项目三

STC15F2K60S2单片机的输入/输出控制

本项目应用C语言编程，通过STC15F2K60S2单片机输入/输出端口从片外读取数据，并将处理后的数据传递到片外。

知识点：

- ◇ 设置STC15F2K60S2单片机I/O端口为输入方式的方法；
- ◇ C语言的赋值语句；
- ◇ while、if、for、switch/case语句；
- ◇ LED数码管显示的基本原理；
- ◇ LED数码管的静态显示与动态显示。

技能点：

- ◇ 熟练掌握while、if、for、switch/case语句的应用编程；
- ◇ 掌握LED数码管显示接口的硬件设计与软件设计；
- ◇ 掌握各任务程序的调试方法(Keil C仿真、Proteus仿真、在线调试)。

教学法：

针对任务实例，应用while、if、for、switch/case语句对比编程，理解各语句的编程特点与差异，熟练掌握while、if、for、switch/case语句的编程技巧。

任务1　STC15F2K60S2单片机的基本输入/输出

任务说明

本任务一是介绍单片机如何通过I/O输出端口与片外设备进行数据交换，二是利用C语言对I/O端口的应用编程。

相关知识

一、赋值运算符与表达式

1. 简单赋值运算

在C语言中,最常见的赋值运算符为“=”。利用赋值运算符将一个变量与一个表达式连接起来的式子称为赋值表达式。在赋值表达式的后面加一个“;”便构成了语句。例如:

```
y = 6 ; 将 6 赋值给变量 y
y = x ; 变量 x 的值赋给变量 y
```

2. 复合赋值运算符

在赋值运算符“=”的前面加上其他运算符,就构成了复合赋值运算符。复合赋值运算符首先对变量进行某种运算,然后将运算结果赋值给该变量。复合运算的一般形式为

```
变量 复合赋值运算符 表达式
```

C语言中有以下几种复合赋值运算符:

- +=为加法赋值运算符。例如,“x+=3;”等效于“x=x+3;”。
- -=为减法赋值运算符。例如,“x-=3;”等效于“x=x-3;”。
- *=为乘法赋值运算符。例如,“x*=3;”等效于“x=x*3;”。
- /=为除法赋值运算符。例如,“x/=3;”等效于“x=x/3;”。
- %=为取模(余)赋值运算符。例如,“x%=3;”等效于“x=x%3;”。
- >>=为右移位赋值运算符。例如,“x>>=3;”等效于“x=x>>3;”。
- <<=为左移位赋值运算符。例如,“x<<=3;”等效于“x=x<<3;”。
- &=为逻辑与赋值运算符。例如,“a&=b;”等效于“a=a&b;”。
- |=为逻辑或赋值运算符。例如,“a|=b;”等效于“a=a|b;”。
- ^=为逻辑异或赋值运算符。例如,“a^=b;”等效于“a=a^b;”。
- ~=为逻辑非赋值运算符。例如,“a~=b;”等效于“a=~b;”。

二、函数的定义

函数是C语言程序的基本模块。所有的函数在定义时是相互独立的,它们之间是平衡关系,不能在一个函数中定义另外一个函数,即不能嵌套定义。函数之间可以相互调用,但不能调用主函数。

C语言系统提供功能强大、资源丰富的标准函数库。在进行程序设计时,应善于利用这些资源,以提高效率,节省开发时间。

1. 函数定义的一般形式

函数定义的一般形式有两种。

1) 第1种形式

```
函数类型标识符  函数名(形式参数)
形式参数类型说明列表
```

```
{
局部变量定义
函数体语句
}
```

2）第 2 种形式

```
函数类型标识符　函数名(形式参数类型说明列表)
{
局部变量定义
函数体语句
}
```

人们一般习惯于使用第 2 种形式。

3）说明

函数类型标识符：说明了函数返回值的类型。当“函数类型标识符”缺省时，默认为整型。

函数名：是程序设计人员自己设计的名字。

形式参数类型说明列表：是主调用函数与被调用函数之间传递数据的形式参数。若定义是无参函数，形式参数类型说明列表用“void”来注明。

局部变量定义：是对在函数内部使用的局部变量进行定义。

函数体语句：是为完成该函数的特定功能而设置的各种语句。

2. 函数的参数和函数的返回值

1）函数的参数

C 语言采用函数之间的参数传递方式，使一个函数能对不同变量进行处理，从而提高函数的通用性与灵活性。在函数调用时，通过主调函数的实际参数与被调函数的形式参数之间传递数据来实现函数间参数的传递。

2）函数的返回

在被调用函数最后，通过 return 语句返回函数的值给主调函数。其格式为：

```
return (表达式);
```

对于不需要有返回值的函数，可以将该函数类型定义为“void”类型。为了使程序减少出错，保证函数的正确使用，凡是不要求有返回值的函数，都应将其定义为“void”类型。

3）函数的分类

从函数定义的形式看，分为无参数函数、有参数函数和空函数三种。

（1）无参数函数。

无参数函数在调用时无参数，主调函数并不将数据传送给被调用函数。无参数函数可以返回或不返回函数值，一般不返回值的居多。

（2）有参数函数。

调用有参数函数时，在主调函数与被调函数之间有参数传递。主调函数可以将数据传送给被调函数使用，被调函数中的数据也可以返回供主调函数使用。

（3）空函数。

如果定义函数时只给出一对花括号“{}”，不给出局部变量和函数体语句，即函数体内部

是空的，则该函数称为空函数。

3. 函数的声明与调用

C 语言程序中的函数是可以互相调用的，但不能调用主函数。所谓函数调用就是在一个函数体中引用另外一个已经定义的函数，前者称为主调用函数，后者称为被调用函数。

1）调用函数的一般形式

```
函数名(实际参数列表)
```

函数名：指出被调用的函数。

实际参数列表：实际参数的作用是将它的值传递给被调用函数中的形式参数，可以包含多个实际参数，各个参数之间用逗号分开。需要注意的是，函数调用中的实际参数与函数定义中的形式参数必须在个数、类型和顺序上严格保持一致，以便将实际参数的值正确传送给形式参数。如果调用的是无参函数，可以没有实际参数列表，但圆括号不能省略。

2）函数调用的方式

在实际编程中，可采用三种方式完成函数的调用。

(1) 函数语句调用。

在主调函数中，将函数调用作为一条语句，例如：

```
Funl();
```

这是无参调用，它不需要被调函数返还一个确定的值。

(2) 函数表达式调用。

在主调函数中，将函数调用作为一个运算对象直接出现在表达式中。这种表达式称为函数表达式，例如：

```
d = power(x, n) + power(y, m) ;
```

它包括两个函数调用，每个函数调用都有一个返回值，将两个返回值相加的结果赋值给变量 d。因此，这种函数调用方式要求被调用函数返回一个确定的值。

(3) 作为函数参数调用。

在主调函数中，将函数调用作为另一个函数调用的实际参数，例如：

```
m = max( a, max(b, c)) ;
```

max(b, c)是一次函数调用，它的返回值作为函数 max 另一次调用的实参。这种在调用一个函数中又调用一个函数的方式，称为嵌套函数调用。

3）调用函数必须满足“先声明、后调用”的原则

(1) 调用函数与被调用函数位于同一个程序文件中。

如果被调用函数是在调用函数前面定义的，可直接调用；如果被调用函数是在调用函数后面定义的，需要在调用前声明被调用函数。例如：

```
#include <REG51.H>
#define x P1
void delay(void);                    //语句 3,声明延时子函数
/--------- LED 灯驱动函数 ---------/
void light(void)
```

```
{
    x = ~x;
}

/* -------- 主函数 ---------- */
void main(void)
{
    while(1)
    {
        light();                //light()在主函数前面定义,因此可直接调用
        delay();                //delay()在主函数后面定义,调用前必须先声明,见语句 3
    }
}
/----------------------/
void delay(void)
{
    unsigned int i,j;
    for(i = 0;i<500;i++)
    {
        for(j = 0;j<121;j++)
        {;}
    }
}
```

(2) 函数的连接。

当程序中的子函数与主函数不在同一个程序文件中时,要通过连接的方法实现有效的调用。一般有两种方法,即外部声明与文件包含。

① 外部声明。

设 delay()和 light()两个子函数与调用主函数不在一个程序文件中,则当主函数要调用 delay()和 light()时,可在调用前做外部声明。例如:

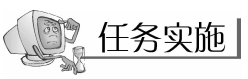
```
#include <REG51.H>
extern void delay(void);        /* 声明该函数在其他文件中 */
extern void light(void);        /* 声明该函数在其他文件中 */
/----------------------/
void main(void)
{
    while(1)
    {
        light();
        delay();

    }
}
```

② 文件包含。

当主函数需要调用分属在其他程序文件中的子函数时,也可用包含语句将含有该子函数的程序文件包含进来。包含的意思可以理解为将包含文件的内容放在包含语句位置处。

设 delay()和 light()两个子函数在 test.c 程序文件中定义,主函数要调用 delay()和 light()时,可用包含语句将 test.c 程序文件包含进来,类似包含头文件的意思。例如:

```
#include <reg51.h>
#include "test.c"
void main(void)
{
    while(1)
    {
        light();
        delay();
    }
}
```



任务实施

一、程序功能

P1 口的输出跟随 P2 输入端数据,并用 LED 显示 P2 口的输入数据。LED 灯亮表示高电平,LED 灯灭表示低电平。

二、硬件设计

用 P2 作为输入口,P1 作为输出口,驱动 8 只 LED 灯,电路如图 3-1-1 所示。

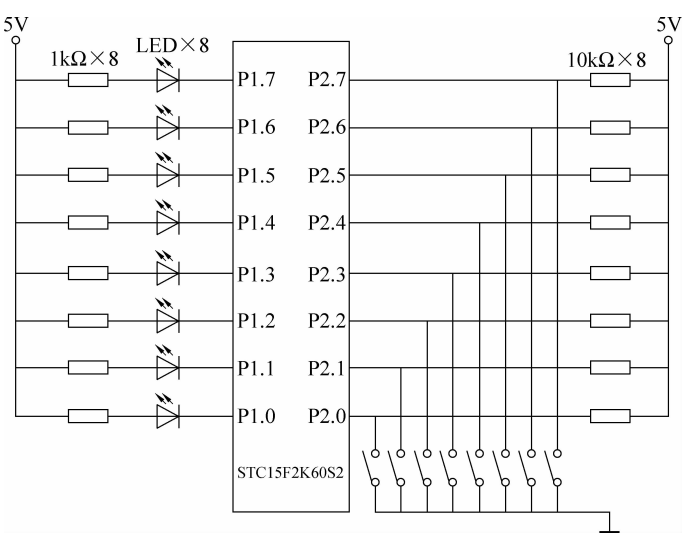


图 3-1-1　基本输入/输出控制

三、程序设计

```
#include <reg51.h>
#define y P1
#define x P2
void main(void)
{
    x = 0xff;                    //置 P2 口为输入模式
```

```
    while(1)
    {
        y = ~x;                    //因 LED 灯是低电平驱动,故 P1 口的输出取 P2 输入的非信号
    }
}
```

四、调试

(1) 用 Proteus 绘制电路原理图或在实验板上连接电路。

(2) 用 Keil C 编辑、编译程序,生成机器代码文件。

(3) 联机调试。改变输入端开关的状态,观察与记录 LED 灯的状态。

思考与提高

用 P1 口的低 4 位作为输入,接 4 只开关;用 P1 口的高 4 位作为输出,控制 4 只 LED 灯,用低电平驱动。画出电路图,编写程序并调试。

习题

1. STC15F2K60S2 单片机的 I/O 端口用作输入时,应注意什么?

2. 一般情况下,驱动 LED 灯应加限流电阻。请问如何计算限流电阻?

3. 数据类型隐式转换的优先顺序是什么?

4. 位运算符的优先顺序是什么?

5. 试说明下列语句的含义。

(1)
```
unsigned char x;
unsigned char y;
k = (bit)(x + y);
```

(2)
```
#define uchar unsigned char
uchar a;
uchar b;
uchar min;
```

(3)
```
#define uchar unsigned char
uchar tmp;
P1 = 0xff;
temp = P1;
temp &= 0x0f;
```

任务 2　STC15F2K60S2 单片机的逻辑控制

任务说明

逻辑控制能力是单片机的核心能力之一,单片机突出控制的具体体现,对于

STC15F2K60S2 单片机也是如此。本任务主要学习应用 C 语言的分支语句、开关语句、循环语句等对单片机逻辑控制的应用编程。这里要强调的是,C 语言的分支语句、开关语句、循环语句是单片机 C 语言编程中最重要、最常见的语句。

相关知识

一、运算符与表达式

C 语言对数据有很强的表达能力,具有丰富的运算符,利用这些运算符可以组成各种表达式及语句。运算符是完成某种特定运算的符号;表达式是由运算符及运算对象所组成的具有特定含义的式子。C 语言是一种表达式语言,在任意一个表达式的后面加一个分号“;”就构成一条表达式语句。

按照运算符在表达式中所起的作用,分为算术运算符、关系运算符、逻辑运算符、赋值运算符、增量与减量运算符、逗号运算符、条件运算符、位运算符、指针和地址运算符、强制类型转换运算符和 sizeof 运算符等。

1. 关系运算符与表达式

C 语言有以下关系运算符:

>为大于;

<为小于;

>=为大于或等于;

<=为小于或等于;

==为测试等于;

!=为测试不等于。

>、<、>=、<= 4 种关系运算符具有相同的优先级,==、!= 2 种运算符也具有相同的优先等级,但前 4 种的优先级高于后 2 种。

关系运算符用来判断某个条件是否满足,其结果只有“真”和“假”2 种值。当所指定的条件满足时,结果为“1”;条件不满足时,结果为“0”。“1”表示“真”,“0”表示“假”。

2. 逗号运算符与表达式

逗号运算符可以将两个或多个表达式连接起来,构成逗号表达式。逗号表达式的一般形式为:

```
表达式 1,表达式 2,表达式 3,…,表达式 n
```

逗号表达式的运算过程为:先算表达式 1,再算表达式 2,…,依次算到表达式 n 为止。

3. 条件运算符与表达式

条件运算符要求有 3 个运算对象,用它将 3 个表达式连接构成 1 个条件表达式。条件表达式的一般形式为

```
表达式 1?表达式 2: 表达式 3
```

其运算过程为:首先计算表达式 1,根据表达式 1 的结果来判断。当表达式 1 的结果为

“真”(非“0”值)时,将表达式 2 的值作为整个表达式的值;当表达式 1 的结果为“假”(“0”值)时,将表达式 3 的值作为整个表达式的值。

二、分支语句与分支选择结构

1. 表达式语句与复合语句

1) 表达式语句

C 语言提供了十分丰富的程序控制语句,表达式语句是最基本的一种语句。在表达式的后边加一个分号“;”就构成表达式语句。例如:

```
x = 10 ;
y = 100 ;
pjz = (x + y)/2 ;
```

2) 空语句

仅有一个分号“;”构成的语句,称为空语句。空语句是表达式语句的一个特例。空语句通常有两种用法。

(1) 在程序中为有关语句提供标号,例如:

```
loop: ;
…
if (y == 6) goto loop ;
```

(2) 在用 while 语句构成的循环语句后面加一个分号“;”,形成一个空语句循环。例如:

```
{
    char f ;
    while (!RI) ;                //循环检测 RI 标志,直至为“1”为止
    f = RI ;
    RI = 0 ;
    return f ;
}
```

这段程序是读取 8051 单片机串口数据的函数。其中,“while (!RI) ;”用来等待单片机串行接收结束。

3) 复合语句

复合语句是由若干条语句组合而成的一种语句,它是用一个花括号“{}”将若干条语句组合而成的一种功能块。复合语句不需要以分号“;”结束,但其内部各条单语句必须以分号“;”结束。

```
{
局部变量定义 ;
语句 1 ;
语句 2 ;
…
语句 n ;

}
```

在执行复合语句时,其中的各条单语句依次执行。整个复合语句在语法上等价于一条语句。复合语句允许嵌套,即在复合语句中可以包含别的复合语句。实际上,函数体就是一种复合语句。在复合语句中定义的变量为局部变量,仅在当前复合语句中有效。

2. 条件分支语句

条件语句又称为分支语句,它由关键字 if 构成,有 3 种格式。

1) 格式 1

```
if(条件表达式)语句
```

若条件表达式的结果为“真”(非“0”值),就执行后面的语句;若条件表达式的结果为假(“0”值),就不执行后面的语句。这里的语句也可以是复合语句。

2) 格式 2

```
if(条件表达式)语句 1
else 语句 2
```

若条件表达式的结果为“真”(非“0”值),就执行后面的语句 1;若条件表达式的结果为“假”(“0”值),就执行语句 2。这里的语句 1 和语句 2 均可以是复合语句。

3) 格式 3

```
if(条件表达式 1)语句 1
else if(条件表达式 2)语句 2
    else if(条件表达式 3)语句 3
    …
        else if(条件表达式 n)语句 n
            else 语句 n + 1
```

这种条件语句常用来实现多方向条件分支,它是 if-else 语句嵌套而成的。在这种结构中,else 总是与临近的 if 相配对。

3. 开关语句

switch/case 开关语句是一种多分支选择语句,是用来实现多方向条件分支的语句。

1) switch/case 开关语句的格式

```
switch(表达式)
{
    case 常量表达式 1: {语句 1}break;
    case 常量表达式 2: {语句 2}break;
            ⋮              ⋮
    case 常量表达式 n: {语句 n}break;
    default:           {语句 n + 1}break;
}
```

2) 开关语句说明

(1) 当 switch 后面表达式的值与某一“case”后面的常量表达式的值相等时,就执行该“case”后面的语句。遇到 break 语句时,就退出 switch 语句。