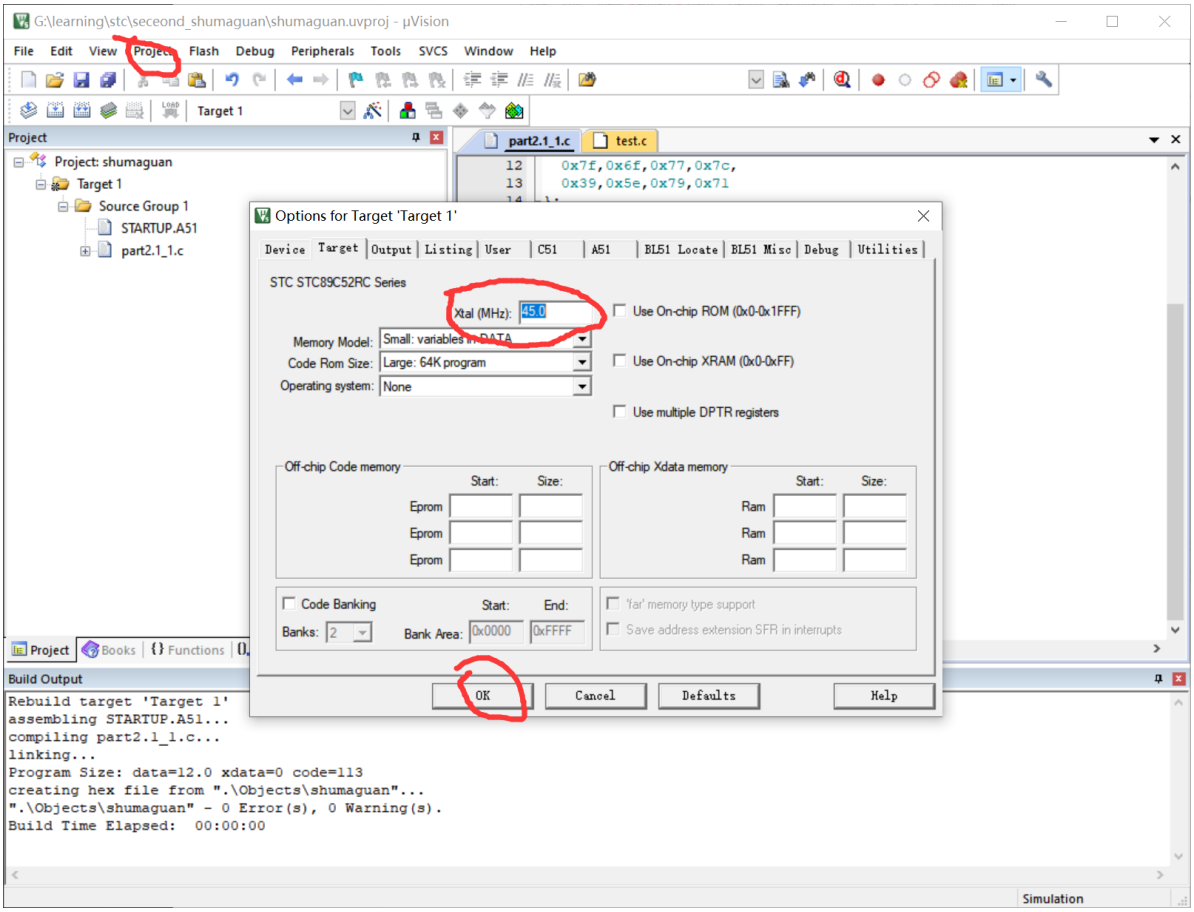
输出hex文件



该勾选含义是在build时输出能够下载到单片机的hex文件。

仿真（需要计算时间）

震荡频率设置



仿真操作