# 电路基本知识

## 基础知识

电压：U 单位：V（伏特） 1V=1000 mV

电流：I 单位：A（安） 1A=1000 mA

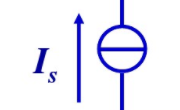
电阻：R 单位：Ω（欧姆） 1KΩ = 1000Ω

电导：G=

单位：S

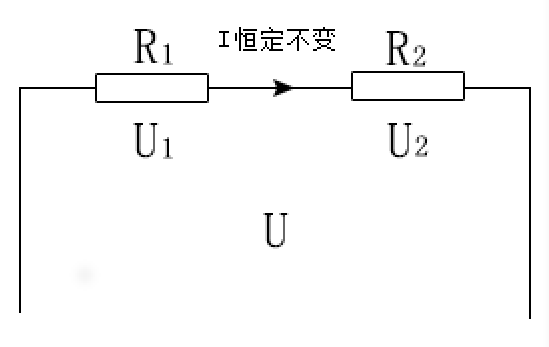
欧姆定律： U = I ×R I= R = 

电流源和电压源

 电流源

电压源

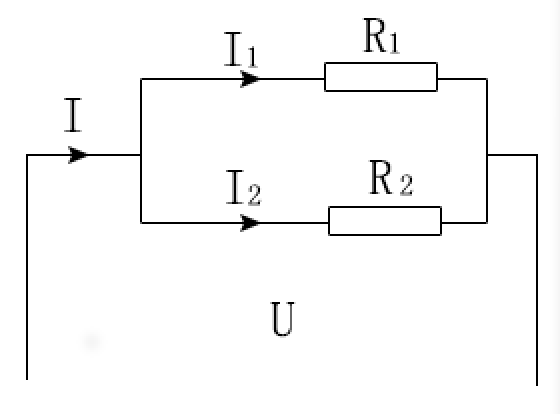
串联电路



串连特性： U = U1 + U2 I = I1 = I2 R = R1 + R2

U1 =  U U2 =  U  = 

并联电路



并连特性： U = U1= U2 I = I1 + I2 R = R1 + R2

I1 = I I2 = I  =  R = 

## 电平

在[数字逻辑电路](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E9%80%BB%E8%BE%91%E7%94%B5%E8%B7%AF" \t "https://baike.baidu.com/item/%E9%AB%98%E7%94%B5%E5%B9%B3/_blank)中，低电平表示0，高电平表示1。一般规定低电平为0~0.25V，高电平为3.5~5V。

## 电阻

电阻，用中学的物理定义说，就是电路中某元件阻碍电流流通的能力

1. 限流：为使通过用电器的电流不超过额定值或实际工作需要的规定值,以保证用电器的正常工作,通常可在电路中串联一个可变电阻。当改变这个电阻的大小时,电流的大小也随之改变。我们把这种可以限制电流大小的电阻叫做限流电。
2. 分流：当在电路的干路上需同时接入几个额定电流不同的用电器时,可以在额定电流较小的用电器两端并联接入一个电阻,这个电阻的作用是分流。
3. 分压：一般用电器上都标有额定电压值,若电源比用电器的额定电压高,则不可把用电器直接接在电源上，在这种情况下,可给用电器串接一个合适阻值的电阻,让它分担一部分电压,用电器便能在额定电压下工作。我们称这样的电阻为分压电阻。
4. 将电能转化为内能：电流通过电阻时,会把电能全部（或部分）转化为内能.用来把电能转化为内能的用电器叫电热器.如电烙铁、电炉、电饭煲、取暖器等等。

目前我已知的常用的，是用来分压

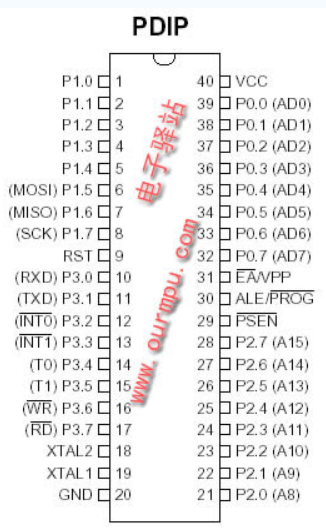
## 电容

uf

pf

# 单片机

## AT89S51 40 PDIP



### 主电源引脚（2根）

[IMG_256](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) VCC(Pin40)：电源输入，接＋5V电源  
[IMG_257](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) GND(Pin20)：接地线

### 外接晶振引脚（2根）

[IMG_256](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) XTAL1(Pin19)：片内振荡电路的输入端  
[IMG_257](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) XTAL2(Pin20)：片内振荡电路的输出端

### 控制引脚（4根）

[IMG_256](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) RST/VPP(Pin9)：复位引脚，引脚上出现2个机器周期的高电平将使单片机复位。  
[IMG_257](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) ALE/PROG(Pin30)：地址锁存允许信号  
[IMG_258](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) PSEN(Pin29)：外部存储器读选通信号  
[IMG_259](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) EA/VPP(Pin31)：程序存储器的内外部选通，接低电平从外部程序存储器读指令，如果接高电平则从内部程序存储器读指令。

### 可编程输入/输出引脚（32根）

AT89S51单片机有4组8位的可编程I/O口，分别位P0、P1、P2、P3口，每个口有8位（8根引脚），共32根。  
　 [IMG_256](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) PO口（Pin39～Pin32）：8位双向I/O口线，名称为P0.0～P0.7  
　 [IMG_257](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) P1口（Pin1～Pin8）：8位准双向I/O口线，名称为P1.0～P1.7  
　 [IMG_258](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) P2口（Pin21～Pin28）：8位准双向I/O口线，名称为P2.0～P2.7  
　 [IMG_259](http://www.51hei.com/UpFiles/Pic/mcu/2007-08/20070801012949696.gif) P3口（Pin10～Pin17）：8位准双向I/O口线，名称为P3.0～P3.7

## 晶振

晶振通常分为无源晶振和有源晶振两种类型，无源晶振一般称之为 crystal（晶体），而有源晶振则叫做 oscillator（振荡器）。

无源晶振分两脚和四脚，两脚的不分正负极

接法 每个脚对地，中间分别接一个30Pf的瓷片电容

## 单片机的烧录 //todo

# C语言

linux安装指令

yum install -y gcc

## 进制

在线转换工具：https://www.sojson.com/hexconvert/16to10.html

### 16进制

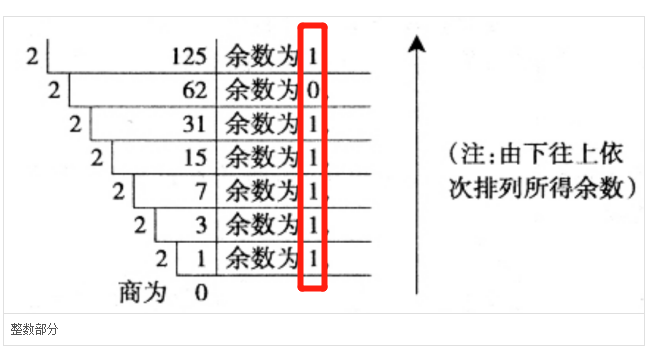
0x 代表这个数为16进制

十六进制（简写为hex或下标16）在[数学](https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6" \t "https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6/_blank)中是一种逢16进1的[进位制](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D%E5%88%B6" \t "https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6/_blank)。一般用数字0到9和字母A到F（或a~f）表示，其中:A~F表示10~15，这些称作十六进制数字。

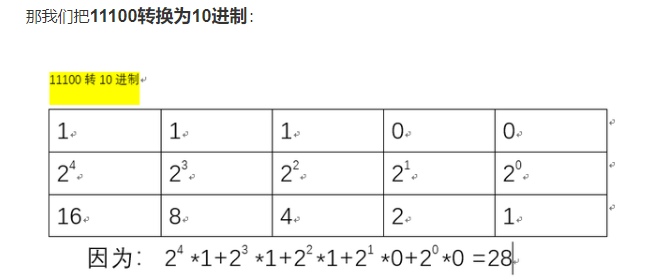
### 2进制

用两个不同的符号0（代表零）和1（代表一）来表示

10进制转2进制



2进制转10进制



## 关键字

|  |  |
| --- | --- |
| **关键字** | **说明** |
| auto | 声明自动变量 |
| case | 开关语句分支 |
| char | 声明字符型变量或函数返回值类型 |
| const | 定义常量，如果一个变量被 const 修饰，那么它的值就不能再被改变 |
| continue | 结束当前循环，开始下一轮循环 |
| default | 开关语句中的"其它"分支 |
| do | 循环语句的循环体 |
| double | 声明双精度浮点型变量或函数返回值类型 |
| else | 条件语句否定分支（与 if 连用） |
| enum | 声明枚举类型 |
| extern | 声明变量或函数是在其它文件或本文件的其他位置定义 |
| float | 声明浮点型变量或函数返回值类型 |
| for | 一种循环语句 |
| goto | 无条件跳转语句 |
| if | 条件语句 |
| int | 声明整型变量或函数 |
| long | 声明长整型变量或函数返回值类型 |
| register | 声明寄存器变量 |
| return | 子程序返回语句（可以带参数，也可不带参数） |
| short | 声明短整型变量或函数 |
| signed | 声明有符号类型变量或函数 |
| sizeof | 计算数据类型或变量长度（即所占字节数） |
| static | 声明静态变量 |
| struct | 声明结构体类型 |
| switch | 用于开关语句 |
| typedef | 用以给数据类型取别名 |
| unsigned | 声明无符号类型变量或函数 |
| union | 声明共用体类型 |
| void | 声明函数无返回值或无参数，声明无类型指针 |
| volatile | 说明变量在程序执行中可被隐含地改变 |
| while | 循环语句的循环条件 |

## 整数类型

|  |  |  |
| --- | --- | --- |
| **类型** | **存储大小** | **值范围** |
| char | 1 字节 | -128 到 127 或 0 到 255 |
| unsigned char | 1 字节 | 0 到 255 |
| signed char | 1 字节 | -128 到 127 |
| int | 2 或 4 字节 | -32,768 到 32,767 或 -2,147,483,648 到 2,147,483,647 |
| unsigned int | 2 或 4 字节 | 0 到 65,535 或 0 到 4,294,967,295 |
| short | 2 字节 | -32,768 到 32,767 |
| unsigned short | 2 字节 | 0 到 65,535 |
| long | 4 字节 | -2,147,483,648 到 2,147,483,647 |
| unsigned long | 4 字节 | 0 到 4,294,967,295 |

## 浮点类型

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **存储大小** | **值范围** | **精度** |
| float | 4 字节 | 1.2E-38 到 3.4E+38 | 6 位小数 |
| double | 8 字节 | 2.3E-308 到 1.7E+308 | 15 位小数 |
| long double | 16 字节 | 3.4E-4932 到 1.1E+4932 | 19 位小数 |

## 位运算符

用了位运算符，都会把数字转为二进制，然后进行运算

& | ^ ~ << >>

### &（按位与）两个都是1才是1

参加运算的两个运算量，如果两个数相应位的值都是1，则该位的结果值为1，否则为0。即：0 & 0 =0；0 & 1 =0；1 & 0 =0；1 & 1 =1。

1. 将某些二进制位屏蔽掉（保留一个数据中的某些位）。
2. 判断一个数据的某一位是否为1。

例如：

6&3

6转为二进制是110，3转为二进制是011，按位与之后是010,转为二进制后是2

### | （按位或）两个有一个是1就是1

例如：

6|3

6转为二进制是110，3转为二进制是011，按位或之后是111,转为二进制后是7

### ^（按位异或）

运算规则：参加运算的两个运算量，如果两个数的相应位的值不同，则该位的结果值为1，否则为0。

例如：

6|3

6转为二进制是110，3转为二进制是011，虽然最左边的一位数6是1，数3没有，相当于0，但这个时候继承数6的，左边第二位数3数6都是1，所以为0，左边第一位一个为0，一个为1，所以为1，转化结果为101，转为二进制后是5

### ~（按位取反）等于 ！

例如：

~12

12转为二进制为1100，内存中补码为00001100，取反为11110011 ，当内存中的补码转为原码后：正数的补码转为原码不变，负数的补码转为原码是对补码（除符号位）逐位取反后，并在最低位＋1。 即为00001100 +1 = -00001101 = -13

### << （左移）

将一个运算对象的各二进制位全部左移若干位右边补0。

例如：

1 << 1

1转为2进制是1，左移一位且右边补0就是10， 转为10进制就是2

1 << 2

1转为2进制是1，左移两位且右边补0就是100， 转为10进制就是4

1 << 3

1转为2进制是1，左移三位且右边补0就是1000， 转为10进制就是8

### >> （右移）

二进制右移运算符。将一个数的各二进制位全部右移若干位，正数左补0，负数左补1，右边丢弃。

3 << 1

3转为2进制是11，右移一位且左边补0就是1， 转为10进制就是1

3 << 2

3转为2进制是11，右移一位且左边补0就是0， 转为10进制就是0

## 关键区别点

### !与~的异同

!: 代表值得取反，对于整形变量，只要不为0，使用 ! 取反都是0，0取反就是1。就像 bool 只有真假一样。

~: 代表位的取反，对于整形变量，对每一个二进制位进行取反，0变1，1变0。

# 单片机代码实际运用

## 引入单片机头文件

#include<reg51.h>

#include<reg52.h>

这两个哪个都可以，52的功能更全一些

里边定义了P0、P1、P2、P3等各种参数，还定义了sfr 和sbit

//sfr     特殊功能寄存器的定义

//sbit   特殊位的定义

定义一个位置

sbit LED1 =P0^0;

^0 代表P0里边的第一个口，以此类推

以下这段算固定写法

void main(){

while(1){

}

}

main 是入口主方法

while(1) // while语句是一直去判断括号里的1是不是真，1等价于1=1，是真的，所有会永远不停的去循环

在里边 把状态值设为0即为点亮，1即为熄灭

LED1 = 0;

把之前的状态改为相反的状态

LED1 = ~LED1 ;

bit和sbit都是C51扩展的变量类型

## 16进制对应的状态

|  |  |  |
| --- | --- | --- |
| 16进制 | 状态 | 2进制 |
| 0x00 | 全亮 | 00000000 |
| 0xff | 全熄 | 11111111 |
| 其中一个亮，其余都熄灭 | | |
| 0xfe | 第一个亮，其它全熄 | 11111110 |
| 0xfd | 第二个亮，其它全熄 | 11111101 |
| 0xfb | 第三个亮，其它全熄 | 11111011 |
| 0xf7 | 第四个亮，其它全熄 | 11110111 |
| 0xef | 第五个亮，其它全熄 | 11101111 |
| 0xdf | 第六个亮，其它全熄 | 11011111 |
| 0xbf | 第七个亮，其它全熄 | 10111111 |
| 0x7f | 第八个亮，其它全熄 | 01111111 |
| 其中一个熄灭，其余都亮 | | |
| 0xfe | 第一个亮，其它全熄 | 11111110 |
| 0xfd | 第二个亮，其它全熄 | 11111101 |
| 0xfb | 第三个亮，其它全熄 | 11111011 |
| 0xf7 | 第四个亮，其它全熄 | 11110111 |
| 0xef | 第五个亮，其它全熄 | 11101111 |
| 0xdf | 第六个亮，其它全熄 | 11011111 |
| 0xbf | 第七个亮，其它全熄 | 10111111 |
| 0x7f | 第八个亮，其它全熄 | 01111111 |

## [\_crol\_ 与 \_cror\_ 的用法](http://www.baidu.com/link?url=Xxe6JyKAppSCnHpenmVmgZdfgugJ8RcEYobxDW3zt-YU-VrugZSV6veOOrrr8xErUzcQ99LMBMd7AiHKOsC1X-b6e8GhgNh7QPfv9oQ2Ilm" \t "https://www.baidu.com/_blank)（0的移动）

LED=0xfe //11111110 11111101

LED=\_cror\_(LED,1);

[\_crol\_” 与“\_cror\_”的用法](http://www.baidu.com/link?url=Xxe6JyKAppSCnHpenmVmgZdfgugJ8RcEYobxDW3zt-YU-VrugZSV6veOOrrr8xErUzcQ99LMBMd7AiHKOsC1X-b6e8GhgNh7QPfv9oQ2Ilm" \t "https://www.baidu.com/_blank)

## 数码管的显示值

0~9

#define uchar unsigned char

共阴

const uchar numCode[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

共阳

const uchar numCode[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};

使用时 numCode[X]

## 消抖

If(0 == P0^0){

Delay\_ms(10)

If(0 == P0^0){

// 在这里边写实际的逻辑代码

}

}

## 中断系统

INT0：外部中断0，由P3．2端口线引入，低电平或下跳沿引起。

INT1：外部中断1，由P3．3端口线引入，低电平或下跳沿引起。

T0：定时器／计数器0中断，由T0计满回零引起。

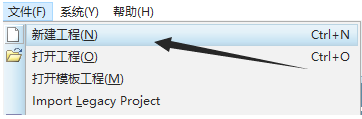
T1：定时器／计数器l中断，由T1计满回零引起。

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。

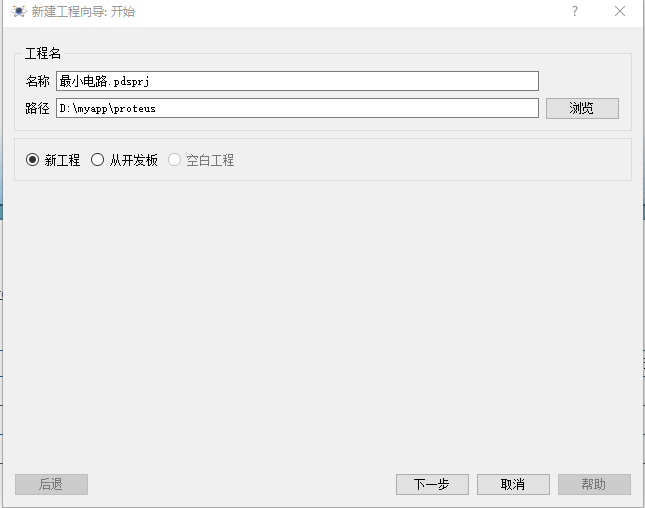
# Proteus

## 如何创建

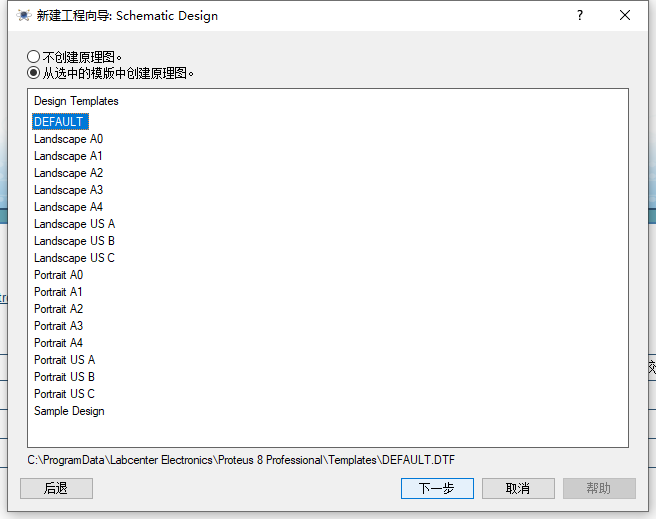
1. 文件---->新建



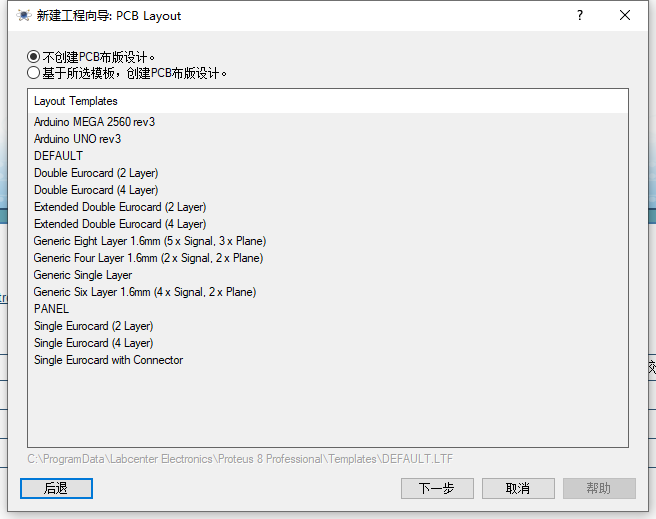
1. 输入名称，选择路径



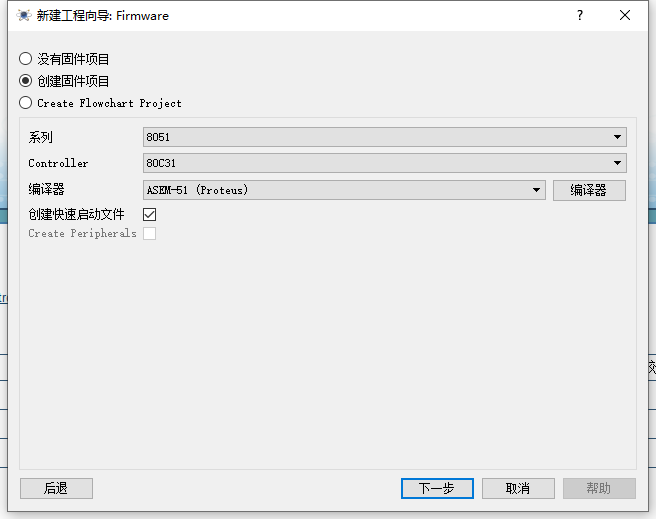
1. 选择默认,点击下一步



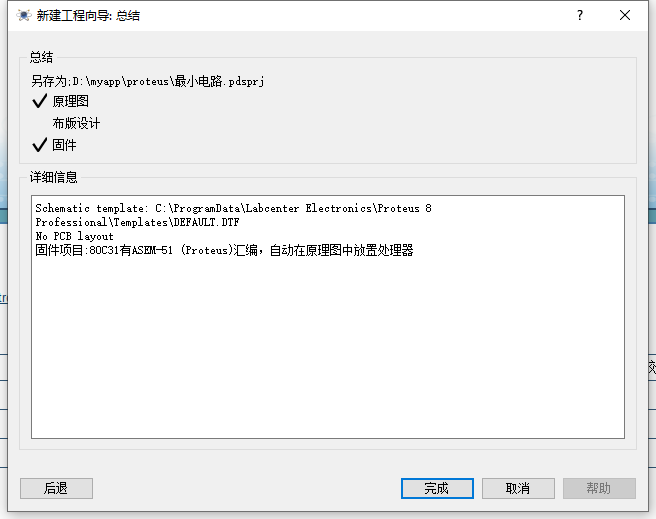
1. 不选,点击下一步



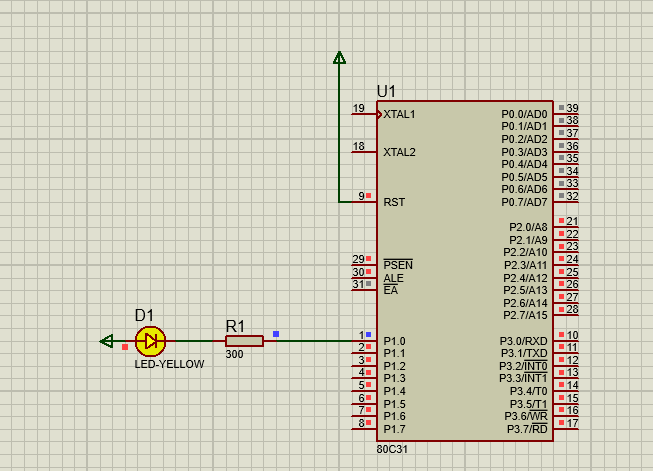
1. 选择创建固件,点击下一步

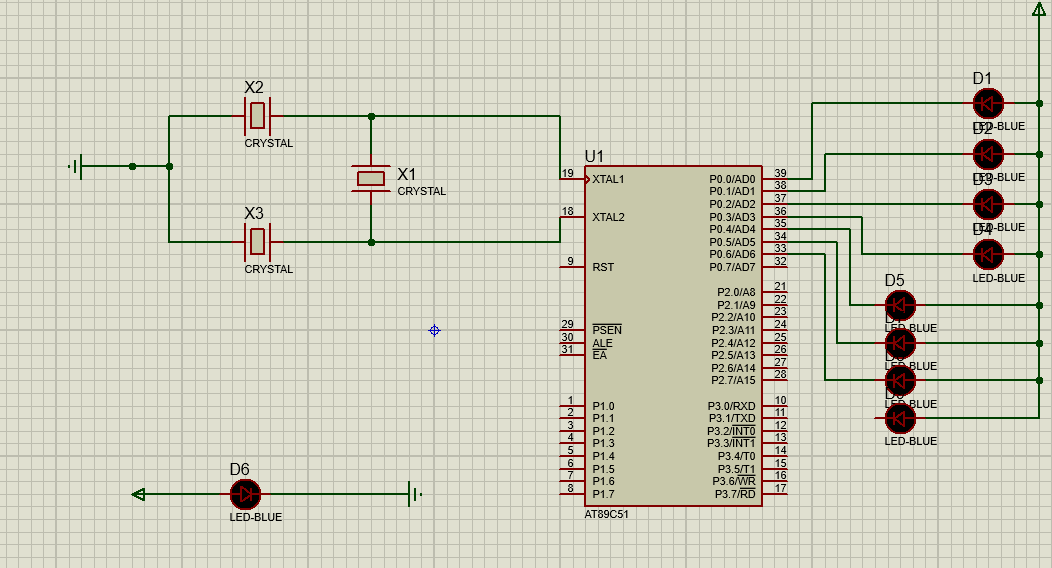


1. 点击完成



## 最小电路





## 元件英文

终端模式下：

电源正级：POWER

接地线：GROUND

原件模式下：（点P可搜索）

晶振：CRYSTAL  
电容：CAP

电阻：RES  
电解电容：CAP-ELEC

LED灯： LED-BLUE

一位共阳数码管：7SEG-MPX1-CA

一位共阴数码管：7SEG-MPX1-CC

蜂鸣器：BUZZER

## 动态数码管

0X06 代表1 110 （不够7位的在前边补0 这里就成了 0000110）

0x5b 代表2 1011011

0x4f 代表3 1001111

0X66 代表4 1100110  
0x6d 代表5 1101101

0x7d 代表6 1111101

0x07 代表7 111 （不够7位的在前边补0 这里就成了 0000111）

0x7f 代表8 1111111

0x6f 代表9 1101111

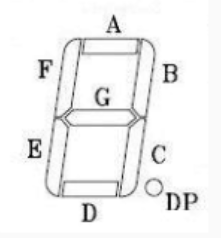
数码管bai dp g f e d c b a （这个就意味着 是从二进制倒着算过来）

数码管分动态和静态的，这里演示的是动态的

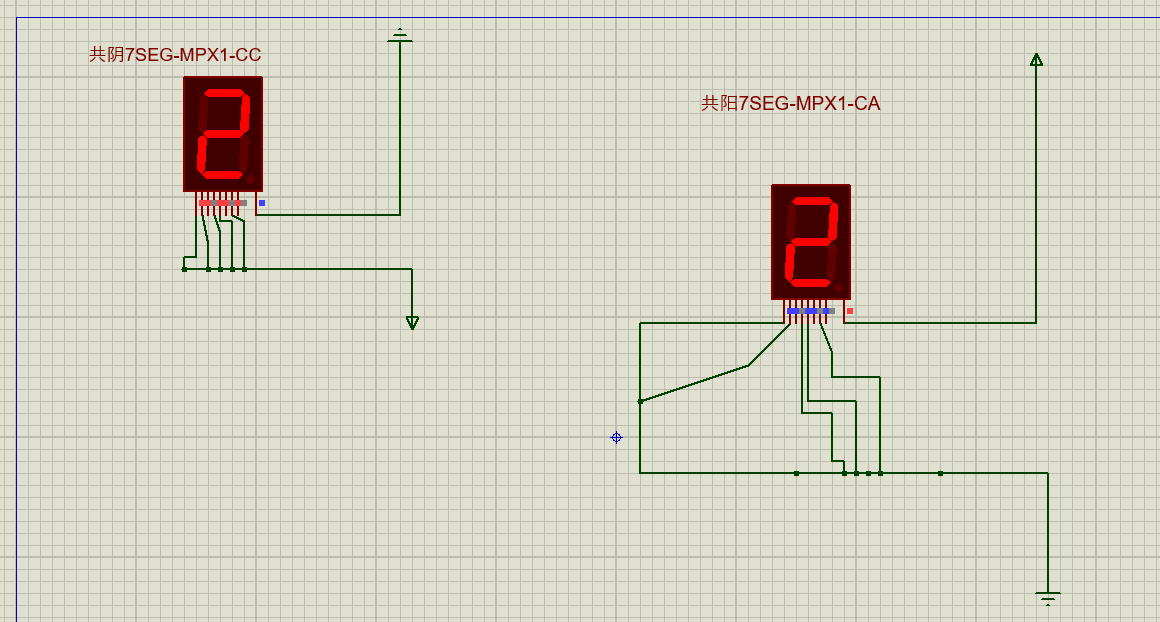
### 1位数码管

一位共阳数码管：7SEG-MPX1-CA

一位共阴数码管：7SEG-MPX1-CC



总结：单独的一头，接正级的是共阳，接负级的是共阴



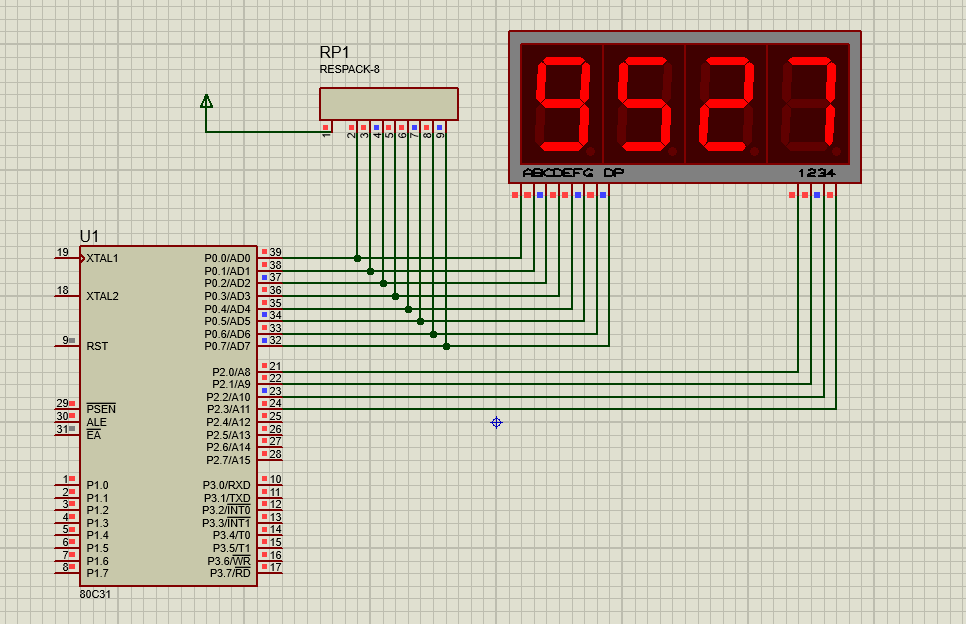
### 2位数码管

二位共阳数码管：7SEG-MPX2-CA

二位共阴数码管：7SEG-MPX2-CC

接法与实现方法与4位数码管相同

### 4位数码管



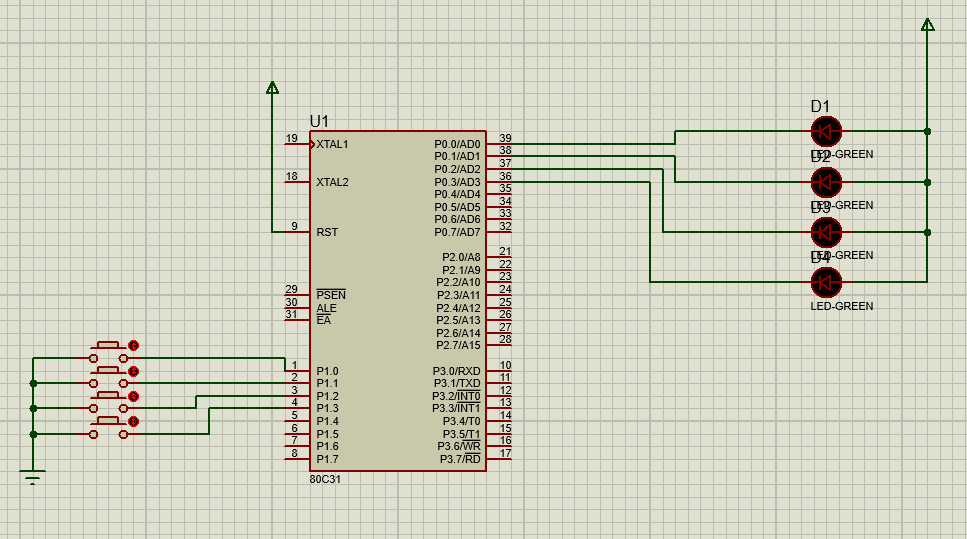
RESPACK-8 排阻为必须的

示例代码

#include<reg51.h> //?????  
#define uchar unsigned char //???  
#define uint unsigned int  
  
void delay(uint ms) //???????  
{  
uint i,j;  
for(i=ms;i>0;i--) //????ms??????????   
 for(j=110;j>0;j--);   
}  
  
void main() //???  
{  
while(1)  
{  
 P2 = 0Xfe; // 1111 1110  
 P0 = 0x6f; // 0000 0110 ??1  
 delay(10);  
  
 P2 = 0Xfd; // 1111 1101  
 P0 = 0x6d; // ??2  
 delay(10);   
  
 P2 = 0Xfb; // 1111 1011  
 P0 = 0x5b; //??3  
 delay(10);  
   
 P2 = 0Xf7; // 1111 0111  
 P0 = 0x07; //??4  
 delay(10);   
   
 }  
}

## 按键

### 独立按键控制独立led



示例代码

#include <reg51.h>

typedef unsigned char uint8;

typedef unsigned int uint16;

void delay(uint16 x)

{

uint16 i,j;

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

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

}

void Move\_LED()

{

if ((P1 & 0x01) == 0x00) P0 ^= 0x01;

else if((P1 & 0x02) == 0x00) P0 ^= 0x02;

else if((P1 & 0x04) == 0x00) P0 ^= 0x04;

else if((P1 & 0x08) == 0x00) P0 ^= 0x08;

}

void main()

{

uint8 Key = 0x00;

while(1)

{

if(P1 != Key)

{

Key = P1;

Move\_LED();

delay(10);

}

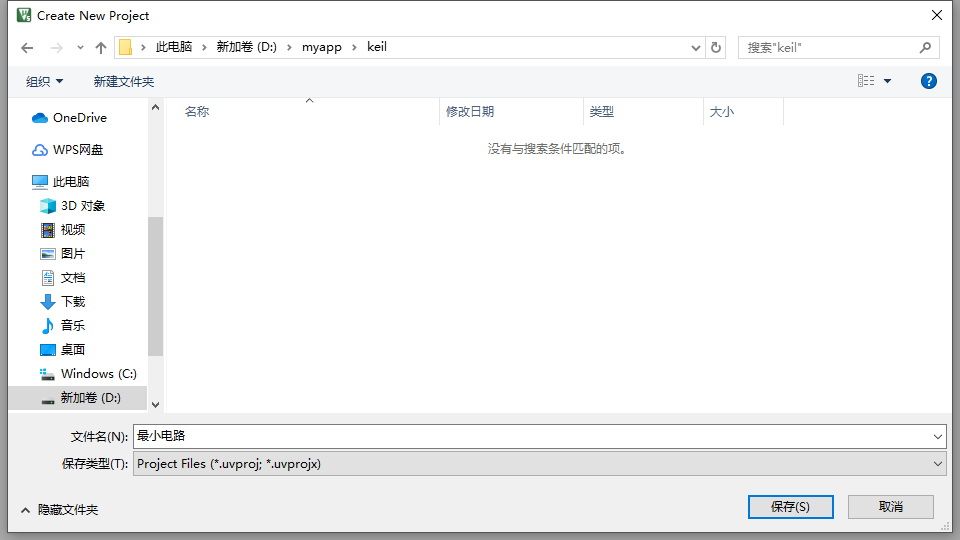
}

}

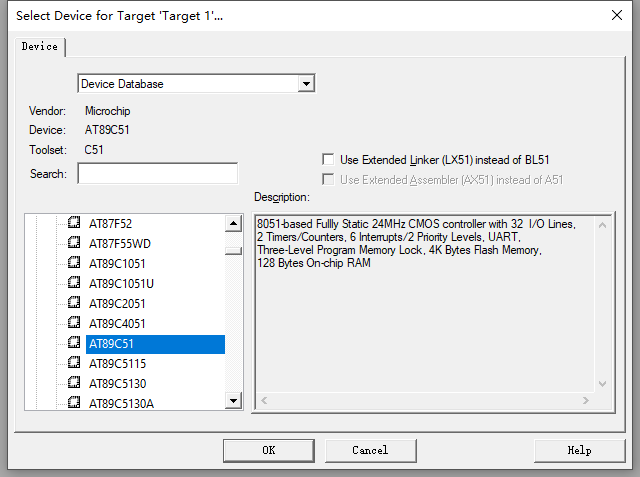
# Keil

## 如何创建

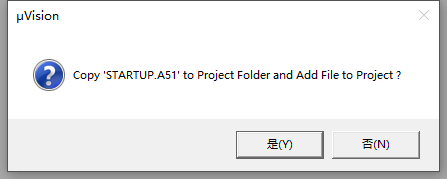
1. Project--new project新建一个项目

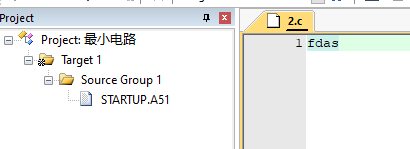


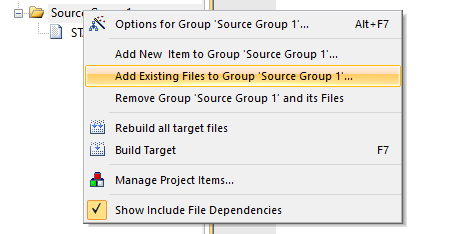
1. 点开Device Database选中 Microchip选中AT89C51,点ok



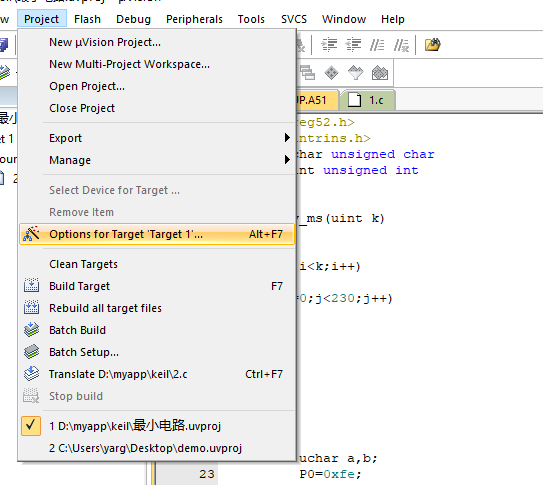
1. 点是

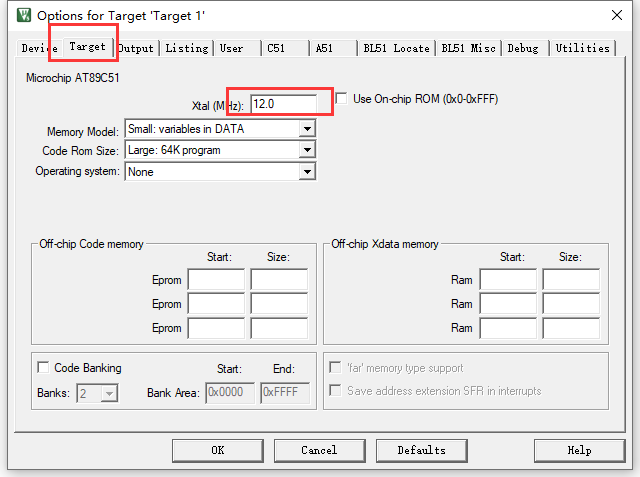


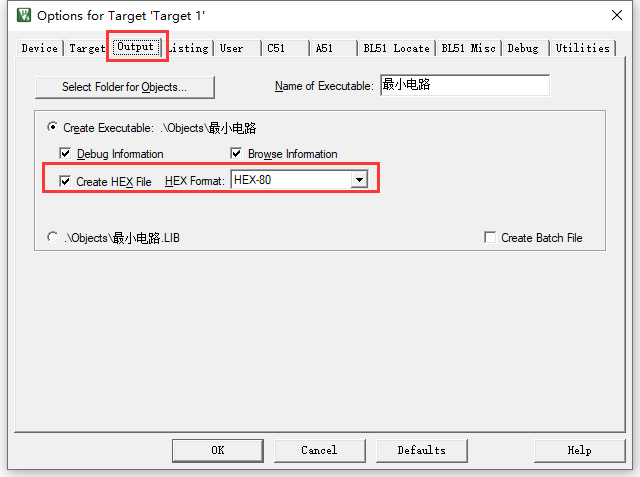
1. File-new 新建一个文件 保存为2.c  
   
2. 对着Source Group1 点右键，第3个，把2.c加入到这个资源下

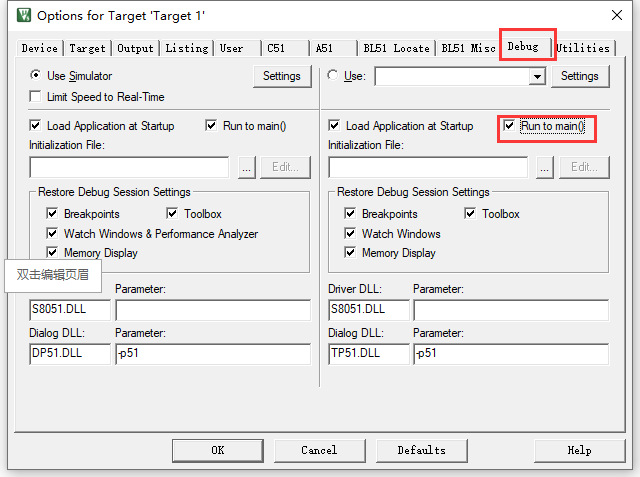


1. 编辑.c文件
2. 配置生成hey









1. keil自带的调试