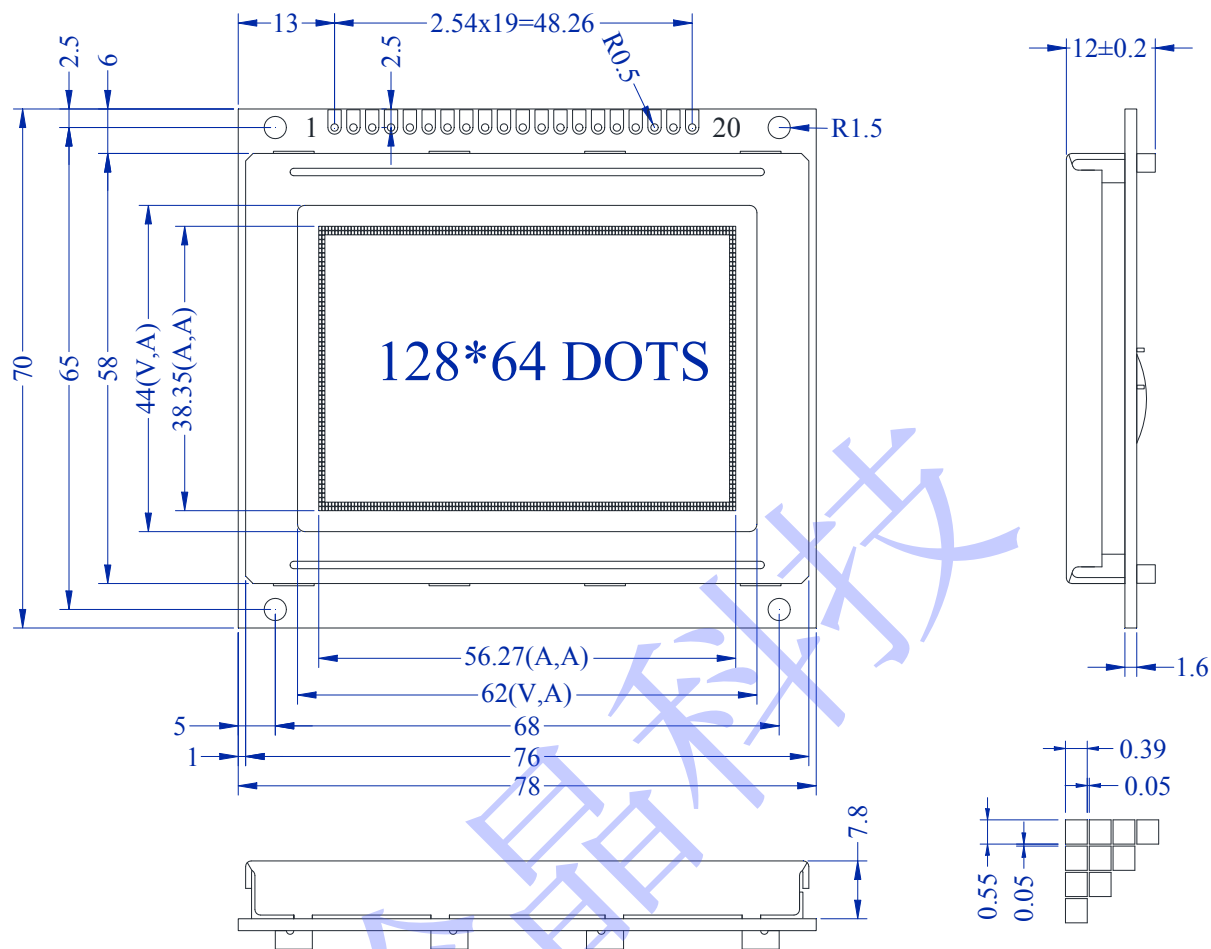


一、模块尺寸



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

✧ 控制器硬件方面

- 工作电压 3.3V 或者 5.0V 可选，默认设定为 5.0V
- 提供与 MPU 通讯方式：8 位并行 8080 时序
- 硬件初始化，完成大部份初始化工作，用户更多时间花在软件编写上
- 128 字型 ROW (CGROM)，ASCII 日文，英文，数据，(默认英文)
- FS：字符大小选择 6X8 或者 8X8 字符选择功能 (预留用户设置)
- /RESET：上电复位重置，低电平重置，高电平正常工作

✧ 控制器软件方面

- 64K 的显示内存供用户自由分配 (文本、图形、自定义字符) 使用 (默认最大 32K)
- 128 个 8X8 点阵的自定义字符 RAM (CGRAM)
 - ◆ 通过设置可以一次装入 32 个汉字或者 64 个汉字，可以利用剩余内存建更多的汉字或者特殊图形。
- 绘图以及文字画面混合显示功能
- 地址自动自动加一或减一功能；可设置平移或滚动效果
- 光标大小形状设置，
- 光标、字符正向、负向、闪烁功能

✧ 模块整体参数

- 驱动 IC 为 SMT 贴片封装
- 预留 VR 可调电阻，可调节对比度 (默认不安装)
- 占空比 1/64 偏压比 1/9
- 可视角 6 点钟 可视角可以根据客户要求做
- 显示颜色效果常用的有：黄绿，蓝白，灰白，可以根据客户要求定制
- 工作温度，宽温：-20 度到+70 度，可以根据客户要求定制

三、接口定义

显示器控制信号端口

接口	名称	方向	说明
1	FG/NC/V0 UT	--	1. 模块铁框引脚或者空脚 2. LCD 电压输出或者输入（默认为输出）用于调节对比度电压
2	VSS	--	电源负端 (0V)
3	VDD	--	电源正端 (+3.3V 或 +5.0V, 默认设定 +5.0V)
4	V0	--	LCD 对比度调节, 取 VOUT 电压为参考 (与 VDD 间 9.4V 时显示效佳)
5	WR#	I	写数据端口, 低电平有效。
6	RD#	I	读数据端口, 低电平有效。
7	CE	I	模块的片选信号, CS 为低时, 模块才能接受指令, 否则不接收指令。
8	C/D	I	指令/数据控制信号, CS 为高时, 控制指令存取; CS 为低时, 控制数据存取。
9	REST	I	用于重置模块, 高电平时为正常模式, 低电平时为控制器重置模式。
10-17	DB0 ~ DB7	I/O	单片机与模块之间并口的数据传送通道。
18	FS	I	字符大小选择, 为高时 6X8 点阵大小, 为低时 8X8 点阵大小
19	LEDA	-	背光电源的正极 (3.3V 或者 5.0V) 默认是 5.0V
20	LEDK	-	背光电源负极/若用户需要也可以空脚

四. 电性参数(直流)

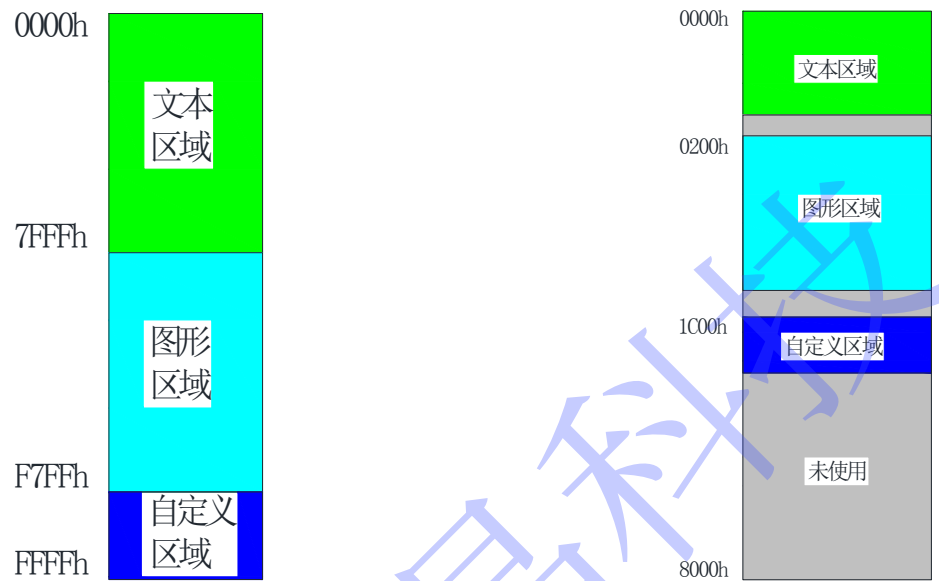
浙江日报科技

TEL: 0755-23146001 FAX: 0755-23146002 www.huijinglcm.com

动加一功能。控制器以这个计数器值为地址从显示存储器中提取图形数据送入液晶显示驱动系统显示。该地址计数器还作为文本属性地址计数器与文本地址计数器一起控制文本的显示效果。

文本地址计数器：16 位计数器。作为显示扫描时文本显示器区的扫描地址。它具有自动加一功能。控制器以这个计数器为地址从显示存储中提取字符代码送至字符字模库的地址线上，然后从所选通的字符字模库单元读出字模数据送入液晶显示驱动系统显示。

CGRAM 偏置地址寄存器：5 位寄存器。作为自定义字符库的高 5 位地址指针圈定了自定义字符库在显示存储器中的 2K 字节的区域，凡控制器从文本显示区提取的字符代码属于自定义字符范围的，都将与这些寄存器的内容一起作为地址选通自定义字符字模单元读出字模数据送入液晶显示驱动系统显示。



可以 64K 显示内存的分配图（未用）

实际 32K 显示内存（本例程的分配图）

b) CGROM 内部字符表

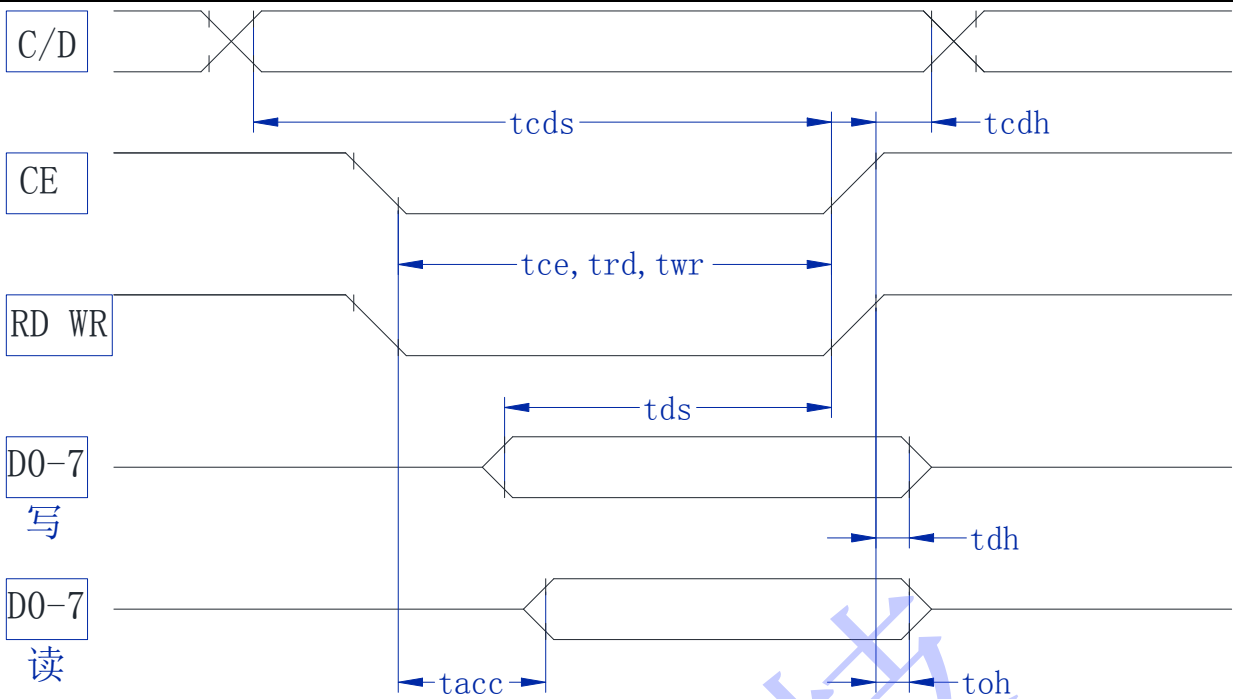
默认英文

LSB MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
1	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
2	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
3	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
4	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
5	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
6	Q	ü	é	à	â	ä	å	q	è	ë	ê	ï	î	ï	Ä	Å
7	É	æ	Ê	Ç	ö	ø	û	ü	ö	ó	ô	õ	ø	æ	À	á

可选日文

LSB MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
1	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
2	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
3	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
4	千	万	円	+	-	■	マ	ア	イ	ウ	エ	オ	カ	キ	ク	ケ
5	一	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
6	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
7	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ウ	ヴ	。

六. 时序图



VDD=+5V±5%,GND=0V,Ta= -20 to +70℃

项目	符号	测试条件	最小	最大	单位
C/D成立时间	Tcde	-	100	-	Ns
C/D保持时间	Tcdr	-	10	-	Ns
CE,RD,WR脉冲宽度	Tce, trd, twr	-	80	-	Ns
数据保持时间	Tacc	-	40	-	Ns
存取时间	Tacc	-	-	150	Ns
输出保持时间	Toh	-	10	50	ns

七. 控制器指令说明

1、控制器指令表:

NO	命令	代码	参数1 (D1)	参数2 (D2)	功能
1	地址指针设置 (Registers)	21h	X地址	Y地址	设置光标指针
		22h	数据	00h	设置偏移寄存器

	Setting)	24h	低位地址	高位地址	设置地址指针
2	显示区域设置(Set Control Word)	40h	低位地址	高位地址	文本显示区首地址
		41h	列数据	00h	文本显示区宽度
		42h	低位字节	高位字节	图形显示区首地址
		43h	列数据	00h	图形显示区宽度
3	显示方式设置 (Mode Set)	80h	-	-	文本与图形以逻辑“或”
		81h	-	-	文本与图形以逻辑“异或”
		83h	-	-	文本与图形以逻辑“与”
		84H	-	-	文本显示特征以双字节表示
		80h	-	-	00-7FH CGROM(自带128个ASCII);80-FFH CGRAM(用户自定义, 可放汉字32个)
		88h	-	-	00-FFH CGRAM(用户自定义, 可放汉字64个)
4	显示状态设置 (display mode)	90h	-	-	显示关
		92h	-	-	光标开, 不闪烁
		93h	-	-	光标开, 并闪烁
		94h	-	-	文本开, 图形关
		98h	-	-	文本关, 图形开
		9ch	-	-	文本开, 图形开
5	光标形状设置 (Cursor Pattern Select)	A0H	-	-	1x8光标
		A1H	-	-	2x8光标
		A2h	-	-	3x8光标
		A3h	-	-	4x8光标
		A4h	-	-	5x8光标
		A5h	-	-	6x8光标
		A6h	-	-	7x8光标
		A7h	-	-	8x8光标
6	数据自动读写设置 (Data auto Read/Write)	B0h	-	-	设置自动写数据
		B1h	-	-	设置自动读数据
		B2h	-	-	自动复位
7	数据一次读写设置 (Data Read/Write)	C0h	Data	-	自动化数据处理增量和数据写
		C1h	-	-	自动化数据处理增量各数据读
		C2h	Data	-	自动化数据消减量各数据写
		C3h	-	-	自动化数据消耗量和数据读
		C4h	Data	-	自动化数据不变数据写
		C5h	-	-	自动化数据处理不变数据读
8	屏读(一字节)设置 (Screen Peek)	E0h	-	-	屏幕拷贝
9	屏拷贝(一行)设置 (Screen Copy)	E1h			屏幕拷贝
10	位操作 (Bit Set/Reset)	F0h	-	-	位复位
		F1h	-	-	位设置
		F0h	-	-	位0(低)

	F1h	-	-	位1
	F2h	-	-	位2
	F3h	-	-	位3
	F4h	-	-	位4
	F5h	-	-	位5
	F6h	-	-	位6
	F7h	-	-	位7(高)

2、指令的详细介绍

1) 地址指针设置(Registers Setting):

代码	HEX	功能	参数1 (D1)	参数2 (D2)
00100001	21H	光标地址设置	水平位置 (低7位有效)	垂直位置 (低5位有效)
00100010	22H	CGRAM偏置地址设置	偏置地址 (低5位有效)	00H
00100100	24H	显示地址设置	低字节	高字节

a) 光标地址设置21H (D1, D2)

控制器的光标控制是独立显示地址控制的，它专门有一个光标指针寄存器存放当前的光标地址，而且光标地址不会自动控制修改。

该指令设置或称修改了光标在显示屏的位置。光标的地址以二维坐标形式，以字符为单位设置。水平方向的位置由D1参数确定，取值范围在00H~4FH (1~80字符位)，表示显示屏左起第几个字符位（本款显示屏只用到16个字符位）；垂直方向由D2参数确定，取值范围在00H~1FH (1~32字符行)，表示显示屏上起第几个字符行（本款显示屏只用到8个字符行）。如下整理：

D1参数	X地址	00~4FH (1~80字符位)	用到16字符 X8=128点
D2参数	Y地址	00~1FH (1~32字符行)	用到8字符 X8=64点

b) CGRAM偏置地址设置22H (D1, D2)

控制器可以管理2K的CGRAM。在显示存储器内要划出2K的区域作CGRAM使用，只需要确认

16位地址的高5位（ad15~ad11）即可。CGRAM偏置地址寄存器就是用来存储这个地址的。用户可以通过将这个寄存器的内容与自定义字符代码值组合出显示存储器中该字符字模数组所在的首地址；

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
C4	C3	C2	C1	C0	D7	D6	D5	D4	D3	D2	D1	D0	R2	R1	R0
偏置地址					字符代码								行地址指针		

R0-R2从0至7指向将该字符8个字节的字模。

偏置地址的设置由参数D1的低5位值来实现，D2设置00H。例如：

指令 03H-00H-22H 设置字符代码为80H，那么该字符字模组在显示存储的首地址为1C00H。字模存放在1C00H-1C07H单元内。

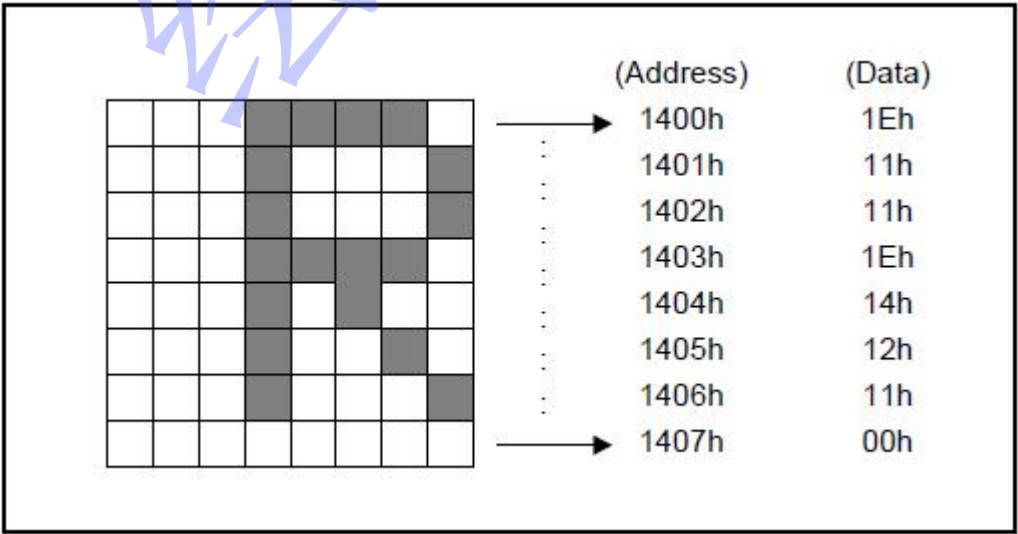
显示内存地址和偏移寄存器之间的关系

偏置数据 CGRAM地址HEX（开始到结束）

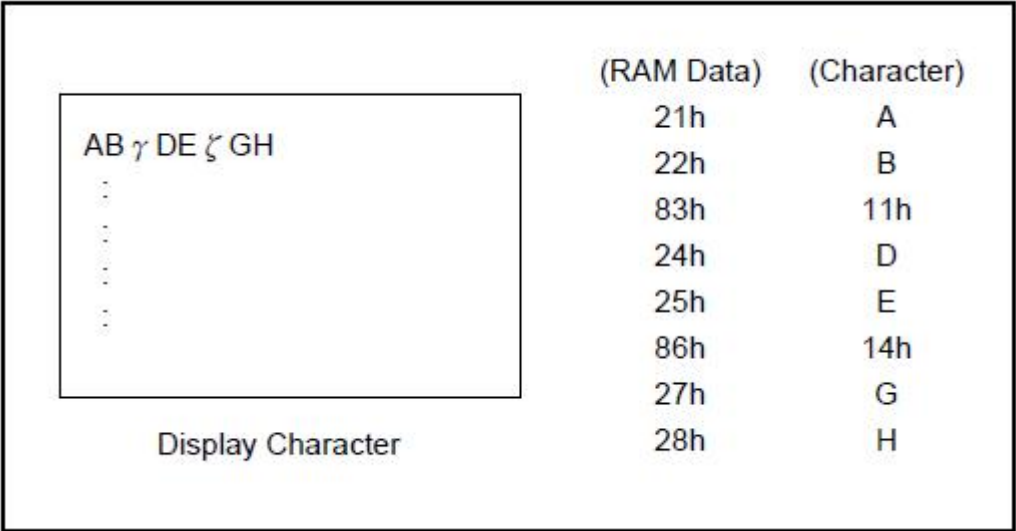
00000	0000-07FF	H
00001	0800-0FFF	H
00010	1000-17FF	H
00011	80H 1800-1FFF	H 1C00-1C07 H 字节地址
11100	E000-E7FF	H
11101	E800-EFFF	H
11110	F000-F7FF	H
11111	F800-FFFF	H

例子一)

偏置地址	02H
字符代码	80H
字符发生器内存起始地址	0001 0100 0000 0000
	1 4 0 0 H



例子二) 显示内存的数据和显示特征之间的关系



“γ”和“ζ”是由字符发生器内存显示

c) 显示地址设置24H (D1, D2)

该指令将计算机所要访问的显示存储器的地址写入显示器控制器的地址指针计数器中。该地址指针计数器为16位字长，需要两个字节。D1为低8位地址，D2为高8位地址

写显示地址代码流程



2) 显示区域设置 (Set Control Word):

代码	HEX	功能	参数1 (D1)	参数2 (D2)
0100000	40h	文本显示区首地址	低字节	高字节
01000001	41h	文本显示区宽度	字节数	00h
01000010	42h	图形显示区首地址	低字节	高字节
01000011	43h	图形显示区宽度	字节数	00h

a) 文本显示区首地址40h(d1,d2)

该指令设置了文本显示区在显示存储器中的起始地址。该地址对应显示屏左上角的第一个字符位(home)。定时间定间隔地修改这个地址将会产生显示画面平滑滚动。参数D1为该地址的低8位，D2为该地址的高8位。

b) 文本显示区宽度41h(d1,d2)

该指令规定了文本显示区中作为一行显示所占的单元(字节)数，该数据与文本显示首地址一起确定了显示单元与显示屏上各点像素的对应关系。比如设文本显示区首地址为SDA，文本显示区宽度为CR，N为字符总位数，则文本显示单元与显示屏上字符位的对应关系如图B01

SEG \ COM	1(字符位)	2(字符位)	3(字符位)	、 、 、 、	N(字符位)
1(字符行)	SAD	SAD+1	SAD+2	、 、 、 、	SAD+CR-1
2(字符行)	SAD+CR	SAD+CR+1	SAD+CR+2	、 、 、 、	SAD+2CR-1

当设置的显示区宽度超过了实际显示屏上显示所需的单元数，那么超出的部分不显示。但是要注意的是这里使用软件设置的显示区域宽度仅是控制器作为计算每行显示数据传输的起始单元地址的参数，实际传输的字节数已由硬件引脚设置完成。比如本款显示器，通常为了计算地址方便，设置文本显示区域宽度为20H(32字节)，但是控制器实际一行传输数据为40个字节，也就是说控制器将所设置的下一行单元的数据也作为这一行的数据传输了，只是由于屏点阵数小于所传输的数据量不显示而已。但控制器在传输下一行数据时要使用这个宽度参数计算起始单元的地址，如图B01

c) 图形显示区首地址42h(d1,d2)

该指令设置了图形显示区在显示存储器中起始地址。该地址对应显示屏左上角的第一个8点列像素(home)。一个水平8点像素作为一个像素组由一个字节表示。定时间定间隔地修改这个地址将会产生显示画面为平滑流动。参数D1为该地址的低8位，D2为该地址的高8位。

d) 图形显示区宽度43h(d1,d2)

该指令规定了在图形显示区中作为一行显示所占的单元(字节)数。该数据与图形显示首地址一起确定了显示单元与显示屏上各像素组(字节)的对应关系。比如设图形显示区首地址为SAD，图形显示区宽度为CR，N为像素点数，则图形显示单元与显示屏上像素组的对应关系，如图D01

SEG \ COM	1(字符位)	2(字符位)	3(字符位)	、 、 、 、	N(字符位)
1(点行)	SAD	SAD+1	SAD+2	、 、 、 、	SAD+CR-1
2(点行)	SAD+CR	SAD+CR+1	SAD+CR+2	、 、 、 、	SAD+2CR-1

当设置的显示区宽度超过了实际显示屏上显示所需的单元数，那么超出的部分不显示。但是要注意的是这里使用软件设置的显示区域宽度仅是控制器作为计算每行显示数据传输的起始单元地址的参数，实际传输的字节数已由硬件引脚设置完成，比如本款显示模块，通常为了计算方便，设置文本显示区域宽度为20H（32字节），但控制器实际一行传输数据为40个字节，也就是说控制器将所设置的下一行单元的数据也作为这一行的数据传输了，只是由于屏点阵数小于传输的数据量不显示而已。但是控制器在传输下一行数据时要使用这个宽度参数计算起始单元的地址，如图D01所示那样。

3) 显示方式设置 (Mode Set)：80h

无参数指令

7	6	5	4	3	2	1	0
1	0	0	0	CG	N2	N1	N0

CG位 字符发生器选择位

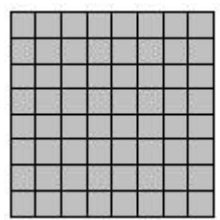
当CG=0时，启用内部字符发生器CGROM，该字符库有128种字符，其代码00H~7FH；同时可以建立128种8X8点阵的自定义字符发生器CGRAM，其字符代码规定在80H~FFH范围内；可以装32个汉字；

当CG=1时，禁止内部CGROM，字符显示完全取自定义字符发生器CGRAM，该字符库为2K字节容量，字符代码为00H~FFH。可以装64个汉字；

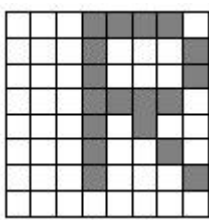
N2-N0位为显示方式设置位，它们的组合所产生的显示方式如下表

代码	HEX	功能说明	参数
10000000	80h	逻辑或：文本与图形以逻辑“或”的关系合成显示	无参数
10000001	81h	逻辑异域：文本与图形以逻辑“异或”的关系合成显示	
10000011	83h	逻辑与：文本与图形以逻辑“与”的关系合成显示	
10000100	84h	文本属性：文本显示特征以双字节表示	
10000XXX	F0H	内部CG ROM模式（00-7F ROM）（外部80-FF RAM可自定义32个汉字）	
10001XXX	F8h	外部CORAM（00-FF个8X8点阵）可自定义64个汉字	

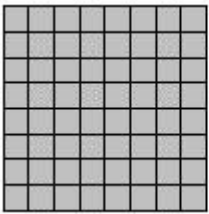
显示例子



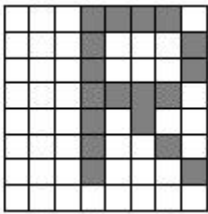
图形



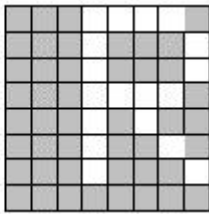
文本



“或”



“与”



“异或”

在设置了文本属性显示方式后，图形显示区将转换成文本属性区，用于存储字符的属性代码，其地址与显示屏上的对应关系与文本显示区相同。因此在显示屏上某位置上显示的字符是由双字节数据组成。第一字节为字符代码存储在文本显示区内，第二字节为属性代码存储在文本属性区内。这种表示方法与计算机的字符显示数据相似。在文本属性显示方式下，字符的属性代码由一字节的低4位组成；

7	6	5	4	3	2	1	0
x	x	x	x	D3	D2	D1	D0

其中 D3位是字符闪烁控制位。D3=0为不闪烁；D3=1为闪烁。

D2-D0组合功能如表

CG	N2	N1	N0	功能
0	0	0	0	正向显示
0	1	0	1	反向显示
0	0	1	1	禁止正向显示 (Inhibit Display)
0	1	0	0	禁止反向显示
1	0	0	0	正向显示 闪烁
1	1	0	1	反向显示 闪烁
1	0	1	1	禁止正向显示 闪烁 (Blink of Inhibit Display)
1	1	0	0	禁止反向显示 闪烁

4) 光标状态设置(display mode)：90h

无参数指令

7	6	5	4	3	2	1	0
1	0	0	1	N3	N2	N1	N0

该指令设置了当前的显示状态，有4个设置位，每一位都代表一种显示状态的设置，它们可以同时有效，也可以部分有效，也可以都无效。功能如下：

- N0 光标闪烁设置开关 N0=1为启用光标闪烁；N0=0为禁止光标闪烁。
- N1 光标显示设置开关 N1=1为启用光标显示；N1=0为禁止光标显示。
- N2 文本显示设置开关 N2=1为启用文本显示；N2=0为禁止文本显示。
- N3 图形显示设置开关 N3=1为启用图形显示；N3=0为禁止图形显示。

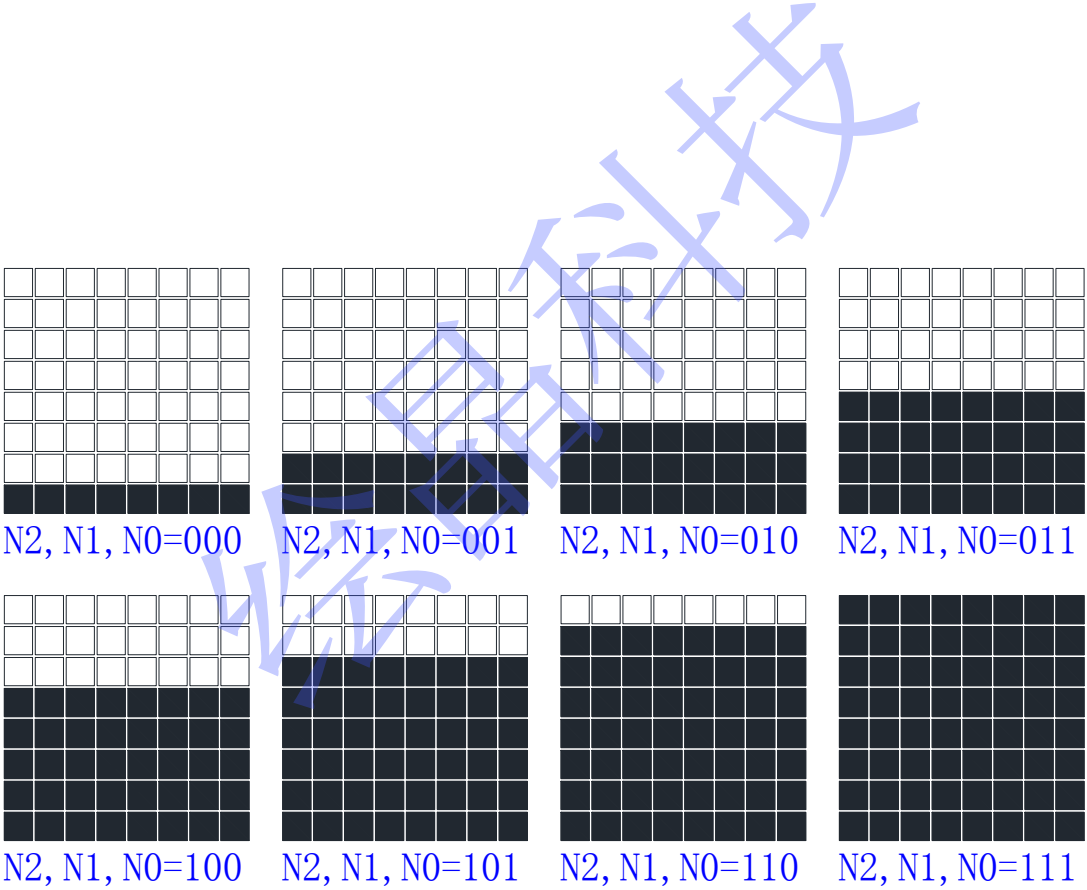
在文本显示与图形显示合成显示时，文本显示形状与图形显示开关应同时启用，在文本属性显示方式下，图形显示开关也应启用，只是特性不同。光标显示及光标闪烁功能的启用要在文本显示启用时进行，否则无效。

5) 光标形状设置(Cursor Pattern Select): A0h

无参数指令

7	6	5	4	3	2	1	0
1	0	1	0	0	N2	N1	N0

该指令设置光标的显示形状。三个设置位设置了光标点数。光标是以8点列xN行，行的取值由设置位N2，N1，N0组合完成。光标设置位的取值与光标显示形状的对应关系 如图光标大小设置7-2-5



光标大小设置7-2-5

6) 数据自动读写设置(Data auto Read/Write): B0h

7	6	5	4	3	2	1	0
1	0	1	1	0	0	N1	N0

使用该指令进入或退出数据的自动读或自动写入方式。在自动读或自动写方式中，计算机可以连续地将显示数据写入显示存储器中或从显示存储器中读取数据。在每次读或写的操作后，显示地址自动加一。进入自动读方式或自动写方式时，状态位将S2（自动读方式）或S3（自动写方式）代替S1和S0。在自动读或自动写方式完成时要输入退出自动读写方式指令。在自动读写方式中写入其它指令都是无效的。

该指令中有两位设置位N1，N0。这两位将产生三个子指令。

N1	N0	代码	功能
0	0	B0H	进入自动写方式
0	1	B1H	进入自动读方式
1	*	B2H/B3H	退出自动读写方式

自动读写方式的流程图7-2-6



自动读写操作流程7-2-6

7) 数据一次读写设置 (Data Read/Write): C0h

7	6	5	4	3	2	1	0
1	1	0	0	0	N2	N1	N0

该指令是一次读写数据操作指令。在每次读写数据操作后，显示地址都要根据指令代码的设置而修正；加一，减一或不变。该指令在写入数据时，所带的一

个参数就是所要写入的显示数据。当读数据操作时，该指令不带参数，直接写入指令代码，控制器在接收到该指令后将当前显示地址计数器所指的显示存储器单元的内容送入接口部的数据栈内，紧接着计算机的读数据操作将其读出。该指令有三个设置位N2，N1，N0（表），

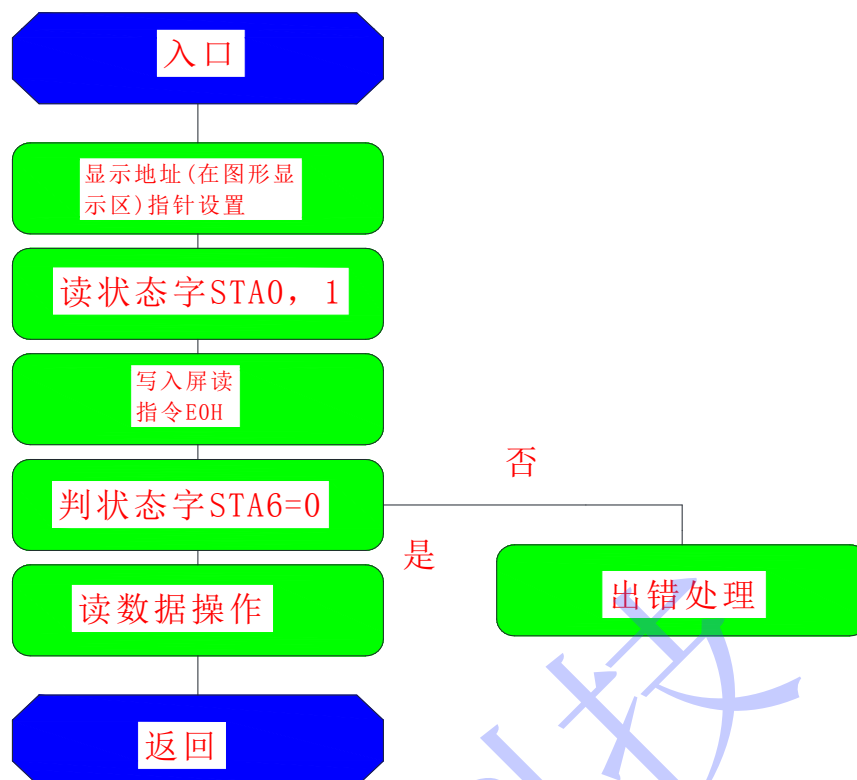
参数D1	N2	N1	N0	代码	功能
数据	0	0	0	C0H	数据写，地址加一
—	0	0	1	C1H	数据读，地址加一
数据	0	1	0	C2H	数据写，地址减一
—	0	1	1	C3H	数据读，地址减一
数据	1	0	0	C4H	数据定，地址不变
—	1	0	1	C5H	数据读，地址不变

8) 屏读（一字节）设置(screen peek): E0h

所谓屏读是指把显示屏上显示的内容取出来作为数据提供给计算机使用，这个内容为一个字节的当前显示数据，它有可能是图形显示数据，也可能是文本显示的某一个字符的某一行的字模数据，更多的是文本与图形合成的内容。屏读指令将使计算机能够直接获得显示屏上的数据，这是其他控制器所没有的功能。屏读指令要求当前显示地址指针指在图形显示区内，所以屏读指令只能在图形显示功能有将时才能使用。在屏读指令写入后要立即检查状态S6，判断该指令执行是否正确。如果执行正确。就可以读取数据。屏读指令的执行流程图7-2-8

该指令不能应用文本属性显示方式下。

由于屏读操作所行到的数据是由控制器驱动部数据合成后反馈产生的，所对应的单元地址与硬件引脚设置窗口长度有关，所以要求显示区域宽度的软件设置与硬件显示数据传输量设置端MD3，MD2所设置的显示窗口长度相同长度相同才能正确的计算出对应单元的地址，屏读出正确的数据来。



9) 屏拷贝（一行）设置(screen copy): E8h

所谓屏拷贝是指把显示屏上的某一行显示的内容取出来作为图形显示数据返写回图形显示区相应的显示单元内，这个内容为一行数个字节的当前显示数据，它有可能是图形显示数据，也可能是文本显示某一个字符上的某一行的字模数据，更多的是文本与图形合成显示的内容。屏拷贝功能将当前显示屏上的显示内容拷贝到图形显示区内作为计算机的处理使用。这是其它控制器所没有的功能。屏拷贝指令当前显示地址指针在图形显示区内，所以屏拷贝指令只能在图形显示功能有效时才能使用。在屏拷贝指令写入后要立即检查状态S6，判断该指令执行得是否正确。如果执行正确，就可以读取数据。屏拷贝指令流程图7-2-9

该指令不能应用在文本属性显示方式下，也不能应用在双屏结构的液晶显示器件的控制上。

由于屏拷贝操作所得到的数据是由控制器驱动部数据合成后反馈产生的，所对应的单元地址与硬件引脚设置窗口长度有关，所以要求显示区域宽度的软件设置与硬件显示数据传输量设置端MD3, MD2所设置的显示窗口长度相同才能正确的计算出单元的地址，屏拷贝出正确的数据来。



10) 位操作 (bit set reset): F0h

该指令可以对当前显示地址指针所指的显示单元中的数据的数据的任一位写“0”或“1”。操作位由N2, N1, N0确定。它们取值在0~7之间，对应着数据的D0~D7位。N3为写入的数据，是“1”表示该位将置“1”；是“0”表示该位将清“0”。该指令一次仅能操作一位。

7	6	5	4	3	2	1	0
1	1	1	1	N3	N2	N1	N0

表

代码	功能	参数
11110XXX	位复位	—
11111XXX	位设置	—
1111X000	位0 (低)	—
1111X001	位1	—
1111X010	位2	—
1111X011	位3	—
1111X100	位4	—
1111X101	位5	—

1111X110	位6	-
1111X111	位7（高）	-

11) 指令写入流程说明

a) 写入指令流程

控制器指令的运行时间有些是不确实的，这是因为有此指令的运行要受到当时控制部的状态影响。

在控制器指令中有的指令需要参数的补充，如地址指针的设置。控制器指令参数的输入是指令代码写入之前，如图7-2-11

如果在带有参数的指令代码写入之前，写入多于该指令参数的数据，那么控制器认为最近写入的数据为该指令的参数。写入流程图7-2-11

b) 读状态字（status read）

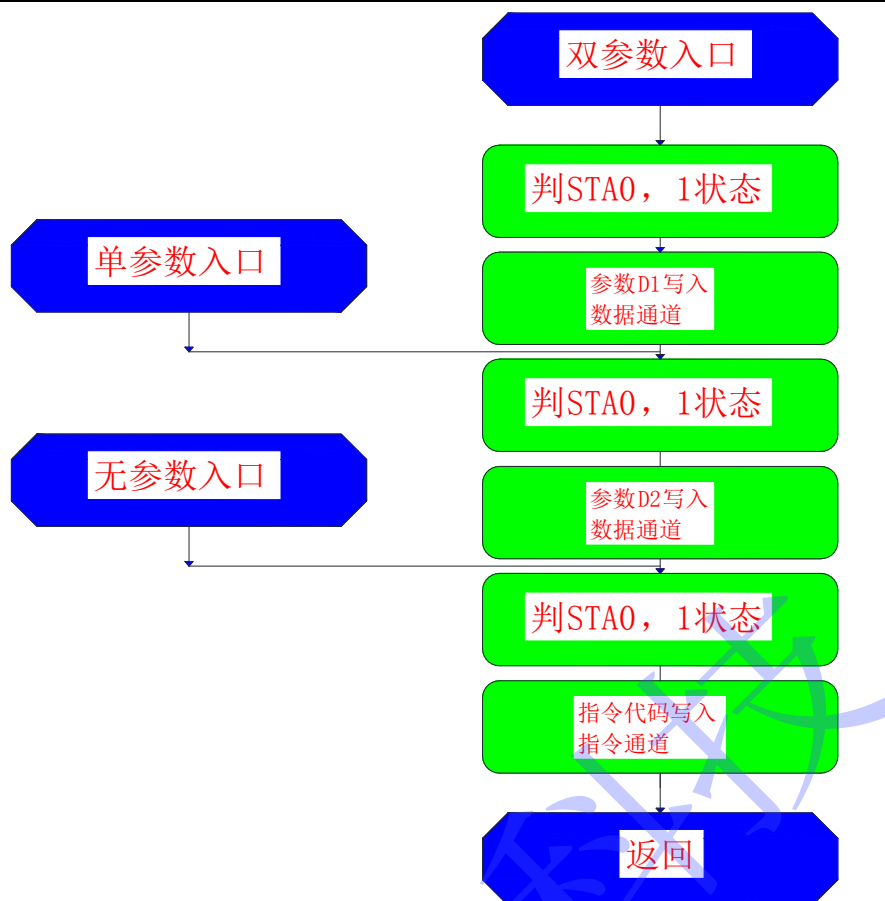
单片机要向控制器内写入指令/数据，都要要确认控制器是不是准备好，这就是控制器状态字在发挥作用。

控制器状态字由7位标志组成，它们是：

STA0	指令读写状态	1; 准备好	0; 忙
STA1	数据读写状态	1; 准备好	0; 忙
STA2	数据自动读状态	1; 准备好	0; 忙
STA3	数据自动写状态	1; 准备好	0; 忙
STA4	未用		
STA5	控制器运行检测可能性	1; 可能	0; 不能
STA6	屏读/屏拷贝出错状态	1; 出错	0; 正确
STA1	闪烁状态检测	1; 显示	0; 显示

这7个标志位各有各的应用应用场合，并非同时有效。在计算机写指令或一次读/写数据时，STA0和STA1要同时有效，即“准备好”状态；当计算机使用自动读/写功能时，STA2或者STA3将取代STA0各STA1作为忙标志位，此时计算机就要差别它是否有效；STA6标志是考察控制器屏读或屏拷贝指令执行情况的标志位。STA5和STA7表示控制器内部运行状态，在控制器应用上不会使用它们。

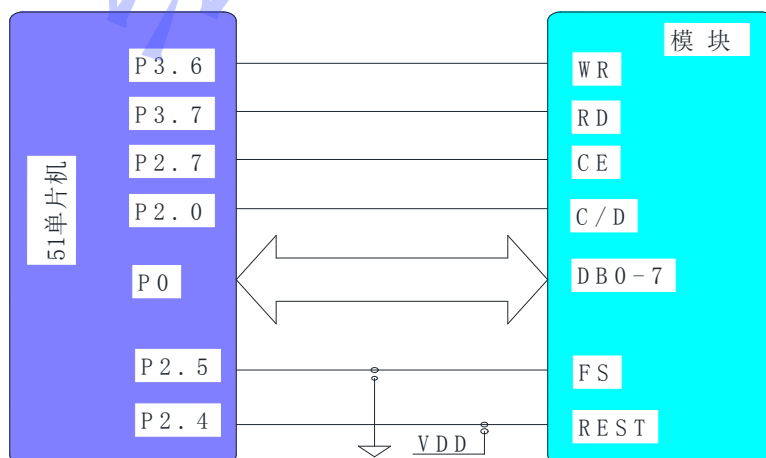
对控制器的软件操作每一次之前都要进行判“忙”。只有仅有在不“忙”的状态下计算机对控制器的操作才有效。



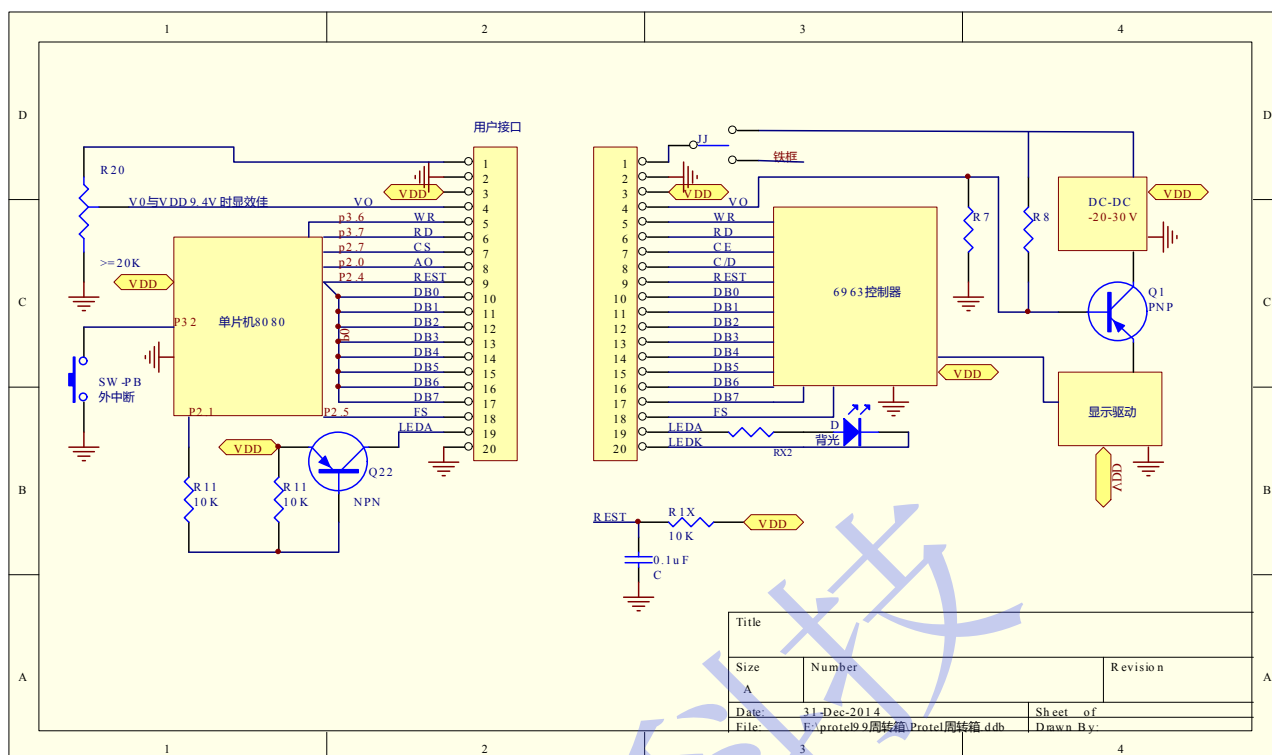
7-2-11

八、单片机接线图

i. 例程中单片机IO口连接图



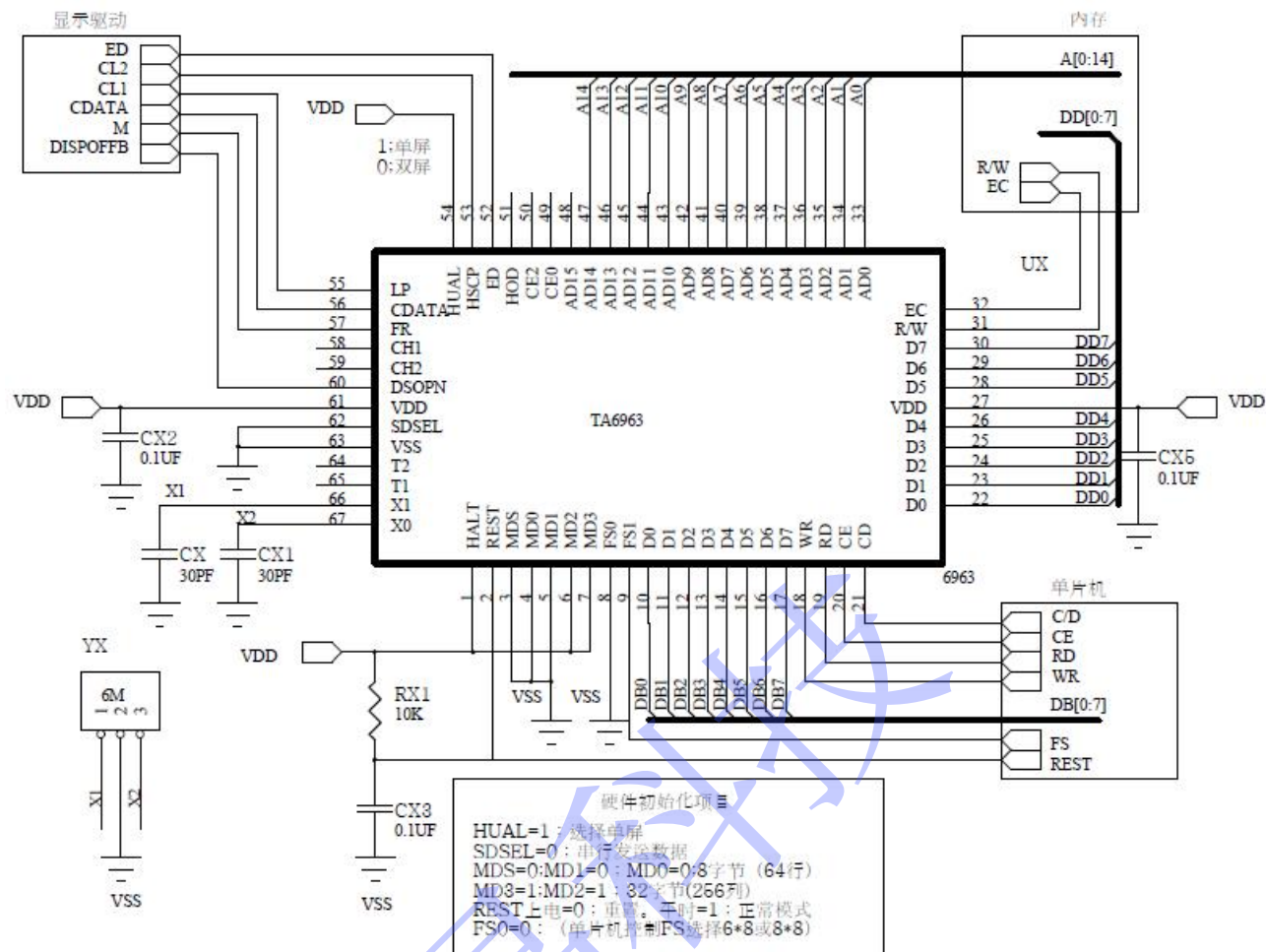
ii. 用户开发环境接线原理图参数



九、模块控制器初始化

a) 硬件初始化

硬件初始化用户不能改动，这里我们公开设置，让用户对我们的控制器有更全面的了解。



驱动方式设置（单屏，串行传输）

HUAL	SDSEL	上半屏数据传输线	下半屏数据传输线
1	1	HOD, ED	-
1	0	ED	-
0	1	HOD, ED	LOD, ED
0	0	ED	ED

显示窗口长度设置（宽度320列/40字符）

MD3	MD2	像素点数	字符数（字节数）
1	1	256	32
1	0	320	40
0	1	512	64
0	0	640	80

显示窗口宽度设置（64行）

MDS	MD1	MD0	像素点数（单/双屏）	占空比系统（N）
0	1	1	16/32	16
0	1	0	32/64	32
0	0	1	48/96	48
0	0	0	64/128	64
1	1	1	80/160	80
1	1	0	96/192	96
1	0	1	112/224	112
1	0	0	128/256	128

显示字符的字体设置

FS1	1	1	0	0
FS0	1	0	1	0
字体	5x8	6x8	7x8	8x8

b) 软件初始化

/*上电初始化*/
//8*8字符字体

REST=1;

REST=0;

REST=1;

FS=0; //8*8字符字体

CE=0;

/*设置文本显示首地址*/

data1=0x00;

data2=0x00;

command=0x40;

WRCCommandTow();

/*设置文本显示区宽度*/

data1=0x10; //16字节16*8=128像素

data2=0x00;

command=0x41;

WRCCommandTow();

/*设置图形显示区首地址*/

data1=0x00;

data2=0x02; /*首地址0x0200*/

```
command=0x42;
WRCommandTow();

/*设置图形显示区宽度*/
data1=0x10;      //16字节16*8=128像素
data2=0x00;
command=0x43;
WRCommandTow();

/*设置显示方式*/
command=0x80;      /*启用内部字符发生器，显示方式是文本与图形逻辑或*/
WRCommandNONE();

/*设置显示状态*/
command=0x9C;      /*禁用光标闪烁、光标显示、启用文本显示、图形显示*/
WRCommandNONE();
```

十、完整的程序例程

//HJ12864F(6963)

```
#include<reg52.h>
#include <intrins.h>
sbit WWR=P3^6;
sbit RRD=P3^7;
sbit CE=P2^7;//8080时序时,这个脚可以直接接VSS
sbit CD=P2^0;
sbit REST=P2^4;  //这个脚可以直接接VCC
sbit FS=P2^5;    //这个脚可以直接接VSS
sbit stop=P3^2;
```

```
typedef unsigned int uint;
typedef unsigned char uchar;
unsigned char command;
unsigned int i,data1,temp,data2,j,k;
uchar code  num_8x16[11][16];
uchar code  num_16x24_2[];
uchar code  num_16x24_4[];
uchar code  num_16x24_6[];
uchar code  num_16x24_x[];
uchar code  num_16x24_1[];
uchar code  num_16x24_8[];
```

```

uchar code hui_24x24[];
uchar code jing_24x24[];
uchar code ke_24x24[];
uchar code ji_24x24[];
uchar code CGRAM_128[];
uchar code CCTAB3[];

void WaitNms(unsigned int x)//0.5ms
{
    unsigned char j;
    while(x--)
        {for(j=0; j<60; j++)
            {;}}
}

void ST01Read(void) {
    P0=0x00;
    while(1) {
        CD=1 ;
        RRD=0;
        temp=P0;                //将PTB口数据送给变量temp
        RRD=1 ;
        if ((temp&0x03)==0x03) break; /*判断指令、数据读写状态是否准备好*/
    }
}

/*读状态位3 */
void ST3Read(void) {
    P0=0x00;
    while(1) {
        CD=1;
        RRD=0 ;
        temp=P0;
        RRD=1 ;
        if ((temp&8)!=0) {break;}    //判断数据自动写状态是否准备好
    }
}

/*写数据*/
void WRData(uchar Data) {
    CE=0;

    CD=0;
    WWR=0;
    P0=Data;
    WWR=1;
}

```



```
}  
  
/*写命令*/  
void WRCommand(uchar command) {  
    CE=0;  
  
    CD=1;  
    WWR=0 ;  
    P0=command;    //将命令送PTB口  
    WWR=1;  
}  
  
/*读数据*/  
/*  
void RDData(void) {  
    ST01Read();    //读状态位0和1  
    CE=0;  
    CD=0;  
    RRD=0;  
    data2=P0;    // 读取数据  
    RRD=1;  
}  
*/  
  
/*无参数指令*/  
void WRCommandNONE(void) {  
    ST01Read();    //读状态位0和1  
    WRCommand(command); /*写命令*/  
}  
  
/*单参数指令*/  
void WRCommandOne(void) {  
    ST01Read();  
    WRData(data2);    /*写数据*/  
    ST01Read();  
    WRCommand(command); /*写命令*/  
}  
  
/*双参数指令*/  
void WRCommandTwo(void) {  
    ST01Read();  
    WRData(data1);    /*写数据*/  
    ST01Read();  
    WRData(data2);    /*写数据*/  
    ST01Read();  
    WRCommand(command); /*写命令*/  
}
```

```
/*初始化T6963 LCD 显示.*/
```

```
void Init LCD Graphic(void)
```

{

REST=1;

REST=0;

REST=1;

```
FS=0; //8*8字符字体
```

$$CE=0;$$

```
/*设置文本显示首地址*/
```

```
data1=0x00;
```

```
data2=0x00:
```

```
command=0x40;
```

```
WRCommandTow();
```

```
/*设置文本显示区宽度*/
```

```
data1=0x10; //16字节16*8=128像素
```

data2=0x00:

```
command=0x41;
```

```
WRCommandTow();
```

```
/*设置图形显示区首地址*/
```

data1=0x00:

```
data2=0x02; /*首地址0x0200*/
```

```
command=0x42;
```

WRCommandTow();

```
/*设置图形显示区宽度*/
```

```
data1=0x10; //16字节16*8=128像素
```

```
data2=0x00;
```

```
command=0x43;
```

WRCommandTow();

```
/*设置显示方式*/
```

```
command=0x80; /*启用内部字符发生器，显示方式是文本与图形逻辑或*/
```

```
WRCommandNONE();
```

```
/*设置显示状态*/
```

```
command=0x9C; /*禁用光标闪烁、光标显示、启用文本显示、图形显示*/
```

```
WRCommandNONE();
```

}

```
/*清显示RAM,文本与图形一起清*/
```

[illegible]

```
{
    data1=0x03;
    data2=0x00;
    command=0x22;
    WRCommandTow(); //自定义字符偏置地址设置
    data1=0x00;
    data2=0x1C;
    command=0x24;
    WRCommandTow(); //存字符的地址:0X1C00开始
    command=0xb0;
    WRCommandNONE(); //开自动写
    for(i=0; i<0x80; i++)
    {
        ST3Read();
        WRData(0x00); //输入空数据
    }
    command=0xb2;
    WRCommandNONE(); //关自动写
}
```

//存入自定义汉字字符

void StoreCGRAM(void)

```
{
    data1=0x03;
    data2=0x00;
    command=0x22;
    WRCommandTow(); //自定义字符偏置地址设置
    data1=0x00;
    data2=0x1C;
    command=0x24;
    WRCommandTow(); //存字符的地址:0X1C00开始
    for(k=0; k<14; k++)
    {
        for(i=0; i<8; i++)
        {
            data2=CCTAB3[2*i+32*k];
            command=0xc0;
            WRCommandOne();
        }
        for(i=0; i<8; i++)
        {
            data2=CCTAB3[2*i+1+32*k];
            command=0xc0;
            WRCommandOne();
        }
    }
}
```

[illegible]

```

command=0xc0;
WRCCommandOne();
//显示汉字左下角
temp=0x0000+Y*16+X+16;          //光标下移一格,则要加16字节间距  <<<<<<<<<-----
addL=temp&0x00ff;                //取TEMP的低八位,
addH=((temp&0xff00)>>8);          //取TEMP的高八位.
data1=addL;
data2=addH;
command=0x24;
WRCCommandTow();
data2=0x82+4*N;
command=0xc0;
WRCCommandOne();
//显示汉字右下角
temp=0x0000+Y*16+X+1+16;          //<<<<<<<<<<-----
addL=temp&0x00ff;                //取TEMP的低八位,
addH=((temp&0xff00)>>8);          //取TEMP的高八位.
data1=addL;
data2=addH;
command=0x24;
WRCCommandTow();
data2=0x83+4*N;
command=0xc0;
WRCCommandOne();
}

//在全屏填满"深圳绘晶科技有限公司液晶模块"共三行
void ShowChineseAll(void)
{ ClearCGRAM();
  StoreCGRAM();
  for(j=0; j<1; j++)
  {k=0;
   for(i=0; i<14; i++)
    if(i<8)
    {
      DisplayOneHZCGRAM(2*i+0,2*j+0,k);
      ++k;
    }
    else
    {
      DisplayOneHZCGRAM(2*i+0,2*j+2,k);
      ++k;
    }
  }
}

```

```
}

//*****
//
//          以下做横竖点扫描
//存入自定义横竖点数据
void wr_cgram(uchar n,uchar *text)
{
    uchar ad; ad=n*8;
    data1=0x03;
    data2=0x00;
    command=0x22;
    WRCommandTow(); //自定义字符偏置地址设置
    data1=0x00;
    data2=0x1C;
    command=0x24;
    WRCommandTow(); //存字符的地址:0X1C00开始
    command=0xb0;
    WRCommandNONE(); //开自动写
    //横一
    for(i=0;i<ad;i++)
    {
        WRData(text[i]); //输入数据
    }
    command=0xb2;
    WRCommandNONE(); //关自动写
}

//把任意一个字符填满全屏
void FillOneCGall(uint N)
{
    data1=0x00;
    data2=0x00;
    command=0x24;
    WRCommandTow(); //显示位置
    command=0xb0;
    WRCommandNONE(); //开自动写
    for(i=0;i<480;i++) //          30*16=480 一屏          <<<<<<<<<-----
    {
        // ST3Read();
        WRData(N); //输入字符数据
    }
    command=0xb2;
    WRCommandNONE(); //关自动写
}
```



```
// 灵活的矩形绘制函数X1, Y1为起点, X2为行程, Y2为列程,
void Wr_reg(uchar x1,uchar y1,uchar x2,uchar y2,uchar d1)
{
    uint addL;
    uint addH;
    uchar j;
    for(i=0;i<y2;i++)
    {
        temp=0x0200+y1*16+x1+16*i;
        addL=temp&0x00ff;          //取TEMP的低八位,
        addH=((temp&0xff00)>>8);    //取TEMP的高八位.
        data1=addL;
        data2=addH;
        command=0x24;              //显示位置
        WRCommandTow();
        command=0xb0;
        WRCommandNONE();          //开自动写
        for(j=0;j<x2;j++)
        {
            WRData(d1);            //输入全黑数据
        }
        command=0xb2;
        WRCommandNONE();          //开自动写
    }
}
```

```
void display_string_8x8(bit fb,uint x1,uint y1,uchar *text)
{
    uint addL;
    uint addH;
    uchar i=0,j;
    temp=0x0000+y1*16+x1+16*i;
    addL=temp&0x00ff;          //取TEMP的低八位,
    addH=((temp&0xff00)>>8);    //取TEMP的高八位.
    data1=addL;
    data2=addH;
    command=0x24;              //显示位置
    WRCommandTow();
    command=0xb0;
    WRCommandNONE();          //开自动
    while(text[i]>0x00)
```

```
{
    if((text[i]>=0x20)&&(text[i]<=0x7e))
    {
        j=text[i]-0x20;
        if(fb==0) WRData(j);        //输入全黑数据
        else WRData(~(j));
        i++;
    }
    else
        i++;
}
command=0xb2;
WRCommandNONE();    //开自动写
}
```

void ds_txt(bit fb,uchar X,uchar Y,uchar tx,uchar ty,uchar *dp)//X为列坐标,以字节为单位;Y为行坐标,以点为单位.

```
{
    uint addL;
    uint addH;
    uchar j;
    for(i=0;i<ty;i++)
    {
        temp=0x0200+Y*16+X+16*i;
        addL=temp&0x00ff;        //取TEMP的低八位,
        addH=((temp&0xff00)>>8);    //取TEMP的高八位.
        data1=addL;
        data2=addH;
        command=0x24;        //显示位置
        WRCommandTow();
        command=0xb0;
        WRCommandNONE();    //开自动
        for(j=0;j<tx;j++)
        {
            if(fb==0) WRData(*dp);
            else WRData(~(*dp));
            dp++;
        }
        command=0xb2;
        WRCommandNONE();    //开自动写
    }
}
```

```

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
}

//~~~~~ ( ) 程序开始
void main(void)
{
    ini_int1();//开中断
    while(1)
    {
        Init_LCD_Graphic(); //初始化屏幕
        ClrGraphic(); //清屏
        ShowAllCGRAM(); //显示内部固定RAM字符库
        WaitNms(1000); //延时片刻
        ClrGraphic(); //清屏
        //~~~~~ ( ) 显示6963自带字符128个

        Wr_reg(0,0,16,2,0xff);//画上框HJ12864
        Wr_reg(0,62,16,2,0xff);//画下框
        Wr_reg(0,2,1,60,0xC0);//左框
        Wr_reg(15,2,1,60,0x03);//右框

        ds_txt(0,2,7,3,24,hui_24x24);//（正显，行，列，字宽，字高，数据）
        ds_txt(0,5,7,3,24,jing_24x24);
        ds_txt(0,8,7,3,24,ke_24x24);
        ds_txt(0,11,7,3,24,ji_24x24);

        ds_txt(0,2,32,2,24,num_16x24_1);
        ds_txt(0,4,32,2,24,num_16x24_2);
        ds_txt(0,6,32,2,24,num_16x24_8);
        ds_txt(0,8,32,2,24,num_16x24_x);
        ds_txt(0,10,32,2,24,num_16x24_6);
        ds_txt(0,12,32,2,24,num_16x24_4);

        WaitNms(1000); //延时片刻
        ClrGraphic(); //清屏
    }
}

```

//~~~~~ () 绘图式汉字、边框

```
ClearCGRAM();
wr_cgram(7,CGRAM_128);
FillOneCGall(0x80);
WaitNms(1000); //延时片刻HJ12864
FillOneCGall(0x81);
WaitNms(1000); //延时片刻
FillOneCGall(0x82);
WaitNms(1000); //延时片刻
FillOneCGall(0x83);
WaitNms(1000); //延时片刻
FillOneCGall(0x84);
WaitNms(1000); //延时片刻
FillOneCGall(0x85);
WaitNms(1000); //延时片刻
FillOneCGall(0x86);
WaitNms(1000); //延时片刻
ClrGraphic(); //清屏
```

//~~~~~ () 扫横七竖八测试版

```
display_string_8x8(0,0,0,"TEL: 0755-23146001");
display_string_8x8(0,0,2,"FAX: 0755-23146002");
display_string_8x8(0,0,4,"www.huijinglcm.com");
display_string_8x8(0,0,6,"E-mail: huijinglcm@sina.com ");
WaitNms(1000); //延时片刻
ClrGraphic(); //清屏
```

//~~~~~ () 应用字符串演示

```
ShowChineseAll(); //显示中文地址
WaitNms(1000); //延时片刻
ClrGraphic(); //清屏
```

//~~~~~ () 自编字符与显示汉字

```
Wr_reg(0,0,16,64,0xff); //全黑
ds_txt(1,2,7,3,24,hui_24x24); // (反白, 行, 列, 字宽, 字高, 数据)
ds_txt(1,5,7,3,24,jing_24x24);
ds_txt(1,8,7,3,24,ke_24x24);
ds_txt(1,11,7,3,24,ji_24x24);

ds_txt(1,2,32,2,24,num_16x24_1);
```

```
ds_txt(1,4,32,2,24,num_16x24_2);
ds_txt(1,6,32,2,24,num_16x24_8);
ds_txt(1,8,32,2,24,num_16x24_x);
ds_txt(1,10,32,2,24,num_16x24_2);
ds_txt(1,12,32,2,24,num_16x24_4);
WaitNms(1000);          //延时片刻
ClrGraphic();           //清屏
//~~~~~ ( ) 绘图整屏反白效果
```

```
}
}
```

```
uchar code num_16x24_2[]={
/*-- 文字: 2 --*/
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=12x24 --*/
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x80,0x39,0xC0,0x70,0xE0,
0x70,0xE0,0x70,0xE0,0x00,0xE0,0x01,0xC0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,
0x18,0x60,0x30,0x60,0x60,0xE0,0xFF,0xE0,0xFF,0xE0,0x00,0x00,0x00,0x00,0x00,
};
```

```
uchar code num_16x24_4[]={
/*-- 文字: 4 --*/
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=12x24 --*/
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x03,0x80,0x03,0x80,0x07,0x80,
0x0F,0x80,0x0F,0x80,0x1B,0x80,0x3B,0x80,0x33,0x80,0x73,0x80,0x63,0x80,0xFF,0xF0,
0x03,0x80,0x03,0x80,0x03,0x80,0x03,0xC0,0x0F,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,
};
```

```
uchar code num_16x24_6[]={
/*-- 文字: 6 --*/
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=12x24 --*/
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xC0,0x1D,0xC0,0x39,0xC0,
0x70,0x00,0x70,0x00,0x70,0x00,0xFF,0x80,0xF9,0xC0,0xF0,0xE0,0xF0,0xE0,0xF0,0xE0,
0x70,0xE0,0x70,0xE0,0x70,0xE0,0x39,0xC0,0x1F,0x80,0x00,0x00,0x00,0x00,0x00,0x00,
};
```

```
uchar code num_16x24_x[]={
/*-- 文字: x --*/
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=12x24 --*/
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0xFF,0xE0,0x39,0xC0,0x39,0x80,0x1F,0x00,0x0F,0x00,0x07,0x00,
```

```
0x0F,0x00,0x1F,0x80,0x19,0xC0,0x31,0xC0,0xFB,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,  
};
```

```
uchar code num_16x24_1[]={  
/*-- 文字: 1 --*/  
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=12x24 --*/  
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/  
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x07,0x00,0x3F,0x00,  
0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,  
0x07,0x00,0x07,0x00,0x07,0x00,0x0F,0x00,0x3F,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,  
};
```

```
uchar code num_16x24_8[]={  
/*-- 文字: 8 --*/  
/*-- 仿宋_GB231218; 此字体下对应的点阵为: 宽x高=12x24 --*/  
/*-- 宽度不是8的倍数, 现调整为: 宽度x高度=16x24 --*/  
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x80,0x39,0xC0,0x70,0xE0,  
0x70,0xE0,0x70,0xE0,0x78,0xE0,0x3D,0xC0,0x1F,0x80,0x3F,0xC0,0x71,0xE0,0x60,0xE0,  
0xE0,0xE0,0xE0,0xE0,0x70,0xE0,0x79,0xC0,0x1F,0x80,0x00,0x00,0x00,0x00,0x00,0x00,  
};
```

```
uchar code hui_24x24[]={  
/*-- 文字: 绘 --*/  
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=24x24 --*/  
0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0x07,0x80,0x0F,0x07,0x80,0x0E,0x07,0x80,0x1C,  
0x0F,0xC0,0x19,0xCE,0xE0,0x39,0xFC,0x70,0x33,0xB8,0x38,0x6F,0xB0,0x1E,0x7F,0x60,  
0xFF,0x06,0xDF,0xF0,0x0D,0x80,0x00,0x1C,0x00,0x00,0x39,0xC0,0x1C,0x7F,0xFF,0xF6,  
0x30,0x07,0x00,0x00,0x0E,0xC0,0x01,0xCC,0xE0,0x1F,0x1C,0x70,0x78,0x38,0x38,0x00,  
0x7F,0xFC,0x00,0x70,0x18,0x00,0x00,0x00,  
};
```

```
uchar code jing_24x24[]={  
/*-- 文字: 晶 --*/  
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=24x24 --*/  
0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0xE0,0x03,0xFF,0xF0,0x03,0x80,0xE0,0x03,  
0x80,0xE0,0x03,0xFF,0xE0,0x03,0x80,0xE0,0x03,0x80,0xE0,0x03,0xFF,0xE0,0x03,0x80,  
0xC0,0x00,0x00,0x00,0x38,0xE6,0x1C,0x3F,0xF7,0xFE,0x38,0xE6,0x1C,0x38,0xE6,0x1C,  
0x38,0xE6,0x1C,0x3F,0xE7,0xFC,0x38,0xE6,0x1C,0x38,0xE6,0x1C,0x38,0xE6,0x1C,0x3F,  
0xE7,0xFC,0x38,0xC6,0x18,0x00,0x00,0x00,  
};
```

```
uchar code ke_24x24[]={  
/*-- 文字: 科 --*/
```

```
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=24x24 --*/
0x00,0x00,0x00,0x00,0x00,0x60,0x00,0xE0,0x78,0x07,0xF0,0x70,0x7F,0x00,0x70,0x03,
0x0E,0x70,0x03,0x07,0x70,0x03,0x03,0x70,0x03,0x70,0x70,0x7F,0xF0,0x70,0x07,0x0C,
0x70,0x07,0x0E,0x70,0x0F,0xE7,0x70,0x0F,0xE7,0x7E,0x1F,0x60,0x7F,0x3B,0x01,0xF8,
0x33,0x7F,0x70,0x63,0x60,0x70,0xE3,0x00,0x70,0x07,0x00,0x70,0x07,0x00,0x70,0x07,
0x00,0x70,0x07,0x00,0x70,0x00,0x00,0x60,
};
```

```
uchar code ji_24x24[]={
/*-- 文字: 技 --*/
/*-- 宋体18; 此字体下对应的点阵为: 宽x高=24x24 --*/
0x00,0x00,0x00,0x04,0x03,0x00,0x07,0x03,0xC0,0x07,0x03,0x80,0x07,0x03,0x80,0x07,
0x03,0x80,0x07,0xE3,0x9E,0xFF,0xFF,0xFE,0x07,0x03,0x80,0x07,0x03,0x80,0x07,0x03,
0x98,0x07,0xFF,0xFC,0x0F,0x1C,0x3C,0x7F,0x0C,0x30,0x77,0x0C,0x70,0x07,0x0E,0x70,
0x07,0x06,0xE0,0x07,0x07,0xC0,0x07,0x03,0x80,0x07,0x07,0xC0,0x07,0x0E,0xF0,0x7E,
0x38,0x7F,0x0F,0xE0,0x1C,0x00,0x00,0x00,
};
```

```
uchar code num_8x16[11][16]={
/*-- 文字: 0 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,
/*-- 文字: 1 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x10,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,
/*-- 文字: 2 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x04,0x04,0x08,0x10,0x20,0x42,0x7E,0x00,0x00,
/*-- 文字: 3 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x3C,0x42,0x42,0x04,0x18,0x04,0x02,0x02,0x42,0x44,0x38,0x00,0x00,
/*-- 文字: 4 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x04,0x0C,0x14,0x24,0x24,0x44,0x44,0x7E,0x04,0x04,0x1E,0x00,0x00,
/*-- 文字: 5 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x7E,0x40,0x40,0x40,0x58,0x64,0x02,0x02,0x42,0x44,0x38,0x00,0x00,
/*-- 文字: 6 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x1C,0x24,0x40,0x40,0x58,0x64,0x42,0x42,0x42,0x24,0x18,0x00,0x00,
/*-- 文字: 7 --*/
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/
0x00,0x00,0x00,0x7E,0x44,0x44,0x08,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,
/*-- 文字: 8 --*/
```

```
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/  
0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x24,0x18,0x24,0x42,0x42,0x42,0x3C,0x00,0x00,  
/*-- 文字: 9 --*/  
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/  
0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x26,0x1A,0x02,0x02,0x24,0x38,0x00,0x00,  
/*-- 文字: : --*/  
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=8x16 --*/  
0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,  
};
```

```
uchar code CCTAB3[]=  
{//横向取模字节不倒序  
/*-- 文字: 深 --//  
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=16x16 --//  
0x40,0x00,0x27,0xFE,0x24,0x04,0x01,0x20,0x81,0x18,0x56,0x08,0x50,0x40,0x10,0x40,  
0x27,0xFC,0x20,0x40,0xC0,0xE0,0x41,0x58,0x46,0x4E,0x58,0x44,0x40,0x40,0x40,0x40,  
/*-- 文字: 圳 --//  
/*-- 宋体12; 此字体下对应的点阵为: 宽x高=16x16 --//  
0x00,0x00,0x12,0x04,0x12,0x44,0x12,0x44,0xFE,0x44,0x12,0x44,0x12,0x44,0x12,0x44,  
0x12,0x44,0x12,0x44,0x3A,0x44,0xC4,0x44,0x04,0x44,0x08,0x04,0x10,0x04,0x00,0x00,  
/*-- 文字: 宝 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x02,0x00,0x01,0x00,0x7F,0xFE,0x40,0x02,0x80,0x04,0x00,0x00,0x7F,0xFC,0x01,0x00,  
0x01,0x00,0x01,0x00,0x3F,0xF8,0x01,0x00,0x01,0x20,0x01,0x10,0xFF,0xFE,0x00,0x00,  
/*-- 文字: 安 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x02,0x00,0x01,0x00,0x3F,0xFC,0x20,0x04,0x42,0x08,0x02,0x00,0x02,0x00,0xFF,0xFE,  
0x04,0x20,0x08,0x20,0x18,0x40,0x06,0x40,0x01,0x80,0x02,0x60,0x0C,0x10,0x70,0x08,  
/*-- 文字: 石 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x00,0x00,0xFF,0xFE,0x02,0x00,0x02,0x00,0x04,0x00,0x04,0x00,0x08,0x00,0x0F,0xF8,  
0x18,0x08,0x28,0x08,0x48,0x08,0x88,0x08,0x08,0x08,0x08,0x08,0x0F,0xF8,0x08,0x08,  
/*-- 文字: 岩 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x01,0x00,0x21,0x08,0x21,0x08,0x21,0x08,0x3F,0xF8,0x00,0x00,0x00,0x00,0x7F,0xFC,  
0x04,0x00,0x04,0x00,0x0F,0xF8,0x18,0x08,0x28,0x08,0xC8,0x08,0x0F,0xF8,0x08,0x08,  
/*-- 文字: 镇 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x20,0x20,0x20,0x20,0x3B,0xFE,0x40,0x20,0x41,0xFC,0x79,0x04,0xA1,0xFC,0x21,0x04,  
0xF9,0xFC,0x21,0x04,0x21,0xFC,0x29,0x04,0x37,0xFE,0x20,0x88,0x01,0x04,0x02,0x02,  
/*-- 文字: 麻 --*/  
/*-- 仿宋_GB231212; 此字体下对应的点阵为: 宽x高=16x16 --*/  
0x01,0x00,0x00,0x80,0x3F,0xFE,0x22,0x10,0x22,0x10,0x22,0x10,0x3F,0xBE,0x22,0x10,  
0x26,0x10,0x27,0x38,0x2A,0xB4,0x2A,0x54,0x52,0x92,0x42,0x10,0x82,0x10,0x02,0x10,
```



```
/*-- 文字： 布 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x02,0x00,0x02,0x00,0x02,0x00,0xFF,0xFE,0x04,0x00,0x09,0x00,0x11,0x00,0x3F,0xF8,
0x51,0x08,0x91,0x08,0x11,0x08,0x11,0x08,0x11,0x28,0x11,0x10,0x01,0x00,0x01,0x00,
/*-- 文字： 第 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x20,0x40,0x3F,0x7E,0x48,0x90,0x85,0x08,0x3F,0xF8,0x01,0x08,0x01,0x08,0x3F,0xF8,
0x21,0x00,0x21,0x00,0x3F,0xFC,0x03,0x04,0x05,0x04,0x19,0x28,0xE1,0x10,0x01,0x00,
/*-- 文字： 二 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,
/*-- 文字： 工 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x00,0x00,0x00,0x00,0x7F,0xFC,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,
0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0xFF,0xFE,0x00,0x00,0x00,0x00,
/*-- 文字： 业 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x04,0x40,0x04,0x40,0x04,0x40,0x04,0x40,0x44,0x44,0x24,0x44,0x24,0x48,0x14,0x48,
0x14,0x50,0x14,0x60,0x04,0x40,0x04,0x40,0x04,0x40,0x04,0x40,0xFF,0xFE,0x00,0x00,
/*-- 文字： 区 --*/
/*-- 仿宋_GB231212; 此字体下对应的点阵为：宽x高=16x16 --*/
0x00,0x00,0x7F,0xFC,0x40,0x00,0x40,0x10,0x44,0x10,0x42,0x20,0x41,0x40,0x40,0x80,
0x41,0x40,0x42,0x20,0x44,0x10,0x48,0x10,0x40,0x00,0x40,0x00,0x7F,0xFE,0x00,0x00,
};
uchar code CGRAM_128[]={
0xff,0x00,0xff,0x00,0xff,0x00,0xff,0x00,//横
0x00,0xff,0x00,0xff,0x00,0xff,0x00,0xff,
0xaa,0xaa,0xaa,0xaa,0xaa,0xaa,0xaa,0xaa,//竖
0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,
0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,//点
0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,//全黑
};
```

//~~~~~完~~~~~