**《嵌入式系统综合实践》实验报告（1）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**跑马灯实验（寄存器版本）**

1. 实验代码

**部分源代码及注释：**

#include "sys.h"

#include "delay.h"

#include "led.h"

int main(void)

{

Stm32\_Clock\_Init(360,25,2,8);//进行时钟的初始化使能，180Mhz

delay\_init(180); //初始化使能延迟函数

LED\_Init(); //初始化LED时钟

while(1)

{

LED0=0; //LED0亮

LED1=1; //LED1灭

delay\_ms(500); //延迟500ms

LED0=1; //LED0灭

LED1=0; //LED1亮

delay\_ms(500); //延迟500ms

}

}

//LED端口定义

#define LED0 PBout(1) // PB0输出

#define LED1 PBout(0) // PB1输出

void LED\_Init(void);//初始化

void LED\_Init(void) //初始化PB0和PB1为输出口.并使能这两个口的时钟

{

RCC->AHB1ENR|=1<<1; //将1该寄存器的bit1置1来使能PORTB时钟

GPIO\_Set(GPIOB,PIN0|PIN1,GPIO\_MODE\_OUT,GPIO\_OTYPE\_PP,GPIO\_SPEED\_100M,GPIO\_PUPD\_PU); //设置PB1和PB0模式为推挽输出，输出速度为100Mhz，输出上拉

LED0=1; //LED0关闭

LED1=1; //LED1关闭

}

void GPIO\_Set(GPIO\_TypeDef\* GPIOx,u32 BITx,u32 MODE,u32 OTYPE,u32 OSPEED,u32 PUPD)

{

u32 pinpos=0,pos=0,curpin=0;

for(pinpos=0;pinpos<16;pinpos++)

{

pos=1<<pinpos; //逐位检查

curpin=BITx&pos;// 检查引脚是否要设置

if(curpin==pos) // /需要设置

{

GPIOx->MODER&=~(3<<(pinpos\*2)); //先清除原来的设置

GPIOx->MODER|=MODE<<(pinpos\*2); //设置新的模式

if((MODE==0X01)||(MODE==0X02)) //如果是输出模式/复用功能模式

{

GPIOx->OSPEEDR&=~(3<<(pinpos\*2)); /清除原来的设置 GPIOx->OSPEEDR|=(OSPEED<<(pinpos\*2));// 设置新的速度值

GPIOx->OTYPER&=~(1<<pinpos) ; // /清除原来的设置

GPIOx->OTYPER|=OTYPE<<pinpos; //设置新的输出模式 }

GPIOx->PUPDR&=~(3<<(pinpos\*2)); //先清除原来的设置

GPIOx->PUPDR|=PUPD<<(pinpos\*2); //设置新的上下拉

}

}

}

**修改之后的代码：**

**（不通过函数，直接修改寄存器的值以达到初始化和控制LED灯亮灭的效果）**

int main(void)

{

Stm32\_Clock\_Init(360,25,2,8);

delay\_init(180);

//LED\_Init(); //不使用该函数

RCC->AHB1ENR|=1<<1; //将1该寄存器的bit1置1来使能PORTB时钟

GPIOB->ODR&=0xFFFFFFFC;

GPIOB->MODER = 0x00000005; //设置模式

GPIOB->OTYPER = 0x00000000; //设置为推挽输出

GPIOB->OSPEEDR = 0x0000000F; //设置速度

GPIOB->PUPDR = 0x00000005; //设置为输出上拉

while(1)

{

GPIOB->BSRR=0xFFFFFFFD; //LED0亮，LED1灭

delay\_ms(500);

GPIOB->BSRR=0xFFFFFFFE; ////LED0灭，LED1亮

delay\_ms(500);

}

}

1. 实验现象

运行程序后开发板上的LED0与LED1初始状态为全部熄灭，之后LED0熄灭LED1点亮，延迟500ms后LED0点亮LED1熄灭，再延迟500ms，并将LED0与LED1交替点亮的操作循环实践以达到LED闪烁的效果。

**图示, 示意图

描述已自动生成**

**《嵌入式系统综合实践》实验报告（2）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**跑马灯实验（HAL库版本）**

1. 实验代码

部分原代码及注释：

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

/\*下面主函数是使用HAL库函数实现控制IO口输出\*/

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

LED\_Init(); //初始化LED

while(1)

{

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_1,GPIO\_PIN\_RESET);

//LED0对应引脚PB1拉低，亮，等同于LED0(0)

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_0,GPIO\_PIN\_SET);

//LED1对应引脚PB0拉高，灭，等同于LED1(1)

delay\_ms(500);//延时500ms

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_1,GPIO\_PIN\_SET);

//LED0对应引脚PB1拉高，灭，等同于LED0(1)

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_0,GPIO\_PIN\_RESET);

//LED1对应引脚PB0拉低，亮，等同于LED1(0)

delay\_ms(500); //延时500ms

}

}

/\*下面主函数使用位带操作实现：\*/

/\*

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

LED\_Init(); //初始化LED while(1)

{

LED0=0; //LED0亮

LED1=1; //LED1灭

delay\_ms(500);

LED0=1; //LED0灭

LED1=0; //LED1亮

delay\_ms(500);

}

}\*/

/\*

下面主函数使用直接操作寄存器方式实现跑马灯

\*/

/\*

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

LED\_Init(); //初始化LED

while(1)

{

GPIOB->BSRR=GPIO\_PIN\_1; //LED0亮

GPIOB->BSRR=GPIO\_PIN\_0<<16; //LED1灭

delay\_ms(500);

GPIOB->BSRR=GPIO\_PIN\_1<<16; //LED0灭

GPIOB->BSRR=GPIO\_PIN\_0; //LED1亮

delay\_ms(500);

}

}

\*/

**修改之后的代码：**

/\*下面主函数使用位带操作实现：\*/

#define Baddr (GPIOB\_BASE+0x14) //GPIO输出数据寄存器地址加上偏移量

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

LED\_Init(); //初始化LED

while(1)

{

/\*位带操作：操作地址取出前4位，加上SRAM区最低的地址，再加上需操作的地址

的后20位乘以32，再加上需操作的位乘以4。\*/

\*((volatile unsigned int\*)((Baddr &0xF0000000)+0x2000000+((Baddr &0xFFFFF)<<5)+(0<<2)))=0x0; //LED0亮

\*((volatile unsigned int\*)((Baddr &0xF0000000)+0x2000000+((Baddr &0xFFFFF)<<5)+(1<<2)))=0x1; //LED1灭

delay\_ms(500);//延时500ms

\*((volatile unsigned int\*)((Baddr &0xF0000000)+0x2000000+((Baddr &0xFFFFF)<<5)+(0<<2)))=0x1; //LED0灭

\*((volatile unsigned int\*)((Baddr &0xF0000000)+0x2000000+((Baddr &0xFFFFF)<<5)+(1<<2)))=0x0; //LED1亮

delay\_ms(500);//延时500ms

}

}

二、实验现象

运行程序后开发板上的LED0与LED1初始状态为全部熄灭，之后LED0熄灭LED1点亮，延迟500ms后LED0点亮LED1熄灭，再延迟500ms，并将LED0与LED1交替点亮的操作以循环实践以达到LED闪烁的效果。

**图示, 示意图

描述已自动生成**

三、实验心得（1）（2）

第一个实验是寄存器版本的跑马灯实验，源程序通过函数的方式修改寄存器的值以达到预期效果。通过阅读源程序与查询资料，我们在源程序上进行修改，直接在主程序中给寄存器赋值以达到同样的效果。通过这次实验让我了解了STM32的开发环境，在开发板上烧录和执行程序的过程。

第二次实验是HAL库函数版本的跑马灯实验，在阅读源代码时，我们对比了它与寄存器版本的跑马灯实验的区别，它的源代码中使用了多种方式实现该功能。在后续自行修改代码时，我们修改了其中用位带操作实验的版本。我们将位带操作的原理通过代码直观地展现出来，以便于更好地记忆以及理解位带区与位带别名区映射的计算。

**《嵌入式系统综合实践》实验报告（3）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**按键输入实验**

1. 实验代码

**部分原代码及注释：**

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

int main(void)

{

u8 key;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

while(1)

{

key=KEY\_Scan(0); //按键扫描

switch(key) //根据按下的按键实现相应的功能

{

case WKUP\_PRES: //控制LED0,LED1互斥点亮

LED1=!LED1;

LED0=!LED1;

break;

case KEY2\_PRES: //控制LED0翻转

LED0=!LED0;

break;

case KEY1\_PRES: //控制LED1翻转

LED1=!LED1;

break;

case KEY0\_PRES: //同时控制LED0,LED1翻转

LED0=!LED0;

LED1=!LED1;

break;

}

delay\_ms(10); //消抖动

}

}

#include "key.h"

#include "delay.h"

//按键初始化函数

void KEY\_Init(void)

{

GPIO\_InitTypeDef GPIO\_Initure;

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); //开启GPIOA时钟

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); //开启GPIOC时钟

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE(); //开启GPIOH时钟

GPIO\_Initure.Pin=GPIO\_PIN\_0; //选择PA0

GPIO\_Initure.Mode=GPIO\_MODE\_INPUT; //输入模式

GPIO\_Initure.Pull=GPIO\_PULLDOWN; //下拉

GPIO\_Initure.Speed=GPIO\_SPEED\_HIGH; //高速(100MHz)

HAL\_GPIO\_Init(GPIOA,&GPIO\_Initure); //将改变的值写入寄存器

GPIO\_Initure.Pin=GPIO\_PIN\_13; //PC13

GPIO\_Initure.Mode=GPIO\_MODE\_INPUT; //输入

GPIO\_Initure.Pull=GPIO\_PULLUP; //上拉

GPIO\_Initure.Speed=GPIO\_SPEED\_HIGH; //高速(100MHz)

HAL\_GPIO\_Init(GPIOC,&GPIO\_Initure); //将改变的值写入寄存器

GPIO\_Initure.Pin=GPIO\_PIN\_2|GPIO\_PIN\_3; //PH2,3

HAL\_GPIO\_Init(GPIOH,&GPIO\_Initure); //将改变的值写入寄存器

}

//按键处理函数

//返回按键值

//mode:0,不支持连续按;1,支持连续按;

//0，没有任何按键按下

//1，WKUP按下 WK\_UP

//注意此函数有响应优先级,KEY0>KEY1>KEY2>WK\_UP!!

u8 KEY\_Scan(u8 mode)

{

static u8 key\_up=1; //按键松开标志

if(mode==1)key\_up=1; //支持连按

if(key\_up&&(KEY0==0||KEY1==0||KEY2==0||WK\_UP==1))//

{

delay\_ms(10); //消抖动

key\_up=0; //表示有按键按下

if(KEY0==0) return KEY0\_PRES; //若有按键按下，返回该按键值

else if(KEY1==0) return KEY1\_PRES;

else if(KEY2==0) return KEY2\_PRES;

else if(WK\_UP==1) return WKUP\_PRES;

}else if(KEY0==1&&KEY1==1&&KEY2==1&&WK\_UP==0)key\_up=1;

return 0; //无按键按下

}

**修改之后的代码：添加检测并识别按键的功能**

u8 key;

int function(u8 key1){

if(key==key1){

delay\_ms(2000); //延时 2000 毫秒

key=KEY\_Scan(1); //将 key 的值赋给 KEY\_Scan(1) 的结果

if(key==key1){ //如果KEY\_Scan(1)的结果等于key1，则进入循环

while(key==key1){

//在循环中，不断将 key 的值赋给 KEY\_Scan(0) 的结果

key=KEY\_Scan(0); //直到 key 的值发生变化，跳出循环

}

return 1;//返回 1，表示按键被按下

}

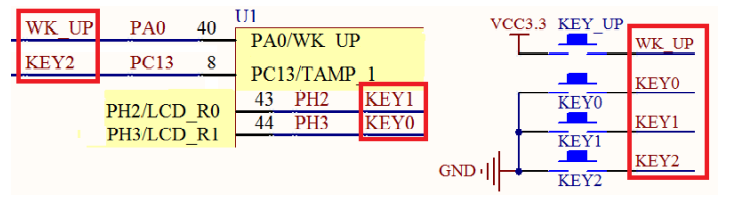
}

return 0;//如果 key 不等于 key1，则返回 0，表示按键未按下

}

1. 实验现象

按下up按键后LED0与LED1会变成一盏灯亮，一盏灯灭；按下KEY0后LED0与LED1会同时改变状态；按下KEY1后，LED1会改变状态；按下KEY2后，LED2会改变状态。修改后我们添加了一个检测并识别按键的功能。当一个按键被按下时，程序会等待 2 秒钟，然后尝试将该按键的值赋给另一个键值（KEY\_Scan(1)），如果二者相等，说明这是一个有效的按键。接着程序会进入一个循环，不断将该按键的值赋给另一个键值（KEY\_Scan(0)），直到键值发生变化。如果循环结束后，键值仍然没有发生变化，说明按键已被识别。此时函数返回 1，表示按键被成功识别；否则返回 0，表示按键未按下。



**《嵌入式系统综合实践》实验报告（4）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**外部中断实验**

1. 实验代码

**部分原代码及注释：**

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "exti.h"

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

EXTI\_Init(); //外部中断初始化

while(1)

{

printf("OK\r\n"); //打印OK提示程序运行

delay\_ms(1000); //每隔1s打印一次

}

}

#include "key.h"

//外部中断初始化

void EXTI\_Init(void)

{

GPIO\_InitTypeDef GPIO\_Initure;

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); //开启GPIOA时钟

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); //开启GPIOC时钟

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE(); //开启GPIOH时钟

GPIO\_Initure.Pin=GPIO\_PIN\_0; //PA0

GPIO\_Initure.Mode=GPIO\_MODE\_IT\_RISING; //上升沿触发

GPIO\_Initure.Pull=GPIO\_PULLDOWN;

HAL\_GPIO\_Init(GPIOA,&GPIO\_Initure);

GPIO\_Initure.Pin=GPIO\_PIN\_13; //PC13

GPIO\_Initure.Mode=GPIO\_MODE\_IT\_FALLING; //下降沿触发

GPIO\_Initure.Pull=GPIO\_PULLUP;

HAL\_GPIO\_Init(GPIOC,&GPIO\_Initure);

GPIO\_Initure.Pin=GPIO\_PIN\_2|GPIO\_PIN\_3; //PH2,3

HAL\_GPIO\_Init(GPIOH,&GPIO\_Initure);

//中断线0-PA0

HAL\_NVIC\_SetPriority(EXTI0\_IRQn,2,0); //抢占优先级为2，子优先级为0

HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn); //使能中断线0

//中断线2-PH2

HAL\_NVIC\_SetPriority(EXTI2\_IRQn,2,1); //抢占优先级为2，子优先级为1

HAL\_NVIC\_EnableIRQ(EXTI2\_IRQn); //使能中断线2

//中断线3-PH3

HAL\_NVIC\_SetPriority(EXTI3\_IRQn,2,2); //抢占优先级为2，子优先级为2

HAL\_NVIC\_EnableIRQ(EXTI3\_IRQn); //使能中断线2

//中断线13-PC13

HAL\_NVIC\_SetPriority(EXTI15\_10\_IRQn,2,3); //抢占优先级为2，子优先级为3

HAL\_NVIC\_EnableIRQ(EXTI15\_10\_IRQn); //使能中断线13

}

//中断服务函数

void EXTI0\_IRQHandler(void)

{

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_0);//调用中断处理公用函数

}

void EXTI2\_IRQHandler(void)

{

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_2);//调用中断处理公用函数

}

void EXTI3\_IRQHandler(void)

{

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_3);//调用中断处理公用函数

}

void EXTI15\_10\_IRQHandler(void)

{

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_13);//调用中断处理公用函数

}

//中断服务程序中需要做的事情

//在HAL库中所有的外部中断服务函数都会调用此函数

//GPIO\_Pin:中断引脚号

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

delay\_ms(100); //消抖

switch(GPIO\_Pin)

{

case GPIO\_PIN\_0:

if(WK\_UP==1)

{

LED1=!LED1;//控制LED0,LED1互斥点亮

LED0=!LED1;

}

break;

case GPIO\_PIN\_2:

if(KEY1==0) //LED1翻转

{

LED1=!LED1;

}

break;

case GPIO\_PIN\_3:

if(KEY0==0) //同时控制LED0,LED1翻转

{

LED0=!LED0;

LED1=!LED1;

}

break;

case GPIO\_PIN\_13:

if(KEY2==0)

{

LED0=!LED0;//控制LED0翻转

}

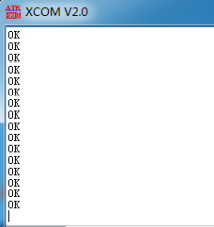
break;

}

}

1. 实验现象

本次实验需要用到XCOM软件来显示内容，在按键按下时，会跳转至中断服务程序，从而运行中断服务程序来改变LED灯的状态。



1. 实验心得（3）（4）

第三次实验是按键输入实验，该实验首次加入了硬件方式控制LED灯的状态，通过本次实验，使我更加深入地了解了如何编写按键控制方面的程序，通过增删delay函数可以实现消抖动的作用。我们在源代码的基础上添加了按键识别功能，可以检测按键是否被按下。

第四次实验使用了外部中断，在主程序运行时，触发中断条件便可跳转至外部中断运行中断服务程序，运行完毕后跳转回主程序，从而做到在主程序运行的途中也能通过一些条件处理中断事件。本次实验使我从实际应用中了解到了中断服务程序的机理，并且学会了中断程序头文件的引用，以及中断服务程序放置位置的多样化。

**《嵌入式系统综合实践》实验报告（5）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**电容触摸按键实验**

1. 实验代码

部分源代码：

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "tpad.h"

int main(void)

{

u8 t=0;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

TPAD\_Init(2); //初始化触摸按键,以90/8=11.25Mhz频率计数

while(1)

{

if(TPAD\_Scan(0)) //成功捕获到了一次上升沿(此函数执行时间至少15ms)

{

LED1=!LED1; //LED1取反

}

t++;

if(t==15)

{

t=0;

LED0=!LED0;

}

delay\_ms(10);

}

}

修改后的代码：在HARDWARE中导入PCF8574与myiic

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "tpad.h"

#include "PCF8574.h"

int main(void)

{

u8 t=0;

u8 beepsta=1;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

TPAD\_Init(2); //初始化触摸按键,以90/8=11.25Mhz频率计数

PCF8574\_Init(); //初始化蜂鸣器

while(1)

{

if(TPAD\_Scan(0)) //成功捕获到了一次上升沿(此函数执行时间至少15ms)

{

beepsta=!beepsta; //蜂鸣器状态取反

PCF8574\_WriteBit(BEEP\_IO,beepsta); //控制蜂鸣器

LED1=!LED1; //LED1取反

}

t++;

if(t==15)

{

t=0;

LED0=!LED0;

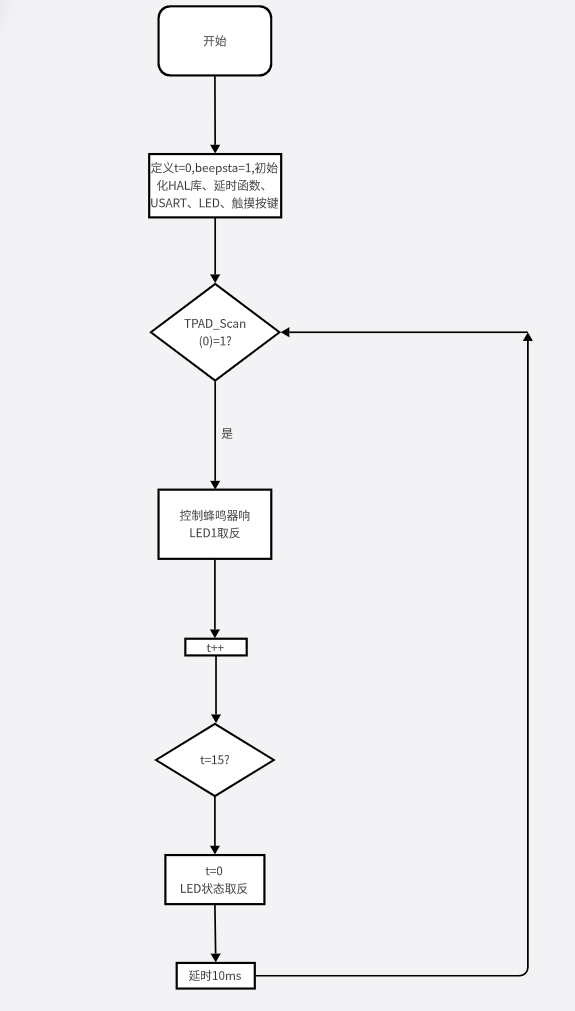
}

delay\_ms(10);

}

}

1. 实验流程图



三、实验现象

本实验,利用开发板板载的电容触摸按键(右下角白色LOGO,即TPAD)，通过TIM2\_CH1（PA5）对电容触摸按键的检测，实现对DS1的控制,下载本代码后，我们通过按开发板右下角的TPAD按钮，就可以控制DS1的亮灭了。 我们修改实验代码后在里面加入了蜂鸣器，下载代码后，可以通过开发板的TPAD按钮控制蜂鸣器发出声响。

**《嵌入式系统综合实践》实验报告（6）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**光照&接近传感器实验**

1. 实验代码

部分实验源代码：

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "lcd.h"

#include "sdram.h"

#include "ap3216c.h"

int main(void)

{

u16 ir,als,ps;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

POINT\_COLOR=RED;

LCD\_ShowString(30,50,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,70,200,16,16,"AP3216C TEST");

LCD\_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,110,200,16,16,"2016/1/16");

while(AP3216C\_Init()) //检测不到AP3216C

{

LCD\_ShowString(30,130,200,16,16,"AP3216C Check Failed!");

delay\_ms(500);

LCD\_ShowString(30,130,200,16,16,"Please Check! ");

delay\_ms(500);

LED0=!LED0; //DS0闪烁

}

LCD\_ShowString(30,130,200,16,16,"AP3216C Ready!");

LCD\_ShowString(30,160,200,16,16," IR:");

LCD\_ShowString(30,180,200,16,16," PS:");

LCD\_ShowString(30,200,200,16,16,"ALS:");

POINT\_COLOR=BLUE; //设置字体为蓝色

while(1)

{

AP3216C\_ReadData(&ir,&ps,&als); //读取数据

LCD\_ShowNum(30+32,160,ir,5,16); //显示IR数据

LCD\_ShowNum(30+32,180,ps,5,16); //显示PS数据

LCD\_ShowNum(30+32,200,als,5,16);//显示ALS数据

LED0=!LED0; ; //提示系统正在运行

delay\_ms(120);

}

}

修改后的代码：

//画矩形

//(x1,y1),(x2,y2):矩形的对角坐标

void LCD\_DrawRectangle(u16 x1, u16 y1, u16 x2, u16 y2)

{

LCD\_DrawLine(x1,y1,x2,y1);

LCD\_DrawLine(x1,y1,x1,y2);

LCD\_DrawLine(x1,y2,x2,y2);

LCD\_DrawLine(x2,y1,x2,y2);

}

//在指定位置画一个指定大小的圆

//(x,y):中心点

//r :半径

void LCD\_Draw\_Circle(u16 x0,u16 y0,u8 r)

{

int a,b;

int di;

a=0;b=r;

di=3-(r<<1); //判断下个点位置的标志

while(a<=b)

{

LCD\_DrawPoint(x0+a,y0-b); //5

LCD\_DrawPoint(x0+b,y0-a); //0

LCD\_DrawPoint(x0+b,y0+a); //4

LCD\_DrawPoint(x0+a,y0+b); //6

LCD\_DrawPoint(x0-a,y0+b); //1

LCD\_DrawPoint(x0-b,y0+a);

LCD\_DrawPoint(x0-a,y0-b); //2

LCD\_DrawPoint(x0-b,y0-a); //7

a++;

//使用Bresenham算法画圆

if(di<0)di +=4\*a+6;

else

{

di+=10+4\*(a-b);

b--;

}

}

}

int main(void)

{

u16 ir,als,ps;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

POINT\_COLOR=RED;

LCD\_ShowString(30,50,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,70,200,16,16,"AP3216C TEST");

LCD\_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,110,200,16,16,"2016/1/16");

while(AP3216C\_Init()) //检测不到AP3216C

{

LCD\_ShowString(30,130,200,16,16,"AP3216C Check Failed!");

delay\_ms(500);

LCD\_ShowString(30,130,200,16,16,"Please Check! ");

delay\_ms(500);

LED0=!LED0; //DS0闪烁

}

LCD\_ShowString(30,130,200,16,16,"AP3216C Ready!");

LCD\_ShowString(30,160,200,16,16," IR:");

LCD\_ShowString(30,180,200,16,16," PS:");

LCD\_ShowString(30,200,200,16,16,"ALS:");

POINT\_COLOR=BLUE; //设置字体为蓝色

while(1)

{

AP3216C\_ReadData(&ir,&ps,&als); //读取数据

LCD\_ShowNum(30+32,160,ir,5,16); //显示IR数据

LCD\_ShowNum(30+32,180,ps,5,16); //显示PS数据

LCD\_ShowNum(30+32,200,als,5,16);//显示ALS数据

LED0=!LED0; ; //提示系统正在运行

LCD\_Draw\_Circle(500,400,ir);

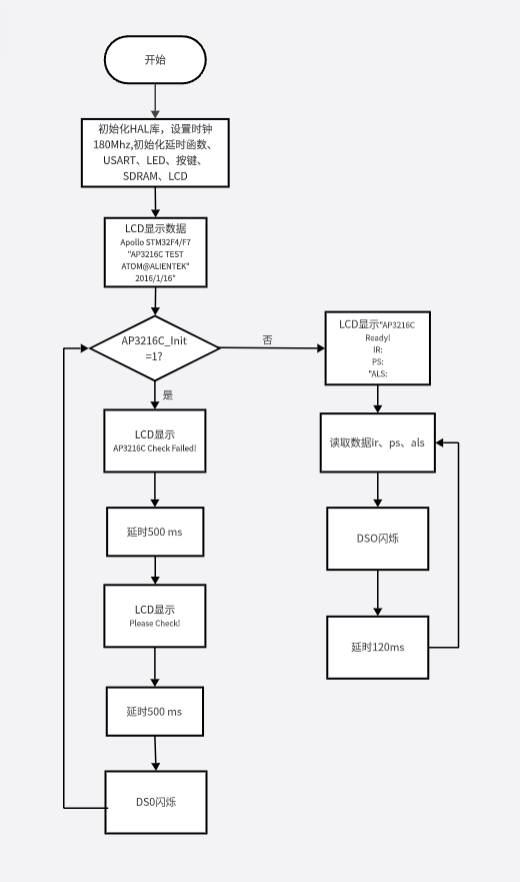
LCD\_DrawRectangle(300,400,ps,als);

delay\_ms(120);

}

}

1. 实验流程图



1. 实验现象

本实验,开机先检测AP3216C是否存在，如检测不到AP3216C，则在LCD屏幕上面显示报错信息。如果检测到AP3216C，则显示正常，并在主循环里面，循环读取ALS+PS+IR的传感器数据，并显示在LCD屏幕上面。同时，DS0闪烁，提示程序正在运行。另外，本例程将AP3216C的读写操作函数加入USMART控制，我们也可以通过USMART对AP3216C进行控制。修改后的代码加入了两个函数分别为LCD\_DrawRectangle与LCD\_Draw\_Circle用来在LCD屏幕画矩形与圆。

1. 实验心得（5）（6）

此节实验课上大部分时间都花费在第一个实验中，我和同组成员想了两个修改代码的方向，但由于对软件的使用掌握不好，没有成功添加进库函数，都以失败告终，后来请教了老师和同学，终于掌握了添加库函数的方法，然后成功修改了代码，实现了使用开发板的TPAD按钮控制蜂鸣器发出声响。第二个实验中我们按照老师给提示的方向修改代码在库函数中寻找可以在LCD屏幕上画出圆和矩形，在主函数中相应位置添加后实现绘图。此次实验让我学会了如何添加库函数,对keil5的使用进一步了解，并让我对阿波罗STM32F429开发板有了新的学习。

**《嵌入式系统综合实践》实验报告（7）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**定时器中断实验**

1. 实验代码

部分源代码：

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "timer.h"

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

TIM3\_Init(5000-1,9000-1); //定时器3初始化，定时器时钟为90M，分频系数为9000-1，

//所以定时器3的频率为90M/9000=10K，自动重装载为5000-1，那么定时器周期就是500ms

while(1)

{

LED0=!LED0; //LED0翻转

delay\_ms(200); //延时200ms

}

}

//回调函数，定时器中断服务函数调用

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

if(htim==(&TIM3\_Handler))

{

LED1=!LED1; //LED1反转

}

}

修改后的代码：在HARDWARE中导入PCF8574与myiic

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "timer.h"

#include "PCF8574.h"

int main(void)

{

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

TIM3\_Init(5000-1,9000-1); //定时器3初始化，定时器时钟为90M，分频系数为9000-1，

//所以定时器3的频率为90M/9000=10K，自动重装载为5000-1，那么定时器周期就是500ms

while(1)

{

LED0=!LED0; //LED0翻转

delay\_ms(200); //延时200ms

}

}

//回调函数，定时器中断服务函数调用

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

if(htim==(&TIM3\_Handler))

{

u8 beepsta=1;

LED1=!LED1; //LED1反转

PCF8574\_Init();

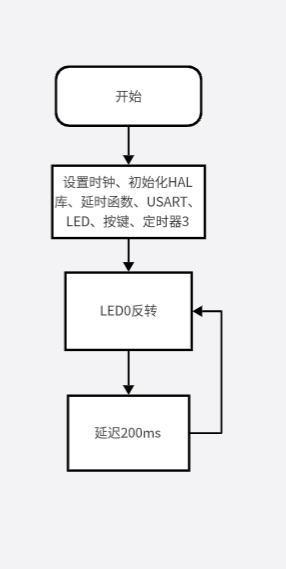
beepsta=!beepsta; //蜂鸣器状态取反

PCF8574\_WriteBit(BEEP\_IO,beepsta); //控制蜂鸣器

}

}

1. 实验流程图



1. 实验结果

本实验,DS0用来指示程序运行，400ms为一个周期。DS1用于定时器中断取反，指示定时器中断状态，1000ms为一个周期。下载完后，可以看到DS0快闪，DS1慢闪。修改后的代码在HARDWARE中导入PCF8574与myiic，可以用定时器中断控制蜂鸣器有规律的发出声响。

**《嵌入式系统综合实践》实验报告（8）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**PWM输出实验**

1. 实验代码

部分源代码：

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "timer.h"

int main(void)

{

u8 dir=1;

u16 led0pwmval=0;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

TIM3\_PWM\_Init(500-1,90-1); //90M/90=1M的计数频率，自动重装载为500，那么PWM频率为1M/500=2kHZ

while(1)

{

delay\_ms(10);

if(dir)

led0pwmval++; //dir==1 led0pwmval递增

else

led0pwmval--; //dir==0 led0pwmval递减

if(led0pwmval>300)

dir=0; //led0pwmval到达300后，方向为递减

if(led0pwmval==0)

dir=1; //led0pwmval递减到0后，方向改为递增

TIM\_SetTIM3Compare4(led0pwmval); //修改比较值，修改占空比

}

}

修改后的代码：

int main(void)

{

//u8 dir=1;

u8 key;

u16 led0pwmval=0;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init();

TIM3\_PWM\_Init(500-1,90-1); //90M/90=1M的计数频率，自动重装载为500(周期500)，那么PWM频率为1M/500=2kHZ

while(1)

{

key=KEY\_Scan(0);

switch(key)

{

case KEY1\_PRES:

if((led0pwmval-10)==0||(led0pwmval-10)<0||(led0pwmval-10)>280)

led0pwmval=0;

else

led0pwmval=led0pwmval-10;

break;

case WKUP\_PRES:

led0pwmval=led0pwmval+10;

break;

}

//if(dir)led0pwmval++; //dir==1 led0pwmval递增

//else led0pwmval--; //dir==0 led0pwmval递减

//if(led0pwmval>300)dir=0; //led0pwmval到达300后，方向为递减

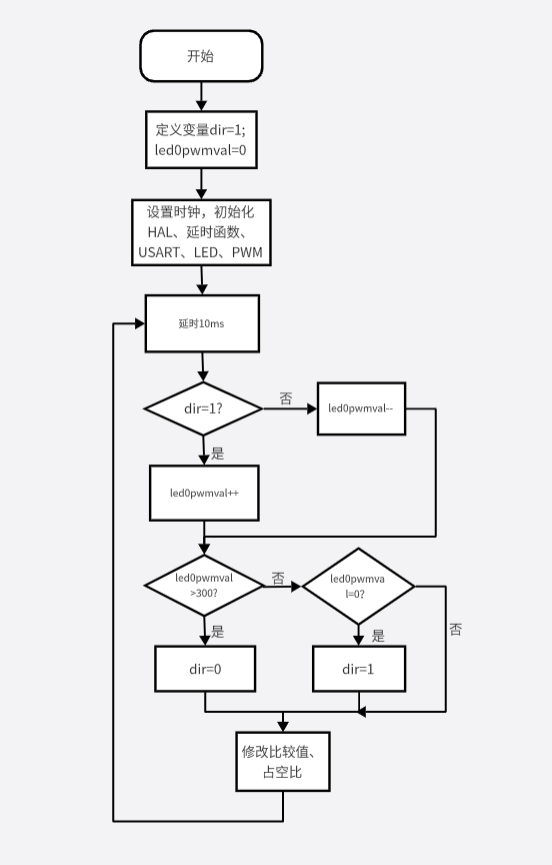
//if(led0pwmval==0)dir=1; //led0pwmval递减到0后，方向改为递增

TIM\_SetTIM3Compare4(led0pwmval); //修改比较值，修改占空比

}

}

1. 实验流程图



1. 实验结果

本实验,DS0由暗到亮,再由亮到暗,再由暗到亮,依次循环。修改后的代码可以实现由按键来控制DSO的亮暗变化的进度。

1. 实验心得（7）（8）

本次实验从程序运行的结果上理解并不算困难，难点在于对于定时器中断的理解上，由于理论课时对于定时器的学习没有非常到位导致在定时器中断的理解上产生了一定的困难，最后通过阅读指南和书籍，渐渐地对理解了程序的意思，加深了我对定时器中断的印象，然后我们在源程序的基础上加了蜂鸣器控制的代码，用定时中断来控制蜂鸣器，使蜂鸣器的声音可以有规律的发出。此次实验使用了PWM（Pulse Width Modulation，脉冲宽度调制），可以通过设定值来控制计数器的递增和递减进而控制电平的值从而控制LED亮度的变化。本次修改参照了之间的按键控制实验，综合二者实现了通过按键进行的计数器的控制从而使计数递增和递减。本次实验使我更进一步了解PWM的运行原理，并且通过修改代码使程序更具有综合性，并且复习了之前实验的内容。

**《嵌入式系统综合实践》实验报告（9）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：**IIC实验**

1. 实验代码

**部分源代码：**

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "lcd.h"

#include "sdram.h"

#include "24cxx.h"

//要写入到24c02的字符串数组

const u8 TEXT\_Buffer[]={"Apollo STM32F4 IIC TEST"};

#define SIZE sizeof(TEXT\_Buffer)

int main(void)

{

u8 key;

u16 i=0;

u8 datatemp[SIZE];

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

AT24CXX\_Init(); //初始化IIC

POINT\_COLOR=RED;

LCD\_ShowString(30,50,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,70,200,16,16,"IIC TEST");

LCD\_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,110,200,16,16,"2016/1/13");

LCD\_ShowString(30,130,200,16,16,"KEY1:Write KEY0:Read"); //显示提示信息

while(AT24CXX\_Check())//检测不到24c02

{

LCD\_ShowString(30,150,200,16,16,"24C02 Check Failed!");

delay\_ms(500);

LCD\_ShowString(30,150,200,16,16,"Please Check! ");

delay\_ms(500);

LED0=!LED0;//DS0闪烁

}

LCD\_ShowString(30,150,200,16,16,"24C02 Ready!");

POINT\_COLOR=BLUE;//设置字体为蓝色

while(1)

{

key=KEY\_Scan(0);

if(key==KEY1\_PRES)//KEY1按下,写入24C02

{

LCD\_Fill(0,170,239,319,WHITE);//清除半屏

LCD\_ShowString(30,170,200,16,16,"Start Write 24C02....");

AT24CXX\_Write(0,(u8\*)TEXT\_Buffer,SIZE);

LCD\_ShowString(30,170,200,16,16,"24C02 Write Finished!");//提示传送完成

}

if(key==KEY0\_PRES)//KEY0按下,读取字符串并显示

{

LCD\_ShowString(30,170,200,16,16,"Start Read 24C02.... ");

AT24CXX\_Read(0,datatemp,SIZE);

LCD\_ShowString(30,170,200,16,16,"The Data Readed Is: ");//提示传送完成

LCD\_ShowString(30,190,200,16,16,datatemp);//显示读到的字符串

}

i++;

delay\_ms(10);

if(i==20)

{

LED0=!LED0;//提示系统正在运行

i=0;

}

}

}

**修改后的代码：**

u8 TEXT\_Buffer[100]={'a'};

#define SIZE sizeof(TEXT\_Buffer)

int main(void)

{

u8 key;

u16 i=0;

u8 datatemp[SIZE];

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

AT24CXX\_Init(); //初始化IIC

POINT\_COLOR=RED;

LCD\_ShowString(30,50,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,70,200,16,16,"IIC TEST");

LCD\_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,110,200,16,16,"2016/1/13");

LCD\_ShowString(30,130,200,16,16,"KEY1:Write KEY0:Read"); //显示提示信息

while(AT24CXX\_Check())//检测不到24c02

{

LCD\_ShowString(30,150,200,16,16,"24C02 Check Failed!");

delay\_ms(500);

LCD\_ShowString(30,150,200,16,16,"Please Check! ");

delay\_ms(500);

LED0=!LED0;//DS0闪烁

}

LCD\_ShowString(30,150,200,16,16,"24C02 Ready!");

POINT\_COLOR=BLUE;//设置字体为蓝色

while(1)

{

key=KEY\_Scan(0);

if(key==KEY1\_PRES)//KEY1按下,写入24C02

{

i++;

TEXT\_Buffer[i]='b';

LCD\_Fill(0,170,239,319,WHITE);//清除半屏

LCD\_ShowString(30,170,200,16,16,"Start Write 24C02....");

AT24CXX\_Write(0,(u8\*)TEXT\_Buffer,SIZE);

LCD\_ShowString(30,170,200,16,16,"24C02 Write Finished!");//提示传送完成

}

if(key==KEY0\_PRES)//KEY0按下,读取字符串并显示

{

LCD\_ShowString(30,170,200,16,16,"Start Read 24C02.... ");

AT24CXX\_Read(0,datatemp,SIZE);

LCD\_ShowString(30,170,200,16,16,"The Data Readed Is: ");//提示传送完成

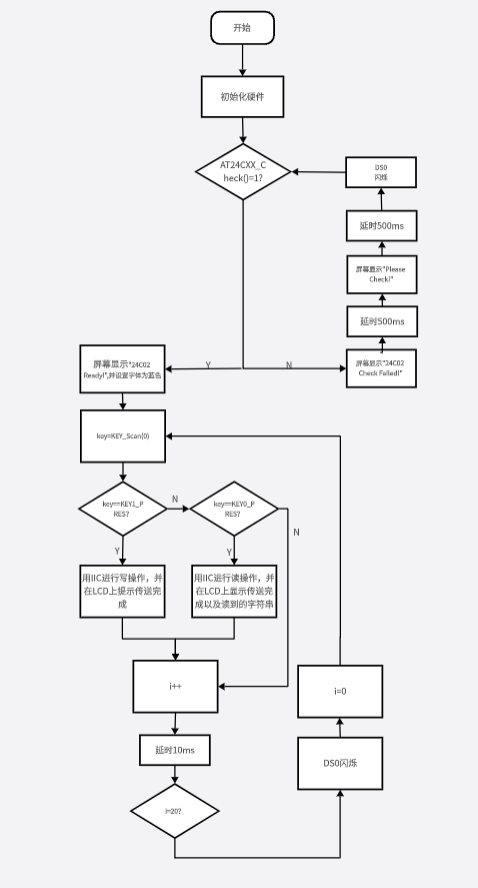
LCD\_ShowString(30,190,200,16,16,datatemp);//显示读到的字符串

}

}

}

1. 实验流程图



1. 实验结果

本实验通过KEY1按键来控制24C02的写入，通过另外一个按键KEY0来控制24C02的读取。并在LCD模块上面显示相关信息。同时，我们可以通过USMART控制在24C02的任意地址写入和读取数据。

我们源代码的基础上将写的内容由固定的变为可修改的，每次按压KEY1按键来控制24C02的写入“b”，按压按键KEY00来控制24C02的读取IIC上的新内容。

|  |  |
| --- | --- |
| 修改前读操作 | 修改前写操作 |
| 4cc46bec41f525b27f8789a91bf106e | 84fb858d740cf1f1d4cef73197807fb |
| 修改后读操作 | 修改后写操作 |
| f0e83db6cc33248127cd27c324cb298 | af2283e4ff047db7e9af296db299012 |

**《嵌入式系统综合实践》实验报告（10）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：SPI**实验**

1. 实验代码

**部分源代码：**

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "lcd.h"

#include "sdram.h"

#include "w25qxx.h"

//要写入到W25Q16的字符串数组

const u8 TEXT\_Buffer[]={"Apollo STM32F4 SPI TEST"};

#define SIZE sizeof(TEXT\_Buffer)

int main(void)

{

u8 key;

u16 i=0;

u8 datatemp[SIZE];

u32 FLASH\_SIZE;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

W25QXX\_Init(); //W25QXX初始化

POINT\_COLOR=RED;

LCD\_ShowString(30,50,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,70,200,16,16,"SPI TEST");

LCD\_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,110,200,16,16,"2016/1/16");

LCD\_ShowString(30,130,200,16,16,"KEY1:Write KEY0:Read"); //显示提示信息

while(W25QXX\_ReadID()!=W25Q256) //检测不到W25Q256

{

LCD\_ShowString(30,150,200,16,16,"W25Q256 Check Failed!");

delay\_ms(500);

LCD\_ShowString(30,150,200,16,16,"Please Check! ");

delay\_ms(500);

LED0=!LED0; //DS0闪烁

}

LCD\_ShowString(30,150,200,16,16,"W25Q256 Ready!");

FLASH\_SIZE=32\*1024\*1024; //FLASH 大小为32M字节

POINT\_COLOR=BLUE; //设置字体为蓝色

while(1)

{

key=KEY\_Scan(0);

if(key==KEY1\_PRES)//KEY1按下,写入W25Q128

{

LCD\_Fill(0,170,239,319,WHITE);//清除半屏

LCD\_ShowString(30,170,200,16,16,"Start Write W25Q256....");

W25QXX\_Write((u8\*)TEXT\_Buffer,FLASH\_SIZE-100,SIZE); //从倒数第100个地址处开始,写入SIZE长度的数据

LCD\_ShowString(30,170,200,16,16,"W25Q256 Write Finished!"); //提示传送完成

}

if(key==KEY0\_PRES)//KEY0按下,读取字符串并显示

{

LCD\_ShowString(30,170,200,16,16,"Start Read W25Q256.... ");

W25QXX\_Read(datatemp,FLASH\_SIZE-100,SIZE); //从倒数第100个地址处开始,读出SIZE个字节

LCD\_ShowString(30,170,200,16,16,"The Data Readed Is: "); //提示传送完成

LCD\_ShowString(30,190,200,16,16,datatemp); //显示读到的字符串

}

i++;

delay\_ms(10);

if(i==20)

{

LED0=!LED0;//提示系统正在运行

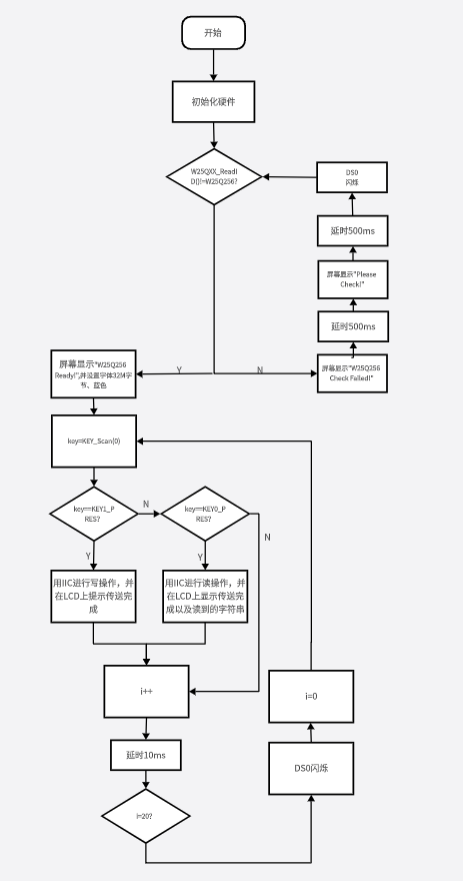
i=0;

}

}

}

1. 实验流程图



1. 实验结果

本实验通过KEY1按键来控制W25Q256的写入，通过另外一个按键KEY0来控制W25Q256的读取。并在LCD模块上面显示相关信息。DS0提示程序正在运行。同时，我们可以通过USMART控制读取W25QXX的ID或者整片擦除。

|  |  |
| --- | --- |
| 读操作 | 写操作 |
| 72ba9ca0d147f6547df0a39b0a6aead | 2b81a4cb8b29b55a4c9bab9e0933d26 |

1. 实验心得（9）（10）

在实验之前，我首先复习了IIC和SPI协议的工作原理和特点。IIC是一种串行通信协议，具有双向通信能力，最多可以连接多个设备。SPI是一种同步通信协议，具有较高的通信速度，通常用于与高速缓存器、ADC（模数转换器）等高速设备通信。通过这次实验，我对STM32F429的IIC和SPI接口有了更深入的了解，了解到如何用软件模拟IIC，掌握了如何在实际项目中使用这两种通信协议进行读写操作。

**《嵌入式系统综合实践》实验报告（11）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称： 网络通信实验

一、实验代码

**部分源代码：**

#include "sys.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "lcd.h"

#include "sdram.h"

#include "lan8720.h"

#include "timer.h"

#include "pcf8574.h"

#include "adc.h"

#include "rtc.h"

#include "malloc.h"

#include "lwip/netif.h"

#include "lwip\_comm.h"

#include "lwipopts.h"

#include "tcp\_client\_demo.h"

#include "tcp\_server\_demo.h"

#include "udp\_demo.h"

#include "httpd.h"

//加载UI

//mode:

//bit0:0,不加载;1,加载前半部分UI

//bit1:0,不加载;1,加载后半部分UI

void lwip\_test\_ui(u8 mode)

{

u8 speed;

u8 buf[30];

POINT\_COLOR=RED;

if(mode&1<<0)

{

LCD\_Fill(30,30,lcddev.width,110,WHITE); //清除显示

LCD\_ShowString(30,30,200,16,16,"Apollo STM32F4/F7");

LCD\_ShowString(30,50,200,16,16,"Ethernet lwIP Test");

LCD\_ShowString(30,70,200,16,16,"ATOM@ALIENTEK");

LCD\_ShowString(30,90,200,16,16,"2016/2/29");

}

if(mode&1<<1)

{

LCD\_Fill(30,110,lcddev.width,lcddev.height,WHITE); //清除显示

LCD\_ShowString(30,110,200,16,16,"lwIP Init Successed");

if(lwipdev.dhcpstatus==2)sprintf((char\*)buf,"DHCP IP:%d.%d.%d.%d",lwipdev.ip[0],lwipdev.ip[1],lwipdev.ip[2],lwipdev.ip[3]);//打印动态IP地址

else sprintf((char\*)buf,"Static IP:%d.%d.%d.%d",lwipdev.ip[0],lwipdev.ip[1],lwipdev.ip[2],lwipdev.ip[3]);//打印静态IP地址

LCD\_ShowString(30,130,210,16,16,buf);

speed=LAN8720\_Get\_Speed();//得到网速

if(speed&1<<1)LCD\_ShowString(30,150,200,16,16,"Ethernet Speed:100M");

else LCD\_ShowString(30,150,200,16,16,"Ethernet Speed:10M");

LCD\_ShowString(30,170,200,16,16,"KEY0:TCP Server Test");

LCD\_ShowString(30,190,200,16,16,"KEY1:TCP Client Test");

LCD\_ShowString(30,210,200,16,16,"KEY2:UDP Test");

}

}

int main(void)

{

u8 t;

u8 key;

HAL\_Init(); //初始化HAL库

Stm32\_Clock\_Init(360,25,2,8); //设置时钟,180Mhz

delay\_init(180); //初始化延时函数

uart\_init(115200); //初始化USART

LED\_Init(); //初始化LED

KEY\_Init(); //初始化按键

SDRAM\_Init(); //初始化SDRAM

LCD\_Init(); //初始化LCD

PCF8574\_Init(); //初始化PCF8574

MY\_ADC\_Init(); //初始化ADC

RTC\_Init(); //初始化RTC

TIM3\_Init(1000-1,900-1); //定时器3初始化，定时器时钟为90M，分频系数为900-1，

//所以定时器3的频率为90M/900=100K，自动重装载为1000-1，那么定时器周期就是10ms

my\_mem\_init(SRAMIN); //初始化内部内存池

my\_mem\_init(SRAMEX); //初始化外部内存池

my\_mem\_init(SRAMCCM); //初始化CCM内存池

POINT\_COLOR=RED;

LED0=0;

lwip\_test\_ui(1); //加载前半部分UI

LCD\_ShowString(30,110,200,16,16,"lwIP Initing...");

while(lwip\_comm\_init()) //lwip初始化

{

LCD\_ShowString(30,110,200,20,16,"LWIP Init Falied! ");

delay\_ms(500);

LCD\_ShowString(30,110,200,16,16,"Retrying... ");

delay\_ms(500);

}

LCD\_ShowString(30,110,200,20,16,"LWIP Init Success!");

LCD\_ShowString(30,130,200,16,16,"DHCP IP configing..."); //等待DHCP获取

#if LWIP\_DHCP //使用DHCP

while((lwipdev.dhcpstatus!=2)&&(lwipdev.dhcpstatus!=0XFF))//等待DHCP获取成功/超时溢出

{

lwip\_periodic\_handle(); //LWIP内核需要定时处理的函数

}

#endif

lwip\_test\_ui(2); //加载后半部分UI

httpd\_init(); //HTTP初始化(默认开启websever)

while(1)

{

key=KEY\_Scan(0);

switch(key)

{

case KEY0\_PRES://TCP Server模式

tcp\_server\_test();

lwip\_test\_ui(3);//重新加载UI

break;

case KEY1\_PRES://TCP Client模式

tcp\_client\_test();

lwip\_test\_ui(3);//重新加载UI

break;

case KEY2\_PRES://UDP模式

udp\_demo\_test();

lwip\_test\_ui(3);//重新加载UI

break;

}

lwip\_periodic\_handle();

delay\_ms(2);

t++;

if(t==100)LCD\_ShowString(30,230,200,16,16,"Please choose a mode!");

if(t==200)

{

t=0;

LCD\_Fill(30,230,230,230+16,WHITE);//清除显示

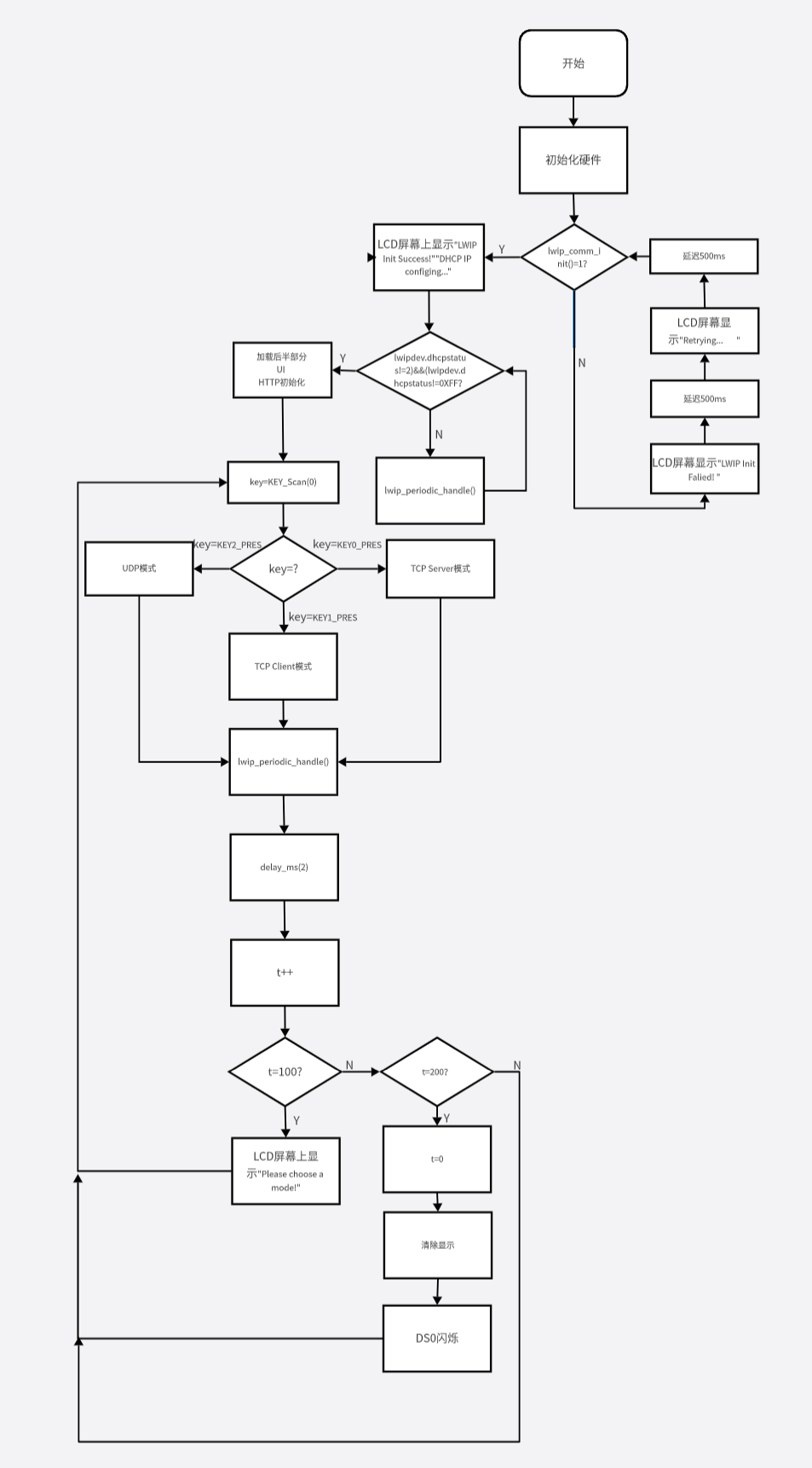
LED0=!LED0;

}

}

}

二、实验流程图



三、实验结果

|  |  |
| --- | --- |
|  | |
| TCP Server | |
|  |  |
| TCP Client | |
|  |  |
| UDP测试 | |
|  |  |
| Web Server | |
|  |  |

本实验开机后，程序初始化LWIP，包括：初始化LAN8720A、申请内存、开启DHCP服务、添加并打开网卡，然后等待DHCP获取IP成功，当DHCP获取成功后，将在LCD屏幕上显示DHCP得到的IP地址，如果DHCP获取失败，那么将使用静态IP（固定为：192.168.1.30），然后开启Web Server服务，并进入主循环，等待按键输入选择需要测试的功能：

KEY0按键，用于选择TCP Server测试功能。

KEY1按键，用于选择TCP Client测试功能

KEY2按键，用于选择UDP测试功能

TCP Server测试的时候，按KEY0发送数据给电脑，电脑端发送过来的数据将会显示在LCD屏幕上。按KEY\_UP可以退出TCP Server测试。

TCP Client测试的时候按KEY1发送数据给电脑，电脑端发送过来的数据将会显示在LCD屏幕上。按KEY\_UP可以退出TCP Server测试。

UDP测试的时候先按开发板KEY2，发送一次数据给电脑，随后才可以电脑发送数据给开发板，实现数据互发。按KEY\_UP可以退出UDP测试。

Web Server的测试在Web界面，可以实现对DS1(LED1)的控制、蜂鸣器的控制、查看ADC1通道5的值、内部温度传感器温度值以及查看RTC时间和日期等。

DS0用于提示程序正在运行。

四、实验心得

通过该实验，我学习到如何用STM32f429进行网络通信。在实验中，我实现了TCP Server、TCP Client、UDP和WebServer的通信功能。通过这些实验，我对嵌入式网络通信的原理和编程方法有了更深入的理解。实验让我认识到，网络通信的关键在于正确处理数据包的发送和接收，以及熟练运用多线程编程。此外，实验还让我了解到实际应用中可能遇到的问题，如网络延迟和数据丢失等，并学会了如何去解决这些问题。总之，这次实验加深了我对网络通信的理解，提高了我的编程能力，为以后从事相关领域的工作奠定了基础。

**《嵌入式系统综合实践》实验报告（12）**

姓名：孟依然 班级：信工211 学号：21012909成绩：

实验名称：开发板间的串口通信

1. 实验代码

**部分源代码及注释：**

**第一个板子：**

#include "led.h"

#include "delay.h"

#include "key.h"

#include "sys.h"

#include "usart.h"

int main(void)

{

u16 t;

u16 len;

u16 times=0;

delay\_init(); //延时函数初始化

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设置NVIC中断分组2:2抢占优先级，2位响应优先级

uart\_init(115200); //串口初始化为115200

LED\_Init(); //LED端口初始化

KEY\_Init(); //初始化与按键连接的硬件接口

while(1)

{

if(USART\_RX\_STA&0x8000)

{

len=USART\_RX\_STA&0x3fff;//得到此次接收到的数据长度

// printf("\r\n您发送的消息为:\r\n\r\n");

for(t=0;t<len;t++)

{

USART\_SendData(USART1, USART\_RX\_BUF[t]);//向串口1发送数据

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET);//等待发送

}

printf("\r\n\r\n");//插入换行

USART\_RX\_STA=0;

}else

{

times++;

if(times%500==0)printf("abcabc\r\n");

if(times%30==0)LED0=!LED0;//闪烁LED灯，提示系统正在运行 delay\_ms(10);

}

}

}

**第二个板子：**

#include "stm32fxxx.h"

#include "delay.h"

#include "led.h"

#include "usart.h"

int main(void)

{

u16 t;

u16 len;

u16 times=0;

delay\_init();

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

uart\_init(115200);

LED\_Init();

while(1)

{

if(USART\_RX\_STA&0x8000)

{

len=USART\_RX\_STA&0x3fff;

// printf("\r\n您发送的消息为\r\n\r\n");

for(t=0;t<len;t++)

{

USART\_SendData(USART1,USART\_RX\_BUF[t]); //向串口1发送数据

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET); //等待发送结束

}

printf("\r\n\r\n"); //插入换行

USART\_RX\_STA=0;

}

else{

times++;

// if(times%500==0)printf("senddata\r\n");

if(times%30==0) LED0=!LED0;

delay\_ms(10);

}

}

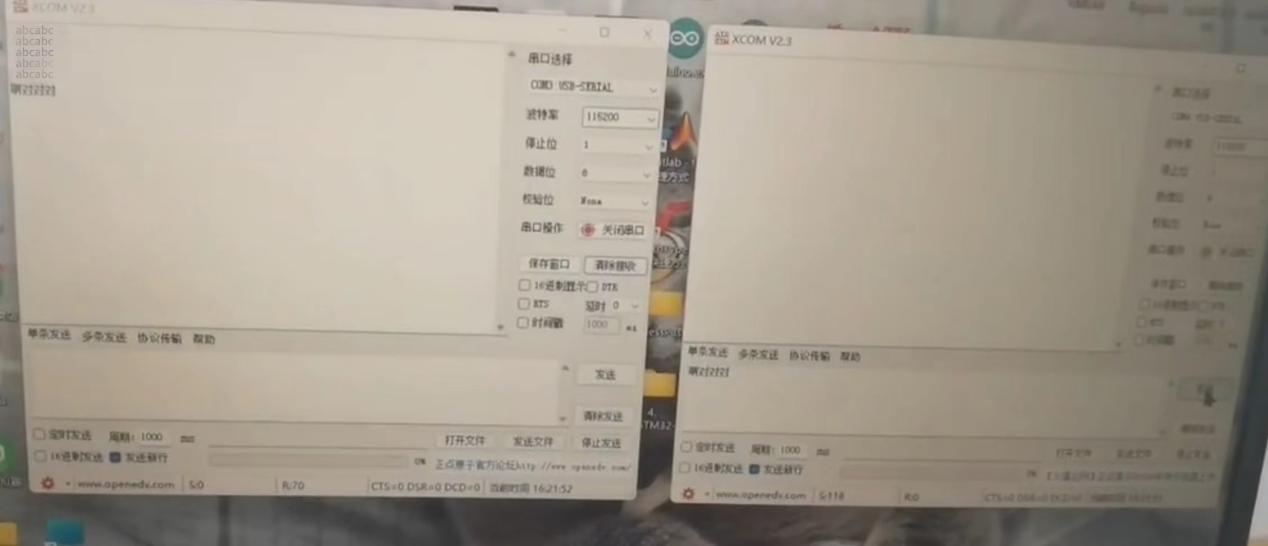
}

1. 实验思路

Stm32开发板1号开两个串口1和2，串口2用来接收数据然后将数据给串口1。Stm32开发板2号开两个串口1和3，串口3接收数据并发送给串口1，两个开发板都通过串口连接电脑的串口调试助手，显示收发的数据。

1. 实验结果

本实验,两个STM32板子通过串口对话，一个作为接收方一个作为发送方。两个板子分别下载程序后，DS0闪烁，提示程序在运行，同时每隔一定时间，通过串口1输出一段信息到电脑。



四、实验心得

本次实验是综合实验，需要自主设计完成一套完整采集+传输+显示等综合设计实践，经老师的点拨，我们首先想到的是两个开发板之间的串口通讯，既运用了所学的串口通讯的知识，还进行了拓展，实现了采集传输和显示的功能，在完成本实验的过程中，遇到了很多困难，我们参考了许多网上的资料，对资料进行合理的修改和运用，最终完成了实验，在这期间加深了我们对串口通讯知识点的学习，学习的知识更加具象化，对stm32开发板的使用更加灵活提高了动手实践能力。