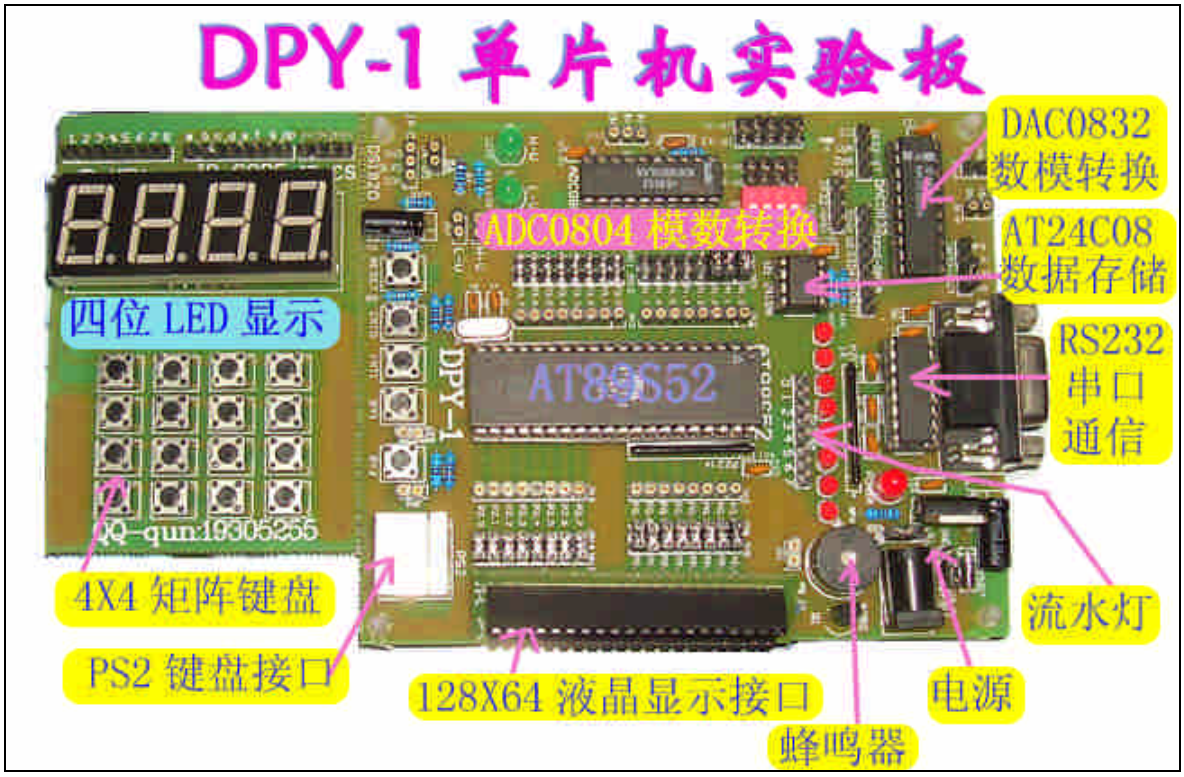
DPY-1 单片机实验板各功能模块应用与源程序



目录

欢迎加入 QQ 群：19305255

Keil 软件的使用

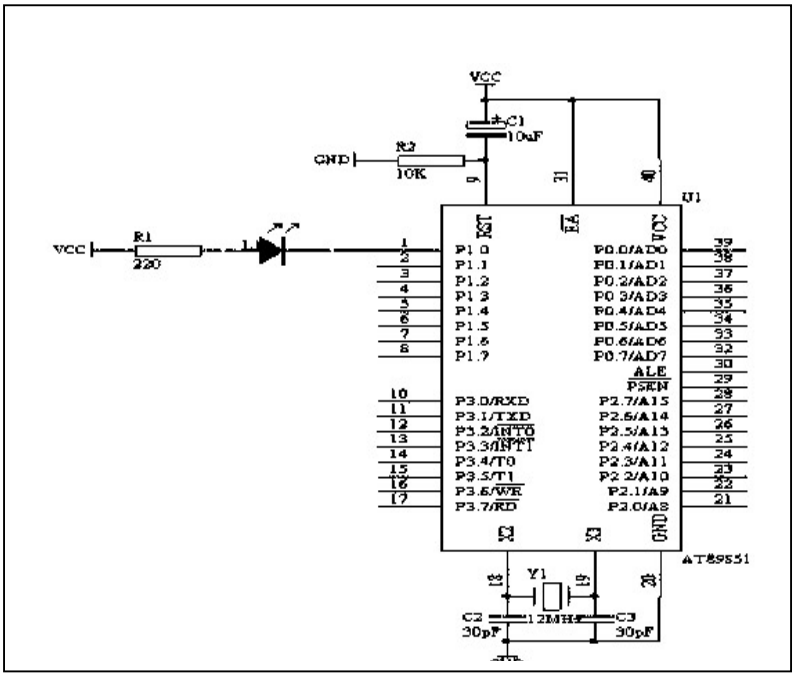
随着单片机开发技术的不断发展，目前已有越来越多的人从普遍使用汇编语言到逐渐使用高级语言开发，其中主要是以 C 语言为主，市场上几种常见的单片机均有其 C 语言开发环境。

这里以最为流行的 80C51 单片机为例来学习单片机的 C 语言编程技术。大家都有 C 语言基础,但是编单片机程序,大家还得找专门的书来学习一下。这里我们只介绍 Keil 这种工具软件的用法。

学习一种编程语言，最重要的是建立一个练习环境，边学边练才能学好。Keil 软件是目最流行开发 80C51 系列单片机的软件，Keil 提供了包括 C 编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器等在内的完整开发方案，通过一个集成开发环境（μVision）将这些部份组合在一起。

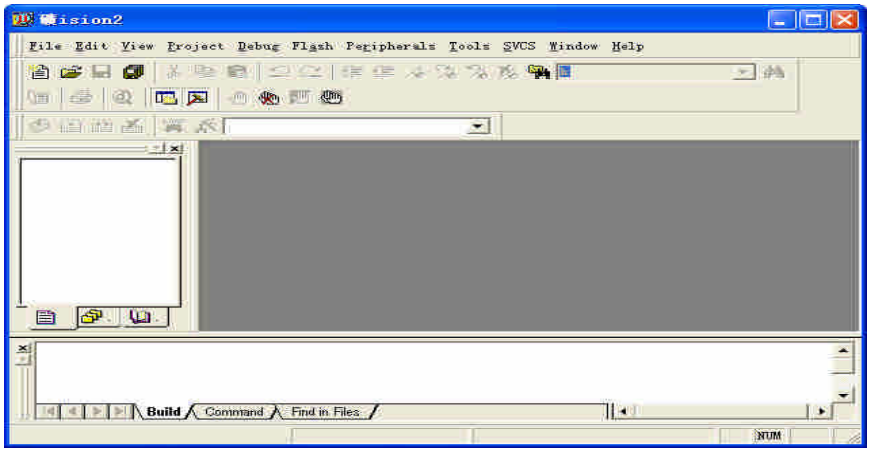
下面我以一个实验举一个例子，一步一步学习 Keil 软件的使用。

首先我们看硬件原理图：

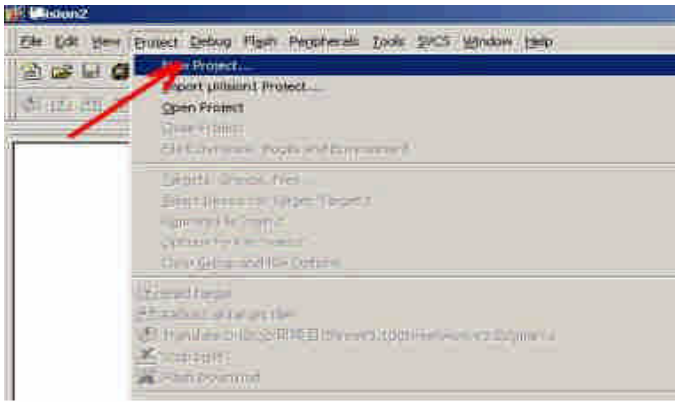


很明显，要点亮使发光二极管，必须使单片机的 I/O 口 P1.0 输出低电平。于是我们的任务就是编程序使 P1.0 输出地电平。

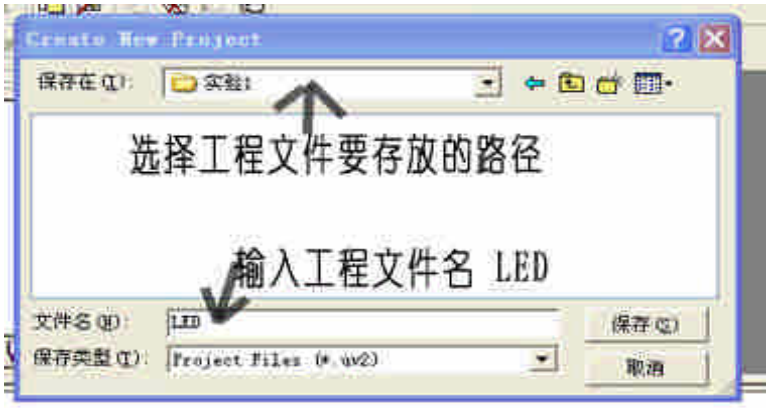
1. 使用 Keil 前必须先安装。安装过程简单，这里不在叙述。
2. 安装好了 Keil 软件以后，我们打开它。打开以后界面如下：



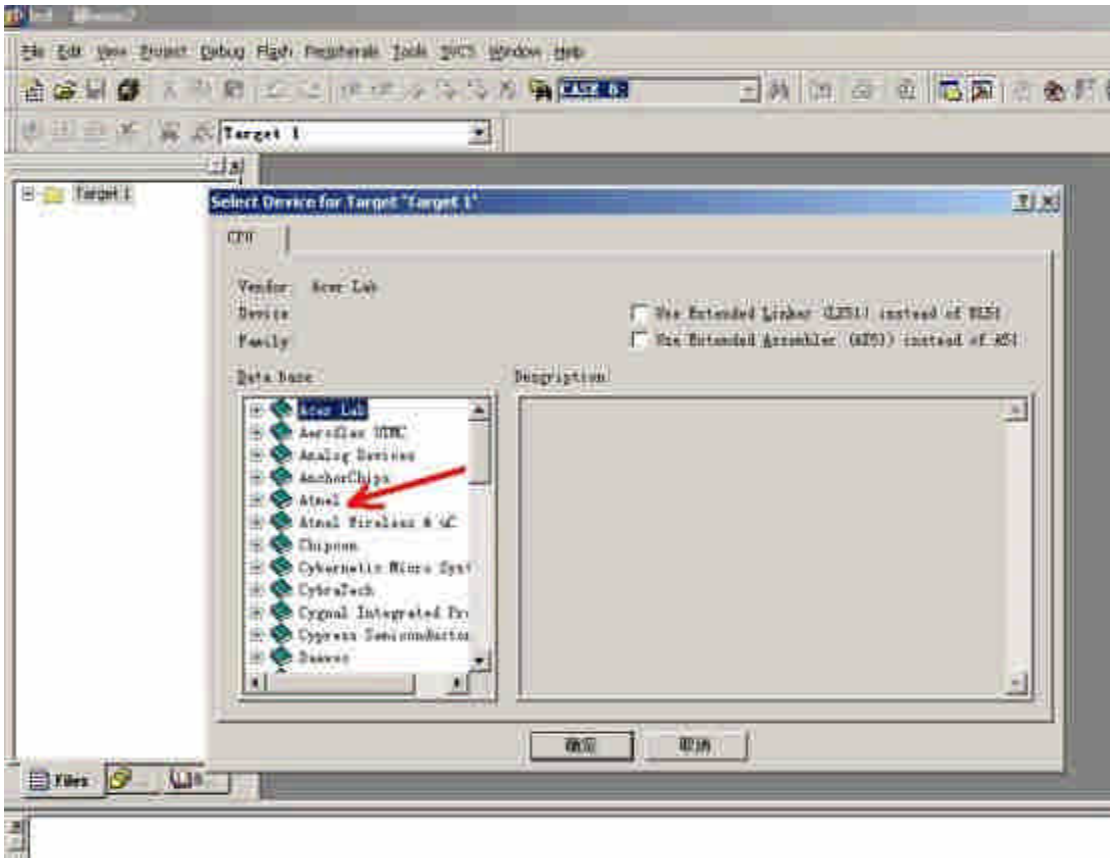
3. 我们先新建一个工程文件，点击“Project->New Project…”菜单，如下图，：



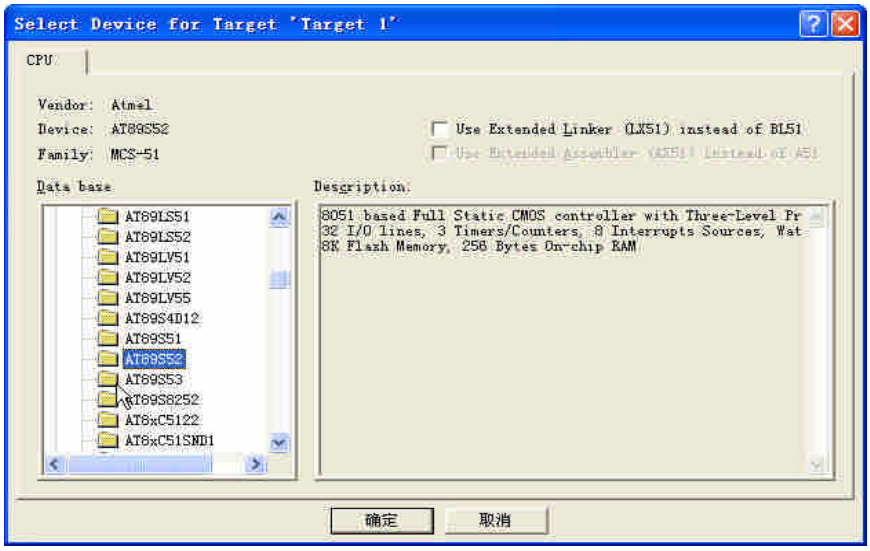
3. 选择工程文件要存放的路径 ,输入工程文件名 LED, 最后单击保存.



4. 在弹出的对话框中选择 CPU 厂商及型号

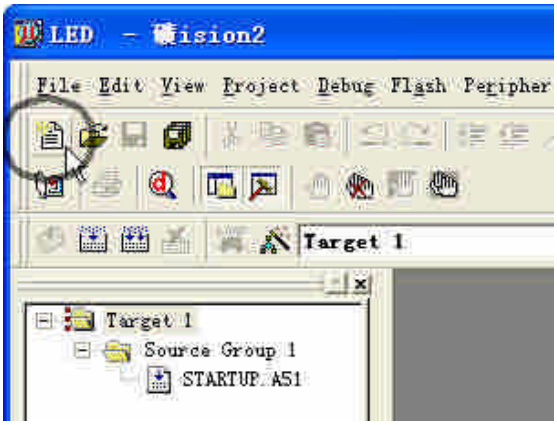


5. 选择好 Atmel 公司的 AT89S52 后，单击确定

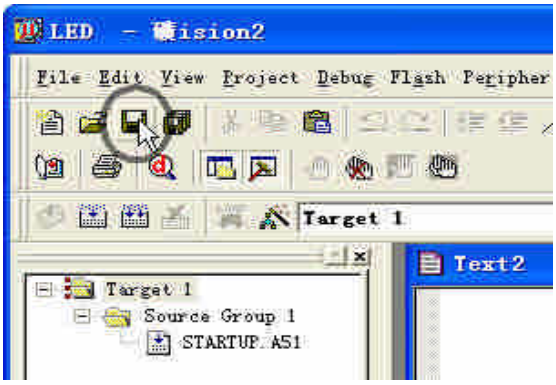


1．在接着出现的对话框中选择“是”。

5. 新建一个 C51 文件，单击左上角的 New File 如下图所示：



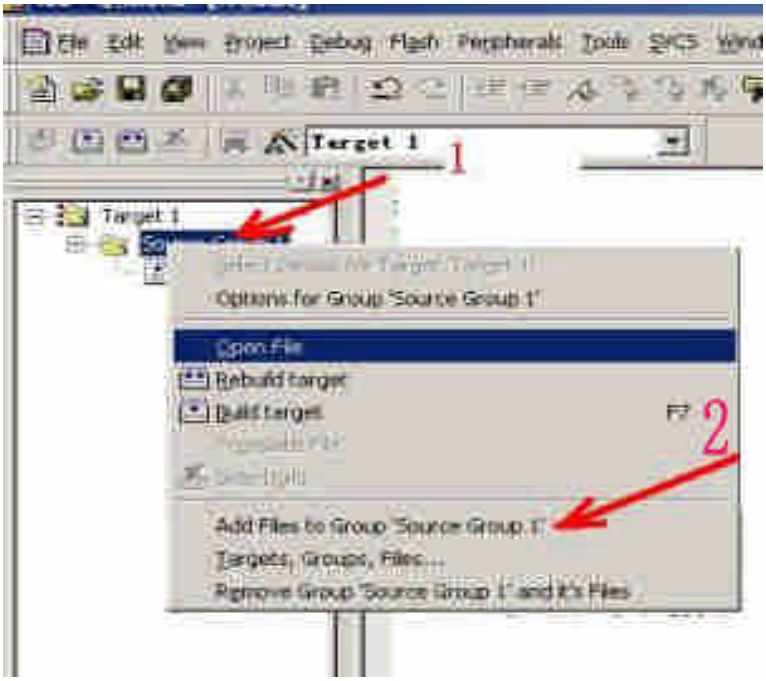
6. 保存新建的文件，单击 SAVE 如下图：



2．在出现的对话框中输入保存文件名 MAIN.C（注意后缀名必须为.C），再单击

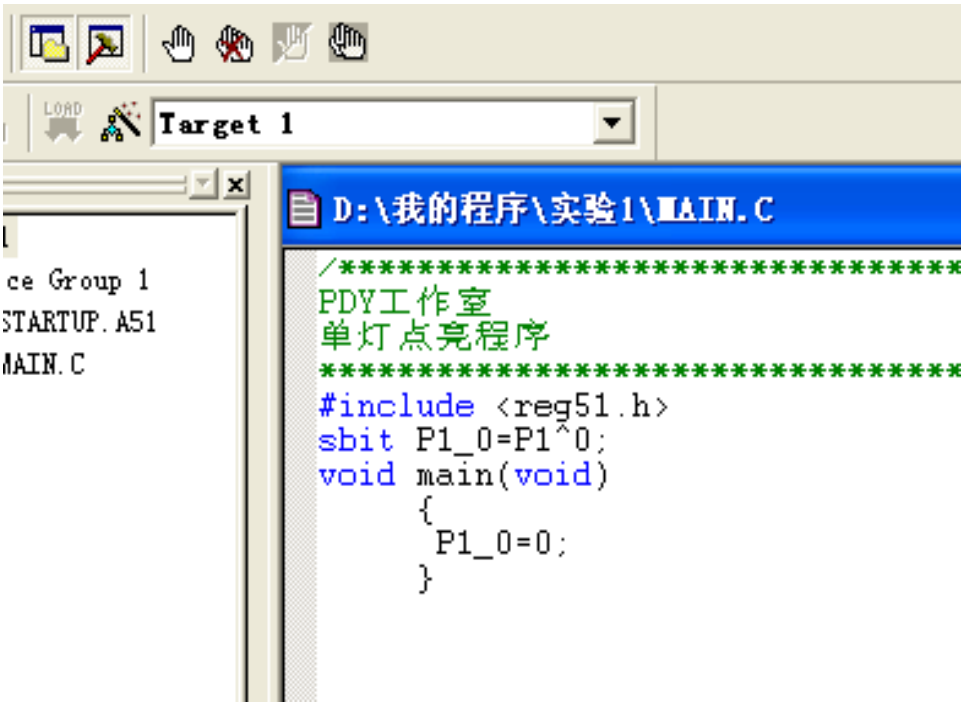
“保存”，如下图；

7. 保存好后把此文件加入到工程中方法如下 ：用鼠标在 Source Group1 上单击右键 ，然后再单击 Add Files to Group ‘Source Group 1' 如下图：

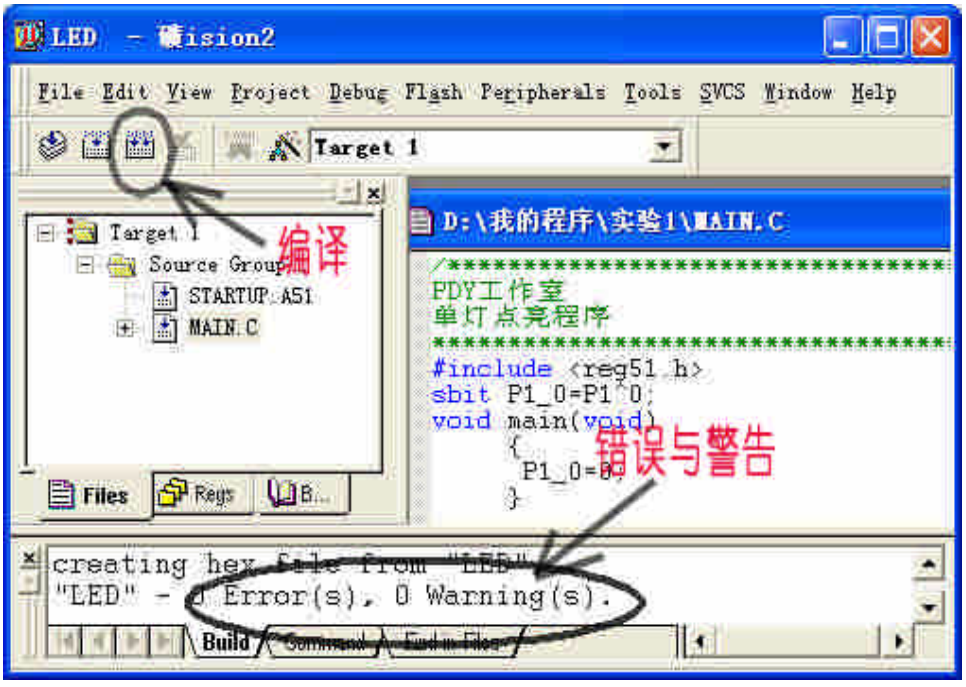


8. 选择要加入的文件 ， 找到 MAIN.C 后 ， 单击 Add, 然后单击 Close

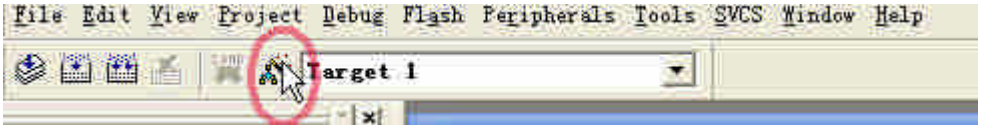
9. 在编辑框里输入如下代码 ：



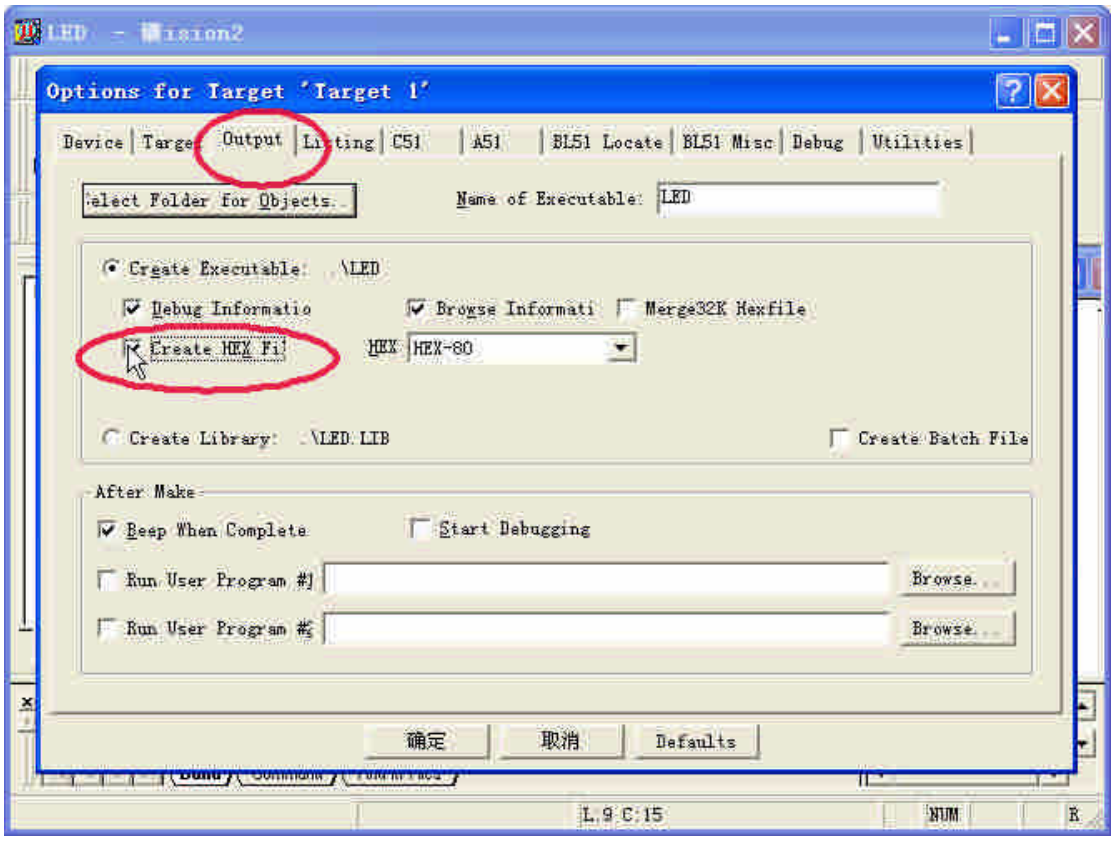
10. 到此我们完成了工程项目的建立以及文件加入工程 ， 现在我们开始编译工程如下图所示 ： 我们先单击编译， 如果在错误与警告处看到 0 Error(s) 表示编译通过 ；



11. 生成 .hex 烧写文件,先单击 Options for Target,如图;



12. 在下图中,我们单击 Output, 选中 Create HEX F。再单击“确定”。



13. 打开文件夹‘实验 1’，查看是否生成了 HEX 文件。如果没有生成，在执行一遍步骤 10 到步骤 12，直到生成。

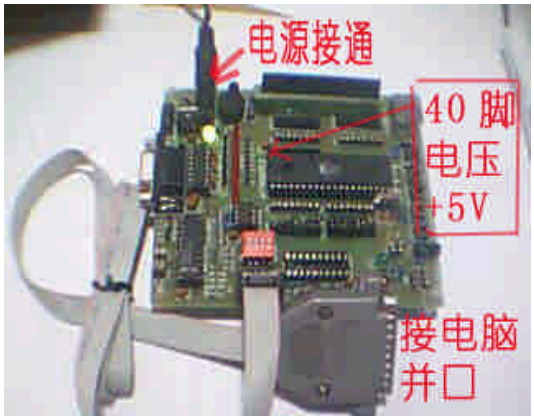
以上是 Keil 软件的基本应用,更多的高级应用请大家去查找资料.

以下将介绍的是如何将 HEX 文件下载到单片机里面。我们用的下载软件是 Easy 51Pro

Easy51Pro v2.0 软件的使用

Easy51Pro v2.0是单片机下载工具: 在我们的QQ群里有这个软件，大家可以下载来直接用，而不用安装。下面我给大家简单的说一下用法:

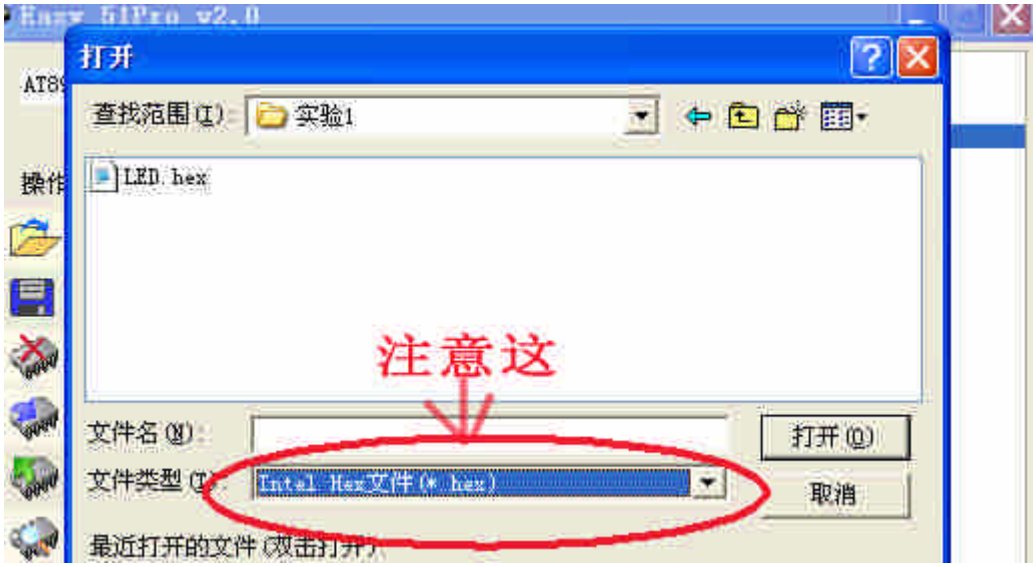
1. **硬件连接；**必须满足下面的每一个条件才能下载。如图



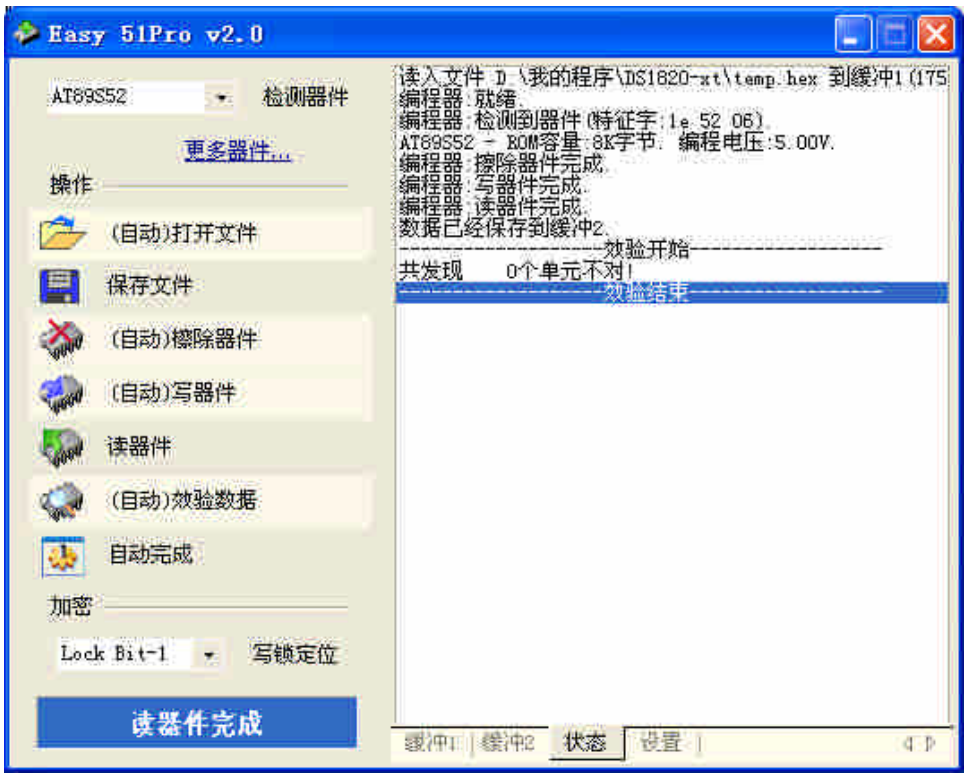
2. 打开软件；

3. 检测器件，注意右边的信息提示。如果没有检测到器件，检查硬件连接。

3. 打开 HEX 文件。先单击“（自动）打开文件”。在“实验 1”的文件夹里找到文件 LED.hex 文件.



4. 最后单击“自动完成”，OK 大功告成。



5. 大功既然告成，就要看到效果。现在我们在实验板的左下角找到 JK7，用导线连到 P1.0 上，LED2 被点亮了把。是不是很有成就感。

如果以上的实验你觉得是小儿科，太简单了。那么请关注以下的内容。

DPY-1 单片机实验板各功能区的实验与应用。

一. 闪烁灯

[实验任务]

在 P1.0 端口上接一个发光二极管 L1，使 L1 在不停地一亮一灭，一亮一灭的时间间隔为 0.2 秒。

[硬件电路]

与上面点亮小灯的连接完全相同。

[C 语言源程序]｛ 我们提供的程序并非最优化程序，仅供学习参考。｝

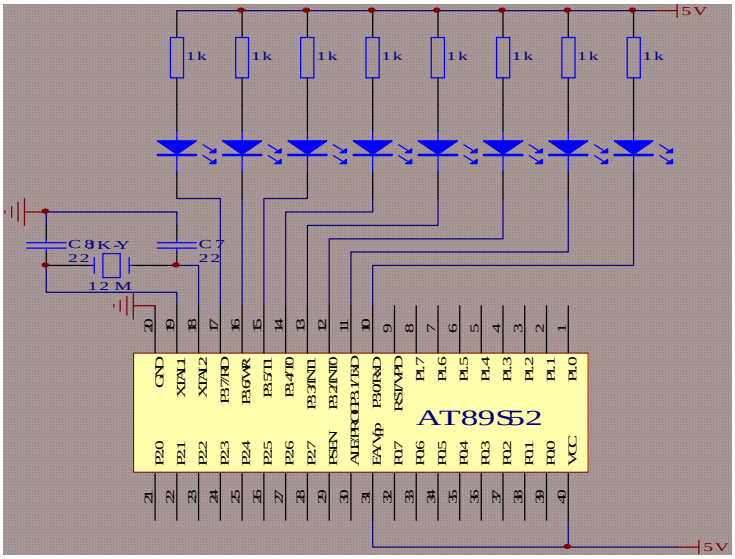
```
#include <AT89X51.H>
/*******************************************************************************
    第一行是一个"文件包含"处理。
  所谓"文件包含"是指一个文件将另外一个文件的内容全部包含进来，所以这里的程序虽然只有 4 行，
  但 C 编译器在处理的时候却要处理几十或几百行。这里程序中包含 REG51.h 文件的目的是为了要使用 P1
  这个符号，即通知 C 编译器，程序中所写的 P1 是指 80C51 单片机的 P1 端口而不是其它变量。
*******************************************************************************/
void delay02s(void)                    //延时 0.2 秒子程序
        {
          unsigned char i,j,k;   //定义 3 个无符号字符型数据。
          for(i=20;i>0;i--)      //作循环延时
          for(j=20;j>0;j--)
          for(k=248;k>0;k--);
         }
void main(void)    //每一个 C 语言程序有且只有一个主函数，
{
  while(1)    //循环条件永远为真，以下程序一直执行下去。
  {
   P1_0=0;       // I/O 口 P1.0 输出低电平，小灯被点亮。
   delay02s();   //延时经过 0.2 秒。
   P1_0=1;       // I/O 口 P1.0 输出高电平，小灯熄灭。
   delay02s();   //延时经过 0.2 秒。
   }
}
```

二. 广告灯的左移右移

[实验任务]

做广告灯的左移右移，八个发光二极管分别接在单片机的 P3.0—P3.7 接口上，输出“0”时，发光二极管亮，开始时 P3.0→P3.1→P3.2→P3.3→┅→P3.7→P3.6→┅→P3.0 亮，重复循环。

[硬件电路]



[DPY-1 实验板连接]

用 8 芯排线把 JP-LSH 连接到 JP12 上。

[C 语言源程序]

```
#include <AT89X52.H>
unsigned char i;
unsigned char temp;
unsigned char a,b;
void delay(void)     //延时子程序
  {
    unsigned char m,n,s;
    for(m=20;m>0;m--)
    for(n=20;n>0;n--)
    for(s=248;s>0;s--);
   }
 void main(void)     //主程序
   {
      while(1)       //循环条件永远为真，以下程序一直执行下去。
       {
        temp=0xfe;
        P3=temp; //直接对 I/O 口 P3 赋值，使.P3.0 输出低点平。
        delay();   //延时
         for(i=1;i<8;i++) //实现广告灯的从右到左移动（以原理图为准）
             {
               a=temp<<i;
               b=temp>>(8-i);
               P3=a|b;
               delay();
             }
          for(i=1;i<8;i++) //实现广告灯的从左到右移动
             {
```

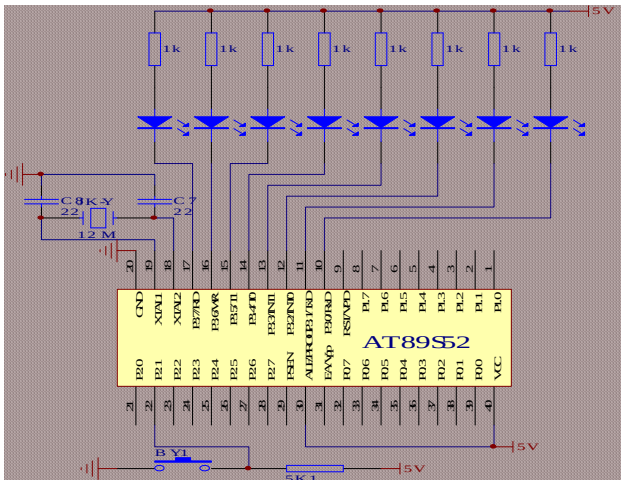
```
            a=temp>>i;
            b=temp<<(8-i);
            P3=a|b;
            delay();
            }
        }
    }
```

三. 按键识别

[实验任务]

通过按下一次按键，使广告灯向左移动一位，直到最后一位，在向右移动。

[硬件电路]



（大家注意到这一个电路图与上一个实验的电路图只多了一个按键和一个 5.1K 的电阻。）

[DPY-1 实验板连接]

用 8 芯排线把 JP-LSH 连接到 JP12 上。用一根导线把 JK5 接到 P2.1 上。

[实验原理]

从图中可以看出 P2.1 在按键没有按下时接的是高电平，按键按下时接的就是低电平了。所以我们只要判断 P2.1 的电平就可以知道按键是否被按下了。

而在按键按下的过程中，由于抖动，将产生干扰，在按下的过程中，一旦有干扰过来，可能造成误触发过程，这并不是我们所想要的。因此在按键按下的时候，要把我们手上的干扰信号以及按键的机械接触等干扰信号给滤除掉，一般情况下，我们可以采用软件滤波的方法去除这些干扰信号，一般情况下，一个按键按下的时候，总是在按下的时刻存在着一定的干扰信号，按下之后就基本上进入了稳定的状态。我们在程序设计时，从按键被识别按下之后，延时 5ms 以上，从而避开了干扰信号区域，我们再来检测一次，看按键是否真得已经按下，若真得已经按下，这时肯定输出为低电平，若这时检测到的是高电平，证明刚才是由于干扰信号引起的误触发，CPU 就认为是误触发信号而舍弃这次的按键识别过程。

[C 语言源程序]

```
#include <reg52.h>
sbit BY1=P2^1;          //定义按键的输入端
unsigned char count; //按键计数,每按一下,count 加 1
unsigned char temp;
unsigned char a,b;
```

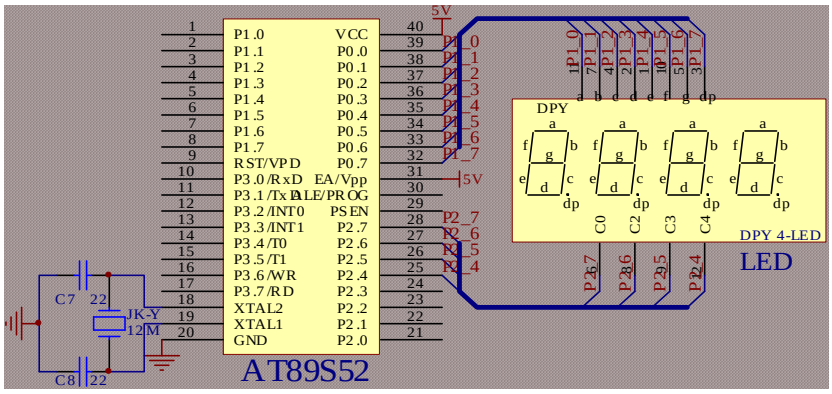
```
void delay10ms(void) //延时程序
{
   unsigned char i,j;
   for(i=20;i>0;i--)
   for(j=248;j>0;j--);
}
key()                  //按键判断程序
{
   if(BY1==0)   //判断是否按下键盘
   {
      delay10ms(); //延时,软件去干扰
      if(BY1==0)     //确认按键按下
       {
       count++;        //按键计数加 1
       if(count==8) //计 8 次重新计数
       {   count=0;   }     //将 count 清零
    }
   while(BY1==0);//按键锁定,每按一次 count 只加 1.
   }
}
 move()               //广告灯向左移动移动函数
{
   a=temp<<count;
   b=temp>>(8-count);
   P3=a|b;
}
main()
{
 count=0;   //初始华参数设置
 temp=0xfe;
 P3=0xff;
 P3=temp;
 while(1)   //永远循环,扫描判断按键是否按下
 {
    key();       //调用按键识别函数
    move();     //调用广告灯移动函数
   }
}
```

四. 数码管动态显示

[实验任务]

P0 端口接动态数码管的字形码笔段，P2 端口接动态数码管的数位选择端，动态显示“1234”字样；

[硬件电路]



注意：在 P0 口还有 1K 的排阻作为上拉电阻，在以后的电路原理图中都是如此，请大家不要忽略了

[DPY-1 实验板连接]

用排线把 JP-CODE 连到 JP8 是，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3……把 JP-CS 连到 JP14 上,注意:4H 接 P2.4;3H 接 P2.5;2H 接 P2.6;1H 接 P2.7;

[实验原理]

七段 LED 显示器内部由七个条形发光二极管和一个小圆点发光二极管组成，根据各管的极管的接线形式，可分成共阴极型和共阳极型。

LED 数码管的 a-dp 七个发光二极管因以不同亮暗的组合就能形成不同的字形，这种组合称之为字形码，下面给出共阴极的字形码

“0”3FH “1”06H “2”5BH “3”4FH
“4”66H “5”6DH “6”7DH “7”07H
“8”7FH “9”6FH “A”77H “b”7CH
“C”39H “d”5EH “E”79H “F”71H

由于显示的数字 0－9 的字形码没有规律可循，只能采用查表的方式来完成我们所需的要求了。这样我们按着数字 0－9 的顺序，把每个数字的笔段代码按顺序排好！建立的表格如下所示：TABLE DB 3FH，06H，5BH，4FH，66H，6DH，7DH，07H，7FH，6FH

动态接口采用各数码管循环轮流显示的方法，当循环显示频率较高时，利用人眼的暂留特性，看不出闪烁显示现象，这种显示需要一个接口完成字形码的输出（字形选择），另一接口完成各数码管的轮流点亮（数位选择）。

[C 语言源程序]

```
#include <reg52.h>
code unsigned char seg7code[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,
                                 0x7d,0x07,0x7f,0x6f}; //显示段码
void Delay(unsigned int tc)       //延时程序
{
    while( tc != 0 )              //如果 tc 为 0 则终止延时
    {
        unsigned int i;           //局部正整数变量 i
        for(i=0; i<100; i++); //执行 400 次将耗时 1 毫秒
        tc--;                     //tc 计数减一
    }
}
void Led(int date)                //显示函数
{
```

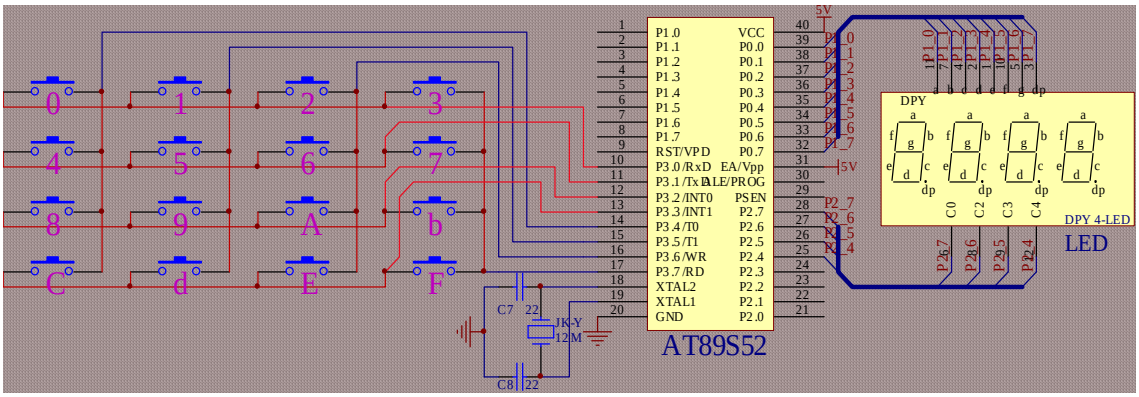
```
    P2=P2&0x7f;                    //P2.7 输出低电平，选通千位数
    P0=seg7code[date/1000]; //取出千位数，查表，输出。
    Delay(8);                    //延时
    P2=P2|0xf0;                   //销隐
     P2=P2&0xbf;                      //P2.6 输出低电平，选通百位数
    P0=seg7code[date%1000/100]; //取出百位数，查表，输出。
     Delay(8);                    //延时
     P2=P2|0xf0;                    //销隐
     P2=P2&0xdf;                     //P2.5 输出低电平，选通十位数
    P0=seg7code[date%100/10];      //取出十位数，查表，输出。
     Delay(8);                    //延时
     P2=P2|0xf0;                    //销隐
     P2=P2&0xef;
    P0=seg7code[date%10];           //取出个位数，查表，输出。
    Delay(8);
    P2=P2|0xf0;
}
main()
{
   int display_date=1234; //定义并赋值要显示的数据
   while(1)
   {
    Led(display_date);//调用显示函数显示数据 display_date
   }
}
```

五. 4×4 矩阵式键盘识别

[实验任务]

用 AT89S51 的并行口 P3 接 4×4 矩阵键盘，以 P3.0—P3.3 作输入线，以 P3.4—P3.7 作输出线；在每一个数码管上显示每个按键的“0—F”序号。

[硬件电路]



[DPY-1 实验板连接]

用排线把 JP-CODE 连到 JP8 是，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3……把 JP-CS 连到 JP14 上,注意:4H 接 P2.4;3H 接 P2.5;2H 接 P2.6;1H 接 P2.7;用排

线把 JP-KEY 连到 JP12 上，注意 1，2，3，4，5，6，7，8，分别对应 P3.0，P3.1，P3.3，P3.4……

[实验原理]

每个按键有它的行值和列值 ，行值和列值的组合就是识别这个按键的编码。矩阵的行线和列线分别通过两并行接口和 CPU 通信。键盘处理程序的任务是：确定有无键按下，判断哪一个键按下，键的功能是什么；还要消除按键在闭合或断开时的抖动。两个并行口中，一个输出扫描码，使按键逐行动态接地，另一个并行口输入按键状态，由行扫描值和回馈信号共同形成键编码而识别按键，通过软件查表，查出该键的功能。

[C 语言源程序]

```
#include <reg52.h>
unsigned char code seg7code[]={0x3f,0x06,0x5b,0x4f, 0x66,0x6d,0x7d,0x07,
                               0x7f,0x6f,0x77,0x7c, 0x39,0x5e,0x79,0x71};
unsigned char k;
void delay10ms(void) //延时程序
{
    unsigned char i,j;
    for(i=20;i>0;i--)
    for(j=248;j>0;j--);
}
 void Getch (  )
{   unsigned char X,Y,Z;
    P3=0xff;
    P3=0x0f;          //先对 P3 置数  行扫描
    if(P3!=0x0f)      //判断是否有键按下
    {delay10ms();     //延时,软件去干扰
     if(P3!=0x0f)    //确认按键按下 X = P3;
    {
    X=P3;             //保存行扫描时有键按下时状态
    P3=0xf0;          //列扫描
    Y=P3;             //保存列扫描时有键按下时状态
    Z=X|Y;            //取出键值
    switch ( Z )    //判断键值（那一个键按下）
    {
        case 0xee: k=0; break; //对键值赋值
        case 0xde: k=1; break;
        case 0xbe: k=2; break;
        case 0x7e: k=3; break;
        case 0xed: k=4; break;
        case 0xdd: k=5; break;
        case 0xbd: k=6; break;
        case 0x7d: k=7; break;
        case 0xeb: k=8; break;
        case 0xdb: k=9; break;
```

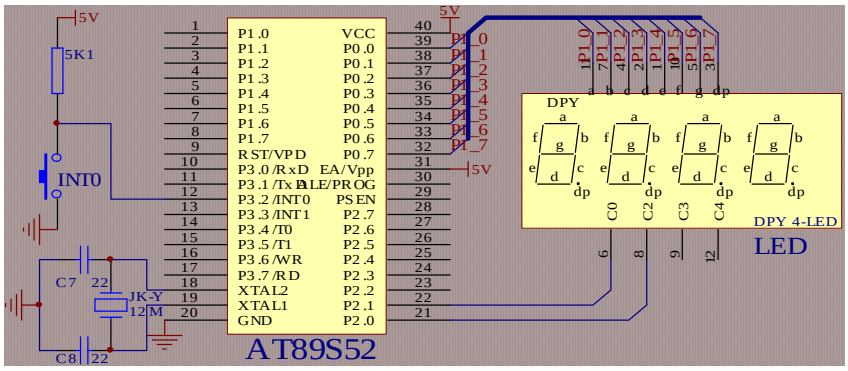
```
            case 0xbb: k=10;break;
            case 0x7b: k=11;break;
            case 0xe7: k=12;break;
            case 0xd7: k=13;break;
            case 0xb7: k=14;break;
            case 0x77: k=15;break;
    }       }       }       }  //请注意写程序时的格式规范，此处是为了节省纸张
void main(void)
{
   while(1)
  { P3=0xff;
     Getch();
     P0=seg7code[k];   //查表 LED 输出
     P2=0x0f;          //输出相同的四位数据。
   }
}
```

六．按键中断识别

[实验任务]

采用中断技术，每按一下按键，计数器加 1，并用 LED 显示出来。

[硬件电路]



注意：我们只用了 4 位数码管中的两位。

[DPY-1 实验板连接]

用排线把 JP-CODE 连到 JP8 上，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3……把 JP-CS 连到 JP14 上,注意： 2H 接 P2.0； 1H 接 P2.1；中断按键已经接好。

[实验原理]

以上的两个关于按键识别的实验的程序都是采用扫描的方式来实现的，CPU 的利用率比较低，在实时性要求高比较高，要求快速响应的场合不太实用。中断方式可以满足快速响应的要求。关于中断技术的具体内容，请大家参照教材。

[C 语言源程序]

```
#include<reg52.h>
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,
                            0x6d,0x7d,0x07,0x7f,0x6f};
unsigned char dispcount=0;          //计数
sbit gewei=P2^0;                    //个位选通定义
```

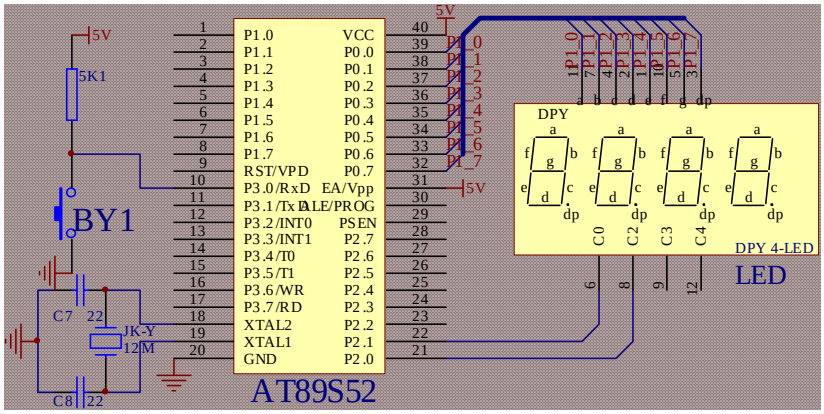
```
sbit shiwei=P2^1;                    //十位选通定义
void Delay(unsigned int tc)          //延时程序
{     while( tc != 0 )
      {
           unsigned int i;
           for(i=0; i<100; i++);
           tc--;
      }
}
void ExtInt0() interrupt 0       //中断服务程序
      {
        dispcount++;                  //每按一次中断按键，计数加一
        if  (dispcount==100)          //计数范围 0-99
           {dispcount=0;}
      }
void LED( )                           //LED 显示函数
{
   if(dispcount>=10)                 //显示两位数
   {
     shiwei=0;
     P0=table[dispcount/10];
     Delay(8);
     shiwei=1;
     gewei=0;
     P0=table[dispcount%10];
     Delay(5);
     gewei=1;
     }
     else                             //显示一位数
     {
     shiwei=1;
     gewei=0;
     P0=table[dispcount];
     Delay(8);
     }
}
void main()
{     TCON=0x01;   //中断设置
        IE=0x81;
        while(1)     //循环执行
        {
        LED();        //只须调用显示函数
        }
}
```

七. 定时器 T0 的应用---9.9 秒计时设计

［实验任务］

开始时，显示“00”，第 1 次按下 BY1 后就开始计时。第 2 次按 BY1 后，计时停止。 第 3 次按 BY1 后，计时归零。

［硬件电路］



注意这一个电路图与上一个的接法只是按键接的 I/O 口不一样。所以只要用导线把 JK5 接到 P3.0 上就可以了。

［DPY-1 实验板连接］

用排线把 JP-CODE 连到 JP8 上，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3……把 JP-CS 连到 JP14,注意： 2H 接 P2.0； 1H 接 P2.1； 用导线把 JK5 接到 P3.0 上。

［C 语言源程序］

```
#include <AT89X51.H>
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,
                            0x6d,0x7d,0x07, 0x7f,0x6f, };
unsigned char sec;            //定义计数值，每过 1/10 秒，sec 加一
unsigned char keycnt=0;
unsigned int tcnt;            //键值判断
sbit gewei=P2^0;              //个位选通定义
sbit shiwei=P2^1;             //十位选通定义
void Delay(unsigned int tc)   //延时程序
{
    while( tc != 0 )
    {
        unsigned int i;
        for(i=0; i<100; i++);
        tc--;
    }
}
void LED()                    //LED 显示函数
{
        shiwei=0;
       P0=table[sec/10];
       Delay(8);
       shiwei=1;
       gewei=0;
```

```
        P0=table[sec%10];
        Delay(5);
        gewei=1;
}
void KEY()              //按键扫描程序
{
unsigned char i,j;
if(P3_0==0)
  {
    for(i=20;i>0;i--)       //延时去干扰
    for(j=248;j>0;j--);
    if(P3_0==0)
      {
        keycnt++;
        switch(keycnt)   //按下次数判断
          {
            case 1:       //第一次按下
              TH0=0x06; //对 TH0 TL0 赋值
              TL0=0x06;
              TR0=1;      //开始定时
              break;
            case 2:       //第二次按下
              TR0=0;      //定时结束
              break;
            case 3:       //第三次按下
             keycnt=0;   //重新开始判断键值
             sec=0;       //计数重新从零开始
              break;
          }
        while(P3_0==0);
      }   }   }    //请注意写程序时的格式规范，此处是为了节省纸张
void t0(void) interrupt 1 using 0   //定时中断服务函数
{
tcnt++;             //每过 250ust tcnt 加一
if(tcnt==400)   //计满 400 次（1/10 秒）时
  {
    tcnt=0;    //重新再计
    sec++;
    if(sec==100) //定时 10 秒，在从零开始计时
      {
        sec=0;
      }
  }
}
```

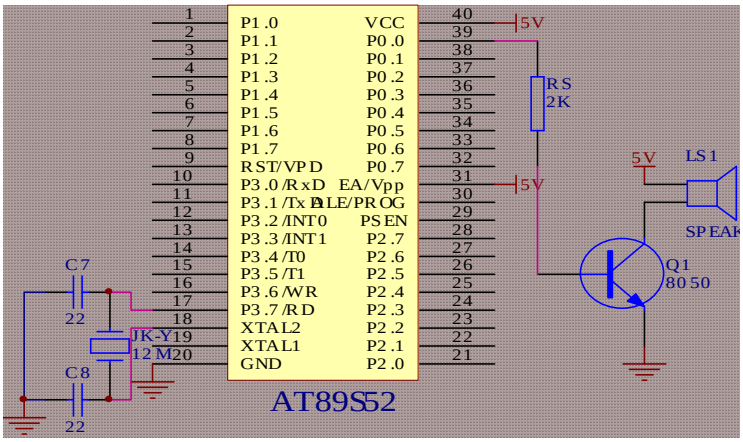
```
void main(void)
{
   TMOD=0x02;   //定时器工作在方式 2
   ET0=1;
   EA=1;
   sec=0;
   while(1)
   {
        KEY();
       LED();
     }
}
```

八. 利用定时器产生乐曲

［实验任务］

利用单片机的 I/O 口演奏乐曲。

［硬件电路图］



［DPY-1 实验板连接］

用导线把 JK1 接到 P0.0 上就可以了。

［实验原理］

乐曲是按照一定的高低，长短和强弱关系组成的关系，在一首乐曲中，每一个音符与频率有关。所以我们只要把有关频率的占空比数据做成表格，在通过查表，在 I/O 口输出相关乐曲的方波，便产生了乐曲。

［C 语言源程序］

```
#include "reg52.h"
unsigned char Count;
sbit _Speak =P0^0 ;                  //讯响器控制脚
unsigned char code SONG[] ={         //祝你平安
0x26,0x20,0x20,0x20,0x20,0x20,0x26,0x10,0x20,0x10,0x20,0x80,0x26,0x20,0x30,0x20,
0x30,0x20,0x39,0x10,0x30,0x10,0x30,0x80,0x26,0x20,0x20,0x20,0x20,0x20,0x1c,0x20,
0x20,0x80,0x2b,0x20,0x26,0x20,0x20,0x20,0x2b,0x10,0x26,0x10,0x2b,0x80,0x26,0x20,
0x30,0x20,0x30,0x20,0x39,0x10,0x26,0x10,0x26,0x60,0x40,0x10,0x39,0x10,0x26,0x20,
0x30,0x20,0x30,0x20,0x39,0x10,0x26,0x10,0x26,0x80,0x26,0x20,0x2b,0x10,0x2b,0x10,
```

```
0x2b,0x20,0x30,0x10,0x39,0x10,0x26,0x10,0x2b,0x10,0x2b,0x20,0x2b,0x40,0x40,0x20,
0x20,0x10,0x20,0x10,0x2b,0x10,0x26,0x30,0x30,0x80,0x18,0x20,0x18,0x20,0x26,0x20,
0x20,0x20,0x20,0x40,0x26,0x20,0x2b,0x20,0x30,0x20,0x30,0x20,0x1c,0x20,0x20,0x20,
0x20,0x80,0x1c,0x20,0x1c,0x20,0x1c,0x20,0x30,0x20,0x30,0x60,0x39,0x10,0x30,0x10,
0x20,0x20,0x2b,0x10,0x26,0x10,0x2b,0x10,0x26,0x10,0x26,0x10,0x2b,0x10,0x2b,0x80,
0x18,0x20,0x18,0x20,0x26,0x20,0x20,0x20,0x20,0x60,0x26,0x10,0x2b,0x20,0x30,0x20,
0x30,0x20,0x1c,0x20,0x20,0x20,0x20,0x80,0x26,0x20,0x30,0x10,0x30,0x10,0x30,0x20,
0x39,0x20,0x26,0x10,0x2b,0x10,0x2b,0x20,0x2b,0x40,0x40,0x10,0x40,0x10,0x20,0x10,
0x20,0x10,0x2b,0x10,0x26,0x30,0x30,0x80,0x00,
  //路边的野华不要采
0x30,0x1C,0x10,0x20,0x40,0x1C,0x10,0x18,0x10,0x20,0x10,0x1C,0x10,0x18,0x40,0x1C,
0x20,0x20,0x20,0x1C,0x20,0x18,0x20,0x20,0x80,0xFF,0x20,0x30,0x1C,0x10,0x18,0x20,
0x15,0x20,0x1C,0x20,0x20,0x20,0x26,0x40,0x20,0x20,0x2B,0x20,0x26,0x20,0x20,0x20,
0x30,0x80,0xFF,0x20,0x20,0x1C,0x10,0x18,0x10,0x20,0x20,0x26,0x20,0x2B,0x20,0x30,
0x20,0x2B,0x40,0x20,0x20,0x1C,0x10,0x18,0x10,0x20,0x20,0x26,0x20,0x2B,0x20,0x30,
0x20,0x2B,0x40,0x20,0x30,0x1C,0x10,0x18,0x20,0x15,0x20,0x1C,0x20,0x20,0x20,0x26,
0x40,0x20,0x20,0x2B,0x20,0x26,0x20,0x20,0x20,0x30,0x80,0x20,0x30,0x1C,0x10,0x20,
0x10,0x1C,0x10,0x20,0x20,0x26,0x20,0x2B,0x20,0x30,0x20,0x2B,0x40,0x20,0x15,0x1F,
0x05,0x20,0x10,0x1C,0x10,0x20,0x20,0x26,0x20,0x2B,0x20,0x30,0x20,0x2B,0x40,0x20,
0x30,0x1C,0x10,0x18,0x20,0x15,0x20,0x1C,0x20,0x20,0x20,0x26,0x40,0x20,0x20,0x2B,
0x20,0x26,0x20,0x20,0x20,0x30,0x30,0x20,0x30,0x1C,0x10,0x18,0x40,0x1C,0x20,0x20,
0x20,0x26,0x40,0x13,0x60,0x18,0x20,0x15,0x40,0x13,0x40,0x18,0x80,0x00,};
void Time0_Init()
{    TMOD = 0x01;
     IE    = 0x82;
     TH0   = 0xD8;
     TL0   = 0xEF;        //12MZ 晶振，10ms
}
void Time0_Int() interrupt 1
{    TH0 = 0xD8;
     TL0 = 0xEF;
     Count++;             //长度加 1
}
void Delay_xMs(unsigned int x) //1MS 延时子程序
{      unsigned int i,j;
      for( i =0;i < x;i++ )
      {for( j =0;j<3;j++ );}
}
void Play_Song(unsigned char i)     //:歌曲播放子程序 i 为播放哪一段曲目
{    unsigned char Temp1,Temp2;
     unsigned int Addr;
     Count = 0;                     //中断计数器清 0
     Addr = i * 217;
     while(1)
```

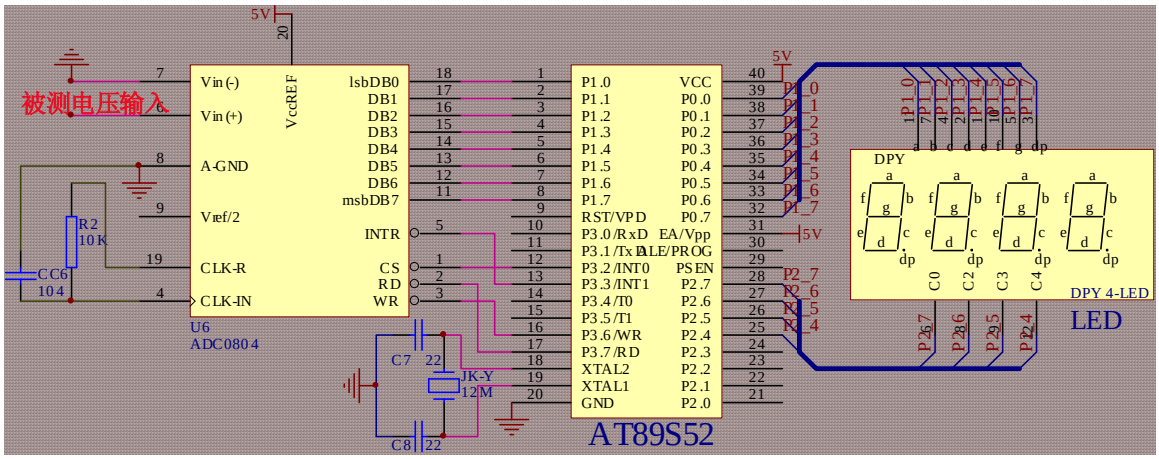
```
{    Temp1 = SONG[Addr++];
     if ( Temp1 == 0xFF )              //休止符
     {TR0 = 0;   Delay_xMs(100); }
     else if ( Temp1 == 0x00 )      //歌曲结束符
     {return;}
     else      {Temp2 = SONG[Addr++];
       TR0 = 1;
        while(1)
        {    _Speak = ~_Speak;
             Delay_xMs(Temp1);
             if ( Temp2 == Count )
             {Count = 0;
              break;
           }  }  }  }  } //请注意写程序时的格式规范，此处是为了节省空间
void main()   //主程序
{Time0_Init();              //定时器 0 中断初始化
while(1)
{Play_Song(0);}     //播放
}
```

九．数摸转换 ADC0804 的应用

［实验任务］

从 ADC0804 的通道 IN+输入 0－5V 之间的模拟量，通过 ADC0804 转换成数字量在数码管上以十进制形成显示出来。

［硬件电路图］



［DPY-1 实验板连接］

用短接帽把 JP3 和 JP4，JP11 和 JP12 全部短接，用排线把 JP-CODE 连到 JP8 是，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3…… 把 JP-CS 连到 JP14 上,注意:4H 接 P2.4;3H 接 P2.5;2H 接 P2.6;1H 接 P2.7;

［实验原理］

ADC0804 是 8 位全 MOS 中速 A/D 转换器、它是逐次逼近式 A/D 转换器，片内有三态数据输出锁存器，可以和单片机直接接口。单通道输入，转换时间大约为 100us。ADC0804 转换时序是：当 CS＝0 许可进行 A/D 转换。WR 由低到高时，A/D

开始转换，一次转换一共需要 66－73 个时钟周期。CS 与 WR 同时有效时启动 A/D 转换，转换结束产生 INTR 信号（低电平有效），可供查询或者中断信号。在 CS 和 RD 的控制下可以读取数据结果。

[C 语言源程序]

```
/*注意：程序下载到 DPY-1 实验板单片机后一定要使 SW DIP1 的开关脱离 ON 档。或者直接将下载线从实验板上拔下。因为下载线接在 I/O 口 P1.5，P1.6，P1.7 上，下载线的电平将会影响测量结果*/
#include <reg52.h>
code unsigned char seg7code[10]={0x3f,0x06,0x5b,0x4f,0x66,
                                 0x6d,0x7d,0x07,0x7f,0x6f}; //显示段码
sbit int1=P3^3;   //定义管脚功能
sbit cs=P3^2;
sbit wr=P3^6;
sbit rd=P3^7;
void Delay(unsigned int tc)        //显示延时程序
{while( tc != 0 )
     {unsigned int i;
      for(i=0; i<100; i++);
      tc--;}
}
unsigned char adc0804( void )        //读 AD0804 子程序
{    unsigned char addata,i;
     rd=1;wr=1;int1=1; //读 ADC0804 前准备
     P1=0xff;              //P1 全部置一准备
     cs=0;wr=0;wr=1; //启动 ADC0804 开始测电压
     while(int1==1); //查询等待 A/D 转换完毕产生的 INT（低电平有效）信号
     rd=0;       //开始读转换后数据
     i=i; i=i;      //无意义语句，用于延时等待 ADC0804 读数完毕
     addata=P1;//读出的数据赋与 addate
     rd=1;cs=1;//读数完毕
    return(addata);//返回最后读出的数据
}
unsigned int datpro(void)//ADC0804 读出的数据处理
{ unsigned char x;
  unsigned int dianyah,dianyal; //用于存储读出数据的高字节和低字节
  unsigned int dianya=0;            //存储最后处理完的结果 注意数据类型
  for(x=0;x<10;x++)                  //将 10 次测得的结果存储在 dianya 中
  {dianya=adc0804()+dianya; }
  dianya=dianya/10; //求平均值
  dianyah=dianya&0xf0; //屏蔽低四位
  dianyah=dianyah>>4;   //右移四位 取出高四位
  dianyal=dianya&0x0f; //屏蔽高四位 取出低四位
  dianya=dianyal*20+dianyah*320; //最后的结果是一个四位数，便于显示
  return(dianya);//返回最后处理结果
```

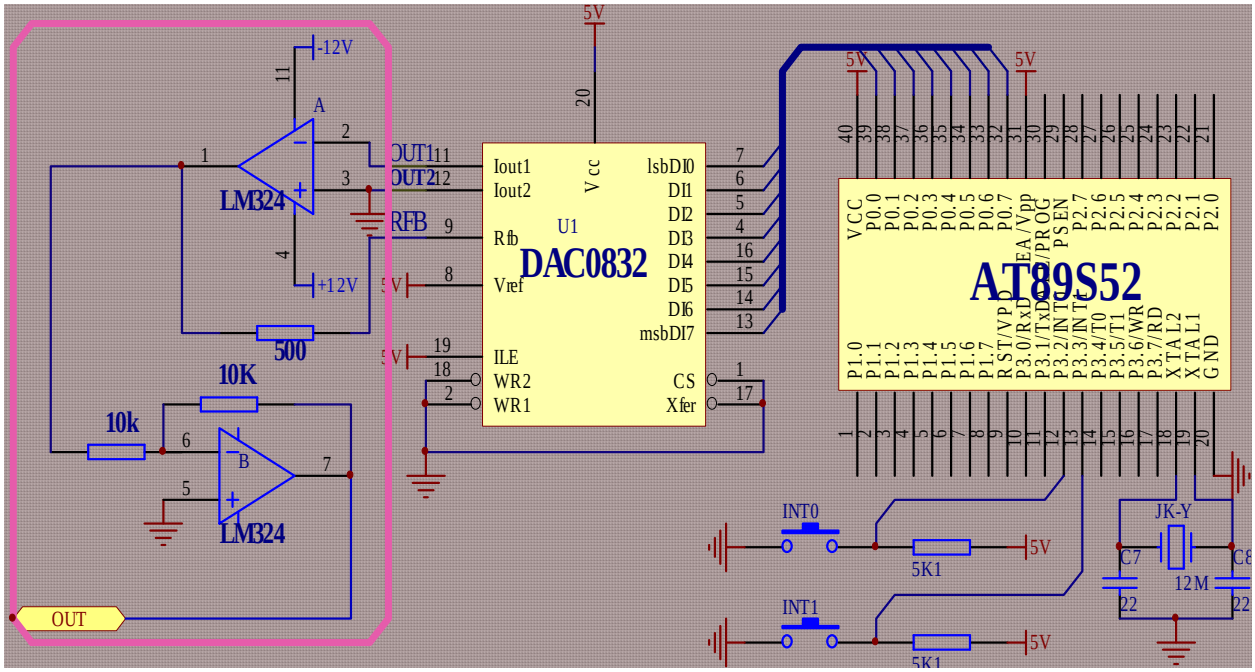
```
 }
 void Led()
 {   unsigned int date;
     date=datpro();   //调用数据处理最后结果
     P2=P2&0xef;
     P0=seg7code[date/1000]|0x80; //输出个位数和小数点
     Delay(8); P2=P2|0xf0; P2=P2&0xdf;
     P0=seg7code[date%1000/100]; //输出小数点后第一位
     Delay(8); P2=P2|0xf0; P2=P2&0xbf;
     P0=seg7code[date%100/10];     //输出小数点后第二位
      Delay(8); P2=P2|0xf0; P2=P2&0x7f;
     P0=seg7code[date%10];          //输出小数点后第三位
     Delay(8); P2=P2|0xf0;
}
main()
{
   while(1)
   {
        Led( ); //只需调用显示函数
   }
}
```

十．摸数转换 DAC0832 的应用

［实验任务］

用两个按键通过单片机控制 DAC0832 的输出，使 OUT 端可以输出 0—5V 的幅值，频率为 1KHZ 的锯齿波和三角波两种波形。通上电源后；按下 INT1 则输出三角波，在按下 INT0 输出锯齿波。

［硬件电路图］



[DPY-1 实验板连接]

用排线把 JP32—DATA 连到 JP8 上，注意 D0 对应 P0.0；D1 对应 P0.1；D2 对应 P0.2…… 将 JP32—XT 连到 JP32—GND 上（使 0832 以直通方式工作）。

按键可以直接使用 INT0 和 INT1 按键（程序中以扫描方式识别按键）。

硬件电路图中的画框的部分，运放 LM324 的连接部分需要用户自己搭建，注意 LM324 使用的是正负 12V 的双电源供电。我们只使用 LM324 四个运放中的两个。

[实验原理]

ADC0804 是 8 位全 MOS 中速 D/A 转换器，采用 R—2RT 形电阻解码网络，转换结果为一对差动电流输出，转换时间大约为 1us。使用单电源+5V－+15V 供电。参考电压为-10V－+10V。在此我们直接选择+5V 作为参考电压。DAC0832 有三种工作方式：直通方式，单缓冲方式，双缓冲方式；在此我们选择直通的工作方式，将 XFER WR 1WR2 CS 管脚全部接数字地。管脚 8 接参考电压，在此我们接的参考电压是+5V。那么经过第一级运放后，输出电压将是-5V－0V，在经过第二级运放反相放大 1 倍以后将可以输出 0V—5V 了。我们在控制 P1 口输出数据有规律的变化将可以产生三角波，锯齿波，梯型波等波形了。

[C 语言源程序]

```
#include <AT89X51.H>
unsigned char keycnt=0;
unsigned char tcnt=0;              //键值判断
bit sjz=0;        //产生三角波时用到的标志
void delayl()       //延时子程序
{       unsigned char i,j;
       for(i=20;i>0;i--)
       for(j=248;j>0;j--); }
void KEY()                  //按键扫描程序
{
if(P3_2==0)
   {
     delayl();       //延时跳过按下时的抖动
       if(P3_2==0)
          {
            keycnt=0;   //定时器产生锯齿波标志
           TR0=0;       //暂时停止波形输出
            TH0=0x256-40; //对 TH0 TL0 赋值
            TL0=0x256-40;
            TR0=1;            //开始定时,产生锯齿波
            while(P3_2==0);   //如果一直按着键，则等待松键开
           delayl();       //延时跳过松开后的抖动
         }    }
if(P3_3==0)
   {
      delayl();       //延时跳过按下时的抖动
      if(P3_3==0)
         {
```

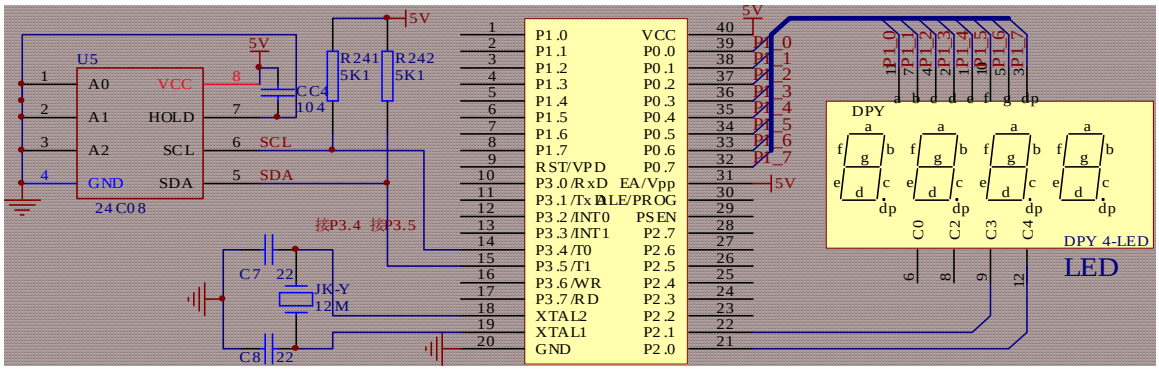
```
            keycnt=1; //定时器产生三角波标志
            TR0=0;      //暂时停止波形输出
            TH0=0x256-40; //对 TH0 TL0 赋值
            TL0=0x256-40;
            TR0=1;      //开始定时  产生三角波
          }
       while(P3_2==0);   //如果一直按着键，则等待松键开
      delayl();      //延时跳过松开后的抖动
        }
       }    //请注意写程序时的格式规范，此处是为了节省纸张
void t0(void) interrupt 1 using 0   //定时中断服务函数
{
 if(keycnt==0)   //产生锯齿波
     {
      P0=tcnt;
      tcnt=+0x0a; //步进 0.2V/一次中断
       if(tcnt==0xfb)
       {tcnt=0;}
     }
 if(keycnt==1)   //产生三角波
       {   if(sjz==0)
            {P0=tcnt;
             tcnt=+0x0a;//步进 0.2V/一次中断
              if(tcnt==0xfa)
                   {sjz=1;}
          }
           if(sjz==1)
             {
              P0=tcnt;
              tcnt=-0x0a;
              if(tcnt==0)
                 {sjz=0;}
              }
        }
}
void main(void)
{
TMOD=0x02;   //定时器工作在方式 2
ET0=1;
EA=1;
while(1)
  {   KEY();   }
}
```

十一．24C08 的读写操作

[实验任务]

利用 24C08 断电以后存储的数据不消失的特点，可以做一个断电保护装置。首先利用单片机做一个 0－99 秒的自动计时器。然后随机关断电源，在 通电以后计时器接着断电前的状态继续计时。

[硬件电路图]



[DPY-1 实验板连接]

用短接帽把 JP11 和 JP12 对应 P3.4 和 P3.5 短接。

用排线把 JP-CODE 连到 JP8 是，注意：a 接 P0.0；b 接 P0.1；c 接 P0.3…… 把 JP-CS 连到 JP14 上，注意：4H 接 P2.0；3H 接 P2.1；

[实验原理]

首先简单的说明以下 I^2C 总线，I^2C 总线是一种串行数据总线，只有二根信号线，一根是双向的数据线 SDA，另一根是时钟线 SCL。在 I^2C 总线上传送的一个数据字节由八位组成。总线对每次传送的字节数没有限制，但每个字节后必须跟一位应答位。数据传送首先传送最高位(MSB)，数据传送按图 1 所示格式进行。首先由主机发出启动信号“S”(SDA 在 SCL 高电平期间由高电平跳变为低电平)，然后由主机发送一个字节的数据。启动信号后的第一个字节数据具有特殊含义：高七位是从机的地址，第八位是传送方向位，0 表示主机发送数据(写)，1 表示主机接收数据(读)。被寻址到的从机设备按传送方向位设置为对应工作方式。标准 I^2C 总线的设备都有一个七位地址，所有连接在 I^2C 总线上的设备都接收启动信号后的第一个字节，并将接收到的地址与自己的地址进行比较，如果地址相符则为主机要寻访的从机，应在第九位答时钟脉冲时向 SDA 线送出低电平作为应答。除了第一字节是通用呼叫地址或十位从机地址之外第二字节开始即数据字节。数据传送完毕，由主机发出停止信号“P”(SDA 在 SCL 高电平期间由低电平跳变为高电平)。

AT24C 系列串行 E2PROM 具有 I^2C 总线接口功能，功耗小，宽电源电压(根据不同型号 2.5V～6.0V)，工作电流约为 3mA，静态电流随电源电压不同为 30μA～110μA，AT24C 系列串行 E2PROM 参数如下

型 号	容 量	器件寻址字节(8 位)		一次装载字节数
AT24C01	128×8	1010A2A1A0	R/W	4
AT24C02	256×8	1010A2A1A0	R/W	8
AT24C04	512×8	1010A2A1P0	R/W	16
AT24C08	1024×8	1010A2P1P0	R/W	16
AT24C16	2048×8	1010P2P1P0	R/W	16

由于 I²C 总线可挂接多个串行接口器件，在 I²C 总线中每个器件应有唯一的器件地址，按 I²C 总线规则，器件地址为 7 位数据(即一个 I²C 总线系统中理论上可挂接 128 个不同地址的器件)，它和 1 位数据方向位构成一个器件寻址字节，最低位 D0 为方向位(读/写)。器件寻址字节中的最高 4 位(D7～D4)为器件型号地址，不同的 I²C 总线接口器件的型号地址是厂家给定的，如 AT24C 系列 E2PROM 的型号地址皆为 1010，器件地址中的低 3 位为引脚地址 A2　A1　A0，对应器件寻址字节中的 D3、D2、D1 位，在硬件设计时由连接的引脚电平给定。

对 AT24C 系列 E2PROM 的读写操作完全遵守 I2C 总线的主收从发和主发从收的规则。

[C 语言源程序]

```
#include <AT89X52.H>
#include <stdio.h>
#include <absacc.h>
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,
                            0x6d,0x7d,0x07, 0x7f,0x6f,};
unsigned char sec;          //定义计数值，每过 1 秒，sec 加 1
unsigned int tcnt;          //定时中断次数
bit   write=0;              //写 24C08 的标志；
sbit gewei=P2^0;            //个位选通定义
sbit shiwei=P2^1;           //十位选通定义
/////////24C08 读写驱动程序////////////////////
sbit scl=P3^4;     // 24c08 SCL
sbit sda=P3^5;    // 24c08 SDA
void delay1(unsigned char x)
{   unsigned int i;
   for(i=0;i<x;i++);
   ;}
void flash()
{   ;   ; }
void x24c08_init()   //24c08 初始化子程序
 {scl=1; flash(); sda=1; flash();}
void start()          //启动 I²C 总线
{sda=1; flash(); scl=1; flash(); sda=0; flash(); scl=0; flash();}
void stop()           //停止 I²C 总线
{sda=0; flash(); scl=1; flash(); sda=1; flash();}
void writex(unsigned char j)   //写一个字节
{   unsigned char i,temp;
    temp=j;
    for (i=0;i<8;i++)
    {temp=temp<<1; scl=0; flash(); sda=CY; flash(); scl=1; flash();}
    scl=0; flash(); sda=1; flash();
}
unsigned char readx()     //读一个字节
{
```

```
    unsigned char i,j,k=0;
    scl=0;   flash();   sda=1;
    for (i=0;i<8;i++)
     {
        flash();   scl=1;   flash();
        if (sda==1) j=1;
        else j=0;
        k=(k<<1)|j;
       scl=0;}
    flash();   return(k);
}
void clock()              // I²C 总线时钟
{
    unsigned char i=0;
    scl=1;   flash();
    while ((sda==1)&&(i<255))i++;
    scl=0; flash();
}
////////从 24c02 的地址 address 中读取一个字节数据/////
unsigned char x24c08_read(unsigned char address)
{
    unsigned char i;
    start(); writex(0xa0);
    clock(); writex(address);
    clock(); start();
    writex(0xa1); clock();
    i=readx(); stop();
    delay1(10);
    return(i);
}
//////向 24c02 的 address 地址中写入一字节数据 info/////
void x24c08_write(unsigned char address,unsigned char info)
{
    EA=0;
    start(); writex(0xa0);
    clock(); writex(address);
    clock(); writex(info);
    clock(); stop();
    EA=1;
    delay1(50);
}
//////////////24C08 读写驱动程序完//////////////////////
void Delay(unsigned int tc)        //延时程序
{
```

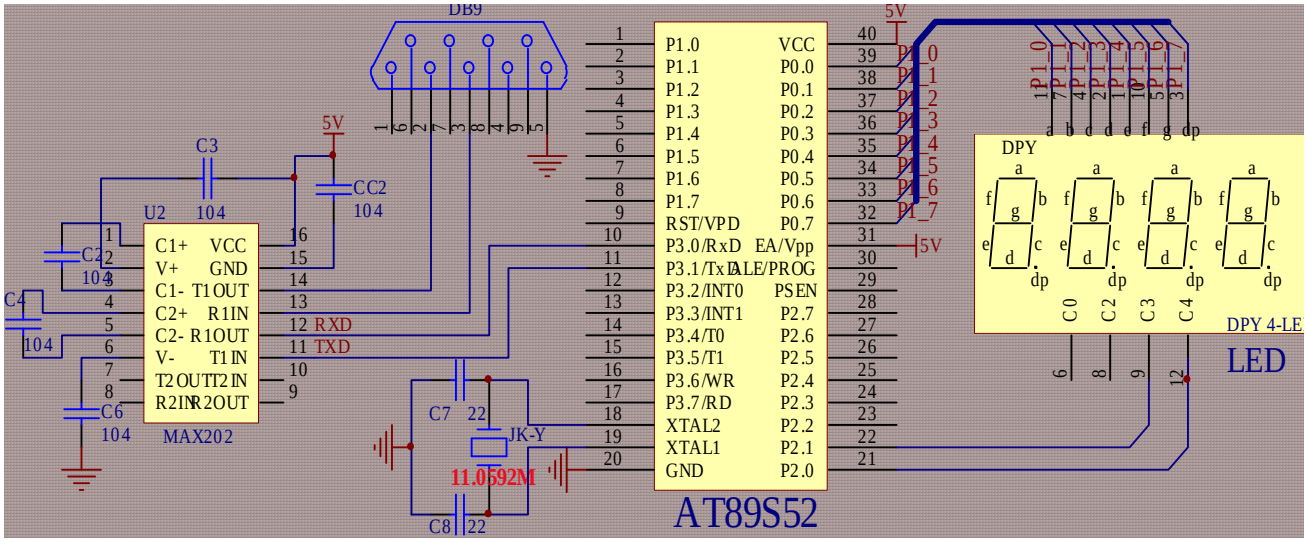
```
    while( tc != 0 )
        {unsigned int i;
         for(i=0; i<100; i++);
         tc--;}
}
void LED()                          //LED 显示函数
{
    shiwei=0; P0=table[sec/10]; Delay(8); shiwei=1;
    gewei=0;   P0=table[sec%10]; Delay(5); gewei=1;
}
void t0(void) interrupt 1 using 0   //定时中断服务函数
{
 TH0=(65536-50000)/256; //对 TH0 TL0 赋值
 TL0=(65536-50000)%256; //重装计数初值
 tcnt++;              //每过 250ust tcnt 加一
 if(tcnt==20)   //计满 20 次（1 秒）时
    {
     tcnt=0;    //重新再计
     sec++;
     write=1;   //1 秒写一次 24C08
     if(sec==100) //定时 100 秒，在从零开始计时
       {sec=0;}
    }
 }
void main(void)
{
TMOD=0x01;   //定时器工作在方式 1
ET0=1;   EA=1;
x24c08_init();             //初始化 24C08
sec=x24c08_read(2);//读出保存的数据赋于 sec
TH0=(65536-50000)/256; //对 TH0 TL0 赋值
TL0=(65536-50000)%256; //使定时器 0.05 秒中断一次
TR0=1;                  //开始计时
 while(1)
  {
    LED();
    if(write==1) //判断计时器是否计时一秒
     {
      write=0;                    //清零
       x24c08_write(2,sec);   //在 24c08 的地址 2 中写入数据 sec
     }
    }
}
```

十二.PC 机与单片机通信(RS232 协议)

[实验任务]

单片机串口通信的应用，通过串口，我们的个人电脑和单片机系统进行通信。个人电脑作为上位机，向下位机单片机系统发送十六进制或者 ASCLL 码，单片机系统接收后，用 LED 显示接收到的数据和向上位机发回原样数据。

[硬件电路图]



[DPY-1 实验板连接]

用短接帽把 JP11 和 JP12 对应 P3.0 和 P3.1 短接。

用排线把 JP-CODE 连到 JP8 是,注意:a 接 P0.0;b 接 P0.1;c 接 P0.3…… 把 JP-CS 连到 JP14 上,注意:4H 接 P2.0;3H 接 P2.1;

用串口连接线 DB9 和个人电脑的串口连接起来。

[实验原理]

RS-232 是美国电子工业协会正式公布的串行总线标准，也是目前最常用的串行接口标准，用来实现计算机与计算机之间、计算机与外设之间的数据通讯。RS-232 串行接口总线适用于：设备之间的通讯距离不大于 15m，传输速率最大为 20kBps。RS-232 协议以-5V－15V 表示逻辑 1；以+5V－15V 表示逻辑 0。 我们是用 MAX232 芯片将 RS232 电平转换为 TTL 电平的。 一个完整的 RS-232 接口有 22 根线，采用标准的 25 芯插头座。我们在 这里使用的是简化的 9 芯插头座。

注意我们在这里使用的晶振是 11.0592M 的，而不是 12M。因为波特率的设置需要 11.0592M 的。

“串口调试助手 V2.1.exe” 软件的使用很简单，只要将串口选择‘CMO1’波特率设置为‘9600’ 数据位为 8 位。打开串口（如果关闭）。然后在发送区里输入要发送的数据，单击手动发送就将数据发送出去了。注意，如果选中‘十六进制发送’那么发送的数据是十六进制的，必须输入两位数据。如果没有选中，则发送的是 ASCLL 码，那么单片机控制的数码管将显示 ASCLL 码值。

[C 语言源程序]

```
#include "reg52.h"              //包函 8051 内部资源的定义
unsigned char dat;              //用于存储单片机接收发送缓冲寄存器 SBUF 里面的内容
sbit gewei=P2^0;                //个位选通定义
```

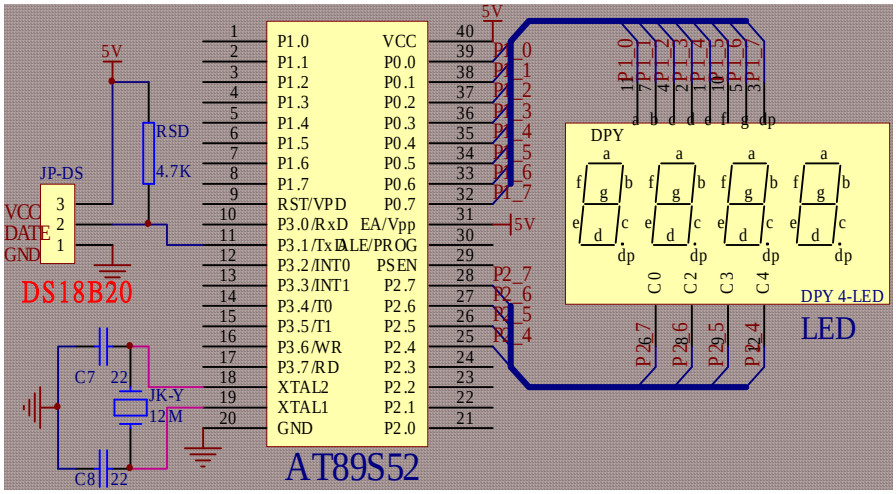
```
sbit shiwei=P2^1;              //十位选通定义
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,0x66,
                            0x6d,0x7d,0x07, 0x7f,0x6f,};
void Delay(unsigned int tc)        //延时程序
{
     while( tc != 0 )
         {unsigned int i;
          for(i=0; i<100; i++);
          tc--;}
}
void LED()      //LED 显示接收到的数据（十进制）
{
    shiwei=0; P0=table[dat/10]; Delay(8); shiwei=1;
    gewei=0;   P0=table[dat%10]; Delay(5); gewei=1;
}
///////功能:串口初始化,波特率 9600，方式 1/////////
void Init_Com(void)
{
        TMOD = 0x20;
        PCON = 0x00;
        SCON = 0x50;
        TH1 = 0xFd;
        TL1 = 0xFd;
        TR1 = 1;
}
/////主程序功能:实现接收数据并把接收到的数据原样发送回去///////
void main()
{
   Init_Com();//串口初始化
   while(1)
    {
         if ( RI )          //扫描判断是否接收到数据，
         {
                dat = SBUF;         //接收数据 SBUF 赋与 dat
                RI=0;                //RI 清零。
                SBUF = dat;         //在原样把数据发送回去
         }
     LED();   //显示接收到的数据
    }
}
///这一个例子是以扫描的方式编写的，还可以以中断的方式编写，请大家思考//////
```

十三.DS18B20 测量温度系统

[实验任务]

用一片 DS18B20 构成测温系统，测量的温度精度达到 0.1 度，测量的温度的范围在－20 度到＋50 度之间，用 4 位数码管显示出来。

[硬件电路图]



[DPY-1 实验板连接]

用排线把 JP-CODE 连到 JP8 是，注意：a 接 P0.0;b 接 P0.1;c 接 P0.3…… 把 JP-CS 连到 JP14 上,注意:4H 接 P2.4;3H 接 P2.5;2H 接 P2.6;1H 接 P2.7;

连接好 DS18B20 注意极性不要弄反，否则可能烧坏。DS18B20 的外型与常用的三极管一模一样，上面右图是它的管脚分布。用导线将 JK—DS 的 DA 端连到 P3.1 上。

[实验原理]

DS18B20 数字温度计是 DALLAS 公司生产的 1—Wire，即单总线器件，具有线路简单，体积小的特点。因此用它来组成一个测温系统，具有线路简单，在一根通信线，可以挂很多这样的数字温度计。DS18B20 产品的特点（1)、只要求一个 I/O 口即可实现通信。(2)、在 DS18B20 中的每个器件上都有独一无二的序列号。（3)、实际应用中不需要外部任何元器件即可实现测温。(4)、测量温度范围在－55。C 到＋125。C 之间。(5)、数字温度计的分辨率用户可以从 9 位到 12 位选择。(6)、内部有温度上、下限告警设置。

DS18B20 详细引脚功能描述 1 GND 地信号；2 DQ 数据输入/输出引脚。开漏单总线接口引脚。当被用着在寄生电源下，也可以向器件提供电源；3 VDD 可选择的 VDD 引脚。当工作于寄生电源时，此引脚必须接地。

DS18B20 的使用方法。由于 DS18B20 采用的是 1—Wire 总线协议方式，即在一根数据线实现数据的双向传输，而对 AT89S51 单片机来说，我们必须采用软件的方法来模拟单总线的协议时序来完成对 DS18B20 芯片的访问。由于 DS18B20 是在一根 I/O 线上读写数据，因此，对读写的数据位有着严格的时序要求。DS18B20 有严格的通信协议来保证各位数据传输的正确性和完整性。该协议定义了几种信号的时序：初始化时序、读时序、写时序。所有时序都是将主机作为主设备，单总线器件作为从设备。而每一次命令和数据的传输都是从主机主动启动写时序开始，如果要求单总线器件回送数据，在进行写命令后，主机需启动读时序完成数据接收。数据和命令的传输都是低位在先。

[C 语言源程序]

```
#include<reg52.h>
code unsigned char seg7code[11]={0x3f,0x06,0x5b,0x4f,0x66,
                                 0x6d,0x7d,0x07,0x7f,0x6f,0x40};  // 显示段码
void Delay(unsigned int tc)        //显示延时程序
{while( tc != 0 )
     {unsigned int i;
      for(i=0; i<100; i++);
      tc--;}
}
sbit TMDAT =P3^1; //DS18B20 的数据输入/输出脚 DQ,根据情况设定
unsigned int sdata;//测量到的温度的整数部分
unsigned char xiaoshu1;//小数第一位
unsigned char xiaoshu2;//小数第二位
unsigned char xiaoshu;//两位小数
bit   fg=1;              //温度正负标志
void dmsec (unsigned int count)           //延时部分
 {
     unsigned char i;
     while(count--)
     {for(i=0;i<115;i++);}
 }
void tmreset (void)           //发送复位
{
     unsigned char i;
     TMDAT=0;     for(i=0;i<103;i++);
     TMDAT = 1; for(i=0;i<4;i++);
}
bit tmrbit (void)           //读一位//
 {
      unsigned int i;
      bit dat;
      TMDAT = 0;
      i++;
      TMDAT = 1;
      i++; i++;   //微量延时     //
      dat = TMDAT;
     for(i=0;i<8;i++);
      return (dat);
 }
unsigned char tmrbyte (void)           //读一个字节
   {
      unsigned char i,j,dat;
```

```
    dat = 0;
    for (i=1;i<=8;i++)
    { j = tmrbit();   dat = (j << 7) | (dat >> 1); }
     return (dat);
  }
void tmwbyte (unsigned char dat)        //写一个字节
{
    unsigned char j,i;
    bit testb;
    for (j=1;j<=8;j++)
    { testb = dat & 0x01;
     dat = dat >> 1;
     if (testb)
     {    TMDAT = 0;              //写 0
          i++; i++;
         TMDAT = 1;
         for(i=0;i<8;i++); }

     else
    {   TMDAT = 0;              //写 0
        for(i=0;i<8;i++);
       TMDAT = 1;
        i++; i++;}
 }
}
void tmstart (void)           //发送 ds1820  开始转换
  {   tmreset();   //复位
    dmsec(1);   //延时
    tmwbyte(0xcc);   //跳过序列号命令
    tmwbyte(0x44);   //发转换命令  44H,
  }
void tmrtemp (void)           //读取温度
    {
        unsigned char a,b;
        tmreset ();   //复位
        dmsec (1);   //延时
        tmwbyte (0xcc);   //跳过序列号命令
        tmwbyte (0xbe);   //发送读取命令
        a = tmrbyte ();   //读取低位温度
        b = tmrbyte ();    //读取高位温度
        if(b>0x7f)          //最高位为 1 时温度是负
        {a=~a;     b=~b+1;          //补码转换，取反加一
         fg=0;          //读取温度为负时 fg=0
        }
```

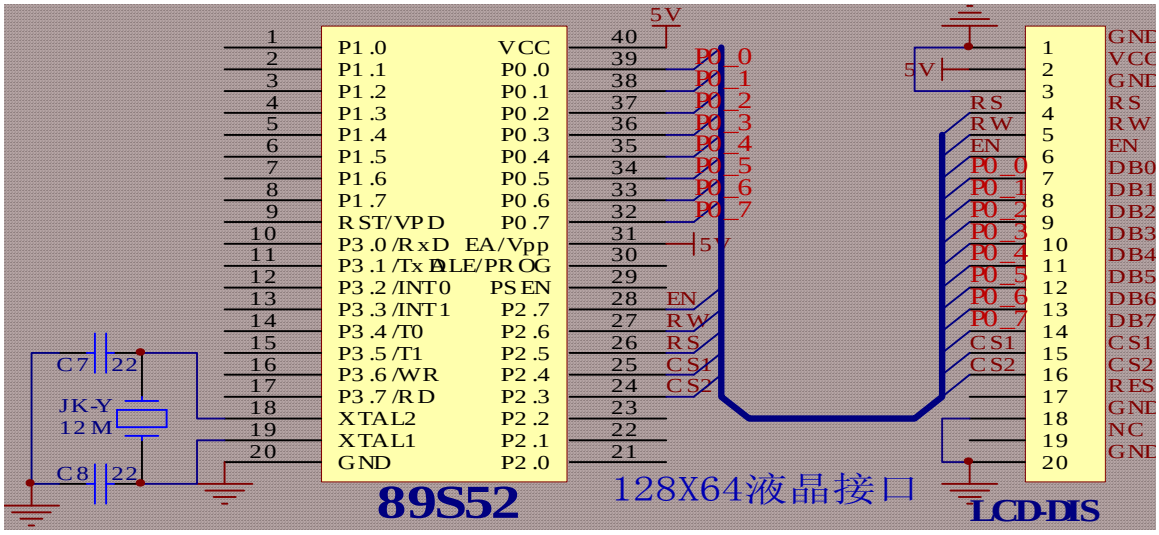
```
        sdata = a/16+b*16;          //整数部分
        xiaoshu1 = (a&0x0f)*10/16; //小数第一位
        xiaoshu2 = (a&0x0f)*100/16%10;//小数第二位
        xiaoshu=xiaoshu1*10+xiaoshu2; //小数两位
}
void DS18B20PRO(void)
{   tmstart();
   //dmsec(5);   //如果是不断地读取的话可以不延时  //
   tmrtemp();    //读取温度,执行完毕温度将存于 TMP 中  //
}
void Led()
{
    if(fg==1)    //温度为正时显示的数据
    {    P2=P2&0xef;
         P0=seg7code[sdata/10];                 //输出十位数
         Delay(8); P2=P2|0xf0; P2=P2&0xdf;
         P0=seg7code[sdata%10]|0x80; //输出个位和小数点
         Delay(8); P2=P2|0xf0; P2=P2&0xbf;
         P0=seg7code[xiaoshu1];    //输出小数点后第一位
         Delay(8); P2=P2|0xf0; P2=P2&0x7f;
         P0=seg7code[xiaoshu2];          //输出小数点后第二位
         Delay(4); P2=P2|0xf0;
    }
    if(fg==0)   //温度为负时显示的数据
    {    P2=P2&0xef;
         P0=seg7code[11];                //负号
         Delay(8); P2=P2|0xf0; P2=P2&0xdf;
         P0=seg7code[sdata/10]|0x80; //输出十位数
         Delay(8); P2=P2|0xf0; P2=P2&0xbf;
         P0=seg7code[sdata%10];    //输出个位和小数点
         Delay(8); P2=P2|0xf0; P2=P2&0x7f;
         P0=seg7code[xiaoshu1];          //输出小数点后第一位
         Delay(4); P2=P2|0xf0;
    }
}
main()
{fg=1;
 while(1)
 {
  DS18B20PRO();
  Led();
 }
}
```

十四. 128X64 液晶显示器的基本应用

[实验任务]

利用 128X64 点阵液晶显示屏显示图片，字符，汉字，画点，实现其基本显示功能。

[硬件电路图]



[DPY-1 实验板连接]

用短接帽把 JP8 和 JP9 对应所有的 I/O 短接。用短接帽把 JP14 和 JP15 对应所有的 I/O 短接。

将 128X64 液晶显示器插到 LCD-DISPLAY 的架子上。

[实验原理]

关于液晶显示的原理请用户自行其查找资料，这里只介绍 128X64 点阵液晶显示屏的具体应用。驱动程序为用户提供了一个应用框架，大家只要直接应用驱动程序提供的功能将可以了。在编程序前要先用液晶字模提取软件提取字模（我们在 QQ 共享资源里提供了两个），复制到 hzzi.h 文档，再在主程序里面编写程序。我们使用的液晶没有自带字库，必须提取字模到程序里。

由于 DPY-1 实验板只有 5V 稳压电源，所以没有点亮液晶的背光。

[C 语言源程序]

```
//注意：以下提供程序并没有包含 128X64 液晶显示驱动程序，只有主程序。
//请大家到 QQ 群：19305255  资源共享里面去下
#include<reg52.h>
#include<lcd12864.h> //包含 128X64 驱动程序
#include<hzzi.H>      //包含图片和汉字库
void delay(int x)       //延时程序
{
    int i,j;
    for(i=0;i<600;i++)
        for(j=0;j<x;j++);
}
void main(void)
{
    unsigned char lie;   //定义列
    unsigned char hang; //定义行
```

```
lcd_init();   //初始化液晶驱动硬件
lcd_clr();    //清屏
while(1)
   {
     lie=15;//列的值可以是 0--127 任意一个
     hang=0;//行的值视情况而定
    lcd_clr();
    Disp_Img(yema);//显示一张 128X64 的点阵的图片
    delay(200);   //延时等待
    lcd_clr();    //清屏使残留的点不影响后面的显示
    hz_disp(lie+0, hang, da);//在第 15 列,第 0 行显示汉字'大'
    hz_disp(lie+16,hang, jia);//显示汉字'家'
    hz_disp(lie+32,hang, yi);//显示汉字'一'
    hz_disp(lie+48,hang, qi);//显示汉字'起'
     hz_disp(lie+64,hang, lai);//显示汉字'来'
    hz_disp(lie+80,hang, xue);//显示汉字'学'
    hang=2;//换行  一行有 8 个点阵  一列只有一个
    hz_disp(lie+24,hang, dan );//显示汉字'单'
    hz_disp(lie+40,hang, pian);//显示汉字'片'
    hz_disp(lie+56,hang, ji);//显示汉字'机'
     hang=4;//换行
    hz_disp(lie+0, hang, huan);
    hz_disp(lie+16,hang, ying);
    hz_disp(lie+32,hang, jia1);
    hz_disp(lie+48,hang, ru);
     hz_disp(lie+64,hang, qq);
    hz_disp(lie+80,hang, qun);
    lie=32; hang=7;//换行.换列.
    delay(50);
    lcd_putchar8x8(lie+0, hang, 1+'0');//显示 8X8 字符
    delay(50);//延时  达到一个字一个字往外蹦的效果
    lcd_putchar8x8(lie+8, hang, 9+'0');delay(50);
    lcd_putchar8x8(lie+16,hang, 3+'0');delay(50);
    lcd_putchar8x8(lie+24,hang, 0+'0');delay(50);
    lcd_putchar8x8(lie+32,hang, 5+'0');delay(50);
    lcd_putchar8x8(lie+40,hang, 2+'0');delay(50);
    lcd_putchar8x8(lie+48,hang, 5+'0');delay(50);
    lcd_putchar8x8(lie+56,hang, 5+'0');
    delay(200);//延时显示
    lcd_clr();
    hang=0;
     for(lie=0;lie<128;lie++)
    {//画点时  行取值 0--64,列取值 0--128
       set_point(lie,hang);
```

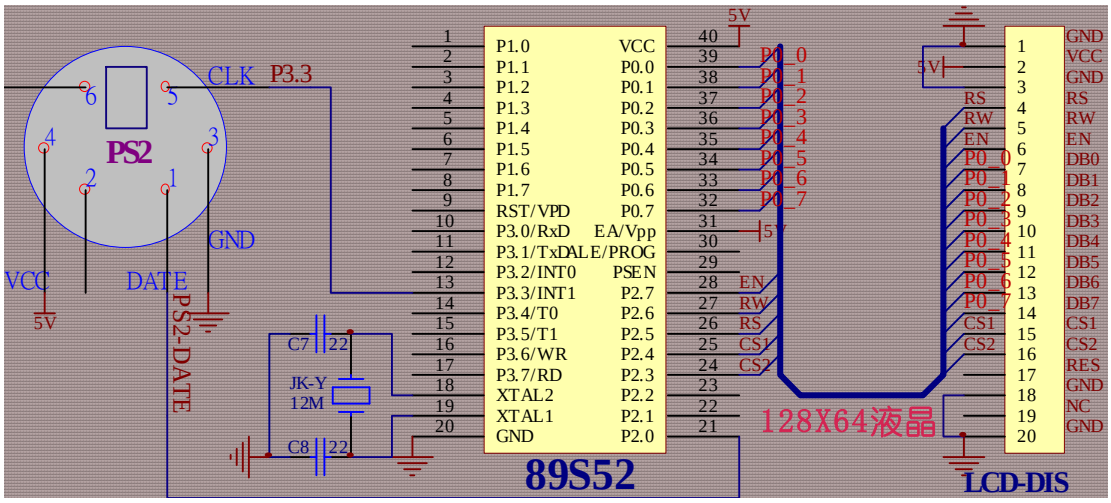
```
            hang++;
            if(hang==32){hang=0;}
            delay(5);
           }
         delay(200);
         hz_disp128x32(0,0,qian33);//显示 128X32 的图片
         hz_disp128x32(0,4,dpy);
         delay(300);
        }
}
////除了以上的例子外，还可以显示 12X12.32X32.120X2.点阵图形///
///用法和上面的例子差不多，大家实践几次就可以熟练应用了/////
```

十五.标准键盘 PS / 2 与单片机通信

[实验任务]

标准键盘 PS2 向单片机输入数字.字母.字符等，用 128X64 液晶显示器显示出来。

[硬件电路图]



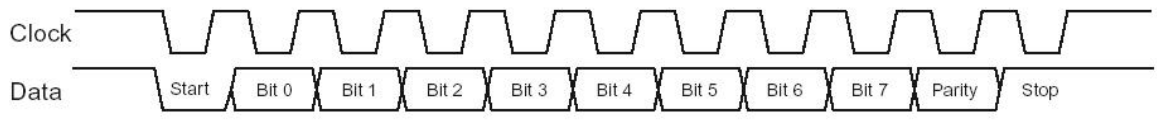
[DPY-1 实验板连接]

用短接帽把 JP8 和 JP9 对应所有的 I/O 短接。用短接帽把 JP14 和 JP15 对应所有的 I/O 短接。P3.3 对应的 I/O 口已经接好，不需要也不能短接。

将 128X64 液晶显示器插到 LCD-DISPLAY 的架子上。

[实验原理]

键盘通过时钟线和数据线和单片机通信，键盘和系统的相互通信都是采用 11 位格式的串行方式。第 1 位是起始位 0；第 2 到 9 位是 8 位数据位，第 10 位是奇偶校验位。第 11 位停止位。时序图如下



[C 语言源程序]

//注意 与第十三个实验 128X64 液晶显示器的基本应用一样，以下提供程序并没

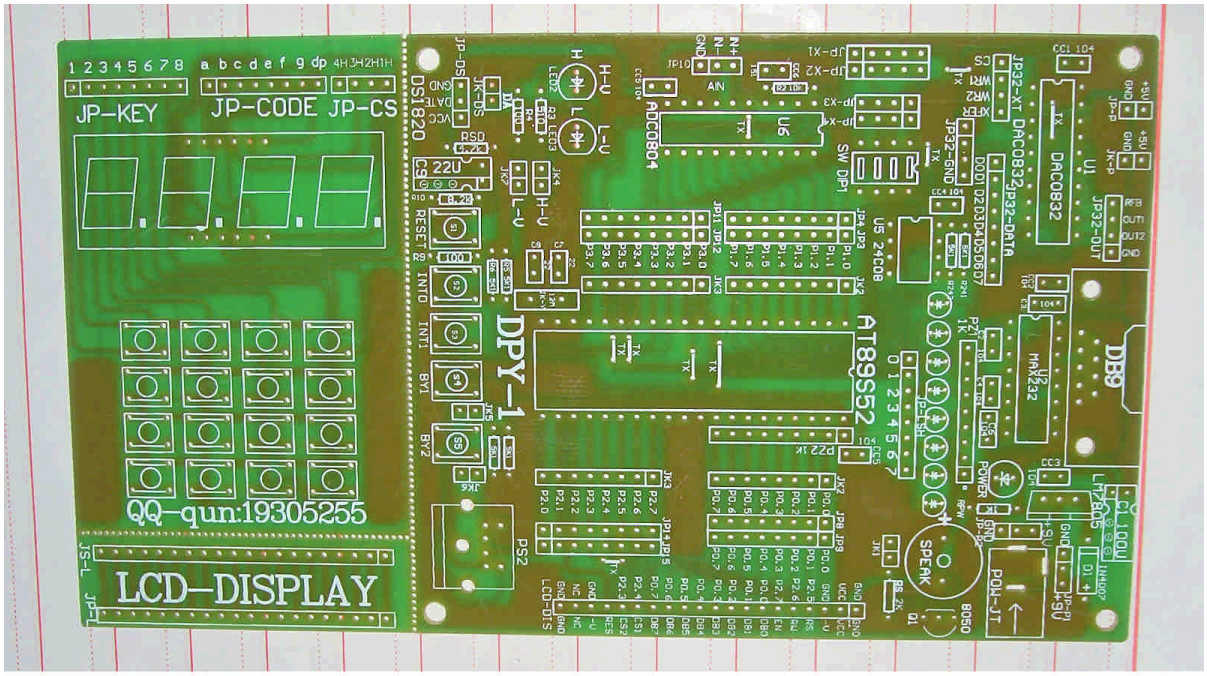
```
//有包含 128X64 液晶显示驱动程序，只有主程序和 PS2 键盘通信的驱动程序。
//请大家到 QQ 群：19305255，资源共享里面去下
//以下程序只有单片机接收 PS2 键盘程序，没有单片机向 PS2 发送命令程序。
//还有键盘数据处理上不是很完善，主要是为用户提供一个参考程序。
//如果把此实验看作是一个系统的话，那么这一个系统就是一个漏洞百出，充满
//Bug 系统。大家在用的时侯可以对它多作改进。
#include<reg52.h>
#include<lcd12864.h>//包含 128X64 驱动程序
#include<hzzi.H>       //包含图片和汉字库
sbit Key_Data =P2^0;//定义 Keyboard 数据端口引脚
sbit Key_CLK=P3^3;     //中断端口，时钟线
static unsigned char KeyV=0X00; //键值
static unsigned char BF = 0; //标识是否有字符被收到
static unsigned char IntNum = 0; //中断次数计数
unsigned char    lie,hang;//lie 为列值, hang 为行值
bit dx=0;                  //大小写标志  dx==1 时大写状态
void Keyboard_out(void) interrupt 2//键盘中断处理  键值存储在  KeyV  中
{
   if ((IntNum>0) && (IntNum <9))
     {
          KeyV = KeyV >> 1; //因键盘数据是低>>高，结合上一句所以右移一位
          if (Key_Data==1)   //当键盘数据线为 1 时
           {KeyV = KeyV | 0x80;}   //存储一位
     }
   IntNum++;     //中断次数加一(中断一次接收一位数据)
   if (IntNum > 10) //中断 11 次后数据发送完毕
     {
          IntNum = 0; //当中断 11 次后表示一帧数据收完，清变量准备下一次接收
          BF = 1;       //标识有字符输入完了
          EA = 0;       //关中断等显示完后再开中
     }
}
void Decode() //键值处理
{
      unsigned char data TempCyc;
       signed char data k;
      TempCyc=KeyV;
       if(BF==1)   //接收完一个有效数据时
      {
        BF=0;   //准备下一次接收
        switch ( TempCyc ) //键值与显示字符的对应关系
          {       //键值//              //对应字符//
            case 0x8A: k=0;    break; //0                  case 0x3C: k=2;    break; //2
            case 0x2C: k=1;    break; //1                  case 0x4C: k=3;    break; //3
```

```
    case 0x4A: k=4;    break; //4                case 0x88: k=24; break; //o
    case 0x5C: k=5;    break; //5                case 0x9A: k=25; break; //p
    case 0x6C: k=6;    break; //6                case 0x2A: k=26; break; //q
    case 0x7A: k=7;    break; //7                case 0x5A: k=27; break; //r
    case 0x7C: k=8;    break; //8                case 0x36: k=28; break; //s
    case 0x8C: k=9;    break; //9                case 0x58: k=29; break; //t
    case 0x38: k=10; break; //a                 case 0x78: k=30; break; //u
    case 0x64: k=11; break; //b                 case 0x54: k=31; break; //v
    case 0x42: k=12; break; //c                 case 0x3A: k=32; break; //w
    case 0x46: k=13; break; //d                 case 0x44: k=33; break; //x
    case 0x48: k=14; break; //e                 case 0x6A: k=34; break; //y
    case 0x56: k=15; break; //f                 case 0x34: k=35; break; //z
    case 0x68: k=16; break; //g                 case 0x98: k=63; break; //;
    case 0x66: k=17; break; //h                 case 0xAA: k=65; break; //=
    case 0x86: k=18; break; //i                 case 0xF2: k=68; break; //+
    case 0x76: k=19; break; //j              case 0x52: k=-16;break; //空格键
    case 0x84: k=20; break; //k                  case  0xB0:  k=101;break;    //
    case 0x96: k=21; break; //l          大小写控制键 Caps lock
    case 0x74: k=22; break; //m                }
    case 0x62: k=23; break; //n

 if(k==101) {dx=!dx;} //按下大小写控制键时  大小写标志取反
if(k==-16) //空格键按下时
 {
   lcd_putchar8x8(lie, hang, -16+'0'); //清除光标
   lcd_putchar8x8(lie+8, hang, 79+'0');//光标移一位
 }
 if(dx==0) //小写状态时显示
 {
   lcd_putchar8x8(lie, hang, k+'0');//小写字符
     lcd_putchar8x8(lie+8, hang, 79+'0');//光标
  }
 if(dx==1)//大写状态时显示
  {
    if(k>9 && k<=35)
        {
          lcd_putchar8x8(lie, hang, (k+26)+'0');//大写字符
          lcd_putchar8x8(lie+8, hang, 79+'0');//光标
        }
        if(k>=0 && k<=9)
        {
          lcd_putchar8x8(lie, hang, (k-10)+'0');//数字键对应的符号
          lcd_putchar8x8(lie+8, hang, 79+'0');//光标
        }
```

```
            }
           if(lie==120)   //写满一行
           {lie=0;hang=hang+1;//换行
           lcd_putchar8x8(lie, hang, 79+'0');}//光标下移
            else {lie=lie+8;}
            EA=1;
       }
}
void delay(int x)     //延时程序
{    int i,j;
     for(i=0;i<600;i++)
     for(j=0;j<x;j++); }

void wlcome()          //开机画面和欢迎界面
{
     lcd_init();   //初始化液晶驱动硬件          hz_disp(lie+24,hang, shu );//'输'
     lcd_clr();     //清屏                      hz_disp(lie+40,hang, ru);//'入'
     Disp_Img(yema);//显示一张图片              hz_disp(lie+56,hang, xi);//'系'
     delay(200);   //延时等待                   hz_disp(lie+72, hang, tong);//'统'
    lie=20;   hang=1;                          delay(200);lcd_clr();
     lcd_clr();                                lie=8; hang=0;
     hz_disp(lie+0, hang, huan);//'欢'          hz_disp(lie+16,hang, qing);// '请'
     hz_disp(lie+16,hang, ying);//'迎'          hz_disp(lie+32,hang, shu);//'输'
     hz_disp(lie+32,hang, shi);//'使'           hz_disp(lie+48,hang, ru);//'入'
     hz_disp(lie+48,hang, yong);//'用'          hz_disp(lie+64,hang, zi);//'字'
     hz_disp(lie+64,hang, DP);//'DP'            hz_disp(lie+80,hang, fu);//'符'
     hz_disp(lie+80,hang, Y);//'Y'           for(lie=0;lie<128;lie++)//画一条直线
     lie=8;hang=4;                                 {set_point(lie,15);}
}
void main()
{
      wlcome() ; //调用开机画面和欢迎界面子程序
      IT1 = 1;   //设外部中断 1 为下降沿触发
     EA = 1;     //开总中断
     EX1=1;      //开中断 1
     hang=2; lie=0;//输入的字符从第 2 行 第 0 列开始显示
     while(1)
      {
          Decode();
          delay(50);
     }
}
```



PCB 板正面俯视图

谢谢对我们的支持

祝大家学习愉快