Copyright

目录

Delay @24MHz

```c
// delay.c
#include "__type__.h"

extern void _nop_(void);

void Delay5us(u8 t)
{ // 误差 0%
    u8 i;
    _nop_();
    _nop_();
    while (t)
    {
        i = t == 1 ? 21 : 26;
        while (--i)
            ;
        t--;
    }
}

void Delay1ms(u16 t)
{ // 平均误差不足 0.005% 越大误差越小，几乎是无误差
    u8 i, j;
    while (t)
    {
        for (i = 24; i; --i)
            for (j = 248; j; --j)
                ;
        for (i = 8; i; --i)
            ;
        --t;
    }
}
```

__config__.h

```h
#ifndef __CONFIG___H
#define __CONFIG___H

#include <STC15.H>

#include "__type__.h"

/* ========================================================================== */

#define MODULE 1 // 0: 智慧农业; 1: 智能小车; 2: 智能音箱; 3: 工业互联网

// 兼容 sbit 的引脚定义
#define _SBIT(P, B) P ^ (B)
#define SBIT(P)     _SBIT(P) // 延迟宏展开 : Px,B - > Px^B

/* ========================================================================== */
// 公共资源

// unused
#define UNUSED_DHT11
#define UNUSED_STEP_MOTOR
#define UNUSED_ADC
#define UNUSED_PWM
#define UNUSED_HC_SR04
#define UNUSED_DS18B20

// 计算 ADC
#define VOLTAGE 3.3
#define V_TOTAL 1024

// 需要定义推挽输出的引脚
#define _LCD_RS P2, 0
#define _LCD_RW P2, 1
#define _LCD_EN P2, 2
#define _RELAY  P5, 2
#define _BUZZER P5, 5
// _PWM_LED
// _DC_MOTOR
// _STEP_MOTOR

// main
#define LED    P5 ^ 3        // led
#define RELAY  SBIT(_RELAY)  // relay
#define BUZZER SBIT(_BUZZER) // buzzer
// DCM
// INF
// PWM_LED
```

```
// lcd12864
#define LCD_DATA P0
#define LCD_RS   SBIT(_LCD_RS)
#define LCD_RW   SBIT(_LCD_RW)
#define LCD_EN   SBIT(_LCD_EN)

// key_4x4
#define KEY_4X4_PIN   P7
#define KEY_4X4_DELAY delay_1ms(5)

// i2c, at24c
#define I2C_SDA P6 ^ 6
#define I2C_SCL P6 ^ 7

// ds1302
#define DS1302_SCK P3 ^ 5
#define DS1302_SDA P3 ^ 6
#define DS1302_CE  P5 ^ 4

/* ========================================================================== */
#if MODULE == 0 // 智慧农业

#undef UNUSED_DHT11
#undef UNUSED_ADC
#undef UNUSED_PWM

// dc motor
#define _DC_MOTOR P1, 5
#define DCM       SBIT(_DC_MOTOR)

// dht11
#define DHT11_DQ P1 ^ 1 // 和步进电机冲突，设置为双向IO口

// adc
#define LDR 0 // 光敏电阻 P10

// 红外对管检测
#define INF P1 ^ 4

// pwm led
#define _PWM_LED P1, 7
#define PWM_LED  SBIT(_PWM_LED)
#define PWM_ID   7 // 7: P17

/* ========================================================================== */
#elif MODULE == 1 // 智能小车

#undef UNUSED_STEP_MOTOR
#undef UNUSED_HC_SR04
#undef UNUSED_DS18B20
#undef UNUSED_ADC
#undef UNUSED_PWM
```

```
// dc motor, 可以 pwm, 最好大于50%
#define _DC_MOTOR    P1, 6
#define DCM          SBIT(_DC_MOTOR)

// step motor
#define _STEP_MOTOR  P1, 1       // 风格保持一致
#define STEP_MOTOR   _STEP_MOTOR // 兼容库定义

// hc_sr04
#define HC_SR04_TRIG P1 ^ 5
#define HC_SR04_ECHO P3 ^ 4

// ds18b20
#define DS18B20_DQ   P1 ^ 7

// adc
#define POT          0 // 电位器 P10

// pwm dc motor
#define PWM_ID       6 // 6: P16

// 红外测速 P33 INT1(外部中断1)  IT1 = 1;  EX1 = 1;

/* ================================================================================ */
#elif MODULE == 2      // 智能音箱

#endif

/* ================================================================================ */

// 很多模块可能会用到延迟, 所以放在 config 中申明
extern void _nop_(void);
extern void delay_5us(u8 t);
extern void delay_1ms(u16 t);

// 定义一些有用的宏函数
#define BIT_H(N, B) (N) |= (1 << (B))
#define BIT_L(N, B) (N) &= ~(1 << (B))

static u16 s_count_i; // 在同一个源文件中不可以嵌套, 不同源文件中可以嵌套
#define for_c(N) for (s_count_i = (N); s_count_i; --s_count_i)

#endif // __CONFIG___H
```

__function__.h

```h
#ifndef __FUNCTION___H
#define __FUNCTION___H

#include "__type__.h"

/* ========================================================================== */

// cstdio
extern int sprintf(char*, const char*, ...);
extern int vsprintf(char*, const char*, char*);

// delay
// extern void delay_5us(u8 t);
// extern void delay_1ms(u16 t);

// lcd12864
extern void lcd_cmd(u8 cmd);
extern void lcd_show(u8 dat);
extern void lcd_printf(const char* format, ...);

// key_4x4
extern u8 key_value(void);

// ds1302
extern void ds1302_init(void);
extern void ds1302_set(u8* set);
extern void ds1302_get(u8* get);

// at24c256
extern void at24c_read(u16 addr, u8* dat, u8 len);
extern void at24c_write(u16 addr, u8* dat, u8 len);

/* ========================================================================== */

// dht11
extern bit dht11_read(float* r, float* t);

// ds18b20
extern void ds18b20_convert();
extern float ds18b20_read_temp();

// hc_sr04
extern float hc_sr04_result(void);

// adc
extern u16 adc_result(u8 ch);

// step motor
```

```
extern void step_motor_run(bit s, u16 size);

// pwm
extern void pwm_init(void);
extern void pwm_clk(u8 ps);
extern void pwm_set(u16 first, u16 second, u16 cycles);

#endif // __FUNCTION___H
```

system.c

```c
#include "system.h"

#define KEYS_MAX  8
#define KEYS_ADDR 0x0000

extern u8 buff[];

void lcd_init(void)
{
    lcd_cmd(0x01);
    lcd_cmd(0x0e);
    lcd_cmd(0x30);
}

void lcd_action(void)
{
    u8 code s1[17] = "      ABCD      ";
    u8 code s2[17] = "       07       ";
    u8 i;

    // 字从下往上，屏幕从上往下 0x41 -> 0x50 -> 0x60
    lcd_cmd(0x01);

    lcd_cmd(0x88); // 第三行 -> 屏幕外 -> 屏幕外
    lcd_printf(s1);
    lcd_cmd(0xa0); // 屏幕外 -> 第二行 -> 第一行
    lcd_printf(s1);

    lcd_cmd(0x98); // 第四行 -> 第三行 -> 屏幕外
    lcd_printf(s2);
    lcd_cmd(0xb0); // 屏幕外 -> 屏幕外 -> 第二行
    lcd_printf(s2);

    lcd_cmd(0x34);
    lcd_cmd(0x03);

    i = 0x41;
    for_c(16)
    {
        delay_1ms(1000 / 16);
        lcd_cmd(i++);
    }

    delay_1ms(1000);

    for_c(16)
    {
        delay_1ms(1000 / 16);
```

```
        lcd_cmd(i++);
    }

    lcd_cmd(0x30);
    lcd_cmd(0x01);
}

u8 get_key(void)
{
    key = key_value();
    if ((key = key_value()) != 0xff)
    {
        while (key_value() != 0xff)
            ; // key 和 value 都不等于 0xff, 即松开后(0xff)返回按键值
    }
    return key;
}

void reset_password(void)
{
    bit loop = 1;
    u8 i     = 0, num;

    lcd_cmd(0x01);
    lcd_cmd(0x80);
    lcd_printf("New Password:");
    lcd_cmd(0x90);
    lcd_cmd(0x0e); // 开启光标

    while (loop)
    {
        num = 0xff;
        if (get_key() != 0xff)
        {
            switch (key)
            {
            case 0x00:
                num = 1;
                break;
            case 0x01:
                num = 2;
                break;
            case 0x02:
                num = 3;
                break;
            case 0x04:
                num = 4;
                break;
            case 0x05:
                num = 5;
                break;
            case 0x06:
```

```
                num = 6;
                break;
            case 0x08:
                num = 7;
                break;
            case 0x09:
                num = 8;
                break;
            case 0x0a:
                num = 9;
                break;
            case 0x0d:
                num = 0;
                break;
            case 0x03: // 退格
                if (i != 0)
                {
                    i--;
                    lcd_cmd(0x10);
                    lcd_printf("  ");
                    lcd_cmd(0x10);
                }
                break;
            case 0x0f: // 确认
                loop = 0;
                at24c_write(KEYS_ADDR, &i, 1);
                at24c_write(KEYS_ADDR + 1, buff, i);
                break;
            default:
                break;
            }
            if (num != 0xff && i < KEYS_MAX)
            {
                buff[i] = num + '0';
                lcd_show(buff[i]);
                lcd_show(0x20);
                num = 0xff;
                ++i;
            }
        }
    }
    lcd_cmd(0x0c); // 关闭光标
    lcd_cmd(0x01);
}

void enter_password(void)
{
    bit loop, lock = 1;
    u8 right, num, len, i;

    at24c_read(KEYS_ADDR, &len, 1);
    at24c_read(KEYS_ADDR + 1, buff + KEYS_MAX, len);
```

```
    while (lock)
    {
        lcd_cmd(0x01);
        lcd_cmd(0x80);
        lcd_printf("Enter Password:");
        lcd_cmd(0x90);
        lcd_cmd(0x0e);

        loop  = 1;
        right = 0;
        i     = 0;

        while (loop)
        {
            num = 0xff;
            if (get_key() != 0xff)
            {
                switch (key)
                {
                case 0x00:
                    num = 1;
                    break;
                case 0x01:
                    num = 2;
                    break;
                case 0x02:
                    num = 3;
                    break;
                case 0x04:
                    num = 4;
                    break;
                case 0x05:
                    num = 5;
                    break;
                case 0x06:
                    num = 6;
                    break;
                case 0x08:
                    num = 7;
                    break;
                case 0x09:
                    num = 8;
                    break;
                case 0x0a:
                    num = 9;
                    break;
                case 0x0d:
                    num = 0;
                    break;
                case 0x03: // 退格
                    if (i != 0)
```

```
                    {
                        lcd_cmd(0x10);
                        lcd_printf("  ");
                        if (right == i)
                        {
                            right--;
                        }
                        i--;
                        lcd_cmd(0x10);
                    }
                    break;
                case 0x0f: // 确认
                    loop = 0;
                    break;
                default:
                    break;
                }
                if (num != 0xff && i < KEYS_MAX)
                {
                    buff[i] = num + '0';
                    lcd_show(buff[i]);
                    lcd_show(0x20);
                    if (right == i)
                    {
                        right +=
                            ((buff[i] == buff[i + KEYS_MAX]) || (i >= len));
                    }
                    ++i;
                }
            }
        }
        lcd_cmd(0x0c);
        lcd_cmd(0x01);

        lock = (right != len); // 相等则密码正确

        lcd_cmd(0x90);
        if (!lock)
        {
            lcd_printf("Right Password!");
        }
        else
        {
            lcd_printf("Wrong Password!");
        }
        lcd_cmd(0x88);
        lcd_printf("Enter Any Key...");
        while (get_key() == 0xff)
            ;
    }
    lcd_cmd(0x01);
}
```

```c
void updateShow(void)
{
    if (count_t3 > 20)
    {
        lcd_cmd(0x01);
        showTemperature();
        count_t3 = 0;

        lcd_cmd(0x9f);
        lcd_printf("%02d", (u16)count_x1);
        count_x1 = 0;

        lcd_cmd(0x86);
        lcd_printf("%4.2f", adc_result(LDR) * VOLTAGE / 1024); // 显示电压值
    }
    showTime();
}

void showTime(void)
{
    ds1302_get(buff);
    lcd_cmd(0x80);
    lcd_printf("20%02d-%02d-%02d", (u16)buff[6], (u16)buff[4], (u16)buff[3]);
    lcd_cmd(0x90);
    lcd_printf("%02dh %02dm %02ds", (u16)buff[2], (u16)buff[1], (u16)buff[0]);
}

void showTemperature(void)
{
    EA = 0;
    lcd_cmd(0x98);
    lcd_printf("T: %6.2f C", ds18b20_read_temp());
    ds18b20_convert();

    lcd_cmd(0x88);
    lcd_printf("D: %5.1f cm", hc_sr04_result());
    EA = 1;
}
```

main.c

```
#define PM_Z(P)    (P##M1 = P##M0 = 0x00)             // 置零
#define _PPO(P, B) (BIT_L(P##M1, B), BIT_H(P##M0, B)) // 推挽输出
#define PPO(P)     _PPO(P)

void io_init(void)
{
    PM_Z(P0);
    PM_Z(P1);
    PM_Z(P2);
    PM_Z(P3);
    PM_Z(P4);
    PM_Z(P5);
    PM_Z(P6);
    PM_Z(P7);

    // 公共资源推挽

    // LCD12864
    PPO(_LCD_RS); // P20
    PPO(_LCD_RW); // P21
    PPO(_LCD_EN); // P22

    // 继电器
    PPO(_RELAY); // P52

    // 蜂鸣器
    PPO(_BUZZER); // P55

    relay  = 0;
    led    = 1; // 1: 关闭
    buzzer = 1; // 1: 关闭
}

void module_init(void)
{
    // 直流电机
    PPO(_DC_MOTOR); // P15: 智慧农业; P16: 智能小车 可 pwm
    dcm = 0;
#if MODULE == 0
    // 调光LED
    PPO(_PWM_LED);
    pwm_led = 0;
#elif MODULE == 1
    // 步进电机
    PPO(_STEP_MOTOR + 0); // P11, 和智慧农业的温湿度冲突
    PPO(_STEP_MOTOR + 1); // P12
    PPO(_STEP_MOTOR + 2); // P13
    PPO(_STEP_MOTOR + 3); // P14, 和智慧农业的红外冲突
```

```
    // 红外测速
    IT1 = 1;
    EX1 = 1;
#endif
}

void INT_Init(void)
{
    // EA ES ET EX IT(外部中断方式控制)
    // 串口通信: RI TI
    // IE2: - ET4 ET3 ES4 ES3 ET2 - ES2
    // interrupt: t2: 12 t3: 19 t4: 20
    // 定时器/串口可以软件生成

    SCON = 0x50; //REN=1允许串行接受状态，串口工作模式2
    TMOD = 0x00; //定时器1为模式0（16位自动重载）
    AUXR = 0X40; //开启1T模式
    TL1 = (65535 - (24000000 / 4 / 9600)); //设置波特率重装值
    TH1 = (65535 - (24000000 / 4 / 9600)) >> 8;
    TR1 = 1; //开启定时器1
    ES = 1;  //开串口中断

    AUXR &= 0xFB; // 定时器时钟12T模式
    T2L = 0xB0;   // 设置定时初始值
    T2H = 0x3C;   // 设置定时初始值
    AUXR |= 0x10; // 定时器2开始计时

    EA = 1;
    IE2 |= 0x04; // 定时器2中断
    IT1 = 1;     // 外部中断低电平触发
    EX1 = 1;     // 外部中断允许
}
```