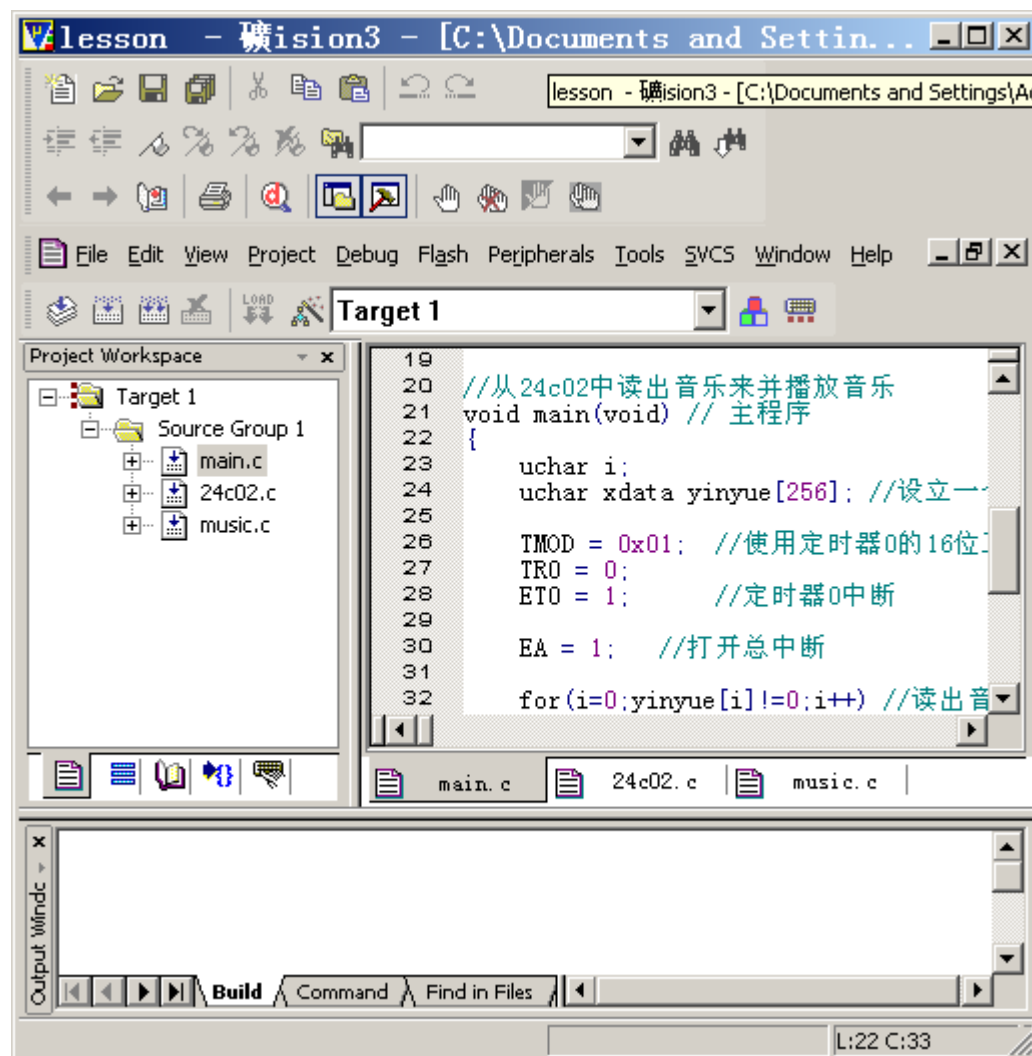


第 30 课，从 24c02 中读出音乐来并播放音乐

这是本入门教程的最后一课，也是最复杂的一个程序。

我们要从 24c02 中读出上一课写入的一个乐谱，并播放出来。这是模拟 mp3 的工作方式。获取外部的存储器的音乐数据并播放，而这个音乐数据是任意的，只要符合播放规则就可以。

本课我们还要学习函数模块的做法。请打开第 30 课的工程：



可以看到，在左边的列表中，有 3 个文件，我们称为 3 个功能模块，main.c, 24c02.c, music.c。我们把对应功能的函数放入其中，这样可以使整个程序工程简洁易懂，维护方便。

那么，我们怎么使各个模块联合工作呢？

答案是，如果在当前模块要调用其他模块里的函数或者变量，在这个模块的前面声明一下就可以了。

如在 main.c 模块的前面，就有：

```
extern uchar Read24c02(uchar address); //声明外部的读 24c02 函数
```

```
extern void play(uchar *songdata); //声明外部的音乐播放函数
```

这样，在 main.c 中就可以直接调用这两个函数了。

如果调用了其他函数中声明的公共变量，也可以这样声明一下就可以了。

在编译的时候，keilC51 会将每个模块分别编译，再一起连接，整个过程不用你操心。

把里面的 error 和 warning 仔细解决掉就可以了。本程序已经编译好了，可以直接使用。

下面分别列出每个模块：

main.c

```
#define uchar unsigned char //定义一下方便使用
#define uint unsigned int
#define ulong unsigned long
#include <reg52.h> //包括一个 52 标准内核的头文件

extern uchar Read24c02(uchar address); //声明外部的读 24c02 函数
extern void play(uchar *songdata); //声明外部的音乐播放函数

char code dx516[3] _at_ 0x003b; //这是为了仿真设置的
```

```
sbit P10=P1^0;
sbit K1= P3^2;
sbit K2= P3^5;
sbit K3= P2^4;
sbit K4= P2^5;
```

//从 24c02 中读出音乐来并播放音乐

void main(void) // 主程序

```
{
    uchar i;
    uchar xdata yinyue[256]; //设立一个缓冲区

    TMOD = 0x01; //使用定时器 0 的 16 位工作模式
    TR0 = 0;
    ET0 = 1; //定时器 0 中断

    EA = 1; //打开总中断

    for(i=0;yinyue[i]!=0;i++) //读出音乐来放到缓冲中
    {
        yinyue[i]=Read24c02(i);
    }
    yinyue[i]=0;

    while(1)
    {
```

```

        play(yinyue); //播放音乐
    }

}

```

24c02.c

```

#define uchar unsigned char //定义一下方便使用
#define uint unsigned int
#define ulong unsigned long
#include <reg52.h> //包括一个 52 标准内核的头文件

#define WriteDeviceAddress 0xa0 //定义器件在 IIC 总线中的地址
#define ReadDviceAddress 0xa1
sbit SCL=P2^7;
sbit SDA=P2^6;

sbit P10=P1^0;
/*
//定时函数
void DelayMs(unsigned int number)
{
    unsigned char temp;
    for(;number!=0;number--)
    {
        for(temp=112;temp!=0;temp--);
    }
}
*/
//开始总线
void Start()
{
    SDA=1;
    SCL=1;
    SDA=0;
    SCL=0;
}

//结束总线
void Stop()
{
    SCL=0;
    SDA=0;
}

```

```

        SCL=1;
        SDA=1;
    }

//发 ACK0
void NoAck()
{
    SDA=1;
    SCL=1;
    SCL=0;
}

//测试 ACK
bit TestAck()
{
    bit ErrorBit;
    SDA=1;
    SCL=1;
    ErrorBit=SDA;
    SCL=0;
    return(ErrorBit);
}

//写入 8 个 bit 到 24c02
Write8Bit(unsigned char input)
{
    unsigned char temp;
    for(temp=8;temp!=0;temp--)
    {
        SDA=(bit)(input&0x80);
        SCL=1;
        SCL=0;
        input=input<<1;
    }
}
/*
//写入一个字节到 24c02 中
void Write24c02(uchar ch,uchar address)
{
    Start();
    Write8Bit(WriteDeviceAddress);
    TestAck();
    Write8Bit(address);
}

```

```

    TestAck();

    Write8Bit(ch);
    TestAck();

    Stop();
    DelayMs(10);
}
*/
//从 24c02 中读出 8 个 bit
uchar Read8Bit()
{
    unsigned char temp,rbyte=0;
    for(temp=8;temp!=0;temp--)
    {
        SCL=1;
        rbyte=rbyte<<1;
        rbyte=rbyte|((unsigned char)(SDA));
        SCL=0;
    }
    return(rbyte);
}

//从 24c02 中读出 1 个字节
uchar Read24c02(uchar address)
{
    uchar ch;

    Start();
    Write8Bit(WriteDeviceAddress);
    TestAck();
    Write8Bit(address);
    TestAck();
    Start();
    Write8Bit(ReadDviceAddress);
    TestAck();
    ch=Read8Bit();
    NoAck();
    Stop();
    return(ch);
}

```

music.c

```

#define uchar unsigned char //定义一下方便使用
#define uint unsigned int
#define ulong unsigned long
#include <reg52.h> //包括一个 52 标准内核的头文件

sbit BEEP=P1^7; //喇叭输出脚
sbit K1= P3^2;
sbit K2= P3^5;
sbit K3= P2^4;
sbit K4= P2^5;

uchar th0_f; //在中断中装载的 T0 的值高 8 位
uchar tl0_f; //在中断中装载的 T0 的值低 8 位

//T0 的值,及输出频率对照表
uchar code freq[36*2]={
    0xA9,0xEF,//00220HZ ,1 //0
    0x93,0xF0,//00233HZ ,1#
    0x73,0xF1,//00247HZ ,2
    0x49,0xF2,//00262HZ ,2#
    0x07,0xF3,//00277HZ ,3
    0xC8,0xF3,//00294HZ ,4
    0x73,0xF4,//00311HZ ,4#
    0x1E,0xF5,//00330HZ ,5
    0xB6,0xF5,//00349HZ ,5#
    0x4C,0xF6,//00370HZ ,6
    0xD7,0xF6,//00392HZ ,6#
    0x5A,0xF7,//00415HZ ,7
    0xD8,0xF7,//00440HZ 1 //12
    0x4D,0xF8,//00466HZ 1# //13
    0xBD,0xF8,//00494HZ 2 //14
    0x24,0xF9,//00523HZ 2# //15
    0x87,0xF9,//00554HZ 3 //16
    0xE4,0xF9,//00587HZ 4 //17
    0x3D,0xFA,//00622HZ 4# //18
    0x90,0xFA,//00659HZ 5 //19
    0xDE,0xFA,//00698HZ 5# //20
    0x29,0xFB,//00740HZ 6 //21
    0x6F,0xFB,//00784HZ 6# //22
    0xB1,0xFB,//00831HZ 7 //23
    0xEF,0xFB,//00880HZ `1
    0x2A,0xFC,//00932HZ `1#
    0x62,0xFC,//00988HZ `2

```

```

    0x95,0xFC,//01046HZ `2#
    0xC7,0xFC,//01109HZ `3
    0xF6,0xFC,//01175HZ `4
    0x22,0xFD,//01244HZ `4#
    0x4B,0xFD,//01318HZ `5
    0x73,0xFD,//01397HZ `5#
    0x98,0xFD,//01480HZ `6
    0xBB,0xFD,//01568HZ `6#
    0xDC,0xFD,//01661HZ `7 //35
};

//定时中断 0,用于产生唱歌频率
timer0() interrupt 1
{
    TL0=tl0_f;TH0=th0_f; //调入预定值
    BEEP=~BEEP; //取反音乐输出 IO
}

//*****
//音乐符号串解释函数
//入口:要解释的音乐符号串,输出的音调串,输出的时长串
changedata(uchar *song,uchar *diao,uchar *jie)
{
    uchar i,i1,j;
    char gaodi; //高低+/-12 音阶
    uchar banyin; //有没有半个升音阶
    uchar yinchang; //音长
    uchar code jie7[8]={0,12,14,16,17,19,21,23}; //C 调的 7 个值

    *diao=*song;
    for(i=0,i1=0;;)
    {
        gaodi=0; //高低=0
        banyin=0; //半音=0
        yinchang=4; //音长 1 拍
        if((*song+i)=='|' || (*song+i)==' ') i++;
        //拍子间隔和一个空格过滤

        switch>(*song+i)
        {
            case '!': gaodi=-12;i++; //低音
            break;

            case '^': gaodi=12;i++; //高音

```

```

        break;
    }

    if(*(song+i)==0) //遇到 0 结束
    {
        *(diao+i1)=0; //加入结束标志 0
        *(jie+i1)=0;
        return;
    }

    j=*(song+i)-0x30; i++; //取出基准音
    j=jie7[j]+gaodi; //加上高低音

yinc: switch(*(song+i))
    {
        case '#': //有半音 j 加一个音阶
            i++;j++;
            goto yinc;

        case '!': //有一个音节加长
            yinchang+=4;
            i++;
            goto yinc;

        case '_': //有一个音节缩短
            yinchang/=2;
            i++;
            goto yinc;

        case ':': //有一个加半拍
            yinchang=yinchang+yinchang/2;
            i++;
            goto yinc;

    }

    *(diao+i1)=j; //记录音符
    *(jie+i1)=yinchang; //记录音长
    i1++;
}

```



```

}
//*****
//奏乐函数
//入口:要演奏的音乐符号串
void play(uchar *songdata)
{
    uchar i,c,j=0;
    uint n;
    uchar xdata diaodata[112]; //音调缓冲
    uchar xdata jiedata[112]; //音长缓冲

    changedata(songdata,diaodata,jiedata); //解释音乐符号串
    TR0=1;
    for(i=0;diaodata[i]!=0;i++) //逐个符号演奏
    {
        tl0_f=freq[diaodata[i]*2]; //取出对应的定时值送给 T0
        th0_f=freq[diaodata[i]*2+1];
        for(c=0;c<jiedata[i];c++) //按照音长延时
        {
            for(n=0;n<32000;n++);
            if(!K1||!K2||!K3||!K4)//发现按键,立即退出播放
            {
                TR0=0;
                return;
            }
        }
        TR0=0;
        for(n=0;n<500;n++); //音符间延时

        TR0=1;
    }
    TR0=0;
}

```

请打开工程，好好体验一下分模块的好处，并仔细查看一下各个函数模块，试着编译一次。

点全速运行，看结果。

可以看到，上一课预先写入的“老鼠爱大米”被不断地播放。

作业：返回第 29 课，写入一首新的音乐。再回到第 30 课，播放你写入的音乐。

结束语：

这 30 个 C51 课程，只是在一个小小的试验板实现，试验板上只有 4 个按钮，4 个 LED，1 个蜂鸣器，1 个 24c02。但是我们却学到了各种复杂的控制和编程方法。我们已经熟悉了 C51 的编程，甚至可以进行已经学习过的资源的程序开发。相信你已经兴趣倍增，对于更加深入地学习单片机系统充满信心！

由于您已经在学习本系列课程中形成了基础，也就是已经入门并基本可以熟练操作了，对于本系列课程没有涉及到的资源，如串口、总线等，你都可以在日后信手拈来，自己在探索中去研究。

本教程由大虾电子网 (<http://www.daxia.com>) 创办人聂小猛 (网名丁丁) 在 2006 年 6 月足球世界杯期间编写，版权所有，请勿用在商业场合！

大虾电子网简介：

大虾电子网前身 51 单片机世界，在 2000 年创办。本站拥有国内人气最旺的单片机类专业技术论坛，并拥有大量资料，访问量在国内同类网站名列前茅。2006 年计划增加开放性商城板块和大虾博客板块，欢迎各界电子网友前来访问。