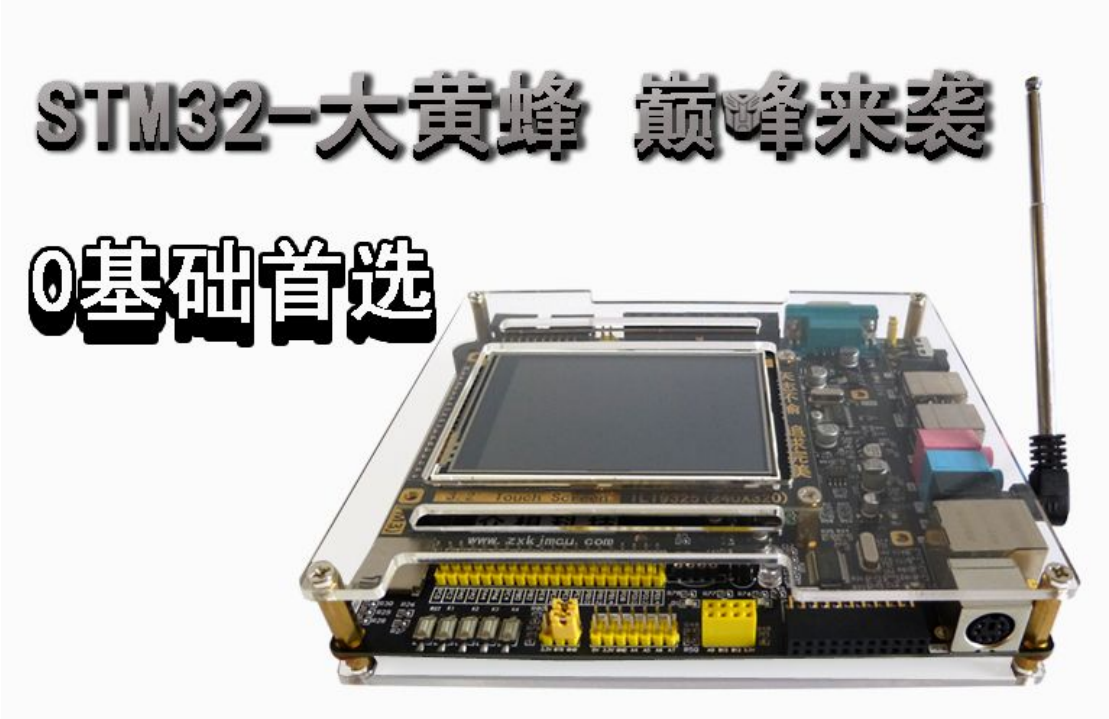
学 ARM 从 STM32 开始

STM32 开发板库函数教程--实战篇



官方网站：http://www.zxkjmcu.com
官方店铺：http://zxkjmcu.taobao.com
官方论坛：http://bbs.zxkjmcu.com
刘洋课堂：http://school.zxkjmcu.com

4.2 蜂鸣器发声实验

4.2.1 概述

本节给大家实现怎样用 STM32 驱动蜂鸣器发声和 Systick 定时器的使用，通过设置 Systick 定时器使蜂鸣器非常精确的按照设计的时间发声。在做实验之前我们要先了解蜂鸣器的结构与原理。

4.2.1.1 蜂鸣器概述

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，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件。蜂鸣器主要分为压电式蜂鸣器和电磁式蜂鸣器两种类型。蜂鸣器在电路中用字母“H”或“HA”（旧标准用“FM”、“LB”、“JD”等）表示。

4.2.1.2 结构原理

1.压电式蜂鸣器:压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配器及共鸣箱、外壳等组成。有的压电式蜂鸣器外壳上还装有发光二极管。多谐振荡器由晶体管或集成电路构成。当接通电源后(1.5~15V 直流工作电压),多谐振荡器起振,输出 1.5~2.5kHZ 的音频信号，阻抗匹配器推动压电蜂鸣片发声。压电蜂鸣片由锆钛酸铅或铌镁酸铅压电陶瓷材料制成。在陶瓷片的两面镀上银电极，经极化和老化处理后，再与黄铜片或不锈钢片粘在一起。

2.电磁式蜂鸣器: 电磁式蜂鸣器由振荡器、电磁线圈、磁铁、振动膜片

及外壳等组成。接通电源后，振荡器产生的音频信号电流通过电磁线圈，使电磁线圈产生磁场。振动膜片在电磁线圈和磁铁的相互作用下，周期性地振动发声。

4.2.1.3 制作工艺

(1)制备电磁铁 M:在长约 6 厘米的铁螺栓上绕 100 圈导线,线端留下 5 厘米作引线,用透明胶布把线圈粘好,以免线圈松开,再用胶布把它粘在一个盒子上,电磁铁就做好了.

(2)制备弹片 P：从铁罐头盒上剪下一条宽约 2 厘米的长铁片，弯成直角，把电磁铁的一条引线接在弹片上，再用胶布把弹片紧贴在木板上.

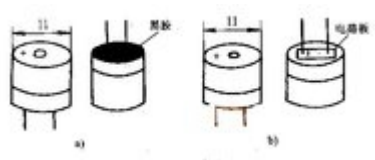
(3)用曲别针做触头 Q，用书把曲别针垫高，用胶布粘牢，引出一条导线。

(4)调节 M 与 P 之间的距离(通过移动盒子)，使电磁铁能吸引弹片，调节触点与弹片之间的距离，使它们能恰好接触，通电后就可以听到蜂鸣声。

4.2.1.4 分类

教你区分有源蜂鸣器和无源蜂鸣器：

现在市场上出售的一种小型蜂鸣器因其体积小(直径只有 11mm)、重量轻、价格低、结构牢靠，而广泛地应用在各种需要发声的电器设备、电子制作和单片机等电路中。有源蜂鸣器和无源蜂鸣器的外观如图 a、b 所示。



图：有源和无源蜂鸣器的外观

a 有源 b)无源。从图 a、b 外观上看，两种蜂鸣器好像一样，但仔细看，两者的高度略有区别，有源蜂鸣器 a，高度为 9mm，而无源蜂鸣器 b 的高度为 8mm。如将两种蜂鸣器的引脚都朝上放置时，可以看出有绿色电路板的一种是无源蜂鸣器，没有电路板而用黑胶封闭的一种是有源蜂鸣器。

进一步判断有源蜂鸣器和无源蜂鸣器，还可以用万用表电阻档 Rx1 档测试:用黑表笔接蜂鸣器 "-"引脚，红表笔在另一引脚上来回碰触，如果触发出咔、咔声的且电阻只有 8Ω(或 16Ω)的是无源蜂鸣器;如果能发出持续声音的，且电阻在几百欧以上的，是有源蜂鸣器。

有源蜂鸣器直接接上额定电源(新的蜂鸣器在标签上都有注明)就可连续发声;而无源蜂鸣器则和电磁扬声器一样，需要接在音频输出电路中才能发声。

有源[1]蜂鸣器与无源蜂鸣器的区别：

注意：这里的“源”不是指电源，而是指震荡源。也就是说，有源蜂鸣器内部带震荡源，所以只要一通电就会叫而无源内部不带震荡源，所以如果用直流信号无法令其鸣叫。必须用 2K-5K 的方波去驱动它有源蜂鸣器往往比无源的贵，就是因为里面多个震荡电路

无源蜂鸣器的优点是：

1.便宜

2. 声音频率可控，可以做出“多来米发索拉西”的效果

3. 在一些特例中，可以和 LED 复用一个控制口

有源蜂鸣器的优点是：程序控制方便。

4.2.1.5 驱动模块

在单片机应用的设计上，很多方案都会用到蜂鸣器，大部分都是使用蜂鸣器

来做提示或报警，比如按键按下、开始工作、工作结束或是故障等等。这里对单片机在蜂鸣器驱动上的应用作一下描述。

4.2.1.6 驱动方式

由于自激蜂鸣器是直流电压驱动的，不需要利用交流信号进行驱动，只需对驱动口输出驱动电平并通过三极管放大驱动电流就能使蜂鸣器发出声音，很简单，这里就不对自激蜂鸣器进行说明了。这里只对必须用 1/2duty 的方波信号进行驱动的他激蜂鸣器进行说明。

单片机驱动他激蜂鸣器的方式有两种：一种是 PWM 输出口直接驱动，另一种是利用 I/O 定时翻转电平产生驱动波形对蜂鸣器进行驱动。PWM 输出口直接驱动是利用 PWM 输出口本身可以输出一定的方波来直接驱动蜂鸣器。在单片机的软件设置中有几个系统寄存器是用来设置 PWM 口的输出的，可以设置占空比、周期等等，通过设置这些寄存器产生符合蜂鸣器要求的频率的波形之后，只要打开 PWM 输出，PWM 输出口就能输出该频率的方波，这个时候利用这个波形就可以驱动蜂鸣器了。比如频率为 2000Hz 的蜂鸣器的驱动，可以知道周期为 500μs，这样只需要把 PWM 的周期设置为 500μs，占空比电平设置为 250μs，就能产生一个频率为 2000Hz 的方波，通过这个方波再利用三极管就可以去驱动这个蜂鸣器了。

而利用 I/O 定时翻转电平来产生驱动波形的方式会比较麻烦一点，必须利用定时器来做定时，通过定时翻转电平产生符合蜂鸣器要求的频率的波形，这个波形就可以用来驱动蜂鸣器了。比如为 2500Hz 的蜂鸣器的驱动，可以知道周期为 400μs，这样只需要驱动蜂鸣器的 I/O 口每 200μs 翻转一次电平就可以产生一个频率为 2500Hz，占空比为 1/2duty 的方波，再通过三

极管放大就可以驱动这个蜂鸣器了。

4.2.1.7 驱动电路

由于蜂鸣器的工作电流一般比较大，以致于单片机的 I/O 口是无法直接驱动，所以要利用放大电路来驱动，一般使用三极管来放大电流就可以了。蜂鸣器驱动电路一般都包含以下几个部分：一个三极管、一个蜂鸣器、一个续流二极管和一个电源滤波电容。

1.蜂鸣器

发声元件，在其两端施加直流电压(有源蜂鸣器)或者方波(无源蜂鸣器)就可以发声，其主要参数是外形尺寸、发声方向、工作电压、工作频率、工作电流、驱动方式(直流/方波)等。这些都可以根据需要来选择。

2.续流二极管

蜂鸣器本质上是一个感性元件，其电流不能瞬变，因此必须有一个续流二极管提供续流。否则，在蜂鸣器两端会产生几十伏的尖峰电压，可能损坏驱动三极管，并干扰整个电路系统的其它部分。

3.滤波电容

滤波电容 C1 的作用是滤波，滤除蜂鸣器电流对其它部分的影响，也可改善电源的交流阻抗，如果可能，最好是再并联一个 220uF 的电解电容。

4.三极管

三极管 Q5 起开关作用，其基极的低电平使三极管饱和导通，使蜂鸣器发声；而基极高电平则使三极管关闭，蜂鸣器停止发声。

4.2.3 硬件设计

大黄蜂 STM32 开发板上有 1 个电子蜂鸣器，PCB 线路板上的标号分别是

FM。与原理图对应的标号是(1.PE2-FMQ)。原理图上标号的命名规则如下:

“1”指的是芯片管脚序号 1;

“PE2”指的是 IO 口定义;

“FMQ”指的是电子元件编号。

我们以程序书写过程为例说明:

MCU 芯片管脚序号(1)+I/O 引脚(PE2)+功能标号(FMQ).

原理图与 PCB 线路板上已经都连接好了，大家不用自己再连接了，直接把程序下载到大黄蜂 STM32 开发板上运行就可以了。

FMQ 原理图

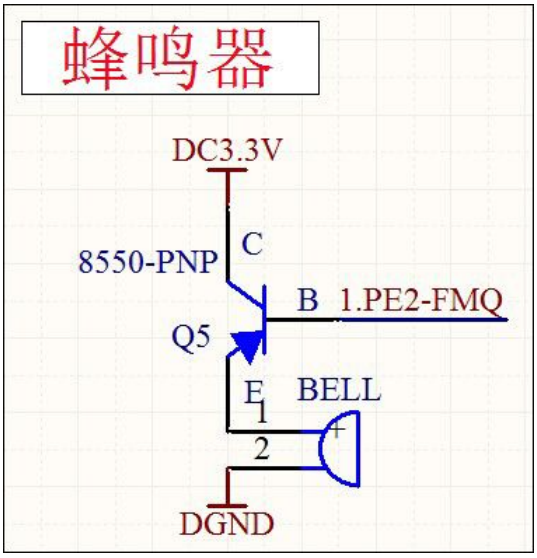


图 4.1

4.2.4 软件设计

4.2.4.1 STM32 库函数文件

```
stm32f10x_gpio.c
stm32f10x_rcc.c
```

本节实验我们主要用到的库文件，其中 stm32f10x_gpio.h 头文件包含了 GPIO 端口的定义。stm32f10x_rcc.h 头文件包含了系统时钟配置函数以

及相关的外设时钟使能函数，所以我们要把这两个头文件对应的 stm32f10x_gpio.c 和 stm32f10x_rcc.c 都加到工程中。

4.2.4.2 自定义头文件

```
pbdata.h
pbdata.c
```

同时我们自己也创建了两个公共的文件，这两个文件主要存放我们自定义的公共函数和全局变量，以方便以后每个功能模块之间传递参数。

4.2.4.3 pbdata.h 文件里的内容是

```
#ifndef _pbdata_H
#define _pbdata_H
#include "stm32f10x.h"

//定义函数
void RCC_HSE_Configuration(void);
void delay(u32 nCount);
void delay_us(u32 nus);
void delay_ms(u16 nms);
#endif
```

语句 #ifndef、#endif 是为了防止 pbdata.h 文件被多个文件调用时出现错误提示。如果不加这两条语句，当两个文件同时调用 pbdata 文件时，会提示重复调用错误。 stm32f10x.h 头文件是我们每个工程都需要调用的，里面包括了 STM32 内部寄存器地址的定义。

4.2.4.4 pbdata.c 文件里的内容是

```
   #include "pbdata.h" //很重要，引用这个头文件

   void RCC_HSE_Configuration(void) //HSE 作为 PLL 时钟，PLL 作为 SYSCLK
   {
      RCC_DeInit(); /*将外设 RCC 寄存器重设为缺省值 */
      RCC_HSEConfig(RCC_HSE_ON); /*设置外部高速晶振（HSE） HSE 晶振打开(ON)*/
      if(RCC_WaitForHSEStartUp() == SUCCESS) {  /*等待 HSE 起振，  SUCCESS：HSE
晶振稳定且就绪*/
      RCC_HCLKConfig(RCC_SYSCLK_Div1);/*设置 AHB 时钟(HCLK)RCC_SYSCLK_Div1—
—AHB 时钟 = 系统时*/
```

```
      RCC_PCLK2Config(RCC_HCLK_Div1); /*设置高速 AHB 时钟(PCLK2)RCC_HCLK_Div1
——APB2 时钟 = HCLK*/
      RCC_PCLK1Config(RCC_HCLK_Div2); /*设置低速 AHB 时钟(PCLK1)RCC_HCLK_Div2
——APB1 时钟 = HCLK / 2*/
      RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);/*设置 PLL 时钟源及
倍频系数*/
      RCC_PLLCmd(ENABLE);    /*使能 PLL */
      while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) ; /*检查指定的 RCC
标志位(PLL 准备好标志)设置与否*/
      RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);  /*设置系统时钟（SYSCLK） */
      while(RCC_GetSYSCLKSource() != 0x08);      /*0x08：PLL 作为系统时钟 */
    }
  }
  void delay(u32 nCount)
  {
     for(;nCount!=0;nCount--);
  }

  /*************************************************************************
*****
  * 名    称：delay_us(u32 nus)
  * 功    能：微秒延时函数
  * 入口参数：u32  nus
  * 出口参数：无
  * 说    明：
  * 调用方法：无
  *************************************************************************
****/
  void delay_us(u32 nus)
  {
      u32 temp;
      SysTick->LOAD = 9*nus;
      SysTick->VAL=0X00;//清空计数器
      SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
      do
      {
       temp=SysTick->CTRL;//读取当前倒计数值
      }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达

      SysTick->CTRL=0x00; //关闭计数器
      SysTick->VAL =0X00; //清空计数器
  }
  /*************************************************************************
*****
```

```
    * 名    称：delay_ms(u16 nms)
    * 功    能：毫秒延时函数
    * 入口参数：u16 nms
    * 出口参数：无
    * 说    明：
    * 调用方法：无
    ****************************************************************************
****/
    void delay_ms(u16 nms)
    {
        u32 temp;
        SysTick->LOAD = 9000*nms;
        SysTick->VAL=0X00;//清空计数器
        SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
        do
        {
         temp=SysTick->CTRL;//读取当前倒计数值
        }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
        SysTick->CTRL=0x00; //关闭计数器
        SysTick->VAL =0X00; //清空计数器
    }
```

延时函数可以通过很多种方法实现，采用精密微秒延时函数“delay_us(u32 nus)”和毫秒延时函数“delay_ms(u16 nms)”，由于本节实验对延时的精度要比较高，主要是为了学习者掌握精确定时的使用方法，所以我们专用的延时函数。

4.2.5 STM32 系统时钟配置 SystemInit()

每个工程都必须在开始时配置并启动 STM32 系统时钟。关于 STM32 系统时钟我们在前面的章节中已经详细的说明了，在这里就不再叙述了。

4.2.6 GPIO 引脚时钟使能

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
```

本节实验只用到了 PE 端口，所以要把 PE 端口的时钟打开。

4.2.7 GPIO 管脚电平控制函数

在主程序中采用 while(1)循环语句，设置 PE 端口的高低电平变化，中

间嵌套延时函数，就实现蜂鸣器的间歇有规律的响动，达到自己设计目的。

```
    {
     GPIO_SetBits(GPIOE,GPIO_Pin_2);//PE2 管脚输出高电平
     delay_ms(500);//0.5S
     GPIO_ResetBits(GPIOE,GPIO_Pin_2);//PE2 管脚输出低电平
     delay_ms(500);//0.5S
     }
```

两个函数中传递的参数都是一样的，第一个参数是端口(GPIOE)，第二个参数是管脚序号(GPIO_Pin_2)。

4.2.8 main.c 文件里的内容是

```
    #include "pbdata.h"  //引用头文件

    void RCC_Configuration(void);//声明系统时钟初始化函数
    void GPIO_Configuration(void);//声明端口初始化函数

    int main(void)
    {
       RCC_Configuration(); //系统时钟初始化
       RCC_HSE_Configuration(); //系统时钟初始化(自定义)
       GPIO_Configuration();//端口初始化

       while(1)
       {
           GPIO_SetBits(GPIOE,GPIO_Pin_2);//PE2 管脚输出高电平
           delay_ms(500);//0.5S
           GPIO_ResetBits(GPIOE,GPIO_Pin_2);//PE2 管脚输出低电平
           delay_ms(500);//0.5S
       }
    }

    void RCC_Configuration(void) //定义系统时钟初始化函数
    {
        SystemInit();
       RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
    }

    void GPIO_Configuration(void)//定义端口初始化函数，也可以说是端口初始化子函数
```

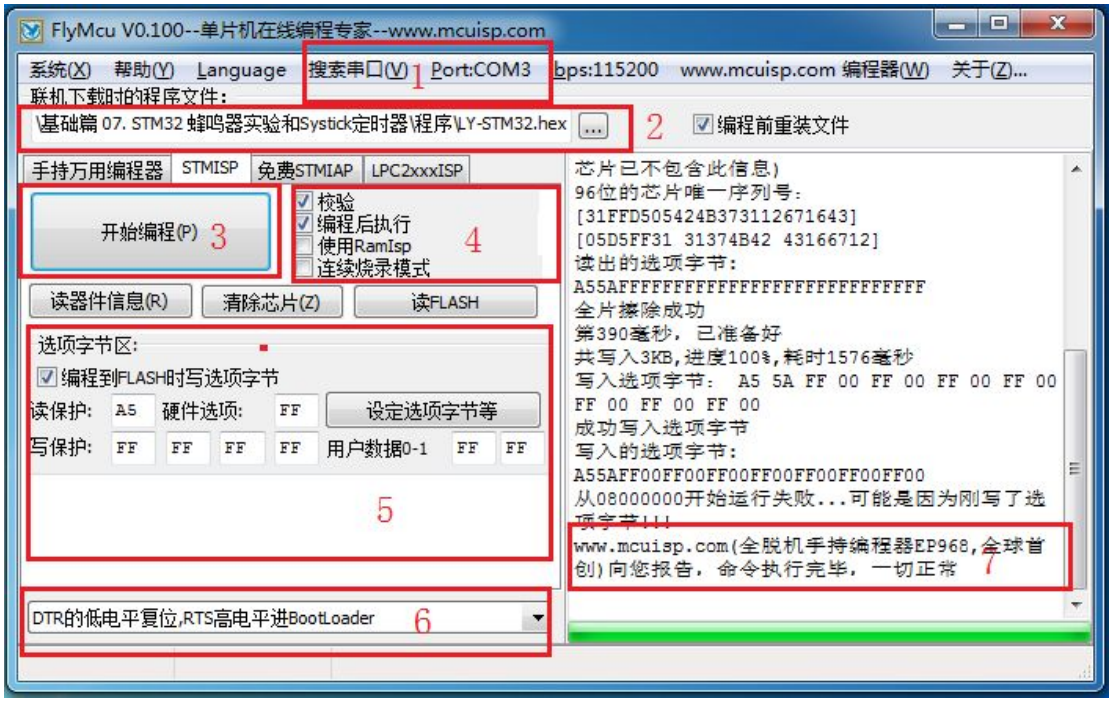
```
{
    GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_2;
   GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
   GPIO_Init(GPIOE,&GPIO_InitStructure);
}
```

在 main(void)程序体中代码不多，开始调用了两个函数，分别是系统时钟初始化函数 RCC_Configuration()和端口初始化函数 GPIO_Configuration()，接下来进入 while(1)循环体，在循环体中，每间隔 0.5 秒钟切换一次 PE2 引脚的输出状态，使其不断的高低电平变化输出。当 PE2 引脚输出低电平时蜂鸣器发声，当 PE2 引脚输出高电平时蜂鸣器停止工作。

4.2.9 程序下载

请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM3。

（2）号区域请自己选择程序所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。



本节实验的源代码在光盘中：（LY-STM32 光盘资料\1.课程\1,基础篇\基础篇 07. STM32 蜂鸣器实验和 Systick 定时器\程序）

4.2.10 实验效果

蜂鸣器发声通过图片看不出来，请通过实验来体验程序实际效果，以上程序比较好理解，只要认真看书和看视频，一定会产生事半功倍的效果，再加上自己的亲自操作和按照自己的想法修改程序，一定会印象深刻。