**嵌入式代码**

**实验一**

**使LED以0.5s的速率闪烁。**

#include "mbed.h"

DigitalOut led(LED1);

int main() {

    while (1) {

        led = !led;//改变led的状态

        wait(0.5);//延迟0.5s

    }

}

**1-2在串行终端中显示“Hello World”。**

#include "mbed.h"

int main() {

    printf("Hello World\n");

}

**1-3使4个LED以不同的频率闪烁（0.2秒、0.5秒、1.0秒和2.0秒）。 在串行终端中打印每个LED的状态。**

#include "mbed.h"

int main() {

    //定义四个led输出，并给出其引脚

    DigitalOut led1(p5);

    DigitalOut led2(p6);

    DigitalOut led3(p7);

    DigitalOut led4(p8);

    //设置四个LED的初始状态

    led1=0;

    led2=0;

    led3=0;

    led4=0;

    int i = 0;//因为最小周期为0.1s，所有通过i来表示循环的次数

    while (1) {

        i+=1;

        if(i%2==0){

            led1=!led1;//改变led1的状态

            printf("led1 is now %d\n",led1.read());//输出当前led1的状态

        }

        if(i%5==0){

            led2=!led2;

            printf("led2 is now %d\n",led2.read());

        }

        if(i%10==0){

            led3=!led3;

            printf("led3 is now %d\n",led3.read());

        }

        if(i%20==0){

            led4=!led4;

            printf("led4 is now %d\n",led4.read());

        }

        wait(0.1);//最小周期为0.1秒

    }

}

**实验二**

**2-1 定时器中断。**

#include"mbed.h"

Timeout flipper;

DigitalOut led1(LED1);

DigitalOut led2(LED2);

void flip(){

    led2=!led2;

}

int main(){

    led2=1;

    flipper.attach(&flip,2);//两秒后执行flip函数

    while(1){

        led1 = !led1;

        wait(0.5);

    }

}

**2-2 在LCD中显示“Hello World”消息。**

#include "mbed.h"

#include"C12832.h"" //显示屏库

C12832 lcd(SPI\_MOSI,SPI\_SCK,SPI\_MISO,p8,p11);//显示屏

int main() {

    lcd.printf("Hello World!\n");

}

**2-3 使用定时器中断设置LED闪烁频率，当LED亮起时显示“XXXX”，当LED熄灭时显示“OOOO”。可以选择四种闪烁频率：0.2秒、0.5秒、1.0秒和2.0秒**

#include"mbed.h"

#include"C12832.h"

//中断

Timeout flipper;

Timeout flipper2;

Timeout flipper3;

//闪烁频率

float rate = 2;//2秒

void flip(){

    rate = 1.5;

}

void flip2(){

    rate = 1.0;

}

void flip3(){

    rate = 0.5;

}

DigitalOut led1(LED1);//led灯

C12832 lcd(SPI\_MOSI,SPI\_SCK,SPI\_MISO,p8,p11);//lcd显示屏

int main(){

    led1=0;

    flipper.attach(&flip,4);//4秒后修改闪烁频率为1.5s

    flipper2.attach(&flip2,8);//8秒后修改闪烁频率为1.0s

    flipper3.attach(&flip3,12);//12秒后修改闪烁频率为0.5s

    while(1){

        led1=!led1;//改变led1的状态

        //判断led1的状态

        if(led1==1){//led1亮

            lcd.locate(40,10);

            lcd.printf("X X X X");

        }

        else{

            lcd.locate(40,10);

            lcd.printf("O O O O");

        }

        wait(rate);//闪烁频率

        printf("Hello World! %f\n",rate);

    }

}

**实验三**

**3-1 将8个不同的LED连接到同一个引脚，并使所有LED闪烁。**

#include"mbed.h"

DigitalOut led(p5);//led全部连接在p5引脚上

int main(){

    led = 1;

    while(1){

       led = !led;

       printf("LED is %d\n",led.read());

       wait(0.5);

    }

}

**3-2 在3个LED亮起时使其余5个LED闪烁（只需使用不同的引脚）。**

#include"mbed.h"

DigitalOut led(p5);//5个led连接在p5引脚上

DigitalOut led2(p6);//3个led连接在p6引脚上

int main(){

    led = 1;

    while(1){

       led2 = !led2;

       wait(0.5);

    }

}

**3-3 将1个LED连接到开关。**

#include "mbed.h"

DigitalOut led(LED1);

DigitalIn toggle1(p21);//传感器——开关

int main() {

    while (1) {

        led = 0;

        if(toggle1==1){

           led = 1;

       }

        wait\_ms(500);

    }

}

**3-4 将8个不同的开关连接至8个LED，并在LCD上显示传感器的激活情况**

#include "mbed.h"

#include"C12832.h"

C12832 lcd(SPI\_MOSI,SPI\_SCK,SPI\_MISO,p8,p11);//显示屏定义初始化

//led定义初始化

DigitalOut led1(p5),led2(p6),led3(p7),led4(p8),led5(p9),led6(p10),led7(p11),led8(p12);

//传感器（开关）定义初始化

DigitalIn toggle1(p21),toggle2(p22),toggle3(p23),toggle4(p24),toggle5(p25),toggle6(p26),toggle7(p27),toggle8(p28);//传感器——开关

int main() {

    while (1) {

        if(toggle1==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 1");

            led1=1;

            wait(0.5);

            led1=0;

        }

        if(toggle2==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 2");

            led2=1;

            wait(0.5);

            led2=0;

        }

        if(toggle3==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 3");

            led3=1;

            wait(0.5);

            led3=0;

        }

        if(toggle4==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 4");

            led4==1;

            wait(0.5);

            led4=0;

        }

        if(toggle5==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 5");

            led5=1;

            wait(0.5);

            led5=0;

        }

        if(toggle6==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 6");

            led6=1;

            wait(0.5);

            led6=0;

        }

        if(toggle7==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 7");

            led7=1;

            wait(0.5);

            led7=0;

        }

        if(toggle8==1){

            lcd.locate(0,3);//文字输出位于0，3位置

            lcd.printf("Sensor 8");

            led8=1;

            wait(0.5);

            led8=0;

        }

        wait\_ms(500);

    }

}

**实验四**

4-1 激活触摸检测并显示在屏幕上绘制的内容。

#include "mbed.h"

#include"stm32f413h\_discovery\_ts.h"

#include"stm32f413h\_discovery\_lcd.h"

TS\_StateTypeDef TS\_State = {0};

int main() {

    printf("Draw on the screen\n");

    //初始化

    if(BSP\_TS\_Init(BSP\_LCD\_GetXSize(),BSP\_LCD\_GetYSize())==TS\_ERROR){

        printf("初始化失败\n");

    }

    //清屏

    BSP\_LCD\_Clear(LCD\_COLOR\_WHITE);//

    //输出提示框属性设置

    BSP\_LCD\_FillRect(0,0,BSP\_LCD\_GetXSize(),40);//输出提示框大小

    BSP\_LCD\_SetTextColor(LCD\_COLOR\_WHITE);//字体颜色

    BSP\_LCD\_SetBackColor(LCD\_COLOR\_BLACK);//背景颜色

    BSP\_LCD\_SetFont(&Font20);//字体大小

    BSP\_LCD\_DisplayStringAt(0,15,(uint8\_t \*)"Touch the screen",CENTER\_MODE);//输出提示，并居中

    while (1) {

        BSP\_TS\_GetState(&TS\_State);

        if(TS\_State.touchDetected){//触摸屏有触摸提示

            uint16\_t x1 = TS\_State.touchX[0];//触摸点的x坐标

            uint16\_t y1 = TS\_State.touchY[0];//触摸点y坐标

            BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLUE);//触摸点颜色

            BSP\_LCD\_FillCircle(x1,y1,2);//触摸点的半径

        }

        wait\_ms(5);

    }

}

4-2 在触摸屏上绘制4×4键盘，接收来自虚拟键盘的输入，并将 其显示在LCD上。

#include "mbed.h"

#include"C12832.h"

#include"stm32f413h\_discovery\_ts.h"

#include"stm32f413h\_discovery\_lcd.h"

TS\_StateTypeDef TS\_State = {0};//触摸屏

C12832 lcd(SPI\_MOSI,SPI\_SCK,SPI\_MISO,p8,p11);//显示屏定义初始化

//键盘——二维字符串数组定义

const char \* keyboard[][4] = {

{"1","2","3","F"},{"4","5","6","E"},{"7","8","9","D"},{"A","0","B","C"}

};

int c = 0,b = 0;

//键盘键背景输出

void BackPrint(){

//背景颜色

BSP\_LCD\_SetTextColor(LCD\_COLOR\_LIGHTGRAY);//背景颜色

//输出背景框（起点坐标）

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

BSP\_LCD\_FillRect(7.5+j\*(5+52.5),7.5+i\*(5+52.5),52.5,52.5);//起点坐标不同，每个键的背景框大小相同（均为52.5\*52.5）

}

}

return;

}

//键盘字输出

void CharacterPrint(){

//键盘字属性

BSP\_LCD\_SetBackColor(LCD\_COLOR\_LIGHTGRAY);//字体背景颜色

BSP\_LCD\_SetTextColor(LCD\_COLOR\_WHITE);//字体颜色

BSP\_LCD\_SetFont(&Font20);//字体大小

//输出

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

BSP\_LCD\_DisplayStringAt(52.5/3+7.5+j\*(5+52.5),52.5/3+7.5+i\*(5+52.5),(uint8\_t \*)keyboard[i][j],LEFT\_MODE);

}

}

}

void KeyboardPrint(){

BackPrint();//背景输出

CharacterPrint();//键盘字输出

}

void CharacterTouch(uint16\_t x1,uint16\_t y1){

//水平方向判断

if(x1>7.5 && x1<60) c = 0;

if(x1>65&&x1<117.5) c =1;

if(x1>122.5&&x1<175) c =2;

if(x1>180&&x1<232.5) c =3;

//竖直方向判断

if(y1>7.5 && y1<60) b = 0;

if(y1>65&&y1<117.5) b =1;

if(y1>122.5&&y1<175) b =2;

if(y1>180&&y1<232.5) b =3;

}

int main() {

printf("Draw on the screen\n");

//初始化

if(BSP\_TS\_Init(BSP\_LCD\_GetXSize(),BSP\_LCD\_GetYSize())==TS\_ERROR){

printf("初始化失败\n");

}

//清屏

BSP\_LCD\_Clear(LCD\_COLOR\_WHITE);

BSP\_LCD\_FillRect(0,0,BSP\_LCD\_GetXSize(),BSP\_LCD\_GetYSize());

BSP\_LCD\_SetBackColor(LCD\_COLOR\_BLACK);//键盘整体背景颜色

KeyboardPrint();//键盘输出

while(1){

BSP\_TS\_GetState(&TS\_State);

if(TS\_State.touchDetected){//触摸屏有触摸提示

uint16\_t x1 = TS\_State.touchX[0];//触摸点的x坐标

uint16\_t y1 = TS\_State.touchY[0];//触摸点y坐标

CharacterTouch(x1,y1);//根据输入坐标，判断输出哪个键盘字

lcd.locate(60,10);

lcd.printf(keyboard[b][c]);//注意坐标与二维字符串数组对应关系

}

wait\_ms(1);

}

}

**大作业**

#include "mbed.h"

#include"C12832.h"

#include"stm32f413h\_discovery\_ts.h"

#include"stm32f413h\_discovery\_lcd.h"

#include<string.h>

TS\_StateTypeDef TS\_State = {0};//触摸屏

C12832 lcd(SPI\_MOSI,SPI\_SCK,SPI\_MISO,p8,p11);//显示屏定义初始化

DigitalOut led(p5);//二极管连接在p5引脚上

//键盘——二维字符串数组定义

char \* keyboard[][3] = {

{"1","2","3"},{"4","5","6"},{"7","8","9"},{"C","0","#"}

};

//密码锁密码

char \*LockPsw[]={"2","0","1","3","1","4"};

//提示语设置为字符串数组，是便于进行替换

char \*Prompt[]={"Code : ","\_"," ","\_"," ","\_"," ","\_"," ","\_"," ","\_"};

int col = 0,row = 0;//全局变量——用来记录输入的密码

void lcd\_Init();//lcd显示屏初始化

void Keyboard\_Init();//触摸屏初始化

void Back\_Print();//键盘键背景输出

void Character\_Print();//键盘字输出

void Character\_Judge(uint16\_t x1,uint16\_t y1);//判断输出的键盘字

bool Login(uint16\_t x1,uint16\_t y1);//判断是否输入密码

bool Psw\_Input();

int main() {

int cnt = 3;//输入错误的次数

Keyboard\_Init();//触摸屏初始化

lcd\_Init();//显示屏初始化

led=0;//二极管初始化（初始状态灭）

while(1){

lcd.cls();

lcd.locate(13,10);

lcd.printf("Press 'C' to continue\n");

BSP\_TS\_GetState(&TS\_State);

if(TS\_State.touchDetected){

uint16\_t x1 = TS\_State.touchX[0];//触摸点的x坐标

uint16\_t y1 = TS\_State.touchY[0];//触摸点y坐标

if(Login(x1,y1)){

//登录成功——密码正确

lcd.cls();

lcd.locate(12,10);

lcd.printf("Login successfully");

//二极管常亮

led = 1;

wait(8);

led = 0;

break;

}else{

//登陆失败

cnt-=1;

//判断失败次数

if(cnt==0){

lcd.cls();

lcd.locate(12,5);

lcd.printf("Login failed\n");

lcd.locate(12,15);

lcd.printf("No chance");

wait(3);

lcd.cls();

lcd.locate(5,10);

lcd.printf("Please wait 10s to continue");

wait(10);

continue;

}

lcd.cls();

lcd.locate(16,5);

lcd.printf("Login failed\n");

lcd.locate(16,17);

lcd.printf("%d chance\n",cnt);

//二极管闪烁——闪5次

int i = 0;

while(i++<10){

led=!led;

wait(0.4);

}

lcd.cls();

continue;//跳过当前循环

}

}

}

lcd.cls();

lcd.locate(12,10);

lcd.printf("Welcome Next");

}

void lcd\_Init(){

lcd.locate(5,10);

lcd.printf("Welcome to The Code Lock\n");//LCD显示屏初始化

wait(3);

}

void Character\_Print(){

//键盘字属性

BSP\_LCD\_SetBackColor(LCD\_COLOR\_LIGHTGRAY);//字体背景颜色

BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLUE);//字体颜色

BSP\_LCD\_SetFont(&Font20);//字体大小

//输出

for(int i=0;i<4;i++){

for(int j=0;j<3;j++){

BSP\_LCD\_DisplayStringAt(70/3+8+j\*(7+70),52.5/3+7.5+i\*(5+52.5),(uint8\_t \*)keyboard[i][j],LEFT\_MODE);

}

}

}

void Back\_Print(){

//背景颜色

BSP\_LCD\_SetTextColor(LCD\_COLOR\_LIGHTGRAY);//背景颜色

//输出背景框（起点坐标）

for(int i=0;i<4;i++){

for(int j=0;j<3;j++){

BSP\_LCD\_FillRect(8+j\*(7+70),8+i\*(5+52.5),70,52.5);//起点坐标不同，每个键的背景框大小相同（均为52.5\*52.5）

}

}

return;

}

void Keyboard\_Init(){

//初始化

if(BSP\_TS\_Init(BSP\_LCD\_GetXSize(),BSP\_LCD\_GetYSize())==TS\_ERROR){

printf("Initialized error\n");

}

//清屏

BSP\_LCD\_Clear(LCD\_COLOR\_WHITE);

BSP\_LCD\_FillRect(0,0,BSP\_LCD\_GetXSize(),BSP\_LCD\_GetYSize());

BSP\_LCD\_SetBackColor(LCD\_COLOR\_BLACK);//键盘整体背景颜色

//键盘输出

Back\_Print();//背景色

Character\_Print();//键盘字

}

void Character\_Judge(uint16\_t x1,uint16\_t y1){

//水平方向判断

if(x1>8 && x1<78) col = 0;

if(x1>85&&x1<155) col =1;

if(x1>162&&x1<232) col =2;

//竖直方向判断

if(y1>7.5 && y1<60) row = 0;

if(y1>65&&y1<117.5) row =1;

if(y1>122.5&&y1<175) row =2;

if(y1>180&&y1<232.5) row =3;

}

bool Login(uint16\_t x1,uint16\_t y1){

//提示字符串数组重新更新——将第一遍的\*改成\_

int i = 0;

while(i<6){

Prompt[2\*i+1]="\_";

i++;

}

Character\_Judge(x1,y1);//根据输入坐标，判断输出哪个键盘字

lcd.cls();

loop1:

if(keyboard[row][col]=="C"){

lcd.locate(12,10);

//输出提示语

for(int i=0;i<12;i++){

lcd.printf(Prompt[i]);

}

//输入密码并进行比对

if(Psw\_Input()){

return true;

}

else{//密码输入错误一次

return false;

}

}else{

lcd.locate(35,5);

lcd.printf("Illegal input！\n");

lcd.locate(18,15);

lcd.printf("Press 'C' to continue\n");

goto loop1;

}

}

//

bool Psw\_Input(){

//密码输入

char \*Psw[6]={""};//字符串数组

//总共需要输入六次

int i = 0;

while(i<6){

BSP\_TS\_GetState(&TS\_State);

if(TS\_State.touchDetected){

uint16\_t x1 = TS\_State.touchX[0];//触摸点的x坐标

uint16\_t y1 = TS\_State.touchY[0];//触摸点y坐标

Character\_Judge(x1,y1);//根据输入坐标，判断输出哪个键盘字

if(keyboard[row][col]!="C"){//如果不为C，则进行字符串连接

//替换短杠为星号

Prompt[2\*i+1]="\*";//\_短杠均为奇数位，用\*进行替换

//输出提示语

lcd.cls();

lcd.locate(12,10);

for(int j=0;j<12;j++){

lcd.printf(Prompt[j]);

}

Psw[i]=keyboard[row][col];//字符串数组不能用strcpy函数（针对字符串）

i++;//只有输入有效时，才会加一

}

else{//为C删除一位

//将\*符号替换

Prompt[2\*(i-1)+1]="\_";

//将上一位输入的字符修改

Psw[i-1]="";//最后添加的一位改为'\0'

//输出提示语

lcd.cls();

lcd.locate(12,10);

for(int j=0;j<12;j++){

lcd.printf(Prompt[j]);

}

i--;//删去一位数字，有效位数减一

}

wait(0.2);//延迟0.3s

}

}

//密码检查

int flag = -1;//判断变量

for(int i=0;i<6;i++){

if(strcmp(Psw[i],LockPsw[i])!=0){//不相等

flag = 1;

}

}

if(flag == 1){

return false;

}else{

return true;

}

}