批次/序号：

桂林电子科技大学 电子工程与自动化学院

智能仪器实验 预习报告

实验名称：		专业：		教师评阅意见：	
学　号：		姓名：			
实验日期：		格式规范性得分：		成绩：	教师签名：

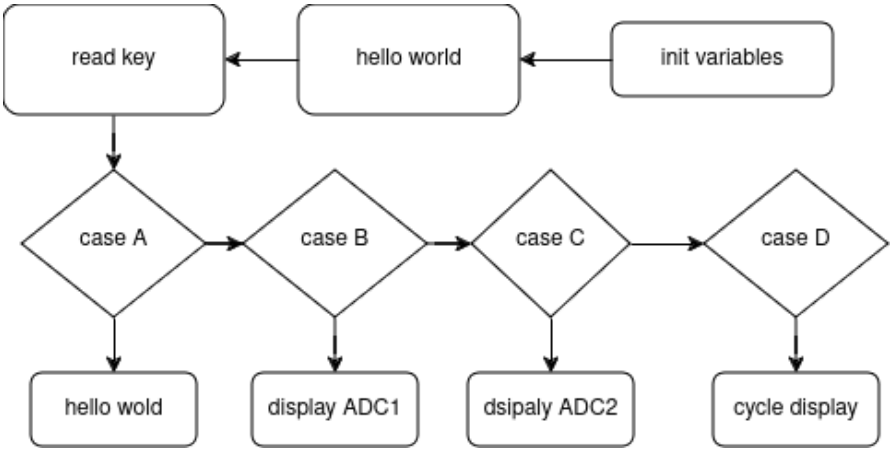
一、实验原理理解和任务分析（20分，得分：　）

1. 如何使用循环扫描按键
2. 如何通过行列值查找按键的数码表
3. 如何使用使用c语言实现液晶显示
4. 如何存储按键的值并进行运行逻辑切换
5. 如何读取ADC数据
6. 如何处理并显示ADC数据

二、设计思路介绍（25分，得分：　）

1. 使用main初始化界面显示hello
2. 使用switch case 切换功能
3. 使用循环加delay函数实现循环显示
4. 使用函数实现read key和display key实现读取和显示的解耦
5. 使用while循环实现key的循环扫描
6. 使用封装adc_dispaly函数方便调用

三、程序流程图介绍（25分，得分：　）



四、主程序介绍（20分，得分： ）

main function 计算总线地址并启动主循环开始扫描按键并显示

```
void main(void)
{
    uchar key_pos = 0;
    uchar Code [] = {0x1c,0x1d,0x1e,0x00,0x01,0x02,0x03};
    uchar index;

    //LCD初始化
    Init();
    Clear();

    //display name
    for(index = 0; index < 2; index++)
    {
        Page_ = 0x03;
        Column = (0x00+index)<<4 + 5;
        Code_ = Code[index];
        WriteCHN16x16(Page_,Column,Code_);
    }
    while(read_key() == 0xff);
    Clear();

    while(1)
    {
        key_pos = read_key();
        if(key_pos != 0xff)
            precess_keyfn(KEY_NUMBER[key_pos]);

    }

}
```

read_key funtion 控制74h374循环输出0并读取键，如果读到了键将行放在高8位列放在低8位

```
// sacnning keys from 0xX000 to 0xX008
unsigned char read_key(void)
{
    unsigned char scan_data = 0x20;//列扫描用IO输出数据
    unsigned char row = 0, col = 0;
    unsigned char key_pos = 0xff;
    //检测是否有按键按下
    XBYTE[ADDR_KEY_WRITE] = 0x00;
    if((XBYTE[ADDR_KEY_READ] & 0x0f) != 0x0f)//有按键按下
    {
        //按键消抖
        Delay5ms();

        //进行列检测
        XBYTE[ADDR_KEY_WRITE] = scan_data;
        while((XBYTE[ADDR_KEY_READ] & 0x0f) != 0x0f)//检测到0x0f时即找到该列
        {
            col ++;
            scan_data  = scan_data >> 1;
            XBYTE[ADDR_KEY_WRITE] = scan_data;
        }

        //进行行检测
        XBYTE[ADDR_KEY_WRITE] = 0x00;
        switch(XBYTE[ADDR_KEY_READ] & 0x0f)
```

```
        {
            case 0x07:row = 0;break;
            case 0x0b:row = 1;break;
            case 0x0d:row = 2;break;
            case 0x0e:row = 3;break;
            default:return 0xff;

        }
        //合成按键位置
        key_pos = row*6 + col;
        while((XBYTE[ADDR_KEY_READ] & 0x0f) != 0x0f);
        return key_pos;
    }
    else
        return 0xff;
}
```

display_bus function 控制液晶输出key

```
// 中文显示子程序
void WriteCHN16x16()
{
  unsigned char i,j,k;

  i = 0;
  j = 0;
  while(j<2) {
    Command = ((Page_ + j) & 0x03) | 0xb8;    // 设置页地址
    WriteCommandE1();
    WriteCommandE2();
    k = Column;                    // 列地址值
    while(k < Column + 16){
      if (k < PD1) {               // 为左半屏显示区域(E1)
        Command = k;
        WriteCommandE1();          // 设置列地址值
        LCDData = CCTAB[Code_][i]; // 取汉字字模数据
        WriteDataE1();             // 写字模数据
      } else{                      // 为右半屏显示区域(E2)
        Command = k-PD1;
        WriteCommandE2();          // 设置列地址值
        LCDData = CCTAB[Code_][i]; // 取汉字字模数据
        WriteDataE2();             // 写字模数据
      };

      i++;
      if( ++k >= PD1 * 2) break;  // 列地址是否超出显示范围
    } ;
    j++;
  };
}

//英文显示子程序
void WriteEN8x8(void)
{
      unsigned char i,j,k;

        i = 0;
        j = 0;

        Command = ((Page_ + j) & 0x03) | 0xb8;    // 设置页地址
    WriteCommandE1();
    WriteCommandE2();
    k = Column;                    // 列地址值
```

```
      if (k < PD1) {               // 为左半屏显示区域(E1)
        Command = k;
        WriteCommandE1();          // 设置列地址值
        LCDData = CCTAB[Code_][i]; // 取汉字字模数据
        WriteDataE1();             // 写字模数据
      } else{                      // 为右半屏显示区域(E2)
        Command = k-PD1;
        WriteCommandE2();          // 设置列地址值
        LCDData = CCTAB[Code_][i]; // 取汉字字模数据
        WriteDataE2();             // 写字模数据
      };

      i++;
//      if( ++k >= PD1 * 2)  // 列地址是否超出显示范围

}```
```

adc funtion 读adc数据和显示adc数据

```c
uchar ADC_CONVERT(sbit START_ALE,sbit EOC,sbit OE)         //ADC转换开始函数
{
    uchar ADC_Date=0;
    START_ALE=1;                   //一个正脉冲
    START_ALE=0;
    while(!EOC);
    OE=1;
    ADC_Date=P0;
    return ADC_Date;
}

void display_ADC_Date(uchar ADC_Date)
{
    uint i=0;
    uchar display_bit ;

    for(i=0;i<3;i++){
        display_bit=ADC_Date%10;
        WriteCHN16x16(Page_=0x03,Column=(0x01+i)<<4,Code_=display_bit);                //显示ADC转换
值
        ADC_Date=ADC_Date*10;                  //清除显示
    }
}
```

key function 功能切换

```
void keyfn(unsigned char key, uchar ADC_Date)
{
    unsigned char number_index = len_name + len_charcter + len_string;
    unsigned char charcter_index = len_name + len_string;
    unsigned int charcter;
    unsigned int number;
    sbit ADDA=P1^0;             //   地址A
    sbit ADDB=P1^1;             //   地址B
    sbit ADDC=P1^2;             //   地址C
    sbit OE=P1^5;               //   输出使能
    sbit EOC=P1^6;              //    转换标志位
    sbit START_ALE=P1^4;        //   地址装载，转换启动脚

    uint col;
```

```
    switch(key){
    case 'A': { // =
        display_name();
        }
        break;
    case 'B':{

        ADDA=0;
        ADDB=0;
        ADDC=0;
        ADC_Date = ADC_CONVERT(START_ALE,EOC,OE);
        ADC_Date = ADC_Date/51;
        display_ADC_Date(ADC_Date);
        }
        break;
    case 'C':{
        ADDA=0;
        ADDB=0;
        ADDC=1;
        ADC_Date = ADC_CONVERT(START_ALE,EOC,OE);
        ADC_Date = ADC_Date/51;
        display_ADC_Date(ADC_Date);
        }
        break;
    case 'D':{
        ADDA=0;
        ADDB=0;
        ADDC=1;
        ADC_Date = ADC_CONVERT(START_ALE,EOC,OE);
        ADC_Date = ADC_Date/51;
        display_ADC_Date(ADC_Date);
        void delay_ms(3000);                    // 延时程序
        ADDA=0;
        ADDB=0;
        ADDC=1;
        ADC_Date = ADC_CONVERT(START_ALE,EOC,OE);
        ADC_Date = ADC_Date/51;
        display_ADC_Date(ADC_Date);
        }
    default:{
        }
        break;
    }

}
```