

手把手教你学单片机的 C 语言程序设计(十七)

◆ 吕超亚

点阵图形液晶模块的简介

点阵图形液晶模块是一种用于显示各类图像、符号、汉字的显示模块,其显示屏的点阵像素连续排列,行和列在排布中没有间隔,因此可以显示连续、完整的图形。当然它也能显示字母、数字等字符。点阵图形液晶模块依控制芯片的不同,其功能及控制方法与点阵字符液晶模块相比略有不同。点阵图形液晶模块的控制芯片生产厂家较多,以下为典型的几种。

HD61202[日立公司产品]、T6963C[东芝公司产品]、HD61830(B)[日立公司产品]、SED1330(E-1330)[精工公司产品]、MSM6255[冲电气公司产品]。

这里以市场上常见的 128x64 点阵图形液晶模块为例来做介绍,该液晶模块采用日立的 HD61202 和 HD61203 芯片组成。128x64 点阵图形液晶模块,表示横向有 128 点,纵向有 64 点,如果以汉字 16x16 点而言,每行可显示 8 个中文字,4 行共计 32 个中文字。

1. 128x64 点阵图形液晶模块的内部结构

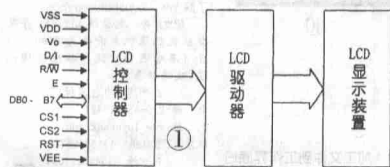
128x64 点阵图形液晶模块的内部结构可分为三个部分:LCD 控制器, LCD 驱动器, LCD 显示装置,如图 1 所示。

点阵图形液晶 128x64 分行、列驱动器, HD61203 是行驱动器, HD61202 是列驱动器。

2. HD61202 的组成及工作原理

HD61202 的内部组成结构如图 2 所示。

(1) I/O 缓冲器



I/O 缓冲器为双向三态数据缓冲器。是 HD61202 内部总线与计算机总线连接部。其作用是将两个不同时钟下工作的系统连接起来,实现通讯。I/O 缓冲器在三个片选信号 /CS1、/CS2 和 CS3 组合有效状态下开放,实现

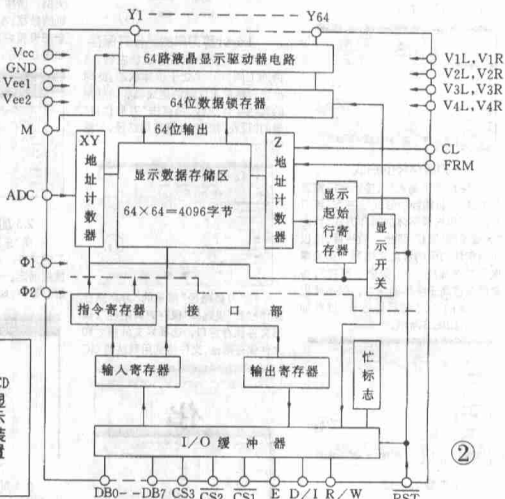
HD61202 与计算机之间的数据传递。当片选信号组合为无效状态时, I/O 缓冲器将中断 HD61202 内部总线与计算机数据总线的联系,对外总线呈高阻状态。

(2) 输入寄存器

输入寄存器用于暂时储存要写入显示 RAM(显示存储器)的数据。因为数据是由 MCU 写入输入寄存器,然后再由内部处理后自动地写入显示存储器内。当 CS=1, D/I=1, 且 R/W=1 时,数据在使能信号 E 的下降沿被锁入输入寄存器。

(3) 输出寄存器

从显示 RAM 中读出的数据首先暂时储存在输出寄存器。MCU 要从输出寄存器读出数据则要令 CS=D/I=R/W=1。不过读数据命令时,存于输出寄存器中的数据是在 E 脚为



高电平时输出；然后在 E 脚信号落为低电平时，地址指针指向的显示数据接着被锁入输出寄存器而且地址指针递增。输出寄存器中，会因读数据的指令而被再写入新的数据，若为地址指针设定指令则数据维持不变。因此，发送完地址设定指令之后随即发送读取数据指令，将无法得到所指定位址的数据，必须再接着读取一次数据，该指定地址的数据才会输出。

(4)显示存储器电路

HD61202 具有 4096 位显示存储器。其结构是以一个 64X64 位的方阵形式排布的。显示存储器的作用一是存储计算机传来的显示数据，二是作为控制信号源直接控制液晶驱动电路的输出。显示存储器为双端口存储器结构。结构原理示意图如图 3 所示。

从数据总线侧看有 64 位，按 8 位数据总线长度分成 8 路，称为页面，由 X 地址寄存器控制；每个页面都有 64

器使用的，仅有输出形式。

HD61202 列驱动器为 64 列驱动输出，正好与显示存储器列向(纵向)单元对应。Z 地址计数器为显示行指针，用来选择当前要传输的数据行。

(5)XY 地址计数器

XY 地址计数器为 9 位的寄存器，它确定了计算机所需访问的显示存储器单元的地址。X 地址计数器为高 3 位，Y 地址计数器为低 6 位，分别有各自的指令来设定 X、Y 地址。计算机在访问显示存储器之前必须要设置 XY 地址计数器。计算机写入或读出显示存储器的数据代表显示屏上某一列上的垂直 8 点的数据。D0 代表最上一点的数据。

X 地址计数器是一个 3 位页地址寄存器，其输出控制着显示存储器中 8 个页面的选择，也就是控制着数据传输通道的八选一选择器。X 地址寄存器可以由计算机以指令形式设置。X

形。Y 地址计数器可以由计算机以指令形式设置，它和页地址指针结合唯一选通显示存储器的一个单元。Y 地址计数器具有自动加一功能。在显示存储器读/写操作后 Y 地址计数器将自动加一。当计数器加至 3FH 后循环归零再继续递加。

(6)显示起始行寄存器

显示起始行寄存器为 6 位寄存器，它规定了显示存储器所对应显示屏上第一行的行号。该行的数据将作为显示屏上第一行显示状态的控制信号。显示起始行寄存器的内容由计算机以指令代码的格式写入。此寄存器指定 RAM 中某一行数据对应到 LCD 屏幕的最上行，可用做荧幕卷动。

(7)Z 地址计数器

Z 地址计数器也为 6 位地址计数器，用于确定当前显示行的扫描地址。Z 地址计数器具有自动加一功能，它与行驱动器的行扫描输出同步，选择相应的列驱动器的数据输出。在行驱动器发来的 CL 时钟信号脉冲的下降沿时加一。在 FRM 信号的高电平时置入显示起始行寄存器的内容，以作为再循环显示的开始。

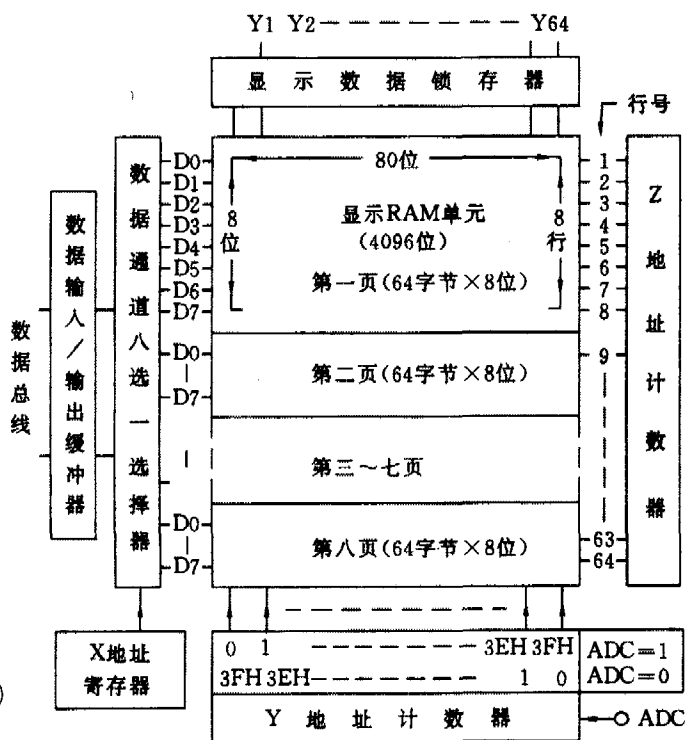
(8)显示开/关触发器

该触发器的输出一路控制显示数据锁存器的清除端，一路返回到接口控制电路作为状态字中的一位表示当前的显示状态。该触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。在触发器输出为“关”电平时，显示数据锁存器的输入被封锁并将输出置“0”，从而使显示驱动输出全部为非选择波形，显示屏呈不显示状态。在触发器输出为“开”电平时，显示数据锁存器受 CL 控制，显示驱动输出受显示驱动数据总线上数据控制，显示屏将呈显示状态。显示开/关触发器受逻辑电路控制，计算机可以通过硬件/RST 复位和软件指令“显示开关设置”的写入来设置显示开/关触发器的输出状态。

(9)指令寄存器

指令寄存器用于接收计算机发来的指令代码，通过译码将指令代码置入相关的寄存器或触发器内。

(10)状态字寄存器



③

个字节，用 Y 地址计数器控制，这一侧是提供给计算机操作的，是双向传输形式。XY 地址计数器选择了计算机所要操作的显示存储器的页面和列地址，从而唯一地确定计算机所要访问的显示存储器单元。从驱动数据传输侧看有 64 位，共 64 行，这一侧是提供给驱动

地址寄存器没有自动修改功能，所以要想转换页面需要重新设置 X 地址寄存器的内容。

Y 地址计数器是一个 6 位循环加一计数器。它管理某一面上的 64 个单元，该数据总线上的 64 位数据直接控制驱动电路输出 Y1~Y64 的输出波

状态字寄存器是 HD61202 与计算机通讯时唯一的“握手”信号。状态字寄存器向计算机表示了 HD61202 当前的工作状态。其中最主要的是忙碌信号(Busy),当忙碌信号为“1”,表示 HD61202 正在忙于内部运作,除了状态读取指令外,其他任何指令部不被接受。忙碌信号(Busy)是由状态字读取指令所读出 DB7 表示。每次要发指令前,应先确定忙碌信号已为“0”。

(11) 显示数据锁存器

数据要从显示数据 RAM 中输出到液晶驱动电路前,先暂时储存在此锁存器中,在时钟信号上升沿时数据被锁存。显示器开/关指令控制此锁存器动作,不会影响显示数据 RAM 中的数据。

HD61202 的工作过程:

计算机要想访问 HD61202,必须首先读取状态字寄存器的内容,主要是要判别状态字中的“Busy”标志;在“Busy”标志表示为 0 时,计算机方可访问 HD61202。在写操作时,HD61202 在计算机写操作信号的作用下将计算机发来的数据锁存进输入寄存器内,使其转到 HD61202 内部时钟的控制之下,同时 HD61202 将 I/O 缓冲器封锁,置“Busy”标志位为 1,向计算机提供 HD61202 正在处理计算机发来的数据的信息。HD61202 根据计算机在写数据时提供的 D/I 状态将输入寄存器的内容送入指令寄存器处理或显示存储器相应的单元,处理完成后,HD61202 将撤消对 I/O 缓冲器的封锁,同时将“Busy”标志位清零,向计算机表示 HD61202 已准备好接收下一个操作。

在读显示数据时,计算机要有一个操作周期的延时,即“空读”的过程。这是因为在计算机读操作下,HD61202 向数据总线提供输出寄存器当前的数据,并在读操作结束时将当前地址指针所指的显示存储器单元的数据写入输出寄存器内,同时将列地址计数器加一。也就是说计算机不是直接读取到显示存储器单元,而是读取一个中间寄存器——输出寄存器的数据。而这个数据是上一次读操作后存入到输出寄存器的内容,这个数据可能是上一地址单

元的内容,也可能是地址修改前某一单元的内容。因此在计算机设置所要读取的显示存储器地址后,第一次的读操作实际上是要要求 HD61202 将所需的显示存储器单元的数据写入输出寄存器中,供计算机读取。只有从下一次计算机的读操作起,计算机才能读取所需的显示数据。

3. 点阵图形液晶模块的控制器指令和操作时序

(详见本期“128×64 点阵液晶的工作原理及其驱动”一文)

4. 单片机与点阵图形液晶模块的接口电路

图形液晶模块与单片机的连接方式也有两种,即直接访问方式(总线方式)和间接控制方式(模拟口线)。我们采用的是间接控制方式电路(见本刊 2005 年 8 月号上刊登的 LED/128*64 图形液晶试验板电路图)。

点阵图形液晶模块的 C 语言应用设计实例

在 LED/128*64 图形液晶试验板上显示汉字:屏幕上第一行显示“朝辞白帝彩云间”,第二行显示“千里江陵一日还”,第三行显示“两岸猿声啼不住”,第四行显示“轻舟已过万重山。”其中第三、四行反白显示。

在我的文档中建立一个文件目录(cs46),然后建立 cs46.uv2 的工程项目,最后建立源程序文件(cs46.c)。

输入下面的程序:

```
#include <REG51.H> // 包含头文件
#define uchar unsigned char // 变量类型标识的宏定义
#define uint unsigned int
sbit p_csl=P3^4 // 引脚定义,左半屏片选信号。
sbit p_csr=P3^3 // 引脚定义,右半屏片选信号。
sbit p_di=P3^7 // 引脚定义,寄存器选择。
sbit p_rw=P3^6 // 引脚定义,读/写选择。
sbit p_e=P3^5 // 引脚定义,使能操作。
sbit p_rst=P3^2 // 引脚定义,复位信号。
```

```
#define lcm P1 // 端口定义,双向数据总线。
/***** 函数列表 *****/
void delay(unsigned long v);
void wcode(uchar c,uchar csl,uchar csr);
void wdata(uchar c,uchar csl,uchar csr);
void set_startline(uchar i);
void set_xy(uchar x,uchar y);
void dison_off(uchar o);
void reset();
void lcd_init(void);
void lw(uchar x, uchar y, uchar dd);
void dh (uchar x, uchar y, uchar n, uchar fb);
uchar code hz[];
/***** 主函数 *****/
void main(void)
{
    uchar loop; // 定义字符型局部变量 loop
    lcd_init(); // LCM 初始化
    delay(1000); // 延时一会
    while(1) // 无限循环
    {
        /***** 显示第一行(8 个字) *****/
        for(loop=0;loop<8;loop++)
        {
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
            dh(2*loop,0,loop,0);
        }
        /***** 显示第二行(8 个字) *****/
        for(loop=0;loop<8;loop++)
        {
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
            dh(2*loop,2,loop+8,0);
        }
        /***** 显示第三行(8 个字) *****/
        for(loop=0;loop<8;loop++)
        {
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
            dh(2*loop,4,loop+16,1);
        }
        /***** 显示第四行(8 个字) *****/
        for(loop=0;loop<8;loop++)
        {
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
            dh(2*loop,6,loop+24,1);
        }
    }
}
```

```

dh(2*loop,6,loop+24,1);
dh(2*loop,6,loop+24,1);
dh(2*loop,6,loop+24,1);
dh(2*loop,6,loop+24,1);}
/*****
delay(10000); //延时一会
}
}
/*----- 延时子函数 ----*/
void delay (unsigned long v) // 函数名为
delay 的延时子函数。定义 v 为长变量
{ //delay 函数开始
while (v!=0)v--; //while 循环体,v 自减后
若不为 0 则继续循环自减
} //delay 函数结束
/*----- 判 LCM 忙子函数 -----*/
void lcd_busy (void) // 函数名为
lcd_busy 的判 LCM 忙子函数
{ //lcd_busy 函数开始
p_di=0;p_rw=1;lcm=0xff; // 选择指令寄存
器,选择读方式,LCM 数据口置全 1
while(1){ //while 循环体,无限循环
p_e=1; // 使能, 将 LCM 的状态读入
MCU
if(lcm<0x80) break; /* 若读入状态的数据
小于 0x80, 说明最高位为 0,LCM 空闲,执
行 break 语句跳出 while 循环体 */
p_e=0; // 禁能
} // while 循环体结束
p_e=0; // 禁能
} //lcd_busy 函数结束
/*----- 写指令到 LCM 子函数 -----*/
void wcode (uchar c,uchar csl,uchar csr)
/* 函数名为 wcode 的写指令到 LCM 子函
数。定义
c,csl,csr 为无符号字符型变量 */
{ //wcode 函数开始
p_csl=csl; /* 将 csl,csr 变量赋予 p_csl,
p_csr,用以选择 LCM
的左半屏或右半屏 */
p_csr=csr;
lcd_busy(); // 调用判 LCM 忙子函数
p_di=0; // 选择指令寄存器
p_rw=0; // 选择写
lcm=c; // 将变量 c 赋予 LCM
p_e=1; // 使能
p_e=0; // 禁能
} //wcode 函数结束
/*----- 写数据到 LCM 子函数 -----*/
void wdata(uchar c,uchar csl,uchar csr) /*
函数名为 wdata 的写数据子函数。定义
c,csl,csr 为无符号字符型变量 */
{ //wdata 函数开始
p_csl=csl; /* 将 csl,csr 变量赋予 p_csl,
p_csr,用以选择 LCM
的左半屏或右半屏 */
p_csr=csr;
lcd_busy(); // 调用判 LCM 忙子函数
p_di=1; // 选择数据寄存器
p_rw=0; // 选择写
lcm=c; // 将变量 c 赋予 LCM
p_e=1; // 使能
p_e=0; // 禁能
} //wdata 函数结束
/*----- 设定起始行子函数 -----*/
void set_startline (uchar i) /* 函数名为
set_startline 的设定起始行子函数。定义
i 为无符号字符型变量 */
{ //set_startline 函数开始
i=0xc0+i; // 设定起始行指令代码
wcode(i,1,1); // 将指令代码写入 LCM 的
左半屏及右半屏
} //set_startline 函数结束
/*-----*/
void set_xy(uchar x,uchar y) // 定位 x 方
向,y 方向的子函数
{
x=x+0x40;
y=y+0xb8;
wcode(x,1,1);
wcode(y,1,1);
}
/*-- 定位 x 方向,y 方向的子函数 ---*/
void set_xy (uchar x,uchar y) /* 函数名为
set_xy 的定位 x 方向,y 方向的子函数。定
义 x,y 为无符号字符型变量 */
{ //set_xy 函数开始
x=x+0x40; // 设定 x 列的指令代码
y=y+0xb8; // 设定 y 页的指令代码
wcode (x,1,1); // 将 x 列的指令代码写入
LCM 的左半屏及右半屏
wcode (y,1,1); // 将 y 页的指令代码写入
LCM 的左半屏及右半屏
} //set_xy 函数结束
/*--- 屏幕开启、关闭子函数 ----*/
void dison_off (uchar o) /* 函数名为
dison_off 的屏幕开启、关闭子函数。定义 o
为无符号字符型变量 */
{ //dison_off 函数开始
o=o+0x3e; // 设定开、关屏幕的指令代
码。o 为 1 开,o 为 0 关
wcode(o,1,1); /* 将开、关屏幕的指令代
码写入 LCM 的左半屏及
右半屏 */
} //dison_off 函数结束
/*----- 复位子函数 ----*/
void reset() /* 函数名为 reset 的复位子
函数 */
{ //reset 函数开始
p_rst=0; // 复位端置低电平
delay(20); // 延时一会
p_rst=1; // 复位端置高电平
delay(20); // 延时一会
} //reset 函数结束
/*-----LCM 初始化子函数 -----*/
void lcd_init (void) /* 函数名为 lcd_init 的
LCM 初始化子函数 */
{uchar x,y; //lcd_init 子函数开始,定义 x,y
为无符号字符型局部变量
reset(); // 调用 LCM 复位子函数
set_startline(0); // 设定起始行为第一行
dison_off(0); // 显示屏关
for(y=0;y<8;y++) // 建立一个循环 8 次
(共 8 页)的 for 循环体
{
for(x=0;x<128;x++)lw(x,y,0); /* 建立
一个循环 128 次(共 128 列)的 for 循环体,
每列置 0 */
}
dison_off(1); // 显示屏开
} //lcd_init 子函数结束
/*---- 写数据至 LCM 子函数 -----*/
void lw(uchar x, uchar y, uchar dd) /* 函数
名为 lw 的写数据至 LCM 子函数。定义 x、
y、dd 为无符号
字符型局部变量 */
{ //lw 子函数开始
if(x>=64) // 若 x 大于等于 64,说明为
右半屏操作
{set_xy(x-64,y);
//x(列)值减去 64,获得右半屏定位
wdata(dd,0,1); // 将 dd 变量中的数据
写入 LCM 右半屏
}
else // 否则 x 小于 64, 说明为左半屏
操作
{set_xy(x,y); // 获得左半屏定位
wdata(dd,1,0); // 将 dd 变量中的数据
写入 LCM 左半屏
} //lw 子函数结束
/*----- 显示汉字子函数 ----*/
void dh (uchar xx, uchar yy, uchar n, uchar
fb) /* 函数名为 dh 的显示汉字子函数。
定义 xx,yy,n,fb 为无符号字符型局部变
量。其中 xx,yy 为列、页定位值,
n 为汉字点阵码表中的第 n 个汉字,fb 为反
白显示选择 */
{ //dh 子函数开始
uchar i,dx; // 定义 i,dx 为无符号字符型局
部变量
for(i=0;i<16;i++)
//for 循环体,用于扫描汉字的上半部分
{dx=hz [2*i+n*32]; // 取得第 n 个汉字的上
半部分数据代码
if(fb)dx=255-dx; // 若 fb 不为 0,获得反白
数据代码
lw(xx*8+i,yy,dx); // 将数据代码写入 LCM
dx=hz[(2*i+1)+n*32]; // 取得第 n 个汉字
的下半部分数据代码
if(fb)dx=255-dx; // 若 fb 不为 0,获得反
白数据代码
lw(xx*8+i,yy+1,dx); // 将数据代码写入
LCM
} //for 循环体结束
} //dh 子函数结束
/***** 汉字点阵码表 *****/
uchar code
hz[]=

```

