LCD

rDuinoStar

HJ12864ZW

LCD Module Technical Manual

Content

This Work Description

Original language: Chinese.
Translated by rDuinoStar Team. It can contain some errors.
Basic bibliography: ST7920 Datasheet.

ST7920 LCD controller/driver IC can display alphabets, numbers, Chinese fonts and self-defined characters. It supports 3 kinds of bus interface, namely 8-bit, 4-bit and serial. All functions, including display RAM, Character Generation ROM, LCD display drivers and control circuits are all in a one-chip solution. With a minimum system configuration, a Chinese character display system can be easily achieved.

ST7920 includes character ROM with 8192 16x16 dots Chinese fonts and 126 16x8 dots half-width alphanumerical fonts. Besides, it supports 64x256 dots graphic display area for graphic display (GDRAM). Mix-mode display with both character and graphic data is possible. ST7920 has built-in CGRAM and provide 4 sets software programmable 16x16 fonts.

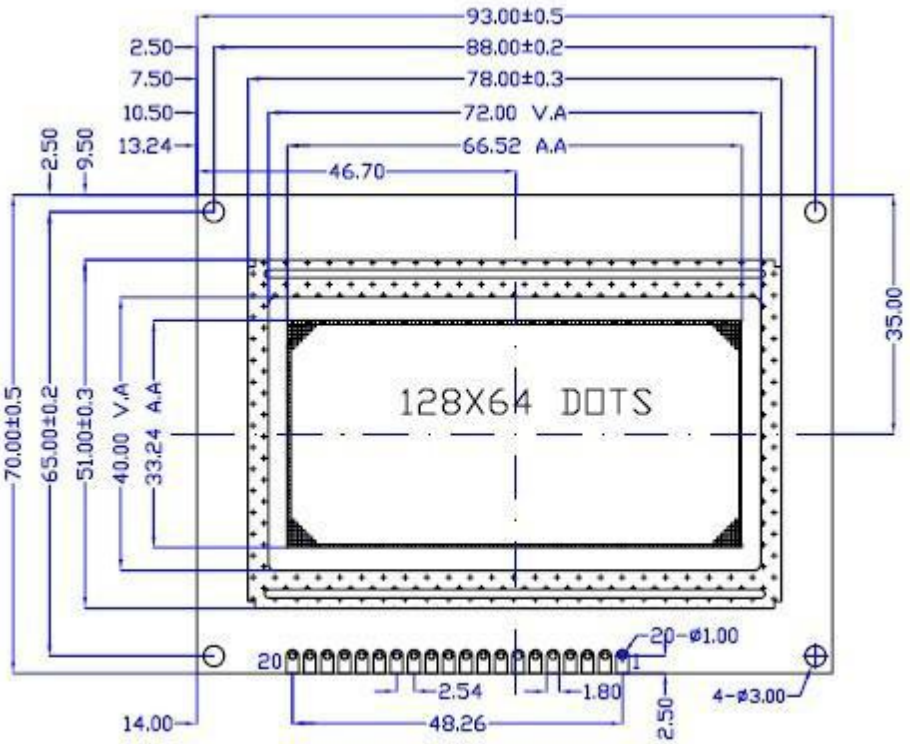
ST7920 has wide operating voltage range (2.7V to 5.5V). It also has low power consumption. So ST7920 is suitable for battery-powered portable device.

ST7920 LCD driver consists of 32-common and 64-segment. Company with the extension segment driver (ST7921) ST7920 can support up to 32-common x 256-segment display.

Overview

HJ12864ZW is a graphical dot matrix liquid crystal displays, mainly by line drive / column driver The 128X64 full dot-matrix liquid crystal display components, to be completed by the graphic display can also display 8X4 (16X16 dot matrix Chinese character, with an external CPU interface can be serial or parallel control The system.

Dimensions



Project	Reference value
The LCM dimensions (LxWxT)	93.0×70.0×13.5
Visual area (LxW)	72.0×40.0
Spacing (LxW)	0.52×0.52
Point size (LxW)	0.48×0.48
Logic operating voltage (Vdd)	+5.0V or +3.3V（factory-set to +5.0V）
The LCD drive voltage (Vdd-V0)	+3.0V to +5.0V
Operating temperature (Ta)	0 to +50℃（room temp) / -20ºC to +70℃（extended temp)
Storage temperature (Tsto)	-10 to +60℃（room temp) / -30 to +80℃（extended temp）
Oper. current except backlight	3.0mA(max)

Pin Description

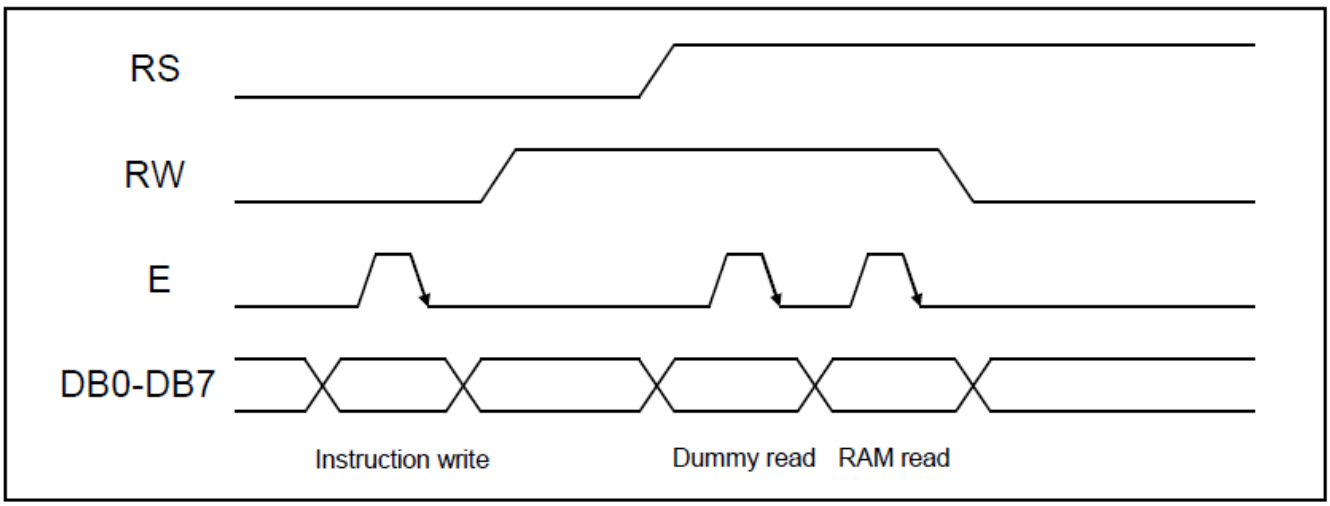
Pin	Name	I/O	Description
1	V_{SS}	--	Negative power supply(0V)
2	V_{DD}	--	Positive power supply(+3.3V to +5.0V (the factory-set))
3	V0	--	LCD driver voltage (adjustable) (<=7V).
4	RS(CS)	I	**Parallel Mode:** Register select. RS=0: - During Write Operation: Selects instruction register or Busy Flag. - During Read Operation: Selects address counter. RS=1: - During Write/Read Operations: Select data register. **Serial mode:** Chip select. CS=1: Chip enabled. CS=0: Chip disabled. *When chip is disabled, SID and SCLK should be set as "H" or "L". Transcient of SID and SCLK is not allowed.*
5	R/W(SID)	I	**Parallel mode:** - R/W=0 write operation. - R/W=1 read operation. **Serial mode:** Serial data input
6	E(SCLK)	I	**Parallel:** enable trigger signal, active high. **Serial:** serial clock signal.
7-14	DB0 to DB3	I/O	Lower nibble data bus of 8-bit interface.
11-14	DB4 to DB7	I/O	Higher nibble data bus of 8-bit interface and data bus for 4-bit interface
15	PSB	I	Serial/Parallel control port to select the interface: 1: 8/4-bit parallel bus mode. 0: serial mode.
16	NC	--	Empty pin.
17	RST	I	System reset input (low active).
18	V_{OUT}	--	LCD voltage doubler output (V_{OUT} <= 7V) (V_{DD}＝+3.3V）
19	LEDA	--	(+) terminal of power supply, used for backlight.(+3.3V or +5.0V， factory set to +5.0V)
20	LEDK	--	(-) terminal of the power supply, used for backlight. (0V)

Timing

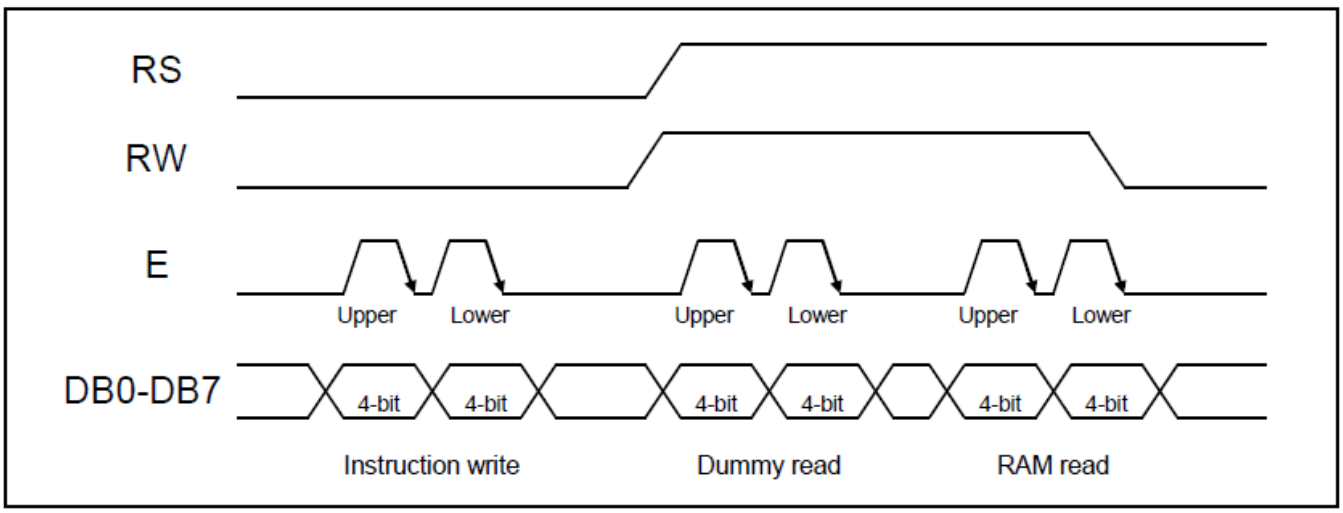
Parallel interface:

LCD module is in parallel mode by pulling up PSB pin. We can select 8-bit or 4-bit bus interface by setting the DL control bit in “Function Set” instruction. MPU can control RS, RW, E and DB0…DB7 pins to complete the data transmission.

In 4-bit transfer mode, every 8-bit data or instruction is separated into 2 parts. The higher 4 bits (bit-7~bit-4) data will be transferred first through data pins (DB7~DB4). The lower 4 bits (bit-3~bit-0) data will be transferred second through data pins (DB7~DB4). The (DB3~DB0) data pins are not used during 4-bit transfer mode.



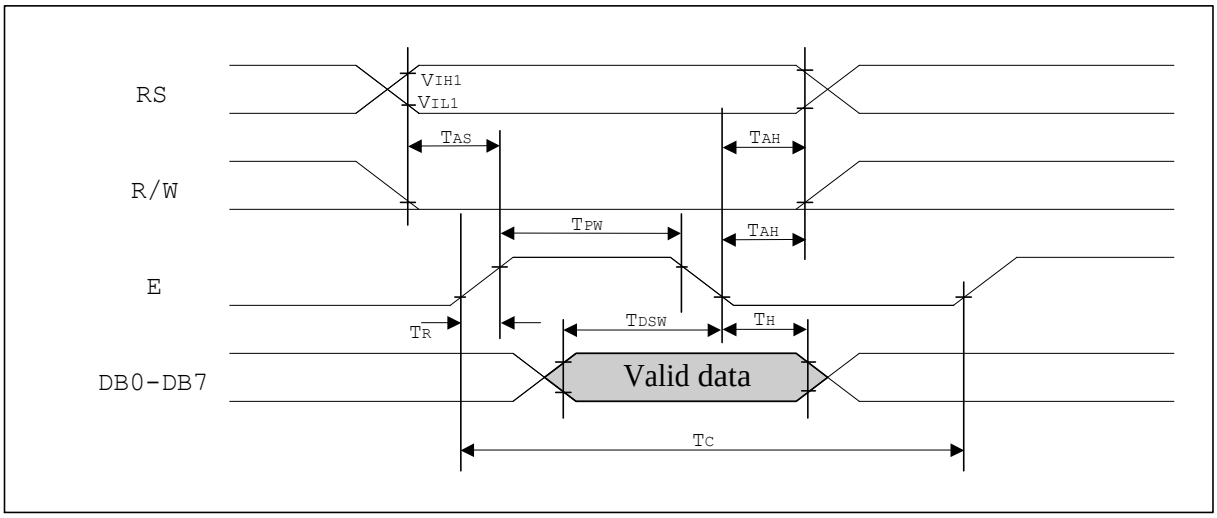
Timing Diagram of 8-bit Parallel Bus Mode Data Transfer



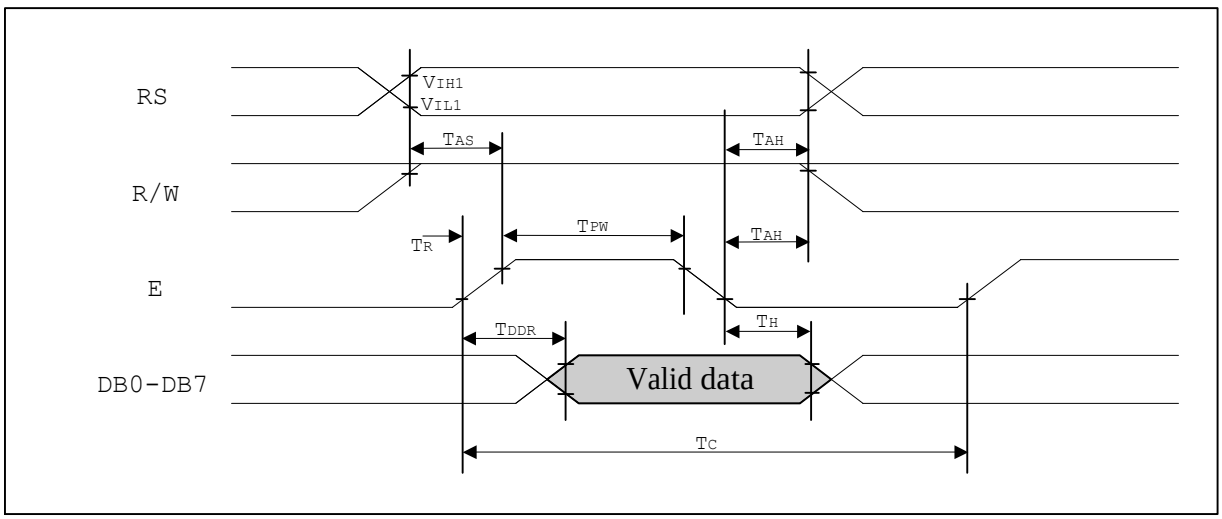
Timing Diagram of 4-bit Parallel Bus Mode Data Transfer

8-bit interface timing diagram

- MPU write data to the LCD Module



- MPU read data from the LCD Module



AC Characteristics (T_A = -25℃, V_{DD} = 4.5V) Parallel Mode Interface

Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
Internal Clock Operation						
f_{OSC}	OSC Frequency	R=33KΩ	480	540	600	KHz
External Clock Operation						
f_{EX}	External Frequency	-	480	540	600	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	μs
Write Mode (Writing data from MPU to ST7920)						
T_e	Enable Cycle Time	Pin E	1200	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	140	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS, RW, E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS, RW, E	20	-	-	ns
T_{DSW}	Data Setup Time	Pins: DBO - DB7	40	-	-	ns
T_H	Data Hold Time	Pins: DBO - DB7	20	-	-	ns
Read Mode (Reading data from ST7920 to MPU)						
T_e	Enable Cycle Time	Pin E	1200	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	140	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS, RW, E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS, RW, E	20	-	-	ns
T_{DDR}	Data Delay Time	Pins: DBO - DB7	-	-	100	ns
T_H	Data Hold Time	Pins: DBO - DB7	20	-	-	ns

Serial interface:

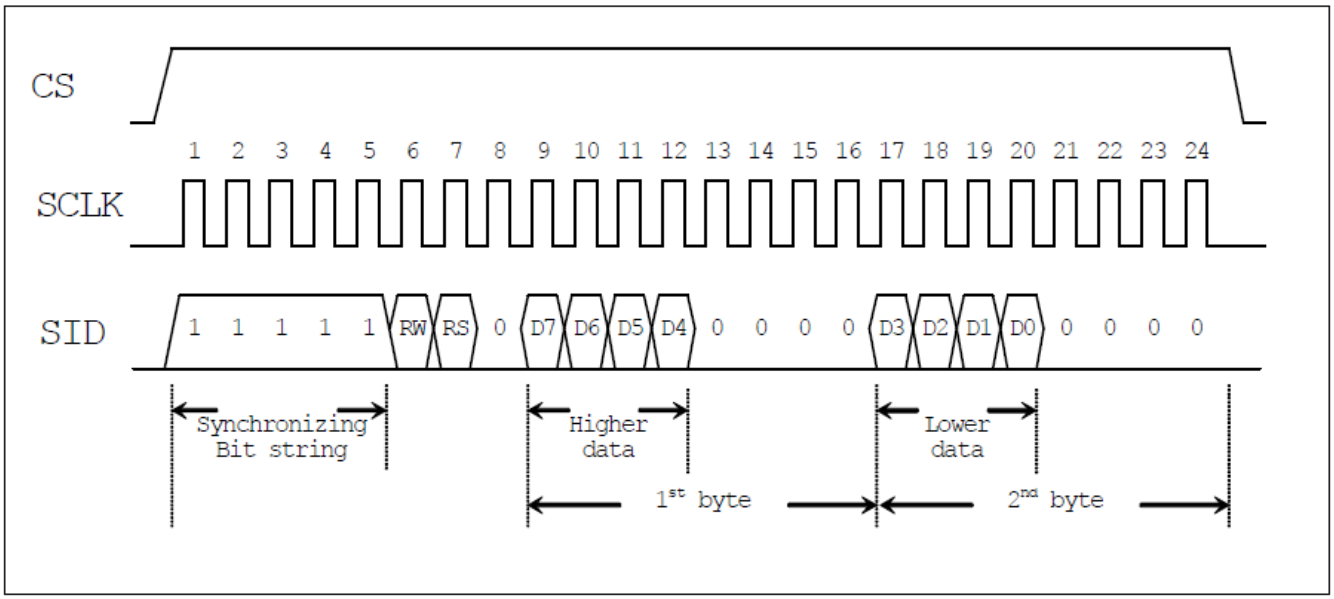
LCD module is in serial interface mode when pulling down PSB pin. Only write data is available in the serial interface mode.

When chip select (CS) is low, LCD module serial clock counter and serial data will be reset. Serial transfer counter is set to the first bit and data register is cleared. After CS is “L”, any further change on SID or SCLK is not allowed. It is recommended to keep SCLK at “L” and SID at the last status before set CS to “L”. For a minimal system with only one ST7920 and one MPU, only SCLK and SID pins are necessary. CS pin should pull to high.

ST7920’s serial clock (SCLK) is asynchronous to the internal clock and is generated by MPU. When multiple instruction/data is transferred, the instruction execution time must be considered. MPU must wait till the previous instruction is finished and then send the next instruction. ST7920 has no internal instruction buffer area.

When starting a transmission, a start byte is required. It consists of 5 consecutive “1” (sync character). Serial transfer counter will be reset and synchronized. Followed by 2-bit flag that indicates: read/write (RW) and register/data selected (RS) operation. Last 4 bits are filled by “0”.

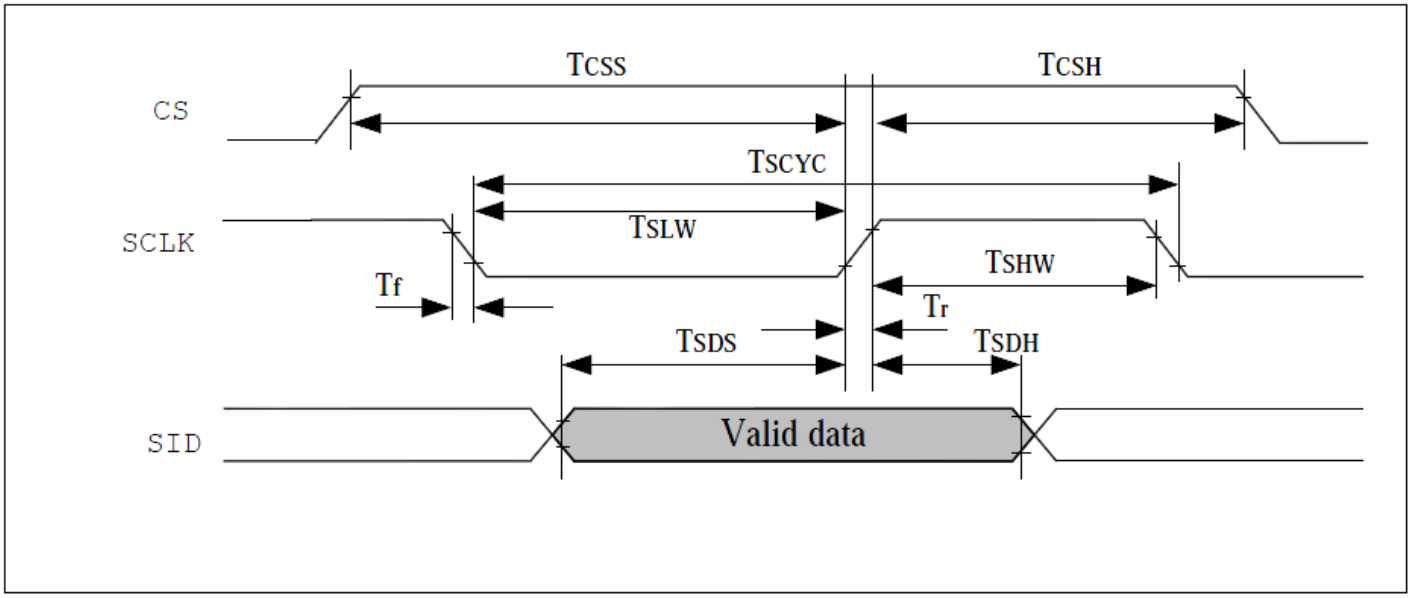
After receiving the sync character, RW and RS bits, every 8 bits instruction/data will be separated into 2 groups. Higher 4 bits (DB7~DB4) will be placed in the first section followed by 4 “0”s. And lower 4 bits (DB3~DB0) will be placed in the second section followed by 4 “0”s.



Timing Diagram of Serial Mode Data Transfer

Serial interface timing diagram

- MPU write data to the LCD Module



Serial mode AC characteristics (TA=25ºC, VDD=4.5V)

Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
		Internal Clock Operation				
f_{OSC}	OSC Frequency	R=33KΩ	470	530	590	KHz
		External Clock Operation				
f_{EX}	External Frequency	-	470	530	590	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	s
T_{SCYC}	Serial Clock Cycle	Pin E	400	-	-	ns
T_{SHW}	SCLK High Pulse Width	Pin E	200	-	-	ns
T_{SLW}	SCLK Low Pulse Width	Pin E	200	-	-	ns
T_{SDS}	SID Data Setup Time	Pin RW	40	-	-	ns
T_{SDH}	SID Data Hold Time	Pin RW	40	-	-	ns
T_{CSS}	CS Setup Time	Pin RS	60	-	-	ns
T_{CSH}	CS Hold Time	Pin RS	60	-	-	ns

Instruction Set

Instruction Set 1: (RE=0: Basic Instruction)

Inst.	Code										Description	Exec time (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Display Clear	0	0	0	0	0	0	0	0	0	1	Fill DDRAM with "20H" and set DDRAM address counter (AC) to "00H".	1.6 ms
Return Home	0	0	0	0	0	0	0	0	1	X	Set DDRAM address counter (AC) to "00H", and put cursor to origin ; the content of DDRAM are not changed	72 us
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Set cursor position and display shift when doing write or read operation	72 us
Display Control	0	0	0	0	0	0	1	D	C	B	D=1: Display ON C=1: Cursor ON B=1: Character Blink ON	72 us
Cursor Display Control	0	0	0	0	0	1	S/C	R/L	X	X	Cursor position and display shift control; the content of DDRAM are not changed	72 us
Function Set	0	0	0	0	1	DL	X	0 RE	X	X	DL=1 8-bit interface DL=0 4-bit interface **RE=1: extended instruction** **RE=0: basic instruction**	72 us
Set CGRAM Address.	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address to address counter (AC) **Make sure that in extended instruction SR=0 (scroll or RAM address select)**	72 us
Set DDRAM Address.	0	0	1	0 AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address to address counter (AC) AC6 is fixed to 0	72 us
Read Busy Flag (BF) & AC.	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Read busy flag (BF) for completion of internal operation, also Read out the value of address counter (AC)	0 us
Write RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data to internal RAM (DDRAM/CGRAM/GDRAM)	72 us
Read RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM/GDRAM)	72 us

Instruction set 2: (RE=1: extended instruction)

Inst.	Code										Description	Exec time (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Standby	0	0	0	0	0	0	0	0	0	1	Enter standby mode, any other instruction can terminate. COM1…32 are halted.	72 us
Scroll or RAM Address. Select	0	0	0	0	0	0	0	0	1	SR	SR=1: enable vertical scroll position SR=0: enable CGRAM address **(basic instruction)**	72 us
Reverse (by line)	0	0	0	0	0	0	0	1	R1	R0	Select 1 out of 4 line (in DDRAM) and decide whether to reverse the display by toggling this instruction **R1,R0 initial value is 0,0**	72 us
Extended Function Set	0	0	0	0	1	DL	X	1 RE	G	0	DL=1 :8-bit interface DL=0 :4-bit interface **RE=1: extended instruction set** **RE=0: basic instruction set** G=1 :graphic display ON G=0 :graphic display OFF	72 us
Set Scroll Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR=1: AC5~AC0 the address of vertical scroll	72 us
Set Graphic Display RAM Address	0	0	1	0 AC6	0 AC5	0 AC4	AC3 AC3	AC2 AC2	AC1 AC1	AC0 AC0	Set GDRAM address to address counter (AC) Set the vertical address first and followed the horizontal address by consecutive writings Vertical address range: AC6…AC0 Horizontal address range: AC3…AC0	72 us

Note:

1. Make sure that it is not in busy state by reading the busy flag before sending instruction or data. If using delay loop instead, please make sure the delay time is enough. Please refer to the instruction execution time.
2. “RE” is the selection bit of basic and extended instruction set. After setting the RE bit, the value will be kept. So that the software doesn’t have to set RE every time when using the same instruction set.

Description of basic instruction set

- Display Clear

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

This instruction will change the following items:

1. Fill DDRAM with "20H"(space code).
2. Set DDRAM address counter (AC) to"00H".
3. Set Entry Mode I/D bit to be "1". Cursor moves right and AC adds 1 after write or read operation.

- Return Home

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

Set address counter (AC) to "00H". Cursor moves to origin. Then content of DDRAM is not changed.

- Enry Mode Set

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

Set the cursor movement and display shift direction when doing write or read operation.

I/D: Address Counter Control: (Increase/Decrease)

When I/D = "1", cursor moves right, address counter (AC) is increased by 1.

When I/D = "0", cursor moves left, address counter (AC) is decreased by 1.

S: Display Shift Control: (Shift Left/Right)

S	I/D	DESCRIPTION
H	H	Entire display shift left by 1
H	L	Entire display shift right by 1

- Display Control

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

Controls display, cursor and blink ON/OFF.
D: Display ON/OFF control bit
When D = "1", display ON
When D = "0", display OFF, the content of DDRAM is not changed
C: Cursor ON/OFF control bit
When C = "1", cursor ON.
When C = "0", cursor OFF.
B: Character Blink ON/OFF control bit
When B = "1", cursor position blink ON. Then display data (character) in cursor position will blink.
When B = "0", cursor position blink OFF

- Cursor/Display Shift Control

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

This instruction configures the cursor moving direction or the display shifting direction. The content of DDRAM is not changed.

S/C	R/L	Description	AC Value
L	L	Cursor moves left by 1 position	AC=AC-1
L	H	Cursor moves right by 1 position	AC=AC+1
H	L	Display shift left by 1, cursor also follows to shift.	AC=AC
H	H	Display shift right by 1, cursor also follows to shift.	AC=AC

- Function Set

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

DL: 4/8-bit interface control bit
When DL = "1", 8-bit MPU bus interface
When DL = "0", 4-bit MPU bus interface
RE: extended instruction set control bit
When RE = "1", extended instruction set
When RE = "0", basic instruction set

In same instruction cannot alter DL and RE at once. Make sure that change DL first then RE.

- Set CGRAM Address

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

Set CGRAM address into address counter (AC)
AC range is 00H…3FH
Make sure that in extended instruction SR=0 (scroll address or RAM address select)

- Set DDRAM Address

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

Set DDRAM address into address counter (AC).
First line AC range is 80H…8FH
Second line AC range is 90H…9FH
Third line AC range is A0H…AFH
Fourth line AC range is B0H…BFH
Please note that only 2 lines can be display with one ST7920.

- Read Busy Flag (BF) and Address

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

Read busy flag (BF) can check whether the internal operation is finished or not. At the same time, the value of address counter (AC) is also read. When BF = “1”, further instruction(s) will not be accepted until BF = “0”.

- Write Data to RAM

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

Write data to the internal RAM and increase/decrease the (AC) by 1
Each RAM address (CGRAM, DDRAM and GDRAM…) must write 2 consecutive bytes for 16-bit data. After receiving the second byte, the address counter will increase or decrease by 1 according to the entry mode set control bit.

- Read RAM Data

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

Read data from the internal RAM and increase/decrease the (AC) by 1
After the operation mode changed to Read (CGRAM, DDRAM and GDRAM…), a “Dummy Read” is required. There is no need to add a “Dummy Read” for the following bytes unless a new address set instruction is issued.

Description of extended instruction set (RE=1)

- Standby

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

This Instruction will set ST7920 entering the standby mode. Any other instruction follows this instruction will terminate the standby mode.
The content of DDRAM remains the same.

- Vertical Scroll or RAM Address Select

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

When SR = "1", the Vertical Scroll mode is enabled.
When SR = "0", “Set CGRAM Address” instruction **(basic instruction)** is enabled.

- Reverse/Highlight

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

Select 1 out of 4 lines to reverse the display and to toggle the reverse condition by repeating this instruction.
R1, R0 initial vale is 00. The first time issuing this instruction, the display will be reversed while the second time will return the display become normal.

R1	R0	Description
L	L	First line normal or reverse
L	H	Second line normal or reverse
H	L	Third line normal or reverse
H	H	Fourth line normal or reverse

Please note that only 2 lines out of 4 lines of display data can be displayed with one ST7920, so It remains like next table.

R1	R0	Description
L	L	First, Third lines normal or reverse
L	H	Second, Fourth lines normal or reverse

- Sleep

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

SL: sleep mode control bit
When SL = “1”, exit sleep mode.
When SL = “0”, go to sleep mode.

- Extended Function Set

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

DL: 4/8-bit interface control bit
When DL = "1", 8-bit MPU interface.
When DL = "0", 4-bit MPU interface.
RE: extended instruction set control bit
When RE = "1", extended instruction set
When RE = "0", basic instruction set
G: Graphic display control bit
When G = "1", Graphic Display ON
When G = "0", Graphic Display OFF

In same instruction cannot alter DL, RE and G at once. Make sure that change DL or G first and then RE.

- Set Scroll Address

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

SR=1: AC5~AC0 is vertical scroll displacement address

- Set Graphic RAM Address

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

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

Set GDRAM address into address counter (AC). This is a 2-byte instruction.
The first instruction sets the vertical address while the second one sets the horizontal address (write 2 consecutive bytes to complete the vertical and horizontal address setting).
Vertical address range is AC5...AC0
Horizontal address range is AC3…AC0
The address counter (AC) of graphic RAM (GRAM) will be increased automatically after the vertical and horizontal addresses are set. After horizontal address is increased upto 0FH, it will automatically return to 00H. However, the vertical address will not increase as the result of the same action.

Display Data RAM (DDRAM)

There are 64x2 bytes RAM spaces for the Display Data RAM. It can store display data such as 16 characters (16x16) by 4 lines or 32 characters (8x16) by 4 lines. However, only 2 character-lines (maximum 32 common outputs) can be displayed at one time. Character codes stored in DDRAM will refer to the fonts specified by CGROM, HCGROM and CGRAM.

ST7920 can display half-width HCGROM fonts, user-defined CGRAM fonts and full 16x16 CGROM fonts. The character codes in 0000H~0006H will use user-defined fonts in CGRAM. The character codes in 02H~7FH will use half-width alpha numeric fonts. The character code larger than A1H will be treated as 16x16 fonts and will be combined with the next byte automatically. The 16x16 BIG5 fonts are stored in A140H~D75FH while the 16x16 GB fonts are stored in A1A0H~F7FFH. In short:

1. To display HCGROM fonts:
   Write 2 bytes of data into DDRAM to display two 8x16 fonts. Each byte represents 1 character.
   The data is among 02H~7FH.
2. To display CGRAM fonts:
   Write 2 bytes of data into DDRAM to display one 16x16 font.
   Only 0000H, 0002H, 0004H and 0006H are acceptable.
3. To display CGROM fonts:
   Write 2 bytes of data into DDRAM to display one 16x16 font.
   A140H~D75FH are BIG5 code, A1A0H~F7FFH are GB code.

The higher byte (D15~D8) is written first and the lower byte (D7~D0) is the next.

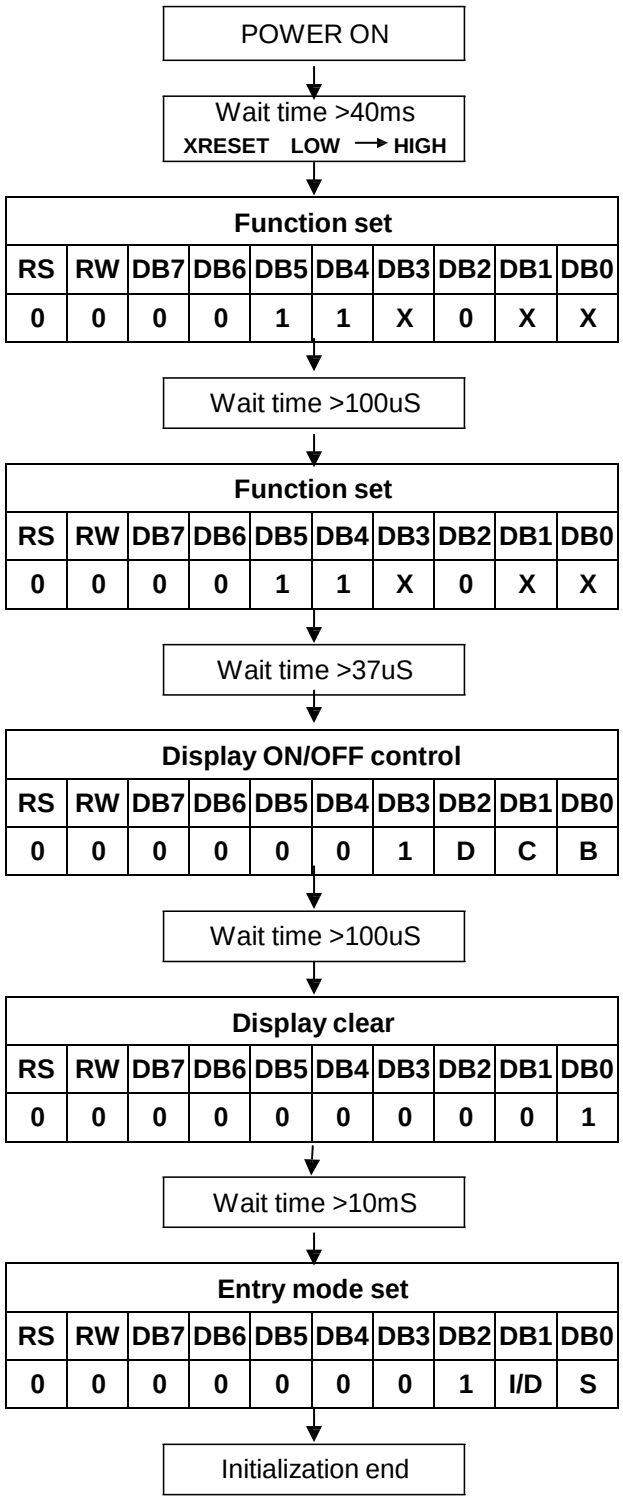
Graphic RAM (GDRAM)

Graphic Display RAM has 64x256 bits bit-mapped memory space. GDRAM address is set by writing 2 consecutive bytes of vertical address and horizontal address. Two-byte data (16 bits) configures one GDRAM horizontal address. The Address Counter (AC) will be increased by one automatically after receiving the 16-bit data for the next operation. After the horizontal address reaching 0FH, the horizontal address will be set to 00H and the vertical address will not change. The procedure is summarized below:

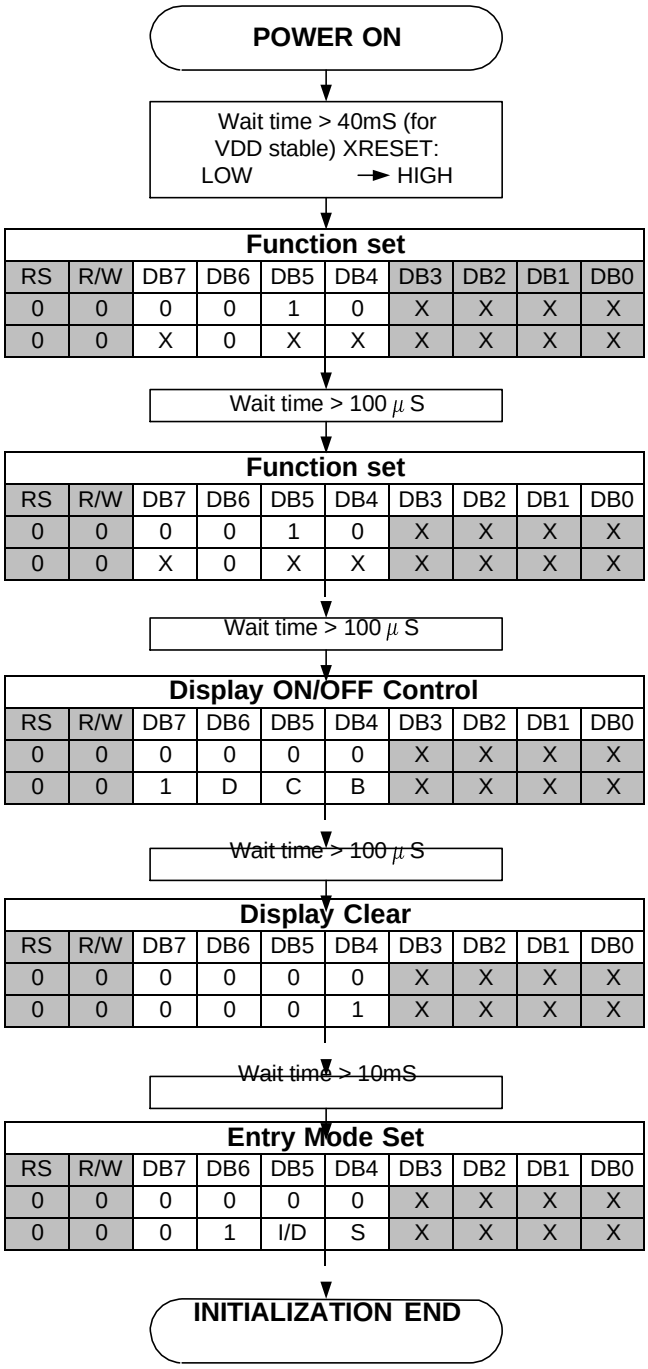
1. Set vertical address (Y) for GDRAM
2. Set horizontal address (X) for GDRAM
3. Write D15~D8 to GDRAM (first byte)
4. Write D7~D0 to GDRAM (second byte)

Initialization Timing

8-bit interface:



4-bit interface:



Application Examples

```
//Shenzhen painted crystal HJ12864 with M Series (controller ST7920A), SCM: 89S52, Crystal: 12M.
//Parallel connection, P3.1-RS,P3.4-RW,P3.5-E
//Design: BO LIANG
#include<reg52.h>
#include <intrins.h>
sbit RS=P3^1;           //serial CS
sbit RW=P3^4;           //serial port for the SID
sbit E=P3^5;            //serial clock SCLK
//sbit stop=P3^2;
sbit PSB=P2^3;
sbit REST=P2^4;

//The following is <at89x51.h> header file definition
/*
#define RS     P2_0
#define RW     P2_1    //define pin
#define E      P2_2
#define PSB P2_3
#define REST P2_4
#define Data   P1
#include<at89x51.h>
*/

#define BF     0x80    //is used to detect LCM status word in the Busy logo
typedef unsigned int Uint;
typedef unsigned char Uchar;
//String Examples
//"F1--English", can also be entered, write the character code, a character from two yards
const Uchar F1English[]={0x46,0x31,0x2d,0x2d,0x45,0x6e,0x67,0x6c,0x69,0x73,0x68,0x00};
const Uchar lengthF1=6;         //length of the string
//Characters, can be written directly shaped
unsigned char code uctech[] = {"绘晶科技有限公司"}; //"painted Crystal Technology Co., Ltd."
const Uchar lengthCF3=8;

Uchar code TAB1[]={
/*--    Transferred to the image: C:\Documents and Settings\Administrator\Desktop\12864.bmp  --*/
/*--    Width x height = 128x64          --*/
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF F,0xFF,0xFF,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
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,0xF8,0x00,0x00,0x01,
```

0xBF,0xFF,0xFF,0xFF,0xF8,0x00,0x32,0x00,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xF8,0x00,0x22,0x00,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xF8,0x00,0x33,0x00,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xF8,0x00,0x63,0x00,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFC,0x00,0x61,0x80,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFC,0x00,0xE1,0x80,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFC,0x00,0xE1,0xC0,0x00,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFE,0x00,0xE1,0xC0,0x01,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFE,0x01,0xE1,0xC0,0x01,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFE,0x01,0xF1,0xE0,0x03,0xFF,0xFF,0xFF,0xF8,0x00,0x02,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0x03,0xE3,0xE0,0x02,0xFF,0xFF,0xFF,0xF8,0x01,0xFF,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0x03,0xE1,0xF0,0x07,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0x4B,0xF1,0xF8,0x06,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF1,0xF8,0x0F,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF5,0xFC,0x1F,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFD,0x3F,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xDF,0xFF,0xFF,0xFF,0xF8,0x00,0x9C,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xCF,0xFB,0xFF,0x00,0x7E,0xFF,0xFF,0xF8,0x01,0xF4,0x01,
0xBF,0xFF,0xFF,0xDE,0x10,0x8F,0xF9,0xFF,0x01,0x2B,0x39,0xFF,0xF8,0x01,0xB4,0x01,
0xBF,0xFF,0xFC,0x42,0xF0,0x1F,0xFD,0xFF,0x80,0x44,0x0A,0xFF,0xF8,0x01,0xBC,0x01,
0xBF,0xFF,0xFD,0x07,0x20,0x3F,0xE9,0xFF,0xC0,0x41,0x45,0xFF,0xF8,0x01,0xB4,0x01,
0xBF,0xFF,0xFE,0xED,0x5A,0x8B,0xC0,0xFF,0xC0,0x05,0xDD,0xFF,0xF8,0x01,0xF4,0x01,
0xBF,0xFF,0xFD,0xB0,0xF4,0x20,0x01,0x00,0x62,0xF7,0xFF,0xFF,0xF8,0x00,0x95,0x01,
0xBF,0xFF,0xFF,0xFF,0xB4,0x0F,0xF9,0x00,0x23,0x7F,0xFF,0xFF,0xF8,0x00,0xB7,0x01,
0xBF,0xFF,0xFF,0xFC,0x14,0x83,0xFF,0xF9,0x3C,0x08,0xFF,0xFF,0xF8,0x00,0x80,0x01,
0xBF,0xFF,0xFF,0xE6,0x56,0x81,0xFF,0xFF,0xFD,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0x00,0xFF,0xFF,0xFA,0x7F,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xDA,0x87,0xFF,0xFF,0xDA,0x3F,0xFF,0xFF,0xF8,0x00,0xFC,0x01,
0xBF,0xFF,0xFF,0xEE,0xFF,0x02,0xFD,0x7F,0xF9,0x7F,0xFF,0xFF,0xF8,0x00,0x84,0x01,
0xBF,0xFF,0xFF,0xFF,0xF8,0xC0,0x01,0x02,0xDA,0xFF,0xFF,0xFF,0xF8,0x00,0xCC,0x01,
0xBF,0xFF,0xFF,0xFF,0xF6,0x11,0x80,0xC0,0x1A,0x3F,0xFF,0xFF,0xF8,0x00,0xB4,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0x0E,0xC1,0xF9,0x41,0xFF,0xFF,0xFF,0xF8,0x00,0xB4,0x01,
0xBF,0xFF,0xFF,0xF8,0xA2,0xAF,0xF0,0xFF,0x93,0xFF,0xFF,0xFF,0xF8,0x00,0xCC,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0x8F,0xF8,0xFF,0x2E,0xBF,0xFF,0xFF,0xF8,0x00,0x85,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0x57,0xF8,0xFF,0x18,0xFF,0xFF,0xFF,0xF8,0x01,0x03,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0x07,0xF1,0xFE,0x47,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xE3,0xF9,0xFE,0x3F,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0x83,0xF1,0xF9,0x7F,0xFF,0xFF,0xFF,0xF8,0x00,0x00,0x01,
0xBF,0xFF,0xFF,0xFF,0xFA,0xC1,0xE1,0xF8,0x7F,0xFF,0xFF,0xFF,0xF8,0x00,0x02,0x01,
0xBF,0xFF,0xFF,0xFF,0xF9,0x00,0xF1,0xF0,0x34,0xFF,0xFF,0xFF,0xF8,0x01,0x5F,0x01,
0xBF,0xFF,0xFF,0xFF,0xF7,0x2C,0xE1,0xE4,0xFF,0xFF,0xFF,0xFF,0xF8,0x01,0xCC,0x01,
0xBF,0xFF,0xFF,0xFF,0xFF,0xD0,0x61,0xC0,0x7F,0xFF,0xFF,0xFF,0xF8,0x01,0x5E,0x01,
0xBF,0xFF,0xFF,0xFF,0xFE,0xAC,0x61,0xC1,0x67,0xFF,0xFF,0xFF,0xF8,0x01,0x5A,0x01,
0xBF,0xFF,0xFF,0xFF,0xFE,0x80,0x21,0xC0,0x3F,0xFF,0xFF,0xFF,0xF8,0x01,0x5A,0x01,
0xBF,0xFF,0xFF,0xFF,0xFD,0xC0,0x21,0x00,0x07,0xFF,0xFF,0xFF,0xF8,0x01,0xDA,0x01,
0xBF,0xFF,0xFF,0xFF,0xFA,0x60,0x01,0x00,0x13,0xFF,0xFF,0xFF,0xF8,0x03,0x4C,0x01,
0xBF,0xFF,0xFF,0xFF,0xFC,0xD0,0x00,0x00,0x1F,0xFF,0xFF,0xFF,0xF8,0x02,0x52,0x01,
0xBF,0xFF,0xFF,0xFF,0xE1,0x00,0x00,0x00,0x08,0xD7,0xFF,0xFF,0xF8,0x00,0x21,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF F,0xFF,0xFF,};

Uchar code TAB2[]={
/*-- Transferred to the image: C:\Documents and Settings\Administrator\Desktop\12864 border.bmp --*/
/*-- Width x height = 128x64--*/
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF F,0xFF,0xFF,

```
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
```

```
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF F,0xFF,0xFF,};



//This is the delay between instructions and data when the serial
/*
void delay10US(Uchar x)
{
        Uchar k;
        for(k=0;k<x;k++);
}
*/
const Uchar delay=250;          //delay time constant
static void Wait1ms(void)       //delay1 ms
{
        Uchar cnt=0;
        while (cnt<delay) cnt++;
}
//Delay n ms
void WaitNms(int n)
{
        Uchar i;
        for(i=1;i<=n;i++)
        Wait1ms();
}

void ini_int1(void)
{ EA=1;
EX0=1;          //allow external INT0 interrupt
IT0=1;          //allow interrupts
}

int scankey1() interrupt 2 using 3 //use the external interrupt 1, the register group
{
while(P3^2==0){for(;;);}
        IE1=0;  //interrupt flag is cleared
}

//*****************************************************************
********//
//The following is a parallel port only open
//Read busy flag
void RDBF(void)
{
        Uchar temp;
        RS=0;   // RS=0
        RW=1;  // RW=1
```

```
        while(1)
        {
                P1=0xFF;          //data line input
                E=1;
                temp=P1;
                E=0;     // E=0
                if ((temp & 0x80)==0) break;
        }
}
//Write data to the instruction register

void WRCommand(Uchar comm)
{
        RDBF();
        RS=0;
        RW=0;
        P1=comm;
        E=1;
        E=0;
}
/ / Write data to the data register

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


////////////////////////////////////////////////////////////////////////////
//Open the serial read and write timing
/*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 to simulate the serial communication, so, coupled with 89S52 delay
}
void WRData(Uchar Data)
{
        RS=0;
        RS=1;
        SPIWR(Data,1);
}
*/
/*********************************************************************
/
//Initialize LCD-8-bit interface

void LCDInit(void)
{       //PSB=0;        //serial port
        PSB=1;          //parallel port selected on a line to cancel
        REST=1;
        REST=0;
        REST=1;
        WRCommand(0x30);     // basic instruction set, 8-bit parallel
        WRCommand(0x06);    // start point is set: the cursor to the right
        WRCommand(0x01);    // Clear display DDRAM
        WRCommand(0x0C);    //display the status switch: Overall open, the cursor display off, cursor display Anti-
white Off

WRCommand(0x02);            //address of zero
}
        //Array of string (semi-width font 16 * 8 dot matrix)
        void ShowQQChar(Uchar addr,Uchar *english,Uchar count)
        {
                Uchar i;
                WRCommand(addr);      // set DDRAM address
                for(i=0;i<count;)
                {
                        WRData(english[i*2]);
                        WRData(english[i*2+1]);
                        i++;
                }
        }
        //Display a continuous string (half-width characters)
        void ShowNUMChar(Uchar addr,Uchar i,Uchar count)
        {
                Uchar j;
                for(j=0;j<count;)
                {
                        WRCommand(addr);     // set DDRAM address
                        WRData(i+j);
                        j++;
                        WRData(i+j);
```

```
                addr++;
                j++;
            }
        }

        // Custom character written to CGRAM
        void WRCGRAM(Uchar data1,Uchar data2,Uchar addr)
        {
        Uchar i;
        for(i=0;i<16;)
        {
            WRCommand(addr+i);          // set CGRAM address
            WRData(data1);
            WRData(data1);
            i++;
            WRCommand(addr+i);          // set CGRAM address
            WRData(data2);
            WRData(data2);
            i++;
        }
}
/ / Display the custom character, and character to fill the full screen 16 * 16
void ShowCGChar(Uchar addr,Uchar i)
{
        Uchar j;
        for(j=0;j<0x20;)
        {
            WRCommand(addr+j);          // set DDRAM address
            WRData(0x00);
            WRData(i);
            j++;
        }
}
void CLEARGDRAM(void)
{
        Uchar j;
        Uchar i;
        WRCommand(0x34);
        WRCommand(0x36);
        for(j=0;j<32;j++)
        {
            WRCommand(0x80+j);
            WRCommand(0x80);    // X coordinates
            for(i=0;i<32;i++)
            {
                WRData(0x00);
            }
        }
}
```

/ / Write the graphics GDRAM Y Y coordinates of the drawing, two bytes of a line, CLONG graphic length, in bytes
/ / Units; HIGHT is the height of the graph, the TAB is a graphical display of graphical data table .12864 M is equivalent to 256 * 32 dot matrix.
/ / By the two-screen 128 * 32 upper and lower screen, the screen does not address the head address of the next screen in the same line immediately.
/ / In the serial input, the drawing will be under the parallel port input slower.

```
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
        {                                   // The first half on the screen
          WRCommand(Y1+j);                  // Y coordinates, the first few lines of
          WRCommand(0x80);                  // X coordinate, the horizontal number of the first few bytes to write from the
          for(i=0;i<clong;i++)               //
          {
                WRData(TAB1[clong*j+i]);
        }
                                            // The second half of screen
        for(k=0;k<clong;k++)//
        {
                WRData(TAB1[clong*(j+hight)+k]);
        }
}
}


void menu(void)
{
LCDInit();

ShowQQChar(0x80,uctech,lengthCF3);          //display 'painted Crystal Technology Co., Ltd.', the following
                                            //A total of four lines
ShowQQChar(0x90,uctech,lengthCF3);
ShowQQChar(0x88,uctech,lengthCF3);
ShowQQChar(0x98,uctech,lengthCF3);

//WRGDRAM(0x80,16,32,TAB2);
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                // Clear display DDRAM



ShowNUMChar(0x80,0x01,0x0f);        //display the special symbols of the half-width
ShowNUMChar(0x90,0x30,0x0f);        //display the half-width 0-? Digital punctuation
ShowNUMChar(0x88,0x41,0x0f);        //half width A-P uppercase
ShowNUMChar(0x98,0x61,0x0f);        //display the half-width a-p lowercase
```

```
WaitNms(250);                   //Wait time
WaitNms(250);                   //Wait time
        //stopint();
WRCommand(0x01);                //Clear display DDRAM

WRCGRAM(0xff,0x00,0x40);    //write the cross
WRCGRAM(0x00,0xff,0x50);    //write the cross2
WRCGRAM(0xaa,0xaa,0x60); //write the vertical
WRCGRAM(0x55,0x55,0x70); //write the vertical2
ShowCGChar(0x80,0x00);      //display the cross and fill
WaitNms(250);                   //Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                // Clear display DDRAM

ShowCGChar(0x80,02);            // display the cross and fill
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
//stopint();
WRCommand(0x01);                // Clear display DDRAM

ShowCGChar(0x80,04);            // show the vertical and fill
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                 // Clear display DDRAM

ShowCGChar(0x80,06);            // display vertical and fill
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                // Clear display DDRAM

WRCGRAM(0x00,0x00,0x40); //clear CGRAM1
WRCGRAM(0x00,0x00,0x50); //clear CGRAM2
WRCGRAM(0xaa,0x55,0x40); // write point
WRCGRAM(0x55,0xaa,0x50); // write point 2
ShowCGChar(0x80,00);            //display point and fill
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                // Clear display DDRAM

ShowCGChar(0x80,02);            // display point and fill
WaitNms(250);                   // Wait time
WaitNms(250);                   // Wait time
        //stopint();
WRCommand(0x01);                // Clear display DDRAM
}

void menu2(void)
{
        CLEARGDRAM();
        WRGDRAM(0x80,16,32,TAB1);
        WaitNms(250);  // Wait time
```

```
        WaitNms(250);  // Wait time
                //stopint();
}

// Main function
void main(void)
{
ini_int1();       // open interrupt
menu();           // initialize and half-width characters and points if they had Chinese characters scanning
menu2();          // graphics
for(;;)
{;}

}
```