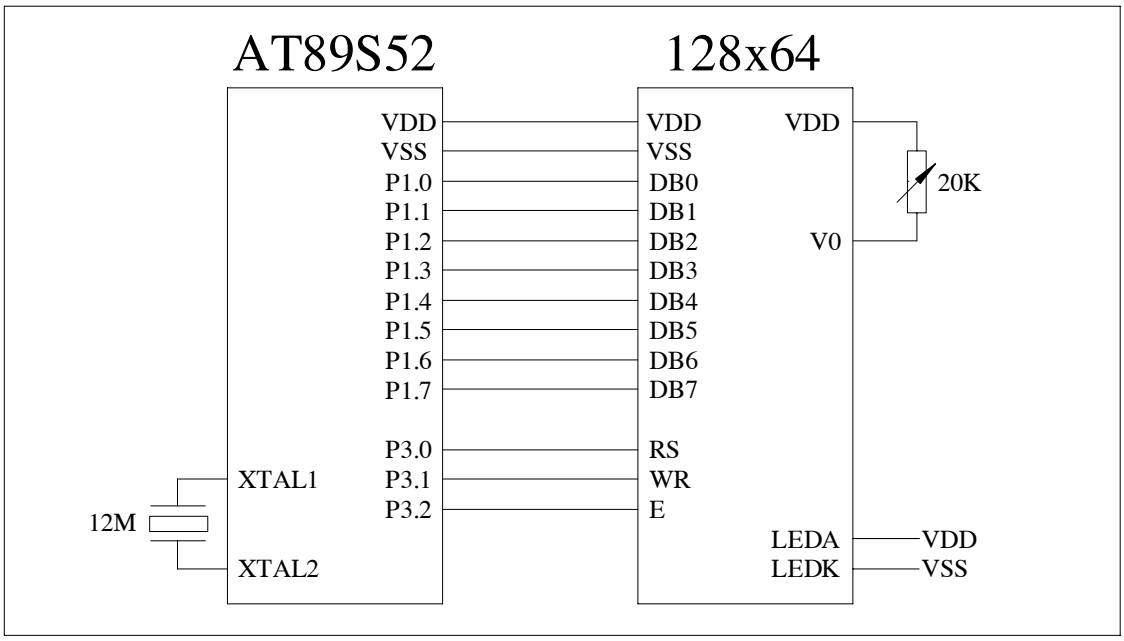
深圳秋田微电子有限公司
SHENZHEN AV-DISPLAY CO.,LTD.

Reference Program for LCD Modules	Version No.	A00
	Type	COB
Remark:	128x64 character and graphic dot matrix series, with ST7920 or compatible IC	

1. Interface



2. Instruction Code

Instruction set 1: (RE=0: basic instruction)

Ins	code										Description	Exec time (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
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
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	72us
ENTRY MODE	0	0	0	0	0	0	0	1	I/D	S	Set cursor position and display shift when doing write or read operation	72us
DISPLAY ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1: display ON C=1: cursor ON B=1: 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 ADDR.	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 ADDR.	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）& ADDR.	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		
STAND BY	0	0	0	0	0	0	0	0	0	1	Enter stand by mode, any other instruction can terminate (Com1..32 halted)	72 us
SCROLL or RAM ADDR. 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	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 00**	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 IRAM or SCROLL ADDR	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR=1: AC5~AC0 the address of vertical scroll	72 us
SET GRAPHIC RAM ADDR.	0	0	1	0 0	0 AC5	0 AC4	AC3 AC3	AC2 AC2	AC1 AC1	AC0 AC0	Set GDRAM address to address counter（AC） First set vertical address and the horizontal address by consecutive writing Vertical address range AC5...AC0 Horizontal address range AC3…AC0	72 us

Note：

1. Make sure that ST7920 is not in busy state by reading the busy flag before sending instruction or data. If use 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. Each time when altering the value of RE it will remain. There is no need to set RE every time when using the same group of instruction set.

Remark: For the detail instruction for the control function, please refer to the related manual data book for controller IC.

3. Reference Program

```
//====== 89S52 MCU, 12M Oscillator ===========
#include <reg51.h>
#include <intrins.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define uint unsigned int
#define uchar unsigned char
#define xchar unsigned char code

sbit  RS    = P3^4;
sbit  RRW   = P3^2;
sbit  E     = P3^3;
sbit  BF    = ACC^7;
xchar xshm[1024]={
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
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,
0x80,0x00,0x60,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x18,0x70,0x00,0x00,0x00,0x00,0x06,0x06,0x00,0x01,0x80,0x00,0x00,0x00,0x01,
0x80,0xFC,0x70,0x00,0x00,0x00,0x03,0x87,0x07,0x00,0x01,0xC0,0x00,0x00,0x00,0xC1,
0x87,0xC0,0x70,0x03,0x00,0x03,0x83,0x87,0x07,0x00,0x01,0xC0,0x01,0xFF,0xFF,0xE1,
0x80,0xC0,0x70,0x03,0xFF,0xFF,0x83,0x3F,0xE6,0x00,0x01,0xC0,0x00,0x00,0x03,0xC1,
0x80,0xC0,0x70,0x03,0x03,0x03,0x86,0x1F,0xE6,0x00,0x01,0xC0,0x00,0x00,0x07,0x01,
0x80,0xC1,0xF1,0xC3,0x03,0x03,0x8C,0x1F,0xEE,0x01,0xC1,0xC1,0xC0,0x00,0x0C,0x01,
0x80,0xDD,0xF3,0x83,0x03,0x03,0x98,0xDF,0xEC,0x31,0xFF,0xFF,0xC0,0x00,0x38,0x01,
0x9F,0xFF,0x77,0x03,0x03,0x03,0x81,0xFF,0xEF,0xF9,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x80,0xC3,0x76,0x03,0x03,0x03,0x81,0xCD,0xCC,0xE1,0xC1,0xC1,0xC0,0x00,0xF0,0x01,
0x81,0xC7,0x7C,0x03,0x03,0x03,0x83,0x80,0x1C,0xC1,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x81,0xC7,0x78,0x03,0x03,0x03,0x83,0x80,0xDC,0xC1,0xC1,0xC1,0xC0,0x00,0xE0,0x79,
0x83,0xF0,0x78,0x03,0xFF,0xFF,0x87,0x9F,0xBC,0xC1,0xFF,0xFF,0xC7,0xFF,0xFF,0xFD,
```

```
0x83,0xF8,0x78,0x03,0x03,0x03,0x87,0x80,0x3C,0xC1,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x87,0xDC,0x78,0x03,0x03,0x03,0x8D,0x98,0x07,0xC1,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x86,0xCC,0xEC,0x03,0x03,0x03,0x99,0x9F,0xC7,0xC1,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x8C,0xC0,0xEC,0x03,0x03,0x03,0x81,0x9D,0xC7,0x81,0xC1,0xC1,0xC0,0x00,0xE0,0x01,
0x8C,0xC0,0xCE,0x03,0x03,0x03,0x81,0x9D,0xC7,0x81,0xFF,0xFF,0xC0,0x00,0xE0,0x01,
0x98,0xC1,0xC6,0x03,0x03,0x03,0x81,0x9D,0xF7,0x81,0xC1,0xC1,0x80,0x00,0xE0,0x01,
0xB0,0xC1,0x87,0x03,0x03,0x03,0x81,0x99,0xE3,0x81,0x81,0xC0,0x30,0x00,0xE0,0x01,
0x80,0xC3,0x83,0x83,0x03,0x03,0x81,0x99,0xC7,0x80,0x01,0xC0,0x30,0x00,0xE0,0x01,
0x80,0xC7,0x03,0xC3,0xFF,0xFF,0x81,0x99,0xCD,0xC0,0x01,0xC0,0x30,0x00,0xE0,0x01,
0x80,0xCC,0x01,0xF3,0x00,0x03,0x81,0xB0,0x38,0xE0,0x01,0xC0,0x38,0x18,0xE0,0x01,
0x80,0xF8,0x00,0xC0,0x00,0x00,0x01,0xE0,0x70,0x70,0x01,0xC0,0x38,0x0F,0xC0,0x01,
0x80,0x30,0x00,0x00,0x00,0x00,0x01,0x81,0xC0,0x00,0x00,0xFF,0xF8,0x03,0xC0,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
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,0xFF,0xFF,0xFF
};

//----------------------------------------------
void delayms(uint v)
{
  while(v!=0)v--;
}

//----------------------------------------------
void delay(uint n)
{
  uint m;
  while(n-->0)
  for(m=0;m<1250;m++)
  { };
}
```

```
//-----------------------------------------
void check_busy()
{
  E=0;RRW=1;RS=0;
  do
  {
    P1=0xff;E=1;delayms(2);ACC=P1;E=0;
  }
  while(BF==1);
}

//-----------------------------------------
void wcomd(uchar cdat)
{
  check_busy();
  E=0;RRW=0;RS=0;P1=cdat;E=1;delayms(2);E=0;
}

//-----------------------------------------
void wdata(uchar ddat)
{
  check_busy();
  E=0;RRW=0;RS=1;P1=ddat;E=1;delayms(2);E=0;
}

//-----------------------------------------
void initial()
{
  wcomd(0x38);  delayms(60);
  wcomd(0x06);  delayms(60);
  wcomd(0x01);  delay(2);
  wcomd(0x0c);  delayms(60);
}

//-----------------------------------------
void disp_all(uchar dat1,uchar dat2)
{
  uchar m=0,n=0;
  wcomd(0x3c);
  wcomd(0x03);
  wcomd(0x36);
  for(m=0;m<8;m++)
  {
    wcomd(0x80+2*m);wcomd(0x80+n);
    for(n=0;n<16;n++)
    { wdata(dat1);}
    for(n=0;n<16;n++)
    { wdata(dat1);}
```

```
    wcomd(0x80+2*m+1);wcomd(0x80+n);
    for(n=0;n<16;n++)
    { wdata(dat2);}
    for(n=0;n<16;n++)
    { wdata(dat2);}
  }
  for(m=0;m<8;m++)
  {
    wcomd(0x90+2*m);wcomd(0x90+n);
    for(n=0;n<16;n++)
    { wdata(dat1);}
    for(n=0;n<16;n++)
    { wdata(dat1);}
    wcomd(0x90+2*m+1);wcomd(0x90+n);
    for(n=0;n<16;n++)
    { wdata(dat2);}
    for(n=0;n<16;n++)
    { wdata(dat2);}
  }
}

//-----------------------------------------
void disp_hz(uchar hdat1,uchar hdat2,uchar hdat3,uchar hdat4)
{
  uchar m=0,n=0;
  wcomd(0x80);
  for(n=0;n<16;n++)
  { wdata(hdat1);wdata(hdat2+n);}
  wcomd(0x90);
  for(n=0;n<16;n++)
  { wdata(hdat3);wdata(hdat4+n);}
}

//---------------------------------------
void disp_bmp(xchar *str)
{
  uchar m=0,n=0;
  w_cd(0x3c,0);
  w_cd(0x03,0);
  w_cd(0x36,0);
  for(m=0;m<16;m++)
  {
    w_cd(0x80+m,0);w_cd(0x80+n,0);
    for(n=0;n<16;n++)
    { w_cd(str[n+16*m],1);}
    for(n=0;n<16;n++)
    { w_cd(str[512+n+16*m],1);}
  }
  for(m=0;m<16;m++)
```

```
  {
    w_cd(0x90+m,0);w_cd(0x90+n,0);
    for(n=0;n<16;n++)
    { w_cd(str[256+n+16*m],1);}
    for(n=0;n<16;n++)
    { w_cd(str[768+n+16*m],1);}
  }
}

//-----------------------------------------
///////////////  MAIN  ////////////////////////
/////////////////////////////////////////////
void main()
{
  delay(2);
  while(1)
  {
    initial();
    disp_bmp(xshm);     delay(150);
    disp_all(0xff,0xff);delay(100);
    disp_all(0x00,0x00);delay(100);
    disp_all(0xaa,0x55);delay(100);
    disp_all(0x55,0xaa);delay(100);
    disp_all(0xaa,0xaa);delay(100);
    disp_all(0x55,0x55);delay(100);

    initial();
    disp_hz(0xa3,0xb0,0xa3,0xc0); delay(100);
    disp_hz(0xb0,0xa1,0xb0,0xb1); delay(100);
    disp_hz(0xb0,0xc1,0xb0,0xd1); delay(100);
  }
}
```