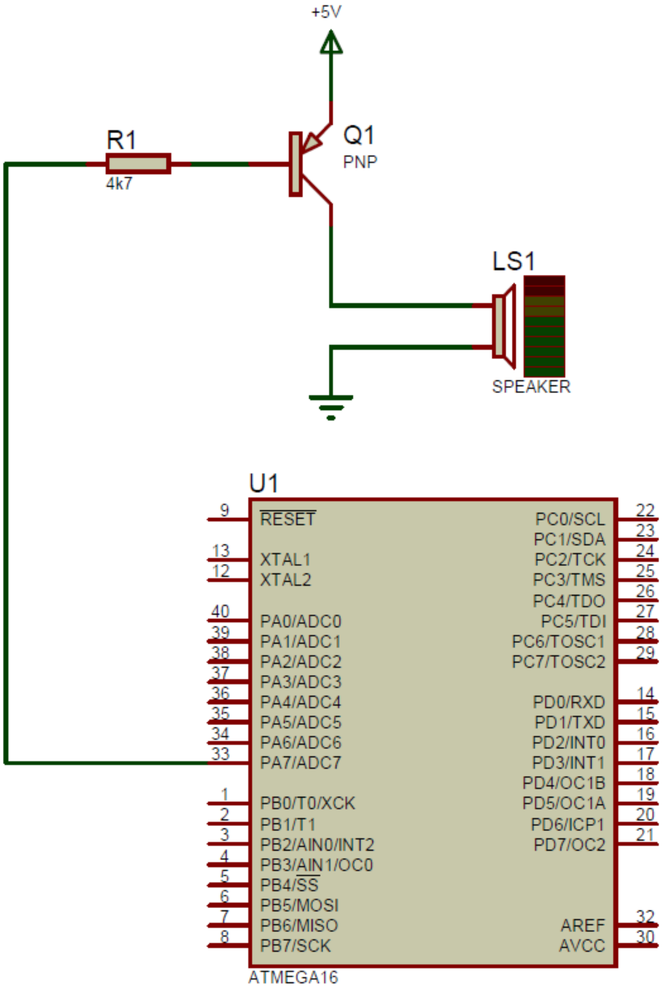
实验 8：小喇叭唱歌实验

1. 试验描述：

本实验使用 PA7 控制无源蜂鸣器，翻转电平使蜂鸣器发声，改变翻转的频率使蜂鸣器发出不同音调的声音，并且通过延时控制节拍的时间。其中，通过定时器 1 设置不同的计数值，并且再溢出中断时翻转电平，从而使 PA7 的电平以不同的频率翻转。控制节拍时间的延时使用软延时。

2. 系统框图：

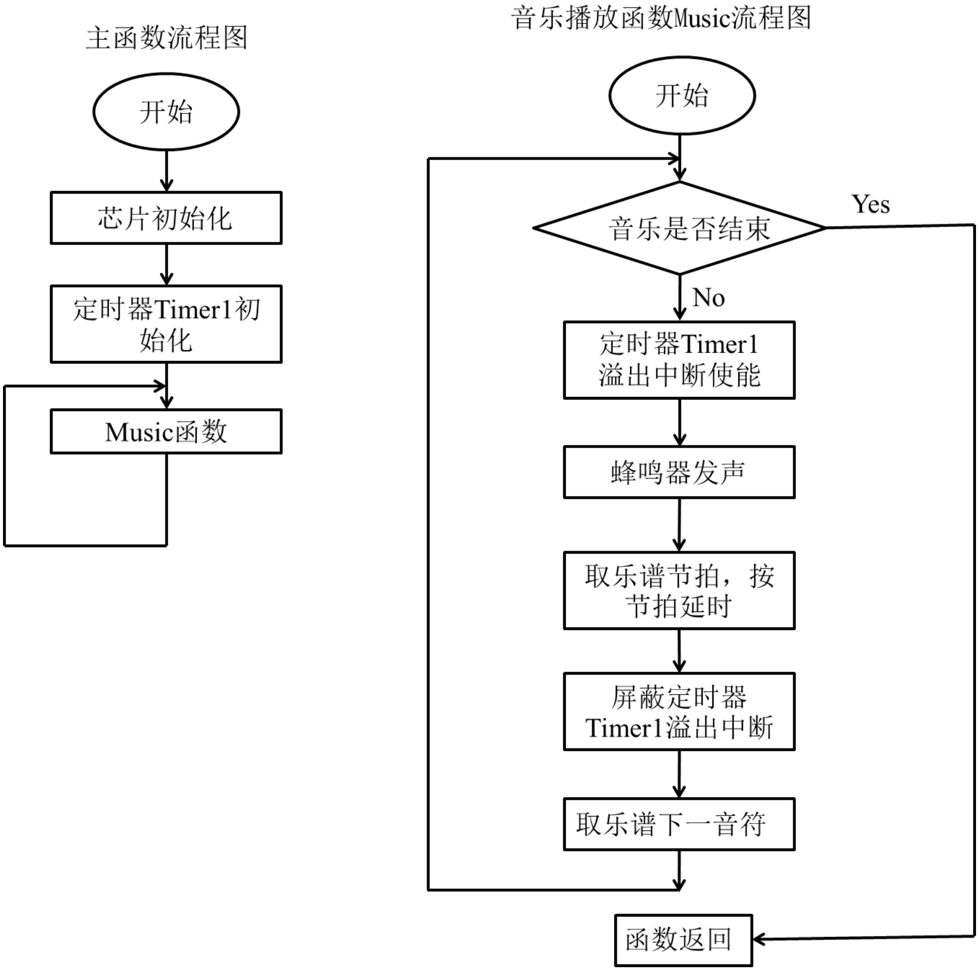
- ➢ 硬件电路



➢ 元件清单

单片机 ATmega16	电阻 4.7k 欧姆	PNP 晶体管
蜂鸣器		

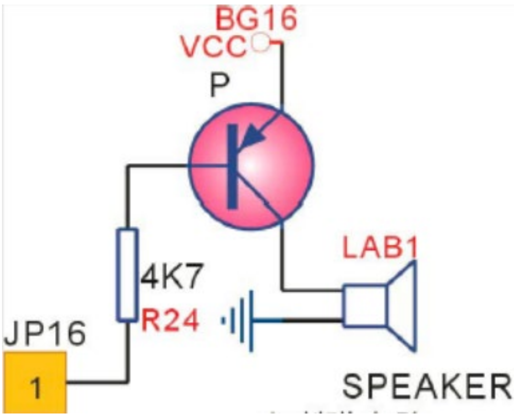
➢ 软件流程



➢ 蜂鸣器

蜂鸣器是一种一体化结构的电子讯响器，采用直流电压供电，可以分为有源蜂鸣器和无源蜂鸣器两种。这里的“源”是指振荡源，也就是有源蜂鸣器内部带振荡器，只要通电就会叫；无源蜂鸣器的内部不带有振荡器，所有用直流信号无法令其鸣叫，必须使用 2K～5K 的方波驱动。

由于蜂鸣器的工作电流一般比较大，单片机的 I/O 口无法直接驱动，所以要利用放大电路来驱动，一般使用三极管放大电流达到驱动目的。



XL2200 4 CORE AVR 实验箱 小喇叭电路图

➢ 歌曲音频

对于一首歌曲由音调和节拍两个主要的元素组成。对于蜂鸣器来说，频率的高低决定了音调的高低。所有不同频率的信号都是从同一个基准频率分频得来的。由于音阶频率多为非整数，而分频系数又不能为小数，故必须将计算得到的分频数四舍五入取整。若基准频率过低，则由于分频比太小，四舍五入取证后的误差较大。若基准频率过高，虽然误差变小，但分频数将变大。实际的设计在尽量减小频率误差的前提下去合适的基准频率。

给蜂鸣器输入相应的频率，可以使其发出低音、中音、高音的 do～xi 的声音。将其按照音乐演奏的规律组合，便可以得到所需要的乐曲。

3. 程序代码：

➢ ICC 程序

bell.h

```
#include <iom16v.h>
#include <macros.h>

#define uchar unsigned char
#define uint unsigned int

void DelayMs(uint ms)
```

```c
{
  uint i, j;
  for (i = 0; i < ms; i++)
    for (j = 0; j < 1000; j++)
      ;
}

void BoardInit(void)
{
  DDRA = 0x80;
}

/*******蜂鸣器音乐常量*********/
#define BEEP PA7
//计时值=65535-8000000/8/2/频率
//音名 计时值 频率 Hz
#define DO_L 63627  //262
#define DOA_L 63731 //277
#define RE_L 63835  //294
#define REA_L 63928 //311
#define MI_L 64021  //330
#define FA_L 64103  //349
#define FAA_L 64185 //370
#define SO_L 64270  //392
#define SOA_L 64331 //415
#define LA_L 64400  //440
#define LAA_L 64463 //466
#define TI_L 64524  //494
#define DO 64580    //523
#define DOA 64633   //554
#define RE 64684    //587
#define REA 64732   //622
#define MI 64777    //659
#define FA 64820    //698
#define FAA 64860   //740
#define SO 64898    //784
#define SOA 64934   //831
#define LA 64968    //880
#define LAA 65000   //932
#define TI 65030    //988
#define DO_H 65058  //1046
#define DOA_H 65085 //1109
#define RE_H 65110  //1175
#define REA_H 65134 //1245
```

```
#define MI_H 65157  //1318
#define FA_H 65178  //1397
#define FAA_H 65198 //1480
#define SO_H 65217  //1568
#define SOA_H 65235 //1661
#define LA_H 65252  //1760
#define LAA_H 65268 //1865
#define TI_H 65283  //1976
#define ZERO 0      //休止符
```

main.c

```
#include <iom16v.h> //包含型号头文件
#include <macros.h> //包含"位"操作头文件
#include <stdio.h>  //标准输入输出头文件
#include "bell.h"   //包含自定义常量头文件

#pragma interrupt_handler Timer1_Ov:9
/*-------------两只老虎-------------*/
const uchar MusicTable2[77] = {
    13, 2, 15, 2, 17, 2, 13, 1, 0, 1,
    13, 2, 15, 2, 17, 2, 13, 1, 0, 1,
    17, 2, 18, 2, 20, 2, 0, 2,
    17, 2, 18, 2, 20, 2, 0, 2,
    20, 1, 22, 1, 20, 1, 18, 1, 17, 2, 13, 2,
    20, 1, 22, 1, 20, 1, 18, 1, 17, 2, 13, 2,
    15, 2, 8, 2, 13, 2, 0, 2,
    15, 2, 8, 2, 13, 2, 0, 2,
    0xff};
/*-----------------新年好------------------*/
const uchar MusicTable1[129] = {
    13, 1, 13, 1, 13, 2, 8, 2, //音符,拍数,
    17, 1, 17, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 20, 2, 20, 2,
    18, 1, 17, 1, 15, 2, 0, 2,
    15, 1, 17, 1, 18, 2, 18, 2,
    17, 1, 15, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 15, 2, 8, 2,
    12, 1, 15, 1, 13, 2, 0, 2,
    13, 1, 13, 1, 13, 2, 8, 2, //音符,拍数,
    17, 1, 17, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 20, 2, 20, 2,
    18, 1, 17, 1, 15, 2, 0, 2,
    15, 1, 17, 1, 18, 2, 18, 2,
    17, 1, 15, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 15, 2, 8, 2,
    12, 1, 15, 1, 13, 2, 0, 2,
    0xff};

/***************astronomia*************/
const uchar Astronomia[] = {
    18, 1, 18, 1, 18, 1, 18, 1, 22, 1, 22, 1, 22, 1, 22, 1,
    20, 1, 20, 1, 20, 1, 20, 1, 25, 1, 25, 1, 25, 1, 25, 1,
    27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1,
    20, 1, 18, 1, 17, 1, 13, 1, 15, 1, 0, 1, 15, 1, 22, 1,
    20, 1, 0, 1, 18, 1, 0, 1, 17, 1, 0, 1, 17, 1, 17, 1,
    20, 1, 0, 1, 18, 1, 17, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 22, 1,
    20, 1, 0, 1, 18, 1, 0, 1, 17, 1, 0, 1, 17, 1, 17, 1,
    20, 1, 0, 1, 18, 1, 17, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 18, 1, 18, 1, 18, 1, 18, 1,
    22, 1, 22, 1, 22, 1, 22, 1, 20, 1, 20, 1, 20, 1, 20, 1,
    25, 1, 25, 1, 25, 1, 25, 1, 27, 1, 27, 1, 27, 1, 27, 1,
    27, 1, 27, 1, 27, 1, 27, 1, 20, 1, 18, 1, 17, 1, 13, 1,
    15, 1, 0, 1, 15, 1, 22, 1, 20, 1, 0, 1, 18, 1, 0, 1,
    17, 1, 0, 1, 17, 1, 17, 1, 20, 1, 0, 1, 18, 1, 17, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 22, 1, 20, 1, 0, 1, 18, 1, 0, 1,
    17, 1, 0, 1, 17, 1, 17, 1, 20, 1, 0, 1, 18, 1, 17, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    0xff};

//0,   1,   #1,   2,   #2,   3,   4,   #4,   5,   #5,   6,   #6,   7,
uint ToneTable[37] = {
    ZERO, DO_L, DOA_L, RE_L, REA_L, MI_L, FA_L, FAA_L, SO_L, SOA_L, LA_L, LAA_L, TI_L,
    DO, DOA, RE, REA, MI, FA, FAA, SO, SOA, LA, LAA, TI,
    DO_H, DOA_H, RE_H, REA_H, MI_H, FA_H, FAA_H, SO_H, SOA_H, LA_H, LAA_H, TI_H};
uint tone;

void Timer1_Ov(void)
{
  if (tone) //若不是休止符，则发声
  {
    TCNT1 = tone;       //计数值装入寄存器
    PORTA ^= BIT(BEEP); //蜂鸣器接口电平翻转
  }
}

void Timer1Init(void)
{
  TCCR1A = 0x00; //普通端口操作
  TCCR1B = 0x01; //8 分频
  SREG |= 0x80;  //开放全局中断
}
/********************************************
* 函数名称: Music
* 功能: 完成整曲的音乐演奏
* 参数: pmusic--曲谱数组指针
* 返回值  : 无
* ********************************************/
void Music(const uchar *pMusic)
{
  while (*pMusic != 0xFF) //0xFF 为音乐结尾符
  {
    TIMSK = 0x04;              //Timer1 溢出中断使能
    tone = ToneTable[*pMusic]; //取音调频率
    TCNT1 = tone;              //将频率值对应的计数值写入计时器,开始发声
    pMusic++;                  //乐谱音符指针+1 ,取拍数
    DelayMs((*pMusic) * 10);   //按拍数延时
    DelayMs((*pMusic) * 10);
    TIMSK = 0x00; //发声结束 ,屏蔽 Timer1 溢出中断
    pMusic++;     //乐谱音符指针+1 , 取下一音符
  }
  DelayMs(1000); //曲谱结束,等待
}
/********************************************
* 函数名称: main
* 功能: 演奏指定的音乐
* 参数: 无
* 返回值  : 无
* *********************************************/
void main(void)
{
  BoardInit();  //初始化开发板/
  Timer1Init(); //Timer1 初始化
  while (1)
  {
    //Music(Astronomia);
    Music(MusicTable2); //循环演奏歌曲（改变参数 MusicTable2 可以变换歌曲）
  }
}
```

CVAVR 程序

bell.h

```
#include <mega16.h>

#define uchar unsigned char
#define uint unsigned int

void DelayMs(uint ms)
{
  uint i, j;
  for (i = 0; i < ms; i++)
    for (j = 0; j < 1000; j++)
      ;
}

void BoardInit(void)
{
  DDRA = 0x80;
}


//计时值=65535-8000000/8/2/频率
//音名 计时值 频率 Hz
#define DO_L 63627  //262
#define DOA_L 63731 //277
#define RE_L 63835  //294
#define REA_L 63928 //311
#define MI_L 64021  //330
#define FA_L 64103  //349
#define FAA_L 64185 //370
#define SO_L 64270  //392
#define SOA_L 64331 //415
#define LA_L 64400  //440
#define LAA_L 64463 //466
#define TI_L 64524  //494
#define DO 64580    //523
#define DOA 64633   //554
#define RE 64684    //587
#define REA 64732   //622
```

```
#define MI 64777    //659
#define FA 64820    //698
#define FAA 64860   //740
#define SO 64898    //784
#define SOA 64934   //831
#define LA 64968    //880
#define LAA 65000   //932
#define TI 65030    //988
#define DO_H 65058  //1046
#define DOA_H 65085 //1109
#define RE_H 65110  //1175
#define REA_H 65134 //1245
#define MI_H 65157  //1318
#define FA_H 65178  //1397
#define FAA_H 65198 //1480
#define SO_H 65217  //1568
#define SOA_H 65235 //1661
#define LA_H 65252  //1760
#define LAA_H 65268 //1865
#define TI_H 65283  //1976
#define ZERO 0      //休止符
```

main.c

```
#include <mega16.h> //包含型号头文件
#include "bell.h"   //包含自定义常量头文件

/*-------------两只老虎-------------*/
flash uchar MusicTable2[77] = {
    13, 2, 15, 2, 17, 2, 13, 1, 0, 1,
    13, 2, 15, 2, 17, 2, 13, 1, 0, 1,
    17, 2, 18, 2, 20, 2, 0, 2,
    17, 2, 18, 2, 20, 2, 0, 2,
    20, 1, 22, 1, 20, 1, 18, 1, 17, 2, 13, 2,
    20, 1, 22, 1, 20, 1, 18, 1, 17, 2, 13, 2,
    15, 2, 8, 2, 13, 2, 0, 2,
    15, 2, 8, 2, 13, 2, 0, 2,
    0xff};
/*-----------------新年好-----------------*/
flash uchar MusicTable1[129] = {
    13, 1, 13, 1, 13, 2, 8, 2, //音符,拍数,
    17, 1, 17, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 20, 2, 20, 2,
    18, 1, 17, 1, 15, 2, 0, 2,
    15, 1, 17, 1, 18, 2, 18, 2,
    17, 1, 15, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 15, 2, 8, 2,
    12, 1, 15, 1, 13, 2, 0, 2,
    13, 1, 13, 1, 13, 2, 8, 2, //音符,拍数,
    17, 1, 17, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 20, 2, 20, 2,
    18, 1, 17, 1, 15, 2, 0, 2,
    15, 1, 17, 1, 18, 2, 18, 2,
    17, 1, 15, 1, 17, 2, 13, 2,
    13, 1, 17, 1, 15, 2, 8, 2,
    12, 1, 15, 1, 13, 2, 0, 2,
    0xff};

/***************astronomia*************/
flash uchar Astronomia[] = {
    18, 1, 18, 1, 18, 1, 18, 1, 22, 1, 22, 1, 22, 1, 22, 1,
    20, 1, 20, 1, 20, 1, 20, 1, 25, 1, 25, 1, 25, 1, 25, 1,
    27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1, 27, 1,
    20, 1, 18, 1, 17, 1, 13, 1, 15, 1, 0, 1, 15, 1, 22, 1,
    20, 1, 0, 1, 18, 1, 0, 1, 17, 1, 0, 1, 17, 1, 17, 1,
    20, 1, 0, 1, 18, 1, 17, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    20, 1, 0, 1, 18, 1, 17, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 22, 1,
    20, 1, 0, 1, 18, 1, 0, 1, 17, 1, 0, 1, 17, 1, 17, 1,
    20, 1, 0, 1, 18, 1, 17, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 15, 1, 0, 1, 15, 1, 30, 1,
    29, 1, 30, 1, 29, 1, 30, 1, 18, 1, 18, 1, 18, 1, 18, 1,
    22, 1, 22, 1, 22, 1, 22, 1, 20, 1, 20, 1, 20, 1, 20, 1,
    25, 1, 25, 1, 25, 1, 25, 1, 27, 1, 27, 1, 27, 1, 27, 1,
    27, 1, 27, 1, 27, 1, 27, 1, 20, 1, 18, 1, 17, 1, 13, 1,
    15, 1, 0, 1, 15, 1, 22, 1, 20, 1, 0, 1, 18, 1, 0, 1,
    17, 1, 0, 1, 17, 1, 17, 1, 20, 1, 0, 1, 18, 1, 17, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 22, 1, 20, 1, 0, 1, 18, 1, 0, 1,
    17, 1, 0, 1, 17, 1, 17, 1, 20, 1, 0, 1, 18, 1, 17, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    15, 1, 0, 1, 15, 1, 30, 1, 29, 1, 30, 1, 29, 1, 30, 1,
    0xff};

//0,   1,   #1,   2,   #2,   3,   4,   #4,   5,   #5,   6,   #6,   7,
uint ToneTable[37] = {
    ZERO, DO_L, DOA_L, RE_L, REA_L, MI_L, FA_L, FAA_L, SO_L, SOA_L, LA_
L, LAA_L, TI_L,
    DO, DOA, RE, REA, MI, FA, FAA, SO, SOA, LA, LAA, TI,
    DO_H, DOA_H, RE_H, REA_H, MI_H, FA_H, FAA_H, SO_H, SOA_H, LA_H, LAA
_H, TI_H};
uint tone;

interrupt[TIM1_OVF] void Timer1OVFt(void) //定时器 1 溢出中断
{
  if (tone) //若不是休止符，则发声
  {
    TCNT1 = tone;                             //计数值装入寄存器
    PORTA = (~PORTA & 0x80) | (PORTA & 0x7f); //蜂鸣器接口电平翻转
  }
}

void Timer1Init(void)
{
  TCCR1A = 0x00; //普通端口操作
  TCCR1B = 0x01; //8 分频
#asm("sei");     //开放全局中断
}
/********************************************
* 函数名称: Music
* 功能: 完成整曲的音乐演奏
* 参数: pmusic--曲谱数组指针
* 返回值  : 无
* *******************************************/
void Music(flash uchar *pMusic)
{
  while (*pMusic != 0xFF) //0xFF 为音乐结尾符
  {
    TIMSK = 0x04;               //Timer1 溢出中断使能
    tone = ToneTable[*pMusic]; //取音调频率
    TCNT1 = tone;               //将频率值对应的计数值写入计时器,开始发声
    pMusic++;                   //乐谱音符指针+1 ,取拍数
    DelayMs((*pMusic) * 10);    //按拍数延时
    DelayMs((*pMusic) * 10);
    TIMSK = 0x00; //发声结束 ,屏蔽 Timer1 溢出中断
    pMusic++;      //乐谱音符指针+1 , 取下一音符
  }
  DelayMs(1000); //曲谱结束,等待
}
/********************************************
* 函数名称: main
* 功能: 演奏指定的音乐
* 参数: 无
* 返回值  : 无
* ********************************************/
void main(void)
{
  BoardInit();  //初始化开发板/
  Timer1Init(); //Timer1 初始化

  while (1)
  {
    //Music(Astronomia);
    Music(MusicTable2); //循环演奏歌曲（改变参数 MusicTable2 可以变换歌曲）
  }
}
```

4. 仿真结果：

两只老虎	🔊
新年好	🔊
Astronomia	🔊