SHENZHEN HANSHENG INDUSTRAIL CO.LTD.,

HS1602A

DATASHEET

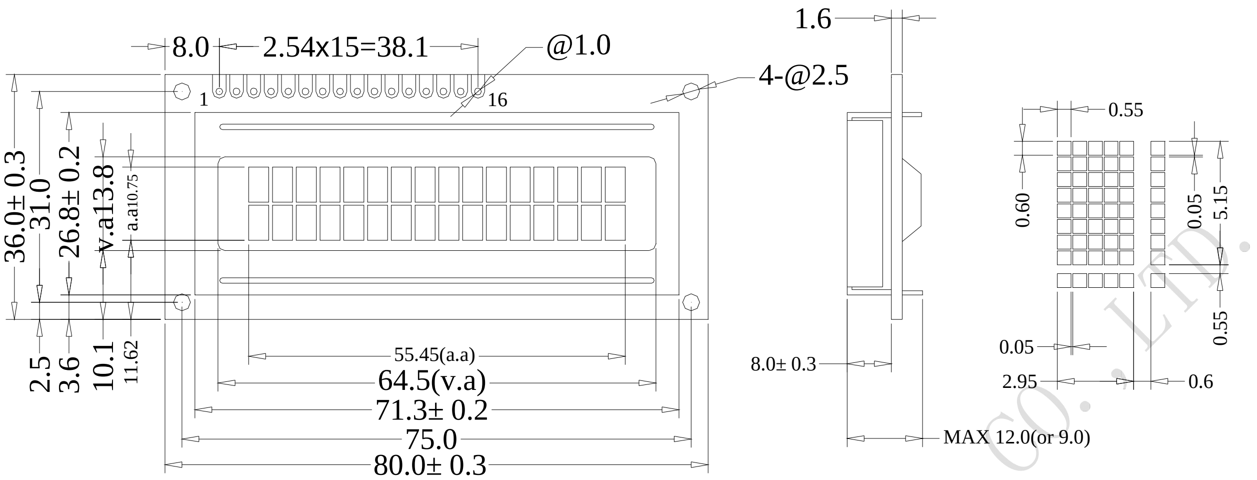
HS	PreparedBy	CheckedBy	ApprovedBy

VER： 1.0	Y/G 2.54*16P
VER： 1.1	

SHENZHEN HANSHENG INDUSTRAIL CO.LTD.,
Add：3th Floor, Building 213, First Industrial Zone,Niucheng Road, Xili Town, Nanshan District, Shenzhen City, Guangdong Province, China
Tel：+86-0755-86114312 86114314
Http：www.hslcm.com   Email:hansheng_lf@163.com

Catalog

一. External Structur



ITEM	NOMINAL DIMEN	UNIT
模块体积	80×36.0×11.0	mm
Visible Area	64.5×13.9	mm
Actual Area	55.7×11.0	mm
Dots	16×2	dots
Dots Distance	0.6×0.54	mm
Dot Size	0.65×0.59	mm
Hold Distance	75.0×31.0（4-◎2.5）	mm

二. LCM summary

HS1602A is a character LCD module.Can display 1line 16 character, Each character is made up of 5 x 8 dot matrix character block set, The LCM uses the COB production process, Stable structure, long service life.

Widely used in intelligent instrumentation, communications, office automation and military fields.

Main characteristics

◆ 8 bit parallel data interface, adapting M6800 time series

◆Optional 4 bit parallel data

◆A character generator ROM

◇192 and 5 x 8 point font characters

◇64 and 5 x 10 point font characters

◆A character generator RAM,

◇8 and 5 x 8 point font characters

◇4 and 5 x 10 point font characters

◆ Low power consumption, high reliability

三. interface definition

1. 2.54*16P

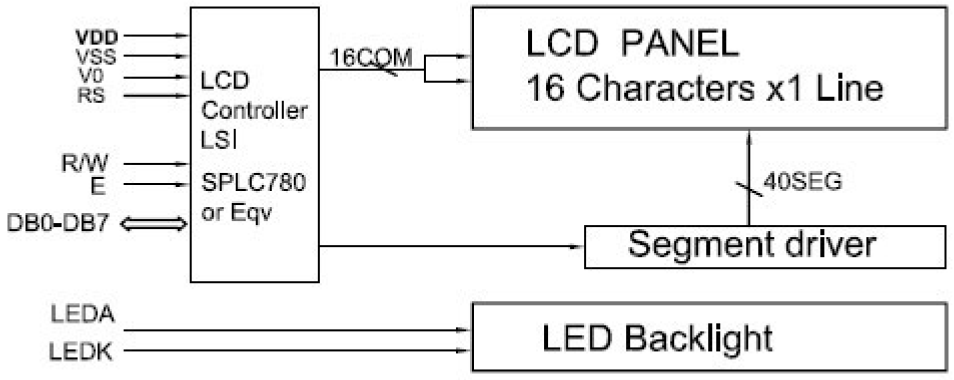
PIN	Name	Power	Functional Description
1	VSS	0V	Power ground
2	VCC	5.0V	Power supply
3	V0	0.2V	*

4	RS	H/L	RS=H, Display data
			RS=L， Instruction

5	R/W	H/L	R/W=H, Read data R/W=L, Write data
6	E	H,H→L	Enable signal
7	DB0	H/L	Data
8	DB1	H/L	Data
9	DB2	H/L	Data
10	DB3	H/L	Data
11	DB4	H/L	Data
12	DB5	H/L	Data
13	DB6	H/L	Data
14	DB7	H/L	Data
15	BLA	---	NC (LEDA)
16	BLK	---	NC (LEDK)

*this pin need connect the VR(10K) to VSS.

Schematic Diagram



2. The Max Scope Of Work

- ◆ Power supply（VDD）:4.9V-5.1V
- ◆ Power ground（VSS): 0V
- ◆ LCD Power supply（Vop）:-0.2V – +0.3V

3. **Electrical characteristics(**Test conditions Ta=25, VDD=5.0V±0.5V**)**

- ◆ High level input（Vih)： 2.2～VDD
- ◆ Low level input(Vil)： -0.3V～0.6V
- ◆ High level Out（Voh)： 2.4V～VDD
- ◆ Low level Out（Vol)： 0.4V Max
- ◆ Working current : 1mA
- ◆ LED current : 18mA

四. **Controller Description**

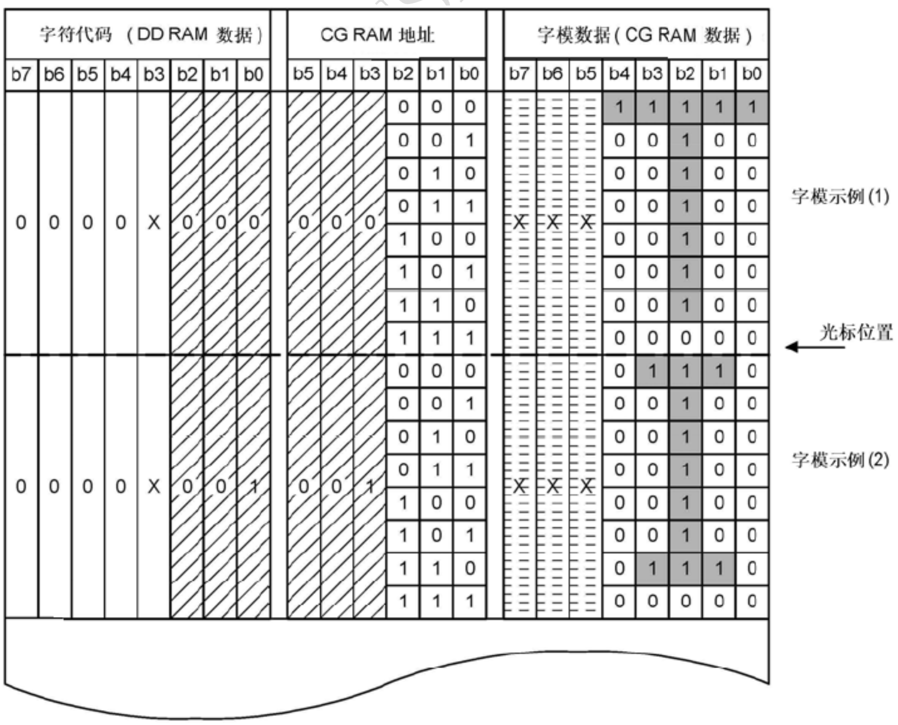
1. （DDRAM）

DDRAM（80×8bits）Is used to store the current to display the character code. DDRAM address pointer address pointer counter is provided by AC. DDRAM each unit corresponds to each character position on the display. After the initialization. DDRAM address the screen corresponding relations are as follows:

Character display position	1	2	3	………	14	15	16
The first line of a DDRAM address	00	01	02		0D	0E	0F
The Second line of a DDRAM address	40	41	42		4D	4E	4F

The correspondence in the setting of a cursor or picture scroll will change, please see the detailed instructions part.

2. Custom Character Storage（CGRAM）



字符代码（DD RAM 数据）								CG RAM 地址						字模数据（CG RAM 数据）							
b7	b6	b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	X	0	0	0	0	0	0	0	0	0	X	X	X	1	1	1	1	1
											0	0	1				0	0	1	0	0
											0	1	0				0	0	1	0	0
											0	1	1				0	0	1	0	0
											1	0	0				0	0	1	0	0
											1	0	1				0	0	1	0	0
											1	1	0				0	0	1	0	0
											1	1	1				0	0	0	0	0
0	0	0	0	X	0	0	1	0	0	1	0	0	0	X	X	X	0	1	1	1	0
											0	0	1				0	0	1	0	0
											0	1	0				0	0	1	0	0
											0	1	1				0	0	1	0	0
											1	0	0				0	0	1	0	0
											1	0	1				0	0	1	0	0
											1	1	0				0	1	1	1	0
											1	1	1				0	0	0	0	0

From here you can see that the custom character memory CGRAM address, CGRAM within the font data and the relationship between the custom character codes. Custom character codes and address CGRAM bit3 ~ bit5 bits of data consistency, Hit "X" is unrelated to the data, When the character code of the bit4 ~ bit7 are 0, a character code corresponding to the custom character.

Example: input the code for the 00H or 08H, which display a custom character" T"

3． Address pointer counter

Address pointer counter AC is read / write counter. He was DDRAM and CGRAM share address pointer counter, CPU has recently written address setting instruction identification code to identify. Can be arranged into a counter and a counter, when the read / write operation after the address pointer counter automatic correction. AC also as the cursor and the flashing position address pointer, indicating the current cursor and flashing location address.

4. Reset circuit

The SPLC780 control unit having a built-in reset circuit, power on the reset hardware initialization:

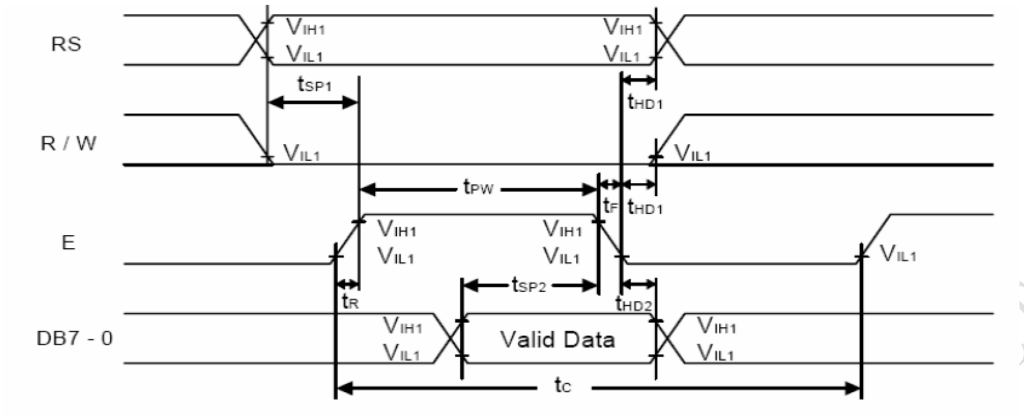
⊙Screen, DDRAM zero
⊙Set to 8 bus interface mode, a 5 x 7 font display
⊙Display off
⊙Input mode for pointer address counter AC plus a form
During initialization, interface of CPU is busy state.

五、Temporal specification (adaptation of M6800 timing)

1. Control sequence table

RS	R/W	E	DB0-DB7	功能
0	0	Falling edge	input	Write instructions
0	1	1	output	Read BF and AC
1	0	Falling edge	input	Write Display Data
1	1	1	output	Read Display Data

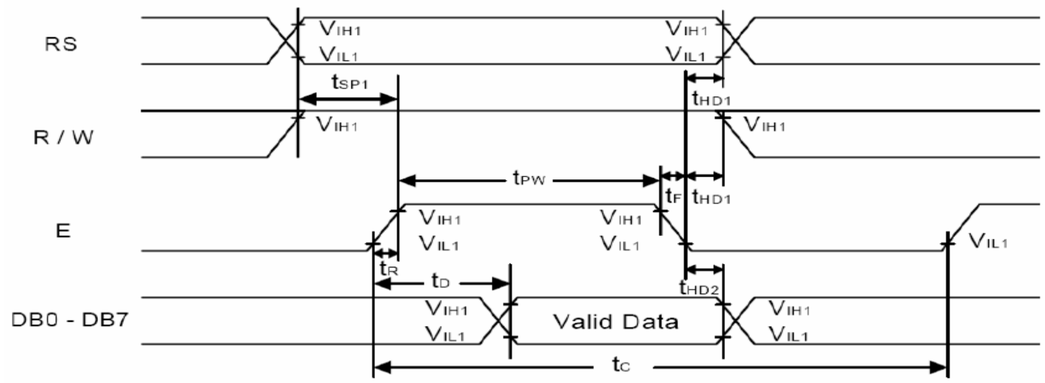
2. Write operation sequence



Write operation（VDD=2.7V～5.5V/4.5V～5.5V　Ta=-20℃～+75℃）

Project	Conform to	Min	Max	Unit
E Cycle Time	Tcyce	1000/500	-	ns
E (H) pulse width	Pweh	450/230	-	ns
E Rising edge/falling edge time	Ter/Tef	-	25/20	ns
Address setting time (RS R/W-E)	Tas	60/40	-	ns
Address retention time	Tah	20/10	-	ns
Data Setup Time	Tdsw	195/80	-	ns
Data retention time	Th	10	-	ns

3. Read operation timing



Read operation（VDD=2.7V～5.5V/4.5V～5.5V　Ta=-20℃～+75℃）

Project	Conform to	Min	Max	Company
E Cycle Time	Tcyce	1000/500	-	ns
E (H) pulse width	Pweh	450/230	-	ns
E Rising edge/falling edge time	Ter/Tef	-	25/20	ns

Address setting time (RS R/W-E)	Tas	60/40	-	ns
Address retention time	Tah	20/10	-	ns
Data Setup Time	Tdsw	-	360/160	ns
Data retention time	Th	5	-	ns

六、 Instructions

1、 Instruction list

Command name	control		set								Time
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D1	
Clear	0	0	0	0	0	0	0	0	0	1	1.64ms
Return Home	0	0	0	0	0	0	0	0	1	x	1.64ms
Enter Mode Set	0	0	0	0	0	0	0	1	I/D	SH	42μs
Display on/off set	0	0	0	0	0	0	1	D	C	B	42μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	X	X	42μs
Function Set	0	0	0	0	1	DL	N	F	X	X	42μs
CGRAM Set address	0	0	0	1	A5	A4	A3	A2	A1	A0	42μs
DDRAM Set address	0	0	1	A6	A5	A4	A3	A2	A1	A0	42μs
Read BF and AC	0	1	BF	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	0μs
Write data	1		data								46μs
Read data			data								46μs

Display condition：f_{osc}=270KHz

X said nothing, 0 or 1 can be

2. Instruction

1. Clear Display, 01H

format								
	0	0	0	0	0	0	0	1

Free code (20H) write to 80 DDRAM strategy unit, clearance; will address pointer counter clear AC, cursor or flashing to HOME; input mode parameter I / D is set to 1, the address pointer AC automatically add a way.

The instruction for power or update the full-screen display con

2. Return Home, 02H

format								
	0	0	0	0	0	0	1	0

The instruction address pointer counter clear AC.. Execute the instruction effect: the cursor or flashing back to the screen with the left bit first character, namely the DDRAM address OOH unit position; this is because the cursor and flashes are to address pointer counter AC current value orientation. If the picture has been rolling, rolling effect is revoked, the screen back to the Home bit.

3. Enter Mode Set, 04H～07H

format								
	0	0	0	0	0	1	I/D	S

The directive function is set to display character input mode, whereby the CPU read / write DDRAM or CGRAM, address pointer counter AC modified methods, reaction in the display effect, when writing a character after the screen or the movement of a cursor. The instruction of the two parameters of I / D and S determine four kinds of character input mode, As shown in the following table:

Input Model	code	I/D(set AC)	Moving and Cursor
Tableau not moving Cursor left	04H	0 (AC is -1 counter)	0 No scroll
The picture scroll right	05H	0 (AC is -1 counter)	1 scroll
Tableau not moving Cursor right	06H	1 (AC is +1 counter)	0 No scroll

The picture scroll left	07H	1(AC is +1 counter)	1 scroll

Note: the picture scroll in the CPU read DDRAM data, or in a read / write CGRAM is invalid, that is to say the directive is mainly used in CPU writing DDRAM data operation.

4. Display on/off Control，08H～0FH

format							
0	0	0	0	1	D	C	B

The instruction control screen, the cursor and flashing on and off. The instructions are three state D, C, B, the three state bit respectively control the picture, the cursor and the flashing display state.

Flash appeared in character or cursor display character position, normal display state for the current character or cursor display, bright display state for the character that all shows. If in character or cursor display character position, normal display state for no display, bright display state for the character bits all display all. The flashing mode can be designed into a block cursor on a computer display, like block cursor blink prompt effect.

The instruction of the 5 states as shown in the following table:

Instruction	FrameDisplayState	CursorDisplayState	BlinkingDisplayState	Function
	D	C	B	
（08H-0BH）	0 Picture off	*	*	Picture off
0CH	1 Picture on	0 Cursor off	0 Blinking off	Picture on
0DH	1 Picture on	0 Cursor off	1 Blinking on	PictureBlinkingOn
0EH	1 Picture on	1 Cursor on	0 Blinking off	Picture CursorOn
0FH	1 Picture on	1 Cursor on	1 Blinking on	画面光标闪烁显示

5. Cursor or Display Shift， 10H、14H、18H、1CH

format							
0	0	0	1	S/C	R/L	0	0

The execution of the instruction will generate pictures or the cursor to the left or right rolling a character position. If the time between the execution of the instruction will turn off the screen or cursor smooth scrolling. The picture scroll is within a line loop, that is to say a line of the first unit and the last one together, to form a closed loop type rolling. Rolling screen display as shown below:

CharacterDisplayPosition	1	2	3	…	38	39	40
The first line of a DDRAM address	00	01	02	…	25	26	27
The Second line of a	40	41	42	…	65	66	67

DDRAM address							

a. The two row shows the DDRAM unit and the display character original position

CharacterDisplayPosition	1	2	3	…	38	39	40
The first line of a DDRAM address	27	00	01	…	24	25	26
The Second line of a DDRAM address	67	40	41	…	64	65	66

b. Picture scroll right when DDRAM unit and a display character bit change

CharacterDisplayPosition	1	2	3	…	38	39	40
The first line of a DDRAM address	01	02	03	…	26	27	00
The Second line of a DDRAM address	41	42	43	…	66	67	40

c. Picture scroll left DDRAM unit and a display character bit change

When a cursor display, perform scrolling instruction does not modify the address pointer counter AC value, when the cursor is displayed, as a result of the implementation of a scroll instruction will cause the cursor to move, so the address pointer counter AC needs to be modified. If the cursor pointer, pointer address counter AC plus and minus one function to explain, can understand the cursor from the first display position left moved to a eightieth display position. Or from eightieth bit to first bit shift display display principle.

Cursor scroll function can be used to search for the need to modify the display character.

The order has 2 parameters, combination of features as shown in the following table:

Instruction	Rolling object	Direction of rolling	Function
	S/C	R/L	
10H	0 Cursor	0 Left	Cursor Left shift
14H	0 Cursor	1 Right	Cursor Right shift
18H	1 Picter	0 Left	**Picter** Left shift
1CH	1 Picter	1 Right	**Picter** Right shift

The instruction and input mode set command can generate cursor or picture scroll, difference is the instruction for scrolling function, a display, a rolling effect. While the input setting instruction is only completed a character input mode settings, only in the CPU on the DDRAM operation can produce rolling effect.

6. Function Set, 30H

format	0	0	1	DL	N	F	0	0

The instruction set of the controller works, including a controller with CPU interface and controller display driver duty ratio coefficient. The order has 3 parameters, combination of features as shown in the following table:

Explain	Data Line set		Line number set		Font set	
	DL		N		F	
	0	1	0	1	0	1
	4 Bit bus	8 Bit bus	1 Line	2 Line	5X7 fount	5X10 fount
Instruction	38H		Data line is 8 bit bus, 2line, font is 5*7, Duty ratio is 1/16。			

The instruction set of the controller works, is the only software reset command. Although SPLC780 has a reset circuit, but in order to work reliably, SPLC780 CPU when in operation the first software reset. That is to say the control character LCD module work to the first software reset.

7. CGRAM Address setting（40H～7FH）

format	0	1	A5	A4	A3	A2	A1	A0

The instructions will be 6 CGRAM address into the address pointer counter AC, then the computer operations on the data on the CGRAM read / write.

8. DDRAM Address setting（80H～FFH）

format	1	A6	A5	A4	A3	A2	A1	A0

The instructions will be 7 DDRAM address into the address pointer counter AC, then the computer operations on the data on the DDRAM read / write.

9. Read busy flag and address pointer value

format	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Computer instruction register read operation that RS 0R / W 1, will read out the format of the busy flag BF value and the 7 address pointer counter AC value.

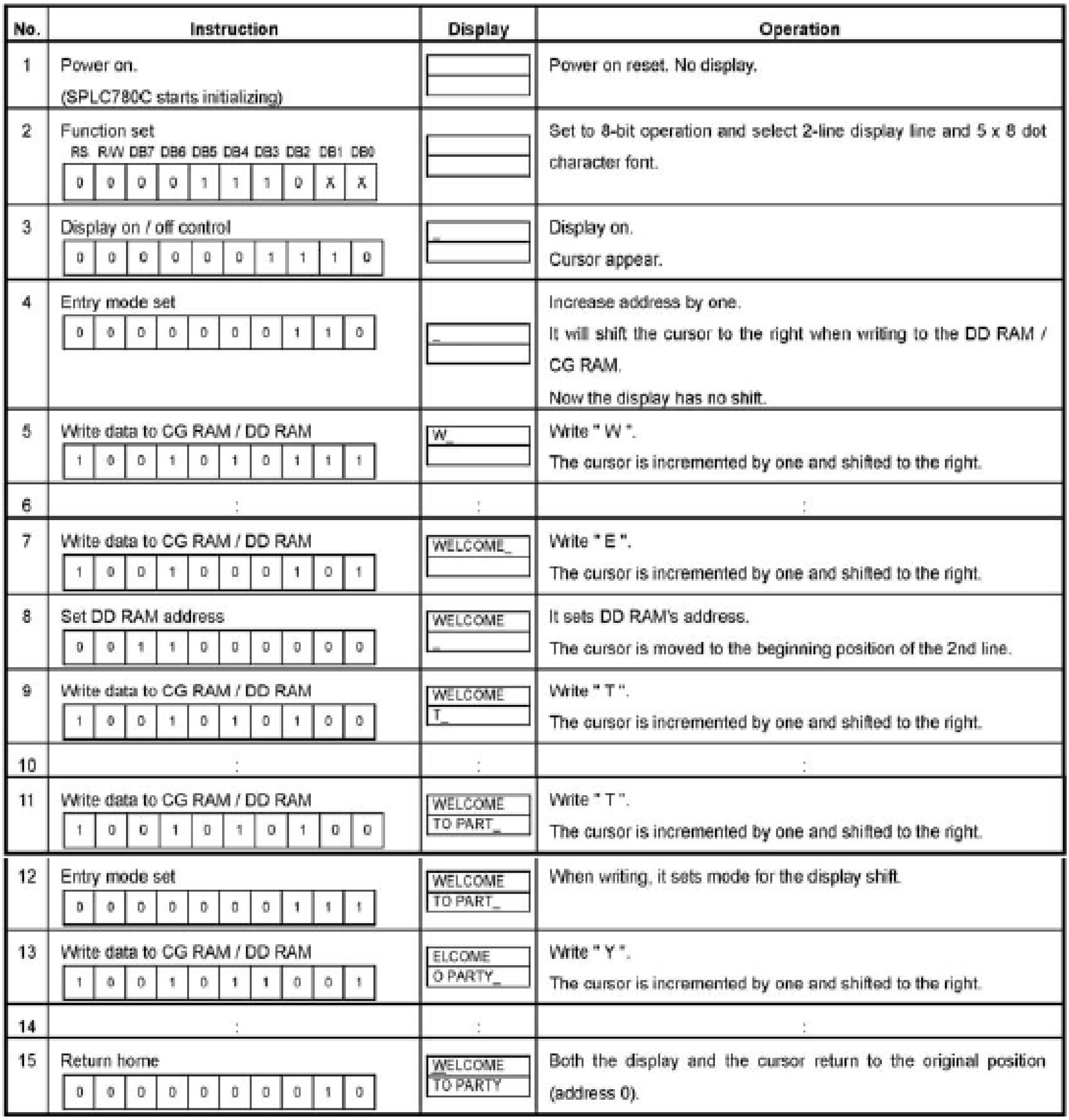
10. Writer Data to CG or DDRAM

CPU to write data register data channel, SPLC780 according to the current address pointer counter AC value attribute and value will be the data into the corresponding memory AC refers to the unit. If the AC value for the DDRAN address pointer, thinks that the written data is the character code and sent to DDRAM AC refers to the unit. If the AC value for the CGRAM address pointer, thinks that the written data is custom character font data and sent to CGRAM AC refers to the unit. So the CPU write data are required before a set address pointer or artificial confirmation address pointer attribute and value. After writing data address pointer counter AC will be based on recent setting input maximum modification. Therefore, CPU in the write operation of data prior to do two jobs, one is set or to confirm the address pointer counter AC value attributes and values, to ensure that the data written to the right place, the second is set or confirm input mode, to ensure continuous write data ac value modified methods to meet the requirements.

11. Read Data From CG or DDRAM

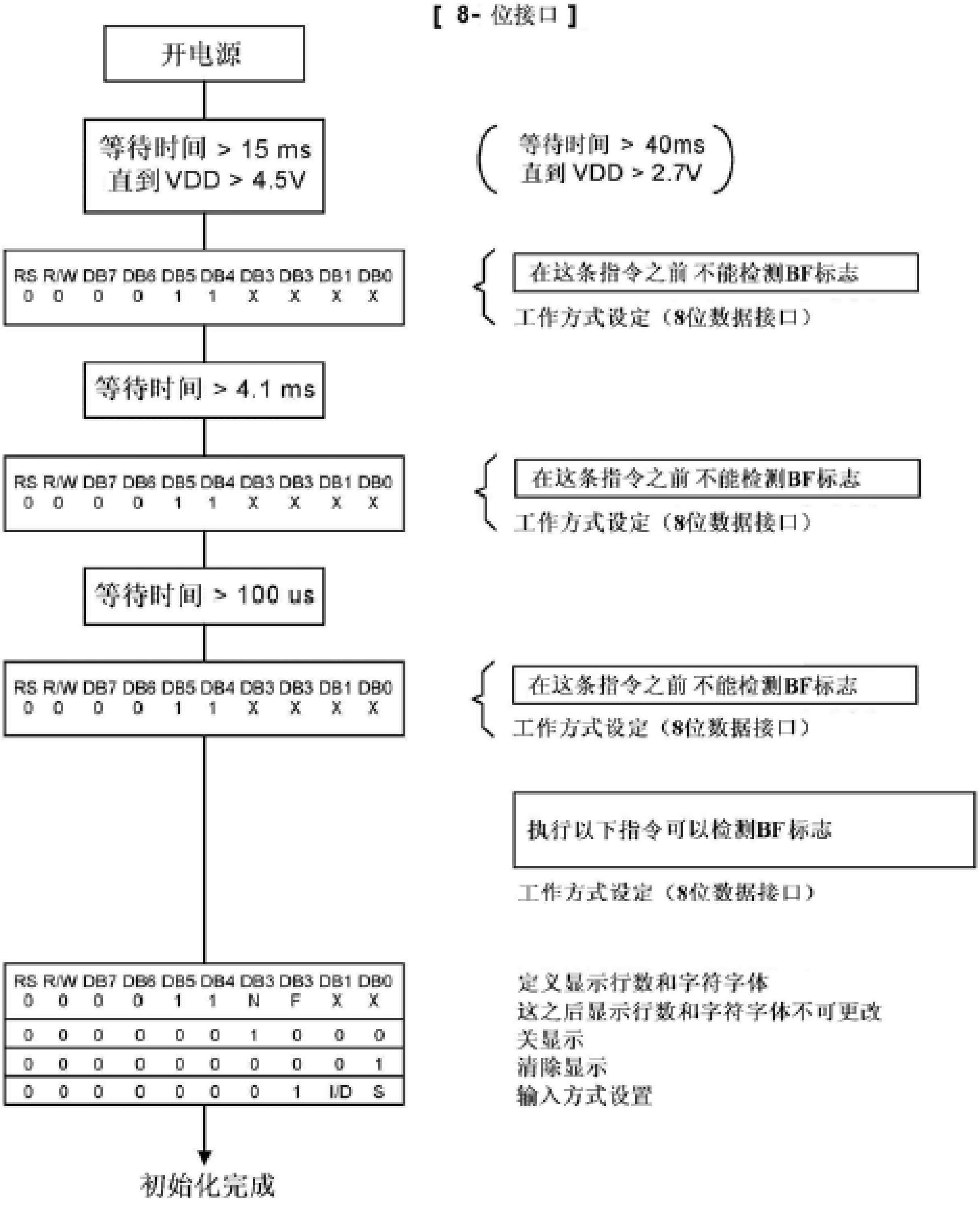
Under the internal timing operation of SPLC780, every modification of the address pointer counter AC value, including the writing of new AC values, the modification of AC values caused by cursor scroll displacement, or the modification of AC values generated by CPU read and write data operations, will be sent to the data output register by SPLC780 for CPU reading. If the AC value is a DDRAM address pointer, it is considered to be reading the character code of the unit pointed to by AC in DDRAM. If the AC value is an address pointer in CGRAM, it is considered to be reading the font data of the custom character of the unit pointed to by AC in CGRAM.

2. Operation demonstration (8 data interface)



No.	Instruction	Display	Operation
1	Power on. (SPLC780C starts initializing)		Power on reset. No display.
2	Function set RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1 1 0 X X		Set to 8-bit operation and select 2-line display line and 5 x 8 dot character font.
3	Display on / off control 0 0 0 0 0 0 1 1 1 0	_	Display on. Cursor appear.
4	Entry mode set 0 0 0 0 0 0 0 1 1 0	_	Increase address by one. It will shift the cursor to the right when writing to the DD RAM / CG RAM. Now the display has no shift.
5	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 1 1	W_	Write " W ". The cursor is incremented by one and shifted to the right.
6	:	:	:
7	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 1 0 1	WELCOME_	Write " E ". The cursor is incremented by one and shifted to the right.
8	Set DD RAM address 0 0 1 1 0 0 0 0 0 0	WELCOME _	It sets DD RAM's address. The cursor is moved to the beginning position of the 2nd line.
9	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 0 0	WELCOME T_	Write " T ". The cursor is incremented by one and shifted to the right.
10	:	:	:
11	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 0 0	WELCOME TO PART_	Write " T ". The cursor is incremented by one and shifted to the right.
12	Entry mode set 0 0 0 0 0 0 0 1 1 1	WELCOME TO PART_	When writing, it sets mode for the display shift.
13	Write data to CG RAM / DD RAM 1 0 0 1 0 1 1 0 0 1	ELCOME O PARTY_	Write " Y ". The cursor is incremented by one and shifted to the right.
14	:	:	:
15	Return home 0 0 0 0 0 0 0 0 1 0	WELCOME TO PARTY	Both the display and the cursor return to the original position (address 0).

3. Procedures (8 data interface)



[8- 位接口]

开电源

等待时间 > 15 ms
直到VDD > 4.5V

（等待时间 > 40ms
直到 VDD > 2.7V）

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB3	DB1	DB0
0	0	0	0	1	1	X	X	X	X

在这条指令之前 不能检测BF标志

工作方式设定（8位数据接口）

等待时间 > 4.1 ms

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB3	DB1	DB0
0	0	0	0	1	1	X	X	X	X

在这条指令之前 不能检测BF标志

工作方式设定（8位数据接口）

等待时间 > 100 us

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB3	DB1	DB0
0	0	0	0	1	1	X	X	X	X

在这条指令之前 不能检测BF标志

工作方式设定（8位数据接口）

执行以下指令可以检测BF标志

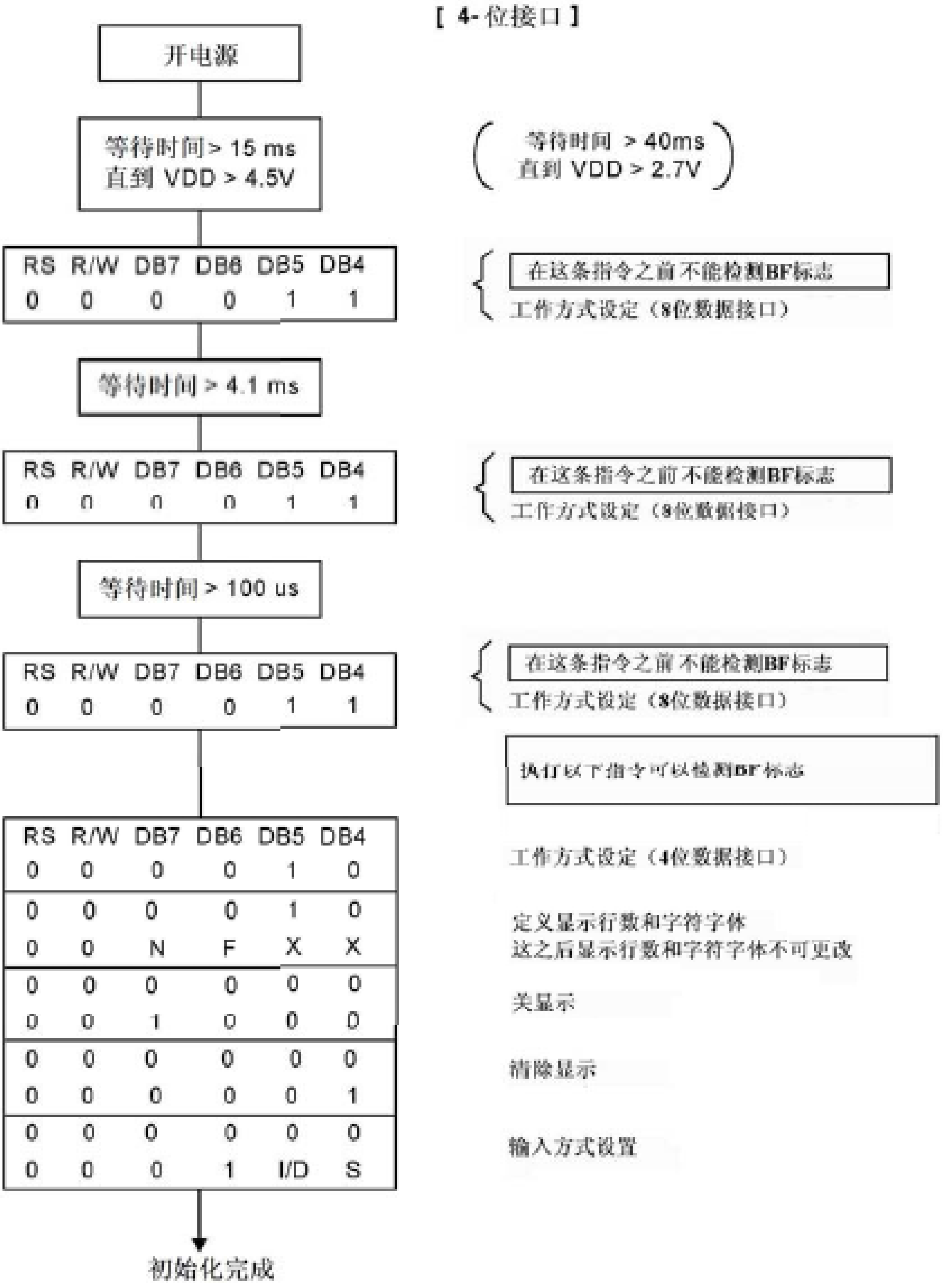
工作方式设定（8位数据接口）

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB3	DB1	DB0
0	0	0	0	1	1	N	F	X	X
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	I/D	S

定义显示行数和字符字体
这之后显示行数和字符字体不可更改
关显示
清除显示
输入方式设置

初始化完成

5. Procedures (4data interface)



[4- 位接口]

开电源

等待时间> 15 ms
直到 VDD > 4.5V

（等待时间 > 40ms
直到 VDD > 2.7V）

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

在这条指令之前不能检测BF标志
工作方式设定（8位数据接口）

等待时间 > 4.1 ms

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

在这条指令之前不能检测BF标志
工作方式设定（8位数据接口）

等待时间 > 100 us

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

在这条指令之前不能检测BF标志
工作方式设定（8位数据接口）

执行以下指令可以检测BF标志

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	0
0	0	0	0	1	0
0	0	N	F	X	X
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	0	0
0	0	0	1	I/D	S

工作方式设定（4位数据接口）

定义显示行数和字符字体
这之后显示行数和字符字体不可更改

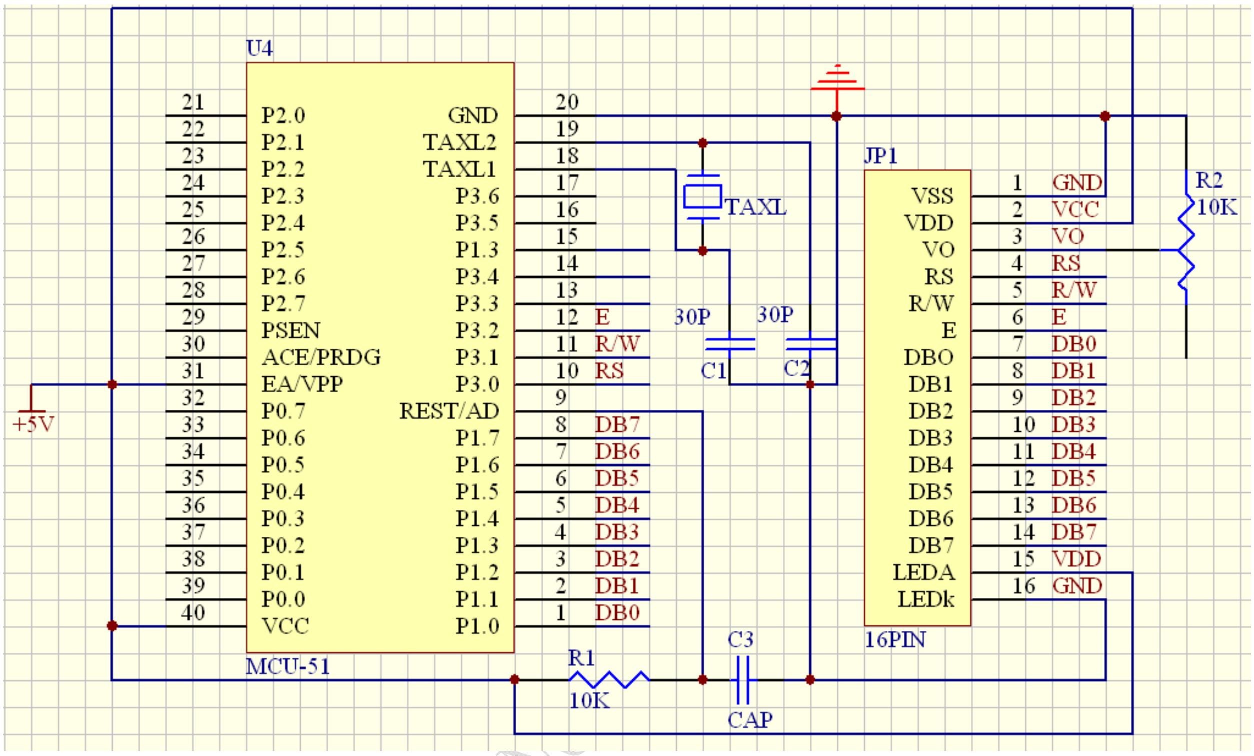
关显示

清除显示

输入方式设置

初始化完成

6、LCM and the MCU connection mode:



6、LCM INIT

```
//windond78e52b
#include<reg52.h>
#define uchar unsigned char
#define uint unsigned int

//sbit dula=P2^6;
//sbit wela=P2^7;
sbit lcdrs=P3^0;
sbit lcdrw=P3^1;
sbit lcden=P3^2;

uchar code table1[]={0x4f,0xf4,0x4f,0xf4,0x4f,0xf4,0x4f,0xf4,0x4f,0xf4,0x4f,0xf4,0x4f,0xf4,0x4f,0xf4};
uchar code
table2[]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f};
```

```
uchar                                                                                    code
table3[]={0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f};
uchar                                                                                    code
table4[]={0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5a,0x5b,0x5c,0x5d,0x5e,0x5f};

uchar                                                                                    code
table5[]={0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f};
uchar                                                                                    code
table6[]={0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7a,0x7b,0x7c,0x7d,0x7e,0x7f};

uchar code table7[]={0xa0,0xa1,0xa2,0xa3,0xa4,0xa5,0xa6,0xa7,0xa8,0xa9,0xaa,0xab,0xac,0xad,0xae,0xaf};
uchar                                                                                    code
table8[]={0xb0,0xb1,0xb2,0xb3,0xb4,0xb5,0xb6,0xb7,0xb8,0xb9,0xba,0xbb,0xbc,0xbd,0xbe,0xbf};

uchar code table9[]={0xc0,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7,0xc8,0xc9,0xca,0xcb,0xcc,0xcd,0xce,0xcf};
uchar                                                                                    code
table10[]={0xd0,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6,0xd7,0xd8,0xd9,0xda,0xdb,0xdc,0xdd,0xde,0xdf};

uchar code table11[]={0xe0,0xe1,0xe2,0xe3,0xe4,0xe5,0xe6,0xe7,0xe8,0xe9,0xea,0xeb,0xec,0xed,0xee,0xef};
uchar code table12[]={0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfd,0xfe,0xff};

uchar num,a,b;
//data=P1;    //MCU(P1.0-P1.7)=LCM(DB0-DB7)
void delay(uint z)
{
    uint x,y;
    for(x=z;x>0;x--)
        for(y=110;y>0;y--);
}
void write_com(uchar com)//写指令函数
{
    lcdrs=0;
    lcdrw=0;
    P1=com;
    delay(5);
    lcden=1;
    delay(5);
    lcden=0;
}

void write_data(uchar date)//写数据函数
{
    lcdrs=1;
```

```
        lcdrw=0;
        P1=date;
        delay(5);
        lcden=1;
        delay(5);
        lcden=0;
}

void init()//初始化函数
{
      lcden=0;//初始化为低电平
      write_com(0x01); //清屏
      write_com(0x38);//显示设置，16*2，5*7 ，8 位数据接口
      write_com(0x0c);//开显示，光标不显示，光标不闪烁
//    write_com(0x06);//写一个字符光标左移

}

void main()
{
    while(1){
    //dula=0;
    //wela=0;

      init();

      write_com(0x40+0x00);//第一屏
          for(a=0;a<16;a++)
                {write_data(0x55);}
      write_com(0x80+0x40);
          for(b=0;b<16;b++)
                {write_data(0x55);}
//              for(num=0;num<16;num++)
//                    {
//                      write_com(0x18);
//                      delay(300);
//                        }
          delay(1000);
          //write_data(0x80);

      write_com(0x80+0x00);//第 2 屏
          for(a=0;a<16;a++)
                {write_data(table1[a]);}
      write_com(0x80+0x40);
      for(b=0;b<16;b++)
          {write_data(table2[b]);}
//        for(num=0;num<16;num++)
//            {
//             write_com(0x18);
//             delay(300);
//               }
      delay(1000);
      //write_data(0x80);

    write_com(0x80+0x00);//第 3 屏
      for(a=0;a<16;a++)
          {write_data(table3[a]);}
    write_com(0x80+0x40);
      for(b=0;b<16;b++)
          {write_data(table4[b]);}
//        for(num=0;num<16;num++)
//            {
//             write_com(0x18);
//             delay(300);
//               }
      delay(1000);
      //write_data(0x80);

    write_com(0x80+0x00);//第 4 屏
      for(a=0;a<16;a++)
          {write_data(table5[a]);}
    write_com(0x80+0x40);
      for(b=0;b<16;b++)
          {write_data(table6[b]);}
//        for(num=0;num<16;num++)
//            {
//             write_com(0x18);
//             delay(300);
//               }
      delay(1000);
      //write_com(0x80);

    write_com(0x80+0x00);//第 5 屏
      for(a=0;a<16;a++)
          {write_data(table7[a]);}
    write_com(0x80+0x40);
      for(b=0;b<16;b++)
          {write_data(table8[b]);}
```

```
//              for(num=0;num<16;num++)
//                    {
//                      write_com(0x18);
//                      delay(300);
//                        }
          delay(1000);
        // write_com(0x80);;

      write_com(0x80+0x00);//第 6 屏
          for(a=0;a<16;a++)
                {write_data(table9[a]);}
      write_com(0x80+0x40);
          for(b=0;b<16;b++)
                {write_data(table10[b]);}
//              for(num=0;num<16;num++)
//                    {
//                      write_com(0x18);
//                      delay(300);
//                        }
          delay(1000);
        // write_com(0x80);

      write_com(0x80+0x00);//第 7 屏
          for(a=0;a<16;a++)
                {write_data(table11[a]);}
      write_com(0x80+0x40);
          for(b=0;b<16;b++)
                {write_data(table12[b]);}
//              for(num=0;num<16;num++)
//                    {
//                      write_com(0x18);
//                      delay(300);
//                        }
                        delay(1000);
          //write_com(0x80);


}
        }
```