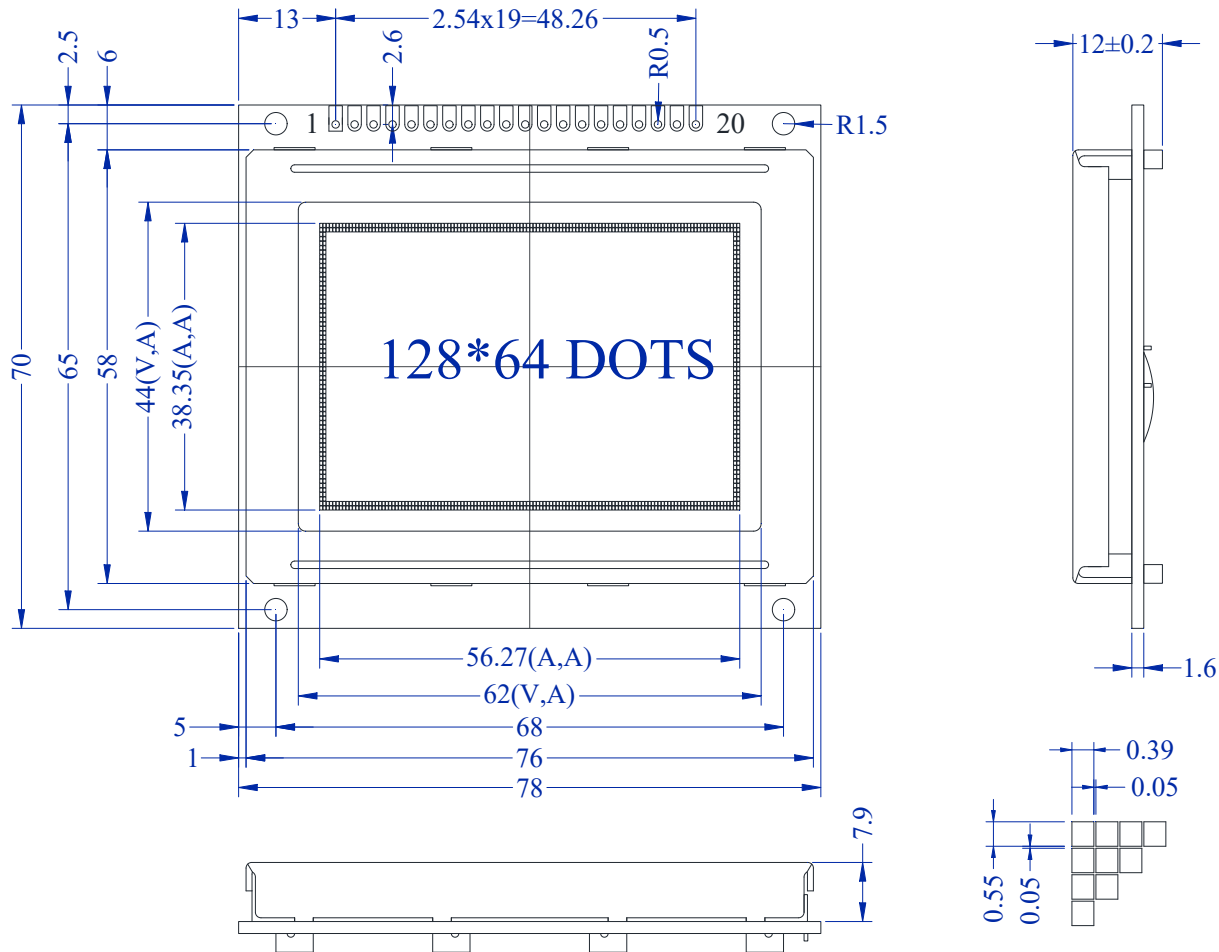


一、模块尺寸



| 项目            | 参考值            |
|---------------|----------------|
| LCM 尺寸（长×宽×厚） | 78.0×70.0×12.0 |
| 可视区域（长×宽）     | 62.0×44.0      |
| 点间距（长×宽）      | 0.44×0.60      |
| 点尺寸（长×宽）      | 0.39×0.55      |

## 二、功能介绍

- ✧ 工作电压 3.3V 或者 5.0V (默认出厂为 5.0V)
- ✧ 提供三种与 MPU 通讯方式：4 位、8 位并行，串行通讯 (默认出厂未固定，用户自己控制 PSB 来选择)
- ✧ 48x16bit 的字符显示 RAM (LCD 最多为 4 行 8 字)
- ✧ 64x128bit 的绘图 RAM (GDRAM)
- ✧ 2M bit 的中文字库 ROM (CGRAM)，总共有 8192 个中文字型
- ✧ 16K bit 的半宽字型 ROW (HCGROM)，总共有 126 个字母符号字形
- ✧ 64x16 bit 的自定义字符 RAM (CGRAM)
- ✧ 自动上电复位功能
- ✧ 外置复位端 (XRESET)
- ✧ 低功耗省电设计(除背光 45MA)
  - 正常模式 (450uA typ VDD=5V)
  - 待机模式 (30uA max VDD=5V)
- ✧ 显示驱动电压 VLCD (V0~VSS)：最大 7V
- ✧ 绘图以及文字画面混合显示功能
- ✧ 多功能指令：
  - 画面清除 (display clear)      显示移位 (display shift)
  - 光标归零 (display clear)      垂直画面旋转 (vertical line scroll)
  - 显示开/关 (display on/off)      反白显示 (by\_line reverse display)
  - 光标显示/隐藏 (cursor on/off)      待机模式 (standby mode)
  - 显示字闪烁 (display character blink)
  - 光标移位 (cursor shift)
- ✧ 占空比 1/32      偏压比 1/6
- ✧ 最佳可视角为 6 点钟，可以更改
- ✧ 显示颜色效果选黄绿，蓝白，灰白，也可以定制
- ✧ 工作温度，宽温：-20 度到+70 度

### 三. 接口定义

| 引脚   | 名称        | 方向  | 说明                                                                                                                                                                                                              |
|------|-----------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | VSS       | --  | 电源负端 (0V)                                                                                                                                                                                                       |
| 2    | VDD       | --  | 电源正端 (+3.3V 或+5.0V, 默认出厂时设定+5.0V)                                                                                                                                                                               |
| 3    | V0        | --  | LCD 驱动电压(可调)                                                                                                                                                                                                    |
| 4    | RS (CS)   | I   | <p>并口方式:</p> <ul style="list-style-type: none"> <li>● RS=0:<br/>当 MPU 进行读模块操作, 指向地址计数器。<br/>当 MPU 进行写模块操作, 指向指令寄存器。</li> <li>● RS=1:<br/>无论 MPU 读/写操作, 均指向数据寄存器。</li> </ul> <p>串口方式:<br/>CS: 串行片选信号, 高电平有效。</p> |
| 5    | R/W (SID) | I   | <p>并口方式:</p> <ul style="list-style-type: none"> <li>● R/W=0 写操作。</li> <li>● R/W=1 读操作。</li> </ul> <p>串口方式:<br/>SID: 串行数据输入端</p>                                                                                 |
| 6    | E (SCLK)  | I   | <p>并口方式: 使能信号, 高电平有效。</p> <p>串口方式: SCLK 串行时钟信号。</p>                                                                                                                                                             |
| 7-14 | DB0 ~ DB7 | I/O | MPU 与模块之间并口的数据传送通道,<br>4 位总线模式下 D0 ~ D3 脚断开                                                                                                                                                                     |
| 15   | PSB       | I   | 串/并口控制选择端: (内部 J2 也可选择)                                                                                                                                                                                         |
| 16   | NC        | I   | 空脚                                                                                                                                                                                                              |
| 17   | RST       | I   | 复位脚 (低电平有效)                                                                                                                                                                                                     |
| 18   | VOUT      | --  | 倍压输出脚。(VDD=+3.3V 时有效)                                                                                                                                                                                           |
| 19   | LEDA      | --  | 背光电源正端 (+3.3V 或+5.0V, 出厂时设定+5.0V)                                                                                                                                                                               |
| 20   | LEDK      | --  | 背光电源负端 (0V)                                                                                                                                                                                                     |

四. 电性参数(直流)

| 名称        | 符号   | 测试条件   | 参数范围    |         |         | 单位 |
|-----------|------|--------|---------|---------|---------|----|
|           |      |        | 最小      | 标准      | 最大      |    |
| 模块工作电压    | VDD  | —      | 4.5/3.1 | 5.0/3.3 | 5.5/3.5 | V  |
| 玻璃电压      | V0   | V0-VDD | 4.5     | 5.0     | 7.0     | V  |
| 背光工作电压    | VLED | —      | 4.5/3.1 | 5.0/3.3 | 5.5/3.5 | V  |
| IO 输入高电平  | VIH  | —      | 0.7VDD  | —       | VDD     | V  |
| IO 输入低电平  | VIL  | —      | —       | —       | 0.6     | V  |
| LCM 输出高电平 | VOH  | —      | 0.8VDD  | —       | —       | V  |
| LCM 输出低电平 | VOL  | —      | —       | —       | 0.4     | V  |
| 模块工作电流    | IDD  | =VDD   | —       | —       | 0.5     | MA |
| 模块待机电流    | ID0  | =VDD   | —       | —       | 10      | uA |
| 背光工作电流    | ILED | =VLED  | 30      | 45      | 60      | MA |

五. 显示原理

DDRAM 地址与 128\*64 点阵显示屏的关系（可以显示 8\*4=32 个汉字）一行最多 16 字

表 1

| Y<br>X |      | 128 点 |   |    |   |    |   |    |   |    |   |    |   |    |   |    |   |
|--------|------|-------|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|
|        |      | H     | L | H  | L | H  | L | H  | L | H  | L | H  | L | H  | L | H  | L |
| 上半屏    | 32 点 | 80    |   | 81 |   | 82 |   | 83 |   | 84 |   | 85 |   | 86 |   | 87 |   |
|        |      | 90    |   | 91 |   | 92 |   | 93 |   | 94 |   | 95 |   | 96 |   | 97 |   |
| 下半屏    | 32 点 | 88    |   | 89 |   | 8A |   | 8B |   | 8C |   | 8D |   | 8E |   | 8F |   |
|        |      | 98    |   | 99 |   | 9A |   | 9B |   | 9C |   | 9D |   | 9E |   | 9F |   |

80 表示 DDRAM 地址，80 代表一个显示汉字的地址，占 16\*16 点阵，只要在 80 地址内写入汉字内码就能显示你的汉字或者字符， 汉字内码有两个字节，高位（H）写在前，低位（L）写在后，注意地址会自动加 1。

现在的单片机编译软件都能引用汉字编译出汉字内码，12864 有一套标准的 GB2312 简体字库，收到汉字内码直接调取内部对应点阵显示在屏幕上。

12864 屏实际上是将一个 25632 点阵屏中间切断，左边的一半 12832 放在上半屏，右边的一半 12832 放在下半屏，组成的 12864 点阵屏，用户在编程的时候要特别注意。

GDRAM 与 12864 点阵的关系（128\*64 点阵的绘图像素）

表 2

| Y 点<br>X 点 |    | 128 点 |   |       |   |        |   |        |   |        |   |        |   |        |   |        |   |
|------------|----|-------|---|-------|---|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|
|            |    | H     | L | H     | L | H      | L | H      | L | H      | L | H      | L | H      | L | H      | L |
|            |    | 0     |   | 1     |   | 2      |   | 3      |   | 4      |   | 5      |   | 6      |   | 7      |   |
| 上半屏        | 0  | Y0X0  |   | Y0X1  |   | Y0X2   |   | Y0X3   |   | Y0X4   |   | Y0X5   |   | Y0X6   |   | Y0X7   |   |
|            | 1  | Y1X0  |   | Y1X1  |   | Y1X2   |   | Y1X3   |   | Y1X4   |   | Y1X5   |   | Y1X6   |   | Y1X7   |   |
|            | 2  | Y2X0  |   | Y2X1  |   | Y2X2   |   | Y2X3   |   | Y2X4   |   | Y2X5   |   | Y2X6   |   | Y2X7   |   |
|            | .. | ...   |   | ...   |   | ...    |   | ...    |   | ...    |   | ...    |   | ...    |   | ...    |   |
|            | 31 | Y31X0 |   | Y31X1 |   | Y31X2  |   | Y31X3  |   | Y31X4  |   | Y31X5  |   | Y31X6  |   | Y31X7  |   |
| 下半屏        | 0  | Y0X8  |   | Y0X9  |   | Y0X10  |   | Y0X11  |   | Y0X12  |   | Y0X13  |   | Y0X14  |   | Y0X15  |   |
|            | 1  | Y1X8  |   | Y1X9  |   | Y0110  |   | Y1X11  |   | Y1X12  |   | Y1X13  |   | Y1X14  |   | Y1X15  |   |
|            | 2  | Y1X8  |   | Y1X9  |   | Y1X10  |   | Y1X11  |   | Y1X12  |   | Y1X13  |   | Y1X14  |   | Y1X15  |   |
|            | .. | ...   |   | ...   |   | ...    |   | ...    |   | ...    |   | ...    |   | ...    |   | ...    |   |
|            | 31 | Y31X8 |   | Y31X9 |   | Y31X10 |   | Y31X11 |   | Y31X12 |   | Y31X13 |   | Y31X14 |   | Y31X15 |   |

Y 为列的位置，X 是水平上的地址（水平一个地址有 16 个点）

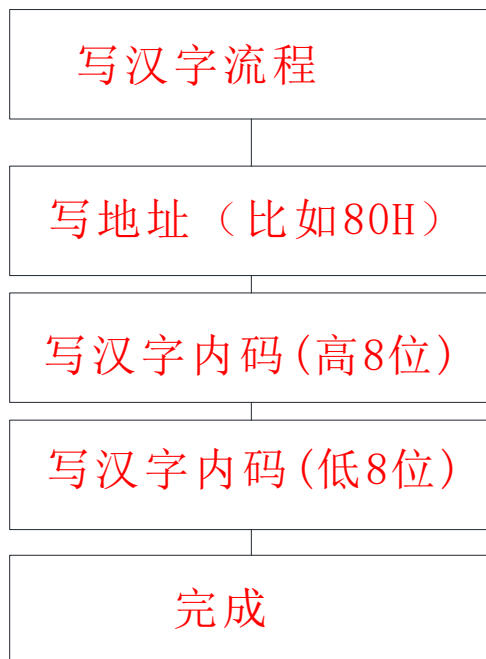
|     |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|-----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|----|
| 位=点 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 字节  | H   |     |     |     |     |     |    |    | L  |    |    |    |    |    |    |    |
| 地址  | 80h |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |

GDRAM 与 12864 点阵的分布图，程序写入过程为，先写入垂直地址，例如（0-31），再写水平 X 地址，例如；（0-7/8-15），一个水平地址有 1\*16 位点阵（分两字节，先写高字节，再写低字节，高位在左边），水平地址可以自动加 1，

这里 0 地址看作 0X80，在写地址的时候都要加上 0X80；举例，如果是在第 2 行的第 33 列，垂直地址为 0x80+1，水平地址为 0x80+2，

要调一幅图片时，使用横向取模取出点阵数据， 你也可以自编图形， 在最后的程序例程中，有绘制边框供你参考

## 写显示数据流程



### a) CGRAM

#### 自编字符 CGRAM

字符预留了几个自编字符空间，编写一些特殊的字符或者符号，使用原理，进入 CGRAM （ 40H）；选择内码地址 02（00 到 08）；然后写入 8\*8 点阵数据（横向取模），要显示自编字符时，写显示 DDRAM 地址，然后写入内码地址 02 就能显示自编符号啦

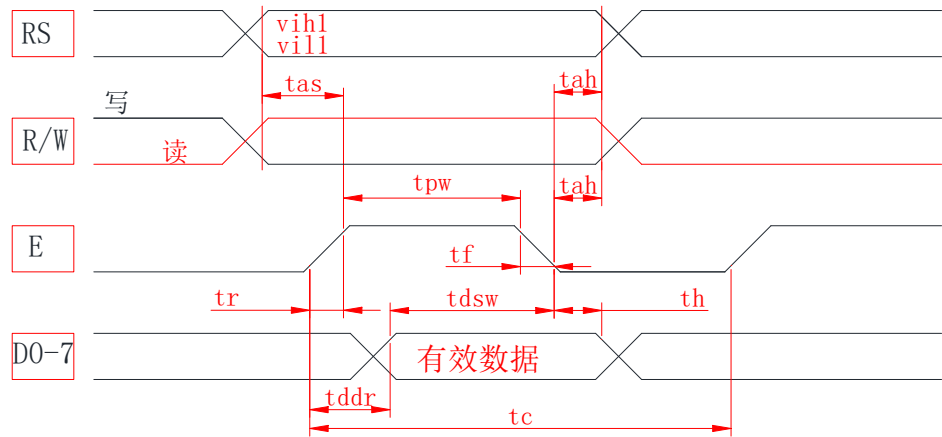
b) HCGROM

内部的字符

|    | 0 | 1 | 2 | 3  | 4  | 5 | 6 | 7 | 8 | 9 | A  | B | C | D | E | F |
|----|---|---|---|----|----|---|---|---|---|---|----|---|---|---|---|---|
| 00 |   | ☺ | ☹ | ♥  | ♦  | ♣ | ♠ | • | ◐ | ◑ | ♂  | ♀ | ♂ | ♀ | ♂ | ♀ |
| 10 | ▶ | ◀ | ↑ | !! | ¶  | § | — | ‡ | ↑ | ↓ | →  | ← | └ | ↕ | ▲ | ▼ |
| 20 |   | ! | " | #  | \$ | % | & | ' | ( | ) | *+ | , | - | . | / |   |
| 30 | 0 | 1 | 2 | 3  | 4  | 5 | 6 | 7 | 8 | 9 | :  | ; | < | = | > | ? |
| 40 | @ | A | B | C  | D  | E | F | G | H | I | J  | K | L | M | N | O |
| 50 | P | Q | R | S  | T  | U | V | W | X | Y | Z  | [ | \ | ] | ^ | _ |
| 60 | ` | a | b | c  | d  | e | f | g | h | i | j  | k | l | m | n | o |
| 70 | p | q | r | s  | t  | u | v | w | x | y | z  | { | ! | } | ~ | △ |

六. 时序图

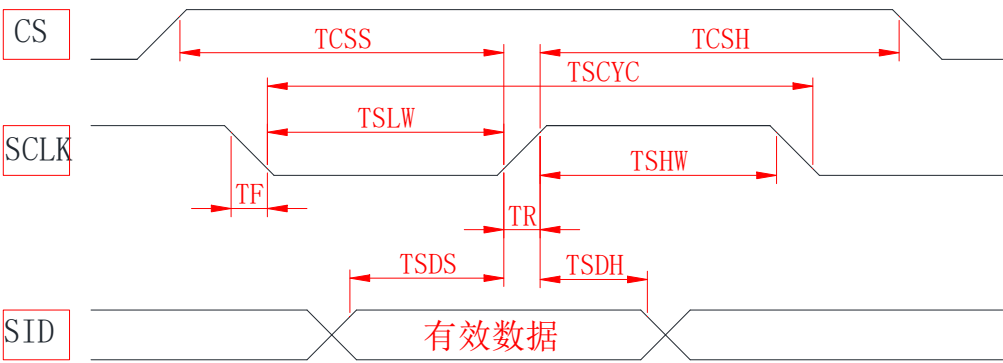
并口时序



并口模式常温（环境25度，VDD=4.5V 或者 VDD=2.7V）

| 参数              | 符号    | 测试条件          | 最小        | 典型      | 最大      | 单位  |
|-----------------|-------|---------------|-----------|---------|---------|-----|
| 内部时钟            |       |               |           |         |         |     |
| 振荡频率            | Fosc  | R=33kr/R=18kr | 480/470   | 540/530 | 600/590 | Khz |
| 外部时钟            |       |               |           |         |         |     |
| 外部频率            | fex   | -             | 480/470   | 540/530 | 600/590 | Khz |
| 占空比             |       |               | 45        | 50      | 55      | %   |
| 上升/下降时间         | Tr,tf | -             | -         | -       | 0.2     | us  |
| 写模式（从单片机写数据到模块） |       |               |           |         |         |     |
| E周期             | Tc    | E             | 1200/1800 | -       | -       | Ns  |
| E脉冲宽度           | Tpw   | E             | 140/160   | -       | -       | Ns  |
| E上升/下降时间        | Tr,tf | E             | -         | -       | 25      | Ns  |
| 地址建立时间          | Tas   | Rs,r/w,e      | 10        | -       | -       | Ns  |
| 地址保持时间          | Tah   | Rs,r/w,e      | 20        | -       | -       | Ns  |
| 数据建立时间          | Tdsw  | D0-d7         | 40        | -       | -       | Ns  |
| 数据保持时间          | Th    | D0-d7         | 20        | -       | -       | Ns  |
| 读模式（从模块读数据到单片机） |       |               |           |         |         |     |
| E周期             | Tc    | E             | 1200/1800 | -       | -       | Ns  |
| E脉冲宽度           | Tpw   | E             | 140/320   | -       | -       | Ns  |
| E上升/下降时间        | Tr,tf | E             | -         | -       | 25      | Ns  |
| 地址建立时间          | Tas   | Rs,r/w,e      | 10        | -       | -       | Ns  |
| 地址保持时间          | Tah   | Rs,r/w,e      | 20        | -       | -       | Ns  |
| 数据建立时间          | Tddr  | D0-d7         | -         | -       | 100/260 | Ns  |
| 数据保持时间          | Th    | D0-d7         | 20        | -       | -       | Ns  |

串口时序



串口模式常温（环境25度，VDD=4.5V 或者 VDD=2.7V）

| 参数              | 符号    | 测试条件          | 最小      | 典型  | 最大  | 单位  |
|-----------------|-------|---------------|---------|-----|-----|-----|
| 内部时钟            |       |               |         |     |     |     |
| 振荡频率            | Fosc  | R=33kr/R=18kr | 470     | 530 | 590 | Khz |
| 外部时钟            |       |               |         |     |     |     |
| 外部频率            | fex   | -             | 470     | 530 | 590 | Khz |
| 占空比             |       |               | 45      | 50  | 55  | %   |
| 上升/下降时间         | Tr,tf | -             | -       | -   | 0.2 | us  |
| 写模式（从单片机写数据到模块） |       |               |         |     |     |     |
| 串行时钟周期          | TSCYC | E/SCLK        | 400/600 | -   | -   | Ns  |
| SCLK高脉宽         | TSHW  | E/SCLK        | 200/300 | -   | -   | Ns  |
| SCLK脉宽          | TSLW  | E/SCLK        | 200/300 | -   | -   | Ns  |
| SID数据建立时间       | TSDS  | RW/SID        | 40      | -   | -   | Ns  |
| SID数据保持时间       | TSDH  | RW/SID        | 40      | -   | -   | Ns  |
| CS建立时间          | TCSS  | RS/CS         | 60      | -   | -   | Ns  |
| CS保持时间          | TCSH  | RS/CS         | 60      | -   | -   | Ns  |

## 七. 指令说明

### 1、指令表：（RE=0：基本指令集）

| 指令            | 指令码 |    |    |      |     |     |     |     |     |     | HEX  | 说明                                                         | 执行时间<br>(540KHZ) |
|---------------|-----|----|----|------|-----|-----|-----|-----|-----|-----|------|------------------------------------------------------------|------------------|
|               | RS  | RW | D7 | D6   | D5  | D4  | D3  | D2  | D1  | D0  |      |                                                            |                  |
| 清除显示          | 0   | 0  | 0  | 0    | 0   | 0   | 0   | 0   | 0   | 1   | 0X01 | 将DDRAM填满”20H”，并且设定DDRAM的地址计数器（AC）到“00H”                    | 1.6ms            |
| 地址归零          | 0   | 0  | 0  | 0    | 0   | 0   | 0   | 0   | 1   | X   | 0X02 | 设定DDRAM的地址计数器（AC）到”00H”，并且将光标移到开头原点位置，这个指令并不改变DDRAM的内容     | 72us             |
| 输入点设定         | 0   | 0  | 0  | 0    | 0   | 0   | 0   | 1   | I/D | S   | 0X4X | 指定在数据的读取与写入时，设定光标的移动方向及指定显示的移位                             | 72us             |
| 显示状态开/关       | 0   | 0  | 0  | 0    | 0   | 0   | 1   | D   | C   | B   | 0x8x | D=1；整体显示ON<br>C=1；光标ON<br>B=1；光标位置反白ON                     | 72us             |
| 光标或显示移位控制     | 0   | 0  | 0  | 0    | 0   | 1   | S/C | R/L | X   | X   | 0x1x | 设定光标的移动与显示的移位控制位；这个指令并不改变DDRAM的内容                          | 72us             |
| 功能设定          | 0   | 0  | 0  | 0    | 1   | DL  | X   | ORE | X   | X   | 0X2X | DL=1；8位控制模式<br>DL=0；4位控制模式<br>RE=1；选择扩展指令集<br>RE=0；选择基本指令集 | 72us             |
| 设定CGRAM地址     | 0   | 0  | 0  | 1    | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0X4X | 设定CGRAM地址到地址计数器（AC）<br>需确认扩展指令中SR=0（卷动地址或RAM地址选择）          | 72us             |
| 设定DDRAM地址     | 0   | 0  | 1  | 0AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0X8X | 设定CGRAM地址到地址计数器（AC）<br>AC6固定为0                             | 72us             |
| 读取忙碌标志（BF）和地址 | 0   | 1  | BF | AC6  | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |      | 读取忙碌标志（BF）可以确认内部动作是否完成，同时可以读出地址计数器（AC）的值                   | 0us              |
| 写数据到RAM       | 1   | 0  | D7 | D6   | D5  | D4  | D3  | D2  | D1  | D0  |      | 写入数据到内部RAM（DDRAM/CGRAM/GDRAM）                              | 72us             |

|          |   |   |    |    |    |    |    |    |    |    |  |                                   |      |
|----------|---|---|----|----|----|----|----|----|----|----|--|-----------------------------------|------|
| 读出RAM的数据 | 1 | 0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |  | 从内部RAM读取数据<br>(DDRAM/CGRAM/GDRAM) | 72us |
|----------|---|---|----|----|----|----|----|----|----|----|--|-----------------------------------|------|

指令表二：（RE=1：扩充指令集）

| 指令                    | 指令码 |    |    |    |      |      |            |            |                 |            | HEX  | 说明                                                                                      | 执行时间<br>(540KHZ) |
|-----------------------|-----|----|----|----|------|------|------------|------------|-----------------|------------|------|-----------------------------------------------------------------------------------------|------------------|
|                       | RS  | RW | D7 | D6 | D5   | D4   | D3         | D2         | D1              | D0         |      |                                                                                         |                  |
| 待机模式                  | 0   | 0  | 0  | 0  | 0    | 0    | 0          | 0          | 0               | 1          | 0X01 | 进入待机模式，执行任何其它指令都可终止待机模式（COM1~32停止动作）                                                    | 72us             |
| 卷动地址<br>或RAM地址选择      | 0   | 0  | 0  | 0  | 0    | 0    | 0          | 0          | 1               | SR         | 0X02 | SR=1；允许输入垂直卷动地址SR=0；允许设定CGRAM地址（基本指令）                                                   | 72us             |
| 反白选择                  | 0   | 0  | 0  | 0  | 0    | 0    | 0          | 1          | R1              | R0         | 0X4X | 选择4行中的任一行反白显示，并可决定反白与否R1，R0初值为‘00，当第一次设定时为反白显示，再一次设定时为正常显示                              | 72us             |
| 扩充功能<br>设定            | 0   | 0  | 0  | 0  | 1    | DL   | X          | 1RE        | G               | 0          | 0X3X | DL=1；8位控制模式<br>DL=0；4位控制模式<br>RE=1；选择扩展指令集<br>RE=0；选择基本指令集<br>G=1；绘图显示ON<br>G=0；绘图显示OFF | 72us             |
| 设定IRAM<br>地址或卷<br>动地址 | 0   | 0  | 0  | 1  | AC5  | AC4  | AC3        | AC2        | AC1             | AC0        | 0X4X | SR=1：AC5~AC0为垂直卷动地址                                                                     | 72us             |
| 设定绘图<br>RAM地址         | 0   | 0  | 1  | 00 | 0AC5 | 0AC4 | AC3<br>AC3 | AC2<br>AC2 | AC1<br>1A<br>C1 | AC0<br>AC0 | 0X8X | 设定（GDRAM地址到地址计数器（AC）<br>先设垂直地址再设水平地址（连续写入两个字节的坐标地址）<br>垂直地址范围AC5~AC0<br>水平地址范围AC3~AC0   | 2us              |

## 2、具体指令介绍：

基本指令（RE=0）

### 1) 清除显示：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0x01 |

功能：将DDRAM 填满” 20H”（空格），把DDRAM 地址计数器调整“00H”，重新进入点设定将I/D设为” 1”，光标右移AC 加1。

### 2) 地址归位：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | x  | 0x02 |

功能：把DDRAM 地址计数器调整为“00H”，光标回原点，该功能不影响显示DDRAM

### 3) 输入点设置：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1  | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|-----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | I/D | S  | 0x04 |

功能：设定光标移动方向并指定整体显示是否移动。

I/D=1 光标右移，AC自动加1； I/D=0 光标左移，AC自动减1。

S=1 且DDRAM为写状态：整体显示移动，方向由I/D决定（I/D=1左移，I/D=0右移）

S=0 或DDRAM 为读状态：整体显示不移动。

### 4) 显示状态开/关：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 1  | D  | C  | B  | 0x08 |

功能：D=1：整体显示ON ； D=0：整体显示OFF。

C=1：光标显示ON ； C=0：光标显示OFF。

B=1：光标位置反白且闪烁 ； B=0：光标位置不反白闪烁。

### 5) 光标或显示移位控制：

|    | RS | RW | D7 | D6 | D5 | D4 | D3  | D2  | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|-----|-----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 1  | S/C | R/L | X  | x  | 0x1x |

功能：S/C：光标左/右移动，AC减/加1。

R/L：整体显示左/右移动，光标跟随移动，AC值不变。

| S/C | R/L | 说明             | AC值     |
|-----|-----|----------------|---------|
| L   | L   | 光标向左移动         | AC=AC-1 |
| L   | H   | 光标向右移动         | AC=AC+1 |
| H   | L   | 显示向左移动，且光标跟着移动 | AC=AC   |
| H   | H   | 显示向右移动，且光标跟着移动 | AC=AC   |

## 6) 功能设定：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 1  | DL | X  | RE | X  | x  | 0x2x |

功能：DL=1： 8-BIT 控制接口； DL=0： 4-BIT 控制接口。

RE=1： 扩充指令集动作； RE=0： 基本指令集动作。

## 7) 设定CGRAM地址：

|    | RS | RW | D7 | D6 | D5  | D4  | D3  | D2  | D1  | D0  | HEX  |
|----|----|----|----|----|-----|-----|-----|-----|-----|-----|------|
| 代码 | 0  | 0  | 0  | 1  | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0x4x |

功能：设定CGRAM地址到地址计数器（AC），需确定扩充指令中SR=0(卷动地址或RAM地址选择)

## 8) 设定DDRAM地址：

|    | RS | RW | D7 | D6  | D5  | D4  | D3  | D2  | D1  | D0  | HEX  |
|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|------|
| 代码 | 0  | 0  | 1  | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0x8x |

功能：设定DDRAM地址到地址计数器（AC）

## 9) 读取忙碌状态（BF）和地址：

|    | RS | RW | D7 | D6  | D5  | D4  | D3  | D2  | D1  | D0  | HEX  |
|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|------|
| 代码 | 0  | 1  | BF | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0xXx |

功能：读取忙碌状态（BF）可以确认内部动作是否完成，同时可以读出地址计数器（AC）的值，当BF=1，表示内部忙碌中此时不可下指令需等BF=0才可下新指令

#### 10) 写资料到RAM:

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 1  | 0  | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0xXx |

功能：写入资料到内部的RAM（DDRAM/CGRAM/GDRAM），每个RAM地址都要连续写入两个字节的资料。

#### 11) 读RAM的值:

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 1  | 1  | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0xXx |

功能：从内部RAM 读取数据（DDRAM/CGRAM/GDRAM），当设定地址指令后，若需读取数据时需先执行一次空的读数据，才会读取到正确数据，第二次读取时则不需要，除非又下设定地址指令。

### 扩充指令（RE=1）

#### 1) 待命模式:

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0x1x |

功能：进入待命模式，执行其它命令都可终止待命模式

#### 2) 卷动地址或RAM地址选:

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | SR | 0x2x |

功能：SR=1：允许输入卷动地址；  
SR=0：允许设定CGRAM地址（基本指令）

#### 3) 反白选择:

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | R1 | R0 | 0x4x |

功能：选择2 行中的任一行作反白显示，并可决定反白与否。第一次设定为反白显示，再次设定时为正常显示

|       |    |    |               |
|-------|----|----|---------------|
| 12864 | R1 | R0 | 行地址参数         |
| 屏     | L  | L  | 第一、三、行反白或正常显示 |
|       | L  | H  | 第二、四、行反白或正常显示 |

#### 4) 睡眠模式：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 0  | 0  | 1  | SL | X  | X  | 0x08 |

功能：SL=1：脱离睡眠模式； SL=0：进入睡眠模式。

#### 5) 扩充功能设定：

|    | RS | RW | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | HEX  |
|----|----|----|----|----|----|----|----|----|----|----|------|
| 代码 | 0  | 0  | 0  | 0  | 1  | DL | X  | RE | G  | X  | 0x3x |

功能：DL=1：8-BIT 控制接口； DL=0：4-BIT 控制接口  
 RE=1：扩充指令集动作； RE=0：基本指令集动作  
 G=1：绘图显示ON； G=0：绘图显示OFF

#### 6) 卷动地址设定：

|    | RS | RW | D7 | D6 | D5  | D4  | D3  | D2  | D1  | D0  | HEX  |
|----|----|----|----|----|-----|-----|-----|-----|-----|-----|------|
| 代码 | 0  | 0  | 0  | 1  | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0x4x |

功能：SR=1，AC5~AC0 为垂直卷动地址

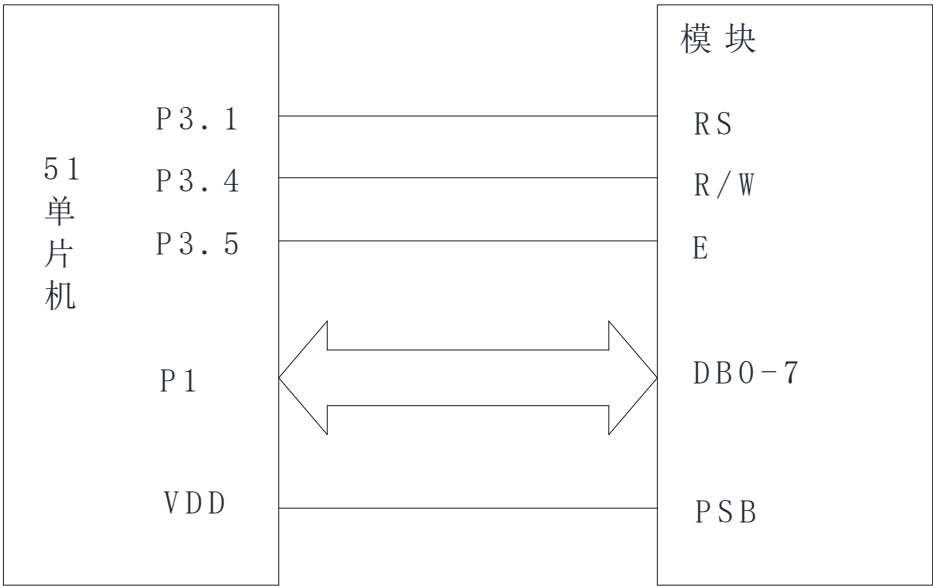
7) 绘图RAM地址设定:

|    | RS | RW | D7 | D6  | D5  | D4  | D3  | D2  | D1  | D0  | HEX  |
|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|------|
| 代码 | 0  | 0  | 1  | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 0x8x |

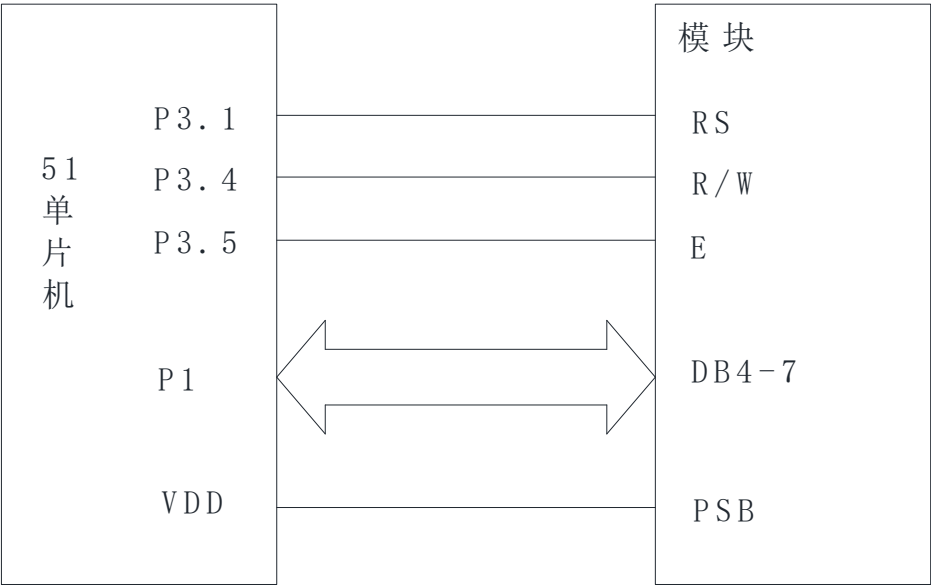
功能：设定GDRAM地址到地址计数器（AC）

八、单片机接线图

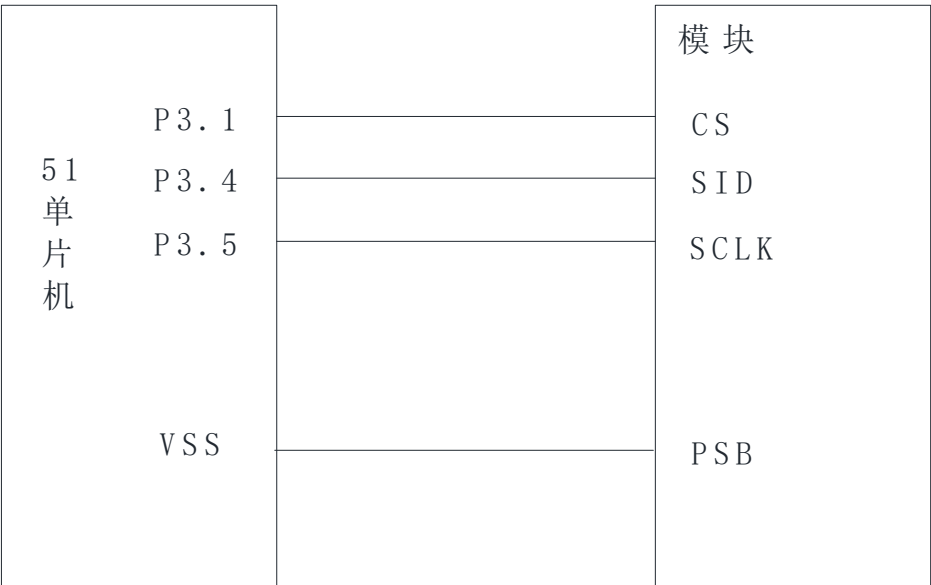
1)、8位并行接口:

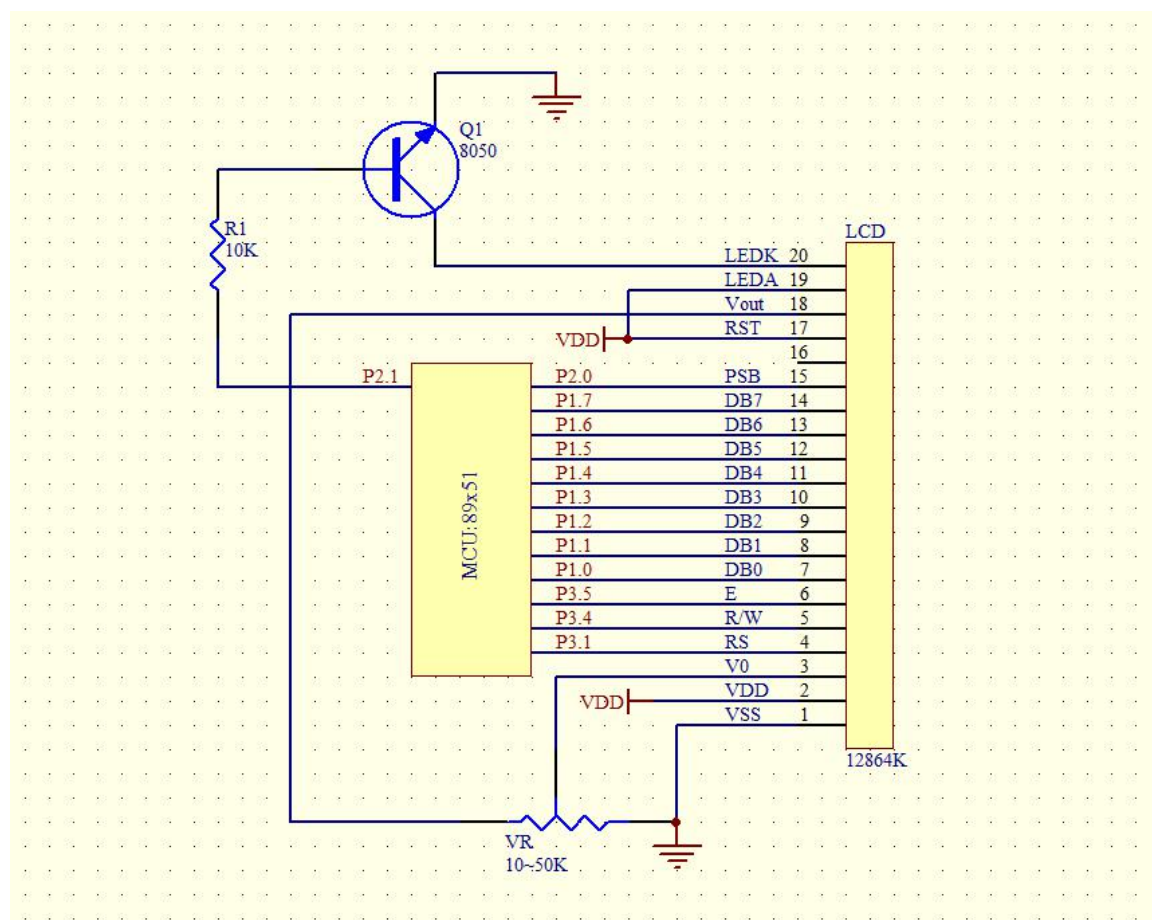


2)、4位并行接口:

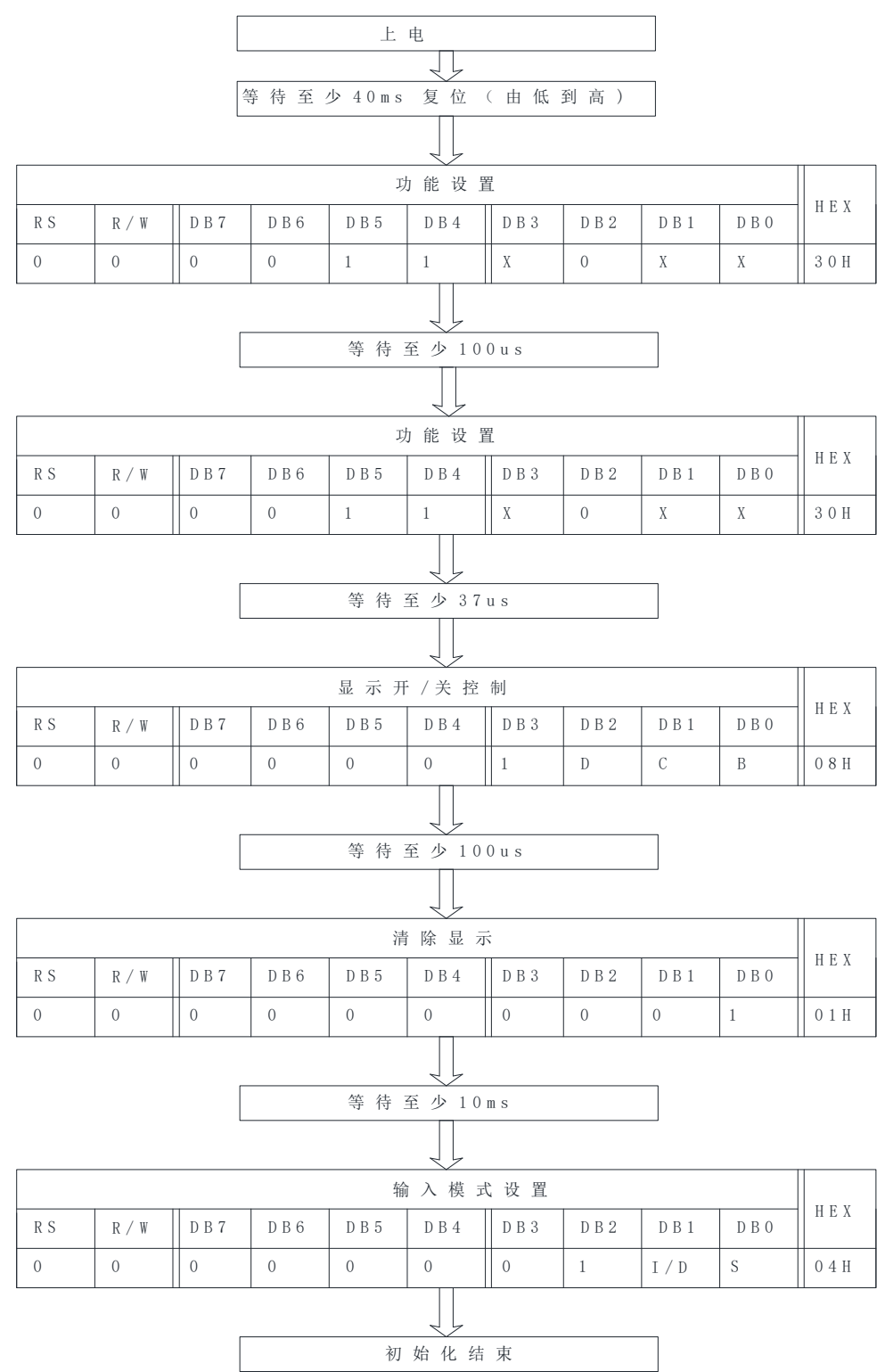


3)、串行接口：

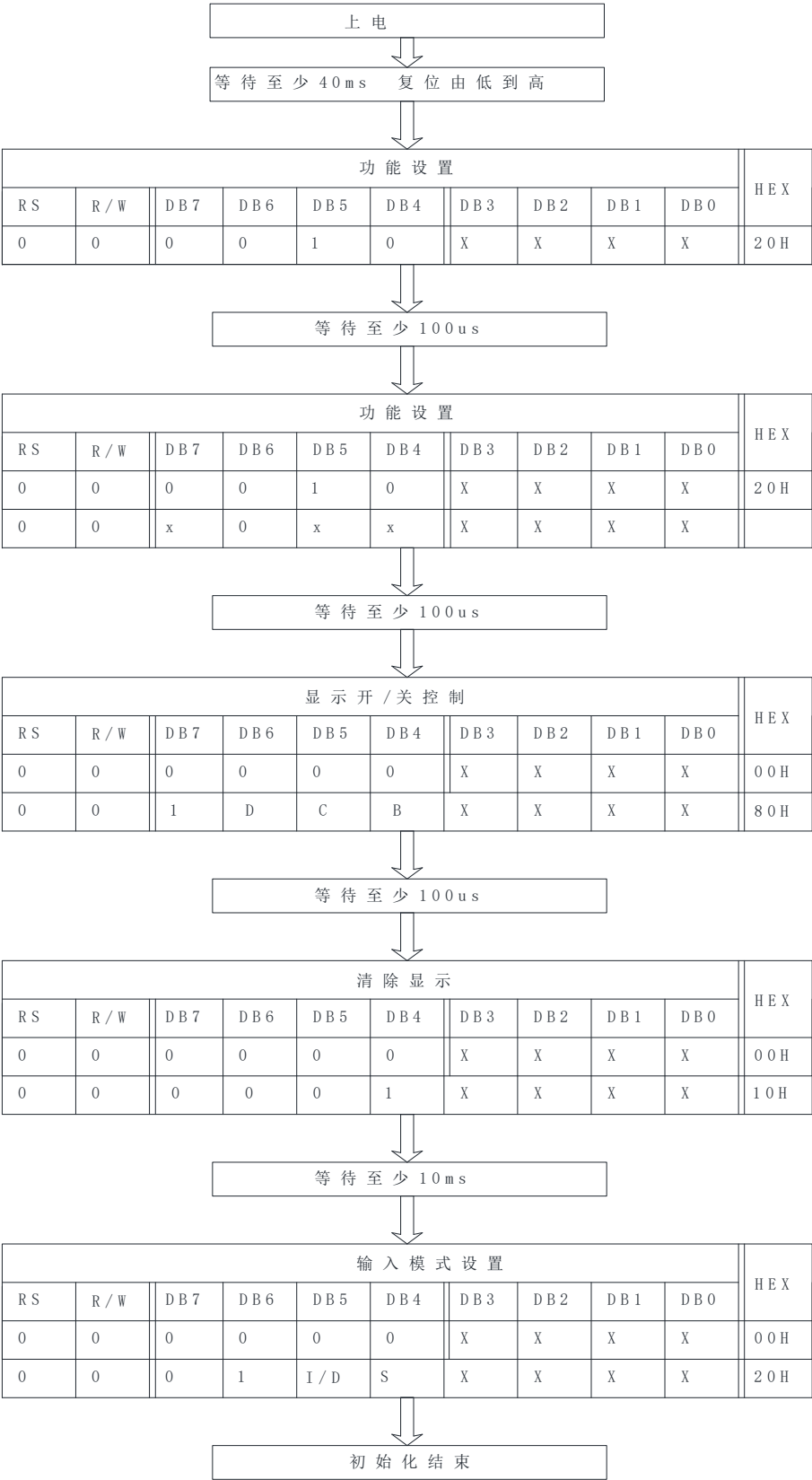




## 8位接口模式



4位接口模式



十、程序例程

```
//深圳绘晶科技有限公司、
//12864点阵中文字库,单片机:89S52, 晶振:12M,
//STC单片机完全兼容
//串并行共用程序

#include<reg52.h>
#include <intrins.h>
//sbit REST=P2^1;
sbit RS=P3^1;//串口时为CS
sbit RW=P3^4; //串口为SID
sbit E1=P3^5; //串口为时钟SCLK
sbit PSB=P2^0;//并口时, PSB=1;串口时, PSB=0
sbit stop=P3^2;//低电平触发

typedef unsigned int Uint;
typedef unsigned char Uchar;
Uchar z,z1,d,d1,s,s1,s10,s100,m1;

//汉字,直接可以写入字形,写入标点符号后要加空格键
unsigned char code uctech[] = {"绘晶科技公司地址区石岩镇麻布第二"};
//显示在第1, 3行
unsigned char code uctech3[] = {"广东省深圳市宝安工业区第4 栋2 楼"};
//显示在第2, 4行
unsigned char code uctech6[] = {"汇          精焦    128*64    至"};
//显示在第1, 3行
unsigned char code uctech7[] = {"聚  绘晶科技  品点  23146001  上"};
//显示在第2, 4行
unsigned char code uctech1[] = {"绘晶科技"};
//显示在第2行
unsigned char code uctech2[] = {" 128*64  "};
//显示在第3行
unsigned char code uctech4[] = {"TIME"};
unsigned char code uctech5[] = {"绘晶科技23146001"};
unsigned char code uctech8[] = {"有8192个中文字型"};
unsigned char code uctech9[] = {"有126 个字母符号"};

//这个是在串口时指令和数据之间的延时
/*
void delay10US(Uchar x)
{
    Uchar k;
    for(k=0;k<x;k++);
}
```

```
}
*/
const Uchar delay=250; //延时时间常数
static void Wait1ms(void)//延迟1 ms
{
    Uchar cnt=0;
    while (cnt<delay) cnt++;
}
//延迟n ms
void WaitNms(int n)
{
    Uchar i;
    for(i=1;i<=n;i++)
        Wait1ms();
}

//*****//
//以下是并口时才开的
//读忙标志,

void RDBF(void)
{
    Uchar temp;
    RS=0;    // RS=0
    RW=1;    // RW=1
    while(1)
    {
        P1=0xFF;    //数据线为输入
        E1=1;
        temp=P1;
        E1=0;    // E=0
        if ((temp&0x80)==0) break;
    }
}
//写数据到指令寄存器

void WRCommandH(Uchar comm)
{
    RDBF();
    RS=0;
    RW=0;
    P1=comm;
    E1=1;
    E1=0;
```

```
}
```

```
//写数据到数据寄存器
```

```
void WRDataH(Uchar TEMP)
```

```
{
    RDBF();
    RS=1;
    RW=0;
    P1=TEMP;
    E1=1;
    E1=0;
}
```

```
////////////////////////////////////
```

```
//以下是串口时开的读写时序
```

```
/*
```

```
void SendByteLCDH(Uchar WLCDData)
```

```
{
    Uchar i;
    for(i=0;i<8;i++)
    {
        if((WLCDData<<i)&0x80)RW=1;
        else RW=0;
        E1=0;
        E1=1 ;
    }
}
```

```
SPIWH(Uchar Wdata,Uchar WRS)
```

```
{
    SendByteLCDH(0xf8+(WRS<<1)); //寄存器选择WRS
    SendByteLCDH(Wdata&0xf0);
    SendByteLCDH((Wdata<<4)&0xf0);
}
```

```
void WRCommandH(Uchar CMD)
```

```
{
    RS=0;
    RS=1;
    SPIWH(CMD,0);
    delay10US(90); //89S52来模拟串行通信,所以,加上89S52的延时,
```

```
}
```

```
void WRDataH(Uchar Data)
```

```
{
```

```
    RS=0;
```

```
    RS=1;
```

```
    SPIWH(Data,1);
```

```
}
```

```
*/
```

```
/******
```

```
//初始化LCD-8位接口
```

```
void LCDInit(void)
```

```
{    //PSB=0; //串口
```

```
    PSB=1;//并口时选这个,上一行取消
```

```
    //  REST=1;
```

```
    //  REST=0;
```

```
    //  REST=1;
```

```
    WRCommandH(0x30);    //基本指令集,8位并行
```

```
    WRCommandH(0x06);    //起始点设定: 光标右移
```

```
    WRCommandH(0x01);    //清除显示DDRAM
```

```
    WRCommandH(0x0C);    //显示状态开关: 整体显示开, 光标显示关, 光标显示反白关
```

```
    WRCommandH(0x02);    //地址归零
```

```
}
```

```
//addr为汉字显示位置,*hanzi汉字指针;count为输入汉字串字符数
```

```
void ShowQQCharH(Uchar addr,Uchar *hanzi,Uchar count)
```

```
{
```

```
    Uchar i;
```

```
    WRCommandH(addr);    //设定DDRAM地址
```

```
    for(i=0;i<count;)
```

```
    {
```

```
        WRDataH(hanzi[i*2]);
```

```
        WRDataH(hanzi[i*2+1]);
```

```
        i++;
```

```
    }
```

```
}
```

```
//addr为半宽字符首个地址,i为首个半宽字符代码,count为需要输入字符个数
```

```
void ShowNUMCharH(Uchar addr,Uchar i,Uchar count)
```

```
{
```

```

    Uchar j;
    for(j=0; j<count;)
    {
        WRCommandH(addr);    //设定DDRAM地址
        WRDataH(i+j); //必为两个16*8位字符拼成一个16*16才能显示
        j++;
        WRDataH(i+j);
        addr++;
        j++;
    }
}

//自定义字符写入CGRAM
//data1是高八位,data2是低八位,一存必须存两个字节,横向存两字节,后纵向累加,共16行
//一个自定义字符为16*16点阵
//第一个存入字节为从40H开始,到4F结束为第一个自定义汉字符,之后调出地址从8000H为始第一个
//addr为存入头地址
void WRCGRAMH(Uchar data1,Uchar data2,Uchar addr)
{
    Uchar i;
    for(i=0; i<16;)
    {
        WRCommandH(addr+i);    //设定CGRAM地址
        WRDataH(data1);
        WRDataH(data1);
        i++;
        WRCommandH(addr+i);    //设定CGRAM地址
        WRDataH(data2);
        WRDataH(data2);
        i++;
    }
}

//自定义字符写入CGRAM
//显示上半屏自定义的字符,并把这个字符填满全屏16*16
//addr为显示地址,把自定义字符当一个汉字调出,从8000H开始为第一个,
//8100H为第二个,8200H为第三个,8300H为第四个,中文字库只能自定义四个字符
//i为自定义字符调出地址,先输入低位,再输入高位
//IC决定,中文字库类,一个IC最多只能显示16*2个汉字
void ShowCGCharH(Uchar addr,Uchar i)
{
    Uchar j;
    for(j=0; j<0x20;)
    {

```

```

        WRCommandH(addr+j); //设定DDRAM地址
        WRDataH(0x00); //字符地址低八位
        WRDataH(i); //字符地址高八位
        j++;
    }
}

void WRGDRAM128X8(Uchar x1,Uchar y1,Uchar d1 )
{
    Uchar j,i;
    WRCommandH(0x34); //去扩展指令寄存器
    WRCommandH(0x36); //打开绘图功能
    for(j=0; j<16; j++) //
    {
        WRCommandH(0x80+y1+j); //Y总坐标,即第几行
        WRCommandH(0x80+x1); //X坐标,即横数第几个字节开始写起,80H为第一个字
节
        for(i=0; i<8; i++) //写入一行
        {
            WRDataH(d1);
            WRDataH(d1);
        }
    }
}

//上半屏清除图形区数据
void CLEARGDRAMH(Uchar c)
{
    Uchar j;
    Uchar i;
    WRCommandH(0x34);
    WRCommandH(0x36);
    for(j=0; j<32; j++)
    {
        WRCommandH(0x80+j);
        WRCommandH(0x80); //X坐标
        for(i=0; i<16; i++) //
        {
            WRDataH(c);
            WRDataH(c);
        }
    }
}

```

```
void WRGDRAM1(Uchar x,Uchar l,Uchar r )
{

    Uchar j;
    Uchar i;
    WRCommandH(0x34);          //去扩展指令寄存器
    WRCommandH(0x36);          //打开绘图功能
    //两横的上边框 下边框
    for(j=0;j<2;j++)           //2行
    {
        WRCommandH(0x80+j);    //Y总坐标,即第几行
        WRCommandH(0x80);      //X坐标,即横数第几个字节开始写起,80H为第一个字节
        for(i=0;i<8;i++)       //写入一行
        {
            WRDataH(x);
            WRDataH(x);
        }
        WRCommandH(0x80+30+j); //Y总坐标,即第几行
        WRCommandH(0x80+8);     //X坐标,即横数第几个字节开始写起,80H为第一个字
节
        for(i=0;i<8;i++)       //写入一行
        {
            WRDataH(x);
            WRDataH(x);
        }
    }
    //上半屏两横的右左边框
    for(j=2;j<32;j++)          //30行 左
    {    //先上半屏
        WRCommandH(0x80+j);    //Y总坐标,即第三行开始
        WRCommandH(0x80);      //X坐标,即横数第几个字节开始写起,80H为第一个字节
        WRDataH(l);
        WRDataH(0x00);
        WRCommandH(0x80+j);    //Y总坐标,即第三行开始
        WRCommandH(0x80+7);    //X坐标,即横数第几个字节开始写起,80H为第一个字
节
        WRDataH(0x00);WRDataH(r);
    }

    //下半屏两横右左的边框
    for(j=0;j<30;j++)          //30行
    {    //
        WRCommandH(0x80+j);    //Y总坐标,即第一行开始
        WRCommandH(0x80+15);   //X坐标,即横数第几个字节开始写起,80H为第一个字
```

节

```
WRDataH(0x00);
WRDataH(r);
WRCommandH(0x80+j); //Y总坐标,即第几行
WRCommandH(0x80+8); //X坐标,即横数第几个字节开始写起,80H为第一个字
```

节

```
WRDataH(1); WRDataH(0x00);
}
}

//P3.2按键中断
void ini_int1(void)
{
EA=1;
EX0=1;//允许外部INT0的中断
IT0=1;// 允许中断
}

int scankey1() interrupt 0 using 3 //使用外部中断1,寄存器组3
{
while(P3^2==0)
{for(;;)
{;}
}
IE1=0;//中断标志清零
}
```

//~~~~~程序从这里开始

//主函数

```
void main(void)
```

```
{
```

//~~~~~测试

```
for(m1=0;m1<50;m1++)
```

```
{
```

```
ini_int1();//开中断
```

```
WaitNms(250);
```

```
LCDInit();//初始化
```

```
ShowNUMCharH(0x80,0x01,32);//显示半宽特殊符号
```

```
ShowNUMCharH(0x90,0x30,32);//显示半宽0~?数字标点
```

```
WaitNms(250); //等待时间
```

//~~~~~1,显示8\*16字符

```
LCDInit();//初始化
```

```
WRCommandH(0x01); //清除显示DDRAM
```

```

WRCGRAMH(0xff,0x00,0x40); //写入横（自编特殊符号）
WRCGRAMH(0x00,0xff,0x50); //写入横2
WRCGRAMH(0xaa,0xaa,0x60); //写入竖
WRCGRAMH(0x55,0x55,0x70); //写入竖2

ShowCGCharH(0x80,0x00); //显示横并填满
WaitNms(250);           //等待时间
//~~~~~2, 隔横显示

WRCCommandH(0x01);      //清除显示DDRAM
ShowCGCharH(0x80,02); //显示横2并填满
WaitNms(250);           //等待时间
//~~~~~3, 隔横显示

WRCCommandH(0x01);      //清除显示DDRAM
ShowCGCharH(0x80,04); //显示竖并填满
WaitNms(250);           //等待时间
//~~~~~4, 隔列显示

WRCCommandH(0x01);      //清除显示DDRAM
ShowCGCharH(0x80,06); //显示竖2并填满
WaitNms(250);           //等待时间
//~~~~~5, 隔列显示

WRCCommandH(0x01);      //清除显示DDRAM
WRCGRAMH(0x00,0x00,0x40); //清CGRAM1（重复使用自编）
WRCGRAMH(0x00,0x00,0x50); //清CGRAM2

WRCGRAMH(0xaa,0x55,0x40); //写入点（自编特殊符号）
WRCGRAMH(0x55,0xaa,0x50); //写入点2

ShowCGCharH(0x80,00); //显示点并填满
WaitNms(250);           //等待时间
//~~~~~6, 隔点显示

WRCCommandH(0x01);      //清除显示DDRAM
ShowCGCharH(0x80,02); //显示点2并填满
WaitNms(250);           //等待时间
//~~~~~7, 隔点显示

//显示汉字一屏
LCDInit(); //初始化
ShowQQCharH(0x80,uctech,16); //调用字库
ShowQQCharH(0x90,uctech3,16);
CLEARGRAMH(0x00); //清除显示绘图
WRGRAM128X8(0,0,0xff); //单独一行反白
WaitNms(250);           //等待时间
//~~~~~8, 显示内部汉字
//~~~~~演示单行反白

```

```

LCDInit();//初始化
ShowQQCharH(0x92,uctech1,4);//显示'绘晶科技'
ShowQQCharH(0x8A,uctech2,4);//显示' 128*64 '
CLEARGRAMH(0x00);    //清除显示绘图
WRGRAM1(0xff,0xc0,0x03);    //绘图演示-画边框
WaitNms(250);    //等待时间
//~~~~~8, 显示图文混合
//~~~~~文字+边框

LCDInit();//初始化
ShowQQCharH(0x80,uctech6,16);//调用字库
ShowQQCharH(0x90,uctech7,16);
CLEARGRAMH(0xff);    //
WaitNms(250);    //等待时间
//~~~~~9, 显示图文混合
//~~~~~文字+全显（反白）
}

//----- -以下老化测试99小时
//~~~~~文字左移

LCDInit();//初始化
ShowQQCharH(0x80,uctech4,2);//调用字库
ShowQQCharH(0x90,uctech9,8);//调用字库
ShowQQCharH(0x88,uctech8,8);//调用字库
ShowQQCharH(0x98,uctech5,8);//调用字库
for(z=0;z<10;z++)

{
    WRCommandH(0x80+2);//写地址
    WRDataH(0x3a);
    WRDataH(0x30+z); //分10
    for(z1=0;z1<10;z1++)
    {
        WRCommandH(0x80+3);//写地址
        WRDataH(0x30+z1);    //分10
        WRDataH(0x3a);
        for(d=0;d<6;d++)
        {
            for(d1=0;d1<10;d1++)
            {
                WRCommandH(0x80+4);//写地址
                WRDataH(0x30+d); //分10
                WRDataH(0x30+d1);    //分01
            }
        }
    }
}
//
WRCommandH(0x10);
for(s=0;s<6;s++)

```

```
{
    WRCommandH(0x80+5); //写地址
    WRDataH(0x3a);
    WRDataH(0x30+s); //秒10

    for(s1=0;s1<10;s1++)
    {
        WRCommandH(0x80+6); //写地址
        WRDataH(0x30+s1);           //秒01
        WRDataH(0x3a);
        WaitNms(5); ///延时 x ms
        WRCommandH(0x18);
        for(s10=0;s10<10;s10++)
        {
            WaitNms(5); ///延时 x ms
            for(s100=0;s100<10;s100++)
            {
                WRCommandH(0x80+7); //写地址
                WRDataH(0x30+s10); //100MS
                WRDataH(0x30+s100); //10MS
                WaitNms(5); ///延时 x ms
            }
        }
    }
}

//
```

完