

第 25 课，乐谱方式输入的音乐播放,仙剑奇侠传

这一课开始，我们就要听到美妙的音乐了，这一课，我们可以听到演奏仙剑奇侠传的乐谱。

这一课的程序，增加了 2 个比较复杂的函数，一个乐谱解释函数，一个音乐播放函数。我们音乐仙剑奇侠传的乐谱以一个我们自己定义的乐谱形式写好，作为一个预定义的字符串。再通过乐谱解释函数解释为“音符频率的序号”和“音符播放的时间”两个数组，在音乐播放函数中，就将音符频率的序号数组对应的频率送入定时器预置数中，再延时对应音符播放的时间。这样音乐就播放出来了。

仙剑奇侠传的乐谱：

```
"|3_3_3_2_3-|2_3_2_2_,6,6_,7_|12_1_,7,6_,5_|,6---|"
"3_3_3_2_3.6_|5_6_5_5_22_3_|45_4_32_1_|3.--3_|"
"67_6_55_3_|5--3_5_|26_5_32_3_|3---|"
"26_6_6-|16_6_66_7_|17_6_76_7_|3.--3_|"
"67_6_55_3_|5--3_5_|67_6_76_7_|3---|"
"26_6_6-|16_6_66_7_|17_6_7.5_|6---|"
```

乐谱书写规则：

1 2 3 4 5 6 7 为 7 个基本音阶

前面加逗号','表示这是低音

前面加上点号'^'表示这是高音

后面加'#',表示这个音符升半个音阶

后面加'!',表示这个音符要再加长自身一半的延时

后面加一个或多个'_',每个表示延时一拍

后面加一个或多个'|',表示这个音符要缩短自身一半的时长，最多支持 2 个'|'。

这些规则对一般的乐谱都可以应付得来了。

下面看程序：

```
#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
```

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

```
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; //调入预定时值
```

```

    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
    {
        t10_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++);
        TR0=0;
        for(n=0;n<500;n++);    //音符间延时
        TR0=1;
    }
    TR0=0;
}

```

//仙剑

```

uchar code xianjian[]={
"|3_3_3_2_3_|2_3_2_2_,6,6_,7_|12_1_,7,6_,5_|,6---|"
"3_3_3_2_3.6_|5_6_5_5_22_3_|45_4_32_1_|3.--3_|"
"67_6_55_3_|5--3_5_|26_5_32_3_|3---|"
"26_6_6_|16_6_66_7_|17_6_76_7_|3.--3_|"
"67_6_55_3_|5--3_5_|67_6_76_7_|3---|"
"26_6_6_|16_6_66_7_|17_6_7.5_|6---|"
};

```

//乐谱方式输入的音乐播放,仙剑奇侠传

```

void main(void)    // 主程序
{
    TMOD = 0x01;    //使用定时器 0 的 16 位工作模式
    TR0 = 0;
    ET0 = 1;
    EA = 1;

    while(1)
    {
        play(xianjian);
    }
}

```

这里最复杂是乐谱解释函数,是逐个字符解释的。基本上以下过程:遇到拍子分隔符和空格跳过,判断是否高低音,读音符,调整为高低音音符,读音符后的升半个音符的“#”,读延长音“-”“.”,读缩短一半音长的“_”,字符串结束符“0x00”。请仔细领会这个函数。

奏乐函数就比较简单,基本上就是从数组中取出音符和时长,送入定时器预置数,再延时即可。在每个音符播放前后,用 TR0 控制是否输出音乐,每个音符之间也有短暂静音,以使音乐更为清晰。

在本程序中,播放音乐函数中,我们使用了 xdata 的空间的 RAM,这是因为乐谱的数据需要比较多的内存,data 和 idata 空间已经放不下了的原因。由于 DX516 内部是有 768 个字 XRAM 可以直接仿真使用的。所以我们仿真不会有任何问题。但是如果你把这个程序烧写到一片没有 XRAM 的芯片中,比如 at89c52 之类,就会出现无法运行的现象。在使用没

有 XRAM 的 51 芯片时，如果使用了 XRAM，则要在总线上外加一个内存芯片，比如 62256 之类。

完全看懂了程序之后，请编译运行，观察结果。按全速，可以听到美妙的仙剑音乐从蜂鸣器中传出，真是太奇妙了！

作业：

编写一个其他的你熟悉的比较简单的乐谱，替换掉仙剑音乐，做本试验。：由于在播放函数里只定义 112 个音符符的空间，注意您的乐谱不要超过这个数目。如果需要调大，注意编译后不要超过仿真器内部的 768 个 XRAM 空间。