# 2020 －2021 学年第 1 学期

编号：

****

### 实 验 报 告



实验课程名称 《嵌入式系统设计基础》

专 业 班 级 电子信息工程1901

学 生 姓 号 31902002

学 生 姓 名 曾瑶瑶

实验指导老师 朱胜

**浙大城市学院实验报告**

课程名称 《嵌入式系统设计基础》

实验项目名称 《实验一》《实验二》《实验三》

学生姓名 曾瑶瑶 专业班级 电子信息工程1901 学号 31902002

实验成绩 指导老师（签名 ） 日期

目录

浙大城市学院实验报告

实验一（GPIO接口）

实验二（Delay\_s函数）

实验三（按键输入）

# 实验一（GPIO接口）

## 一. 实验目的和要求

了解IO 口的高低电平控制

了解IO 口作 为输出使用的方法

## 实验内容

①：灯1隔1s闪一次，并且一直闪烁；

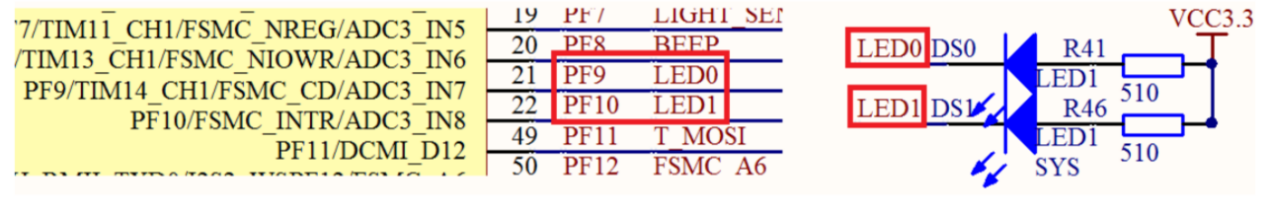
②：灯2隔2s闪一次，同时蜂鸣器叫一次；

③：灯2闪烁5次时，灯1和灯2灭，此时蜂鸣器长鸣5s；

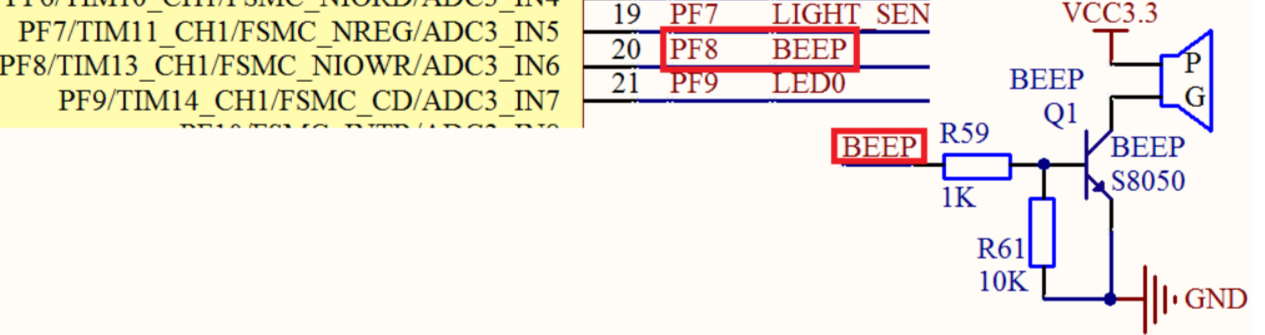
④：重复以上过程。

## 硬件设计

### （1）LED 与 STM32F4 连接原理图



### 蜂鸣器STM32F4 连接原理图



## 四. 实验代码



## 五. 实验结果与分析

①：灯1隔1s闪一次，并且一直闪烁；

②：灯2隔2s闪一次，同时蜂鸣器叫一次；

③：灯2闪烁5次时，灯1和灯2灭，此时蜂鸣器长鸣5s；

④：重复以上过程。

## 六. 讨论、心得

对 STM32F4 开发的软件和硬件平台都有了更较深入的了解了，通过实例，由浅入深，一步步的学习 STM32F4。STM32F4的内部资源非常丰富，最简单的外设，然后一步步深入。结合详细的代码及解释，入手 STM32F4 的各种外设，通过本篇的学习，学会 STM32F4 绝部分外设的使用。

对GPIO更进一步的认识何使用，调用接口过程中，要注意接口函数的定义和引用。

# 实验二（Delay\_s函数）

## 一. 实验目的和要求

掌握时钟系统的结构和原理；

掌握时钟系统函数的编写；

利用时钟系统函数控制LED和蜂鸣器；

## 二．实验内容

①：编写函数，使用此函数把系统时钟变为42MHz；

②：编写delay\_s函数实现以秒为单位的延迟功能；

③：每2s，LED灯闪烁1次，每3s蜂鸣器响一次；

## 三．实验代码

#include "system.h"

#include "stm32f4xx.h"

#include "SysTick.h"

#include "led.h"

#include "beep.h"

int i=0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : RCC\_HSE\_Config

\* 函数功能 : 自定义系统时钟

\* 输 入 : pllm：VCO 输入时钟 分频因子，范围0-63

plln：VCO 输出时钟 倍频因子，范围192-432

pllp：PLLCLK 时钟分频因子，范围2, 4, 6, or 8

pllq：OTG FS,SDIO,RNG 时钟分频因子，范围4-15

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_s(u32 i)

{

while(i--){

delay\_ms(1000);}

}

int main()

{

SysTick\_Init(168);

RCC\_HSE\_Config(8,336,8,7);

LED\_Init();

BEEP\_Init();

while(1)

{

j++;

if(j%3==0)

{BEEP=1;

}

else

{BEEP=0;}

if(j%2==0){

LED1=0;}

else{LED1=1;}

delay\_s(1);

}

}

void delay\_s(u32 i)

{

while(i--){

delay\_ms(1000);}

}

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);

改变时钟的频率，由2变成8.

while(1)

{

j++;

if(j%3==0)

{BEEP=1;

}

else

{BEEP=0;}

if(j%2==0){

LED1=0;}

else{LED1=1;}

delay\_s(1);

}

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

## 四．实验结果与分析

①：编写函数，使用此函数把系统时钟变为42MHz；

②：编写delay\_s函数实现以秒为单位的延迟功能；

③：每2s，LED灯闪烁1次，每3s蜂鸣器响一次；

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);改变时钟的频率，由2变成8.

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

## 五. 讨论、心得

对于改变晶振时间后，对于时间的改变，以至于时间一直对不上。后来了解了晶振频率的含义后对程序进行改变后达到秒的效果。

# 实验三（按键输入）

## 一. 实验目的和要求

利用板载的 4 个按键，来控制板载的两个 LED 的亮灭和蜂鸣器的开关；

了解到 STM32F4 的 IO 口作为输入口的使用方法；

## 二．实验内容和原理

用查询模式，实现按键功能：

①：支持连续按和不连续按；

②：按key0，只亮LED0；

③：按key1，只亮LED1；

④：按key2，只响蜂鸣器；

⑤：按key\_up，LED0和LED1交替亮灭，同时蜂鸣器隔3s响一次。

## 三．实验代码

由于时钟信号的改变，则时钟频率为原来的四分之一。

RCC\_HSE\_Config(8,336,8,7);

改变时钟的频率，由2变成8.

循环内为两秒灯实行亮灭。蜂鸣器3秒鸣叫一次。

#include "system.h"

#include "SysTick.h"

#include "led.h"

#include "beep.h"

#include "key.h"

u8 Keyscan(u8 mode);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

u8 key,i=0;

SysTick\_Init(168);

LED\_Init();

BEEP\_Init();

KEY\_Init();

while(1)

{

key=KEY\_Scan(0); //扫描按键

switch(key)

{

case KEY\_UP\_PRESS: LED2=0;LED1=0;BEEP=0;break;//D1D2亮

case KEY1\_PRESS:LED2=0;LED1=1;BEEP=0;break; //D1亮 case KEY2\_PRESS: LED2=1;LED1=1;BEEP=1;break; //蜂鸣器响

case KEY0\_PRESS: LED2=1;LED1=0;BEEP=0;break; //D2亮

}

}

}

key=KEY\_Scan(0); //扫描按键

switch(key)

{

case KEY\_UP\_PRESS: LED2=0;LED1=0;BEEP=0;break;//D1D2亮

case KEY1\_PRESS:LED2=0;LED1=1;BEEP=0;break; //D1亮 case KEY2\_PRESS: LED2=1;LED1=1;BEEP=1;break; //蜂鸣器响

case KEY0\_PRESS: LED2=1;LED1=0;BEEP=0;break; //D2亮

}

## 四．实验结果与分析

用查询模式，实现按键功能：

①：支持连续按和不连续按；

②：按key0，只亮LED0；

③：按key1，只亮LED1；

④：按key2，只响蜂鸣器；

⑤：按key\_up，LED0和LED1交替亮灭，同时蜂鸣器隔3s响一次。

## 五. 讨论、心得

主函数代码比较简单，先进行一系列的初始化操作，然后在死循环中调用按键扫描函数KEY\_Scan()扫描按键值