手把手教你学单片机的C语言程序设计(十六)

◆吕超亚

16*2点阵字符液晶模块的C语言驱动

16*2点阵字符液晶模块是由点阵字符液晶显示器件和专用的行、列驱动器、控制器及必要的连接件、结构件装配而成,可以显示数字和英文字符。这种点阵字符模块本身具有字符发生器,显示容量大,功能丰富。

液晶点阵字符模块的点阵排列是由5x7或5x8,5x11的一组组像素点阵排列组成的。每组为1位,每位间有一点的间隔,每行间也有一行的间隔,所以不能显示图形。

一般在模块控制、驱动器内具有已固化好192个字符字模的字符库CGROM,还具有让用户自定义建立专用字符的随机存储器CGRAM,允许用户建立8个5x8点阵的字符。这种模块适用于M6800系列计算机接口,接口数据传输可用8位或4位数据传输方式。点阵字符模块具有丰富的显示功能,其驱动芯片大多为HD44780或兼容电路。

液晶显示器和其它显示器相比,具有以下突出的优点:

1.低电压、场致驱动。

2.微功耗,仅1μW/cm2。

3.平板显示,体积小而薄。

4.与集成电路匹配方便、简单。

5.被动显示,不怕光冲刷。

6.可彩色、黑白显示,效果逼真。

7.显示面积可大可小,目前世界上最大的液晶电视尺寸已超过50英寸。

8.易于大批量生产。

9.随着工艺的提高,成品率还会进一步提高,成本也会进一步下降。

液晶显示器的缺点:

1.视角较小。

2.显示质量不算最高。

3.响应速度较慢,对快速移动图像可能有一些拖尾,目前正在克服中。

本刊2005年2月号~7月号连载了《字符型LCD显示器原理及显示技术》,对16*2点阵字符液晶模块的内部结构、引脚排列、工作原理、LCD控制器的指令等已作了详细介绍,读者朋友可参考,这里主要介绍单片机C51程序对它的驱动。

要实现对LCM的高效控制,必须按照模块设计方式,建立起相关的子程序,下面先详细介绍用C51设计的各功能子程序。

```
//**** 延时K*1mS子程序,12.000MHz****
void delay (unsigned int k) // 函数名为delay的延时子函数。定义k为无符号整型变量
{                          //delay 子函数开始
unsigned int i,j;          // 定义i、j为无符号整型变量
for(i=0;i<k;i++){          //for 循环语句开始
for(j=0;j<121;j++)         // for 循环语句开始
{;}}                       // 两个 for 循环体
}                          //delay 子函数结束
//******* 写命令到LCM子程序 ***********
void WriteCommandLCM (unsigned char WCLCM,BusyC) /* 函数名为WriteCommandLCM的写指令到LCM子函数。定义WCLCM、BusyC为无符号字符型变量 */
{
//WriteCommandLCM 子函数开始
if(BusyC)ReadStatusLCM();/* 若BusyC为"1",则调用ReadStatusLCM子函数进行忙检测 */
P1=WCLCM;  // 将变量WCLCM中的指令传送至P1口
LCM_RS=0;  // 选中指令寄存器
LCM_RW=0;  // 写模式
LCM_EN=0;  // 置LCM_EN端为低电平
LCM_EN=0;  // 置LCM_EN端为低电平,延时一会儿
LCM_EN=1;  // 置LCM_EN端为高电平,写入使能
}          // WriteCommandLCM子函数结束
//****** 写数据到LCM子程序 *********
void WriteDataLCM (unsigned char WDLCM) /* 函数名为WriteDataLCM的写数据到LCM子函数。定义WDLCM为无符号字符型变量 */
{          // WriteDataLCM 子函数开始
ReadStatusLCM();// 调用ReadStatusLCM子函数检测忙信号
P1=WDLCM;  // 将变量WDLCM中数据传送至P1口
LCM_RS=1;  // 选中数据寄存器
LCM_RW=0;  // 写模式
LCM_EN=0;  // 置LCM_EN端为低电平
LCM_EN=0;  // 置LCM_EN端为低电平,延时一会儿
LCM_EN=1;  // 置LCM_EN端为高电平,写入使能
}          // WriteDataLCM子函数结束
//***** 从LCM读数据到MCU子程序 ******
unsigned char ReadDataLCM (void) // 函数名为ReadDataLCM的读数据到
//MCU子函数
{          // ReadDataLCM子函数开始
LCM_RS=1;  // 选中数据寄存器
LCM_RW=1;  // 读模式
LCM_EN=0;  // 置LCM_EN端为低电平
LCM_EN=0;  // 置LCM_EN端为低电平,延时一会儿
LCM_EN=1;  // 置LCM_EN端为高电平,读使能
return(P1);  // 返回P1口的内容
}          // ReadDataLCM子函数结束
//*** 从LCM读状态到MCU子程序 ***
unsigned char ReadStatusLCM(void) // 函数名为ReadStatusLCM的读状态到
//MCU子函数
{          //ReadStatusLCM子函数开始
P1=0xFF;   // 置P1口为全1
LCM_RS=0;  // 选中指令寄存器
LCM_RW=1;  // 读模式
LCM_EN=0;  // 置LCM_EN端为低电平
LCM_EN=0;  // 置LCM_EN端为低电平,延时一会儿
LCM_EN=1;  // 置LCM_EN端为高电平,读使能
while(P1&Busy);/* 检测忙信号。当P1口内容与Busy(80H)相与后不为零时,程序原地踏步 */
return(P1);  // 返回P1口的内容
```

```
}          // ReadStatusLCM 子函数结束
//******LCM 初始化 ******
void InitLcd(void)   // 函数名为 InitLcd 的 LCM 初始化子函数
{                    // InitLcd 函数开始
WriteCommandLCM(0x38,1);//8 位数据传送,2 行显示,5*7 字形,检测忙信号
WriteCommandLCM (0x08,1);// 关闭显示,检测忙信号
WriteCommandLCM(0x01,1);// 清屏,检测忙信号
WriteCommandLCM(0x06,1);// 显示光标右移设置,检测忙信号
WriteCommandLCM (0x0c,1);// 显示屏打开,光标不显示、不闪烁,检测忙信号
}
/*-- 显示指定座标一个字符的子函数 --*/
/* 显示指定座标一个字符(X=0~15,Y=0~1) 的子函数,函数名为 DisplayOneChar,定义 X、Y、DData 为无符号字符型变量 */
void DisplayOneChar (unsigned char X,unsigned char Y,unsigned char DData)
{          // DisplayOneChar 子函数开始
Y&=1;    //Y 的变化范围 0~1
X&=15;   // X 的变化范围 0~15
if(Y)Xl=0x40;   // 若 Y 为 1 (显示第二行),地址码 +0x40
Xl=0x80; // 指令码为地址码 +0x80
WriteCommandLCM(X,0);// 将指令 X 写入 LCM,忽略忙信号检测
WriteDataLCM(DData);  // 再将数据 Ddata 写入 LCM
}          // DisplayOneChar 子函数结束
/*-- 显示指定座标一串字符的子函数 --*/
/* 显示指定座标的一串字符(X=0~15,Y=0~1)子函数,函数名为 DisplayListChar,定义 X、Y 为无符号字符型变量,DData 为指向 code 区的无符号字符型指针变量 */
void DisplayListChar (unsigned char X,unsigned char Y,unsigned char code *DData)
{          // DisplayListChar 子函数开始
unsigned char ListLength=0;// 定义 ListLength 为无符号字符型变量,并赋初值为 0
Y&=0x1;        // Y 的变化范围 0~1
X&=0xF;        // X 的变化范围 0~15
while(X<=15)  //X<=15 时进入 while 语句循环
{              //while 语句开始
DisplayOneChar (X,Y,DData[ListLength]);// 显示单个字符
ListLength++; // 数组指针递增
X++;          //X 轴座标递增
}              //while 语句结束
}          //DisplayListChar 子函数结束
```

下面我们做两个实验,看液晶的显示效果。

实验一

在 LED/16*2 字符液晶试验板上实现 LCD 演示程序,第一行显示 -This is a LCD-!,第二行显示 -Design by ZXH-!。通电后两行字符从右向左移出到显示屏,然后向右退出显示屏。接着两行字符显示于屏幕并闪烁 5 次。最后两行字符从右向左滚动显示,无限循环。

在我的文档中建立一个文件目录(cs44),然后建立 cs44.uv2 的工程项目,最后建立源程序文件(cs44.c)。

输入下面的程序:

```
#include <REG51.H>   // 序号(以下同):1
sbit LCM_RS=P3^3;              //2
sbit LCM_RW=P3^4;              //3
sbit LCM_EN=P3^5;              //4
#define Busy 0x80              //5
unsigned char ReadStatusLCM(void);//6
unsigned char code str0 []= {"-This is a LCD-!"};   //7
unsigned char code str1[]={"-Design by ZXH-!"}      ;//8
unsigned char code str2[]={"      "};//9
//******** 延时 K*1mS**************10
void delay(unsigned int k)         //11
{                                  //12
unsigned int i,j;                  //13
for(i=0;i<k;i++){                  //14
for(j=0;j<121;j++)                 //15
{;}}                               //16
}                                  //17
void WriteCommandLCM (unsigned char WCLCM,BusyC)//18
{                                  //19
if(BusyC)ReadStatusLCM();          //20
P1=WCLCM;                          //21
LCM_RS=0;                          //22
LCM_RW=0;                          //23
LCM_EN=0;                          //24
LCM_EN=0;                          //25
LCM_EN=1;                          //26
}                                  //27
//********** 写数据 *****************28
voidWriteDataLCM(unsignedcharWDLCM)
                                   //29
{                                  //30
ReadStatusLCM();                   //31
P1=WDLCM;                          //32
LCM_RS=1;                          //33
LCM_RW=0;                          //34
LCM_EN=0;                          //35
LCM_EN=0;                          //36
LCM_EN=1;                          //37
}                                  //38
//********** 读数据 *****************39
unsigned char ReadDataLCM(void)//40
{                                  //41
LCM_RS=1;                          //42
LCM_RW=1;                          //43
LCM_EN=0;                          //44
LCM_EN=0;                          //45
LCM_EN=1;                          //46
return(P1);                        //47
}                                  //48
//********** 读状态 *****************49
unsigned char ReadStatusLCM(void)//50
{                                  //51
P1=0xFF;                           //52
LCM_RS=0;                          //53
LCM_RW=1;                          //54
LCM_EN=0;                          //55
LCM_EN=0;                          //56
LCM_EN=1;                          //57
while(P1&Busy);                    //58
return(P1);                        //59
}                                  //60
//****** 初始化 *********************61
void InitLcd()                     //62
{                                  //63
WriteCommandLCM(0x38,1);           //64
WriteCommandLCM(0x08,1);           //65
WriteCommandLCM(0x01,1);           //66
WriteCommandLCM(0x06,1);           //67
WriteCommandLCM(0x0c,1);           //68
}                                  //69
//*** 显示指定座标的一个字符,X=0~15,Y=0~1***70
void DisplayOneChar (unsigned char X,unsigned char Y,unsigned char DData)//71
{                                  //72
Y&=1;                              //73
X&=15;                             //74
if(Y)Xl=0x40;                      //75
Xl=0x80;                           //76
WriteCommandLCM(X,0);              //77
WriteDataLCM(DData);               //78
}                                  //79
//*** 显示指定座标的字符串,X=0~15,Y=0~1***80
void DisplayListChar (unsigned char X,unsigned char Y,unsigned char code *DData)
                                   //81
{                                  //82
```

```
unsigned char ListLength=0;          //83
ListLength=0;                        //84
Y&=0x1;                              //85
X&=0xF;                              //86
while(X<=15)                         //87
{                                    //88
DisplayOneChar(X,Y,DData[ListLength]);
                                     //89
ListLength++;                        //90
X++;                                 //91
}                                    //92
}                                    //93
//******** 主函数 **********************94
void main(void)                      //95
{                                    //96
char i,m;                            //97
delay(500);                          //98
InitLcd();                           99
/****** 从右移到显示屏 ********100******/
for(i=15;i>=0;i--)                   //101
{                                    //102
WriteCommandLCM(0x01,1);             //103
DisplayOneChar(i,0,0x20);            //104
DisplayListChar(i,0,str0);           //105
DisplayListChar(i,1,str1);           //106
delay(200);                          //107
}                                    //108
delay(2800);                         //109
/******** 向右退出显示屏 ********110*****/
for(i=0;i<16;i++)                    //111
{                                    //112
WriteCommandLCM(0x01,1);             //113
DisplayOneChar(i,0,0x20);            //114
DisplayListChar(i,0,str0);           //115
DisplayListChar(i,1,str1);           //116
delay(200);                          //117
}                                    //118
WriteCommandLCM(0x01,1);             //119
delay(3000);                         //120
/******** 闪烁 5 次 *********121*****/
for(i=0;i<10;i++)                    //122
{                                    //123
WriteCommandLCM(0x01,1);             //124
delay(500);                          //125
DisplayListChar(0,0,str0);           //126
DisplayListChar(0,1,str1);           //127
delay(500);                          //128
i++;                                 //129
}                                    //130
delay(3000);                         //131
/************************132***********/
while(1)                             //133
{                                    //134
/******** 从右移到显示屏 *****135****/
for(i=15;i>=0;i--)                   //136
{                                    //137
WriteCommandLCM(0x01,1);             //138
DisplayOneChar(i,0,0x20);            //139
DisplayListChar(i,0,str0);           //140
DisplayListChar(i,1,str1);           //141
delay(200);                          //142
}                                    //143
/****** 向左退出显示屏 *****144*******/
for(i=1;i<16;i++)                    //145
{                                    //146
m=16-i;                              //147
WriteCommandLCM(0x01,1);             //148
DisplayOneChar(0,0,0x20);            //149
DisplayListChar(0,0,&str0[i]);       //150
DisplayListChar(0,1,&str1[i]);       //151
DisplayListChar(m,0,str2);           //152
DisplayListChar(m,1,str2);           //153
delay(200);                          //154
}                                    //155
WriteCommandLCM(0x01,1);             //156
delay(200);                          //157
}                                    //158
}                                    //159
```

编译通过后，将生成的 cs44.hex 文件烧录到 89S51 芯片中，将芯片插入到 LED/16*2 字符液晶试验板上，连接好 16*2 字符液晶模组，试验板上接通 9V 电源，可以看到，液晶上显示的内容正是我们设计要求的。

我们对程序进行分析。

序号 1（程序解释，以下同）：包含头文件 REG51.H。
序号 2~4：引脚定义。
序号 5：宏定义。
序号 6：函数声明。
序号 7~9：待显示字符串。
序号 10：程序分隔。
序号 11~17：延时 K*1mS 子函数。
序号 18~27：写命令到 LCM 子程序。
序号 28：程序分隔。
序号 29~38：写数据到 LCM 子程序。
序号 39：程序分隔。
序号 40~48：从 LCM 读数据到 MCU 子程序。
序号 49：程序分隔。
序号 50~60：从 LCM 读状态到 MCU 子程序。
序号 61：程序分隔。
序号 62~69：LCM 初始化。
序号 70：程序分隔。
序号 71~79：显示指定座标一个字符的子函数。
序号 80：程序分隔。
序号 81~93：显示指定座标一串字符的子函数。
序号 94：程序分隔。
序号 95：定义函数名为 main 的主函数。
序号 96：main 主函数开始。
序号 97：定义字符型局部变量。
序号 98：延时 500mS，等电源稳定。
序号 99：调用 LCM 初始化子函数。
序号 100：程序分隔。
序号 101~108：for 循环，将待显示字符串从右向左移到显示屏上进行显示。
序号 109：延时 2800mS，停顿一下。
序号 110：程序分隔。
序号 111~118：for 循环，将待显示字符串从左向右移出显示屏。
序号 119：清屏。
序号 120：延时 3000mS，停顿一下。
序号 121：程序分隔。
序号 122~130：显示屏字符串闪烁 5 次。
序号 131：延时 3000mS，停顿一下。
序号 132：程序分隔。
序号 133：无限循环。
序号 134：无限循环语句开始。
序号 135：程序分隔。
序号 136~143：for 循环，将待显示字符串从右向左移到显示屏上进行显示。
序号 144：程序分隔。
序号 145~155：for 循环，将待显示字符串从左向右移出显示屏。
序号 156：清屏。
序号 157：延时 200mS，停顿一下。
序号 158：无限循环语句结束。
序号 159：main 主函数结束。

实验二

在 LED/16*2 字符液晶试验板上实现 LCD 演示程序，第一行显示预定的字符串，第二行显示移动的 ASCII 字符。

在我的文档中建立一个文件目录（cs45），然后建立 cs45.uv2 的工程项目，最后建立源程序文件（cs45.c）。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
#include<INTRINS.H>                  //2
#define uchar unsigned char          //3
#define uint unsigned int            //4
sbit LCM_RS=P3^3;                    //5
sbit LCM_RW=P3^4;                    //6
sbit LCM_EN=P3^5;                    //7
#define DataPort P1                  //8
#define Busy 0x80                    //9
uchar code exampl []="For an example.
-By xiaoqi\n";//10
void Delay400ms(void);               //11
void Delay5ms(void);                 //12
void WaitForEnable(void);            //13
void WriteDataLCM(uchar data W);//14
void WriteCommandLCM (uchar CMD,
uchar Attribc)                       ;//15
```

```
void InitLcd(void);                    //16
void Display(uchar dd);                //17
void DisplayOneChar (uchar x,uchar y,
uchar Wdata);//18
void ePutstr (uchar x,uchar y,uchar code
*ptr);//19
//*****************************************20
void main(void)                        //21
{                                      //22
    uchar temp;                        //23
    Delay400ms();                      //24
    InitLcd();                         //25
    temp=32;                           //26
    ePutstr(0,0,exampl)     ;          //27
    Delay400ms();                      //28
    Delay400ms();                      //29
    Delay400ms();                      //30
    Delay400ms();                      //31
    Delay400ms();                      //32
    Delay400ms();                      //33
    Delay400ms();                      //34
    Delay400ms();                      //35
    while(1)                           //36
    {                                  //37
        temp&=0x7f;                    //38
        if(temp<32)temp=32     ;//39
        Display(temp++);               //40
        Delay400ms(); //41
    }                                  //42
}                                      //43
//*****************************************44
void ePutstr (uchar x,uchar y,uchar code
*ptr)                                  //45
{                                      //46
uchar i,l=0;                           //47
    while(ptr[l]>31){l++;}             //48
    for(i=0;i<l;i++){                  //49
    DisplayOneChar(x++,y,ptr[i]);//50
    if(x==16){                         //51
        x=0;y^=1;                      //52
    }                                  //53
 }                                     //54
}                                      //55
//*****************************************56
void Display(uchar dd)                 //57
{                                      //58
uchar i;                               //59
    for(i=0;i<16;i++){                 //60
    DisplayOneChar(i,1,dd++);          //61
    dd&=0x7f;                          //62
    if(dd<32)dd=32;                    //63
    }                                  //64
}                                      //65
//*****************************************66
void LocateXY(char posx,char posy)//67
{                                      //68
uchar temp;                            //69
    temp&=0x7f;                        //70
    temp=posx&0x0f;                    //71
    posy&=0x01;                        //72
    if(posy)templ=0x40;                //73
    templ=0x80;                        //74
    WriteCommandLCM(temp,0);//75
}                                      //76
//*****************************************77
void DisplayOneChar (uchar x,uchar y,
uchar Wdata)                           //78
{                                      //79
LocateXY(x,y);                         //80
WriteDataLCM(Wdata);                   //81
}                                      //82
//*****************************************83
void InitLcd(void)                     //84
{                                      //85
WriteCommandLCM(0x38,0);               //86
Delay5ms();                            //87
WriteCommandLCM(0x38,0);               //88
Delay5ms();                            //89
WriteCommandLCM(0x38,0);               //90
Delay5ms();                            //91
WriteCommandLCM(0x38,1);               //92
WriteCommandLCM(0x08,1);               //93
WriteCommandLCM(0x01,1);               //94
WriteCommandLCM(0x06,1);               //95
WriteCommandLCM(0x0c,1);               //96
}                                      //97
//*****************************************98
void WriteCommandLCM (uchar CMD,
uchar Attribc)//99
{                                      //100
if(Attribc)WaitForEnable();            //101
LCM_RS=0;LCM_RW=0;_nop_();             //102
DataPort=CMD;_nop_();                  //103
LCM_EN=1;_nop_();_nop_();LCM_EN=0;
                                       //104
}                                      //105
//*****************************************106
void WriteDataLCM(uchar dataW)//107
{                                      //108
WaitForEnable();                       //109
LCM_RS=1;LCM_RW=0;_nop_();             //110
DataPort=dataW;_nop_();                //111
LCM_EN=1;_nop_();_nop_();LCM_EN=0;
                                       //112
}                                      //113
//*****************************************114
void WaitForEnable(void)               //115
{                                      //116
DataPort=0xff;                         //117
LCM_RS=0;LCM_RW=1;_nop_();             //118
LCM_EN=1;_nop_();_nop_();              //119
while(DataPort&0x80);                  //120
LCM_EN=0;                              //121
}                                      //122
//*****************************************123
void Delay5ms(void)                    //124
{                                      //125
uint i=5552;                           //126
while(i--);                            //127
}                                      //128
//*****************************************129
void Delay400ms(void)                  //130
{                                      //131
uchar i=5;                             //132
uint j;                                //133
    while(i--)                         //134
    {                                  //135
    j=7269;                            //136
    while(j--);                        //137
    }                                  //138
}                                      //139
```

编译通过后，将生成的 cs45.hex 文件烧录到 89S51 芯片中，将芯片插入到 LED/16*2 字符液晶试验板上，连接好 16*2 字符液晶模组，试验板上接通 9V 电源，可以看到，液晶上第一行显示“For an example.”，第二行显示移动的 ASCII 字符。

我们对程序进行分析。

序号 1（程序解释，以下同）：包含头文件 REG51.H。

序号 2：包含头文件 INTRINS.H。

序号 3、4：数据类型的宏定义。

序号 5~8：端口定义。

序号 9：宏定义。

序号 10：待显示字符串。

序号 11~19：函数声明。

序号 20：程序分隔。

序号 21：定义函数名为 main 的主函数。

序号 22：main 主函数开始。

序号 23：定义局部变量。

序号 24：延时 400mS，等电源稳定。

序号 25：调用 LCM 初始化子函数。

序号 26：局部变量赋初值。

序号 27：第一行及第二行显示一个预定字符串。

序号 28~35：保留显示内容 3.2 秒。

序号 36：无限循环。

序号 37：无限循环语句开始。

序号 38：只显示 ASCII 字符。

序号 39：屏蔽控制字符，不予显示。

序号 40：显示 ASCII 字符。

序号 41：延时 400mS，便于观察。

序号 42：无限循环语句结束。

路灯稳压光控器的试制

◆广西贺州市技术工人学校 黄家植

本文介绍笔者独立设计试制的自动稳压光控系统。该系统由光控电路和自动稳压电路组成，经多年运行的实践证明是性能可靠的，值得介绍。本文评述了设计意图、安装调试和使用等有关问题。

作者单位的路灯多达上百盏，功率总额高达 5KW 以上，且市电电压由于距变电站较近而高达 240～250 伏，过高的电压造成路灯灯泡被烧的事情频频发生，换灯泡成了家常便饭。再者，手动控制路灯又是地点分散，不便操作。因此，笔者设计安装了一个自动稳压光控路灯系统。

一、设计意图

为减轻水电管理员手工操作的劳动量，笔者设计一个光控路灯装置，使其夜幕降临时自动开通路灯，黎明来临时自动关闭路灯，夜间闪电时不会使路灯出现突发性熄灭。采用 555 时基集成块作为核心的光控电路来实现这一功能。

为了减少因市电电压过高而造成路灯使用寿命缩短，浪费资金和人力，笔者设计一个稳定电压的装置，使其具备不随电源电压波动的稳压性能，当电源电压在 230～250 伏的电压范围内变化时，该稳压装置的输出电压变化值不超过设定的输出电压的 0.5%。为了节约用电，提高路灯的使用寿命，务必使稳压装置的输出电压在 150～220 伏范围内调整电压，采用具有负反馈的单结触发晶闸管的交流调压电路来实现这一功能。

为了便于试电和检修路灯时能手动控制路灯电源，笔者增设一个手动控制开关来控制光控装置的接

序号 43：main 主函数结束。
序号 44：程序分隔。
序号 45：显示指定座标的一串字符(x=0~15,y=0~1)子函数，函数名为 ePutstr，定义 x、y 为无符号字符型变量，ptr 为指向 code 区的无符号字符型指针变量。
序号 46：ePutstr 子函数开始。
序号 47：定义 i、l 为无符号字符型变量。
序号 48：ptr[l]大于 31 时，为 ASCII 码，进入 while 语句循环，l 累加，计算出字符串长度。
序号 49：进入 for 语句循环。
序号 50：显示单个字符，同时 x 轴座标递增。
序号 51：若 x 等于 16，进入 if 语句。
序号 52：x 赋 0，y 与 1 按位异或(取反)。
序号 53：if 语句结束。
序号 54：for 语句结束。
序号 55：ePutstr 子函数结束。
序号 56：程序分隔。
序号 57：演示第二行移动字符串子函数，函数名为 Display，定义 dd 为无符号字符型变量。
序号 58：Display 子函数开始。
序号 59：定义 i 为无符号字符型变量。
序号 60：进入 for 语句循环。
序号 61：显示单个字符。
序号 62：dd 的变化范围 0~127。
序号 63：dd 的最小值为 32，这样 dd 的变化范围为 32~127。
序号 64：for 语句结束。
序号 65：Display 函数结束。
序号 66：程序分隔。
序号 67：显示光标定位子函数，函数名为 LocateXY，定义 posx、posxy 为字符型变量。
序号 68：LocateXY 子函数开始。
序号 69：定义 temp 为无符号字符型变量。
序号 70：temp 的变化范围 0~15。
序号 71：屏蔽高 4 位。
序号 72：posy 的变化范围 0~1。
序号 73：若 posy 为 1(显示第二行)，地址码 +0x40。
序号 74：指令码为地址码 +0x80。
序号 75：将指令 temp 写入 LCM，忽略忙信号检测。
序号 76：LocateXY 子函数结束。
序号 77：程序分隔。
序号 78：显示指定座标的一个字符(x=0~15,y=0~1)子函数，函数名为 DispOneChar，定义 x、y、Wdata 为无符号字符型变量。
序号 79：DispOneChar 函数开始。
序号 80：调用 LocateXY 函数定位显示地址。
序号 81：将数据 Wdata 写入 LCM。
序号 82：DispOneChar 函数结束。
序号 83：程序分隔。
序号 84~97：LCM 初始化。
序号 98：程序分隔。
序号 99~105：写命令到 LCM 子程序。
序号 106：程序分隔。
序号 107~113：写数据到 LCM 子程序。
序号 114：程序分隔。
序号 115~122：等待使能。
序号 123：程序分隔。
序号 124~128：5Ms 短延时子函数。
序号 129：程序分隔。
序号 130~139：400Ms 短延时子函数。