

单片机第二次作业

【题目】

- 1、将自己的学号以长整形和浮点的形式保存在片外数据区指定的位置，由自己指定位置，分析其在存储器中的表示方法，并将其换算成对应的十进制数，比较是否存在表示误差。
- 2、将 1602 显示模块，与 STC 单片机实验箱正确连接，并在 1602 上显示学号。
- 3、通过外部按键触发中断，实现学号在 1602 上的左移/右移。

【设计思路】

- 1、在 xdata 区的 0x100 处指定一个长整型变量 num1，长整型变量在存储器内以 32 位的二进制数表示，表示范围为 -2^{31} ~ $2^{31}-1$ ；在 0x120 处制定一个浮点型变量 num2，浮点型变量在存储器内用 32 位的二进制数表示，最高位为符号位，0 表示正数，1 表示负数，第二到九位为指数为，表示指数的阶数，后面的二十三位为数值位，表示为整数位为 1 的一个数，整体与阶数相合来表示数值，在这里我改变了小数点的不同位置，然后看得到的十六进制数与我的学号相差的大小，通过 debug 查看对应存储区的变量值。
- 2、因为输入的变量是长整型，而编写的 lcdwritestr 函数需要传入的是 char 的数组类型，因此，这里最关键的一点是将长整型变量转换为 char 的数组类型，因此用到了 sprintf 函数，通过格式化定义，将 num1 的数据传送到预先定义的字符缓冲器 tstr 中，再将其打印出来。为了使展现效果更加美观，我对数组进行了一定的移位操作，使其能够循环的打印我的学号。
- 3、其实上一步的循环显示已经为解答这一题打下了很好的基础，我只需要改变一下数组转换的方向即可。首先编写外部中断 0 的中断响应函数，使其能够在外部中断 0 触发时实现学号向左移动的功能，在我的中断响应函数 move_left 中，我使需要打印的字符串的地址向后移动，而打印的初始位置保存在 0，0 处不变，这样就可以实现学号的左移操作。而外部中断 1 的中断响应函数是使触发时使学号右移，这里直接将打印的初始位置向右移动即可实现。

【调试过程】

- 1、依照老师所给的环境配置过程配置好相应的环境；
- 2、进入 debug 环境，直接运行；
- 3、在 memory1 处输入 x:0x100，按下回车，查看 num1 的值为 78 29 F0 6E，即十进制的 2016014446，即我的学号；

- 4、在 memory1 处输入 x:0x120，按下回车，查看 num2 的值为 4E F0 53 E1，对应十进制的值为 15750113，显然已经失效；
- 5、在 num2 中加入小数点，重复上一步，查看 num2 的值为 44 FC 00 76，对应十进制为 2016.014404，与输入值 2016.014446 相差很小；
- 6、程序运行过程中，LCD 上显示我的学号 2016014446；
- 7、按下 SW17，触发外部中断 0，此时学号出现在左端，并逐渐向左移动；
- 8、待上一中断结束后，按下 SW18，触发外部中断 1，此时学号出现在右端，并逐渐向右移动。

【实验结果】

- 1、存储区对应的数值。

Memory 1		Memory 1	
Address: x:0x100		Address: x:0x100	
X:0x000100:	78 29 F0 6E FF 7F E	X:0x000100:	78 29 F0 6E FF 7F E
X:0x000120:	44 FC 00 76 90 08 C	X:0x000120:	4E F0 53 E1 10 00
X:0x000140:	FF FF FF FF FF DF E	X:0x000140:	FF FF FF FF FF DF E

- 2、显示学号。



- 3、学号左移。



4、学号右移



【程序代码】

```

1、led1602.h
// 条件编译指令，如果未定义_1602_则定义_1602
#ifndef _1602_
#define _1602_

// 引入头文件
#include "reg51.h"
#include "intrins.h"

// 定义 LCD 引脚及寄存器地址
sbit LCD1602_RS = P2^5; // 定义 LCD1602_RS 为 P2.5 引脚
sbit LCD1602_RW = P2^6; // 定义 LCD1602_RW 为 P2.6 引脚
sbit LCD1602_E = P2^7; // 定义 LCD1602_E 为 P2.7 引脚
sfr LCD1602_DB = 0x80; // 定义 LCD1602_DB 为 P0 端口
sfr P0M1 = 0x93; // 定义 P0 端口 P0M1 寄存器地址 0x93
sfr P0M0 = 0x94; // 定义 P0 端口 P0M0 寄存器地址 0x94
sfr P2M1 = 0x95; // 定义 P2 端口 P2M1 寄存器地址 0x95
sfr P2M0 = 0x96; // 定义 P2 端口 P2M0 寄存器地址 0x96

// 申明函数
void lcdwait(); // 等待 LCD
空闲函数
void lcdwritecmd(unsigned char cmd); // 写 LCD 命令函
数
void lcdwritedata(unsigned char dat); // 写 LCD 数据函
数
void lcdinit(); // 初 始 化
LCD 函数
void lcdsetcursor(unsigned char x, unsigned char y); // 设定 LCD 光标位置
函数
void lcdshowstr(unsigned char x, unsigned char y, unsigned char *str); // 打印字符串函数

// 条件预编译指令结束
#endif

```

```

2、led1602.c
// 引入头文件
#include "led1602.h"

// 实现函数
/**
 * 函数名: lcdwait()
 * 参数 : 无
 * 功能 : 等待 LCD 空闲

```

```

*/
void lcdwait() {
    LCD1602_DB = 0xFF;          // 读取前，置 P0 端口为 FF
    _nop();                     // 空操作指令，延迟
    _nop();
    _nop();
    _nop();
    LCD1602_RS = 0;             // 将 RS 信号拉低
    LCD1602_RW = 1;             // 将 RW 信号拉高
    LCD1602_E = 1;              // 将 E 信号拉高
    while(LCD1602_DB & 0x80); // 等待标志 BF 为低，即 LCD 空闲
    LCD1602_E = 0;              // 将 E 信号拉低
}

```

```

/**
 * 函数名: lcdwritecmd()
 * 参数   : cmd          控制指令码
 * 功能   : 写 LCD 命令
 */
void lcdwritecmd(unsigned char cmd) {
    lcdwait();                // 等待 LCD 空闲
    _nop();
    _nop();
    _nop();
    _nop();
    LCD1602_RS = 0;
    LCD1602_RW = 0;
    LCD1602_DB = cmd;         // 将控制指令码放到 P0 端口
    LCD1602_E = 1;
    _nop();
    _nop();
    _nop();
    _nop();
    LCD1602_E = 0;
}

```

```

/**
 * 函数名: lcdwritedata()
 * 参数   : dat          数据码
 * 功能   : 写 LCD 数据
 */
void lcdwritedata(unsigned char dat) {
    lcdwait();                // 等待 LCD 空闲
    _nop();

```

```

        _nop_();
        _nop_();
        _nop_();
        LCD1602_RS = 1;
        LCD1602_RW = 0;
        LCD1602_DB = dat; // 将数据码放到 P0 端口
        LCD1602_E = 1;
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        LCD1602_E = 0;
    }

/**
 * 函数名: lcdinit()
 * 参数   : 无
 * 功能   : 初始化 LCD
 */
void lcdinit() {
    lcdwritcmd(0x38); // 发指令 0x38, 2 行模式, 5x8 点阵, 8 位宽度
    lcdwritcmd(0x0c); // 发指令 0x0c, 打开显示, 关闭光标
    lcdwritcmd(0x06); // 发指令 0x06, 文字不移动, 地址自动加 1
    lcdwritcmd(0x01); // 发指令 0x01, 清屏
}

/**
 * 函数名: lcdsetcursor()
 * 参数   : x           在 LCD 上的行数
 *          y           在 LCD 上的列数
 * 功能   : 设定 LCD 光标位置
 */
void lcdsetcursor(unsigned char x, unsigned char y) {
    unsigned char address; // 存储器地址
    if(y==0)               // 在第一行
        address = 0x00 + x;
    else                   // 在第二行
        address = 0x40 + x;
    lcdwritcmd(address | 0x80); // 写存储器地址指令
}

/**
 * 函数名: lcdshowstr()
 * 参数   : x           在 LCD 上的行数

```

```

*          y          在 LCD 上的列数
*          str          需要打印的字符串
* 功能   : 设定 LCD 光标位置
*/
void lcdshowstr(unsigned char x, unsigned char y, unsigned char *str) {
    lcdsetcursor(x, y);        // 设置光标位置
    while((*str) != '\0') {    // 字符串未到结尾
        lcdwritedata(*str);    // 传送字符数据
        str++;                // 字符串地址自增
    }
}

```

3、main.c

```

// 引入头文件
#include "reg51.h"
#include "stdio.h"
#include "led1602.h"

#define ADC_POWER    0x80    // 定义 ADC_POWER  的值为 0x80
#define ADC_FLAG     0x10    // 定义 ADC_FLAG   的值为 0x10
#define ADC_START    0x08    // 定义 ADC_START  的值为 0x08
#define ADC_SPEEDLL  0x00    // 定义 ADC_SPEEDLL 的值为 0x00
#define ADC_SPEEDL   0x20    // 定义 ADC_SPEEDL  的值为 0x20
#define ADC_SPEEDH   0x40    // 定义 ADC_SPEEDH  的值为 0x40
#define ADC_SPEEDHH  0x60    // 定义 ADC_SPEEDHH 的值为 0x60

sfr AUXR      = 0x8E;        // 申明 AUXR 寄存器的地址为 0x8E
sfr ADC_CONTR = 0xBC;        // 申明 ADC_CONTR 寄存器的地址为 0xBC
sfr ADC_RES   = 0xBD;        // 申明 ADC_RES 寄存器的地址为 0xBD
sfr ADC_RESL  = 0xBE;        // 申明 ADC_RESL 寄存器的地址为 0xBE
sfr P1ASF     = 0x9D;        // 申明 P1ASF 寄存器的地址为 0x9D

// 在外部数据区定义变量，用于存储学号
xdata long  num1_at_0x100;    // 定义长整形变量 num1
xdata float num2_at_0x120;    // 定义浮点数变量 num2

unsigned char ch = 4;
bit flag = 1;                // ADC 转换完成标识
unsigned char tstr[50];      // 数据存储缓冲器

/**
* 函数名: delay
* 参数   : n                延迟时间长度，单位毫秒
* 功能   : 延时指定的毫秒时间

```

```

*/
void delay(unsigned int n) {
    unsigned int i, j;
    for(i = n; i > 0; i--) {
        for(j = 114; j > 0; j--);
    }
}

/**
 * 函数名: adc_int
 * 参数   : 无
 * 功能   : ADC 中断响应函数
 */
void adc_int() interrupt 5 {
    unsigned char i = 0;
    ADC_CONTR &= !ADC_FLAG;           // 将 ADC_FLAG 标志清零
    num1 = 2016014446;
    num2 = 2016014446;
    sprintf(tstr + 10, "%lld", num1); // 将数据传送至缓冲区
    flag = 1;                         // ADC 转换完成
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch; // 启动 ADC
}

/**
 * 函数名: move_left
 * 参数   : 无
 * 功能   : 外部中断 0 中断响应函数，实现学号在 LCD 上左移
 */
void move_left() interrupt 0 {
    unsigned char i;
    for(i = 0; i < 11; i++){
        lcdwritcmd(0x01);           // 清屏
        lcdwait();                 // 等待 LCD 空闲
        lcdshowstr(0, 0, tstr+10+i); // 右移数组地址，写字符串
        lcdwait();
        delay(3000);               // 延时等待
    }
}

/**
 * 函数名: move_right
 * 参数   : 无
 * 功能   : 外部中断 1 中断响应函数，实现学号在 LCD 上右移
 */

```



```

void move_right() interrupt 2 {
    unsigned char i;
    for(i = 6; i < 17; i++){
        lcdwritcmd(0x01);           // 清屏
        lcdwait();                 // 等待 LCD 空闲
        lcdshowstr(i,0,tstr+10);    // 右移光标位置，写字符串
        lcdwait();
        delay(3000);               // 延时等待
    }
}

/**
 * 函数名: main
 * 参数   : 无
 * 功能   : 主函数，在 LCD 上打印学号，响应中断
 */
void main() {
    unsigned int i, k;
    P0M0 = 0;          // 通过 P0M0 和 P0M1 寄存器将 P0 口定义为准双向，弱上拉
    P0M1 = 0;
    P2M0 = 0;          // 通过 P2M0 和 P2M1 寄存器将 P2 口定义为准双向，弱上拉
    P2M1 = 0;
    P1ASF = 0xFF; // 将 P1 端口用于 ADC 输入
    ADC_RES = 0; // 将 ADC_RES 寄存器清零
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch; // 配置 ADC_CONTR 寄存
器
    for(i = 0; i < 10000; i++){
        IE = 0xA0; // 允许 ADC 中断
        lcdwait(); // 等待 LCD 空闲
        lcdinit(); // 初始化 LCD
        IT0 = 1; // 开启外部中断 0 和外部中断 1
        EX0 = 1;
        IT1 = 1;
        EX1 = 1;
        EA = 1;
        while(1) {
            if(flag == 1) {
                flag = 0;
                for(k = 0; k < 16; k++) { // 循环打印学号
                    lcdwritcmd(0x01);
                    lcdwait();
                    lcdshowstr(k, 0, tstr+10);
                    if(k > 6)
                        lcdshowstr(0,0,tstr+26-k);
                }
            }
        }
    }
}

```

```
        lcdwait();  
        delay(500);  
    }  
}  
}
```