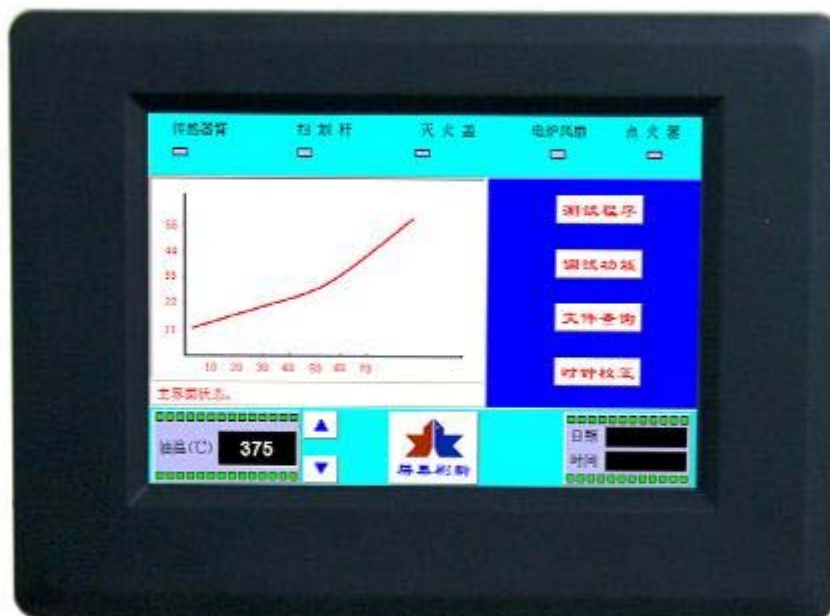


LCD 智能显示终端 VT_TFT8060 说明书



产品背面

一. 功能概述

VT_TFT8060 是一款可用单片机控制的智能显示终端，分辨率为 800×600 ，最大支持 64K 色（16 位色）。

VT_TFT3224 的主要功能：

1. 在指定的 X 轴，Y 轴位置显示 $6 \times 13 / 8 \times 16 / 12 \times 24 / 16 \times 32 / 24 \times 48 / 32 \times 64$ 像素的英文字符以及 $16 \times 16 / 32 \times 32$ 像素的中文字符内置有中英文字库。
2. 在指定的 X 轴，Y 轴位置绘制直线，圆或椭圆。
3. 实时监控数据采集（例如 A/D 转换），以波形的形式显示。
4. 显示预置的图片，支持 BMP 和 JPG 两种格式（图片预存在 VT_TFT8060 的内存中）。
5. 在指定位置绘制指定长宽的矩形。
6. 可选的四线电阻式触摸屏、声波触摸屏驱动或鼠标驱动功能。
7. 自动显示实时时钟（可关闭，可指定位置和时钟显示格式）。
8. 所有的功能均可通过单片机的并行口或 RS232 串行口发送简单的指令完成。
9. 附送的仿真软件可实时地观察各种显示效果，可取得各条指令格式和参数。仿真软件的另一大功能是下载用 PC 机编辑好的图片到智能显示终端。

在工业自动化日益普及的今天，人机界面的设计倍受重视。触摸式控制可称是其中的佼佼者，但在触摸屏驱动和液晶显示开发方面，往往要投入大量的人力资源。采用 LCD 智能显示终端可避免以上两个阶段的开发工作，大大减轻开发人员的负担和提高开发效率。专心于产品的应用开发，避免重复性劳动。

本驱动板采用全数字化设计，核心 PCB 板采用四层布线设计，具有良好的电磁兼容性及抗干扰性。触摸屏驱动采用软件滤波技术，大大提高其抗干扰性及准确性，准确度在 ± 3 像素之内。

LCD 智能显示终端可应用于公共场所的信息查询系统、服务机构的资料登记系统、机电一体化的监控检测系统、自动机床人机控制界面，也可应用于机电、冶金、船舶、航空、铁路、电力、通讯导航等领域的系统设备和智能仪表等等。

二. 硬件连接框图

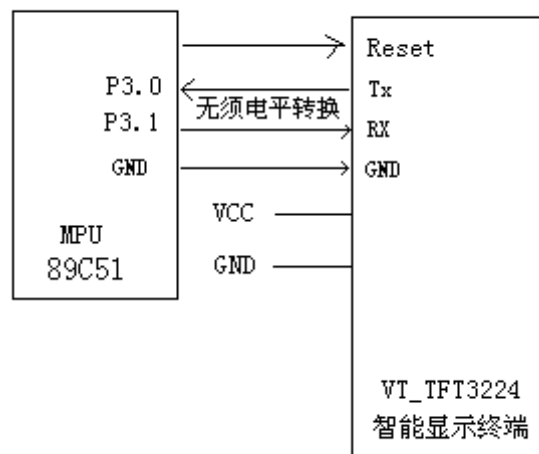


图 2.1 VT_TFT8060 与 51 单片机（或其它系列单片机）串口通信方式连接图

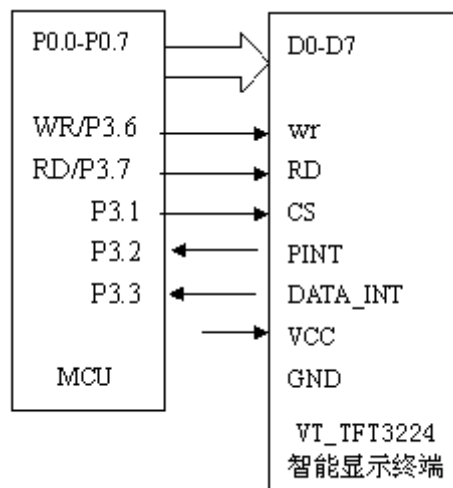


图 2.2 VT_TFT8060 与 51 单片机(或其它系列单片机)的并行总线通信方式连接图

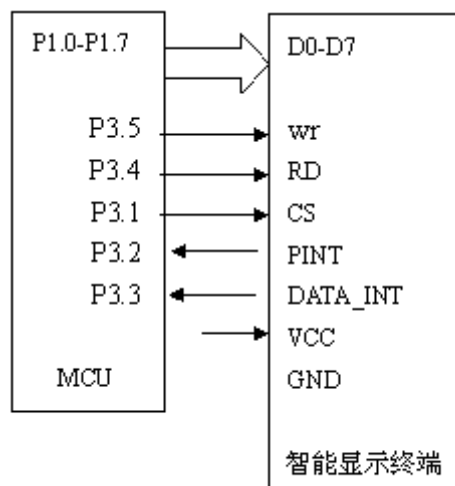


图 2.3 VT_TFT8060 与 51 单片机 (或其它系列单片机)的普通 I/O 口通信方式连接图

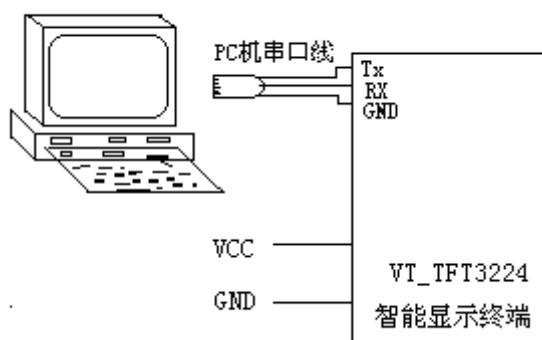


图 2.4 VT_TFT8060 与 PC 机连接进行下载图片或仿真显示

(VT_TFT3224 配有 PC 机专用串口线(3 线 9 脚)，此线内含 R232 电平转换芯片)

注：来自单片机的 TTL 电平信号和来自 PC 机的 $\pm 15V$ 电平信号须经过“MAX3232”等芯片进行电平转换之后，才能使 PC 机和单片机进行通信。

2.1 通信方式

VT_TFT8060 与单片机有三种通信方式：串口通信方式、普通 I/O 口通信方式、并行总线通信方式。

使用并行总线通信方式时，操作时序和操作方法与操作外部普通 RAM 相同，只是忽略了地址总线，即把 VT_TFT3224 当做是一个没有地址的外部 RAM 来操作。详情请看第五章“显示终端与单片机的通信”。

三. 指令功能和指令格式介绍

无论单片机与 VT_TFT8060 处于何种通信方式，功能指令均由连续发送多个字节代码组成。

例如，开时钟显示程序：（注：write_byte()为发送 1 字节函数。）

```
void time_on(unsigned short x,unsigned short y,uchar mode)
```

```
{
    write_byte(0x81);    //指令开始
    Write_byte(0x44);    //指令代码 1（开时钟显示）
    Write_byte(0x54);    //指令代码 2（开时钟显示）
    Write_byte(0x00);    //颜色位（前景色—黑色）
    Write_byte(0xff);    //颜色位（背景色—白色）
    write_byte(x/100);    //时钟显示位置 x 轴坐标（取整运算—x 为两字节变量，高低两
                          //字节 100 进 1）
    write_byte(x%100);    //时钟显示位置 x 轴坐标（取模运算）
    write_byte(y/100);    //时钟显示位置 y 轴坐标
    write_byte(y%100);
    write_byte(mode);    //时钟显示模式
    write_byte(0x84);    //指令结束
}
```

}

请注意：16 进制高低两个字节本是 256 (0xff) 进 1，但“VT-TFT3224”规定高低两个字节 100 (0x64) 进 1，这是由于为了避免与开始/结束指令代码 (0x81/0x84) 冲突，造成识别错误；因此，在取整/取模运算时除数为“100”即“0x64”。也就是说，要向 VT-TFT3224 发送一个两字节整型数时，应先发送除数为 100 的取整运算结果，再发送除数为 100 的取模运算结果。

这种用法可能一开始不太习惯，但熟悉之后并不会影响应用。

3.1 指令的一般格式：

0x81 + 指令代码 + 颜色位 + X 轴坐标 + Y 轴坐标 + ... + 0x84

- (1) 0x81：是通信的握手信号，16 进制，表示指令开始。
- (2) 指令代码：两个字节，每一条指令都有自己唯一的指令代码。
- (3) 颜色位：两个字节，第一个为前景色。第二个为背景色。
- (4) X 轴坐标，Y 轴坐标，... 等等为每条指令所带的数值参数。
- (5) 0x84：指令结束码，表示指令结束。
- (6) 前面五个字节（0x81、指令代码、颜色位）和最后一个 0x84 是每条指令所必须的。中间字节（“.....”）表示有些指令可能额外使用的数据，多少随不同的指令而不同。

特别注意：1. 上面的 0x81, 0x84 均为十六进制。

2. 数值参数中的每一个字节不能大于 0x63(十进制为 99)，即大于 0x63(99) 时需向前一个字节进一。

3. 颜色位是除 0x81, 0x84 外的 0x00 ~ 0xFF 的任意值。这是因为 0x81, 0x84 不能在指令中出现，0x20 是透明色，可以用作背景色，即无底色。

3.2 指令代码表：

指令名称	指令代码 1 (十六进制)	指令代码 2 (十六进制)	指令说明
显示字符 (16*16)	0x44	0x57	在指定位置光标处显示字符串
显示大字符 (32*32)	0x44	0x44	在指定位置显示大字符串

显示各种字号字符 (只限英文字符)	0x59	0xXX	XX=00 在指定位置显示 6X13 的字符 XX=01 在指定位置显示 12X24 的字符 (宋体) XX=02 在指定位置显示 12X24 的字符 (arial 体) XX=03 在指定位置显示 16X32 的字符 (宋体) XX=04 在指定位置显示 16X32 的字符 (arial 体) XX=05 在指定位置显示 24X48 的字符 XX=06 在指定位置显示 32X64 的字符
显示各种字号中英文字符 (需加装 NandFlash 才有这个功能)	0x54	0xXX	XX=00 显示 16X16 的字符串 XX=01 显示 24X24 的字符串 XX=02 显示 32X33 的字符串 XX=03 显示 48X48 的字符串 XX=04 显示 64X64 的字符串
显示变量 (8*16)	0x44	0x56	在指定位置显示 8X16 的变量
显示变量(16*32)	0x44	0x46	在指定位置显示 16X32 的变量
显示各种字号变量	0x56	0xXX	XX=00 在指定位置显示 6X13 的变量 XX=01 在指定位置显示 12X24 的变量 (宋体) XX=02 在指定位置显示 12X24 的变量 (arial 体) XX=03 在指定位置显示 16X32 的变量 (宋体) XX=04 在指定位置显示 16X32 的变量 (arial 体) XX=05 在指定位置显示 24X48 的变量 XX=06 在指定位置显示 32X64 的变量
显示位图 (显示预存于 <u>Nor Flash</u> 的位图)	0x44	0x53	在指定起点的位置显示预置在 <u>Nor Flash</u> 的位图
显示位图 (显示预存于 <u>Nand Flash</u> 的位图)	0x45	0x53	在指定起点的位置显示预置在 <u>Nand Flash</u> 的位图
清除全屏	0x43	0x4c	以指定的背景色清除全屏
绘制矩形	0x43	0x58	以指定的背景色清除指定位置和指定 长宽的矩形, 利用此功能可实现进度 条。

反色矩形	0x43	0x4e	反色指定位置和指定长宽的矩形
绘制圆形	0x44	0x45	在指定位置显示在指定的位置显示圆或椭圆
绘制直线	0x44	0x4c	在指定位置显示直线
开/关时间显示	0x44	0x54	打开时间显示功能
设置时间	0x53	0x54	设置实时时间
读取年月日	0x52	0x44	请求显示终端发送年月日
读取时分秒	0x52	0x54	请求显示终端发送时分秒
自动显示位图	0x5A	0x44	以指定的时间间隔自动循环地在指定的位置显示位图，此功能也可现实简单的动画
显示波形	0x53	0x53	根据接收到的数据产生波形
打开光标	0x43	0x55	打开光标并设置光标的位置
打开校验	0x43	0x4b	打开校验功能。
保存窗口	0x43	0x44	保存当前窗口画面至后台（RAM），以便调用
恢复窗口	0x45	0x44	恢复之前保存于后台的窗口
打开蜂鸣器	0x42	0x42	打开蜂鸣器
开/关背光	0x4C	0x42	开背光：814C426384 关背光：814C420084

3.3 显示字符

功能：在指定的位置或光标处显示 16*16 的中英文字符，当字符到达 LCD 的最左端时会自动转行。**特别注意：**如果要显示的字符超过 140 个（汉字 70 个），执行完这条指令后必须加上 0.1 秒的延时，否则下一条指令将会被忽略，程序无法正常运行（若使用的是串口通信方式可以不加延时）。

举例说明：

注：write_byte() 为发送 1 字节函数。

```
void dis_symbol()
```

```
{ write_byte(0x81);          //指令开始
```



```

Write_byte(0x44);           //指令代码 1 (显示 16*16 字符)
Write_byte(0x57);           //指令代码 2 (显示 16*16 字符)
Write_byte(0xe0);           //颜色位 (前景色—红色)
Write_byte(0x1c);           //颜色位 (背景色—绿色)
write_byte(0x0030/100);      //字符显示位置 x 轴坐标 (取整运算—x 为两字节变量, 高低两字节 100 进 1)

write_byte(0x0030%100);      //字符显示位置 x 轴坐标 (取模运算)
write_byte(240/100);         //字符显示位置 y 轴坐标, 240 为十进制数
write_byte(240%100);
write_byte(0x33);           //发送 “3” 的 ASCLL 码 (0x33 也可以写为'3')
write_byte(0x32);           //发送 “2” 的 ASCLL 码
write_byte(0x84);           //指令结束
}

```

这条指令的功能是在 LCD 的 (0030H,240) 位置显示红色字符“ 32 ”。

指令解释为:

- (1)“81.....84” 通信的握手信号,即表示指令开始和结束。
- (2) “0x44”、“0x57” 显示 16*16 字符的指令代码, 每条指令都有自己唯一的指令代码, 共 2 个字节。
- (3) “0xE0”、“0x1C” 颜色位, 其中 “0xE0” 表示前景颜色。“0x1C” 表示背景颜色。00 为黑色, 03 为蓝色, 1C 为绿色, 1F 为浅蓝色, E0 为红色, EC 为鲜红色, FC 为黄色, FF 是白色, 20 是透明色 (无颜色)。注: 颜色位是除 81, 84 外的 00~FF 的任意值。如果背景色设为 “0x20”, 则显示的字符底色将是透明的, 既背景画面为原来的画面。
- (4)“0x0030” X 轴坐标两个字节, 第一个为高位字节, 第二个为低位字节, 当低位字节大于 63H(十进制为 99)时向高位进一, 以像素为单元。注: 在应用时, x 轴/y 轴坐标用十进制数表示较为方便, 以像素为单位。如果 X 轴的坐标大于 800 则这条指令的功能是在光标处显示字符。
- (5)“240” Y 轴坐标, 以行为单元。
- (6) “33”、“32” 分别是“3”和“2”的 ASCLL 码, 也可以是英数 ASCLL 码或中文内码或者字符串, 例如: “abc”, “微嵌计算机科技有限公司” 等等, 详细情况请参考编程例程。

3.4 显示大字符

功能: 在指定的位置显示 32*32 的中英文字符, 到达 LCD 的最左端时会自动转行。

特别注意: 如果要显示的字符超过 140 个 (汉字 70 个) 这条指令执行完后必须加上 0.1 秒的延时, 否则下一条指令将会被忽略, 程序无法正常运行 (若使用的是串口通信方式

可以不加延时)。

十六进制指令参数举例说明:

81 4444 E01C 0000 0000 C7A7CFB2BFC6BCBC 84

这条指令的功能是在 LCD 的 (0000H,0000H) 位置显示红色字符“千喜科技”。

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “4444” 显示 32*32 大字符的指令代码, 共两个字节。
- (3) “E01C” 颜色位, 其中 “E0” 表示前景颜色。“1C” 表示背景颜色。**注: 颜色位是除 81, 84 外的 00~FF 的任意值。如果背景色设为 “0x20”, 则显示的字符底色将是透明的, 既背景画面为原来的画面。**
- (4) “0000” X 轴坐标, 以像素为单元。**注: 在应用时, x 轴/y 轴坐标用十进制数表示较为方便, 以像素为单位。如果 X 轴的坐标大于 800 则这条指令的功能是在光标处显示字符。**
- (5) “0000” Y 轴坐标, 以行为单元。
- (6) “C7A7CFB2BFC6BCBC” 分别是 “千喜科技” 的中文内码, 即这里可以是多个字节组成的英数 ASCLL 码或中文内码或者字符串, 例如: “abc”, “微嵌计算机科技有限公司” 等等, 详细情况请参考编程例程。

3.5 显示各种字号字符

功能: 在指定的位置显示各种大小的英文字符, 到达 LCD 的最左端时会自动转行。

特别注意: 如果要显示的字符超过 140 个 (汉字 70 个) 这条指令执行完后必须加上 0.1 秒的延时, 否则下一条指令将会被忽略, 程序无法正常运行 (若使用的是串口通信方式可以不加延时)。

十六进制指令参数举例说明:

81 59xx E01C 0000 0000 C7A7CFB2BFC6BCBC 84

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “59” 显示各种字体的变量的指令代码 1。
- (3) “xx” 显示各种字体的字符的指令代码 2。
 XX=00 在指定位置显示 6X13 的字符
 XX=01 在指定位置显示 12X24 的字符 (宋体)
 XX=02 在指定位置显示 12X24 的字符 (arial 体)
 XX=03 在指定位置显示 16X32 的字符 (宋体)
 XX=04 在指定位置显示 16X32 的字符 (arial 体)
 XX=05 在指定位置显示 24X48 的字符
- (4) “E01C” 颜色位, 其中 “E0” 表示前景颜色。“1C” 表示背景颜色。**注: 颜色位是除 81, 84 外的 00~FF 的任意值。如果背景色设为 “0x20”, 则显示的字符底色将是透明的, 既背景画面为原来的画面。**

是透明的，既背景画面为原来的画面。

(5) “0000” X 轴坐标，以像素为单元。注：在应用时，x 轴/y 轴坐标用十进制数表示较为方便，以像素为单位。如果 X 轴的坐标大于 800 则这条指令的功能是在光标处显示字符

(6) “0000” Y 轴坐标，以行为单元。

(7) “C7A7CFB2BFC6BCBC” 分别是“千喜科技”的中文内码，即这里可以是多个字节组成的英数 ASCII 码或中文内码或者字符串，例如：“abc”，“微嵌计算机科技有限公司”等等，详细情况请参考编程例程。

3.6 显示变量(8*16)

功能：在指定的位置显示 8*16 的变量。

十六进制指令参数举例说明：

81 4456 E01C 0000 0005 0064 03 84

这条指令的功能是在 LCD 的 (0000H, 0005H) 位置以十进制数显示变量 0064H，即显示的结果是 100。

指令解释为：

- (1) “81....84” 通信的握手信号。
- (2) “4456” 显示 8*16 变量的指令代码
- (3) “E01C” 颜色位，其中“E0”表示前景颜色。“1C”表示背景颜色。
- (4) “0000” X 轴坐标，以像素为单元。
- (5) “0005” Y 轴坐标，以行为单元。
- (6) “0064” 变量的值，这里的值可以是 16 进制，也可以是十进制，但以十进制方式显示。
- (7) “03” 域宽，指定变量显示的宽度，以字符为单位，不足则在前面补 0。取值范围是 1~5。

请注意：使用这条指令时，确保变量值小于 12900（十进制），否则可能会引起程序运行错误。比如，要显示的变量值为 12900，取整后的结果是 129 即 0x81，这与开始指令代码冲突，造成程序运行错误。

3.7 显示变量(16*32)

功能：在指定的位置显示 8*16 的变量。

十六进制指令参数举例说明：

81 4446 E01C 0000 0005 0064 03 84

这条指令的功能是在 LCD 的 (0000H, 0005H) 位置以十进制数显示变量 0064H，即显示的结果是 100。

指令解释为：

- (1) “81....84” 通信的握手信号。
- (2) “4446” 显示 16*32 变量的指令代码
- (3) “E01C” 颜色位, 其中 “E0” 表示前景颜色。“1C” 表示背景颜色。
- (4) “0000” X 轴坐标, 以像素为单元。
- (5) “0005” Y 轴坐标, 以行为单元。
- (6) “0064” 变量的值, 这里的值可以是 16 进制, 也可以是十进制, 但以十进制方式显示。
- (7) “03” 域宽, 指定变量显示的宽度, 以字符为单位, 不足则在前面补 0。取值范围是 1~5。

请注意: 使用这条指令时, 确保变量值小于 12900 (十进制), 否则可能会引起程序运行错误。比如, 要显示的变量值为 12900, 取整后的结果是 129 即 0x81, 这与开始指令代码冲突, 造成程序运行错误。

3.8 显示各种字号变量

功能: 在指定的位置显示各种大小的变量。

十六进制指令参数举例说明:

81 56xx E01C 0000 0005 0065 04 84

这条指令的功能是在 LCD 的 (0000H, 0005H) 位置以十进制显示变量 0065H, 即显示的结果是 101。

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “56” 显示各种字体的变量的指令代码 1。
- (3) “xx” 显示各种字体的变量的指令代码 2。
 XX=00 在指定位置显示 6X13 的变量。
 XX=01 在指定位置显示 12X24 的变量 (宋体)。
 XX=02 在指定位置显示 12X24 的变量 (arial 体)。
 XX=03 在指定位置显示 16X32 的变量 (宋体)。
 XX=04 在指定位置显示 16X32 的变量 (arial 体)。
 XX=05 在指定位置显示 24X48 的变量。
 XX=06 在指定位置显示 32X64 的变量。
- (4) “E01C” 颜色位, 其中 “E0” 表示前景颜色。“1C” 表示背景颜色。
- (5) “0000” X 轴坐标, 以像素为单元。
- (6) “0005” Y 轴坐标, 以行为单元。
- (7) “0065” 变量的值, 这里的值可以是 16 进制, 也可以是十进制, 但以十进制方式显示。
- (8) “04” 域宽, 指定变量显示的宽度, 以字符为单位, 不足则在前面补 0。取值范围是 1~5。

请注意：使用这条指令时，确保变量值小于 12900（十进制），否则可能会引起程序运行错误。比如，要显示的变量值为 12900，取整后的结果是 129 即 0x81，这与开始指令代码冲突，造成程序运行错误。

3.9 显示位图（显示预存于Nor Flash的位图）

功能：在指定起点的位置显示预存在Nor Flash的位图。

十六进制指令参数举例说明：

81 4453 E01C 0000 0000 03 84

这条指令的功能是在（0，0）位置开始显示序号为 3 的位图。

指令解释为：

- （1）“81....84” 通信的握手信号。
- （2）“4453” 显示位图的指令代码
- （3）“E01C” 颜色位，颜色位在这条指令不起作用，但一定要发送。
- （4）“0000” X 轴坐标，以像素为单元。
- （5）“0000” Y 轴坐标，以行为单元。
- （6）“03” 预置并且已下载到控制板上的位图的序号，0 表示第一幅位图。序号取值范围 0~255，但序号超过 100 时需用两个字节表示。下载位图到 WQ-VGA 驱动板的方法请看第四章“仿真软件的使用”。

3.10 显示位图（显示预存于Nand Flash的位图）

功能：在指定起点的位置显示预存在Nand Flash的位图。

十六进制指令参数举例说明：

81 4553 E01C 0000 0000 03 84

这条指令的功能是在（0，0）位置开始显示序号为 3 的位图。

指令解释为：

- （1）“81....84” 通信的握手信号。
- （2）“4553” 显示位图的指令代码
- （3）“E01C” 颜色位，颜色位在这条指令不起作用，但一定要发送。
- （4）“0000” X 轴坐标，以像素为单元。
- （5）“0000” Y 轴坐标，以行为单元。
- （6）“03” 预置并且已下载到控制板上的位图的序号，0 表示第一幅位图。序号取值范围 0~255，但序号超过 100 时需用两个字节表示。下载位图到 WQ-VGA 驱动板的方法请看第四章“仿真软件的使用”。

说明：图片预存于 Nor-Flash (39VF1601) 和预存于 Nand-Flash (K9F1208U0B)，调用显示的时间稍有不同，图片预存于 Nor-Flash 显示速度要比预存于 Nand-Flash 快，

如下表所示:

图片预存于 Nor_Flash		图片预存于 Nand_Flash	
图片像素	显示时间（一张）	图片像素	显示时间（一张）
120×40	6.5ms	120×40	6.5ms
320×240	22ms	320×240	66.6ms
800×600	122ms	800×600	416ms
1024×768	208ms	1024×768	675ms

3.11 清除全屏

功能： 以指定的背景色清全屏。

十六进制指令参数举例说明：

81 434C E01C 84

这条指令的功能是清全屏为绿色。

指令解释为：

- (1) “81....84” 通信的握手信号。
- (2) “434C” 清全屏的指令代码。
- (3) “E01C” 颜色位，其中 “E0” 表示前景颜色。“1C” 表示背景颜色。前景颜色在这条指令不起作用，但一定要发送。

3.12 绘制矩形

功能： 在指定位置绘制带填充色的指定长宽的矩形；无边框，填充色为背景色。

十六进制指令参数举例说明：

81 4358 E01C 0000 0000 0040 0014 84

这条指令的功能是绘制以(0000H, 0000H)为起始位置，长为 0040H（64）像素，高为 0014H（20）行的矩形，矩形的填充色为 1C 即绿色。

指令解释为：

- (1) “81....84” 通信的握手信号。
- (2) “4358” 绘制矩形的指令代码。
- (3) “E01C” 颜色位，前景色“E0”在这条指令不起作用。
- (4) “0000” X 轴坐标，以像素为单元。
- (5) “0000” Y 轴坐标，以行为单元。
- (6) “0040” 矩形的长（宽），以像素为单元。
- (7) “0014” 矩形的高，以行为单元。

3.13 反色矩形

功能： 反色指定位置和指定长宽的矩形

十六进制指令参数举例说明：

81 434E E01C 0000 002A 0050 003f 84

这条指令的功能是反色以 (0000H,002AH) 为起始位置, 长为 0050H (80) 像素, 高为 003fH (63) 行的矩形。

指令解释为:

- (1) “81....84” 通信的握手信号。
- (2) “434E” 反色矩形的指令代码。
- (3) “E01C” 颜色位, 在这条指令不起作用, 但一定要发送。
- (4) “0000” X 轴坐标。
- (5) “0000” Y 轴坐标。
- (6) “0050” 矩形的长 (宽), 以像素为单元。
- (7) “003f” 矩形的高, 以行为单元。

3.15 绘制圆形

功能: 在指定的位置绘制带填充色的圆或椭圆, 无边框, 填充色为背景色。

十六进制指令参数举例说明:

81 4445 E01C 0100 0100 001E 0032 84

这条指令的功能是绘制一个以(0064H,0064H)为圆心坐标, 001EH (30) 为 X 轴半径, 0032H (50) 为 Y 轴半径的椭圆, 椭圆的填充色为 1C 即绿色。当 X 轴半径与 Y 轴半径相等时, 显示为一个正圆。

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “4445” 显示圆或椭圆的指令代码。
- (3) “E01C” 颜色位, 其中 “E0” 表示前景颜色, 这里不起作用。“1C” 表示背景颜色, 这里是椭圆的填充色。
- (4) “0064” 圆心的 X 轴坐标, 以像素为单元。
- (5) “0064” 圆心的 Y 轴坐标, 以行为单元。
- (6) “001E” 圆的 X 轴半径, 以像素为单元。
- (7) “0032” 圆的 Y 轴半径, 以行为单元。

3.16 绘制直线

功能: 在指定的位置绘制直线。

十六进制指令参数举例说明:

81 444C E01C 0005 0005 0064 0064 0002 84

这条指令的功能是绘制一条以 (0005H,0005H) 为起点坐标, (0064H,0064H) 为终点坐标, 宽度为 0002H 的直线,

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “444C” 绘制直线的指令代码。
- (3) “E01C” 颜色位，其中“E0”表示前景颜色，即线条颜色。“1C”表示背景颜色，在这里不起作用。
- (4) “0005” 直线的起点的 X 轴坐标。
- (5) “0005” 直线的起点的 Y 轴坐标。
- (6) “0064” 直线的终点的 X 轴坐标。
- (7) “0064” 直线的终点的 Y 轴坐标。
- (8) “0002” 直线的宽度，以像素为单元。

3.17 开/关时间显示

功能：打开或关闭时间显示功能。

十六进制指令参数举例说明：

81 4454 E01C 0000 0000 01 84

这条指令的功能是打开自动显示时间，且显示时间的位置是（0000H,0000H），显示的方式是 01。

指令解释为：

- (1) “81.....84” 通信的握手信号。
- (2) “4454” 显示时间的指令代码。
- (3) “E01C” 颜色，其中“E0”表示前景颜色，即时钟颜色。“1C”表示背景颜色，这里不起作用。
- (4) “0000” X 轴坐标。
- (5) “0000” Y 轴坐标。
- (6) “01” 显示方式代码。显示方式控制如下表：

显示方式代码 (十进制)	对应的时间显示格式
0	关闭时间显示功能。
1	打开时间显示功能，显示方式为： “时：分：秒”，字符大小为 16X16。
2	打开时间显示功能，显示方式为： “时：分”，字符大小为 16X16。
3	打开时间显示功能，显示方式为： “年 月 日 时：分：秒”，字符大小为 16X16。
4	打开时间显示功能，显示方式为： “年 月 日 时：分”，字符大小为 16X16。

5	打开时间显示功能，显示方式为： “月 日 时：分”， 字符大小为 16X16。
11	打开时间显示功能，显示方式为： “时：分：秒”， 字符大小为 32X32。
12	打开时间显示功能，显示方式为： “时：分”， 字符大小为 32X32。
13	打开时间显示功能，显示方式为： “年 月 日 时：分：秒”， 字符大小为 32X32。
14	打开时间显示功能，显示方式为： “年 月 日 时：分”， 字符大小为 32X32。
15	打开时间显示功能，显示方式为： “月 日 时：分”， 字符大小为 32X32。

注： 1. 这里只有 10 种显示方式。如客户需要其它的显示方式可向我公司定制。
2. 当关闭时间显示功能时还可看到时间显示，这时只需刷新（换屏或清屏）即可。

3.18 设置时间

功能：调整实时时间。

十六进制指令参数举例说明：

81 5354 E01C 04 08 16 00 00 00 01 84

这条指令的功能是将时间设置为 04 年 08 月 22 日 00:00:00 星期一，注：“04 08 16 00 00 00 01” 均为 16 进制，应用时用十进制较为方便。

指令解释为：

- (1) “81....84” 通信的握手信号。
- (2) “5354” 是设置时间的指令代码。
- (3) “E01C” 颜色，在这条指令不起作用，但一定要发送。
- (4) “04” 年。
- (5) “08” 月。
- (6) “16” 日。
- (7) “00” 时。
- (8) “00” 分。
- (9) “00” 秒。
- (10) “01” 星期。

3.19 请求显示终端发送年月日

功能：单片机请求从显示终端读取年月日。

十六进制指令参数举例说明：

81 5244 E01C 84

显示终端在接收到这条指令后将年月日送至其输出缓冲区。

如果用户用的是**并行通信方式**，则显示终端将 DATA_INT 引脚置低电平以表示数据就绪，用户系统（单片机）可以通过查询 DATA_INT 脚状态或通过中断方式响应显示终端的请求，读取年月日三字节数据。

如果用户用的是**串行通信方式**，则显示终端接到以上指令后将通过串行口连续发送四个字节数据到用户系统（单片机），**第一个字节为 0xF8（代表发送的是年月日数据），后面三个字节分别是年、月、日。**用户系统（单片机）可通过串行中断方式接收数据。

指令解释为：

- （1）“81....84” 通信的握手信号。
- （2）“5244” 读取年月日的指令代码。
- （3）“E01C” 颜色，在这条指令不起作用，但一定要发送。

3.20 请求显示终端发送时分秒

功能：单片机请求从显示终端读取时分秒。

十六进制指令参数举例说明：

81 5254 E01C 84

显示终端在接收到这条指令后将时分秒送至其输出缓冲区。

如果用户用的是**并行通信方式**，则显示终端将 DATA_INT 引脚置低电平以表示数据就绪，用户系统（单片机）可以通过查询 DATA_INT 脚状态或通过中断方式响应显示终端的请求，读取时分秒三字节数据。

如果用户用的是**串行通信方式**，则显示终端接到以上指令后将通过串行口连续发送四个字节数据到用户系统（单片机），**第一个字节为 0xF9（代表发送的是时分秒数据），后面三个字节分别是时、分、秒。**用户系统（单片机）可通过串行中断方式接收数据。

指令解释为：

- （1）“81....84” 通信的握手信号。
- （2）“5254” 读取年月日的指令代码。
- （3）“E01C” 颜色，在这条指令不起作用，但一定要发送。

3.21 自动显示位图：

功能：以指定的时间间隔自动循环地在指定的位置显示位图。巧用这一功能可很方便地做出动画效果，而且不占用用户单片机的时间，特别是在多级菜单应用中。

十六进制指令参数举例说明：

81 5A44 E01C 0000 0000 00 03 02 84

这条指令的功能是打开自动显示位图的功能，自动循环显示序号为 00~03 的位图。四幅位图轮流显示，显示时间间隔为 02。

指令解释为：

- (1) “81.....84” 通信的握手信号。
- (2) “5A44” 是显示时间的指令代码。
- (3) “E01C” 颜色, 颜色位在这条指令不起作用。
- (4) “0000” X 轴坐标。
- (5) “0000” Y 轴坐标。
- (6) “00” 位图序号 N1, 循环的起点。
- (7) “03” 位图序号 N2, 循环的终点。
- (8) “02” 时间间隔 (循环的速度), 当时间间隔为 00 时表示关闭自动显示位图功能。

3.22 显示波形:

功能: 在指定的坐标原点上根据接收到的数据 (每个点的 y 轴幅值), 产生相应波形。

以点描法方式显示波形。接收到的数据为该点的 y 轴幅值 (相对与原点), 每接收到一字节数据后 x 轴坐标自动加 1。

十六进制指令参数举例说明:

81 5353 E01C 0000 012C 01..... 84

这条指令的功能是以(0000,012C)即十进制(0,300)为坐标原点, 显示接收到的数据“.....”的波形。01 表示每个点的 y 轴幅值 (即波形的数据: “.....”) 用一个字节表示。

指令解释为:

- (1) “81.....84” 通信的握手信号。
- (2) “5353” 是显示波形指令代码。
- (3) “E01C” 颜色, 只有前景颜色 E0 在这条指令起作用。
- (4) “0000” X 轴坐标。
- (5) “012C” Y 轴坐标。
- (6) “01” 幅值的数据类型。“01” —— 用一个字节表示, “02” —— 用两个字节表示。如果要显示幅度大于 100 的波形, 每个点的 y 轴幅值必须用两个字节表示。
- (7) “.....” 每个点的 y 轴幅值, 这些数据由用户系统 (单片机) 采集得到, 例如 A/D 转换等等。

特别注意: 执行完这条指令后必须加上 0.1 秒的延时, 否则下一条指令将会被忽略, 程序无法正常运行 (若使用的是串口通信方式可以不加延时)。

应用提示: 这条指令并非实时性指令。但若需要将采集到的数据实时显示, 可以每采集到一定数据后 (比如 10 个 A/D 转换结果), 用该指令发送给 VT-TFT3224 显示, 借此操作实现实时性。

3.23 打开光标并设置光标的位置

功能：打开光标并设置光标的位置，也可用这条指令关闭光标显示功能。

十六进制指令参数举例说明：

81 4355 E01C 0005 0008 01 84

这条指令的功能是打开光标显示功能，并设置光标的坐标位置为(0005H,0008H)。

指令解释为：

- (1) “81.....84” 通信的握手信号。
- (2) “4355” 打开光标的指令代码。
- (3) “E01C” 颜色位，在这条指令中不起作用。但一定要发送。
- (4) “0005” X 轴坐标。
- (5) “0008” Y 轴坐标。
- (6) “01” 光标的开关，01 打开光标显示，00 关闭光标显示。

3.24 保存窗口：

功能：保存当前窗口至后台（VT_TFT3224 的 RAM），以便调用恢复。当切换到其他窗口而需要保存当前画面以供调用时，可以使用此指令，最多能保存两个窗口。

十六进制指令参数举例说明：

81 4344 E01C 01 84

指令解释为：

- (1) “81.....84” 通信的握手信号。
- (2) “4344” 保存窗口的指令代码。
- (3) “E01C” 颜色位，在这条指令中不起作用，但一定要发送。
- (4) “01” 后台编号，有“01”、“02”两个后台供用户使用。

3.25 恢复窗口：

功能：恢复之前保存在后台的窗口。

十六进制指令参数举例说明：

81 4544 E01C 02 84

指令解释为：

- (1) “81.....84” 通信的握手信号。
- (2) “4544” 恢复窗口的指令代码。
- (3) “E01C” 颜色位，在这条指令中不起作用，但一定要发送。
- (4) “02” 后台编号，有“01”、“02”两个后台供用户使用。这里指的是恢复保存于第 2 个后台的窗口。

3.26 打开蜂鸣器：

功能：恢复打开蜂鸣器。

十六进制指令参数举例说明：

81 4242 02 84

指令解释为:

- (5) “81.....84” 通信的握手信号。
- (6) “4242” 打开蜂鸣器的指令代码。
- (7) “02” 打开蜂鸣器的时间, 此值越大蜂鸣器响的时间越长。

四. 仿真调试软件的安装及使用

4.1 软件的安装:

1. 将说明书压缩文件中的 VLCD 目录拷贝到 C 盘根目录下(无论操作系统安装于哪个磁盘)。

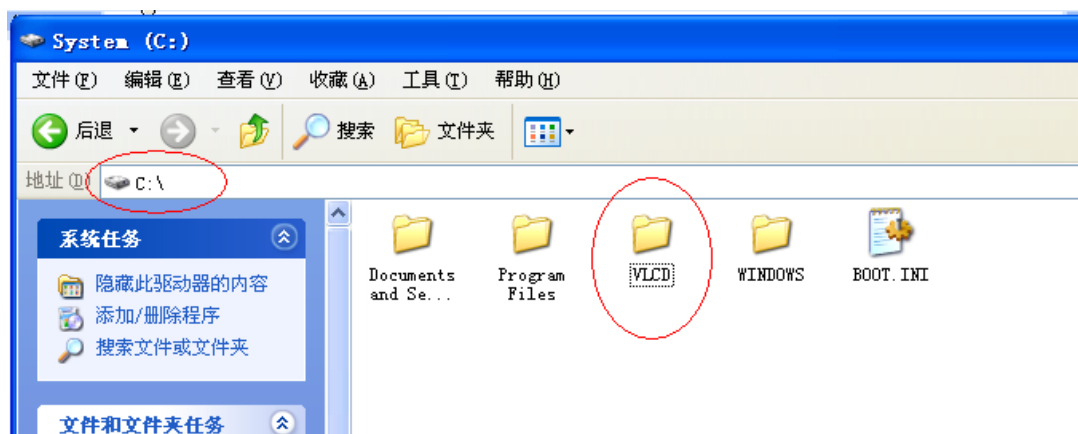


图 4.1 VLCD 目录

2. 将 VT_TFT8060 驱动板通过专用串口线与 PC 机连接, 接上电源。

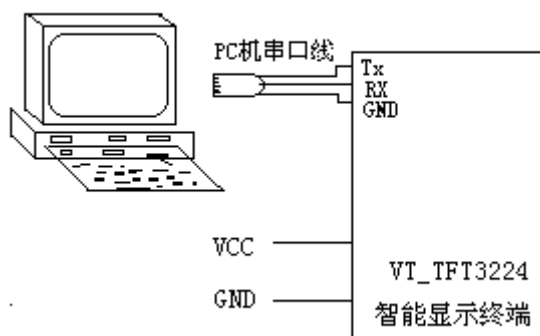


图 4.2 VT_TFT3224 与 PC 机连接进行下载图片或仿真显示

(VT_TFT3224 配有 PC 机专用串口线(3 线 9 脚), 此线内含 R232 电平转换芯片)

注: 来自单片机的 TTL 电平信号和来自 PC 机的 $\pm 15V$ 电平信号须经过“MAX3232”等芯片进行电平转换之后, 才能使 PC 机和单片机进行通信。


3. 执行 VLCD 目录中的  VT_TFT3224.exe。其运行界面如图 4.3 所示。



图 4.3 VT-TFT8060 仿真软件

4. 当在视频输出屏幕上看到“请输入显示指令”时点击【连线测试】，仿真软件右上位置将会提示“Uart 联机成功”。如果提示“Uart 联机失败”，请检查串口连接是否正确，端口选择是否正确。

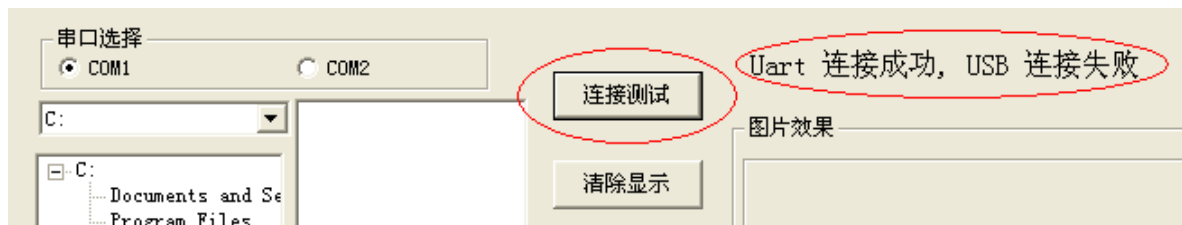


图 4.4 联机测试

提示：

VT-TFT8060 驱动板配有 USB 下载功能（“可选”），在串口通信正常的情况下用配备的 USB 线将 WQ-VGA 驱动板与 PC 连接，安装成功 USB 驱动程序后点击【连接测试】将提示“Uart 联机成功，USB 联机成功”，此时用户可以通过 USB 接口高速下载图片、字库到 WQ-VGA 驱动板。

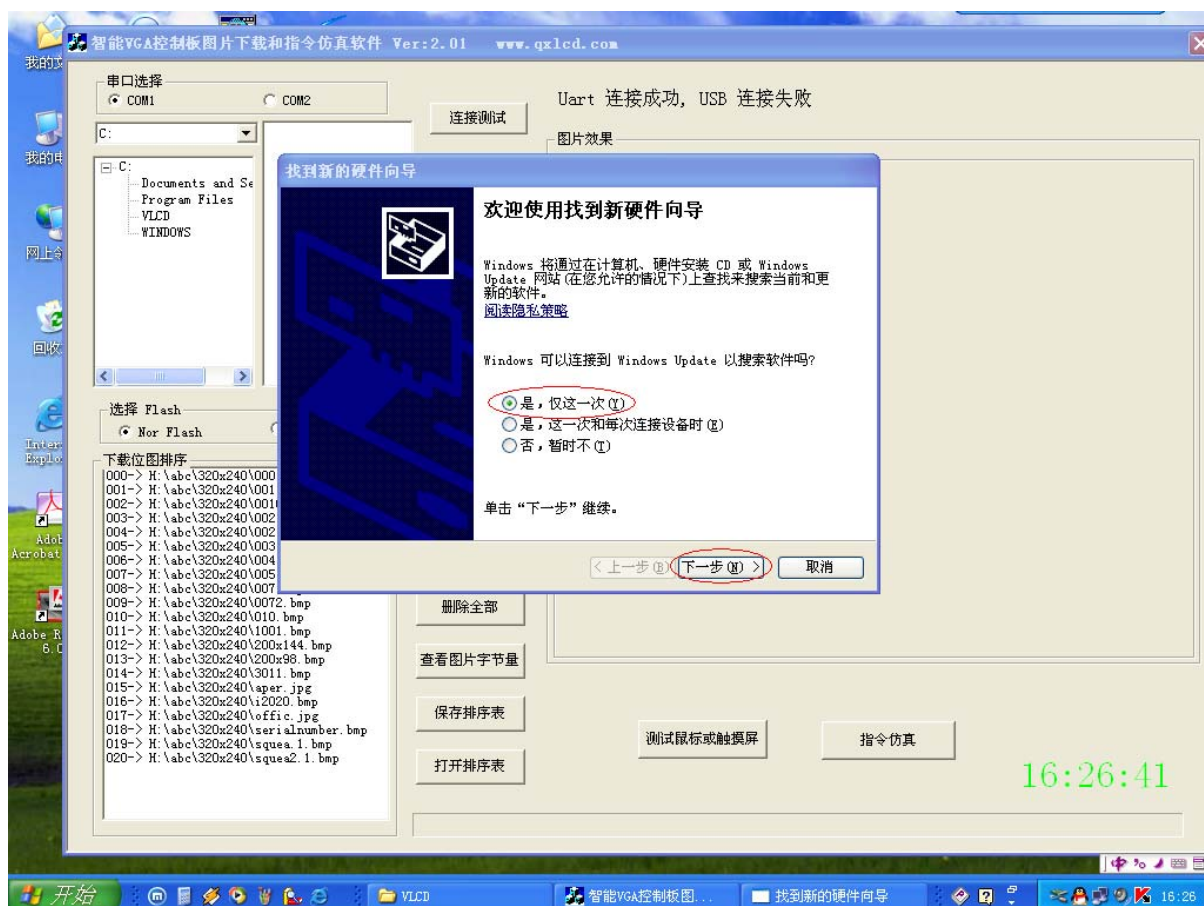


图 4.5 USB 驱动程序安装

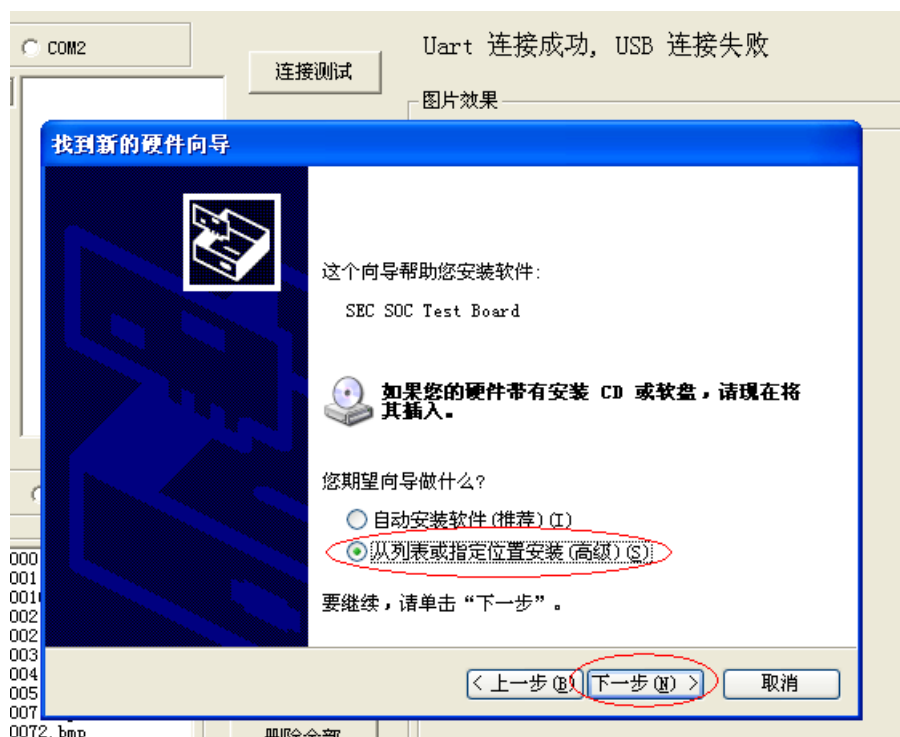


图 4.6 USB 驱动程序安装

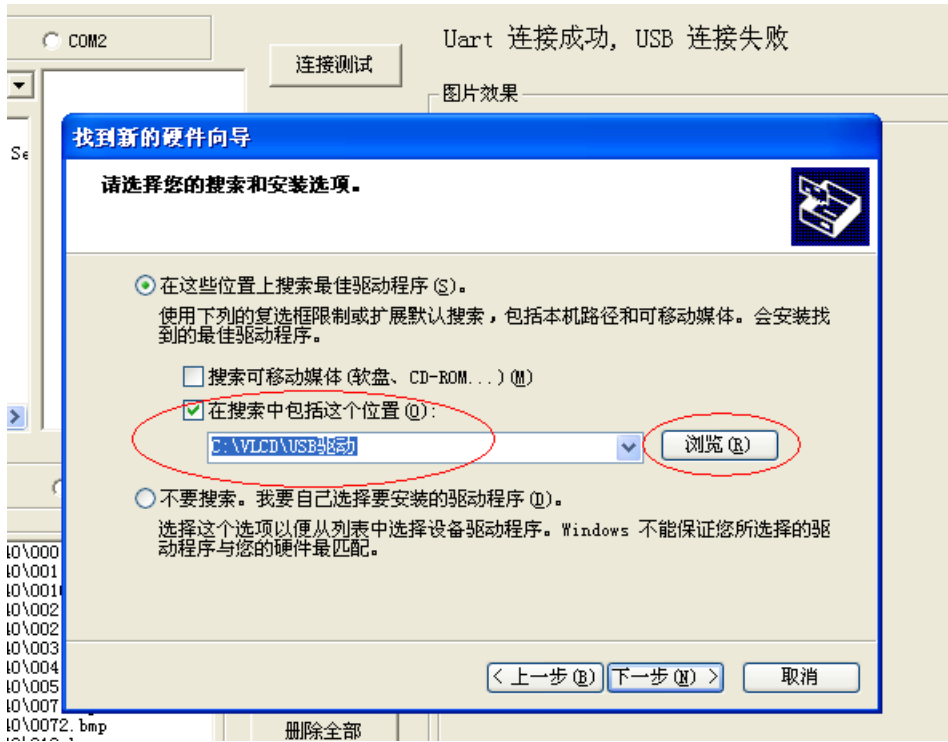


图 4.7 USB 驱动程序安装



图 4.8 USB 驱动程序安装



图 4.9 USB 驱动程序安装

5. 联机成功后就可对控制板进行模拟仿真。

4.2 下载位图：

如图 4，首先在①文本框找到位图存放的目录，在②依次选择所要添加的位图，单击【添加】，在③文本框会显示已添加的位图；若想去掉其中某一幅位图，在③中选择要删除的位图，点击【删除】；若要删除全部位图则点击【重排】；全部位图添加完成后可点击【查看字节量】查看需下载位图的大小是否超过 FLASH 的容量，要下载的位图的字节总容量压缩前不能超过 14.1MB 压缩后不能超过 1.63MB，添加完成后，单击【下载位图】。注意，一经点击【下载位图】就会把上一次下载的位图全部擦除。

注：显示终端并非支持任意格式的图片，支持的图片格式请看第七章“有效管理图片”内容。

五. 显示终端与单片机的通信

仿真软件只是起到下载图片测试指令的作用，用户最终需将显示终端脱离电脑通过串行口或并行口接入到用户的控制电路(一般为单片机电路)中运行，其中并口可以是直接总线方式（就像操作外部 RAM 一样的时序），或间接方式（用普通的 IO 口操作）。

发送指令时无论是串口还是并口，指令的格式、顺序、参数都一样。但是从显示终端读数据时串口和并口就有点不同。

5.1 显示终端与单片机 AT89C51 的串口连接通信

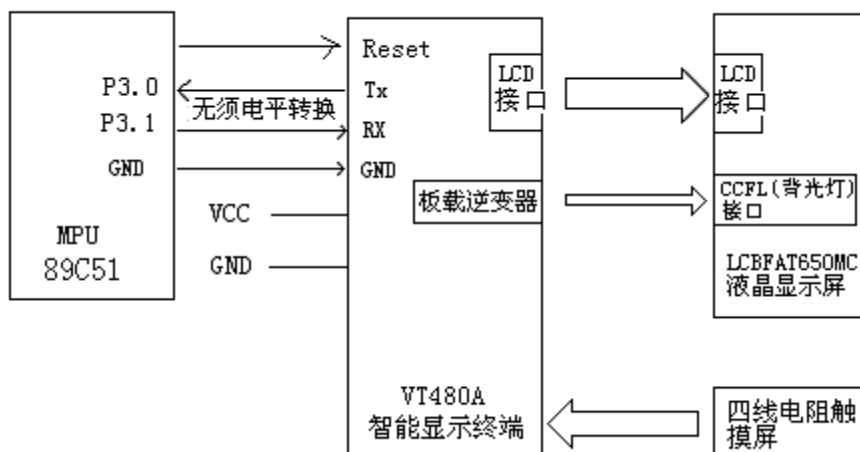


图 2.6 VT_TFT8060 与 51 单片机（或其它系列单片机）串口通信方式连接图

电路的连接如上图所示，在编写单片机 AT89C51 的显示控制软件时需注意以下事项：

- (1) 串行口模式设为模式 1（1 个起始位，8 个数据位，1 个停止位）。
- (2) 波特率设为 19200（TH1=TL1=0FDH，SMOD=1，晶振为 11.0592MHz）。
- (3) 在通信过程中，主要是单片机向显示终端发送控制数据（指令）。有几种情况需要从显示终端读数据。

- 显示终端检测到触摸屏被按下时，显示终端自动连续地发送五个字节到串行口：第一个是 0xF2（表示发送的是触摸屏坐标数据），第二、第三个是 X 轴坐标，第四、第五个是 Y 轴坐标，在触笔离开触摸屏后将发送第六个字节，为 0xF3（注：利用这个特征可以在程序中实现按键检测功能）。
- 当面板上的按钮被按下时，显示终端自动连续地发送三个字节到串行口：第一个是 0xF2，第二个是 0XF2（表示按键事件）、第三个是按键编号（0x0F—KEY1, 0x17—KEY2, 0x1B—KEY3, 0x1D—KEY4, 0x1E—KEY5），在松开按键后将发送第四个字节，为 0xF3（注：利用这个特征可以在程序中实现按键检测功能）。
- 显示终端收到读取年月日指令时，连续地发送五个字节的数据到单片机：第一个是 0xF8（表示发送的是年、月、日、星期数据），第二个是年，第三个是月，第四个是日，第五个是星期。
- 显示终端收到读取时分秒指令时，连续地发送四个字节的数据到单片机：第一个是 0xF9（表示发送的是时、分、秒数据），第二个是时，第三个是分，第四个是秒。

（4）串行接口 B（光偶隔离）使用说明：

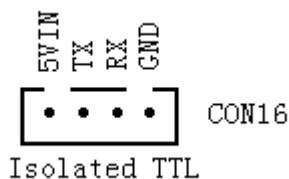


图 2.7 VT_TFT8060 带光电隔离的串口通信接口定义

为了使系统运行更加稳定，建议单片机用户板与 LCD 显示终端串行通信采用带光电隔离的串行接口。要做到真正意义上的光电隔离，单片机用户板就必须采用另一组电源，即 LCD 显示终端和用户板地线和电源都应该是分开的。CON16 的 5VIN 和 GND 分别接用户板的 5V 电源和地。

具体的串口操作例程请看” C51 例程” 文件夹下的” Vterminal-uart.c”。

5.2 显示终端与单片机 AT89C51 的并口连接通信

并口通信有两种操作方式：一是总线通信方式，就像操作普通外部 RAM 一样的时序。二是普通 I/O 口通信方式，控制信号采用普通 I/O 口操作。

5.2.1 总线通信方式：

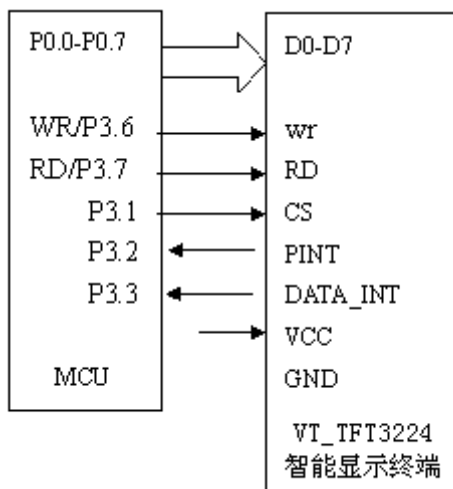


图 2.8 VT_TFT8060 与 51 单片机(或其它系列单片机)的并行总线通信方式连接图

总线通信方式的电路的如上图所示，即把显示终端当作一个普通的外部存储器进行操作。操作时序与操作普通外部 RAM 相同，只是忽略了地址总线，即把 VT_TFT3224 当做是一个没有地址的外部 RAM 来操作。例如：

```
MOV    A,#81H
MOVX   @DPTR,A
```

执行以上指令时，不管 16 位地址寄存器 DPTR 是什么内容，执行的结果是立即数 81H 被发送到 VT_TFT3224 驱动板。

在通信过程中，主要是单片机向显示终端发送控制数据（指令）。有几种情况需要从显示终端读数据。

当单片机检测到 DATA_INT 信号线是低电平时，应马上向显示终端连续读取四个字节的数据。

（1）如果所读取的数据的第一个字节等于 0xF8 则，所读到的四个字节为年月日数据（其中第一个是 0xF8，第二个是年，第三个是月，第四个是日）。

（2）如果所读取的数据的第一个字节等于 0xF9，则所读到的四个字节为时分秒数据（其中第一个是 0xF9，第二个是时，第三个是分，第四个是秒）。

（3）如果所读取的数据的第一个字节小于 0xF0，则所读到的四个字节为触摸屏数据（其中第一，二个是X轴坐标，第三，四个是Y轴坐标）。

注：当按下触摸屏时，触摸屏事件信号输出脚“PINT”将变低电平，直到触笔离开触摸屏后才变回高电平，利用这个特征可以在程序中实现按键检测功能。

注意单片机最好为 DATA_INT 申请一个外中断，当 DATA_INT 产生中断时单片马上连续读取四个字节的数据。

具体的并行总线操作例程请看” C51 例程” 文件夹下的” Vterminal-bus.c”
具体的普通 I/O 口操作例程请看” C51 例程” 文件夹下的” VTerminal-GPIO.c”

表 3 并口引脚功能

管脚号	管脚名称	I/O	功能描述
1	GND	I	电源地
2	DINT	O	请求读数据信号，低电平有效。
3	RD	I	读选通信号输入线，低电平有效。
4	WE	I	写允许信号输入线，低电平有效。
5	PINT	O	触摸屏事件信号输出线，低电平有效。
6	CS	I	片选信号输入线，低电平有效。
7	GND	I	线路地
8	VCC	O	电源+5V 输出
9	D1	I/O	数据线 D1
10	D0	I/O	数据线 D0
11	D3	I/O	数据线 D3
12	D2	I/O	数据线 D2
13	D5	I/O	数据线 D5
14	D4	I/O	数据线 D4
15	D7	I/O	数据线 D7
16	D6	I/O	数据线 D6

- （1）DINT：出现低电平时用户控制系统（单片机）须马上向终端读取四个字节的数据。
- （2）PINT：在触摸屏被按下期间一直维持低电平，即直到触笔离开触摸屏。
- （3）RD,WE,CS 这三根线与标准 RAM 时序是一样的。

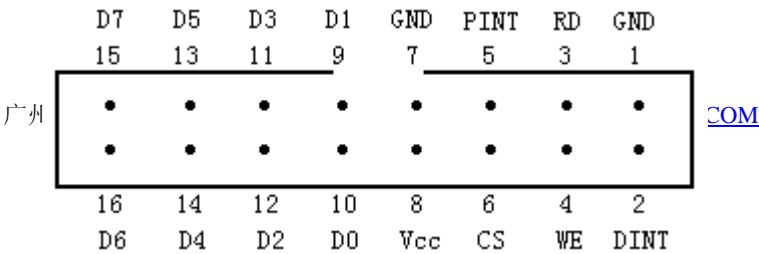


图 2.9 VT_TFT3224 并行接口定义图

5.2.1.1 总线通信方式时序图（读一次触摸屏事件）：

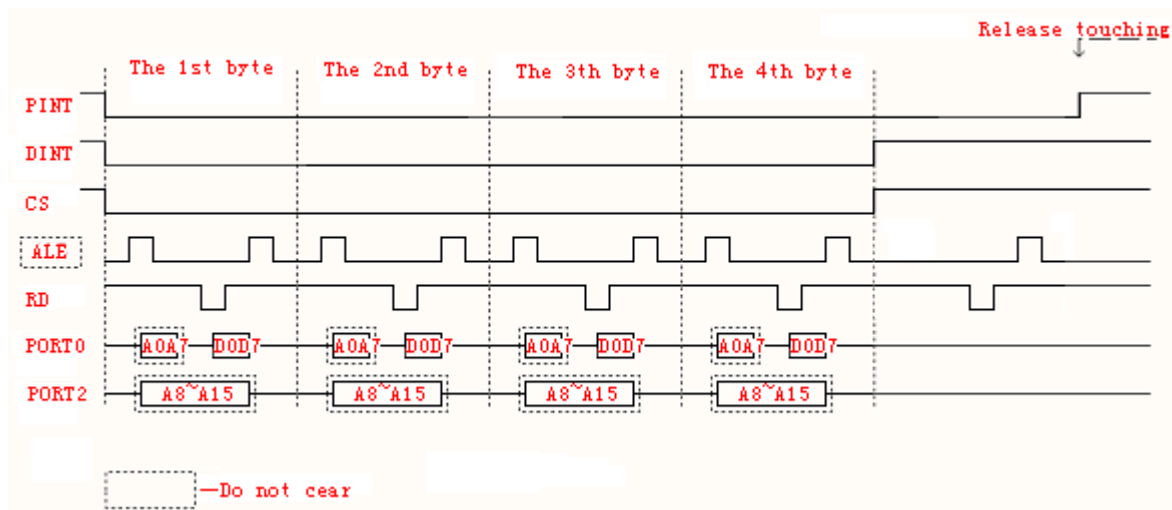


图 3.0 并行总线通信时序图

5.2.2 普通 I/O 口通信方式：

略。

六. 触摸屏的校正

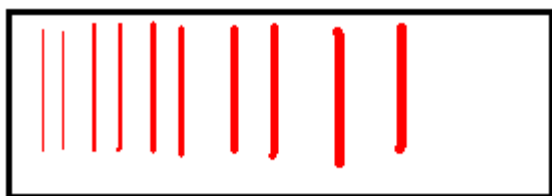
一般情况下产品在出售前都经过校准。用户也可自己校准有较大误差的触摸屏。方法是：

1. 把显示终端连接到 PC 机，并上电。
2. 打开仿真软件 VT_TFT3224.exe。
3. 单击“**校准触摸屏**”按钮。这时显示终端会显示“请按左下角的小圆”。
4. 用触摸笔按住**左下**角的小圆直到小圆消失。这时显示终端会显示“请按右上角的小圆”。
5. 用触摸笔按住**右上**角的小圆直到小圆消失。这时显示终端会显示“正在保存校准值....”。
6. 当看到显示终端显示“已保存校准值”时，校准值已保存到 Flash 里。整个校准过程完成。

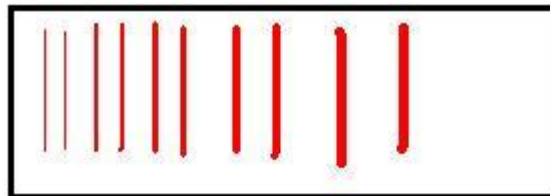
七. 有效管理图片

VT_TFT3224 支持的图片格式有两种 BMP 和 JPG。为了能得到最好的显示效果和存储更多的图片用户必须了解这两种格式的优点和缺点。

- (1) BMP 图片：VT_TFT3224 只支持 64K 色（16 位色）的 BMP 图片。用户在制作表格、按钮等类型的图片时最好保存这种格式。仿真软件在将 BMP 图片下载到 VT_TFT3224 时会作一次无损压缩。BMP 的优点是：从下面的两幅图可以看出 BMP 的小线条没有颜色失真，所以较适合制作表格和按钮。
- (2) JPG 图片：VT_TFT3224 支持通用的 JPG 图片。JPG 图片一般是 32 位色，用户在用到相片等时最好用这种格式。JPG 的缺点是：从下面的两幅图可以看出 JPG 在小线条的颜色失真较大。但是这种失真在相片上是看不出来的，因为相片是大面积着色。JPG 的优点是：压缩比高，颜色丰富。



256 色的 BMP 图片



由左图转换所得的 JPG 格式

对用户来说需遵循的规则是：表格，按钮等自己动手画的图片都最好保存成 256 色的 BMP 格式。来自于摄像头，网络的相片，照片需保存成 JPG 格式。

7.1 可以下载多少张图片的计算方法：

JPG 文件是一种图片的压缩文件，如一张 640*480 的 JPG 图片在计算机里的文件大小是 50K 字节。在显示终端机上解压后的大小是 640*480*2=614.4K 字节，如一张 640*480 的 256 位图图片，仿真软件在下载前会先进行压缩处理，在显示终端机上解压后的大小也是 640*480*2=614.4K 字节。即存放在显示终端机 Flash 上的图片都压缩处理过的图片，显示终端机在每次上电时都需要把 Flash 上的图片解压到内存（SDRAM）。因此显示终端要求所有图片在压缩后小于 1.6M 字节，解压后小于 14.1M 字节。

一般情况下可放 21 张 640*480 或 80 张 320*240 的图片。如用户需要存放更多的图片可向我公司定制更大的存储器。

八. 校验功能

由于现场干扰的方面的原因可能造成发送指令时接收到错误的数数据，主要表现为显示位置错误，花屏等等，采用校验功能可以检测接收到的数据是否正确，如果是错误侧放弃这条指令，也可以让单片机重发。

当板上电启动时默认的是无校验。可以通过发送下面这条指令打开校验功能。

指令: "81 43 4B 00 00 01 84 "。

校验功能打开后每次发指令必须在原指令的基层上加上两个校验和字节。

如原指令:

"81 44 57 E0 1C 00 00 00 00 33 32 84"

加上校验后变为:

"81 44 57 E0 1C 00 00 00 00 33 32 XH XL 84"

即将原指令 84 前的 "81 44 57 E0 1C 00 00 00 00 33 32" 11 个字节相加后的和 XX 分成两个字节插入到 84 前。

11 个字节的和是 0x27D, 十进制为 637, 所以跟坐标的计算方法一样 XH=06, XL=25。

即加上校验后要发送的指令是:

"81 44 57 E0 1C 00 00 00 00 33 32 06 25 84"

打开校验之后应注意: 显示字符的指令不能太长, 一般一条指令只能显示 40 个字符。

下面是增加校验功能后对程序的修改:

```
unsigned short check_add;    //定义一个全局变量, 校验和的值。
unsigned char  check_err;    //定义一个全局变量, 标志是否有志校验和出错
//写一个数据子程序:
Write_Byte(uchar dc_data)
{
    uchar xdata DC;    //定义一个外部 RAM 变量
    CS0 = 0;
    DC = dc_data;      //数据 dc_data 写到外部 RAM (即 LCD 控制板)。
    CS0 = 1;
    check_add += dc_data; //
}
//写一个数据子程序:
Write_Byte2(uchar dc_data)
{
    uchar xdata DC;    //定义一个外部 RAM 变量

    CS0 = 0;
```

```
    DC = dc_data;    //数据 dc_data 写到外部 RAM（即 LCD 控制板）。
    CS0 = 1;
}
//画圆子程序。
DIS_Ellipse(unsigned short x0,unsigned short y0,unsigned short xr,unsigned short yr)
{
    uchar i;

dis_ellipse_start:
    check_add = 0;
    Write_Byte(0x81);
    Write_Byte(0x44);
    Write_Byte(0x45);
    Write_Byte(Fcolor);
    Write_Byte(Bcolor);
    Write_Byte(x0/100);
    Write_Byte(x0%100);
    Write_Byte(y0/100);
    Write_Byte(y0%100);
    Write_Byte(xr/100);
    Write_Byte(xr%100);
    Write_Byte(yr/100);
    Write_Byte(yr%100);
    Write_Byte(x0/100);
    if(check)
    {
        Write_Byte2(check_add/100);
        Write_Byte2(check_add%100);
    }
    Write_Byte(0x84);
    for(i= 0;i<4;i++){;}
    if(check_err) //如果标志校验和出错,重新发送一次指令。
    {
        check_err = 0;
        goto dis_ellipse_start;
    }
}
```

```

}

/*外中断接收数据处理程序*/
void Ex1_int( ) interrupt 2 using 1
{
    uchar temp1,temp2,temp3,temp4;

    temp1 = Read_Byte();
    temp2 = Read_Byte();
    temp3 = Read_Byte();
    temp4 = Read_Byte();
    if(temp1 == 0xF8)      //如果接收到的第一个字节是 0xF8, 则收到的数据是年月日。
    {
        Date[0] = temp1;
        Date[1] = temp2;
        Date[2] = temp3;
        Date[3] = temp4;
    }
    else if(temp1 == 0xF9) //如果接收到的第一个字节是 0xF9, 则收到的数据是时分秒。
    {
        Date[4] = temp1;
        Date[5] = temp2;
        Date[6] = temp3;
        Date[7] = temp4;
    }
    else if(temp1 == 0xFA) //校验和出错
    {
        check_err=1;//标志校验和出错
    }
    else                //收到的数据触摸屏的座标值。
    {
        CHX_DAT = temp1 * 100;
        CHX_DAT = temp2 + CHX_DAT;
        CHY_DAT = temp3 * 100;
        CHY_DAT = temp4 + CHY_DAT;
    }
}

```

}
}