拓普微内置 ST7529 液晶显示模块系列
应用手册

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	2007-10-22	郭强
0.2	更正 P8 第一行 JP10\JP11 为 JP8、JP9	2008-6-3	郭强
0.3	补充产品型号，文字修改，程序修改	2009-1-8	郭强
0.4	在初始化函数中增加读 EEROM 操作及相关指令说明	2009-4-22	郭强

目 录

应用指南

欢迎您进入拓普微模块应用指南流程

首先请您翻到本手册的第一章，了解液晶显示模块的使用注意事项。尤其模块带 FPC 接口线的保护

核对产品型号

在第二章拓普微产品型号一览表中核对您手中模块的型号。如果不在，请您申请适合您模块的应用手册。

OK。 我们现在开始调试这个模块。调试的目标是：

硬件接口电路 OK

驱动子程序 OK

步骤一：硬件连接

A 请翻到第三章，了解模块的接口特性：

PINS 定义、操作的时序关系

B 连接模块与计算机系统。

请参考第四章推荐的接口电路。

C 检查连接线，尤其是电源线和信号线。

如果计算机操作时序快于或接近模块的接口时序要求，建议使用 I/O 寻址电路

步骤二：编写驱动子程序

驱动子程序是计算机与模块对话的基础；

驱动子程序包括有：

指令写入、显示数据写入等函数

第四章提供了驱动函数的样板，

在引用时，请注意选通地址码和信号线的设定

步骤三：调试

运行第四章的初始化函数，观察显示效果

如果显示稳定的花屏，则表示接口电路和驱动程序 OK。

否则请参照第六章提示，继续调试。

恭喜您，完成了接口的调试工作。

下面您可以编制自己需要的显示程序了

请参考第五章提供了各指令的描述

请参考第七章推荐了几个功能函数

我们的技术服务电话：0755-81795805

网址：http//www.topwaydisplay.com

第一章　液晶显示模块使用须知

液晶显示模块由易碎的玻璃盒、易划痕的偏光片和聚集集成电路的 PCB/FPC 板以及背光板等组成，模块属于精密器件，虽然拓普微在产品出厂时已经做了各项可能的保护，但在您使用之前还是希望能仔细阅读以下的注意事项，以免给您造成不必要的损失。

一、 处理保护膜

在模块显示屏表面上贴有一层保护膜，以防止在调试、装配过程中沾污显示屏表面，在整机装配结束前不得揭去。

在剥离保护膜时，有时会产生静电，引起显示屏不正常的显示，这是正常的，模块在短时间上可以自行恢复。

二、 加装衬垫

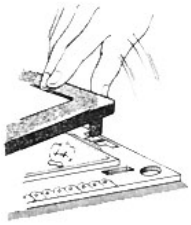
在整机装配时，建议在模块与前面板之间加装约 0.1 毫米厚的衬垫。面板与模块的接触面应保持平整，以免在装配后产生扭曲，并可提高其抗振性能。

三、 严防静电

模块中的控制、驱动电路是低压、微功耗的 CMOS 电路，极易被静电击穿，静电击穿是一种不可修复的损坏，而人体有时会产生高达几十伏或上百伏的高压静电，尤其在干燥的环境中，所以，在操作、装配以及使用中都应极其小心，严防静电。为此：

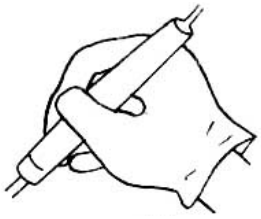
1. 不要用手随意触摸外引线、电路板上的电路及金属框；
2. 如必须直接接触时，应使人体与模块保持在同一电位，或将人体良好接地；
3. 焊接使用的烙铁和操作用的电动工具必须良好接地，没有漏电；
4. 不得使用真空吸尘器进行清洁处理，因为它会产生很强的静电；
5. 空气干燥，也会产生静电，因此，工作间湿度应在 RH60%左右；
6. 取出或放回包装袋或移动位置时，也需小心，防止产生静电。不要随意更换包装或舍弃原包装。

四、 装配操作时的注意事项



1. 模块是经过精心设计组装而成的，请勿随意自行再加工、修整；
2. FPC 虽有柔韧性，但切记不能硬折，否则将导致 FPC 电路折断；
3. 模块通过 FPC 与系统板连接时，要使用专用插座；
4. 金属框爪不得随意扭动、拆卸；
5. 不要随意修改加工 PCB 板外形、装配孔、线路及其部件；
6. 不得修改任何内部支架；
7. 不要碰、摔、折曲、扭动模块。

五、 焊接



如需要焊接模块 FPC 接口时，应按如下规程进行操作。

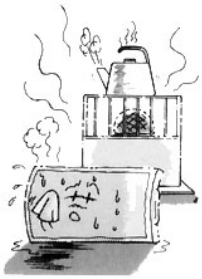
1. 烙铁头一定要平滑，避免 FPC 焊盘出现硬划伤；
2. 烙铁头温度小于 320°C（无铅）280°C（有铅）；
3. 焊接时间小于 3~4s；
4. 不要使用酸性助焊剂；
5. 重复焊接不要超过 3 次，且每次重复需间隔 5 分钟。

六、 模块的使用

1. 模块的外引线决不允许接错，在您想调试液晶模块时，请注意正确接线，尤其是正、负电源的接线不能有错，否则可能造成过流、过压、烧毁电路上的芯片等对液晶模块元器件有损的现象；
2. 模块在使用时，接入电源及断开电源，必须在正电源稳定接入以后，才能输入信号电平。不要在未接通电源或接入的电源电压未稳定时，在信号线上施加电压，这样有可能损坏模块中的 IC 及电路；
3. 因为液晶材料的物理特性，液晶的对比度会随着温度的变化而相应变化，所以，拓普微的产品都在接口 VOUT 端将负电源引出，提供给系统作为对比度的调节。您可做一个温度补偿电路，或者简单的安排一个电位器，然后返回到接口 V0 端；
4. 不应在规定工作温度范围以外使用，不应在存储极限温度范围外存储，如果模块的环境温度低于液晶材料的结晶温度，液晶体就会结晶，相反，如果温度过高，液晶材料将变成各向同性的液体，破坏了分子取向，这两种现象都将使模块丧失显示功能；
5. 显示屏受到轻微压力时，会产生异常显示。这时切断电源，稍待片刻，重新上电，即恢复正常；
6. 液晶显示器件或模块表面结雾时，不要通电工作，因为这将引起电极化学反应，产生断线；
7. COG 或 TAB 形式的 IC 对光比较敏感，在强光环境下，可能会造成 IC 的特性下降，甚至出现损坏。

七、 模块的保养与存储

1. 只能使用异丙基酒精或乙荃酒精清洁模块，其他溶剂（比如水）都有可能损坏模块。
2. 若长期（如几年以上）存储，我们推荐以下方式：



1. 装入聚乙稀口袋（最好有防静电涂层）并将口封住；
2. 在-10°C~ +35°C 之间存储；
3. 放暗处，避强光；
4. 决不能在表面压放任何物品；
5. 严格避免在极限温/湿度条件下存放。

第二章 液晶显示模块产品汇总

深圳市拓普微科技开发有限公司出品的内置 ST7529 的液晶显示模块系列如下表所列。该类型模块的特点是，驱动/控制器为 COG 封装，电路集成度高，体积小；内置 LCD 驱动电源，软件调节对比度；可以实现 32 级灰度显示，每个像素对应 5 位数据，**推荐使用于单色显示；**接口形式多种选择，并行接口适配 INTEL8080 时序，串行接口可选择 3 线或 4 线，通讯规则简单；低电压 3.3V 工作，低功耗，适合于各种仪器仪表上使用。

请您在您所使用的型号栏上作出标记。

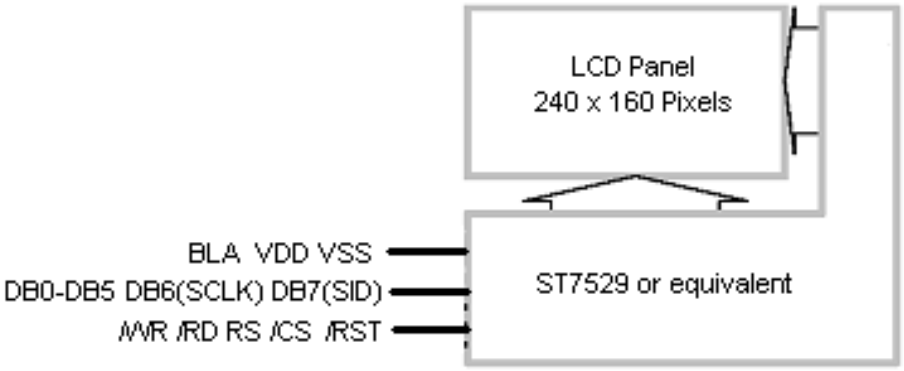
深圳市拓普微科技开发有限公司内置 ST7529 液晶显示模块系列一览表

型 号	点阵数	逻辑电压	背光电压	接口特性（出厂设置值）
LM240160D 系列	240*160	3.3V	3.3V	并口 (80mode)
LM240160G 系列	240*160	3.3V	3.3V	并口 (80mode)
LM160128AFW-1	160*128	3.3V	3.3V	并口 (80mode)

模块接口引脚排序如下表

PIN	LM240160D / G 系列	LM160128AFW-1
1	VDD	VDD
2	VSS	VSS
3	/CS	/CS
4	RS	RS
5	/WR	/WR
6	/RD	/RD
7	DB0	DB0
8	DB1	DB1
9	DB2	DB2
10	DB3	DB3
11	DB4	DB4
12	DB5	DB5
13	DB6(SCLK)	DB6(SCLK)
14	DB7(SI)	DB7(SI)
15	/RST	/RST
16	BLA	BLK

第三章　液晶显示模块的接口特性



图一 内置 ST7529 液晶显示模块系统的方框图

图一例举了深圳市拓普微科技开发有限公司制作的内置 ST7529 液晶显示模块的电路原理框图，模块为 COG 产品，如 LM240160 系列带有背光板和 PCB 接口板，与 MPU 接口为 FPC 形式接口，需要专用插座连接。

深圳市拓普微科技开发有限公司出品的液晶显示模块与 MPU 接口的管脚定义是一致的，但各模块的引脚排序略有差异，管脚定义见下表。

一、与 MPU 接口定义表

管脚符号	状态	管脚定义
/CS	I	片选信号，/CS=0 时选通模块；/CS=1 时模块接口被封锁
/RST	I	复位信号，低电平复位；正常运行时，为高电平状态
RS	I	通道选择信号，当 RS=0 时，选择指令通道；RS=1 时，选择数据通道
WR	I	当使用并行接口时，为写信号/WR，低电平有效；
/RD	I	当使用并行接口时，为读信号/RD，低电平有效；
DB0	I/O/Z	并行接口数据总线
DB1	I/O/Z	并行接口数据总线
DB2	I/O/Z	并行接口数据总线
DB3	I/O/Z	并行接口数据总线
DB4	I/O/Z	并行接口数据总线
DB5	I/O/Z	并行接口数据总线
DB6(SCLK)	I/O/Z	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
DB7(SI)	I/O/Z	并行接口数据总线，当选择串行接口时，为串行数据输入端
VDD		电源正
VSS		电源地
BLA		背光电源正极
BLK		背光电源负极

二、模块接口选择跳线表

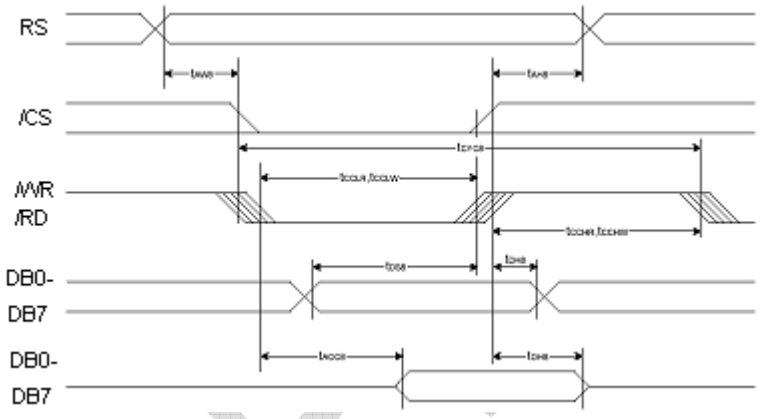
模块为了使用的便利，提供了多种接口的选择，它们通过跳线 JP1-6、JP8、JP9 组合设置。

接口形式	JP1	JP2	JP3	JP4	JP5	JP6	JP8	JP9	备注
并行接口 8080	**CLOSE**	**OPEN**	**CLOSE**	**OPEN**	**OPEN**	**CLOSE**	**OPEN**	**OPEN**	**出品值**
并行接口 6800	OPEN	CLOSE	CLOSE	OPEN	CLOSE	OPEN	OPEN	OPEN	
串行接口 3 线 9 bit	OPEN	CLOSE	OPEN	CLOSE	CLOSE	OPEN	CLOSE	CLOSE	
串行接口 4 线 8bit	OPEN	CLOSE	OPEN	CLOSE	OPEN	CLOSE	CLOSE	CLOSE	

三、与 MPU 的接口时序关系

1、并行接口的时序关系（INTEL8080 时序）

产品出品设置值为并行接口 INTEL8080 时序关系，我们在并行接口上也推荐这种形式。时序信号为 /RD、/WR、/CS、RS 及 8 位数据总线 DB0-DB7 组成。它们之间的时序关系如下：



图二、并行接口时序图（适配 INTEL8080 时序）

时序表　　VDD=3.3/2.7 Ta=-30~85C

参数	描述	信号	最小值	最大值	单位
t_{AH8}	地址保持时间	A0	20/20		ns
t_{AW8}	地址建立时间		20/30		
t_{CYC8}	系统时钟时间		200/250		
t_{CCLW}	写低脉冲宽度	WR	100/150		
t_{CCHW}	写高脉冲宽度		100/100		
t_{CCLR}	读低脉冲宽度	RD	100/150		
t_{CCHR}	读高脉冲宽度		100/100		
t_{DS8}	写数据建立时间	D0-D7	150/200		
t_{DH8}	写地址保持时间		20/20		
t_{ACC8}	读取数据时间			40/40*	
t_{OH8}	读输出无效时间			30/30*	

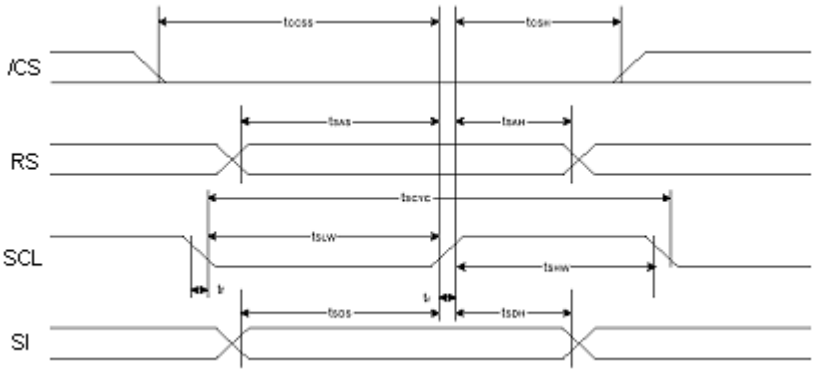
说明：规定输入信号的上升时间 tr 和下降时间 tf 小等于 15ns

所有时序关系在给定的 VDD 的 20%~80%范围内有效。

*项测试条件为 CL=100pf.

2、串行接口的时序关系

模块的串行接口有 3 线制与 4 线制两种，其区别在于 RS 信号上。4 线制串行接口，有 RS 信号，指令和数据通道由 RS 信号选择，如同并行接口；3 线制没有 RS 信号，需要在通讯时的第一数据认定为 RS 信号，随后 8 个数据为指令或数据，但时序关系是一致的，可以用下面的时序图和时序表反映两种接口形式的时序关系。它们的时序关系如下：



图三、串行接口时序图（3 线制串行接口没有 RS 信号）

时序表　　VDD=3.3/2.7 Ta=-30~85C

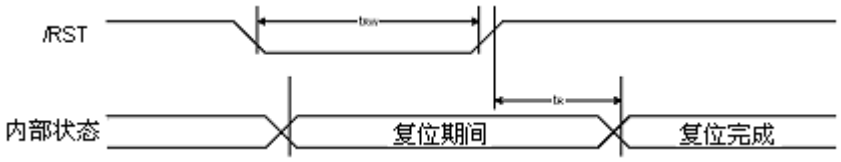
参数	描述	信号	最小值	最大值	单位
t_{SCYC}	串行时钟周期	SCL	100/110		ns
t_{SGW}	SCL“高”脉宽		50/60		
t_{SLW}	SCL“低”脉宽		50/50		
t_{SAS}	地址建立时间	RS	40/40		
t_{SAH}	地址保持时间		30/40		
t_{SDS}	数据建立时间	SI	30/40		
t_{SDH}	数据保持时间		30/40		
t_{CSS}	CS-SCL 时间	XCS	20/30		
t_{SCH}	SCL-CS 时间		50/60		

说明：规定输入信号的上升时间 tr 和下降时间 tf 小等于 15ns

所有时序关系在给定的 VDD 的 20%~80%范围内有效。

去掉 A0 的时序关系，即 3 线串行接口时序关系。

3、复位信号的时序关系



图四、/RST 复位信号时序图

时序表　　VDD=3.3V/2.7V Ta=-40~85C

参数	描述	信号	最小值	最大值	单位
t_R	复位时间			1/ 1.5	us
t_{RW}	复位低电平时间	RST	1/1.5		

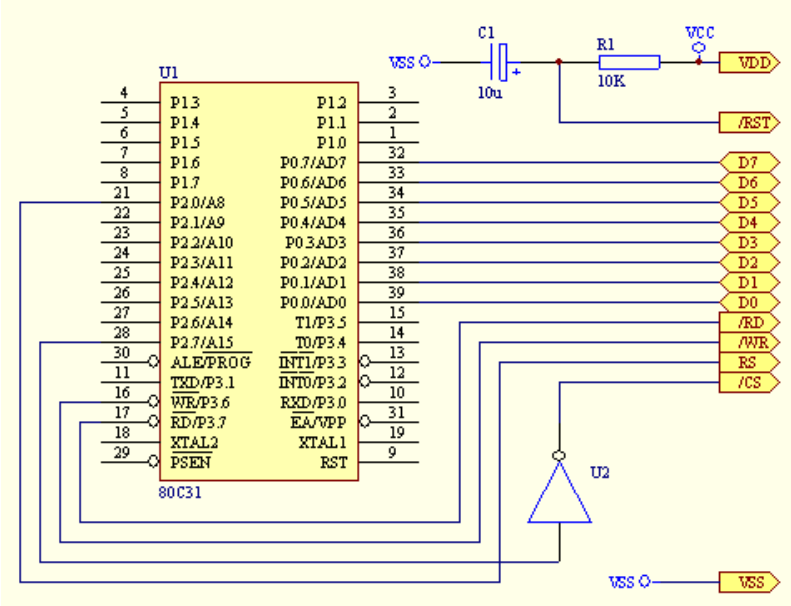
在实际应用时，我们大致需要在复位信号结束后人为延迟 800ms 以上，以保证模块能够正确的接收 MPU 的操作和传过来的数据。

第四章　液晶显示模块接口技术

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的内置 ST7529 液晶显示模块系列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。但我们还是推荐，并行接口使用 INTEL8080 时序信号，串行接口使用 4 线制。本章将以单片机 89S52 为控制系统实例，提供在并行接口形式下的总线寻址方式的接口示意电路和驱动子程序、I/O 寻址方式的接口示意电路和驱动子程序、以及 4 线串行接口的应用方法。

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，如图五所示：



图五　　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，/RD、/WR 作为模块的读、写控制信号， /CS 信号和 RS 信号都由地址线译码产生，模块的/RST 可以接 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）

1、指令代码传送函数

```
void SdCmd(uchar Command)
{
    uchar xdata   *com_addr;
    com_addr = 0x8000;                  // 指令通道地址
    *com_addr = Command;               //写指令操作
}
```

2、显示数据传送函数

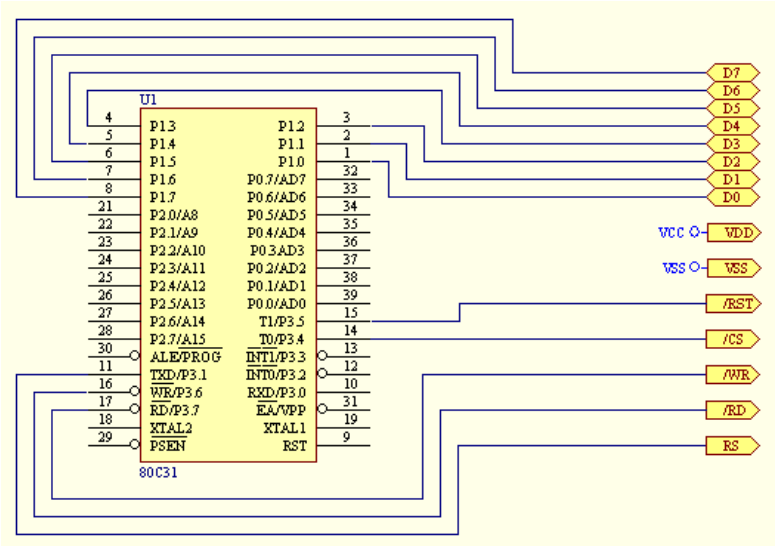
```
void SdData(uchar Ddata)
{
     uchar xdata   *data_addr;
     data_addr = 0x8100;               // 数据通道地址
    *data_addr = Ddata;                //写数据操作
}
```

3、显示数据读取函数

```
void RdData()
{
     uchar Ddata;
```

```
    uchar xdata   *data_addr;
    data_addr = 0x8100;              // 数据通道地址
   Ddata = *data_addr;               // 读数据操作
   return(Ddata);
}
```

二、 I/O 寻址方式接口电路及驱动程序



图六　I/O 寻址方式电路示意图

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，间接实现对模块的控制。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，在图六给出的示例中，我们将 89S52 的 P1 口作为数据总线。P3 口中 5 位端口为控制信号，它们是：P3.7 为/WR 信号，P3.6 为/RD 信号，P3.1 为 RS 信号，P3.4 为/CS 信号，P3.5 为/RES 信号。驱动程序的编程方法需要强调的是/WR 信号和/RD 信号的独立性，即它们的电平变化要求有一条指令独立操作，不希望与其它信号在同一个指令下一起操作。

I/O 寻址方式的驱动子程序如下：

```
//-----------------------------------
//   I/O 接口定义
//-----------------------------------
#define LCDBUS       P1
sbit _CS         = P3^4;
sbit _RST        = P3^5;
sbit RS          = P3^1;
sbit _WR         = P3^7;
sbit _RD         = P3^6;
//-----------------------------------
// 并行接口驱动程序
//-----------------------------------
1、指令代码传送函数
void SdCmd(uchar Command)
{
     RS = 0;                    // 选择指令通道
     LCDBUS = Command;          // 将指令送至数据接口
     _CS = 0;                   // 选择模块
     _WR = 0;                   // 写信号为低电平
     _WR = 1;                   // 写信号为高电平
     _CS = 1;                   // 封锁模块
}
2、显示数据传送函数
void SdData(uchar Ddata)
```

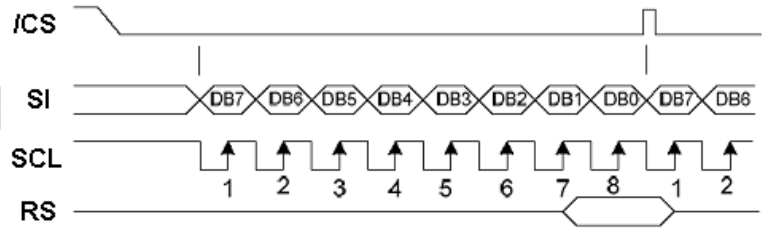
```
{
    RS = 1;                  // 选择数据通道
    LCDBUS = Ddata;          // 将指令送至数据接口
    _CS = 0;                 // 选择模块
    _WR = 0;                 // 写信号为低电平
    _WR = 1;                 // 写信号为高电平
    _CS = 1;                 // 封锁模块
}
```

3、显示数据读取函数

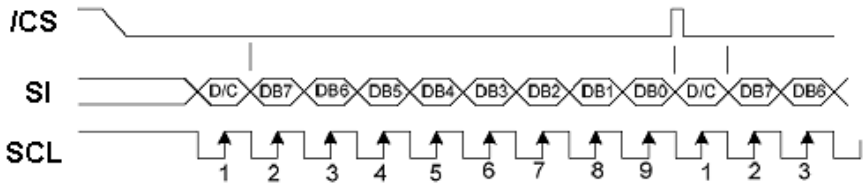
```
void RdData()
{
    uchar Ddata;
    LCDBUS = 0xff;
    RS = 1;                  // 选择数据通道
    _CS = 0;                 // 选择模块
    _RD = 0;                 // 读信号为低电平
    Ddata = LCDBUS;          // 将读数据
    _RD = 1;                 // 读信号为高电平
    _CS = 1;                 // 封锁模块
    return (Ddata);
}
```

三、 串行接口电路及驱动程序

串行接口是深圳拓普微科技开发有限公司该类模块系列所特有特性，在内置 ST7529 液晶显示模块系列的串行接口中我们推荐 4 线制串行接口形式，它需要 4 个信号线，SCL 为串行时钟信号，上升沿有效；SI 为串行数据端；RS 为通道的选择信号，RS=0 打开指令通道，RS=1 打开数据通道；/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，见图七 A 所示。在准备启用串行接口前，要将先将 SCL 信号设置为高电平，设置 RS 状态，然后再将/CS 置“0”，选通模块，SCL 的每个负脉冲的上升沿都将会把 SI 上数据送入内部缓冲器内，当第 8 个脉冲的上升沿时，SCLK 将 RS 状态读入内部控制位，并将已经读入 8 位数据并行送入 RS 所确定的通道寄存器内进行处理。/CS 还有一个作用是清空串行接口寄存器，所以在每个数据传输完成后，无论下面是否还有数据要传输，也要把/CS 置高，重新再开始下一个数据的传输。3 线制串行接口与 4 线制串行接口在硬件上少一个 RS 信号，所以在通讯格式上，增加了 RS 的选择数据位，即形成了 9 位数据传输格式，见图七 B 所示。

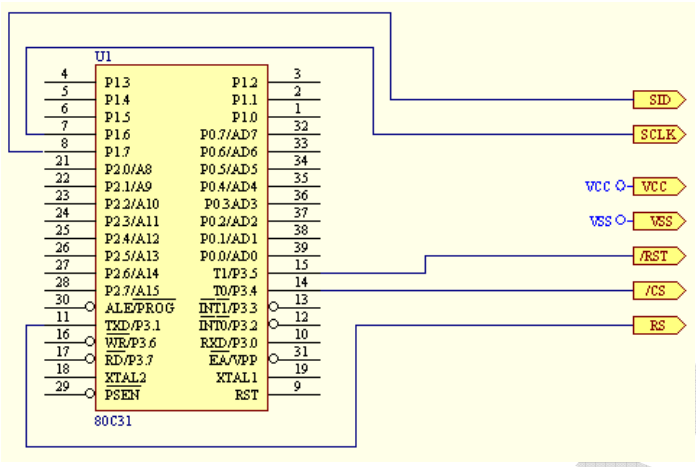


图七 A　4 线制串行接口通讯规则



图七 B　3 线制串行接口通讯规则

本章提供了串行接口电路示意图，见图八所示，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。



图八　串行接口电路示意图（3 线制接口没有 RS 端口连接）

串行接口驱动子程序如下：

```
//-----------------------------------
//   串行接口定义
//-----------------------------------
sbit _RST      = P3^5;
sbit _CS       = P3^4;
sbit RS        = P3^1;
sbit SID       = P1^7;
sbit SCLK      = P1^6;
uchar bdata transdata;        //该变量可为位操作之变量
sbit transbit   = transdata^7;
//-----------------------------------
// 串行接口驱动函数
//-----------------------------------
1、指令代码传送函数
void SdCmd(uchar Command)
{
    uchar j;
    transdata=Command;                  // 指令送位变量
    SCLK = 1;                           // 初始化 SCLK
     RS = 0;                            // 选择指令通道
    _CS = 0;                            // 选通模块
    for(j=0;j<8;j++)
      {
        SID=transbit;                   // 位变量 D7 位送数据口
        SCLK=0;                         // 产生移位脉冲
        SCLK=1;                         // 上升沿有效
        transdata=transdata<<1;         // 位变量数据左移一位
      }
    _CS=1;                              // 封锁模块
}
2、显示数据传送函数
void SdData(uchar Ddata)
{
    uchar j;
    transdata=Ddata;                    // 指令送位变量
    SCLK = 1;                           // 初始化 SCLK
    RS   = 1;                           // 选择数据通道
    _CS = 0;                            // 选通模块
    for(j=0;j<8;j++)
      {
```

```
        SID=transbit;                  // 位变量 D7 位送数据口
        SCLK=0;                        // 产生移位脉冲
        SCLK=1;                        // 上升沿有效
        transdata=transdata<<1;   // 位变量数据左移一位
      }
   _CS=1;                              // 封锁模块
}
```

在 3 线制的串行接口的驱动程序可以从上述的 4 线串行驱动程序中变形而来，将 RS 的状态直接设置给 SID 端口，再产生一个 SCLK 脉冲写入，如：

```
        SID=0;（指令通道）  或  SID=1;（数据通道）
        _CS=0;
        SCLK=0;
        SCLK=1;
```

然后再执行下面的传输程序。

四、初始化子程序

在内置 ST7529 液晶显示模块系列的初始化程序中，我们启用了内部 LCD 驱动电压，在硬件电路上，我们设置了 6X 倍压，V0 设置在 14V 多，为出厂检测值，用户可以根据使用环境再次设置。

程序设置数值以 LM240160DCW 为主

1、初始化函数

```
void initLCDM(void)
{
    _RST=0; delayms(2);         //硬件复位
    _RST=1; delayms(800);       //复位后延迟 800ms 以上时间
    SdCmd(0x30);                                        // 选择 ext0 指令集
    SdCmd(0x94);                                        // 退出休眠模式
    SdCmd(0xd1);                                        // 内部 OSC 运行
    SdCmd(0x20); SdData(0x08);                          // 启动内部升压电路，VB=1
    delayms(1);
    SdCmd(0x20); SdData(0x0b);                          // 启动内部 LCD 驱动电压电路 VB=VF=VR=1
    SdCmd(0x81); SdData(0x0f); SdData(0x04);      // 设置对比度值，适应 LM240160DCW
    // 设置显示扫描模式：CLD=0(clk/2); DT=39=160/4-1;FI=0;LF=0;
    SdCmd(0xca); SdData(0x04); SdData(0x27); SdData(0x00);//显示扫描设置
    SdCmd(0xa6);                                        // 正性显示
    SdCmd(0xbb); SdData(0x01);                          // 设置行扫描方向== 0-79,159-80
    //设置数据传输形式 LI=1;CI=0 ;C/L=0 ;CLR=0 ;GS=2 3b3p
    SdCmd(0xbc); SdData(0x01); SdData(0x00); SdData(0x02);//数据传输形式设置
    SdCmd(0x31);                                        // 选择 ext1 指令集
    //设置驱动电路参数  FR=62.5khz;Fbooster=3hz;bias=1/12
    SdCmd(0x32); SdData(0x05); SdData(0x00); SdData(0x02);//驱动电路设置
    //---------------        READ EEROM         ---------------
    SdCmd(0x30);                                  //   选择 ext0 指令集
    SdCmd(0x07); SdData(0x19);                    // 准备工作，使用补偿 ACK 信号功能
    SdCmd(0x31);                                  // 选择 ext0 指令集
    SdCmd(0xcd); SdData(0x00);                    // 设置 EEROM 读功能
    delayms(100);
    SdCmd(0xfd);                                  // 数据读出 EEROM
    delayms(100);
    SdCmd(0xcc);                                  // 终止操作 EEROM
    //-----------------------------------------------------------------
    SdCmd(0x30);                                  // 选择 ext0 指令集
    SdCmd(0xaf);                                  // 开显示
  }
```

程序说明：

1、产品推荐设置如下表

型号 \ 指令参数 \ 指令代码	81H	32H	bcH	caH	bbH
LM240160DCW	0f 04	05 00 02	01 00 02	04 27 00	01
LM240160DFW	3c 03	05 00 02	01 00 02	04 27 00	01
LM240160GCW	09 04	05 00 02	01 00 02	04 27 00	01
LM240160GFW	04 04	05 00 02	01 00 02	04 27 00	01
LM160128AFW-1	10 04	04 00 02	02 01 02	04 1f 00	01

2、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。这样当运行初始化函数后，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，我们可以把初始化函数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和驱动程序以及时序关系。

第五章　液晶显示模块指令系统

内置 ST7529 液晶显示模块系列的指令系统比较简单。需要特别说明与关注的是模块内的显示 RAM 的结构与数据的格式。

一、显示 RAM 的结构与数据传输格式

Memory Map (3B3P, 8-bit mode)

			Column									
LCD read direction	CI = 0		0			1				84		
	CI = 1		84			83				0		
	Pixel		P0	P1	P2	P3	P4	P5		P252	P253	P254
	Data Line		$D7'_{1,0}$ $D6'_{1,0}$ $D5'_{1,0}$ $D4'_{1,0}$ $D3'_{1,0}$	$D7'_{2,0}$ $D6'_{2,0}$ $D5'_{2,0}$ $D4'_{2,0}$ $D3'_{2,0}$	$D7'_{3,0}$ $D6'_{3,0}$ $D5'_{3,0}$ $D4'_{3,0}$ $D3'_{3,0}$	$D7'_{1,1}$ $D6'_{1,1}$ $D5'_{1,1}$ $D4'_{1,1}$ $D3'_{1,1}$	$D7'_{2,1}$ $D6'_{2,1}$ $D5'_{2,1}$ $D4'_{2,1}$ $D3'_{2,1}$	$D7'_{3,1}$ $D6'_{3,1}$ $D5'_{3,1}$ $D4'_{3,1}$ $D3'_{3,1}$		$D7'_{1,84}$ $D6'_{1,84}$ $D5'_{1,84}$ $D4'_{1,84}$ $D3'_{1,84}$	$D7'_{2,84}$ $D6'_{2,84}$ $D5'_{2,84}$ $D4'_{2,84}$ $D3'_{2,84}$	$D7'_{3,84}$ $D6'_{3,84}$ $D5'_{3,84}$ $D4'_{3,84}$ $D3'_{3,84}$
Block	LI = 0	LI = 1										
0	0	159										
	1	158										
	2	157										
	3	156										
1	4	155										
	5	154										
	6	153										
	7	152										
2	8	151										
	9	150										
38	152	7										
	153	6										
	154	5										
	155	4										
39	156	3										
	157	2										
	158	1										
	159	0										
SEGout			0	1	2	3	4	5		252	253	254

图九　内置 ST7529 液晶显示模块 RAM 单元的地址结构

显示 RAM 的结构图见图九所示，在模块的 RAM 中，一个单元的地址由行地址和列地址组合而成，地址定义如下：

行地址：按照像素行定义，以显示屏自上而下顺序排序，取值范围是 1~160 行。（以 240*160 点阵为例）

列地址：以水平像素点为基础，每 3 个像素点为一组，由一个列地址唯一指定。我们可以用下面 2 个思路来理解列地址：

1、RAM 单元的数据宽度。在模块 RAM 单元的数据位为 15 位，每 5 位数据为一个像素点的显示数据，一个 RAM 单元容纳了 3 个像素的数据。

2、列块表示。以 3 个像素点的数据组合成一个列块，每一次的读写 RAM 时，必须以 3 个数据连续写入/读出为一个最小操作。

这两种理解，前者有利于对如此结构的认同与理解，后者则为读写操作时的提示。列地址（或列块地址取值范围是 0-79，（以 240*160 点阵为例）

行块地址：每 4 行为一组，将 160 行分为 1-40 行块，（以 240*160 点阵为例），形成行块地址。行块地址将应用于局部显示和卷动功能上。

一个像素的数据为 5 位组成，在数据字节中占据了 DB7~DB3，该字节的低 3 位 DB3~DB0 无用。DB7~DB3 设置数据不同，将产生不同的灰度级显示，最大能产生 32 级灰度；数据的 DB2~DB0 位无用，一般我们设置为 0。但如果使用该模块控制器内部的 LCD 驱动电源驱动，就其驱动能力而言，我们建议使用单色显示为宜。

二、指令描述

内置 ST7529 液晶显示模块系列的指令系统　分基础基本指令集 EXT0 和扩展指令集 EXT1 两组，将实现对模块的初始设置，包括启动 LCD 驱动电源工作、灰度数据设置、调节对比度电压、扫描时序的设置等；显示数据的操作，有数据的读写、修改写、局部显示以及卷动设置等。下面逐一说明指令的使用。

1、指令名称：INSTRUCTION SET　CHANGE　　指令代码：30H/31H

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

指令描述：该指令选择当前要使用的指令集。

EXT：指令集选择位。当 EXT=0 时，选择基本指令集 EXT0；当选择 EXT=1 时，选择扩展指令集 EXT1。

EXT0 指令集指令描述

2、指令名称：DISPLAY ON/OFF　　指令代码：AFH/AEH

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

指令描述：该指令为显示开关指令。开显示指令在休眠模式下无效。

ON/OFF：显示开关位。当 ON/OFF 设置为“1”时，为开显示指令 AFH；当 ON/OFF 设置为“0”时，为关显示指令 AEH。

3、指令名称：NORMAL DISPLAY　　指令代码：A6H/A7H

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

指令描述：该指令为正性/负性显示指令。

INV：显示设置位。当 INV 设置为“0”时，为正性显示指令 A6H，显示为正常的“黑底白字”；当 INV 设置为“1”时，为负性显示指令 A7H，显示为“白底黑字”。

4、指令名称：COMMON SCAN　　指令代码：BBH + 1 参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1	0	1	1
X	X	X	X	X	CD2	CD1	CD0

指令描述：该指令规定了行扫描电路连接的顺序。该指令带一个参数。

CD[2:0]：为 LCD 行电极驱动顺序设置，LM240160D 的初始设置为 01H。设置表如下：

CD2	CD1	CD0	行扫描顺序				
			COM0PIN		COM79PIN	COM80PIN	COM159PIN
0	0	0	0		79	80	159
0	0	1	0	→	79	159	→ 80
0	1	0	79		0	80	159
0	1	1	79		0	159	80

5、指令名称：DISPLAY CONTROL　　指令代码：CAH + 3 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	1	0	1	0
X	X	X	0	0	CLD	0	0
X	X	DT5	DT4	DT3	DT2	DT1	DT0
X	X	X	FI	LF3	LF2	LF1	LF0

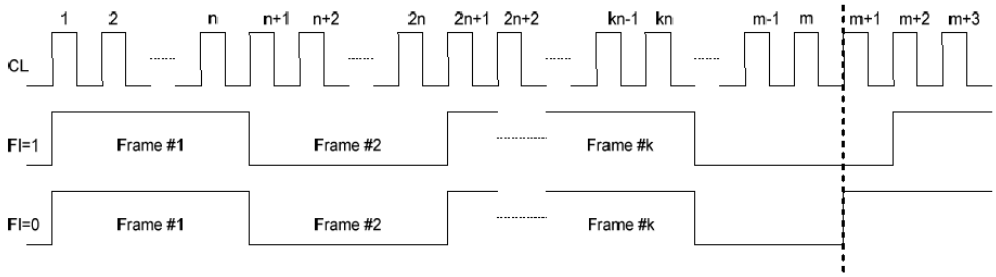
指令描述：该指令设置了显示驱动的时序关系。该指令必须在退出休眠模式（94H 指令执行）之前设置，并且在开显示后不能修改。该指令带有 3 个参数，它们是：

CLD：内部显示时钟 CL 的分频比率。用于改变内部或外部时钟的分频状态。CLD=0，不分频；CLD=1，二分频。

DT[5:0]：显示行块的占空比设置值。DT[]=显示行数/4 – 1，初始值为 0。

LF[3:0]： 行循环量的设置值，表示在一帧周期内扫描的行数。LF[]= 一帧扫描行数-1。取值范围是 2~16，初始值为 AH

FI：帧翻转类型设置。见图十所示，当占空比设置值不是每帧的行扫描设置值的整倍数时，在每个扫描行结束时帧翻转的方式，FI=0，为在第 m+1 个 CL 脉冲时翻转；FI=1，为在第 m+2 个 CL 脉冲时翻转。其中：m = n * k + r （m：占空比倒数；n：一帧的扫描行数 LF[]；k：倍数 r：m/n 的余数）



图十　帧翻转类型

6、指令名称：SLEEP　　指令代码：95H/94H

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

指令描述：该指令为控制器休眠模式的控制。

SLP：为休眠模式的设置位。当 SLP=1（95H）时，控制器进入休眠工作模式；当 SLP=0（94H0 时，控制器退出休眠模式。

7、指令名称：COLUMN ADRESS SET　　指令代码：15H + 2 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	0	1	0	1
SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0
EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

指令描述：该指令设置了 MPU 读写显示 RAM 时的列地址区域。在 MPU 读写显示 RAM 后，（满足条件），列地址自动加一，当列地址加一到列地址设置区域的结束地址后，将返回到列地址区域的起始地址。该指令带有 2 个参数，第一参数为列地址区域的起始地址列号 SC，第二参数为列地址区域的结束地址列号 EC，该指令的参数要成对设置，且 SC<EC。

SC[7:0]：起始列地址值，取值范围为 0~79；

EC[7:0]：结束列地址值，取值范围为 1~79。

8、指令名称： LINE　ADRESS SET　　指令代码：75H +2 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	1	0	1	0	1
SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0
EL7	EL6	EL5	EL4	EL3	EL2	EL1	EL0

指令描述：该指令设置了 MPU 读写显示 RAM 时的行地址区域。MPU 每操作一次显示 RAM 后，行

地址自动加一，当行地址为行地址区域的结束地址时，列地址自动加一，且行地址返回到行地址区域的起始地址。该指令带有 2 个参数，第一参数为行地址区域的起始地址行号 SL，第二参数为行地址区域的结束地址行号 EL，该指令的参数要成对设置，且 SL<EL。

SL[7:0]：起始行地址值，取值范围为 0~159；

EL[7:0]：结束行地址值，取值范围为 1~159。

9、指令名称：DATA SCAN DIRECTION　　指令代码：BCH + 3 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1	1	0	0
X	X	X	X	X	C/L	CI	LI
X	X	X	X	X	X	X	CLR
X	X	X	X	X	GS2	GS1	GS0

指令描述：该指令设置了显示数据存储作时的操作参数，LM240160D 初始值为 01、00、02。

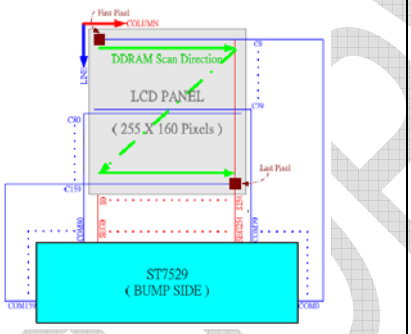
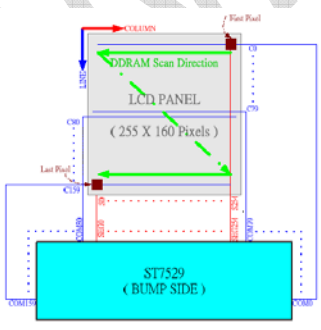
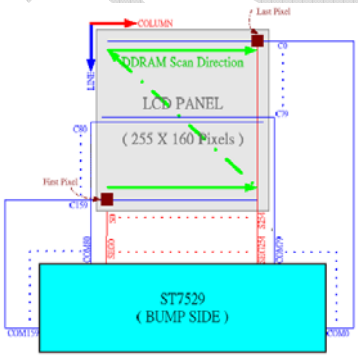
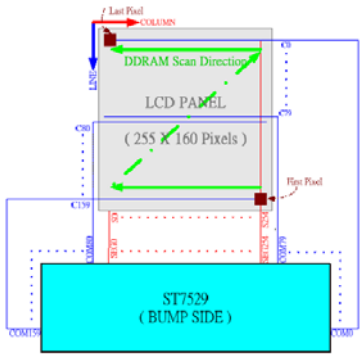
LI：行地址方向设置。当 LI=0 为正序（自上向下）；LI=1 为反序（自下而上）。

CI：列地址方向设置。当 CI=0 为正序（自左向右）；CI=1 为反序（自右向左）。

C/L：地址扫描方向设置。C/L=0 沿列方向；C/L=1 沿行方向。

CLR：列地址中像素排列顺序设置。CLR=0，为正序（P1、P2、P3）；CLR=1 为反序（P3、P2、P1）。（见图九示意）

GS[2:0]：灰度数据写入格式设置。GS[]=1 为 2B3P 格式；GS[]=2 为 3B3P 格式。

LI \ CI	0	1
0		
1		

10、指令名称：MEMORY WRITE　　指令代码：5CH + 若干显示数据

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

指令描述：该指令将进入数据输入模式，随后的数据写入，都从已设置的行、列地址起始地址指向的单元开始，每写一组（3 个像素）数据后，行或列地址都将自动修正。数据输入模式将由任意一个指

令的写入而终止。

11、指令名称：MEMORY READ　　指令代码：5DH + 若干显示数据

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

指令描述：该指令将进入数据读模式。随后的数据读取，都从已设置的行、列地址起始地址指向的单元开始，每写一组（3 个像素）数据后，行或列地址都将自动修正。数据输入模式将由任意一个指令的写入而终止。

需要说明的是，在首次读取数据时，第一个读数据操作应该为空操作，该数据不是该地址的数据。

12、指令名称：PARTIAL DISPLAY MODE　　指令代码：A8H/A9H + 2 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1	0	0	PON/OFF
X	X	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0
X	X	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0

指令描述：该指令为局部显示模式的开关。当 PON/OFF=0（A8H）时，控制器进入局部显示模式，所谓局部显示是控制器将显示进入节电模式，仅显示选择的行区域。显示的区域将由随后的 2 个参数设置，起始行块地址和结束行块地址。局部显示是以基础行块地址为基础的，一个基础行块包含 4 行，见图九示意。当 PON/OFF=1（A9H）时，控制器退出局部显示模式，该指令写入时，不带参数。

PON/OFF：局部显示模式开关。PON/OFF=0 为进入局部显示模式；PON/OFF=1 为退出局部显示模式。

PTS[5:0]：起始基础行块号，取值范围是 0-39。

PTE[5:0]：结束基础行块号，取值范围是 0-39。

13、指令名称：READ MODIFY WRITE IN　　指令代码：E0H

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

指令描述：该指令设置将进入修改写模式。在进入该模式之前要先设置好行地址和列地址。进入修改写模式后，读数据后地址不变，只有写数据时，行或列地址才自动修改。

14、指令名称：READ MODIFY WRITE END　　指令代码：EEH

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

指令描述：该指令将退出修改写模式。

15、指令名称：AREA SCRLL SET　　指令代码：AAH + 4 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1	0	1	0
X	X	TB5	TB4	TB3	TB2	TB1	TB0
X	X	BB5	BB4	BB3	BB2	BB1	BB0
X	X	NSB5	NSB4	NSB3	NSB2	NSB1	NSB0
X	X	X	X	X	X	SCM1	SCM0

指令描述：该指令进入局部卷动的模式，该指令带 4 个参数，规定了卷动的区域、卷动方式、以及不参与卷动的区域，见图十一所示。它们的设置是：

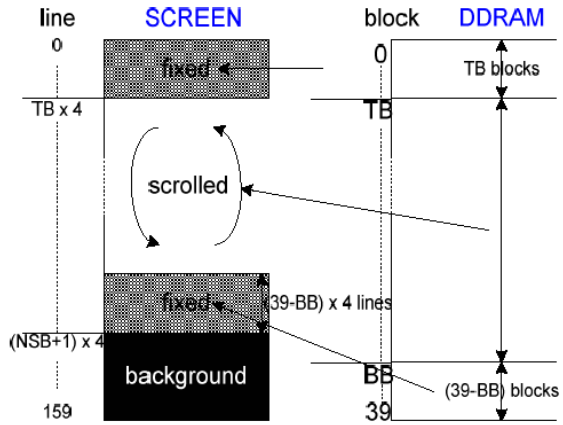
TB[5:0]：卷动+背景区域的上边界，取值为基础行块地址 0-39；

BB[5:0]：卷动+背景区域的下边界，取值为基础行块地址 0-39；

NSB[5:0]：卷动区域的底边界行数 0-39；

SCM[1:0]：卷动方式的设置

SCM1	SCM0	卷动方式	设置值		
			TB	BB	NSB
0	0	中心方式	上不动区域高度为上边界地址	下不动区域高度为 39-下边界地址	下边界地址=规定值
0	1	上方式	0	下不动区域高度为 39-下边界地址	下边界地址=规定值
1	0	下方式	上不动区域高度为上边界地址	39	39
1	1	全屏	0	39	39



图十一　　卷动设置示意图

16、指令名称：SCROLL START ADDRESS SET　指令代码：ABH +1 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1	0	1	1
X	X	SB5	SB4	SB3	SB2	SB1	SB0

指令描述：该指令设置了显示 RAM 的显示起始行地址，即显示顶行的行地址，它决定了屏幕的显示内容。该指令带有一个参数，为显示起始行号。定时间间隔定间距修改该参数，将产生卷动的效果。当该指令执行时，不要重复执行 AREA SCRLL SET 指令（AAH）。该指令在 AREA SCRLL SET 指令（AAH）设置后，可以实现局部区域的卷动功能。

SB[5:0]：显示起始行块地址

17、指令名称：INTERNAL OSCILLATION WORK 指令代码：D1H/D2H

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

指令描述：该指令启动/关闭内部振荡器电路。启动指令只有在管脚 CLS 设置为 VDD 时有效。内部振荡器电路在 RESET 时为关闭状态。LM240160D 初始化时需要启动内部振荡器。

18、指令名称：POWER CONTROL SET　　指令代码：20H +1 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	0	0	0
X	X	X	X	VB	0	VF	VR

指令描述：该指令设置了内部 LCD 驱动电源的工作状态，带有一个参数。当启动内部 LCD 驱动电源工作时，推荐先启动升压电路，延迟 1ms 后，再开启电压跟随器和发生器。

VB：升压电路开关。VB=1，开；VB=0，关

VF：电压跟随器开关。VF=1，开；VF=0，关

VR：参考电压发生器开关。VR=1，开，VR=0，关

19、指令名称：ELECTRONIC VOLUME CONTROL　　指令代码：81H +2 个参数

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	0	1
X	X	VPR5	VPR4	VPR3	VPR2	VPR1	VPR0
X	X	X	X	X	VPR8	VPR7	VPR6

指令描述：该指令将优化 LCD 电源电压 V0。该指令与 VOLUP 和 VOLDOWN 指令一起实现 LCD 显示的对比度调节。该指令带有 1 个参数，分 2 个字节传输。

VPR[8:0]：V0 设置值。该参数计算公式为：V0=0.04 x VPR[8:0]+3.6。

20、指令名称：INCREMENT ELECTRONIC CONTROL　　指令代码：D6H

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

指令描述：该指令基于 ELECTRONIC VOLUME CONTROL 指令设置的 VPR 值上加 1（0.04V）。增加范围是 000000B—111111B。

21、指令名称：DECREMENT ELECTRONIC CONTROL　　指令代码：D7H

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

指令描述：该指令基于 ELECTRONIC VOLUME CONTROL 指令设置的 VPR 值上减 1（0.04V）。减少范围是 000000B—111111B。

上述两个指令（20、21 条指令）所产生的变化值，在执行第 19 条指令后复位归另。

22、指令名称：NOP　　指令代码：25H

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

指令描述：该指令为空操作指令，对运行没有任何影响，但可以终止 IC 的测试模式。因此建议周期性的执行它，以避免控制器因干扰而失效。

23、指令名称：STATUS READ

D7	D6	D5	D4	D3	D2	D1	D0
SCM1	SCM0	RMW ON/OFF	SCAN DIR	DISP ON/OFF	EEPROM	NOR DISP	PART DISP

指令描述：该操作为读状态字。状态字含义如下：

SCM[1:0]：为 AREA SCRLL SET 的 SCM 设置

RWM ON/OFF：0 为输出；1 为输入

SCAN DIR：0 为列方向；1 为行方向

DISP ON/OFF：0 为关显示；1 为开显示

EEPROM：0 为输出存取；1 为输入存取

NOR DISP：0 为负性显示；1 为正性显示

PART DISP：0 为关闭局部显示；1 为进入局部显示

24、指令名称：INITIAL CODE1　　指令代码：07H + 19H

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

指令描述：该指令用于 EEPROM 内部 ACK 信号的产生。建议在 EEPROM 读写操作前使用该指令。该指令在不稳定的电源系统下将改善 ACK 信号。

EXT1 指令集指令描述

控制器使用 16 级灰度和 2FRC 方式组合实现 32 级灰度显示。每个灰度级色度直接由 31-PWM（5BIT）控制，下面 2 个指令（24、25 指令）将设置各个灰度级色度数值。

25、指令名称：SET GRAY 1 VALUE　　指令代码：20H +16 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	0	0	0
X	X	X	G0F14	G0F13	G0F12	G0F11	G0F10
X	X	X	G1F14	G1F13	G1F12	G1F11	G1F10
X	X	X	G2F14	G2F13	G2F12	G2F11	G2F10
X	X	X	G3F14	G3F13	G3F12	G3F11	G3F10
X	X	X	G4F14	G4F13	G4F12	G4F11	G4F10
X	X	X	G5F14	G5F13	G5F12	G5F11	G5F10
X	X	X	G6F14	G6F13	G6F12	G6F11	G6F10
X	X	X	G7F14	G7F13	G7F12	G7F11	G7F10
X	X	X	G8F14	G8F13	G8F12	G8F11	G8F10
X	X	X	G9F14	G9F13	G9F12	G9F11	G9F10
X	X	X	G10F14	G10F13	G10F12	G10F11	G10F10
X	X	X	G11F14	G11F13	G11F12	G11F11	G11F10
X	X	X	G12F14	G12F13	G12F12	G12F11	G12F10
X	X	X	G13F14	G13F13	G13F12	G13F11	G13F10
X	X	X	G14F14	G14F13	G14F12	G14F11	G14F10

指令描述：该指令设置了奇数帧的灰度 PWM 设置值。

26、]指令名称：SET GRAY 2 VALUE　　指令代码：21H +16 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	0	0	1
X	X	X	G0F24	G0F23	G0F22	G0F21	G0F20
X	X	X	G1F24	G1F23	G1F22	G1F21	G1F20
X	X	X	G2F24	G2F23	G2F22	G2F21	G2F20
X	X	X	G3F24	G3F23	G3F22	G3F21	G3F20
X	X	X	G4F24	G4F23	G4F22	G4F21	G4F20
X	X	X	G5F24	G5F23	G5F22	G5F21	G5F20
X	X	X	G6F24	G6F23	G6F22	G6F21	G6F20
X	X	X	G7F24	G7F23	G7F22	G7F21	G7F20
X	X	X	G8F24	G8F23	G8F22	G8F21	G8F20
X	X	X	G9F24	G9F23	G9F22	G9F21	G9F20
X	X	X	G10F24	G10F23	G10F22	G10F21	G10F20
X	X	X	G11F24	G11F23	G11F22	G11F21	G11F20
X	X	X	G12F24	G12F23	G12F22	G12F21	G12F20
X	X	X	G13F24	G13F23	G13F22	G13F21	G13F20
X	X	X	G14F24	G14F23	G14F22	G14F21	G14F20

指令描述：该指令设置了偶数帧的灰度 PWM 设置值。

27、指令名称：ANALOG CIRCUIT SET　　指令代码：32H +3 个参数

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	0	1	0
X	X	X	X	X	OSF2	OSF1	OSF0
X	X	X	X	X	X	BE1	BE0
X	X	X	X	X	BS2	BS1	BS0

指令描述：该指令设置了 LCD 驱动的一些工作参数。带有 3 个参数，它们是 ：

OSF[2:0]：振荡器 OSC 频率调节值 FCL（HZ）=帧频*（扫描行数+1），决定模块的帧扫描频率。

OSC[2:0]	0	1	2	3	4	5	6	7
频率值(KHz)	12.7	13.2	14.3	15.7	17.3	19.3	21.9	25.4

BE[1:0]：升压效率设置，决定驱动电源 V0 的稳定性。初始值 6K

BE[1:0]	0	1	2	3
升压电容上的频率(Hz)	3K	6K	12K	24K

BS[2:0]：LCD 驱动偏压设置

BS[2:0]	0	1	2	3	4	5	6	7
LCD 偏压值	1/14	1/13	1/12	1/11	1/10	1/9	1/7	1/5

28、指令名称：SOFTWARE INITIAL　　指令代码：34H

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

指令描述：该指令为软件复位。

29、**指令名称：CONTROL EEPROM　　指令代码：CDH + 1 个参数**

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

指令描述：该指令设置了 EEPROM 的读写状态。带有 1 个参数。

EEWR：读写选择位。当 EEWR=1 时，EEPROM 为写使能状态；当 EEWR=0 时，EEPROM 为读使能状态。

30、指令名称：CANCEL EEPROM　　指令代码：CCH

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

指令描述：该指令为终止 EEPROM 的读写操作使能。

31、指令名称：WRITE DATA TO EEPROM　　指令代码：FCH

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

指令描述：该指令为向 EEPROM 里写数据。

32、指令名称：READ DATA FROM EEPROM　　指令代码：FDH

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

指令描述：该指令将从 EEPROM 里读取数据。

第六章　液晶显示模块的调试指导

调试的工作主要是经验和实践的结合，我们要具备冷静、细心的态度。首先我们排除接口连线错误的因素，因为这需要读者自己的检查。内置 ST7529 液晶显示模块接口提供多种形式，使得我们根据 MPU 系统提供的时序接口来选择模块的适配时序，从而把硬件电路的时序问题给回避掉了。在采用软件编程实现时序关系，如 I/O 寻址方式和串行方式，会给我们带来不易察觉的时序匹配问题，希望多给予关注。我们就我们的经验提示如下：

一、硬件方面

连接线接触是否实在

因为作为调试使用，系统板可能还没有专门为模块设计接口，或者直接连到仿真器上测试，这时可能会使用临时制作的连线。所以请检查连接线接头处的接触是否实在，包括检查焊接点是否虚焊，连接点是否隔离很好等。况且模块提供了 FPC 的焊接或插座的焊接，这都是需要焊接技术的操作，也是接口最容易损坏的位置，因此要格外的小心操作。

建议在模块的 PCB 板上接口使用插座。

我们推荐使用模块的内部 LCD 驱动电源，，在内部驱动电源的外部元件连接时要注意倍压电容的连接电路和电容的选择。

模块的复位脚/RES 的连接

一定要接一个复位电路的。当产品使用环境比较好时，可以直接采用我们在管脚定义里提供的 RC 复位电路，但当产品使用环境比较恶劣时，也可以将/RES 接到 MPU 的口线上，这样不仅可以用口线复位，还可以进行定时刷新，预防一些其它干扰。

/RES 信号复位的原则是：

i)上电后应该比 MPU 提前复位，同时在程序上电执行液晶模块初始化函数之前先运行一段延时程序。

ii)上电后，/RES 保持低电平至少 10ms，低电平拉高后至少保持 800ms，再对模块进行软件初始化。

对比度调节

模块使用了内部 LCD 驱动电源，通过指令调节对比度，所以在接口没有调通时，我们是看不到显示效果的。一旦接口调通，运行初始化程序后，只要正常就会出现显示的效果，可以通过检测 PCB 板上各电容上的电压值，如果检测有高达 14V 的电压，则表示模块内的 LCD 升压电路已经工作，从而验证初始化设置、驱动程序以及接口电路的正确与否。

时序的搭配

如果选用的 MPU 为高速器件，或者它的总线读写周期小于 500ns，建议您还是采用 I/O 寻址方式为稳妥。

二 软件问题

主要问题在于：

由于模块上电时复位后，内部电路需要稳定时间，一般在几百 ms，所以在调试时，需要在给模块上电后等待一会儿再运行程序，如果直接在 MPU 板上烧入程序，上电运行，那么，在复位后请多加延迟时间。

在 I/O 寻址方法下，程序完成了时序关系建立，要记住的是/WR、/RD 信号的变化要独立进行，不要与其它信号端（如/CS、RS 等）的设置共用，如果同时设置这几个信号，将会出现读写错误，而且不容易检查到。

第七章　　液晶模块的功能函数

由于该系列模块的数据格式为 5bits/像素，为了将常用的单色显示数据转换成 5bits/像素的数据，我们首先定义显示色度变量，作为写入数据的依据。

```
uchar Fgcolor,Bgcolor;          //前景色变量（为 1 的数据），背景色变量（为 0 的数据）
```

我们提供的功能函数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，在函数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、清屏程序

1、清屏函数

```
void ClearRAM()
{
// 以背景色为清屏底色
  uint i;
  SdCmd(0x30);                                    // 选择 ext0 指令集
  SdCmd(0x15); SdData(0x00); SdData(0x4f);        // 列地址区域设置
  SdCmd(0x75); SdData(0x00); SdData(0x9f);        // 行地址区域设置
  SdCmd(0x5c);                                    // 数据写入模式
  for (i=0;i<160*80;i++)                          // 循环总的数据量（列块*行数）
      {
        SdData(Bgcolor);                          // 写入数据
        SdData(Bgcolor);                          // 写入数据
        SdData(Bgcolor);                          // 写入数据
     }
}
```

二、字符写入程序

字符写入要根据控制器的内部 DDRAM 的结构设置。由于控制器将列地址定义为列块地址，且一个列块含有 3 个像素点，列地址的修正也以 3 为单位进行，所以在字符写入程序中，考虑了这个特殊结构，因此我们设定了字符写入的格式：西文字符为 6*8 点阵，汉字为 18*16 点阵。

2、西文字符写入函数

```
void PrintASCII(uchar x,y, asciicode)
{
 // 坐标 X 为像素列块 0-79 列块，3 像素点数据/列块
  // 坐标 Y 为像素点行数 0-159
  // asciicode 为西文字符代码，对应字库为 ASCIITAB
 // 写入字符前，要先设置显示前景色和背景色
     uint p;
   uchar i,j,k,Ddata;
   SdCmd(0x30);                                  // 选择 ext0 指令集
   SdCmd(0x15); SdData(x); SdData(x+1);// 列地址区域设置（6 点列）
   SdCmd(0x75); SdData(y); SdData(y+7);// 行地址区域设置 （8 点行）
   SdCmd(0x5c);                                  // 数据写入模式
   p=asciicode*6;                               // 计算字模数据起始地址
   for(i=0;i<8;i++)                            // 循环 8 行像素写入
     {
      for (j=0;j<6;j++)                       // 循环 6 列像素写入
        {
        k=ASCIITAB[p+j];                       // 取字模数据
        k=k>>i;                                // 字模数据移位
        if((k&0x01)==0x00)                     // 根据数据位值设置显示色数据
            {
            Ddata=Bgcolor;                     // 数据为 0，选择背景色数据
            }
        else
```

```
                {
                Ddata=Fgcolor;                  // 数据为 1，选择前景色数据
                }
             SdData(Ddata);                     // 数据写入
            }
          }
}
```

3、汉字写入函数

```
//--中文写入以 18*16 点阵格式写入，其中第 17、18 像素数据为背景色，作为字间距出现
void PrintGB(uchar x, y, chinesecode)
// 坐标 X 为像素列块 0-79 列块，3 像素点数据/列块
// 坐标 Y 为像素点行数 0-159
// chinesecode 为中文字符代码，对应字库为 CCTAB
// 写入字符前，要先设置显示前景色和背景色
{
    uint p;
    uchar i,j,k,Ddata;
    SdCmd(0x30);                                    // 选择 ext0 指令集
    SdCmd(0x15); SdData(x); SdData(x+5);   // 列地址区域设置（18 点列）
    SdCmd(0x75); SdData(y); SdData(y+15); // 行地址区域设置（16 点行）
    SdCmd(0x5c);                                    // 数据写入模式
    p=chinesecode*32;                               // 计算字模数据起始地址
    for(i=0;i<16;i++)                               // 循环 16 行像素写入
      {
       k=CCTAB[p+i];                                // 取汉字左半部分字模数据
       for (j=0;j<8;j++)                            // 循环汉字 8 列像素写入
         {
         if((k&0x80)==0x00)                         // 根据数据位值设置显示色数据
             {
              Ddata=Bgcolor;                        // 数据为 0，选择背景色数据
             }
         else
             {
              Ddata=Fgcolor;                        // 数据为 1，选择前景色数据
             }
          SdData(Ddata);                            // 数据写入
          k=k<<1;                                   // 字模数据移位
         }
       k=CCTAB[p+i+16];                             // 取汉字右半部分字模数据
       for (j=0;j<8;j++)                            // 循环汉字 8 列像素写入
         {
         if((k&0x80)==0x00)                         // 根据数据位值设置显示色数据
             {
              Ddata=Bgcolor;                        // 数据为 0，选择背景色数据
             }
          else
             {
              Ddata=Fgcolor;                        // 数据为 1，选择前景色数据
             }
           SdData(Ddata);                           // 数据写入
            k=k<<1;                                 // 字模数据移位
           }
       SdData(Bgcolor);                             // 空点写入
       SdData(Bgcolor);                             // 空点写入
     }
}
```

三、绘图程序

根据控制器内部 DDRAM 的结构特点（以 3 点像素为一列块地址）和图形特点（8 点像素为 1 个字节数据），该程序要求图形文件的数据格式在水平方向的图形宽度以 3 和 8 的倍数同时出现。

4、图画写入函数

```
// 图形格式要求：水平方向宽度要以 3 和 8 的倍数出现
// 8 的倍数是数据以字节形式出现
// 3 的倍数是因为 DDRAM 列地址以列块形式计算的，即 3 点像素数据为一个列块
void ShowBMP(uchar x,y,width,high,uchar bmp[])
{
// 坐标 X 为像素列块 0-79 列块，3 像素点数据/列块
// 坐标 Y 为像素点行数 0-159
// 图形宽度 width 为水平方向点列数，要求该值为 3 和 8 的倍数
// 图形高度 high 为垂直方向点行数，取值为 0-159
// 图形数组 bmp[]为所要写入的图形数据，以 1bpp（8dots/byte)、水平排列格式表示
// 写入图形前，要先设置显示前景色和背景色
    uint i,p;
    uchar j,k,Ddata;
    SdCmd(0x30);                                    // 选择 ext0 指令集
    SdCmd(0x15); SdData(x); SdData(x+width/3-1);    // 列地址区域设置
    SdCmd(0x75); SdData(y); SdData(y+high-1);       // 行地址区域设置
    SdCmd(0x5c);                                    // 数据写入模式
    p=0;                                            // 数组指针初始化
    width=width/8;                                  // 计算水平字节数==图形宽度/8
    for(i=0;i<high*width;i++)                       // 循环总的数据量（字节数）
      {
      k=bmp[p];                                     // 取图形数据
      for (j=0;j<8;j++)                             // 循环 8 点列数据写入（按字节方式）
          {
          if((k&0x80)==0x00)                        // 根据数据位值设置显示色数据
              {
              Ddata=Bgcolor;                        // 数据为 0，选择背景色数据
              }
          else
              {
              Ddata=Fgcolor;                        // 数据为 1，选择前景色数据
              }
          SdData(Ddata);                            // 数据写入
          k=k<<1;                                   // 字模数据移位
          }
      p++;                                          // 数组指针加 1
      }
}
```

5、绘点函数

```
void Draw_Dot(uint x,y)
// 坐标(x,y)，x 为水平方向像素点列，y 为垂直方向像素点行
{
  uchar k1,k2,k3,m;
    SdCmd(0x30);
    SdCmd(0x15); SdData(x/3); SdData(x/3+1);        // 列地址区域设置
    SdCmd(0x75); SdData(y); SdData(y+1);            // 行地址区域设置
    SdCmd(0xe0);                                    // 修改写模式进入
    k1=RdData();                                    // 空读
    k1=RdData();                                    // 读数据
    k2=RdData();
    k3=RdData();
    m=x%3;
    switch (m)
        {
        case 0 :    k1=Fgcolor;break;               // 写点
        case 1 :    k2=Fgcolor;break;
        case 2 :    k3=Fgcolor;break;
        }
    SdData(k1);                                     // 复写数据
    SdData(k2);
```

```
    SdData(k3);
    SdCmd(0xee);                                    // 退出修改写模式

}
```

6、绘线函数

```
void Draw_Line(uint x1,y1,x2,y2)
// x 为水平方向像素点列，y 为垂直方向像素点行
// (x1,y1)为起始点坐标，(x2,y2)为终止点坐标
{
   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);
          }
       }
    }

}
```

深圳市拓普微科技开发有限公司制作