

深圳绘晶科技有限公司

SHEN ZHEN HUI JING TECHNOLOGY CO., LTD

# 液晶显示模块技术手册

HJ12864ZW

T E L : 0755-33860339

F A X : 0755-33860565

公司地址：深圳市宝安区西乡镇流塘新村东区 7 栋 8 楼

E - mail : [huijinglcd-vip@qq.com](mailto:huijinglcd-vip@qq.com)

# 12864ZW 使用说明书

## 液晶显示器使用手册

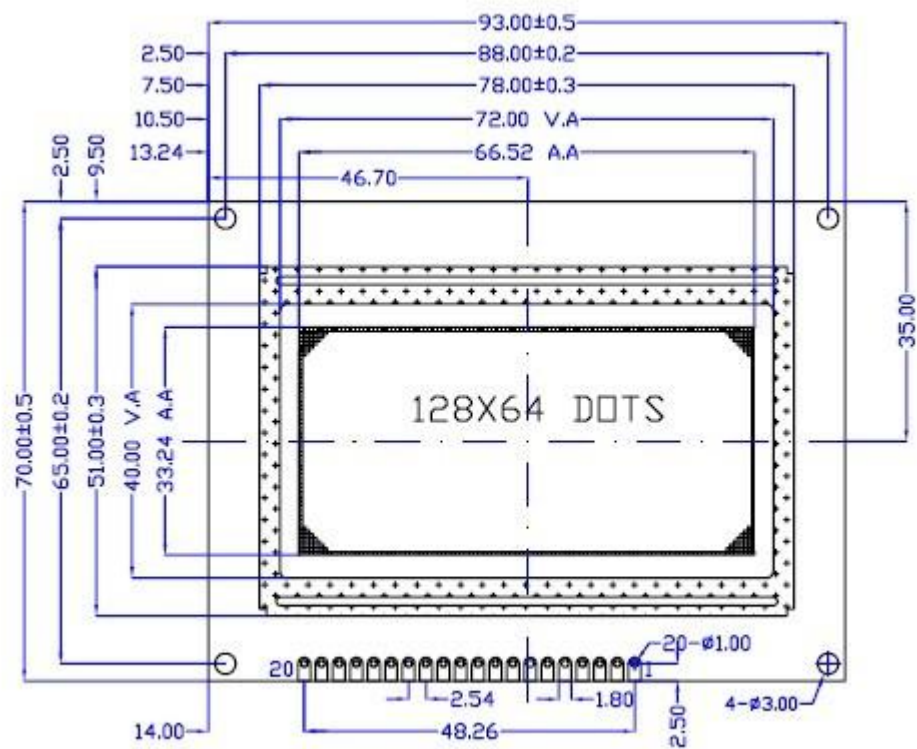
### 目录

- (一) 概述
- (二) 外形尺寸
- (三) 模块外部接口
- (四) 时序说明
- (五) 指令集说明
- (六) 显示步骤
- (七) 初始化时序
- (八) 应用举例

一、概述

HJ12864ZW是一种图形点阵液晶显示器，它主要由行驱动器/列驱动器及128X64全点阵液晶显示器组成，可完成图形显示，也可以显示8X4个（16X16点阵汉字，与外部CPU接口可采用串行或并行方式控制。

二、外形尺寸图



项目	参考值
LCM 尺寸（长×宽×厚）	93.0×70.0×13.5
可视区域（长×宽）	72.0×40.0
点间距（长×宽）	0.52×0.52
点尺寸（长×宽	0.48×0.48
逻辑工作电压（Vdd	+5.0V 或+3.3V（出厂时设定+5.0V）
LCD 驱动电压（Vdd-V0）	+3.0 ~ +5.0V
工作温度（Ta）	0 ~ +50℃（常温）/ -20 ~ +70℃（宽温）
储存温度(Tsto)	-10 ~ +60℃（常温）/ -30 ~ +80℃（宽温）
工作电流（背光除外）	3.0mA(max)

### 三. 模块外部接口

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, 出厂时设定+5.0V)
3	V0	--	LCD 驱动电压(可调)
4	RS(CS)	I	并口方式:
			● RS=0:
			当 MPU 进行读模块操作, 指向地址计数器。
			当 MPU 进行写模块操作, 指向指令寄存器。
			● RS=1:
			无论 MPU 读/写操作, 均指向数据寄存器。
			串口方式:
			CS: 串行片选信号, 高电平有效。
5	R/W(SID)	I	并口方式:
			● R/W=0 写操作。
			● R/W=1 读操作。
			串口方式:
			串行数据输入端
6	E(SCLK)	I	并口方式: 使能信号, 高电平有效。
			串口方式: 串行时钟信号。
7-14	DB0 ~ DB7	I/O	MPU 与模块之间并口的数据传送通道,
			4 位总线模式下 D0 ~ D3 脚断开
15	PSB	I	串/并口控制选择端:
			● H: 并口控制;
			● L: 串口控制。
16	NC	--	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	LED <sub>A</sub>	--	背光电源正端(+3.3V 或+5.0V, 出厂时设定+5.0V)
20	LED <sub>K</sub>	--	背光电源负端(0V)

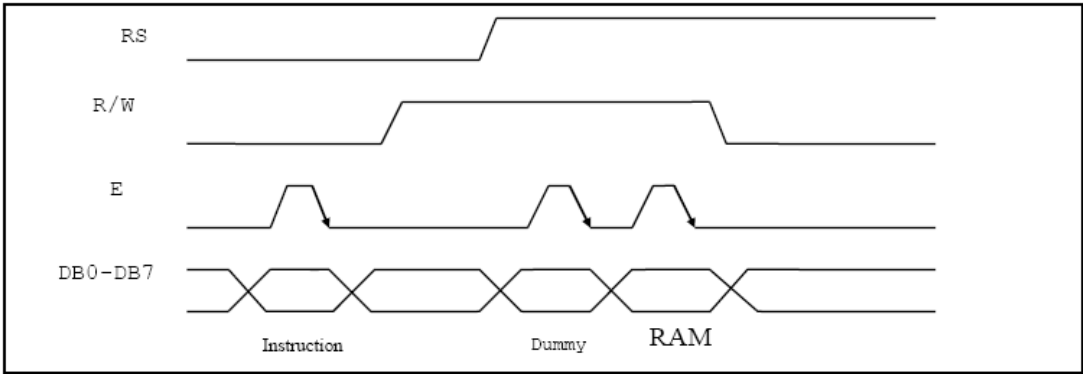
说明: LED<sub>A</sub>与LED<sub>K</sub> 的背光正负极性可根据客户要求变更:

1. 使用并口驱动方式时, PSB=VDD。
2. 使用串口驱动方式时, PSB=VSS, 且DB0~DB7悬空不接。
3. 串并口选择方式: 除接口第15 脚 (PSB) 外, 也可使用P S焊盘点(IC 面)选择并口/串口。
4. 中间与右部焊盘短路, 表示串口控制方式。
5. 中间与左部焊盘短路, 表示并口控制方式

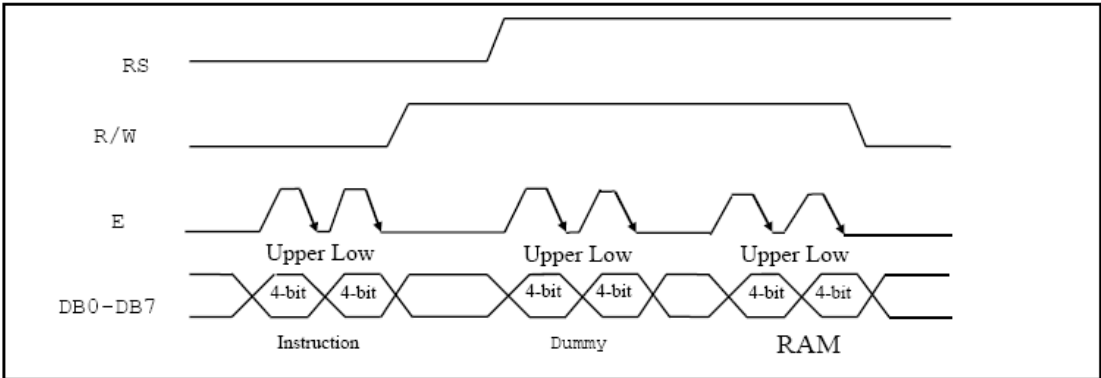
## 四. 时序说明

### 1、并口传输方式:

当 PSB 脚（串/并口选择）接高电平时，模块将进入并口模式，在并口模式下可由指令 DL FLAG 来选择 8-位或 4-位接口，主控制系统将配合( RS、RW、E、DB0..DB7 )来达成传输动作。从一个完整的流程来看，当设定地址指令后 (CGRAM、DDRAM)若要读取数据时需先 DUMMY READ 一次，才会读取到正确数据第二次读取时则不需 DUMMY READ 除非又下设定地址指令才需再次 DUMMY READ。在 4-位传输模式中，每一个八位的指令或数据都将被分为两个字节动作：较高 4（DB7~DB4）的资料将会被放在第一个字节（DB7~DB4）部分，而较低 4 位（DB3~DB0）的资料则会被放在第二个字节的（DB7~DB4）部分，至于相关的另四位则在 4-位传输模式中 DB3~DB0 接口未使用。相关接口传输讯号请参考下图说明：



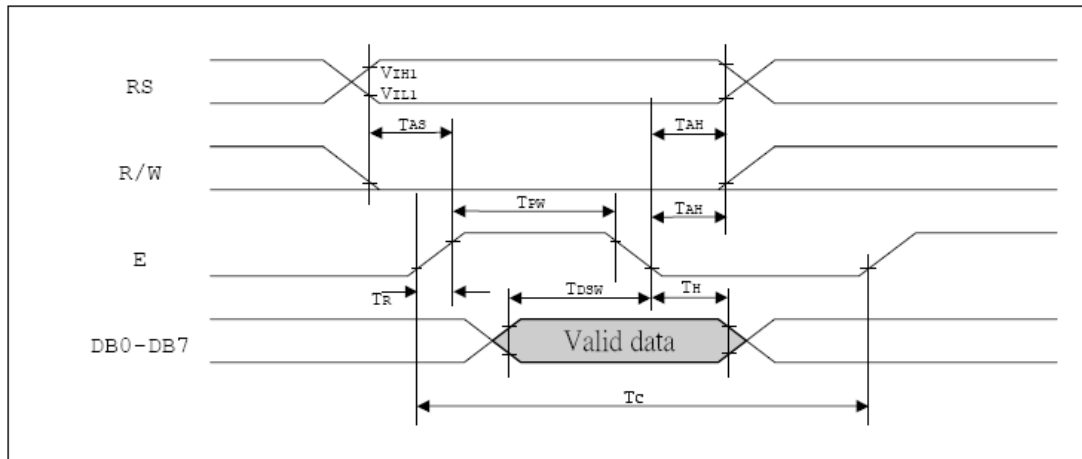
Timing Diagram of 8-bit Parallel Bus Mode Data Transfer



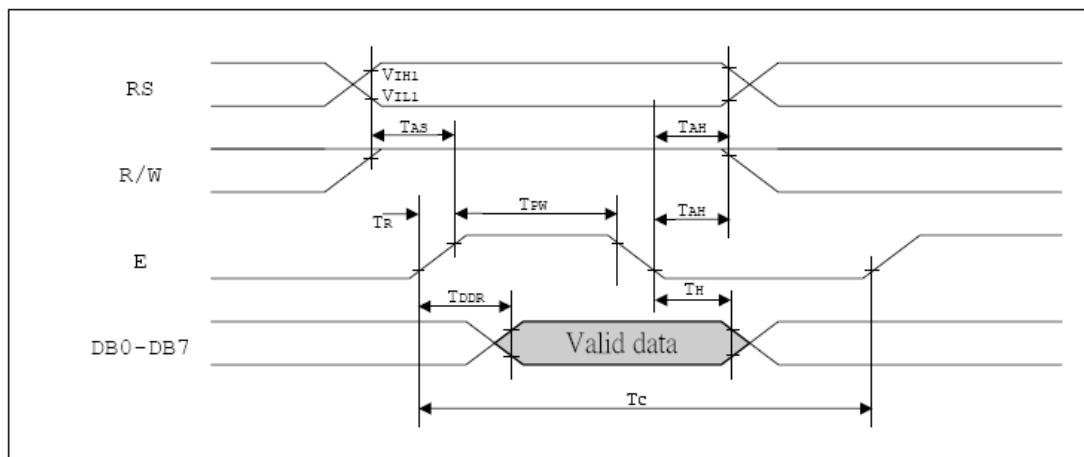
Timing Diagram of 4-bit Parallel Bus Mode Data Transfer

## 2、并行接口时序图

### 1) MPU写数据到液晶显示模块



### 2) MPU从液晶显示模块读数据

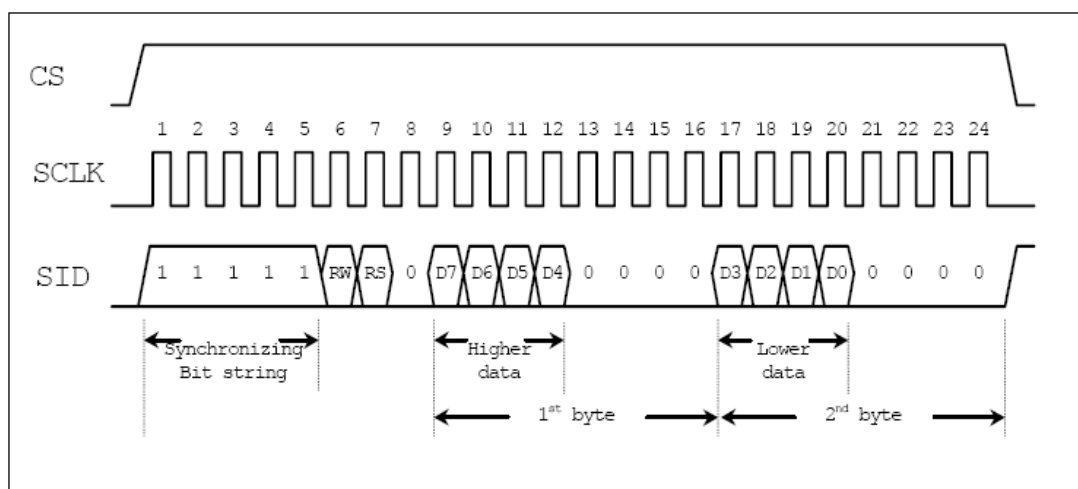


### 3) 并行模式AC特性 (T<sub>A</sub>=25℃,VDD=4.5V)

<i>Internal Clock Operation</i>						
f <sub>OSC</sub>	OSC Frequency	R = 33KΩ	480	540	600	KHz
<i>External Clock Operation</i>						
f <sub>EX</sub>	External Frequency	-	480	540	600	KHz
	Duty Cycle	-	45	50	55	%
T <sub>R</sub> ,T <sub>F</sub>	Rise/Fall Time	-	-	-	0.2	μs
<i>Write Mode (Writing data from MPU to ST7920)</i>						
T <sub>C</sub>	Enable Cycle Time	Pin E	1200	-	-	ns
T <sub>PW</sub>	Enable Pulse Width	Pin E	140	-	-	ns
T <sub>R</sub> ,T <sub>F</sub>	Enable Rise/Fall Time	Pin E	-	-	25	ns
T <sub>AS</sub>	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T <sub>AH</sub>	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T <sub>DSW</sub>	Data Setup Time	Pins: DB0 - DB7	40	-	-	ns
T <sub>H</sub>	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns
<i>Read Mode (Reading Data from ST7920 to MPU)</i>						
T <sub>C</sub>	Enable Cycle Time	Pin E	1200	-	-	ns
T <sub>PW</sub>	Enable Pulse Width	Pin E	140	-	-	ns
T <sub>R</sub> ,T <sub>F</sub>	Enable Rise/Fall Time	Pin E	-	-	25	ns
T <sub>AS</sub>	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T <sub>AH</sub>	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T <sub>DDR</sub>	Data Delay Time	Pins: DB0 - DB7	-	-	100	ns
T <sub>H</sub>	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns

### 3、串口传输方式：

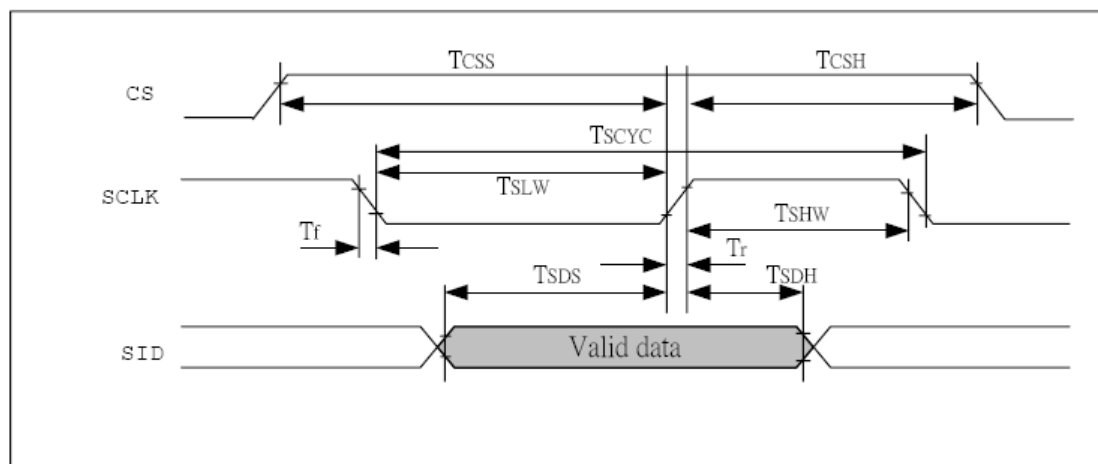
当PSB脚（串/并口选择）接低电位时，模块将进入串口模式。从一个完整的串口传输流程来看，一开始先传输起始字节，它需先接收到五个连续的‘1’（同步位字符串），在起始字节，此时传输计数将被重置并且串行传输将被同步，再跟随的两个位字符串分别指定传输方向位（RW）及寄存器选择位（RS），最后第八的位则为‘0’。在接收到同步位及RW和RS资料的起始字节后，每一个八位的指令将被分为两个字节接收到：较高4位（DB7~DB4）的指令资料将会被放在第一个字节的LSB部分，而较低4位（DB3~DB0）的指令资料则会被放在第二个字节的LSB部分，至于相关的另四位则都为0。串行传输讯号请参考下图说明



Timing Diagram of Serial Mode Data Transfer

### 4、串行接口时序图：

#### （1）MPU写数据到液晶显示模块





(2) 串行模式AC特性 (TA=25°C,VDD=4.5V)

Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
<i>Internal Clock Operation</i>						
f <sub>OSC</sub>	OSC Frequency	R = 33KΩ	470	530	590	KHz
<i>External Clock Operation</i>						
f <sub>EX</sub>	External Frequency	-	470	530	590	KHz
	Duty Cycle	-	45	50	55	%
T <sub>R</sub> ,T <sub>F</sub>	Rise/Fall Time	-	-	-	0.2	μs
TSCYC	Serial clock cycle	Pin E	400	-	-	ns
TSHW	SCLK high pulse width	Pin E	200	-	-	ns
TSLW	SCLK low pulse width	Pin E	200	-	-	ns
TSDS	SID data setup time	Pins RW	40	-	-	ns
TSDH	SID data hold time	Pins RW	40	-	-	ns
TCSS	CS setup time	Pins RS	60	-	-	ns
TCSH	CS hold time	Pins RS	60	-	-	ns

## 五、用户指令集说明:

### 1、指令表一：（RE=0：基本指令集）

指令	指令碼										說明	執行時間 (540KHZ)	
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
清除顯示	0	0	0	0	0	0	0	0	0	1	將 DDRAM 填滿 "20H"，並且設定 DDRAM 的位址計數器 (AC) 到"00H"	1.6 ms	
位址歸位	0	0	0	0	0	0	0	0	0	1	X	設定 DDRAM 的位址計數器 (AC) 到"00H"，並且將游標移到開頭原點位置；這個指令並不改變 DDRAM 的內容	72us
進入點設定	0	0	0	0	0	0	0	0	1	I/D	S	指定在資料的讀取與寫入時，設定游標的移動方向及指定顯示的移位	72us
顯示狀態 開/關	0	0	0	0	0	0	1	D	C		B	D=1: 整體顯示 ON C=1: 游標 ON B=1: 游標位置反白 ON	72 us
游標或顯示 移位控制	0	0	0	0	0	1	S/C	R/L	X	X	X	設定游標的移動與顯示的移位控制位元；這個指令並不改變 DDRAM 的內容	72 us
功能設定	0	0	0	0	1	DL	X	0 RE	X	X	X	DL=1 8-BIT 控制介面 DL=0 4-BIT 控制介面 <u>RE=1: 擴充指令集動作</u> <u>RE=0: 基本指令集動作</u>	72 us
設定 CGRAM 位址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0		設定 CGRAM 位址到位址計數器 (AC) <u>需確認擴充指令中 SR=0 (移動位址或 RAM 位址選擇)</u>	72 us
設定 DDRAM 位址	0	0	1	0 AC6	AC5	AC4	AC3	AC2	AC1	AC0		設定 DDRAM 位址到位址計數器 (AC) AC6 固定為 0	72 us

讀取忙碌旗標 (BF) 和位址	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	讀取忙碌旗標 (BF) 可以確認內部動作是否完成，同時可以讀出位址計數器 (AC) 的值	0 us
寫資料到 RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	寫入資料到內部的 RAM (DDRAM/CGRAM/IRAM/GDRAM)	72 us
讀出 RAM 的值	1	1	D7	D6	D5	D4	D3	D2	D1	D0	從內部 RAM 讀取資料 (DDRAM/CGRAM/IRAM/GDRAM)	72 us

指令表二：（RE=1：扩充指令集）

指令	指令碼										說明	執行時間 (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
待命模式	0	0	0	0	0	0	0	0	0	1	進入待命模式，執行任何其他指令都可終止待命模式 (Com1..32 停止動作,只保留 Com33 ICON 顯示)	72 us
捲動位址或 RAM 位址選擇	0	0	0	0	0	0	0	0	1	SR	SR=1: 允許輸入垂直捲動位址 SR=0: 允許輸入 IRAM 位址(扩充指令) SR=0: 允許設定 CGRAM 位址(基本指令)	72 us
反白選擇	0	0	0	0	0	0	0	1	R1	R0	選擇 4 行中的任一行作反白顯示，並可決定反白與否 R1,R0 初值為 00 當第一次設定時為反白顯示在一次設定時為正常顯示	72 us
擴充功能設定	0	0	0	0	1	DL	X	1	RE	G	DL=1 8-BIT 控制介面 DL=0 4-BIT 控制介面 RE=1: 擴充指令集動作 RE=0: 基本指令集動作 G=1 :繪圖顯示 ON G=0 :繪圖顯示 OFF	72 us
設定 IRAM 位址或捲動位址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR=1: AC5~AC0 為垂直捲動位址 SR=0: AC3~AC0 為 ICON RAM 位址	72 us
設定繪圖 RAM 位址	0	0	1	0	0	0	AC3	AC2	AC1	AC0	設定 GDRAM 位址到位址計數器 (AC) 先設垂直位址再設水平位址(連續寫入兩個位元組的資料來完成垂直與水平的座標位址) 垂直位址範圍 AC6...AC0 水平位址範圍 AC3...AC0	72 us

备注说明:

- 当模块在接受指令前，微处理顺必须先确认模块内部处于非忙碌状态，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迟一段较长的时间，即是等待前一个指令确实执行完成，指令执行的时间请参考指令表中的个别指令说明。
- “RE”为基本指令集与扩充指令集的选择控制位，当变更“RE”位后，往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，不需每次重设“RE”位。

## 2、具体指令介绍：

基本指令（RE=0）

### 1) 清除显示：

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	x

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) 点设置：

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	I/D	S

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	1	D	C	B

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	1	S/C	R/L	x	x

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。

6) 功能设定:

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	X	RE	x	x

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	0	D7	D6	D5	D4	D3	D2	D1	D0

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	1	D7	D6	D5	D4	D3	D2	D1	D0

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	SR

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	R1	R0

功能：选择4 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

R1	R0	Description
L	L	第一行反白或正常顯示
L	H	第二行反白或正常顯示
H	L	第三行反白或正常顯示
H	H	第四行反白或正常顯示

注意：HJ12864系列中，（根据DDRAM关系，有如下变化效果）

R0=0、R1=0：为一、三行反白；

R0=0、R1=1：为二、四行反白。

**4) 睡眠模式：**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	1	SL	X	X

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	x	RE	G	x

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	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

功能: SR=1, AC5~AC0 为垂直卷动地址

7) 绘图RAM地址设定:

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

功能: 设定GDRAM地址到地址计数器 (AC)

## 五、显示步骤:

### 1、显示资料RAM (DDRAM)

显示数据RAM 提供 $64 \times 2$  个字节的空間，最多可以控制4 行16个中文字型（ $16 \times 16$ 点阵），或4行 $\times 32$ 个字符（ $8 \times 16$ ）。当写入显示资料RAM时，可以分别显示CGROM, HCGROM 与CGRAM 的字型；此中文字库系列模块可以显示三种字型，分别是半宽的HCGROM 字型、自定义CGRAM 字型及中文CGROM 字型，三种字型的選擇，由DDRAM 中写入的编码選擇，在0000H ~ 0006H 的编码中将選擇CGRAM 的自定义字型，02H ~ 7FH 的编码中将選擇半宽英数字的字型，至于A1 以上的编码将自动的结合下一个字节，组成两个字节的编码达成中文字型

的编码简繁体中文BIG5（A140 ~ D75F），简体中文GB(A1A0 ~ F7FF)，详细各种字型编码如下：

- 1) 显示半宽字型：将8 位资料写入DDRAM 中，范围为02H ~ 7FH 的编码。
- 2) 显示CGRAM 字型：将16 位资料写入DDRAM 中，总共有0000H, 0002H, 0004H, 0006H 四种编码。
- 3) 显示中文字形：将16 位资料写入DDRAM 中，范围为A140H ~ D75FH 的繁体中文编码(BIG5)，A1A0H ~ F7FFH 的简体中文编码(GB)。将16 位资料写入DDRAM 方式为透过连续写入两个字节的资料来完成，先写入高字节(D15 ~ D8)再写入低字节(D7 ~ D0)。

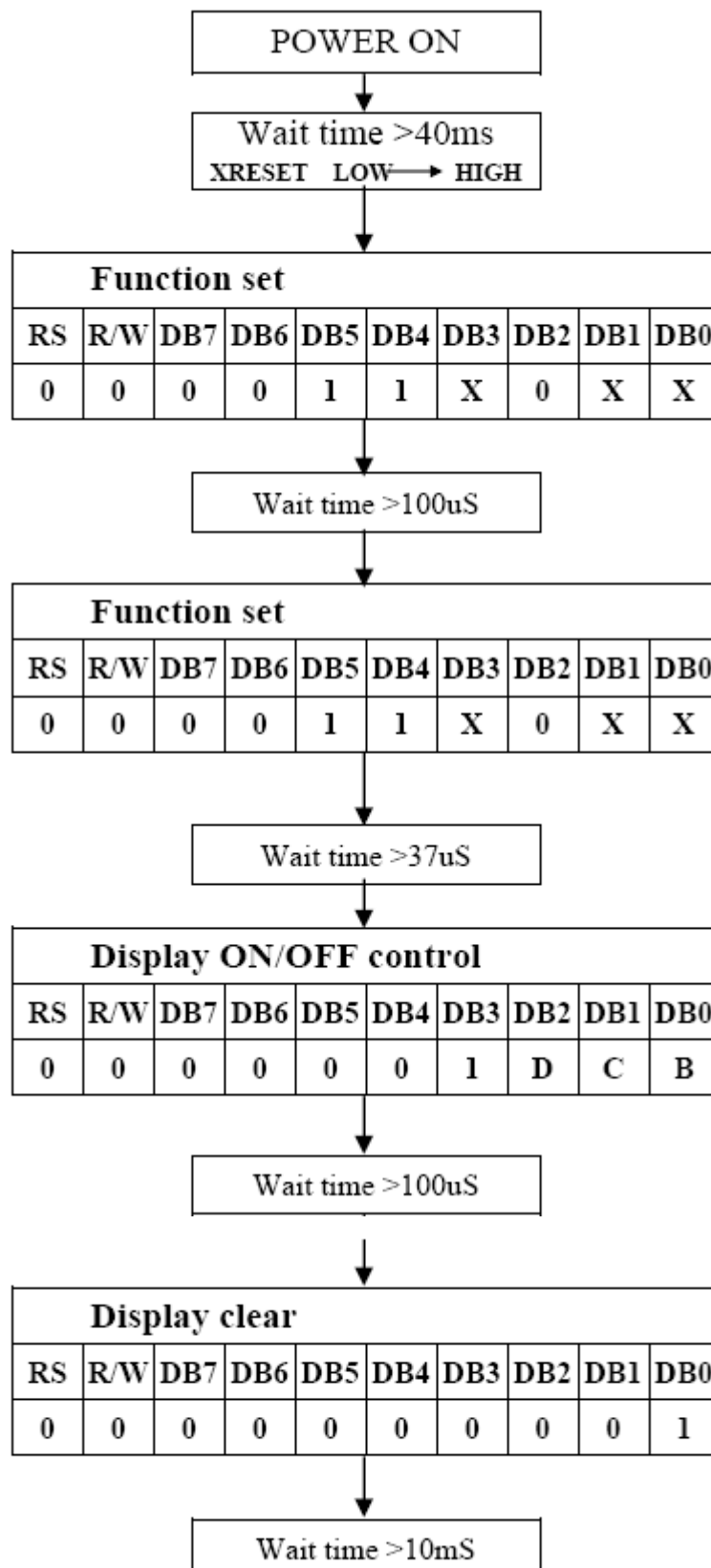
### 2、绘图RAM (GDRAM)

绘图显示RAM 提供 $64 \times 32$  个字节的记忆空间（由扩充指令设定绘图RAM 地址），最多可以控制 $256 \times 64$ 点的二维绘图缓冲空间，在更改绘图RAM 时，由扩充指令设定GDRAM 地址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），再写入两个8 位的资料到绘图RAM，而地址计数器（AC）会自动加一,整个写入绘图RAM 的步骤如下：

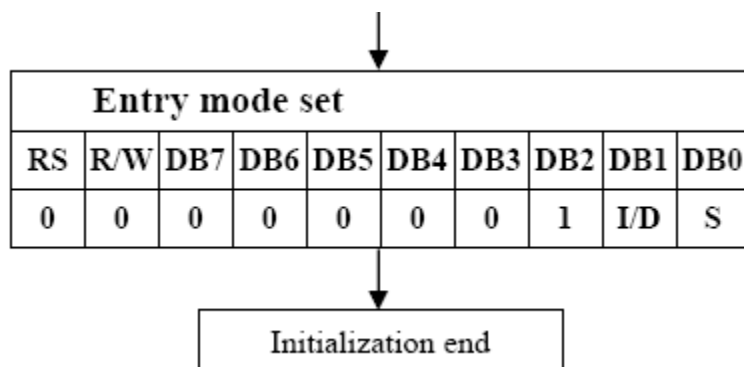
- 1) 绘图清屏。
- 2) 先将垂直的字节坐标（Y）写入绘图RAM 地址。
- 3) 再将水平的字节坐标（X）写入绘图RAM 地址。
- 4) 将D15 ~ D8 写入到RAM 中（写入第一个Bytes）。
- 5) 将D7 ~ D0 写入到RAM 中（写入第二个Bytes）。

## 六、初始化时序：

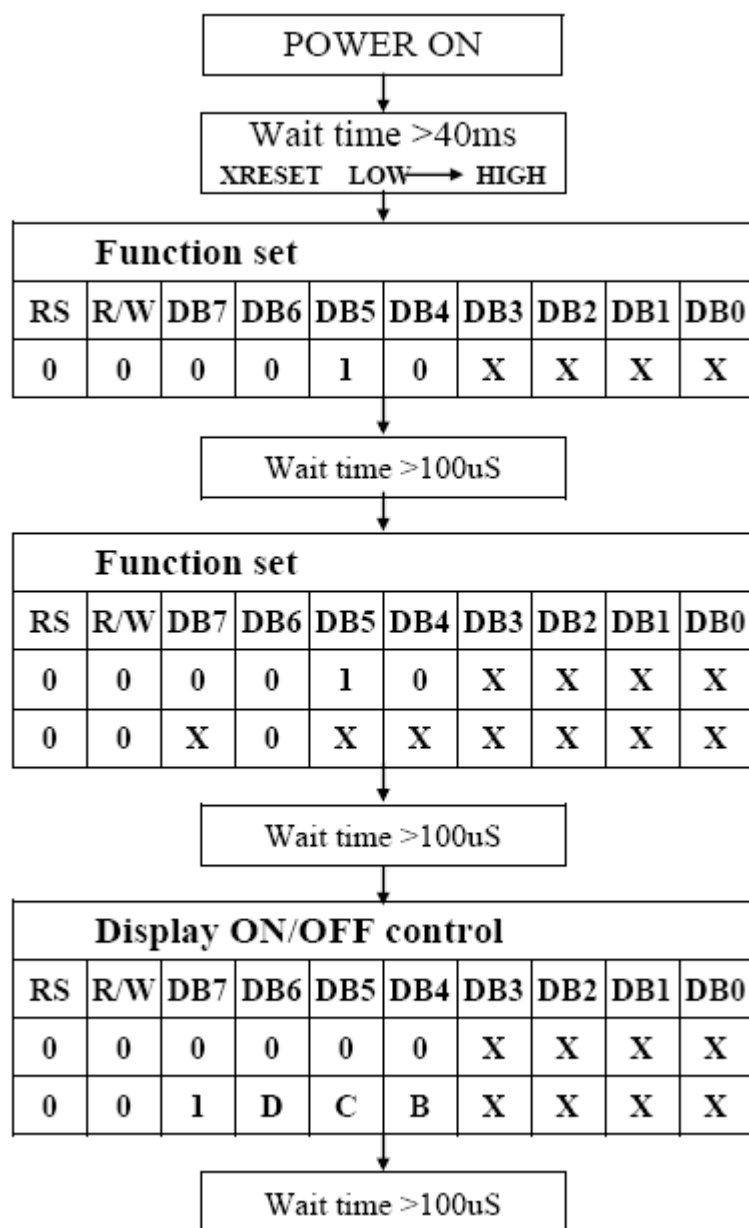
### 1)、8位并联接口：

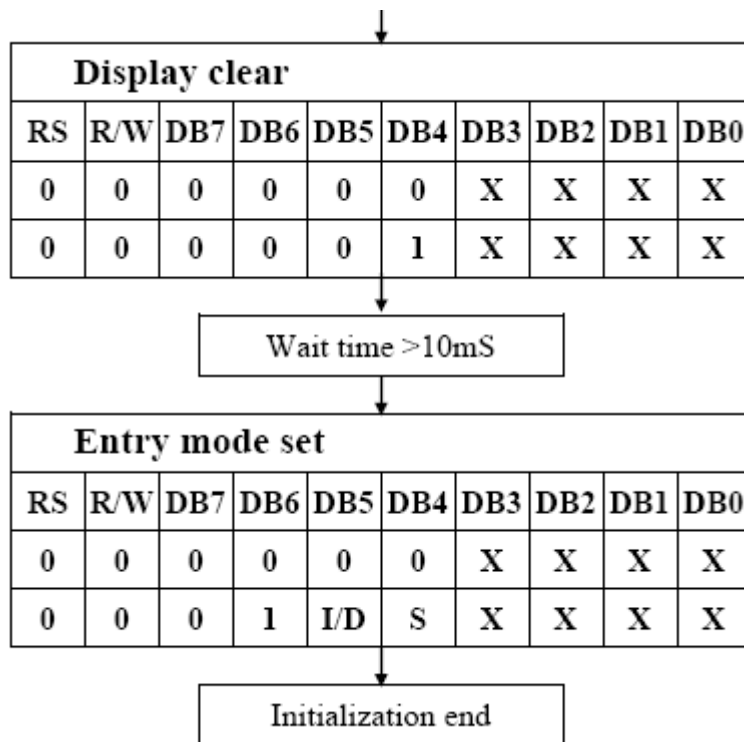






## 2、4位并联接口：





## 八、应用举例：

//深圳绘晶科技HJ12864带M系列(控制器ST7920A), 单片机: 89S52, 晶振: 12M,  
//并行连接方式, P3. 1-RS, P3. 4-RW, P3. 5-E

//设计: B0 LIANG

```
#include<reg52.h>
```

```
#include <intrins.h>
```

```
sbit RS=P3^1; //串口时为CS
```

```
sbit RW=P3^4; //串口为SID
```

```
sbit E=P3^5; //串口为时钟SCLK
```

```
//sbit stop=P3^2;
```

```
sbit PSB=P2^3;
```

```
sbit REST=P2^4;
```

//以下是用<at89x51.h>头文件的定义

```
/*
```

```
#define RS P2_0
```

```
#define RW P2_1 //定义引脚
```

```
#define E P2_2
```

```
#define PSB P2_3
```

```
#define REST P2_4
```

```
#define Data P1
```

```
#include<at89x51.h>
```

```
*/
```

```

#define BF    0x80 //用于检测LCM状态字中的Busy标识
typedef unsigned int Uint;
typedef unsigned char Uchar;
//字符串例子
//"F1--English",也可以往里面写入汉字码,一个汉字由两个码组成
const Uchar
F1English[]={0x46, 0x31, 0x2d, 0x2d, 0x45, 0x6e, 0x67, 0x6c, 0x69, 0x73, 0x68,
0x00};
const Uchar lengthF1=6; //字符串长度
//汉字,直接可以写入字形
unsigned char code uctech[] = {"绘晶科技有限公司"};
const Uchar lengthCF3=8;

Uchar code TAB1[]={
/*-- 调入了一幅图像: C:\Documents and Settings\Administrator\桌面
\12864.bmp --*/
/*-- 宽度x高度=128x64 --*/
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x3F, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, 0x00, 0x0F, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x04, 0x00, 0x03, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x04, 0x00, 0x07, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x04, 0x00, 0x03, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x02, 0x00, 0x03, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x02, 0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x12, 0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x01,
0xBF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x12, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x00

```

0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x32, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x22, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x33, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x63, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x61, 0x80, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0xE1, 0x80, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0xE1, 0xC0, 0x00, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0xE1, 0xC0, 0x01, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x01, 0xE1, 0xC0, 0x01, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x01, 0xF1, 0xE0, 0x03, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x02, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x03, 0xE3, 0xE0, 0x02, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0xFF, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x03, 0xE1, 0xF0, 0x07, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x4B, 0xF1, 0xF8, 0x06, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF1, 0xF8, 0x0F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF5, 0xFC, 0x1F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFD, 0x3F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xDF, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x9C, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xCF, 0xFB, 0xFF, 0x00, 0x7E, 0xFF, 0xFF, 0xF8, 0x0  
1, 0xF4, 0x01,  
0xBF, 0xFF, 0xFF, 0xDE, 0x10, 0x8F, 0xF9, 0xFF, 0x01, 0x2B, 0x39, 0xFF, 0xF8, 0x0  
1, 0xB4, 0x01,  
0xBF, 0xFF, 0xFC, 0x42, 0xF0, 0x1F, 0xFD, 0xFF, 0x80, 0x44, 0x0A, 0xFF, 0xF8, 0x0  
1, 0xBC, 0x01,  
0xBF, 0xFF, 0xFD, 0x07, 0x20, 0x3F, 0xE9, 0xFF, 0xC0, 0x41, 0x45, 0xFF, 0xF8, 0x0  
1, 0xB4, 0x01,  
0xBF, 0xFF, 0xFE, 0xED, 0x5A, 0x8B, 0xC0, 0xFF, 0xC0, 0x05, 0xDD, 0xFF, 0xF8, 0x0

1, 0xF4, 0x01,  
0xBF, 0xFF, 0xFD, 0xB0, 0xF4, 0x20, 0x01, 0x00, 0x62, 0xF7, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x95, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xB4, 0x0F, 0xF9, 0x00, 0x23, 0x7F, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xB7, 0x01,  
0xBF, 0xFF, 0xFF, 0xFC, 0x14, 0x83, 0xFF, 0xF9, 0x3C, 0x08, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x80, 0x01,  
0xBF, 0xFF, 0xFF, 0xE6, 0x56, 0x81, 0xFF, 0xFF, 0xFD, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFD, 0x00, 0xFF, 0xFF, 0xFA, 0x7F, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xDA, 0x87, 0xFF, 0xFF, 0xDA, 0x3F, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xFC, 0x01,  
0xBF, 0xFF, 0xFF, 0xEE, 0xFF, 0x02, 0xFD, 0x7F, 0xF9, 0x7F, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x84, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF8, 0xC0, 0x01, 0x02, 0xDA, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xCC, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF6, 0x11, 0x80, 0xC0, 0x1A, 0x3F, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xB4, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFD, 0x0E, 0xC1, 0xF9, 0x41, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xB4, 0x01,  
0xBF, 0xFF, 0xFF, 0xF8, 0xA2, 0xAF, 0xF0, 0xFF, 0x93, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0xCC, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFD, 0x8F, 0xF8, 0xFF, 0x2E, 0xBF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x85, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFD, 0x57, 0xF8, 0xFF, 0x18, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0x03, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFD, 0x07, 0xF1, 0xFE, 0x47, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE3, 0xF9, 0xFE, 0x3F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0x83, 0xF1, 0xF9, 0x7F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x00, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFA, 0xC1, 0xE1, 0xF8, 0x7F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
0, 0x02, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF9, 0x00, 0xF1, 0xF0, 0x34, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0x5F, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xF7, 0x2C, 0xE1, 0xE4, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0xCC, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFF, 0xD0, 0x61, 0xC0, 0x7F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0x5E, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0xAC, 0x61, 0xC1, 0x67, 0xFF, 0xFF, 0xFF, 0xF8, 0x0  
1, 0x5A, 0x01,  
0xBF, 0xFF, 0xFF, 0xFF, 0xFE, 0x80, 0x21, 0xC0, 0x3F, 0xFF, 0xFF, 0xFF, 0xF8, 0x0



[illegible]

[illegible]



```

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x01,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF
F, 0xFF, 0xFF,

```

```
};
```

//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();
```

```
}
```

```

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 2 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
}

//*****
//*****//
//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
        E=1;
        temp=P1;
        E=0; // E=0
        if ((temp&0x80)==0) break;
    }
}
//写数据到指令寄存器

void WRCommand(Uchar comm)
{
    RDBF();
    RS=0;
    RW=0;
    P1=comm;
    E=1;
    E=0;
}
//写数据到数据寄存器

```

```

void WRData(Uchar TEMP)
{
    RDBF();
    RS=1;
    RW=0;
    P1=TEMP;
    E=1;
    E=0;
    //stopint();
}

////////////////////////////////////
////////////////////////////////////
//以下是串口时开的读写时序
/*void SendByteLCD(Uchar WLCDData)
{
    Uchar i;
    for(i=0; i<8; i++)
    {
        if((WLCDData<<i)&0x80)RW=1;
        else RW=0;
        E=0;
        E=1 ;
    }
}
SPIWR(Uchar Wdata,Uchar WRS)
{
    SendByteLCD(0xf8+(WRS<<1));
    SendByteLCD(Wdata&0xf0);
    SendByteLCD((Wdata<<4)&0xf0);
}
void WRCommand(Uchar CMD)
{
    RS=0;
    RS=1;
    SPIWR(CMD, 0);
    delay10US(90); //89S52来模拟串行通信, 所以, 加上89S52的延时,
}
void WRData(Uchar Data)
{
    RS=0;
    RS=1;
    SPIWR(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;
  WRCommand(0x30); //基本指令集,8位并行

  WRCommand(0x06); //起始点设定:光标右移

  WRCommand(0x01); //清除显示DDRAM

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

  WRCommand(0x02); //地址归零
}
//显示数组字符串(显示半宽字型16*8点阵)
void ShowQQChar(Uchar addr,Uchar *english,Uchar count)
{
  Uchar i;
  WRCommand(addr); //设定DDRAM地址
  for(i=0;i<count;)
  {
    WRData(english[i*2]);
    WRData(english[i*2+1]);
    i++;
  }
}
//显示连续字符串(半宽字符)
void ShowNUMChar(Uchar addr,Uchar i,Uchar count)
{
  Uchar j;
  for(j=0;j<count;)
  {
    WRCommand(addr); //设定DDRAM地址
    WRData(i+j);
    j++;
    WRData(i+j);
  }
}

```

```

        addr++;
        j++;
    }
}
//自定义字符写入CGRAM
void WRCGRAM(Uchar data1,Uchar data2,Uchar addr)
{
    Uchar i;
    for(i=0;i<16;)
    {
        WRCommand(addr+i);    //设定CGRAM地址
        WRData(data1);
        WRData(data1);
        i++;
        WRCommand(addr+i);    //设定CGRAM地址
        WRData(data2);
        WRData(data2);
        i++;
    }
}
//显示自定义的字符,并把这个字符填满全屏16*16
void ShowCGChar(Uchar addr,Uchar i)
{
    Uchar j;
    for(j=0;j<0x20;)
    {
        WRCommand(addr+j);    //设定DDRAM地址
        WRData(0x00);
        WRData(i);
        j++;
    }
}
void CLEARGRAM(void)
{
    Uchar j;
    Uchar i;
    WRCommand(0x34);
    WRCommand(0x36);
    for(j=0;j<32;j++)
    {
        WRCommand(0x80+j);
        WRCommand(0x80); //X坐标
        for(i=0;i<32;i++)//
        {

```

```

        WRData(0x00);
    }
}

}
//写入GDRAM 绘图,Y是Y绘图坐标,2个字节一行,CLONG是图形长度,以字节
//为单位;HIGHT是图形高度,TAB是图形数据表.12864M的图形显示是相当于
256*32点阵.
//由两屏128*32上下两屏组成,同一行的下屏的头地址紧接上屏的末地址。
//绘图在串口输入时,会比在并口下的输入要慢一些
void WRGDRAM(Uchar Y1,Uchar clong,Uchar hight,Uchar *TAB1)
{
    Uint k;
    Uchar j;
    Uchar i;
    WRCommand(0x34);
    WRCommand(0x36);
    for(j=0;j<hight;j++)//32
    { //先上半屏
        WRCommand(Y1+j); //Y总坐标,即第几行
        WRCommand(0x80); //X坐标,即横数第几个字节开始写起
        for(i=0;i<clong;i++)//
        {
            WRData(TAB1[clong*j+i]);
        }
        //后下半屏

        for(k=0;k<clong;k++)//
        {
            WRData(TAB1[clong*(j+hight)+k]);
        }
    }
}

void menu(void)
{
    LCDInit();

    ShowQQChar(0x80,uctech,lengthCF3); //显示' 绘晶科技有限公司',以下
共四行
    ShowQQChar(0x90,uctech,lengthCF3);
    ShowQQChar(0x88,uctech,lengthCF3);
    ShowQQChar(0x98,uctech,lengthCF3);

```

```
//WRGDRAM(0x80, 16, 32, TAB2);  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```
ShowNUMChar(0x80, 0x01, 0x0f); //显示半宽特殊符号  
ShowNUMChar(0x90, 0x30, 0x0f); //显示半宽0~?数字标点  
ShowNUMChar(0x88, 0x41, 0x0f); //显示半宽A~P大写  
ShowNUMChar(0x98, 0x61, 0x0f); //显示半宽a~p小写  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```
WRCGRAM(0xff, 0x00, 0x40); //写入横  
WRCGRAM(0x00, 0xff, 0x50); //写入横2  
WRCGRAM(0xaa, 0xaa, 0x60); //写入竖  
WRCGRAM(0x55, 0x55, 0x70); //写入竖2  
ShowCGChar(0x80, 0x00); //显示横并填满  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```
ShowCGChar(0x80, 02); //显示横2并填满  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```
ShowCGChar(0x80, 04); //显示竖并填满  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```
ShowCGChar(0x80, 06); //显示竖2并填满  
WaitNms(250); //等待时间  
WaitNms(250); //等待时间  
//stopint();  
WRCommand(0x01); //清除显示DDRAM
```

```

    WRGRAM(0x00, 0x00, 0x40); //清GRAM1
    WRGRAM(0x00, 0x00, 0x50); //清GRAM2
    WRGRAM(0xaa, 0x55, 0x40); //写入点
    WRGRAM(0x55, 0xaa, 0x50); //写入点2
    ShowCGChar(0x80, 00); //显示点并填满
    WaitNms(250); //等待时间
    WaitNms(250); //等待时间
    //stopint();
    WRCommand(0x01); //清除显示DDRAM

    ShowCGChar(0x80, 02); //显示点2并填满
    WaitNms(250); //等待时间
    WaitNms(250); //等待时间
    //stopint();
    WRCommand(0x01); //清除显示DDRAM

}

void menu2(void)
{
    CLEARGRAM();
    WRGRAM(0x80, 16, 32, TAB1);
    WaitNms(250); //等待时间
    WaitNms(250); //等待时间
    //stopint();
}

//主函数
void main(void)
{
    ini_int1(); //开中断
    menu(); //初始化及半宽字符和点横竖汉字扫描
    menu2(); //绘图显示
    for(;;)
    {;}
}

```