# 12864 液晶

一、概述

带中文字库的 128X64 是一种具有 4 位/8 位并行、2 线或 3 线串行多种接口方式，内部含有国标一级、二级简体中文字库的点阵图形液晶显示模块；其显示分辨率为 128×64,

内置 8192 个 16*16 点汉字，和 128 个 16*8 点 ASCII 字符集.利用该模块灵活的接口方式和简单、方便的操作指令，可构成全中文人机交互图形界面。可以显示 8×4 行 16×16 点阵的汉字.

也可完成图形显示.低电压低功耗是其又一显著特点。由该模块构成的液晶显示方案与同类型的图形点阵液晶显示模块相比，不论硬件电路结构或显示程序都要简洁得多，且该模块的价格也略低于相同点阵的图形液晶模块。

基本特性:

1 低电源电压（VDD:+3.0--+5.5V）
1 显示分辨率:128×64 点
1 内置汉字字库,提供8192 个16×16 点阵汉字(简繁体可选)
1 内置 128 个 16×8 点阵字符
1 2MHZ 时钟频率
1 显示方式：STN、半透、正显
1 驱动方式：1/32DUTY，1/5BIAS
1 视角方向：6 点
1 背光方式:侧部高亮白色 LED,功耗仅为普通 LED 的 1/5—1/10
1 通讯方式：串行、并口可选
1 内置 DC-DC 转换电路，无需外加负压
1 无需片选信号，简化软件设计
1 工作温度：0℃ - +55℃ ,存储温度：-20℃ - +60℃

模块接口说明

2.1 串口接口管脚信号

| 管脚号 | 名称 | LEVEL | 功能 |
|---|---|---|---|
| 1 | VSS | 0V | 电源地 |
| 2 | VDD | +5V | 电源正(3.0V——5.5V) |
| 3 | V0 | - | 对比度（亮度）调整 |
| 4 | CS | H/L | 模组片选端，高电平有效 |
| 5 | SID | H/L | 串行数据输入端 |
| 6 | CLK | H/L | 串行同步时钟：上升沿时读取 SID 数据 |
| 15 | PSB | L | L：串口方式（见注释 1） |
| 17 | /RESET | H/L | 复位端，低电平有效（见注释 2） |
| 19 | A | VDD | 背光源电压+5V（见注释 3） |
| 20 | K | VSS | 背光源负端 0V （见注释 3） |

*注释 1：如在实际应用中仅使用串口通讯模式，可将 PSB 接固定低电平，也可以将模块上的 J8 和“GND”用焊锡短接。

*注释 2：模块内部接有上电复位电路，因此在不需要经常复位的场合可将该端悬空。

*注释 3：如背光和模块共用一个电源，可以将模块上的 JA、JK 用焊锡短接。

2.2 并行接口

| 管脚号 | 管脚名称 | 电平 | 管脚功能描述 |
|---|---|---|---|
| 1 | VSS | 0V | 电源地 |
| 2 | VCC | 3.0+5V | 电源正 |
| 3 | V0 | - | 对比度（亮度）调整 |
| 4 | RS(CS) | H/L | RS=“H”,表示 DB7——DB0 为显示数据<br>RS=“L”,表示 DB7——DB0 为显示指令数据 |
| 5 | R/W(SID) | H/L | R/W=“H”,E=“H”,数据被读到 DB7——DB0<br>R/W=“L”,E=“H→L”， DB7——DB0 的数据被写到 IR 或 DR |
| 6 | E(SCLK) | H/L | 使能信号 |
| 7 | DB0 | H/L | 三态数据线 |
| 8 | DB1 | H/L | 三态数据线 |
| 9 | DB2 | H/L | 三态数据线 |
| 10 | DB3 | H/L | 三态数据线 |

| | | | |
|---|---|---|---|
| 11 | DB4 | H/L | 三态数据线 |
| 12 | DB5 | H/L | 三态数据线 |
| 13 | DB6 | H/L | 三态数据线 |
| 14 | DB7 | H/L | 三态数据线 |
| 15 | PSB | H/L | H：8 位或 4 位并口方式，L：串口方式（见注释 1） |
| 16 | NC | - | 空脚 |
| 17 | /RESET | H/L | 复位端，低电平有效（见注释 2） |
| 18 | VOUT | - | LCD 驱动电压输出端 |
| 19 | A | VDD | 背光源正端（+5V）（见注释 3） |
| 20 | K | VSS | 背光源负端（见注释 3） |

## 接口定义

| 引脚号 | 标识 | 说明 |
| --- | --- | --- |
| PIN1 | GND | 接0V |
| PIN2 | VCC | 接4.8V-5V |
| PIN3 | V0 | VCC和VEE接可调电阻，中间抽头接至V0 |
| PIN4 | RS CS | 并行模式：RS=0，指令寄存器；RS=1，数据寄存器。 串行模式：片选 |
| PIN5 | R/W SID | 并行模式：R/W=0，写；R/W=1，读。 串行模式：数据 |
| PIN6 | E SCK | 并行模式：允许信号。串行模式：脉冲 |
| PIN7 | D0 | 并行模式：数据0； 串行模式：不连接 |
| PIN8 | D1 | 并行模式：数据1； 串行模式：不连接 |
| PIN9 | D2 | 并行模式：数据2； 串行模式：不连接 |
| PIN10 | D3 | 并行模式：数据3； 串行模式：不连接 |
| PIN11 | D4 | 并行模式：数据4； 串行模式：不连接 |
| PIN12 | D5 | 并行模式：数据5； 串行模式：不连接 |
| PIN13 | D6 | 并行模式：数据6； 串行模式：不连接 |
| PIN14 | D7 | 并行模式：数据7； 串行模式：不连接 |
| PIN15 | PSB | 并行模式：PSB=1；串行模式：PSB=0 |
| PIN16 | NC | 不需连接 |
| PIN17 | /RST | 复位 |
| PIN18 | NC | 不需连接 |
| PIN19 | LED+ | 背光正极，接4.8V - 5V |
| PIN20 | LED- | 背光负极，接0V |

## 基本参数

| 参数 | 说明 |
| --- | --- |
| 驱动芯片 | ST7920 ST7921 |
| 背光 | 黄光 /蓝光 |
| 字色 | 黑色 /白色 |
| 字库 | 中文，英文，数字，基本符号 |
| 类型 | STN |
| 液晶模块尺寸(mm) | 93 * 70 * 14 |

*注释 1：如在实际应用中仅使用并口通讯模式，可将 PSB 接固定高电平，也可以将模块上的 J8 和“VCC”用焊锡短接。

*注释 2：模块内部接有上电复位电路，因此在不需要经常复位的场合可将该端悬空。

*注释 3：如背光和模块共用一个电源，可以将模块上的 JA、JK 用焊锡短接。

**四.模块主要硬件构成说明**

**控制器接口信号说明：**

**1、RS，R/W 的配合选择决定控制界面的 4 种模式：**

| RS | R/W | 功能说明 |
|---|---|---|
| L | L | MPU 写指令到指令暂存器（IR） |
| L | H | 读出忙标志（BF）及地址记数器（AC）的状态 |
| H | L | MPU 写入数据到数据暂存器（DR） |
| H | H | MPU 从数据暂存器（DR）中读出数据 |

**2、E 信号**

| E 状态 | 执行动作 | 结果 |
|---|---|---|
| 高——>低 | I/O 缓冲——>DR | 配合/W 进行写数据或指令 |
| 高 | DR——>I/O 缓冲 | 配合 R 进行读数据或指令 |
| 低/低——>高 | 无动作 | |

● 忙标志:BF

BF 标志提供内部工作情况.BF=1 表示模块在进行内部操作,此时模块不接受外部指令和数据.BF=0 时,模块为准备状态,随时可接受外部指令和数据.

利用 STATUS RD 指令,可以将 BF 读到 DB7 总线,从而检验模块之工作状态.● 字型产生 ROM（CGROM）

字型产生 ROM（CGROM）提供 8192 个此触发器是用于模块屏幕显示开和关的控制。DFF=1 为开显示（DISPLAY ON),DDRAM

的内容就显示在屏幕上，DFF=0 为关显示（DISPLAY OFF)。 DFF 的状态是指令 DISPLAY ON/OFF 和 RST 信号控制的。●

显示数据 RAM（DDRAM）模块内部显示数据 RAM 提供 64×2 个位元组的空间，最多可控制 4 行 16 字（64 个字）的中文字型显示，当写入显示数据 RAM 时，可分别显示 CGROM 与 CGRAM 的字型；此模块可显示三种字型，分别是半角英数字型(16*8)、CGRAM 字型及 CGROM 的中文字型，三种字型的选择，由在 DDRAM 中写入的编码选择，在 0000H—0006H 的编码中（其代码分别是 0000、0002、0004、0006 共 4 个）将选择 CGRAM 的自定义字型，02H—7FH 的编码中将选择半角英数字的字型，至于 A1 以上的编码将自动的结合下一个位元组，组成两个位元组的

编码形成中文字型的编码 BIG5（A140—D75F），GB（A1A0-F7FFH）。

● 字型产生 RAM(CGRAM) 字型产生 RAM 提供图象定义(造字)功能,

可以提供四组 16×16 点的自定义图象空间，使用者可以将内部字型没有提供的图象字型自行定义到CGRAM中，便可和CGROM中的定义一样地通过 DDRAM 显示在屏幕中。●

地址计数器 AC 地址计数器是用来贮存 DDRAM/CGRAM 之一的地址,它可由设定指令暂存器来改变，之后只要读取或是写入 DDRAM/CGRAM 的值时，地址计数器的值就会自动加一，当 RS 为“0”时而 R/W 为“1”时，地址计数器的值会被读取到 DB6——DB0 中。

● 光标/闪烁控制电路

此模块提供硬体光标及闪烁控制电路，由地址计数器的值来指定 DDRAM 中的光标或闪烁位置。

原理的另一半在 12864 液晶 原理-2 中

## 五、指令说明

模块控制芯片提供两套控制命令，基本指令和扩充指令如下：

指令表 1：（RE=0：基本指令）

| 指令 | 指令码 | | | | | | | | | | 功能 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
| 清除显示 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 将DDRAM填满"20H",并且设定DDRAM的地址计数器(AC)到"00H" |
| 地址归位 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 设定DDRAM的地址计数器(AC)到"00H",并且将游标移到开头原点位置;这个指令不改变 DDRAM 的内容 |
| 显示状态开/关 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | D=1: 整体显示 ON<br>C=1: 游标 ON<br>B=1:游标位置反白允许 |
| 进入点设定 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | 指定在数据的读取与写入时,设定游标的移动方向及指定显示的移位 |
| 游标或显示移 | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | X | X | 设定游标的移动与显示的移位控制位;这个指令不改变 DDRAM |

| 位控制 | | | | | | | | | | | 的内容 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 功能设定 | 0 | 0 | 0 | 0 | 1 | DL | X | RE | X | X | DL=0/1：4/8 位数据<br>RE=1：扩充指令操作<br>RE=0：基本指令操作 |
| 设定 CGRAM 地址 | 0 | 0 | 0 | 1 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 设定 CGRAM 地址 |
| 设定 DDRAM 地址 | 0 | 0 | 1 | 0 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 设定 DDRAM 地址（显示位址）<br>第一行：80H－87H<br>第二行：90H－97H |
| 读取忙标志和地址 | 0 | 1 | BF | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 | 读取忙标志(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 |
| 写数据到 RAM | 1 | 0 | 数据 | | | | | | | | 将数据 D7——D0 写入到内部的 RAM (DDRAM/CGRAM/IRAM/GRAM) |
| 读出 RAM 的值 | 1 | 1 | 数据 | | | | | | | | 从内部 RAM 读取数据 D7——D0<br>(DDRAM/CGRAM/IRAM/GRAM) |

指令表 2：（RE=1：扩充指令）

| 指令 | 指令码 | | | | | | | | | | 功能 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
| 待命模式 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 进入待命模式,执行其他指令都棵终止<br>待命模式 |
| 卷动地址开关开启 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | SR | SR=1：允许输入垂直卷动地址<br>SR=0：允许输入 IRAM 和 CGRAM 地址 |
| 反白选择 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | R1 | R0 | 选择 2 行中的任一行作反白显示，并可反白与否。初始值 R1R0＝00，第一次设反白显示，再次设定变回正常 |
| 睡眠模 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | SL | X | X | SL=0：进入睡眠模式<br>SL=1：脱离睡眠模式 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 式 | | | | | | | | | | | |
| 扩充功能设定 | 0 | 0 | 0 | 0 | 1 | CL | X | RE | G | 0 | CL=0/1：4/8 位数据<br>RE=1：扩充指令操作<br>RE=0：基本指令操作<br>G=1/0：绘图开关 |
| 设定绘图 RAM 地址 | 0 | 0 | 1 | 0<br>AC6 | 0<br>AC5 | 0<br>AC4 | AC3<br>AC3 | AC2<br>AC2 | AC1<br>AC1 | AC0<br>AC0 | 设定绘图 RAM<br>先设定垂直(列)地址 AC6AC5…AC0<br>再设定水平(行)地址 AC3AC2AC1AC0<br>将以上 16 位地址连续写入即可 |

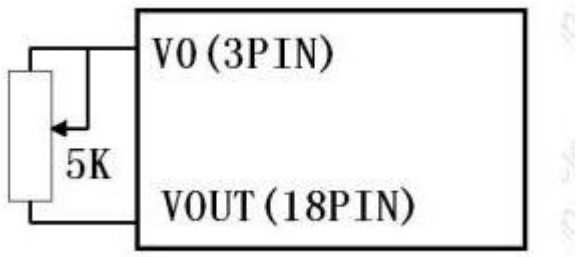
备注;当 IC1 在接受指令前,微处理器必须先确认其内部处于非忙碌状态,即读取 BF 标志时,BF 需为零,方可接受新的指令;如果在送出一个指令前并不检查 BF 标志,那么在前一个指令和这个指令中间必须延长一段较长的时间,即是等待前一个指令确实执行完成。

应用举例：

1 、使用前的准备

先给模块加上工作电压，再按照下图的连接方法调节 LCD 的对比度，使其显示出黑色的底影。

此过程亦可以初步检测 LCD 有无缺段现象。




2、字符显示

带中文字库的 128X64-0402B 每屏可显示 4 行 8 列共 32 个 16×16 点阵的汉字，每个显示 RAM 可显示 1 个中文字符或 2 个 16×8 点阵全高 ASCII 码字符，即每屏最多可实现 32 个中文字符或 64 个 ASCII 码字符的显示。带中文字库的 128X64-0402B 内部提供 128×2 字节的字符显示 RAM 缓冲区（DDRAM）。字符显示是通过将字符显示编码写入该字符显示 RAM 实现的。根据写入内容的不同，可分别在液晶屏上显示 CGROM（中文字库）、HCGROM（ASCII 码字库）及 CGRAM

（自定义字形）的内容。三种不同字符/字型的选择编码范围为：0000～0006H（其代码分别是 0000、0002、0004、0006 共 4 个）显示自定义字型，02H～7FH 显示半宽 ASCII 码字符，A1A0H～F7FFH 显示 8192 种 GB2312 中文字库字形。字符显示 RAM 在液晶模块中的地址 80H～9FH。字符显示的 RAM 的地址与 32 个字符显示区域有着一一对应的关系，其对应关系如下表所示。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 80H | 81H | 82H | 83H | 84H | 85H | 86H | 87H |
| 90H | 91H | 92H | 93H | 94H | 95H | 96H | 97H |
| 88H | 89H | 8AH | 8BH | 8CH | 8DH | 8EH | 8FH |
| 98H | 99H | 9AH | 9BH | 9CH | 9DH | 9EH | 9FH |

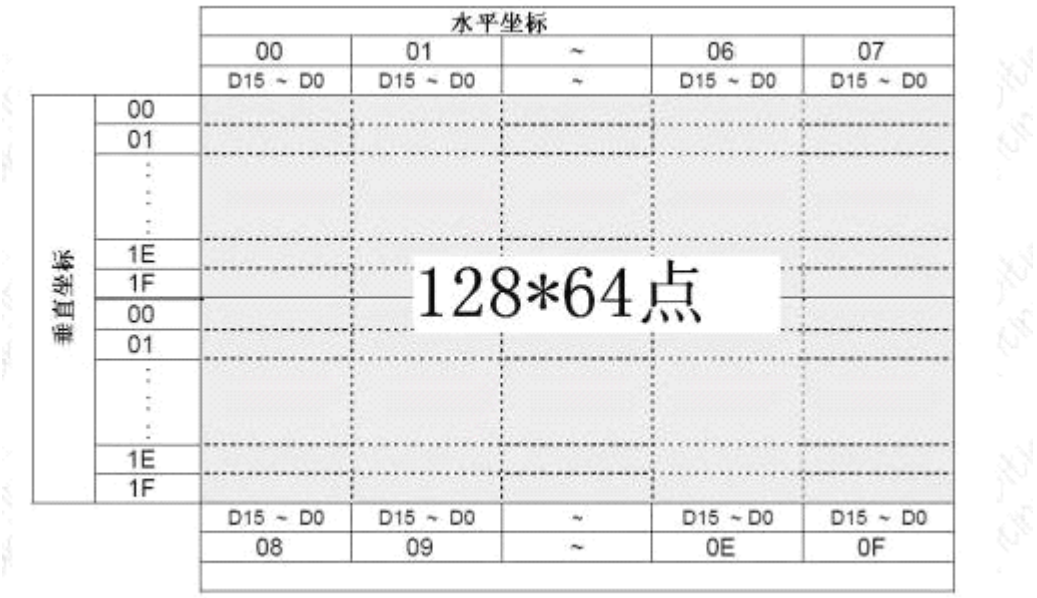
3 、图形显示

先设垂直地址再设水平地址(连续写入两个字节的资料来完成垂直与水平的坐标地址)

垂直地址范围 AC5...AC0

水平地址范围 AC3…AC0

绘图 RAM 的地址计数器（AC）只会对水平地址(X 轴)自动加一,当水平地址=0FH 时会重新设为 00H 但并不会对垂直地址做进位自动加一，故当连续写入多笔资料时，程序需自行判断垂直地址是否需重新设定。GDRAM 的坐标地址与资料排列顺序如下图：




3、应用说明

用带中文字库的 128X64 显示模块时应注意以下几点：

①欲在某一个位置显示中文字符时，应先设定显示字符位置，即先设定显示地址，再写入中文字符编码。

②显示 ASCII 字符过程与显示中文字符过程相同。不过在显示连续字符时，只须设定一次显示地址，由模块自动对地址加 1 指向下一个字符位置，否

则，显示的字符中将会有一个空 ASCII 字符位置。

③当字符编码为 2 字节时，应先写入高位字节，再写入低位字节。

④模块在接收指令前，向处理器必须先确认模块内部处于非忙状态，即读取 BF 标志时 BF 需为“0”，方可接受新的指令。如果在送出一个指令前不检查 BF 标志，则在前一个指令和这个指令中间必须延迟一段较长的时间，即等待前一个指令确定执行完成。指令执行的时间请参考指令表中的指令执行时间说明。⑤“RE”为基本指令集与扩充指令集的选择控制位。当变更“RE”后，以后的指令集将维持在最后的状态，除非再次变更“RE”位，否则使用相同指令集时，无需每次均重设“RE”位。

## 五、指令描述

### 1、显示开/关设置

CODE： R/W D/I DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| L | L | L | L | H | H | H | H | H | H/L |

功能：设置屏幕显示开/关。

DB0=H，开显示；DB0=L，关显示。不影响显示 RAM(DD RAM)中的内容。

### 2、设置显示起始行

CODE： R/W D/I DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| L | L | H | H | 行地址（0～63） | | | | | |

功能：执行该命令后，所设置的行将显示在屏幕的第一行。显示起始行是由 Z 地址计数器控制的，该命令自动将 A0-A5 位地址送入 Z 地址计数器，起始地址可以是 0-63 范围内任意一行。Z 地址计数器具有循环计数功能，用于显示行扫描同步，当扫描完一行后自动加一。

### 3、设置页地址

CODE： R/W D/I DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| L | L | H | L | H | H | H | 页地址（0～7） | | |

功能：执行本指令后，下面的读写操作将在指定页内，直到重新设置。页地址就是 DD RAM 的行地址，页地址存储在 X 地址计数器中，A2-A0 可表示 8 页，读写数据对页地址没有影响，除本指令可改变页地址外，复位信号(RST)可把页地址计数器内容清零。

DD RAM 地址映像表

Y 地址

| 0 | 1 | 2 | ····· | 61 | 62 | 63 | |
|---|---|---|---|---|---|---|---|
| DB0 ∫ DB7 | | | PAGE0 | | | | X=0 |
| DB0 ∫ DB7 | | | PAGE1 | | | | X=1 |

| | |
|---|---|
| ⋮<br>⋮<br>⋮<br>⋮ | |
| DB0<br>∫<br>DB7 PAGE6 | X=7 |
| DB0<br>∫<br>DB7 PAGE7 | X=8 |
| | |

**4、设置列地址**

CODE： R/W D/I DB7 DB6 DB5

| DB4 | DB3 | DB2 | DB1 | DB0 |
|---|---|---|---|---|
| L | L | L | H | 列地址（0～63） |

功能： DD RAM 的列地址存储在 Y 地址计数器中，读写数据对列地址有影响，在对 DD RAM 进行读写操作后，Y 地址自动加一。

**5、状态检测**

CODE： R/W D/I DB7 DB6 DB5

| DB4 | DB3 | DB2 | DB1 | DB0 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| H | L | BF | L | ON/OFF | RST | L | L | L | L |

功能：读忙信号标志位(BF)、复位标志位(RST)以及显示状态位(ON/OFF)。

BF=H：内部正在执行操作； BF=L：空闲状态。

RST=H：正处于复位初始化状态； RST=L：正常状态。

ON/OFF=H：表示显示关闭； ON/OFF=L：表示显示开。

**6、写显示数据**

CODE： R/W D/I DB7 DB6 DB5

| DB4 | DB3 | DB2 | DB1 | DB0 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| L | H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

功能：写数据到 DD RAM，DD RAM 是存储图形显示数据的，写指令执行后 Y 地址计数器自动加 1。D7-D0 位数据为 1 表示显示，数据为 0 表示不显示。写数据到DD RAM 前，要先执行“设置页地址”及“设置列地址”命令。

**7、读显示数据**

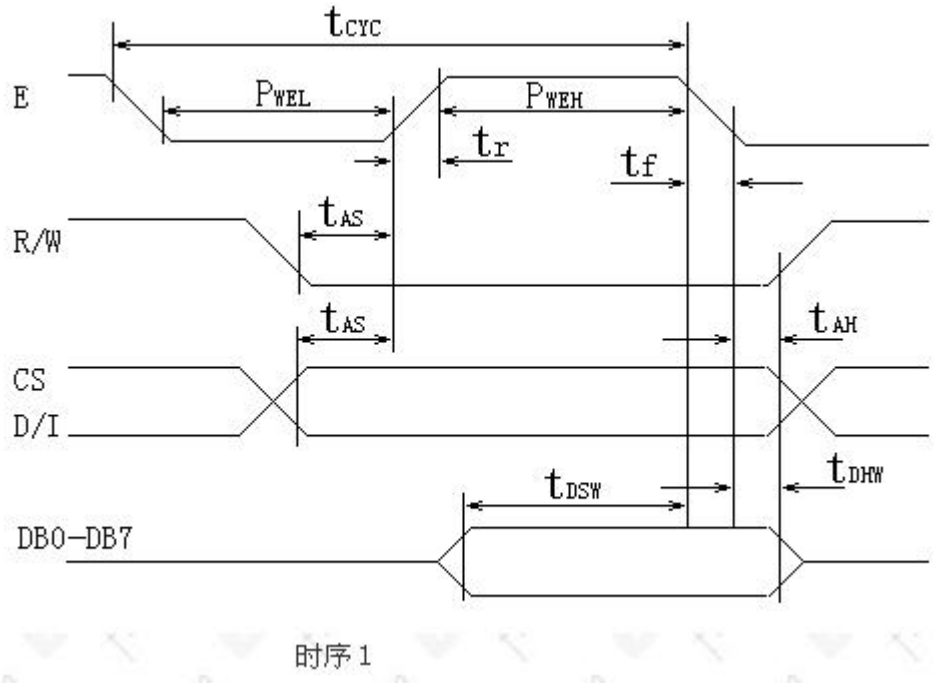
CODE： RS R/W DB7 DB6 DB5

| DB4 | DB3 | DB2 | DB1 | DB0 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| H | H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

功能：从 DD RAM 读数据，读指令执行后 Y 地址计数器自动加 1。从 DD RAM 读数据前要先执行“设置页地址” 及“设置列地址”命令。

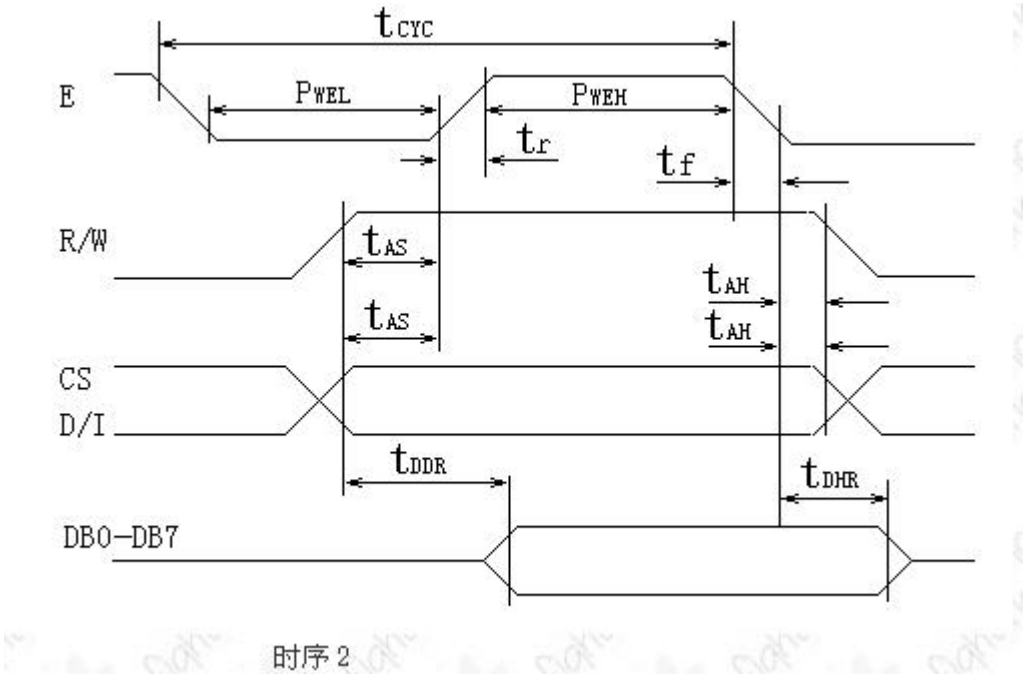
**六、接口时序**

**1.写操作时序**




时序 1

**4. 读操作时序**




时序 2

时序参数表：

| 名称 | 符号 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|
| E 周期时间 | Tcyc | 1000 | | | ns |

| | | | | | |
|---|---|---|---|---|---|
| E 高电平宽度 | Pweh | 450 | | | ns |
| E 低电平宽度 | Pwel | 450 | | | ns |
| E 上升时间 | Tr | | | 25 | ns |
| E 下降时间 | Tf | | | 25 | ns |
| 地址建立时间 | Tas | 140 | | | ns |
| 地址保持时间 | taw | 10 | | | ns |
| 数据建立时间 | Tdsw | 200 | | | ns |
| 数据延迟时间 | Tddr | | | 320 | ns |
| 写数据保持时间 | Tdhw | 10 | | | ns |
| 读数据保持时间 | Tdhr | 20 | | | ns |

## 七、屏幕显示与DD RAM地址映射关系

| | | Y1 | Y2 | Y3 | Y4 | …… | Y62 | Y63 | Y64 | |
|---|---|---|---|---|---|---|---|---|---|---|
| X=0 | Line 0 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB0 |
| | Line 1 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB1 |
| | Line 2 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB2 |
| | Line 3 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB3 |
| | Line 4 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB4 |
| | Line 5 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB5 |
| | Line 6 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB6 |
| | Line 7 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB7 |
| …… …… …… | | | | | | | | | | |
| X=7 | Line60 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB4 |
| | Line61 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB5 |
| | Line62 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB6 |
| | Line63 | 1/0 | 1/0 | 1/0 | 1/0 | …… | 1/0 | 1/0 | 1/0 | DB7 |

## 12864液晶 串行口传输进行图象显示程序-C语言

2007年10月10日 星期三 下午 10:09

```
#include <reg52.h>

#define uint      unsigned int
#define uchar     unsigned char
#define x1        0x80
#define x2        0x88
#define y         0x80
#define comm      0
#define dat       1
```

```
sbit std  = P2^1;
sbit sclk = P2^2;


uchar code tab1[]={
"本系列中文模块内"
"任意位置反白显示"
"置二级字库，可在"
"使用更方便更灵活"
};

uchar code tab32[]={
/*--      调入了一幅图像：F:\梁\画图\HOC012832.bmp      --*/
/*--      宽度 x 高度=128x32      --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,
0x00,0x1E,0x00,0x00,0x00,0x00,0x00,0x07,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,
0x00,0x7F,0x80,0x00,0x00,0x00,0x00,0x18,0x0C,0x00,0x00,0x01,0x00,0x00,0x00,0x00,
0x00,0x00,
0x01,0xFF,0x80,0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x13,0x10,0x03,0xFE,0x00,
0x03,0xFF,0xC0,0x00,0x00,0x00,0x00,0x58,0x00,0x00,0x00,0x3F,0x30,0x1F,0xFF,0xC0,
0x03,0xFF,0xE0,0x00,0x00,0x00,0x00,0x8C,0x03,0xF0,0x00,0x7F,0xE0,0x7C,0x01,0xE0,
0x03,0xFF,0xF0,0x00,0x00,0x00,0x01,0x36,0x06,0xC0,0x00,0x5F,0xC0,0xFF,0xFC,0x60,
0x01,0xFF,0xF0,0x00,0x00,0x00,0x02,0x1B,0x0F,0x80,0x00,0xFF,0x01,0xFE,0x0F,0x30,
0x00,0xEF,0xF0,0x00,0x00,0x00,0x02,0x6D,0x9F,0x00,0x00,0x3E,0x03,0xFF,0xF1,0x90,
0x00,0xFF,0xF8,0x00,0x00,0x00,0x04,0x36,0xFE,0x00,0x01,0xFF,0x07,0xFF,0xFC,0x90,
0x00,0xEF,0xFF,0xFF,0x80,0x00,0x04,0xDB,0x7E,0x00,0x03,0xFF,0x87,0xFF,0xFC,0xD0,
0x00,0x0F,0xFF,0xFF,0xC0,0x00,0x04,0x6D,0xFC,0x00,0x07,0xFF,0x8F,0xFF,0xFE,0x50,
0x00,0x0F,0xFF,0xFF,0xE0,0x00,0x04,0x36,0xFC,0x10,0x07,0xFF,0x8F,0xFF,0xFE,0x90,
0x00,0x0F,0xFF,0xFF,0xE0,0x00,0x04,0x1B,0xF8,0x10,0x07,0xFF,0xCF,0xFF,0xFE,0x80,
0x00,0x0F,0xFF,0xFF,0xF0,0x00,0x04,0x0F,0xF8,0x10,0x07,0xFF,0xFF,0xFF,0xFA,0x00,
```

```
0x00,0x07,0xFF,0xFF,0xF0,0x00,0x04,0x07,0xF0,0x10,0x07,0xFF,0xFF,0xFF,
0xFA,0x00,
0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x02,0x03,0xF0,0x20,0x07,0xFF,0xFF,0xFF,
0xBA,0x00,
0x00,0xFD,0xFF,0xFF,0xFC,0x00,0x02,0x03,0xF0,0x20,0x03,0xFF,0xFF,0xDF,
0xB8,0x00,
0x00,0xC1,0xC0,0x3F,0xFC,0x00,0x01,0x01,0xE0,0x40,0x00,0xFF,0xFF,0xDF,
0xB0,0x00,
0x00,0x81,0xC0,0x3F,0xCE,0x00,0x00,0x81,0xE0,0x80,0x00,0x7F,0xFF,0xDF,
0xA0,0x00,
0x00,0x81,0x80,0x1D,0xCF,0x00,0x00,0x41,0xE1,0x00,0x00,0x3F,0xFF,0x9B,
0x00,0x00,
0x01,0x83,0x80,0x1F,0xC7,0x80,0x00,0x21,0xE2,0x00,0x00,0x1F,0xFD,0xB6,
0x00,0x00,
0x01,0xC3,0x00,0x0E,0xE6,0x80,0x00,0x19,0xEC,0x00,0x00,0x07,0xFE,0x20,
0x00,0x00,
0x00,0xC3,0x00,0x07,0x67,0x40,0x00,0x07,0xF0,0x00,0x00,0x03,0x3E,0x00,
0x00,0x00,
0x00,0x02,0x00,0x03,0xE7,0xA0,0x00,0x00,0x00,0x00,0x00,0x02,0x8E,0x00,
0x00,0x00,
0x00,0x06,0x00,0x03,0x83,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x03,0x00,
0x00,0x00,
0x00,0x06,0x00,0x07,0x03,0x00,0x77,0x46,0x74,0x24,0x80,0x06,0x04,0x00,
0x00,0x00,
0x00,0x1C,0x00,0x06,0x00,0x00,0x55,0x45,0x54,0x57,0x80,0x00,0x00,0x00,
0x00,0x00,
0x00,0x1C,0x00,0x0E,0x00,0x00,0x45,0x45,0x74,0x57,0x80,0x08,0x00,0x00,
0x00,0x00,
0x00,0x00,0x00,0x1C,0x00,0x00,0x55,0x45,0x44,0x74,0x80,0xF0,0x00,0x00,
0x00,0x00,
0x00,0x00,0x00,0x38,0x00,0x00,0x77,0x76,0x47,0x54,0x80,0x00,0x00,0x00,
0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00
};


uchar code tab5[]={
/*--      调入了一幅图像：F:\梁\画图\COCK.bmp      --*/
/*--      宽度 x 高度=128x64      --*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x3D, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x20, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x22, 0x0C, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x22, 0x04, 0x00, 0x02, 0x00, 0x40, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x62, 0x03, 0x00, 0x03, 0x00, 0x40, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0x00, 0x00,
0x00, 0x01, 0xC2, 0x01, 0x00, 0x01, 0x00, 0x40, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0x00, 0x00,
0x00, 0x03, 0x03, 0x00, 0x00, 0x00, 0x80, 0x40, 0x00, 0x00, 0x00,
0x00, 0x00, 0x80, 0x00, 0x00,
0x00, 0x06, 0x01, 0x00, 0x00, 0x00, 0xFF, 0x40, 0x00, 0x00, 0x00,
0x00, 0x01, 0x80, 0x00, 0x00,
0x00, 0x1C, 0x01, 0x00, 0x00, 0x00, 0x40, 0x40, 0x00, 0x00, 0x00,
0x08, 0x01, 0x00, 0x00, 0x00,
0x00, 0x10, 0x01, 0x00, 0x00, 0x00, 0x40, 0x40, 0x00, 0x00, 0x00,
0x00, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0xC1, 0x00, 0x00, 0x00, 0x40, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0x41, 0x00, 0x00, 0x00, 0x4C, 0x80, 0x00, 0x00, 0x00,
0x00, 0x06, 0x00, 0x00, 0x00,
0x00, 0x00, 0x41, 0x00, 0x7C, 0x00, 0x79, 0x80, 0x00, 0x00, 0x80,
0x00, 0xE6, 0x00, 0x00, 0x00,
0x00, 0x00, 0x47, 0xFF, 0xC0, 0x00, 0x00, 0x00, 0x01, 0xE0, 0x78,
0x01, 0xBC, 0x07, 0x00, 0x00,
0x00, 0x00, 0x7C, 0x80, 0x00, 0x00, 0x01, 0xFC, 0x03, 0x31, 0x0C,
0x01, 0x1C, 0x0D, 0x80, 0x00,
0x00, 0x07, 0xC0, 0x80, 0x00, 0x00, 0x0E, 0x00, 0x02, 0x01, 0x04,
0x13, 0x18, 0x18, 0xC0, 0x00,
0x00, 0x7C, 0x40, 0x80, 0x00, 0x00, 0x38, 0x00, 0x06, 0x01, 0x06,
0x12, 0x18, 0x10, 0x45, 0xC0,
0x0F, 0xC0, 0x40, 0x80, 0x00, 0x01, 0xE8, 0x00, 0x04, 0x01, 0x02,
0x12, 0x30, 0x20, 0xC7, 0x38,

0x00, 0x00, 0x40, 0x80, 0x00, 0x1E, 0x04, 0x00, 0x06, 0x01, 0x02,
0x12, 0x30, 0xE0, 0x86, 0x00,
0x00, 0x00, 0x40, 0x80, 0x00, 0x10, 0x04, 0x00, 0x03, 0x01, 0x02,
0x32, 0x50, 0x7F, 0x8C, 0x00,
0x00, 0x00, 0x60, 0x80, 0x00, 0x00, 0x04, 0x00, 0x01, 0xC1, 0x06,
0x23, 0xD8, 0x40, 0x0C, 0x00,
0x00, 0x00, 0x60, 0x40, 0x00, 0x01, 0x04, 0x00, 0x00, 0x63, 0x0C,
0x20, 0x08, 0x40, 0x18, 0x00,
0x00, 0x00, 0xC0, 0x40, 0x40, 0x01, 0x04, 0x70, 0x00, 0x22, 0x10,
0x60, 0x00, 0x30, 0x18, 0x00,
0x00, 0x01, 0xC0, 0x40, 0x40, 0x01, 0x07, 0xC0, 0x00, 0x23, 0xE0,
0x00, 0x00, 0x0F, 0x18, 0x00,
0x00, 0x06, 0x40, 0x40, 0x80, 0x01, 0x04, 0x00, 0x00, 0x2E, 0x00,
0x00, 0x00, 0x00, 0x30, 0x00,
0x00, 0x0C, 0x40, 0x61, 0x80, 0x01, 0x04, 0x00, 0x00, 0xE2, 0x00,
0x00, 0x00, 0x00, 0x30, 0x00,
0x00, 0x18, 0x40, 0x21, 0x00, 0x01, 0x04, 0x00, 0x07, 0x82, 0x00,
0x00, 0x00, 0x00, 0x30, 0x00,
0x00, 0x30, 0x40, 0x23, 0x00, 0x01, 0x04, 0x00, 0x00, 0x02, 0x00,
0x00, 0x00, 0x00, 0x20, 0x00,
0x00, 0xE0, 0x40, 0x26, 0x00, 0x03, 0x04, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x20, 0x00,
0x01, 0x04, 0x40, 0x14, 0x00, 0x07, 0x86, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x06, 0x40, 0x18, 0x00, 0x04, 0x42, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x02, 0xC0, 0x38, 0x10, 0x18, 0x70, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0x80, 0xE8, 0x20, 0x30, 0x18, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0x01, 0x8C, 0x20, 0x60, 0x0F, 0x00, 0x00, 0x08, 0x00,
0x00, 0x00, 0xC0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x04, 0x20, 0x00, 0x01, 0xC0, 0x00, 0x08, 0x00,
0x00, 0x01, 0xF8, 0x00, 0x00,
0x00, 0x00, 0x00, 0x06, 0x40, 0x00, 0x00, 0x70, 0x00, 0x08, 0x01,
0x00, 0x01, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x03, 0x40, 0x00, 0x00, 0x0F, 0x80, 0x08, 0x01,
0x39, 0xF3, 0x04, 0x20, 0x00,
0x00, 0x00, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x00, 0x70, 0x10, 0x01,
0x6F, 0x16, 0x14, 0x3C, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x10, 0x01,
0xCC, 0x14, 0x34, 0x66, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
0x98, 0x34, 0x6C, 0x62, 0x00,

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
0x18, 0x24, 0xF8, 0xC2, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06,
0x10, 0x67, 0xB0, 0x82, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04,
0x00, 0x40, 0x21, 0x82, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1D, 0x82, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x84, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00,
};


/*----------------------------------*/
void delay (uint us)      //delay time
{
    while(us--);
```

```
}
//*************************************


//*************************************

void delay1 (uint ms)
{
     uint i,j;
     for(i=0;i<ms;i++)
     for(j=0;j<15;j++)
     delay(1);
}

//*************************************

/*--------------------------------------*/
//我发现在读指令码的时候，程序先调用了 6 位的低电平，然再在 1 调用了 10 位的指令码 。总共调用了 16 位的数据。
void wr_lcd (uchar dat_comm,uchar content)
{
     uchar a,i,j;
     delay (50);
     a=content;
     sclk=0;
     std=1;
     for(i=0;i<5;i++)
     {
       sclk=1;
       sclk=0;
     }
     std=0;
     sclk=1;
     sclk=0;
     if(dat_comm)
       std=1;      //data
     else
       std=0;      //command
     sclk=1;
     sclk=0;
     std=0;
     sclk=1;
     sclk=0;
     for(j=0;j<2;j++)
     {
```

```
        for(i=0;i<4;i++)
        {
          a=a<<1;
          std=CY;
          sclk=1;
          sclk=0;
        }
        std=0;
        for(i=0;i<4;i++)
        {
          sclk=1;
          sclk=0;
        }
      }
}
//*****************************************

/*-----------------初始化----------------*/
void init_lcd (void)
{
     wr_lcd (comm,0x30);      /*30---基本指令动作*/
     wr_lcd (comm,0x01);      /*清屏，地址指针指向 00H*/
     delay (100);
     wr_lcd (comm,0x06);      /*光标的移动方向*/
     wr_lcd (comm,0x0c);      /*开显示，关游标*/
}
//*****************************************

/*-------------显示点阵---------------*/
//经过调试发现显示点阵是非常有意思的，最主要的在于 data1,data2 上的两组数据的取值，
//不同的取值对应着不同的点阵效果。
void lat_disp (uchar data1,uchar data2)
{
     uchar i,j,k,x;
     x=x1;
     for(k=0;k<2;k++)
     {
       for(j=0;j<16;j++)
       {
         for(i=0;i<8;i++)
         {
           wr_lcd (comm,0x34); //扩充指令操作。
           wr_lcd (comm,y+j*2);
```

```
            wr_lcd (comm,x+i);
            wr_lcd (comm,0x30);
            wr_lcd (dat,data1);
            wr_lcd (dat,data1);
          }
          for(i=0;i<8;i++)
          {
            wr_lcd (comm,0x34);
            wr_lcd (comm,y+j*2+1);
            wr_lcd (comm,x+i);
            wr_lcd (comm,0x30);
            wr_lcd (dat,data2);
            wr_lcd (dat,data2);
          }
        }
        x=x2;
      }
      wr_lcd (comm,0x36);
}
/*-------------------------------------------*/

/*--------------显示汉字或字符----------------*/
void chn_disp (uchar code *chn)
{
      uchar i,j;
      wr_lcd (comm,0x30);
      wr_lcd (comm,0x80);
      for (j=0;j<4;j++)
      {
        for (i=0;i<16;i++)
        wr_lcd (dat,chn[j*16+i]);
      }
}
//*********************************************

/*-------------------------------------------*/
//当 data1=0xff,data2=0xff 时,在 x0,y0 处反白显示 16x1*y1.我发现如果固定 yo 的值为 0x80，
//那么再去改变 x0 的值的话，x0 的值是多少就在那一块反白，反白的范围应该是 32*16。但前提是
//x1=2,y1=16.
//如果 x1=1,y1=8. 那么反白的范围应该是 16*8。
void con_disp (uchar data1,uchar data2,uchar x0,uchar y0,uchar
x1,uchar y1)
```

```
{
     uchar i,j;
     for(j=0;j<yl;j++)
     {
       for(i=0;i<xl;i++)
       {
         wr_lcd (comm,0x34);
         wr_lcd (comm,y0+j);
         wr_lcd (comm,x0+i);
         wr_lcd (comm,0x30);
         wr_lcd (dat,data1);      //写数据到 RAM.这类语句都是一个道理。
         wr_lcd (dat,data2);
       }
     }
     wr_lcd (comm,0x36);
}
//*****************************************************

/*-------------清 DDRAM-----------------*/
void clrram (void)
{
     wr_lcd (comm,0x30);
     wr_lcd (comm,0x01);      //清除显示指令。
     delay (180);
}
/*-------------------------------------*/

/*-------------下半屏显示图形-------------*/
//显示图形的过程是这样的：首先先设垂直地址再设水平地址(连续写入两个字节的资料来完成垂直与水平的坐标地址)
//然后在每个地址里写上 16 位的数据。
void img_disp1 (uchar code *img)
{
     uchar i,j;
     for(j=0;j<32;j++)
     {
       for(i=0;i<8;i++)
       {
         wr_lcd (comm,0x34);
         wr_lcd (comm,y+j);
         wr_lcd (comm,x2+i);
         wr_lcd (comm,0x30);
         wr_lcd (dat,img[j*16+i*2]);
         wr_lcd (dat,img[j*16+i*2+1]);
```

```
        }
      }
      wr_lcd (comm,0x36); //扩充功能指令，开绘图开关。
}
//*****************************************

/*----------------显示图形-----------------*/
void img_disp (uchar code *img)
{
      uchar i,j;
      for(j=0;j<32;j++)
      {
        for(i=0;i<8;i++)
        {
          wr_lcd (comm,0x34); //扩充指令操作。
          wr_lcd (comm,y+j); //设定绘图 RAM 地址。
          wr_lcd (comm,x1+i); //同上。
          wr_lcd (comm,0x30); //基本指令操作。
          wr_lcd (dat,img[j*16+i*2]);
          wr_lcd (dat,img[j*16+i*2+1]);
        }
      }
      for(j=32;j<64;j++)
      {
        for(i=0;i<8;i++)
        {
          wr_lcd (comm,0x34);
          wr_lcd (comm,y+j-32);
          wr_lcd (comm,x2+i);
          wr_lcd (comm,0x30);
          wr_lcd (dat,img[j*16+i*2]);
          wr_lcd (dat,img[j*16+i*2+1]);
        }
      }
      wr_lcd (comm,0x36);
}
//*****************************************

/*-----------------主程序-------------------*/
void main ()
{
      SP=0x5f;
      init_lcd ();
      while (1)
```

```
     {
       lat_disp (0x00,0x00);
delay1(1000);
       chn_disp (tab1);
delay1(1000);
       con_disp (0xff,0xff,0x81,0x80,1,8);
       delay1 (4000);
       clrram();
       lat_disp (0x00,0x00);
       img_disp1 (tab32);
       delay1 (4000);
       clrram();
       img_disp (tab5);
       delay1 (8000);
     }
}
```

## 12864 液晶 串行口传输程序-C 语言

2007 年 09 月 20 日 星期四 下午 03:49

```
//12864(ST7920)串口 C51 程序

    #include     <regx51.h>
    #include     <intrins.h>


sbit     E_CLK     =P3^2;//clock     input
                同步时钟输入端
sbit     RW_SID=P3^1;//data     input/output
   串行数据输入、输出端


void     delay(unsigned     int     n)
    {
        unsigned     int     i;
        for(i=0;     i<n;     i++);
    }


     //串行发送一字节数据
void     SendByte(unsigned     char     dat)
    {
            unsigned     char     i;
```

```
                for(i=0;i<8;i++)
                            {
                                        E_CLK=0;
                                        if(dat&0x80)RW_SID=1;else     R
W_SID=0;
                                        E_CLK=1;
                                        dat=dat<<1;
                             }
    }


//串行接收一字节数据
unsigned     char     ReceieveByte(void)
    {
                unsigned     char     i,d1,d2;
                for(i=0;i<8;i++)
                            {
                                        E_CLK=0;delay(100);
                                        E_CLK=1;
                                        if(RW_SID)d1++;
                                        d1=d1<<1;
                             }
                for(i=0;i<8;i++)
                            {
                                        E_CLK=0;delay(100);
                                        E_CLK=1;
                                        if(RW_SID)d2++;
                                        d2=d2<<1;
                             }
                return     (d1&0xF0+d2&0x0F);
    }


//写控制命令
void     SendCMD(unsigned     char     dat)
    {
              SendByte(0xF8);//11111,00,0     RW=0,RS=0     同步标志

                SendByte(dat&0xF0);//高四位
                SendByte((dat&0x0F)<<4);//低四位
    }


//写显示数据或单字节字符
```

```
void     SendDat(unsigned     char     dat)
    {
            SendByte(0xFA);//11111,01,0     RW=0,RS=1
             SendByte(dat&0xF0);//高四位
             SendByte((dat&0x0F)<<4);//低四位
    }


    /*              写汉字到 LCD      指定的位置
             x_add 显示 RAM 的地址
             dat1/dat2 显示汉字编码
    */
void     display(unsigned     char     x_add,unsigned     char     da
t1,unsigned     char     dat2)
    {
             SendCMD(x_add);//1xxx,xxxx      设定 DDRAM      7 位地址
xxx,xxxx 到地址计数器 AC
             SendDat(dat1);
             SendDat(dat2);
    }


    //初始化      LCM
void     initlcm(void)
    {
             delay(100);
             SendCMD(0x30);//功能设置，一次送 8 位数据，基本指令集

           SendCMD(0x0C);//0000,1100        整体显示，游标 off，游标位
置 off
             SendCMD(0x01);//0000,0001      清 DDRAM
             SendCMD(0x02);//0000,0010      DDRAM 地址归位
             SendCMD(0x80);//1000,0000      设定 DDRAM      7 位地址
000，0000 到地址计数器 AC
    }


void    set_wenzi(void)
{
     SendCMD(0x80);//1000,0001      设定 DDRAM      7 位地址 000，0001 到
地址计数器 AC.
             SendDat(0x48);     //将 ASCII 码调出来,显示在液晶屏幕上.
下同
             SendDat(0x65);
```

```
          SendDat(0x6c);
          SendDat(0x6c);
SendDat(0x6f);
SendDat(0x00);
SendDat(0x4d);
SendDat(0x72);
SendDat(0x2e);
SendDat(0x5a);
SendDat(0x68);
SendDat(0x6f);
SendDat(0x75);

SendCMD(0x90);    //设置液晶屏的显示地址,下同.
SendDat(0x4d);
SendDat(0x79);
SendDat(0x20);
SendDat(0x6e);
SendDat(0x61);
SendDat(0x6d);
SendDat(0x65);
SendDat(0x20);
SendDat(0x69);
SendDat(0x73);
SendDat(0x20);
SendDat(0x59);
SendDat(0x75);

SendCMD(0x88);
SendDat(0x32);
SendDat(0x30);
SendDat(0x3f);
SendDat(0x3f);
display(0x8a,0xc4,0xea);    //将中文字调出来."年"的码值是:c4ea.
SendDat(0x3f);
SendDat(0x3f);
display(0x8c,0xd4,0xc2);
SendDat(0x3f);
SendDat(0x3f);
display(0x8e,0xc8,0xd5);

SendCMD(0x98);
display(0x98,0xd0,0xc7);
display(0x99,0xc6,0xda);
SendDat(0x3f);
```

```
        SendDat(0x3f);
        SendDat(0x3f);
        SendDat(0x3f);
        SendDat(0x3a);
        SendDat(0x3f);
        SendDat(0x3f);

}


void      main(void)
     {
                initlcm();       //12864 初始化程序
                set_wenzi();      //将想要设置的文字显示在液晶屏幕上。

      while(1);
}
```

此程序显示出来的图片: