华泓视讯技术有限公司

HH-DP-VC 液晶显示控制板技术手册

HH-DP-VC 是 一款把 模拟视频信号（CVBS）和 CPU 数字信号同时显示在 5 寸 640*480，5.6 寸 640*480，10.4 寸 640*480，7 寸 800*480 10.2 寸 800*480，8 寸 800*600，10.4 寸 800*600，12.1 寸 800*600 的数字屏上的一个控制板。此板就是为了实现用户同时想观看 CVBS 视频信号和 CPU 数字信息而设计的。视频输入通道接受的是复合视频信号，在 LCD 上显示的图象是真彩色的。CPU 通道是一个 8 位的数据总线加上几根控制线，让用户能以简单的而标准的硬件连线来实现对控制板的访问。采用 I/O 总线连接方式，可显示 256 色；为提高读写速度、简化程序，显示屏中每个点影射显示缓存中的一个字节，显示屏中的行列号与缓存器的行列号一一对应，因此，只需输入行列号，便可直接读写相应点数据，不用计算点在显示缓存中的位置。用户可以把需要做成动画的图象先写进 SRAM 的非当前显示区域,然后用一个命令就可以实现动画,也可以实现画中画的功能.用户还可以实现 OSD 方式来写屏幕(只修改需要修改的地方,不需要修改的数据保持原来的数据).用户可以用单个命令实现重复写入小于 800 或 600 个点(就是，写入一个数据，每次命令可以画一行).这个功能能够使画线和填充都可以比较快速的实现.行方向的地址递增和列方向的地址递增的变化方式让用户比较方便的绘图.用户可以把需要显示的数据写进 SRAM 显存后,以 OSD 或开窗口的方式在活动视频上显示出来，这样，就是以图形方式实现视频 OSD 的功能，这样的 OSD 功能很好，能实现复杂的普通的 OSD 芯片无法实现的复杂图案和中西文字符以及不同点阵的字符的混合显示。本控制板具有非常好的块拷贝和块移动的功能，非常方便做动画和大面积填充操作。

适配 CPU：51,96,X86,8088,Z80,DSP,ARM,AVR,MSP430 等 CPU

一、接口定义：

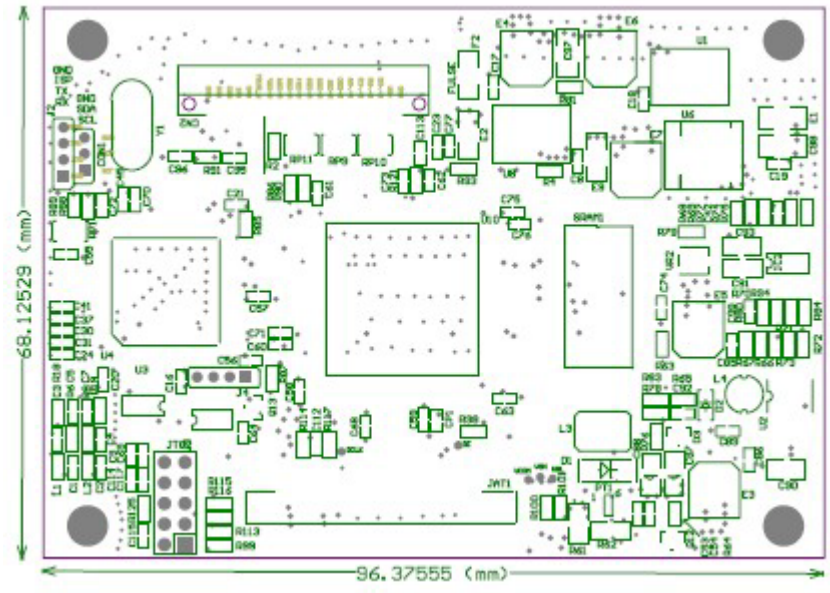
1、CPU 侧接口（DF14-20 1.25mm 间距 贴片）

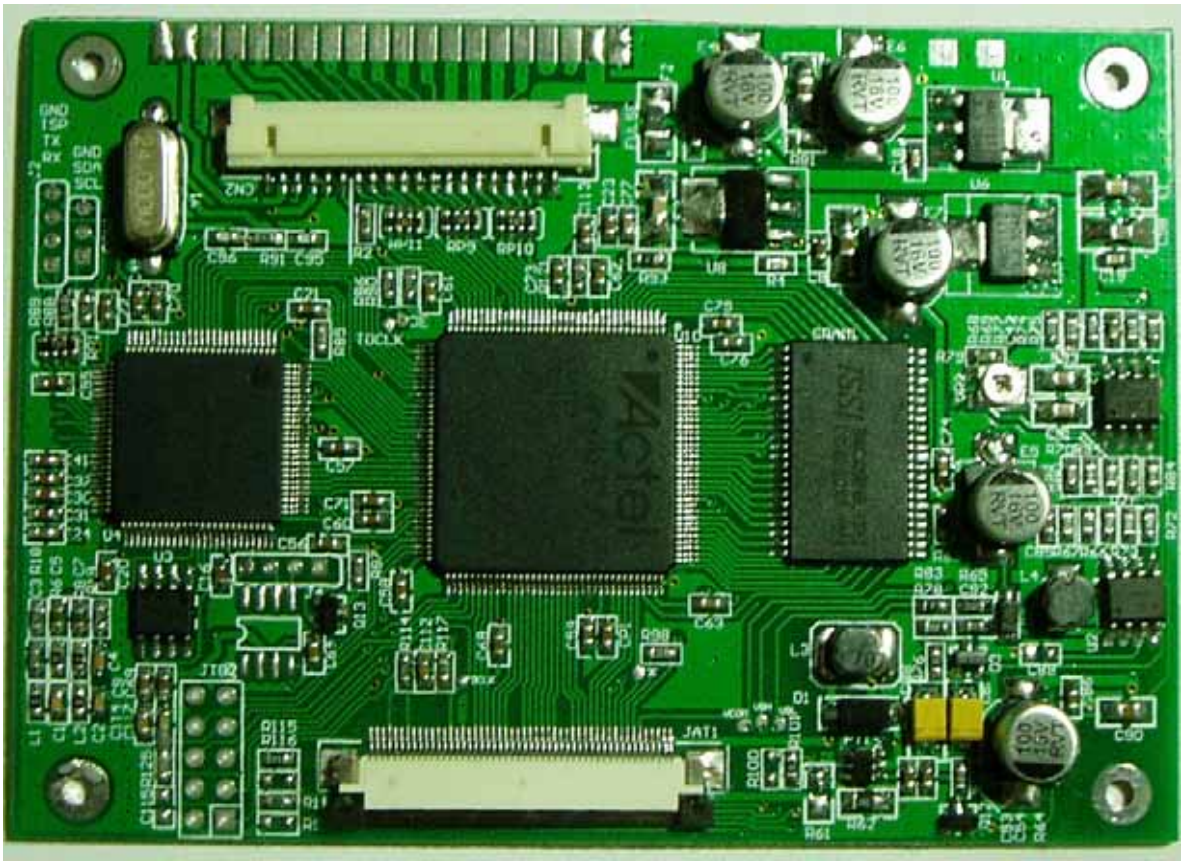
引脚	符号	功能	备注
1	GND	电源地	0V
2	5V	控制板电源电源(5-20V 定)	5V IN
3	RD	读操作信号，低电平有效	IN 3.3V
4	WR	写操作信号，低电平有效	IN 3.3V
5	CS	片选信号，低电平对屏操作有效	IN 3.3V
6	RS	1---对数据通道操作，0---写命令寄存器	IN 3.3V
7	DATA0	数据总线	INOUT3.3V
8	DATA1	数据总线	INOUT3.3V
9	DATA2	数据总线	INOUT3.3V
10	DATA3	数据总线	INOUT3.3V
11	DATA4	数据总线	INOUT3.3V
12	DATA5	数据总线	INOUT3.3V
13	DATA6	数据总线	INOUT3.3V
14	DATA7	数据总线	INOUT3.3V
15	B/R	忙闲判断	OUT3.3V
16	GND	CVBS 视频 2 的地	
17	AV2	CVBS 视频 2 信号输入	
18	GND	CVBS 视频 1 的地	
19	AV1	CVBS 视频 1 信号输入(默认是 AV1)	
20	GND	电源地	

本控制板有如下性能：(本板上已经包含背光部分，可 8 级调光)

1、 本控制板可工作于 8 位数据总线模式，A/B/C 的后缀是针对 AU，PVI，LG/SHARP 等的型号。
2、 本控制板三种写屏方式。单点写，8 点写，多点写。这三种方式都可以在行方向或列方向进行。
3、 本控制板的写周期可做到 50ns,无需任何等待。
4、 连续写入时，地址自动加 1，遇到行末自动跳到下一行。
5、 控制板支持块填充和块移动的命令，适合做动画等.

2 TTL 接口控制板尺寸





(1) CS 与 RS,WR,RD,DATA 组合功能如下：

CS	RS	DATA[7:0]	WR	RD	功能
0	0	0x00	0	1	保留
0	1	0xXX	0	1	保留
0	0	0x01	0	1	设置控制板是单点，多点，8 点及地址是水平 竖直方向的变化
0	1	0xXX	0	1	写工作模式寄存器 1
0	0	0x02	0	1	地址指向工作模式控制寄存器 2
0	1	00xXX	0	1	低 3 位是背光亮度调节参数，可 8 级调节
0	0	0x03	0	1	地址指向行地址寄存器
0	1	00xXX	0	1	写入行地址的低 8 位
0	0	0x04	0	1	地址指向列高地址寄存器
0	1	X	0	1	地址指向行地址的高 2 位和列地址的高 2 位 寄存器【5：4】=行地址的【9:8】 寄存器【1：0】=列地址的【9:8】
0	0	0x05	0	1	地址指向列的低地址寄存器
0	1	X	0	1	写入列地址的低 8 位
0	0	0x06	0	1	多点重复写时的低8位,开窗口做OSD图块显示时列起始坐标的低 8 位
0	1	0xXX	0	1	设置多点重复写时的低 8 位，设置开窗口做 OSD 图块显示时列起始坐标的低 8 位
0	0	0x07	0	1	地址指向背景色寄存器，开窗口做 OSD 图块显示时行的起始坐标
0	1	0xXX	0	1	设置背景色寄存器的值开窗口做 OSD 图块显示时行的起始坐标
0	0	0x08	0	1	地址指向多点重复写入时的高位或开窗口做 OSD 显示时读图象块的列起始坐标的高位
0	1	0xXX	0	1	写入多点重复写入时的高位或开窗口做 OSD 显示时读图象块的列起始坐标的高位
0	0	0x09	0	1	地址指针指向窗口左上角坐标的寄存器的行寄存器
0	1	0xXX	0	1	设置窗口寄存器的行寄存器的值
0	0	0x0a	0	1	地址指针指向窗口左上角坐标寄存器的列寄存器的低 8 位
0	1	0xXX	0	1	设置窗口寄存器的列寄存器的低 8 位

					值
0	0	0x0b	0	1	地址指针指向窗口左上角坐标寄存器的列寄存器的第 9 位
0	1	0xXX	0	1	设置窗口寄存器的列寄存器的第 9 位的值
0	0	0x0c	0	1	地址指针指向窗口高度寄存器
0	1	0Xxx	0	1	设置窗口的高度的值
0	0	0x0d	0	1	地址指针指向窗口宽度寄存器的低 8 位
0	1	0xXX	0	1	设置窗口的宽度低 8 位的值
0	0	0x0e	0	1	地址指针指向窗口宽度寄存器的第 9 位
0	1	0xXX	0	1	设置窗口的宽度第 9 位的值
0	0	0x0F	0	1	地址指向前景色寄存器,
0	1	0xXX	0	1	设置前景色寄存器的值
0	0	0x1F	0	1	地址指针指向数据通道寄存器
0	1	0xXX	0	1	把显示数据写进显存里
0	1	0xXX	1	0	从显存里读取显示数据
1		×	×	×	不选通

说明：在读写显示数据时,要保证指令寄存器的值设为 0x1F,选择数据通道

注意：对于本控制板，如果用户想使用读 SRAM 的数据这个功能时，要保证 RD 这个信号的低电平达到一定宽度.否则，读出的数据可能是错误的.有了读显存功能，用户可以把显存中的数据读出后取反再写入，可实现移动光标

1,工作模式控制寄存器 CONREG1 说明

控制寄存器的设置是本控制板能否使用好的重点，其定义如下：

D7	D6	D5	D4	D3	D2	D1	D0
保留	保留	保留	保留	保留	WrDir	WrMod1	WrMod0

D1D0--- 写操作屏幕的方式设置

00 单点写，读显存操作时，必须设置为单点模式

01 8 点带背景写，比如写一个字符，字符背景就用设定的背景色显示

10 多点写，设置好重复次数后和前景色后，设置这个命令可写一整行或整列

此命令适合画行列方向的直线

11 8 点忽略背景色写操作，比如写一个字符，字符的有效点阵就显示出来，无效的空格数据位置上，保留屏幕上原先显示的内容

D2------读写操作的地址变化方向

0－－－从左到右的水平方向地址变化

1------从上到下的竖直方向地址变化

2,工作模式控制寄存器 CONREG2 说明

D7	D6	D5	D4	D3	D2	D1	D0
DISP2	DISP1	DISP0	保留	PWM_ADJBL3	PWM_ADJBL2	PWM_ADJBL1	PWM_ADJBL0

D3D2D1D0---产生 PWM 脉冲来调节背光的亮度，

0000----最亮

0001 ----次亮

……

1111----关闭背光，共有 16 级调光

D4------ 保留

D7D6D5----视频与 CPU 数据显示模式设置

000----显示整屏的 CPU 产生的数据　目的是为了与没有视频信号,只有 CPU 信号的控制板兼容

001----屏幕显示整屏的活动视频信号

010----在整屏活动视频数据上开窗口显示 CPU 送到显存中的数据;CPU 的数据完全不透明显示。这个模式要求用户设定 CPU 数据显示窗口的起始位置（左上角坐标）和窗口的宽度高度参数,最多可以开 4 个 CPU 窗口，窗口的参数及窗口的打开关闭使能，参考以下寄存器：

0 号窗口　0x0706 是窗口左上角的水平方向坐标----两个字节

0x0908 是窗口左上角的竖直方向坐标----两个字节

0x0B0A 是窗口的高度

0x0D0C 是窗口的宽度

0x40=1 是打开 0 号窗口，0x40=0 是关闭 0 号窗口，

1 号窗口　0x1716 是窗口左上角的水平方向坐标----两个字节

0x1918 是窗口左上角的竖直方向坐标----两个字节

0x0B0A 是窗口的高度

0x0D0C 是窗口的宽度

0x41=1 是打开 1 号窗口，0x41=0 是关闭 1 号窗口，

2 号窗口　0x2726 是窗口左上角的水平方向坐标----两个字节

0x2928 是窗口左上角的竖直方向坐标----两个字节

0x2B2A 是窗口的高度

0x2D2C 是窗口的宽度

0x42=1 是打开 2 号窗口，0x42=0 是关闭 2 号窗口，

3 号窗口　0x3736 是窗口左上角的水平方向坐标----两个字节

0x3938 是窗口左上角的竖直方向坐标----两个字节

0x3B3A 是窗口的高度

0x3D3C 是窗口的宽度

0x43=1 是打开 3 号窗口，0x43=0 是关闭 3 号窗口，

0x0706----0x07 放地址的高字节，0x06 放地址的低字节，其他寄存器仿此

011----在整屏显存的显示背景数据上开个窗口显示活动视频；

这个模式要求用户设定 CPU 数据显示窗口的起始位置（左上角坐标）和窗口的宽度高度参数

100---- 把显存中的数据以 OSD 的方式在活动视频上显示出来，在活动视频上不需要显示出来的数据在 SRAM 中用背景色设定（先用背景色把整个 SRAM 的显示区域填满,这个操作可以使用硬件清屏来完成，请详细参考程序范例）。对于需要在活动视频上显示出来的数据，使用的是直接 SRAM 中的非背景数据。这个功能可以开任意形状的 CPU 数据图片

101----开任意形状的视频窗口，屏幕上大背景是 CPU 的数据信号显示，在与设定的背景色不同的位置上的像素就显示成视频信号

做法是： 设定一个背景色，在显存中，与此设定背景色不同值的部分，均会用视频数据代替显示出来。这样就可以实现任意形状的窗口了。

110----在整屏活动视频数据上开窗口显示 CPU 送到显存中的数据;CPU 的数据是以半透明的方式显示的

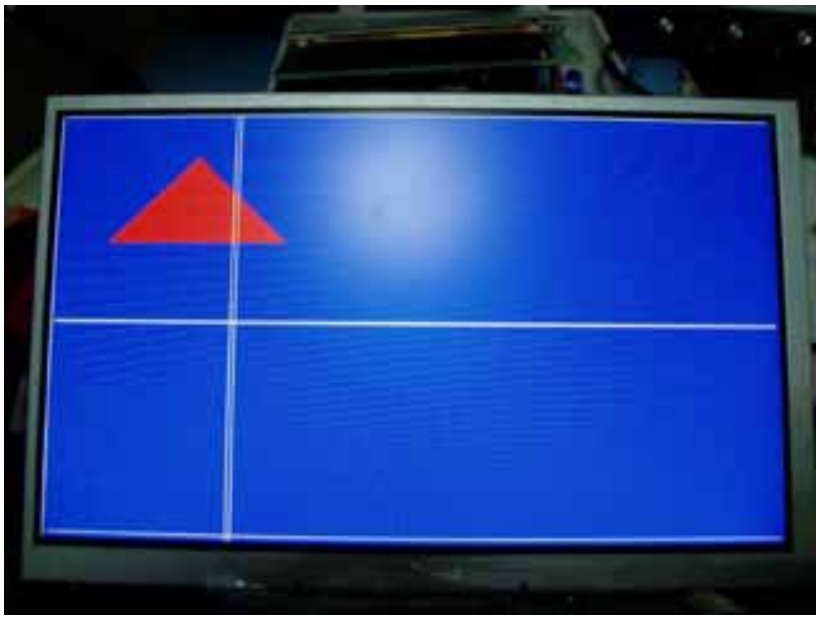
这个模式要求用户设定 CPU 数据显示窗口的起始位置（左上角坐标）和窗口的宽度高度参数

111---- 把显存中的某个区域的 CPU 数据，显示到活动视频上任意指定的坐标上

比如：一个位与显存中固定位置上的一个 BMP 图片，希望其在活动视频上的任意位置显示出来 这样最方便用户在活动视频上面 叠加活动的 bmp 图片

此功能有带完善，未调试好

D7D6D5==000 DispFullCpuDat(0x00)； 全屏显示 CPU 数据



D7D6D5=001 DispFullVideo(0x20);全屏显示视频数据

D7D6D5=010 void WindowForCpuOnVideo(unsigned int CpuWinStaRow,CpuWinStaCol,
unsigned int CpuWinHigh,CpuWinWidth,
unsigned char DispMode) ,在规定的区域内，显示 CPU 数据，区域外显示视频数据　　DispMode＝0X40

D7D6D5=011 void WindowForVideoOnCpu(unsigned int CpuWinStaRow,CpuWinStaCol,
unsigned int CpuWinHigh,CpuWinWidth,
unsigned char DispMode) ,在规定的区域内，显示视频数据，区域外显示 CPU 数据　　DispMode＝0X60

D7D6D5=100 CpuOnFullVideo(unsigned char BackColo, unsigned char DispMode)

DispMode=0x80 设定一个背景色，把显存中与背景色不一样的数据在以 OSD 方式显示在活动视频上 ，如下图：

D7D6D5=101 VideoOnFullCpu(unsigned char BackColo, unsigned char DispMode)
DispMode=0xa0 ,设定一个背景色，在显存中与背景色不一样的数据的位置上，显示为活动视频信号,，这样可实现慨任意形状的视频窗口 开多窗口

D7D6D5=110 FadingWindowForCpuOnVideo(unsigned int CpuWinStaRow,CpuWinStaCol,
unsigned int CpuWinHigh,CpuWinWidth,
unsigned char DispMode)

DispMode=0XC0

功能是把显存中指定区域的数据以半透明的形式显示在活动视频上

D7D6D5=111

```
void MovBmpOnVideo(unsigned int  BmpInSramStaRow,BmpInSramStaCol,//指定 BMP 在显存中的行列地址
                   unsigned int  BmpOnVideoStaRow,BmpOnVideoStaRow,//指定屏幕上显示BMP 的位置
                    unsigned int  BmpHigh,BmpWidth,//指定 BMP 的宽 高
                    unsigned char DispMode)
```

DispMode=0xe0

对控制板的写入方式的设计，单点写入，8 点写入，多点重复写入，是此类控制板的关键的指标。本公司设计的这种显示数据的写入方式的控制板，非常适合用户进行字符显示，图形填充，画线制表格，曲线等操作。

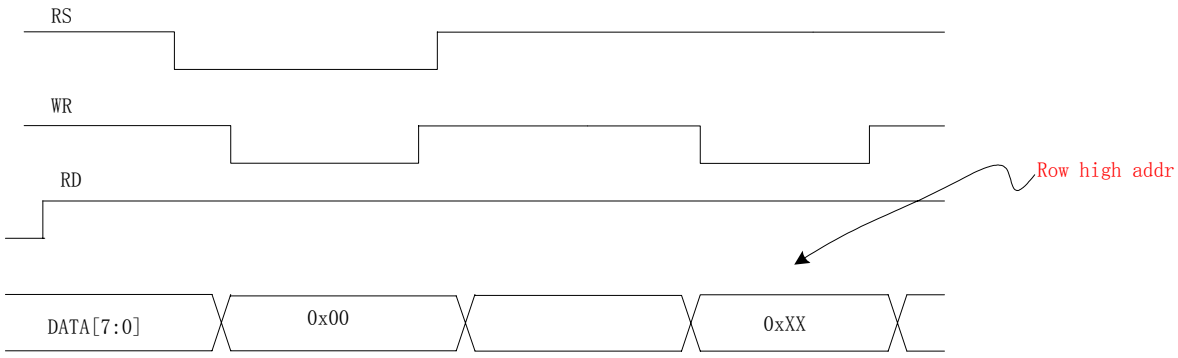
256 色的设置方法:

	D7	D6	D5	D4	D3	D2	D1	D0
256 色	R2	R1	R0	G2	G1	G0	B1	B0

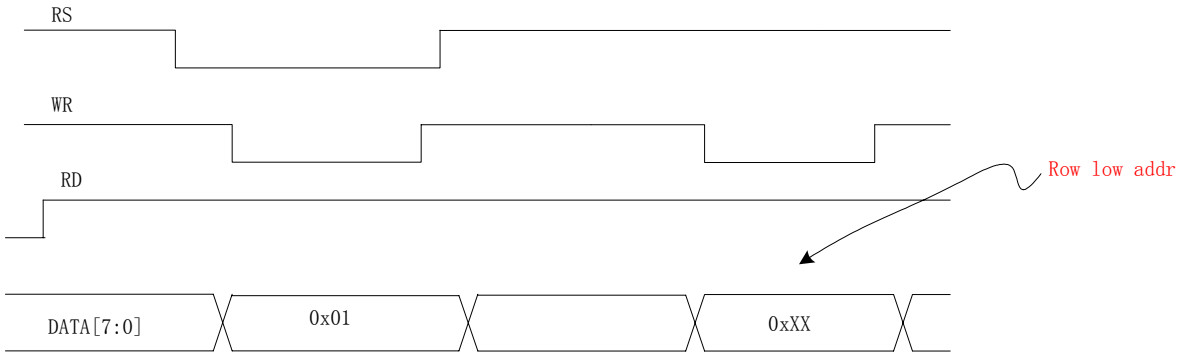
256 色

	颜色灰度	R2、R1、R0	G2、G1、G0	B1、B0
基本颜色	最黑	000	000	00
	亮蓝	000	000	11
	亮绿	000	111	00
	亮青	000	111	11
	亮红	111	000	00
	亮紫	111	000	11
	亮黄	111	111	00
	亮白	111	111	11
蓝色灰度	最黑	000	000	00
	较暗	000	000	01
	较亮	000	000	10
	最亮	000	000	11
绿色灰度	最黑	000	000	00
	较暗	000	001	00
	…	…	…	…
	较亮	000	110	00
	最亮	000	111	00
红色灰度	最黑	000	000	00
	较暗	001	000	00
	…	…	…	…
	较亮	110	000	00
	最亮	111	000	00

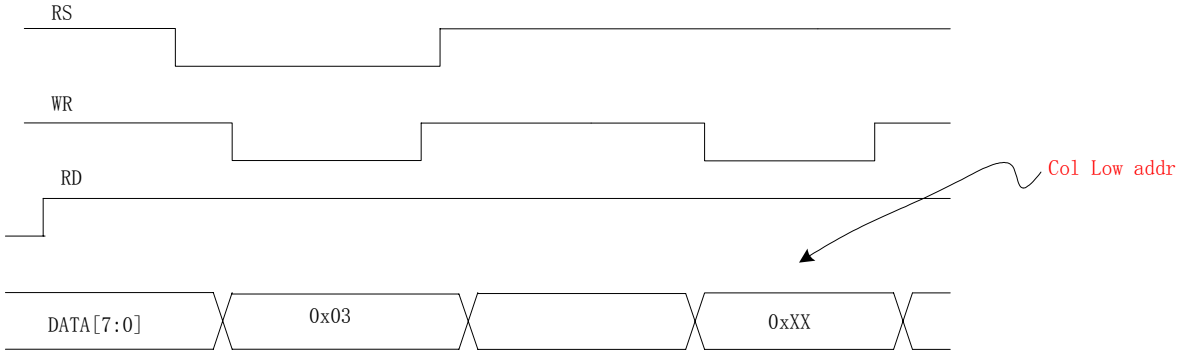
1 写工作模式寄存器 1 的时序（SetReg(0x01,0xXX)）



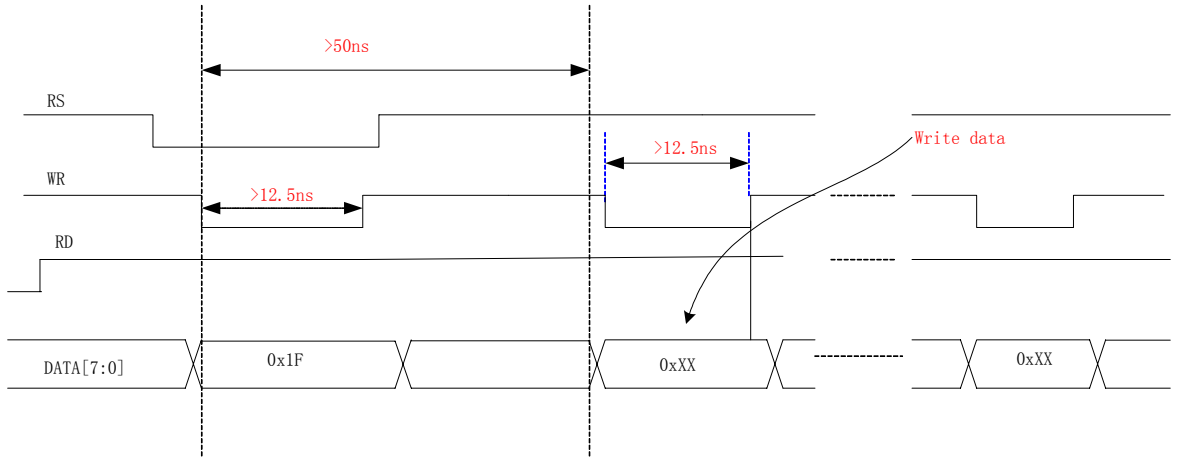
2 写工作模式寄存器 2 的时序（SetReg(0x02,0xXX)）



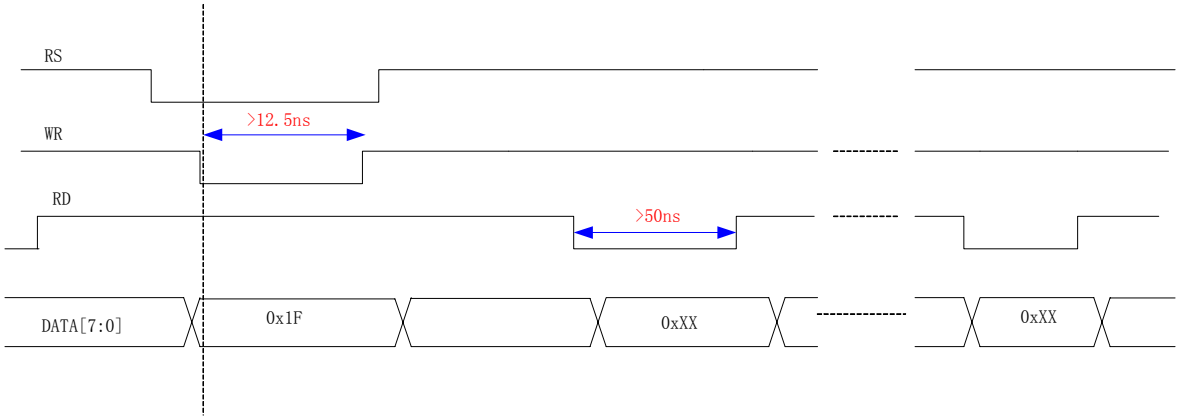
3 写行地址的时序



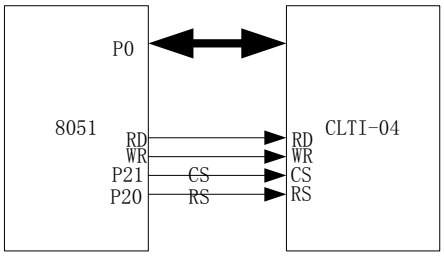
4 写显示数据进显存中的时序



6 从显存中读出显示数据的时序



四、应用框图



8051 控制方式

16×16 点阵汉字字符输入程序(51)：

汉字左上角为汉字在显示屏中坐标，

；汉字字模有两种方式，如下表，以下演示程序是以第二种字模编写的。

左半字模	右半字模
第 1 字节	第 2 字节
第 3 字节	第 4 字节
…	…
第 31 字节	第 32 字节

左半字模	右半字模
第 1 字节	第 17 字节
第 2 字节	第 18 字节
…	…
第 16 字节	第 32 字节

根据客户的取字模式来把字符中的点阵信息显示出来。显示的时候，要用前景色去修改点阵数据中为 1 的 BIT 对应的点的显示数据。用背景色去修改点阵数据中为 0 的 BIT 的数据。

假如某个字符的第 1 个字节的点阵数据如下，而我们想把这个字符显示在 33 行，100 列上（33，

100)。我们把前景色设为白色（0xffh）,背景色设为兰色（0x03）,我们用 FontData 来保存字模的点阵数据.

0	0	1	0	1	1	0	0

当用户设置好工作模式寄存器 1，前景色寄存器，被景色寄存器以及该字符显示位置的起始点后，就可以把这个点阵数据送给控制板了。

本手册上的 51 单片机的连接方式,命令地址为

```
//////////////////////////////////////////////
//把数据写进数据通道,在调用写数据的函数之前，一定要保证地址指针已经设置为
//08h，否则，会写得不对
void WriteData(unsigned char Wrdata)
{
    //WCMD=0x08; //送数据通道的地址
    WDAT=Wrdata;
}
////////////////////////////////////////////////////////////////////////////////
///////////
///下面的程序是用来测试带图形加速功能的
//////////////////////////////////////////////
void SetReg(unsigned char RegAddr,unsigned char mode)
{
    WCMD=RegAddr;
    WDAT=mode;
}

//////////////////////////////////////////////
//////////////////////////////////////////////////////
//每次画图时，都要进行坐标的初始化
void DrawInitail(unsigned int RowStarAddr,unsigned int ColStarAddr)
{
    unsigned  int xdata RowTemp,ColTemp;
  ColTemp= >>8;

  WCMD=0x03; //送行地址的前缀命令
  WDAT=RowStarAddr; //送行的地址

  WCMD=0x04; //送列的高两位地址的前缀命令  相当于 mov dptr,#CMDAddr,mov a,#00h,movx
             //@dptr,a,这三条汇编指令
  WDAT=ColTemp; //送列的高两位的地址

  WCMD=0x05; //送列的低 8 位地址的前缀命令
  WDAT= ColStarAddr; //送列的低 8 位地址
```

```
  WCMD=0x1F; //送数据通道的地址

}


//////////////////////////////////////
FontColo 保存前景色的寄存器
BackColo 保存背景色的寄存器
//////////////////////////////////////////////////////////////
/画线的函数  入口参数---线的起点和终点----
//  DrawLine(300,0,300,233,0xe0)
void DrawLine(unsigned int ysta,unsigned int xsta,unsigned int yend,
          unsigned int xend,unsigned char color)
{  unsigned int  DeltaX;
   unsigned int  DeltaY;
   unsigned int  X;  //列号，列间距
   unsigned int  Y; //两行之间的间距
   unsigned int  i;
       DeltaX=xend-xsta; //列间距
       DeltaY=yend-ysta;//行间距

   if(DeltaX==0) //这是纵向画的线,在同一列上画线
    { //纵向画线，每次都需要送一次点的坐标

       X=xsta;
        for (i=0;i<DeltaY;i++)
       {
          Y=ysta+i;//要实现画纵向的点,每次必须跨一行
          DrawDot(Y,X,color);

       }
    }
   else if(DeltaY==0) //这是须横向画的线,行间距为 0，在同一行上画线
    {
     Y=ysta;
     X=xsta;
     DrawInitail(Y,X);
     for (i=0;i<DeltaX;i++)//在列方向上跨步前进
       {
         WDAT=color;

       }

    }
```

```
  // else  //这是斜线


}
////////////////////////////////////////////////////////////////////////
//显示一个 16X16 点阵汉字的程序，有 32 个字节，列方向两个字节，行方向有 16 个字节
    j=0;
  while(j<16)
  {
     DrawInitail(y+j,x);//把字符将要显示的左上角坐标设置好
    FontDat=FontArr[CntFontData];//得到 8 个点的显示信息
            //字符被放在一个数组里面
    for(i=0;i<8;i++)  //处理字模的左半部分
      {
       c=(bit)(FontDat&0x80);
       FontDat=FontDat<<1;
       if (c==0)
        WDAT=BackColor;
       else
        WDAT=FontColor;
      }
    CntFontData=CntFontData+1;
       FontDat=FontArr[CntFontData];
        for(i=0;i<8;i++)  //处理字模的右半部分
      {
       c=(bit)(FontDat&0x80);
       FontDat=FontDat<<1;
       if (c==0)
        WDAT=BackColor;
       else
        WDAT=FontColor;
      }
       CntFontData=CntFontData+1;
       j++; //j 是行的计数器
    }
```

本控制板提供有可提供参考源程序。如果你有什么意见和建议，请email给我们，我们将尽量满足您的要求。联系：lijie6618@163.com