深圳市拓普微科技开发有限公司
SHENZHEN TOPWAY TECHNOLOGY CO., LTD.

液晶显示控制器 UC1608
应用指南

深圳市拓普微科技开发有限公司

版本	描述	日期	编者
0.1	新版本	2009-06-24	郭强
0.2	P7 SdCmd(0x2C);改为 SdCmd(0x26);	2010-1-23	郭强

目 录

第一章 液晶显示控制器接口特性

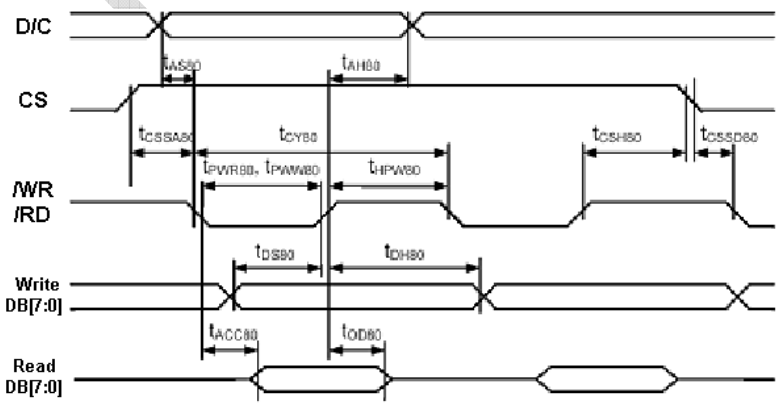
UC1608 控制器的基本特性如下，使用 UC1608 的液晶显示模块利用了这些特性构造了模块产品应用的主要功能：

- 工作电源：3V
- 显示功能：单显示 RAM 区域、垂直卷动等
- 接口信号：

管脚符号	管脚定义
CS	片选信号输入端，高电平有效。CS=1 时选通模块；CS=0 时模块接口被封锁
/RES	复位信号输入端，低电平复位；正常运行时，为高电平状态
D/C	通道选择信号输入端，当 D/C=0 时，选择指令通道；D/C=1 时，选择数据通道
/WR（R/W）	输入端。当并行接口 INTEL8080 时序时，为写信号/WR，低电平有效； 当并行接口 INTEL6800 时序时，为读/写选择信号，R/W=1，为读状态，R/W=0，为写状态
/RD（E）	输入端。当并行接口 INTEL8080 时序时，为读信号/RD，低电平有效； 当并行接口 INTEL6800 时序时，为使能信号，E 为高电平时，为读操作，E 下降沿为写操作
DB0（SCK）	输入/输出/三态。并行接口数据总线，当选择串行接口时，为串行时钟信号输入端
DB1	输入/输出/三态。并行接口数据总线，当选择串行接口时，接 VSS
DB2	输入/输出/三态。并行接口数据总线，当选择串行接口时，接 VSS
DB3（SDA）	输入/输出/三态。并行接口数据总线，当选择串行接口时，为串行数据输入端
DB4	输入/输出/三态。并行接口数据总线，当选择 4 位并行接口或串行接口时，接 VSS
DB5	输入/输出/三态。并行接口数据总线，当选择 4 位并行接口或串行接口时，接 VSS
DB6	输入/输出/三态。并行接口数据总线，当选择 4 位并行接口或串行接口时，接 VSS
DB7	输入/输出/三态。并行接口数据总线，当选择 4 位并行接口时，接 VSS；选择串行接口时，接 VDD

- 接口形式：8 位/4 位并行接口、4 线/3 线串行接口
- 操作模式：80mode(默认) 和 68mode
- 时序关系：

1、并口时序（80mode）

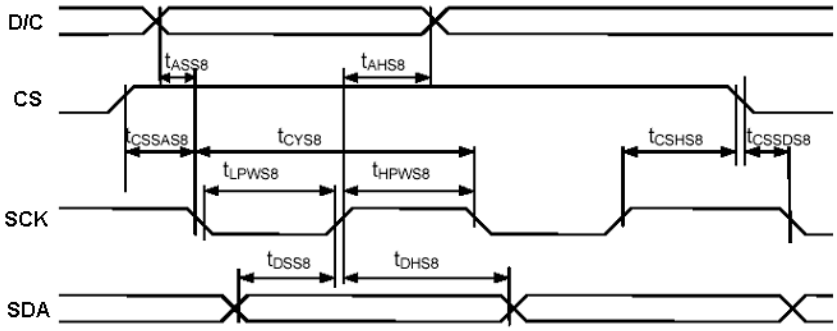


图一 80mode 模式时序图

时序参数表 （测试条件：2.7≤VDD＜3.3V,ta=-30--+85C）

描述	符号	信号	条件	最小	最大	单位
地址建立时间 地址保持时间	t_{AS80} t_{AH80}	D/C		0 20		ns
系统时钟周期 8/4 位总线（读/写）	t_{CY80}			140		ns
读脉冲宽度（8/4 位）	t_{PWR80}	/RD		65		ns
写脉冲宽度（8/4 位）	t_{PWW80}	/WR		35		ns
高脉冲宽度 写信号（8/4 位） 读信号（8/4 位）	t_{HPW80}	/WR /RD		35 65		ns
数据建立时间 数据保持时间	t_{DS80} t_{DH80}	DB0-DB7		30 20		ns
读取时间 输出无效时间	t_{ACC80} t_{OD80}		C_L=100pF	- 12	60 20	ns
片选建立时间	t_{SSA80} t_{CSSD80} t_{CSH80}	CS		10 10 20		ns

2、串口时序：



图二、4 线 SPI 串行接口时序图

时序参数表 （测试条件：2.7≤VDD＜3.3V,ta=-30--+85C）

符号	描述	信号	MIN	MAX	单位
t_{SCYC}	串行时钟周期	SCK	50	-	ns
t_{SHW}	SCL 高脉冲宽度		25	-	
t_{SLW}	SCL 低脉冲宽度		25	-	
t_{SAS}	地址建立时间	D/C	20	-	
t_{SAH}	地址保持时间		10	-	
t_{SDS}	数据建立时间	SDA	20		
t_{SDH}	数据保持时间		10	-	
t_{CSS}	CS 建立时间	CS	20		
t_{CSH}	CS 保持时间		40		

第二章　液晶显示控制器接口技术

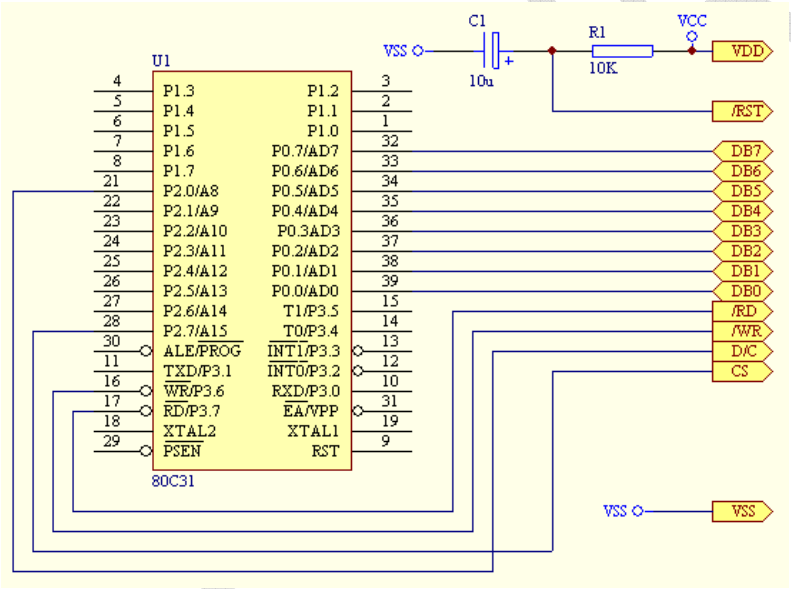
对于液晶显示模块的接口而言，也是液晶显示控制器的接口，因此我们在探讨控制器的接口技术时，也是在探讨模块的接口技术。所以下面我们将“控制器”用“模块”代替，以期更清楚的描述控制器的应用。

一般来说，在计算机系统里，液晶显示模块属于低速外设，所以在与计算机连接时，需要注意双方的时序搭配。深圳市拓普微科技开发有限公司使用 UC1608 的液晶显示模块提供了多种的接口形式，用户可以根据自己的控制系统时序特性和系统资源，进行合理的选择。本章将以单片机 89S52 为控制系统，以模块 LM240120A 的 INTEL8080 时序接口的为实例 提供在总线寻址方式下和 I/O 寻址方式下接口的示意电路和驱动子程序，同时还推荐了串行接口的应用方法。

在并行接口方式下，模块还可以使用 4 位并行方式，在 4 位方式下，模块使用了数据线的低 4 位（DB3-DB0）作为数据总线，数据传输格式是先高 4 位，再低 4 位。传输时序关系同 8 位数据总线形式，只是一个字节的数据需要传输两次完成，这里将不做讨论。

一、 总线寻址方式接口电路及驱动程序

MPU 使用总线方式与液晶显示模块直接连接，模块接口时序采用 INTEL8080 时序，如图三所示：



图三　　总线寻址方式接口电路示意图

总线寻址方式是模块的数据总线直接挂在 89S52 的数据总线上，MPU 的/RD、/WR 作为模块的读、写控制信号， CS 信号和 D/C 信号都由地址线译码产生，模块的/RST 接 RC 复位电路。

总线寻址方式驱动子程序如下：（地址定义，根据用户平台接口修改）

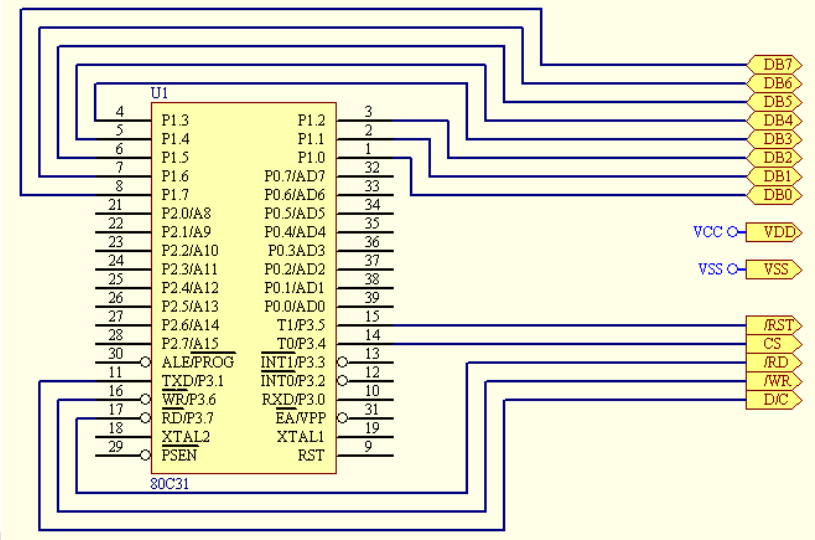
```
//-----指令代码写入函数----------------------
void SdCmd(uchar Command)
{
    uchar xdata *wcom_addr;
    wcom_addr =0x8000;                    // 指令口地址
    *wcom_rdata_addr = Command;           // 写指令操作
}
//-----数据写入函数--------------------------
void SdData(uchar DData)
{
    uchar xdata *wdata_addr;
```

```
    wdata_addr =0x8100;                    // 数据口地址
    *wdata_addr = DData;                   // 写数据操作
}

//-----数据读取函数-------------------------
uchar RdData()
{
  uchar DData;
  uchar xdata *rdata_addr;
  rdata_addr = 0x8100;          // 读数据地址
  DData = *rdata_addr;          // 读数据操作
   return(DData);               //返回数据值
}
```

二、 I/O 寻址方式接口电路及驱动程序

I/O 寻址方式是 MPU 通过 I/O 并行接口连接模块，通过软件编程模拟信号之间的时序关系，间接实现对模块进行控制。该方式能够很好的回避 MPU 和模块接口之间的时序差异。根据模块的接口信号要求，需要占用 MPU 的 2 个并行接口，在图四给出的示例中，我们将 89S52 的 P1 口作为数据总线。P3 口中 4 位端口为控制信号，它们是：P3.1 为 D/C 信号，P3.7 为/RD 信号，P3.6 为/WR 信号，P3.4 为 CS 信号，P3.5 为/RST 信号。深圳市拓普微科技开发有限公司制作



图四　I/O 寻址方式电路示意图

I/O 寻址方式的驱动子程序如下：

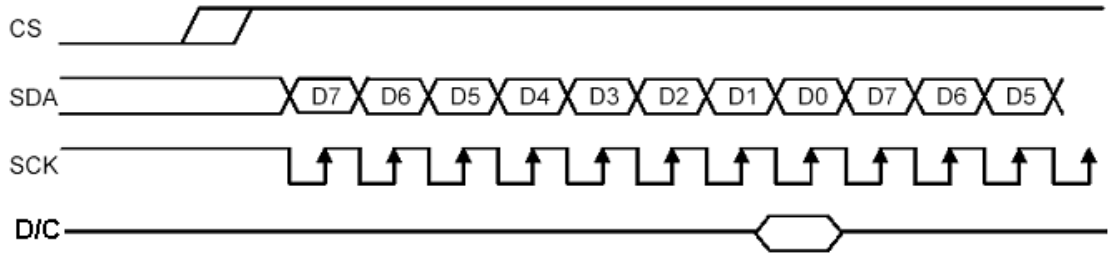
```
//-----------------------------------
#define LCDBUS      P1
sbit      _RD             = P3^7;
sbit      _WR             = P3^6;
sbit      D_C             = P3^1;
sbit      CS              = P3^4;
sbit      _RST            = P3^5;
//-------指令代码写入函数-------------------------
void SdCmd(uchar Command)
{
    D_C   = 0;                          // 选择指令通道
    LCDBUS   = Command;                 // 设置指令代码
    CS    = 1;                          // 选通模块
    _WR   = 0;                          // 写信号有效
    _WR   = 1;                          // 写信号无效
    CS    = 0;                          // 封锁模块
}
//------数据写入函数-----------------------------
void SdData(uchar DData)
```

```
{
    D_C   = 1;                      // 选择数据通道
    LCDBUS   = DData;               // 设置数据
    CS    = 1;                      // 选通模块
    _WR   = 0;                      // 写信号有效
    _WR   = 1;                      // 写信号无效
    CS    = 0;                      // 封锁模块
}

//-------数据读取函数-------------------------
uchar RdData()
{
   uchar DData;
    D_C =1;                         // 选择数据通道
    LCDBUS=0xff;
    CS=1;                           // 选通模块
    _RD   = 0;                      // 读操作信号有效
    DData = LCDBUS;                 // 读数据
    _RD   = 1;                      // 读操作信号无效
    CS=0;                           // 封锁模块
    return(DData);                  // 返回数据值
}
```

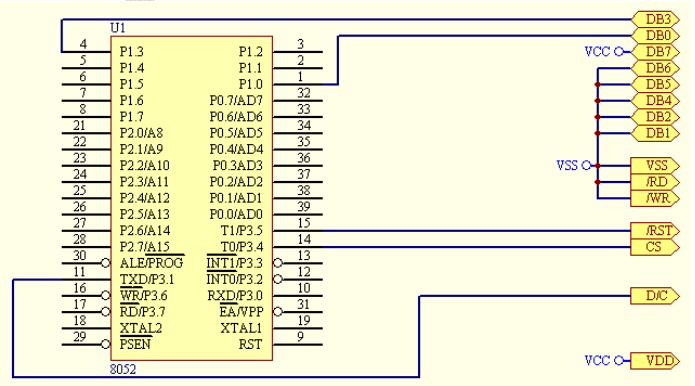
三、 串行接口

使用 UC1608 的液晶显示模块的串行接口仅需要 4 个信号线，SCK 为串行时钟信号，上升沿有效；SDA 为串行数据端；D/C 为数据的属性，D/C=0 为指令码，D/C=1 为显示数据； CS 仍为模块的片选信号。串行数据传输的通讯协议也非常简单，见图五所示。在准备启用串行接口前，要将先将 SCK 信号设置为高电平，设置 D/C 状态，然后再将 CS 置“1”，选通模块，再将 SCK 置成“0”；SCK 的每个正脉冲的上升沿都将会把 SDA 数据送入内部缓冲器内，当第 8 个脉冲的上升沿时，SCK 将 D/C 状态读入内部控制位，并将已经读入 8 位数据并行送入 D/C 所确定的寄存器内进行处理。



图五　串行接口通讯规则

本章提供了串行接口电路示意图，见图六所示，并根据该电路制作了驱动子程序如下。需要说明的是，在串行接口的通讯中，只有向模块的写入功能，没有读取功能。深圳市拓普微科技开发有限公司制作



图六 串行接口电路示意图

串行接口驱动子程序如下：

```
//---  串行通讯方式驱动子程序------------------------------
uchar bdata transdata;        //该变量可为位操作之变量
sbit transbit    = transdata^7;
sbit       CS              = P3^4;
sbit       _RST            = P3^5;
sbit       D_C             = P3^1;
sbit       SCK             = P1^0;
sbit       SDA             = P1^3;
//-------指令代码写入函数-----------------------
void SdCmd(uchar Command)
{
    uchar j;
    transdata=Command;              // 指令送位寄存器
     SCK = 1;                       // 初始化 SCK
     D_C = 0;                       // 选择指令通道
      CS = 1;                       // 选通模块
     SCK = 0;
    for(j=0;j<8;j++)
      {
       SDA=transbit;                // 位寄存器 D7 位送数据口
       SCK=1;                       // 产生移位脉冲
       SCK=0;                       // 上升沿有效
       transdata=transdata<<1;      // 位寄存器数据左移一位
       }
    CS=0;                           // 封锁模块
}
//-----数据写入函数--------------------------
void SdData(uchar DData)
{
    uchar j;
    transdata=DData;                // 指令送位寄存器
    SCK = 1;                        // 初始化 SCLK
    D_C = 1;                        // 选择数据通道
     CS = 1;                        // 选通模块
    SCK = 0;
    for(j=0;j<8;j++)
      {
       SDA=transbit;                // 位寄存器 D7 位送数据口
       SCK=1;                       // 产生移位脉冲
       SCK=0;                       // 上升沿有效
       transdata=transdata<<1;      // 位寄存器数据左移一位
       }                            // 深圳市拓普微科技开发有限公司制作
    CS=0;                           // 封锁模块
}
```

三、初始化子程序

UC1608 的初始化程序中，我们设置了启用内部 LCD 驱动电压，对比度参数 ContrastLevel 的设置根据模块和使用环境设置。

1、初始化函数

```
void initLCDM(void)
{
_RST=1;                     // 硬件初始化复位
delayms(100);
_RST=0;
delayms(1);
_RST=1;
delayms(800);               // 推荐延迟时间

ContrastLevel=0x28;         // 设置对比度初始值为 0x28

SdCmd(0x26);                // 设置占空比 1/128，温度系数设置 0.1%/C
```

```
SdCmd(0x40);                // 设置显示起始行=0
SdCmd(0x89);                // 设置页/列地址自动加 1 功能
SdCmd(0xAF);                // 开显示
SdCmd(0x90);                // 固定区域行数=0
SdCmd(0xC4);                // 设置水平映象为逆序，垂直映象为正序，数据排序为 D0-D7
SdCmd(0xEA);                // 设置占空比为 1/12
SdCmd(0x2D);                // 设置内部 LCD 电源
SdCmd(0x81);                // 设置对比度
SdCmd(ContrastLevel);       // 对比度初始值
}
```

2、清屏函数

清屏程序利用了初始化设置的列/页地址自动加 1 功能，所以全 RAM 区域写入时，只需要设置一次地址即可。

```
void ClearRAM()
{
  uchar i,j;
      SdCmd(0xb0);              // 页地址设置
      SdCmd(0x00);              // 列地址低 4 位
      SdCmd(0x10);              // 列地址高 4 位

      for (i=0;i<16;i++)        // 循环写 16 页
      {
      for(j=0;j<240;j++)        // 循环写 240 单元
        {
          SdData(0);            // 数据设置为 0
        }
      }
}
```

程序说明：

初始化函数设置了模块运行的基本设置，显示状态为开显示。这样当运行初始化程序后，在模块显示屏上应该能看到有一定对比度的稳定显示的花屏，这是因为在初始化中没有对显示 RAM 进行清“0”，所以在屏幕上显示出来的都是显示 RAM 在上电时的随机数。这是正常的。由此，我们可以把初始化程序作为接口的调试程序，如果没有出现上述现象，则需要重新检查电路，或将初始化程序中的对比度参数再向大设置，在运行观察。

第三章　液晶显示控制器指令系统

UC1608 的指令系统比较简单。内部的显示 RAM 结构见图七所示，RAM 分 16 页，每页 240 个单元，每个单元的显示数据对应的显示屏像素是某一列的 8 点行像素，而且是数据的最低位 DB0 位对应本页中最上一行的像素。

图七　UC1608 的 RAM 的地址结构

指令描述如下：

显示开关指令　　**指令代码：AEH / AFH**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1	1	1	1/0

功能描述：设置显示开关。指令 AEH 为关显示，模块进入 SLEEP 模式。所有驱动器输出、电压发生器和时序电路都将终止电源。指令 AFH 为开显示，退出 SLEEP 模式，进入正常显示。

设置所有像素点亮　　**指令代码：A4H / A5H**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	1	0	0/1

功能描述：该指令强迫所有列驱动输出为 ON 信号，实现全屏点亮。该指令不影响 RAM 内容。指令码 A4H 为正常显示，指令码 A5H 为全屏点亮。

正/负向显示设置　　**指令代码：A6H / A7H**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	1	1	0/1

功能描述：强制所有列驱动取反数据输出，实现全屏反显效果。不影响 RAM 数据。指令 A6H 为正常显示，A7H 为全屏反显效果。

显示起始行设置　　**指令代码：40H ~7FH**

D7	D6	D5	D4	D3	D2	D1	D0
0	1	S5	S4	S3	S2	S1	S0

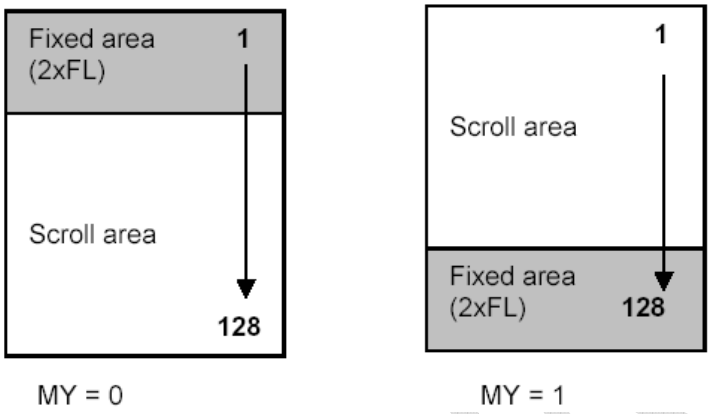
功能描述：该指令设置了模块的显示起始行号，该起始行号是把固定区域与卷动区域的边缘行作为计算基点，顺序计算出的第 S[5:0]行。如果固定区域行数为 0，则 S[5:0]即为显示屏的最顶行。S[5:0]取值为 0-3FH，对应显示行数为 0-63 行。如果定时间隔等量修改该设置，将会出现显示向上或向下卷动或局部卷动的效果。

设置固定行 **指令代码：90H~9FH**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	FL3	FL2	FL1	FL0

功能描述：设置固定区域的行数，应用于局部卷动功能。模块将显示区分为固定区域和卷动区域，该指令将定义固定的区域的宽度（行数），该区域在卷动功能下将不受卷动功能的影响。固定行区域为在 MY=0 时从顶行起 2*FL 行的区域或在 MY=1 时从底行起 2*FL 行的区域。见图八所示。

FL[3:0]：固定行数，0-15。



图八 显示区域分固定区域和卷动区域

RAM 页地址设置 **指令代码：B0H~BFH**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	P3	P2	P1	P0

功能描述：该指令设置了显示 RAM 的页地址。P[3:0]取值为 0-15，对应的显示 RAM 页为 0-15 页。拓普微产品 LM240120A 的页地址为 0~14，第 15 页将不被使用。

RAM 列地址设置 **指令代码：00H~0FH + 10H~1EH**

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	C3	C2	C1	C0
0	0	0	1	C7	C6	C5	C4

功能描述：该指令设置了显示 RAM 的列地址，指令为双字节格式，需要连续写入。列地址是为 1 个字节宽度，在设置时被分解为两部分，指令的第一字节低 4 位为列地址低 4 位，第二字节低 4 位为列地址高 4 位。C[7:0]取值为 0-239。

在 MPU 读、写显示 RAM 操作后列地址自动加 1，以实现 MPU 连续对显示 RAM 的操作。达到最大列地址后，将根据 **RAM 地址控制设置**指令停止修改或者归 0，从头开始。

RAM 地址控制设置 **指令代码：88H、89H、8CH、8DH**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	AC2	AC1	AC0

功能描述：列地址增加模式固定为+1；当列地址增加到 239 时，下一个读写 RAM 的操作，列地址或返回到列地址 0，或驻留在地址 239 不变。页地址范围为 0 页到 15 页，当列地址增加到右边缘时，下一个读写 RAM 操作，页地址可以是加 1 方式或减 1 方式或不变。该指令将设置了 RAM 地址的增量控制。

AC2：页地址自动增加方向。AC2=0，页地址加 1；AC2=1，页地址减 1；

AC1：保留位，设置为 0；

AC0：列/页地址自动增量，AC0=0 将禁止地址增量，即在列地址达到最大值时 RAM 地址（列、页）将驻留不变；当 AC0=1 时，在列地址达到最大值时，列地址将归 0 位和页地址根据 AC2 设置而修改。

拓普微产品 LM240120A 初始设置为 89H，页地址自动加 1，自动加 1 功能。

设置 LCD 映象控制　　指令代码：CXH

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	MY	MX	0	MSF

功能描述：设置显示 RAM 与驱动输出的对应关系。

MY：选择 RAM 行地址与驱动输出 COM 电极之间的对应关系。不影响 RAM 数据。当 MY 设置为 0 时，RAM 行地址与行驱动输出是正序排序关系，即 RAM 行地址 0 对应 COM0；当 MY 设置为 1 时，RAM 行地址与行驱动输出是逆序排序关系，即 RAM 行地址 0 对应 COM127。该设置一经写入，驱动立刻实现转换，改变当前的图象显示效果。

MX：选择 RAM 列地址与列驱动输出 SEG 的对应关系。当 MX 设置为 0 时，RAM 列地址与列驱动输出是正序排序关系，即 RAM 列地址 0 对应 SEG0 输出；当 MX 设置为 1 时，RAM 列地址与列驱动输出为逆序排序关系，即 RAM 列地址 0 对应 SEG239。该设置写入后，当前已写入到 RAM 的内容不改变显示排序，在之后的写入数据将按照新的设置方式写入与显示，即影响到设置后的 RAM 读/写操作。

MSF：设置数据位与驱动输出的排序关系。当 MSF 设置为 0 时，在页地址中，数据线自上而下排序 D0-D7；当 MSF 设置为 1 时，数据线自上而下排序为 D7-D0。该设置写入后，当前已写入到 RAM 的内容不改变显示排序，在之后的写入数据将按照新的设置方式写入与显示，即影响到设置后的 RAM 读/写操作。

拓普微产品 LM240120A 初始设置为 C4H，MY=0，MX=1，MSF=0。

LCD 偏压设置　　指令代码：E8H ~EBH

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	1	0	BR1	BS0

功能描述：该指令选择了 LCD 驱动的电压偏压比。拓普微 LM240120A 设置为 EAH，为 1/12 偏压比。

BR[1:0]：00（1/10）；01(1/11)；**10（1/12）**；11（1/13）。

拓普微产品 LM240120A 初始设置为 EAH。

设置驱动占空比和温度补偿　指令代码：20H~27H

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	MR	TC1	TC0

功能描述：该指令设置了显示的占空比和温度补偿系数。

MR：占空比系数设置，1 为 96 行扫描；1 为 128 行扫描；

TC[1;0]：设置温度补偿曲线斜率

00：-0.00%/C；01：-0.05%/C；10：-0.10%/C；11：-0.20%/C

拓普微产品 LM240120A 初始设置为 26H。

电源控制　　指令代码：28H~2FH

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1	PC2	PC1	PC0

功能描述：设置 VLCD 和 LCD 负载的选择。

PC2：选择 VLCD 的来源；0 选自外部 VLCD；1 使用内部 VLCD。

PC[1:0]：根据 LCD 屏的电容负载选取下面设置：

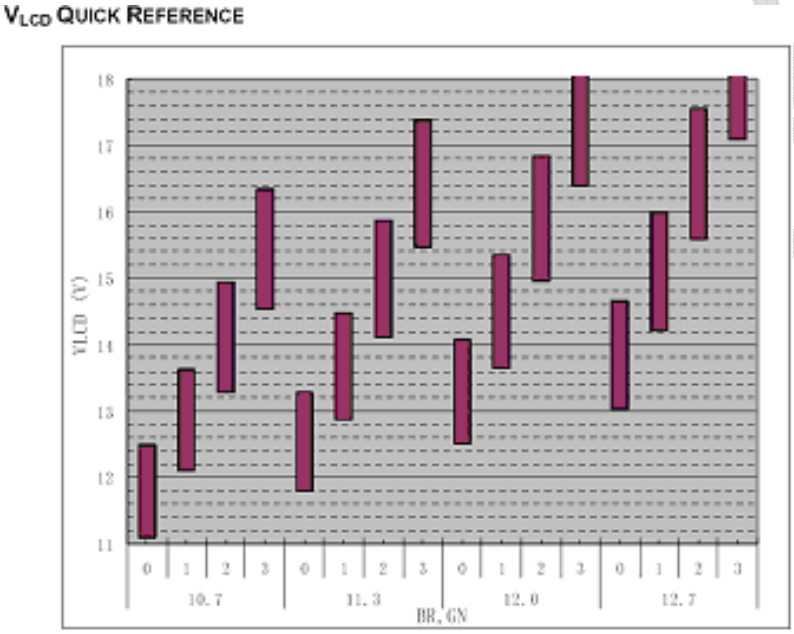
00：CLCD<26nf；01：**26nf<CLCD<43nF**；10：43nf<CLCD<60nf；11：60nf<CLCD<90nf

拓普微产品 LM240120 初始设置为 2DH，其中 PC2=1，启用内部 VLCD，CLCD=00，26nf。

对比度设置 **指令代码： 81H + 00H~FFH**

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	1
GN1	GN0	PM5	PM4	PM3	PM2	PM1	PM0

功能描述：该指令为双字节指令，设置 LCD 驱动电压 VBIAS 和 VLCD。GN[3:0]参数为 VLCD 的粗调设置，PM[5:0]为 VLCD 的细调设置。它们与偏压比 BR 参数一起，确定了 VLCD 的输出范围。见图九所示。初始化时为了调试电路，可以将该值可以设置大些，便于观察显示效果。



图九 GN、PM、BR 设置与 VLCD 电压关系

GN[3:0]：取值 0~3。设置 VLCD4 个电压档。

PM[5:0]：取值 0~63。在 GN 确定的电压档内等值调节 VLCD 电压。

拓普微 LM240120A 初始对比度设置为 28H（ BR 为 1/128）

修改写功能设置 **指令代码：EFH**

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	1	1	1	1

修改写结束指令 **指令代码：EEH**

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	1	1	1	0

功能描述：该组指令完成修改写操作功能。修改写功能码 EFH 与修改写结束指令 EEH 成对出现。在写入指令码 EFH 后，所有的读取 RAM 数据的操作都不对列地址进行加 1 修正，只有向显示 RAM 写数据后才进行列地址的加 1 操作。这种功能将给绘图操作带来极大的方便，因为绘图需要在原图案的基础上增加相应的图形，所以需要先读出单元的数据，与新图形合成后再写回原单元。当完成修改写操作后，需要写入结束指令 EEH，将列地址恢复到原进入修改写操作时的地址。

写数据操作

D7	D6	D5	D4	D3	D2	D1	D0
显示数据							

功能描述：

写数据操作是向数据通道（D/C=1）直接写入数据，数据被写到当前由页地址和列地址所指定的显示 RAM 单元内。写操作完成后，列地址和页地址将根据设置自动修改。MPU 可以连续向显示 RAM 写入数据。

读数据操作

D7	D6	D5	D4	D3	D2	D1	D0
显示数据							

功能描述：

读数据操作是从数据通道（D/C=1）的输出寄存器中读出显示数据，同时当前由页地址和列地址指向的显示 RAM 单元的数据传送到这个寄存器内。因此在设置新地址后第一次读数据时，需要有一次空读操作，以便将当前单元的数据传送到接口的输出寄存器内。读操作完成后，列地址和页地址将根据设置自动修改。MPU 可以连续从 RAM 读出数据。

软件复位指令　　指令代码：E2H

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	0	1	0

功能描述：该指令码 E2H 是模块的软件复位，该复位将花费 15ms 时间。它将完成指令寄存器的初始化操作，其作用同接口的/RESET 脚功能。显示 RAM 单元内容不受影响。当系统对模块的/RESET 硬件复位初始化后，该指令复位可以不做。

空操作指令

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	0	1	1

功能描述：该指令不运行任何功能，其作用为调节内部运行时序。

读状态字

D7	D6	D5	D4	D3	D2	D1	D0
BZ	MX	DE	RS	WA	GN1	GN0	1

功能描述：该寄存器为状态字寄存器，为 MPU 提供模块的内部运行信息。其中：

BZ：忙标志位。BUSY=1 表示模块内部正在处理指令码或在复位状态，此时不接受 MPU 发送的指令和数据。BUSY=0 表示芯片可以接收 MPU 发送的新数据。

MX：表示显示 RAM 的列地址与列驱动输出的对应关系。当 MX=0 时，表示同顺序对应，即列地址序号与列驱动输出序号相同，列地址 0-239 对应列驱动输出 0-239；MX=1 时，表示反顺序对应，即列地址序号与列驱动输出序号反方向对应，列地址 0-239 对应列驱动输出 239-0。

DE：表示显示开关状态，DE=0 表示显示开状态，DE=1 表示显示关状态。

RS：表示模块运行状态。RS=0 为芯片在正常运行状态中，RS=1 表示芯片处于复位状态，该复位状态字将指示硬件/RES 信号或软件复位指令的状态。

WA：列、页地址自动修改状态，0 无效，1 有效。

GN[1:0]：VBIAS 和 VLCD 的设置状态

第四章 液晶显示控制器应用函数

我们提供的功能子程序完全使用显示画面上的坐标（X，Y）为显示数据读写操作位置，在程序中将计算出实际读写的 RAM 单元地址，因此模块使用者可以不必考虑实际的 RAM 地址。

一、对比度调节程序

1、对比度增强子程序

```
void LCD_Darker(void)
// 对比度参数 ConTrastLevel 初始值在初始化程序中设置
{
    if (ContrastLevel<0xff)       // 限制上限值
        {
        ContrastLevel++;          // 对比度参数加 1
        }
    SdCmd(0x81);                  // 对比度设置指令代码
    SdCmd(ContrastLevel);         // 写入对比度值
}
```

2、对比度减弱子程序

```
void LCD_Lighter(void)
// 对比度参数 ConTrastLevel 初始值在初始化程序中设置
{
    if (ContrastLevel>0x00)       // 限制下限值
        {
         ContrastLevel--;         // 对比度参数减 1
        }
    SdCmd(0x81);                  // 对比度设置指令代码
    SdCmd(ContrastLevel);         // 写入对比度值
}
```

二、字符写入程序

3、ASCII 字符串写入程序

```
void PrintASCII(uint x,y,uchar *pstr)
 // 坐标(x,y)，x 为水平方向像素列；y 为垂直方向字符行（8 点像素/行）
{
  uchar j;
  uint addr;
  SdCmd(y|0xb0);                  // 设置页地址
  SdCmd(x&0x0f);                  // 设置列地址低 4 位
  SdCmd((x>>4)|0x10);             // 设置列地址高 4 位

  while(*pstr>0)
  {
  addr=*pstr++;                   // 取字符代码
  addr=(addr-0x20)*8;             // 计算字符字模起始地址
  for (j=0;j<6;j++)               // 设置循环量，显示 6*8 点阵字符
    {                             //深圳市拓普微科技开发有限公司制作
    SdData(ASCIITAB[addr+j]);   //   写字模数据
    }
  }

}
```

4、汉字写入子程序

```
void PrintGB(uchar x,y,uchar *pstr)
// 坐标(x,y)，x 为水平方向像素列；y 为垂直方向字符行（8 点像素/行）
{
 uint addr;
 uchar j,n;
 while(*pstr>0)
```

```
    {
    addr=*pstr++;                          // 取汉字代码
    addr=(addr-1)*32;                      // 计算汉字字模起始地址
     for (n=0;n<2;n++)
       {
       SdCmd(y|0xb0);                      // 设置页地址
       SdCmd(x&0x0f);                      // 设置列地址低 4 位
       SdCmd((x>>4)|0x10);                 // 设置列地址高 4 位
                                           //深圳市拓普微科技开发有限公司制作
       for (j=0;j<16;j++)                  // 写 16 字节字模数据
         {
         SdData(CCTAB[addr+j+16*n]);       // 写字模数据
         }
       y=y+1;
       }                                   // 页地址加 1
     y=y-2;                                // 页地址修正原值
     x=x+16;                               // 列地址修正下一个汉字位置
     }
}
```

三、绘图程序

5 绘点程序

```
void Draw_Dot(uint x,y)
// 坐标(x,y)，x 为水平方向像素列；y 为垂直方向页（8 点像素/页）
{
   uchar k,m;
       k=y/8;
       SdCmd(k|0xb0);              // 设置页地址
       SdCmd(x&0x0f);              // 设置列地址低 4 位
       SdCmd((x>>4)|0x10);         // 设置列地址高 4 位

       SdCmd(0xef);                // 设置修改写模式
       y=y%8;
       m=1;
       y=m<<y;
       m=RdData();                 // 空读操作
       m=RdData();                 // 读数据
       SdCmd(0xe3);                // 空操作
       m=m|y;
       SdData(m);                  // 写数据
       SdCmd(0xee);                // 退出修改写模式
}
```

6 绘线程序

```
void Draw_Line(uint x1,y1,x2,y2)
//   x 为水平方向像素列；y 为垂直方向页（8 点像素/页）
//   坐标(x1,y1)为线起始地址坐标；坐标(x2,y2)为线终止地址坐标。
{
   uint temp;
   int dalt_x,dalt_y,err=0;
  if (y1>y2)
      {
      temp=x1;
      x1=x2;
      x2=temp;
      temp=y1;
      y1=y2;
      y2=temp;
      }
  Draw_Dot(x1,y1);
  dalt_x=x2-x1;
  dalt_y=y2-y1;
  if(dalt_x>=0)
```

```
        {
          if(dalt_y>dalt_x)//k>1
            {
              while(y1<y2)
              {
                if(err<0)
                {
                  x1=x1+1;
                  y1=y1+1;
                  err=err+dalt_y-dalt_x;
                  }
                  else
                  {
                  y1=y1+1;
                  err=err-dalt_x;
                  }
                 Draw_Dot(x1,y1);
              }
            }
          else   // 0<=k=<1
           {
            if (dalt_y==0)
                y1=y1-1;
            while(x1<x2)
              {
                if(err<0)
                {
                  x1=x1+1;
                  err=err+dalt_y;
                  }
                  else
                  {
                  y1=y1+1;
                  x1=x1+1;
                  err=err+dalt_y-dalt_x;
                  }
                 Draw_Dot(x1,y1);
              }
          }
      }
    else
      {
        dalt_x=x1-x2;
         if(dalt_y>dalt_x)//k<-1
          {
            while(y1<y2)
             {
               if(err<0)
                {
                  x1=x1-1;
                  y1=y1+1;
                  err=err+dalt_y-dalt_x;
                }
               else
                {
                  y1=y1+1;
                  err=err-dalt_x;
                }
               Draw_Dot(x1,y1);
            }
          }
         else           //0>k>=-1
          {
            if (dalt_y==0)
                y1=y1-1;
           while(x1>x2)
             {
```

```
            if(err<0)
             {
               x1=x1-1;
               err=err+dalt_y;
             }
            else
             {
               x1=x1-1;
               y1=y1+1;
               err=err+dalt_y-dalt_x;
             }
            Draw_Dot(x1,y1);
          }
        }
      }
    }
```

7、图画写入子程序

```
void ShowBMP(uchar x,y,width,high,uchar *bmp)
// 坐标(x,y)，x 为水平方向像素列；y 为垂直方向页（8 点像素/页）
//width：图形水平像素点数；high：图形垂直页数；bmp[]：图形数组名
{
  uchar i,j;
  for(i=0;i<high;i++)
    {
     SdCmd(y|0xb0);              // 设置页地址
     SdCmd(x&0x0f);              // 设置列地址低 4 位
     SdCmd((x>>4)|0x10);         // 设置列地址高 4 位
     for(j=0;j<width;j++)
         {
         SdData(*bmp++);         // 写图形数据
         }
     y=y+1;                      // 页地址修正
     }
}
```

深圳市拓普微科技开发有限公司制作