WINSTAR Character Application Note

Character type

Last part number “T” for Sitronix

Version 0.0

Date: 2013/03/13

WINSTAR Character Application Note

1 RECORD OF REVISION

Revision Date	Page	Contents	Editor
2013/03/13	-	New Release	Austin

2. DESCRIPTION

WINSTAR character WH utilizing CMOS Technology specially, It can display 2 lines x 8 (5 x 8 dot format) characters or 1 lines x 8 (5 x 8 or 5 x 11dot format) characters. It is ideal for multi-language application.

2.1 FUNCTION

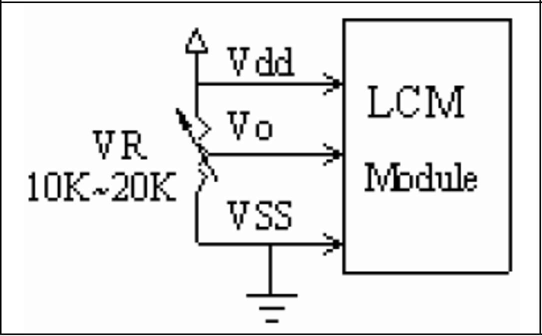
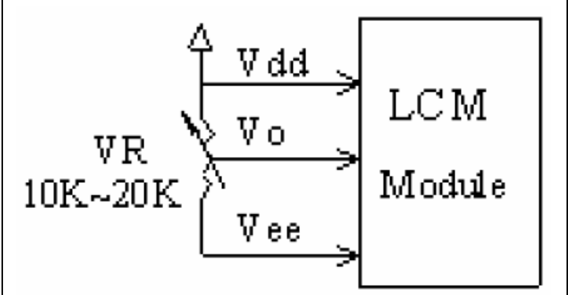
- Character type dot matrix LCD driver & controller
- Internal drivers: 16common and 40 segment for 2 line display, 1/16 duty (N=1, F=0).
  11 common and 40 segment for 1 line display, 1/11 duty (N=0, F=1).
- 8 common and 40 segment for 1 line display, 1/8 duty (N=0, F=0).
- Easy interface with 4-bit or 8-bit 68 series MPU Interface.
  5 x 8 dot matrix font or 5 x 11 dot matrix font
- Various instruction functions.
- Automatic reset circuit that initializes the controller/driver after power on

2.2 FEATURES

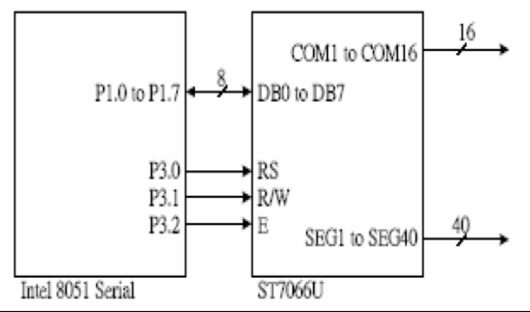
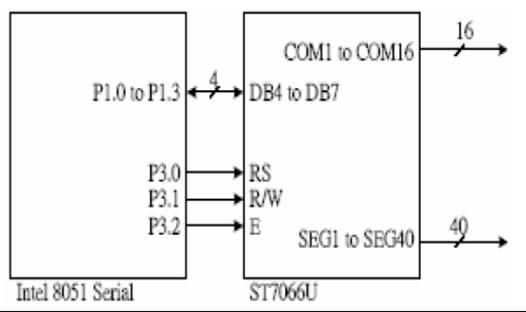
- Internal Memory
- Character Generator ROM (CGROM): 13200 bits (240 characters: 5 x 8 dot or 5 x 11 dot)
- Character Generator RAM (CGRAM): 64 x 8 bits (8 characters 5 x 8 dot or 4 characters 5 x 11)
- Display Data RAM (DDRAM): 80 x 8 bits (80 characters max).
- Power supply voltage range: 2.7 ~ 5.5 V (VDD)
- LCD Drive voltage range: 3.0 ~ 10V (V0 – VSS)
- CMOS process
- Programmable duty cycle: 1/16, 1/11, 1/8.
- Low power consumption
- QFP80 and Bare chip available

3. APPLICATION OF INPUT POWER

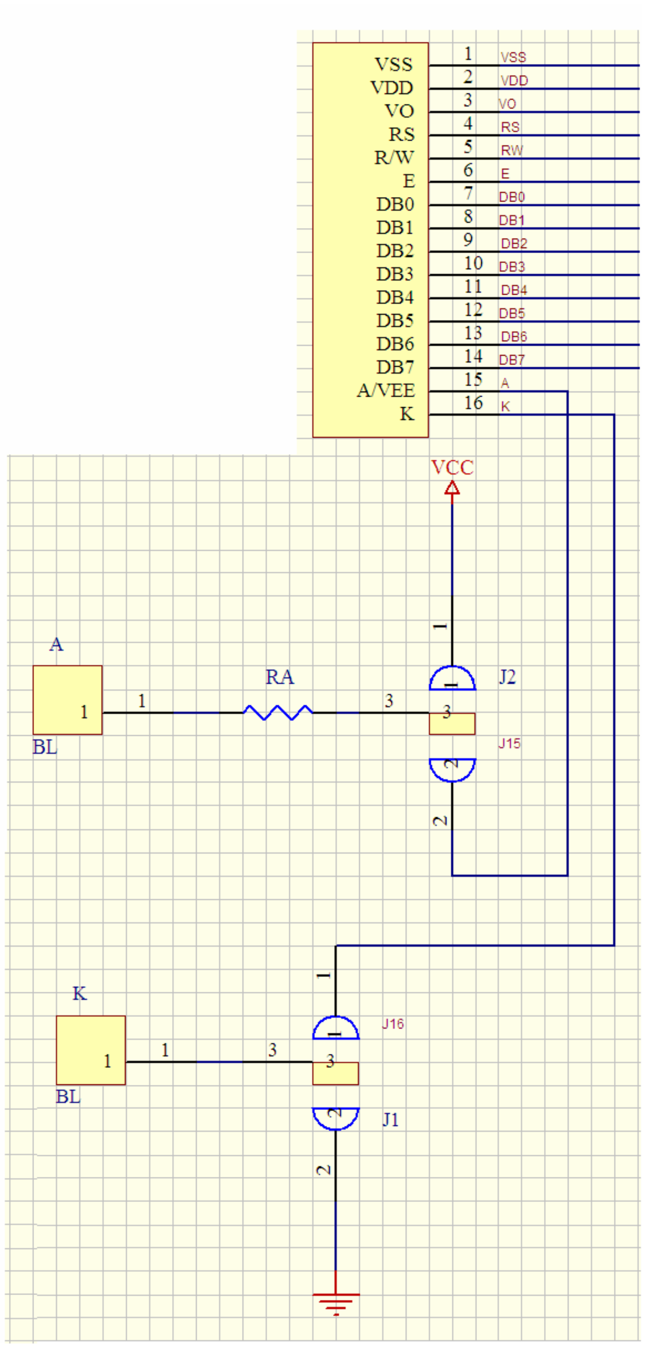
3.1 Input Power

DC 5V Input	DC 3V Input (need negative voltage)
	

3.2 How to connect

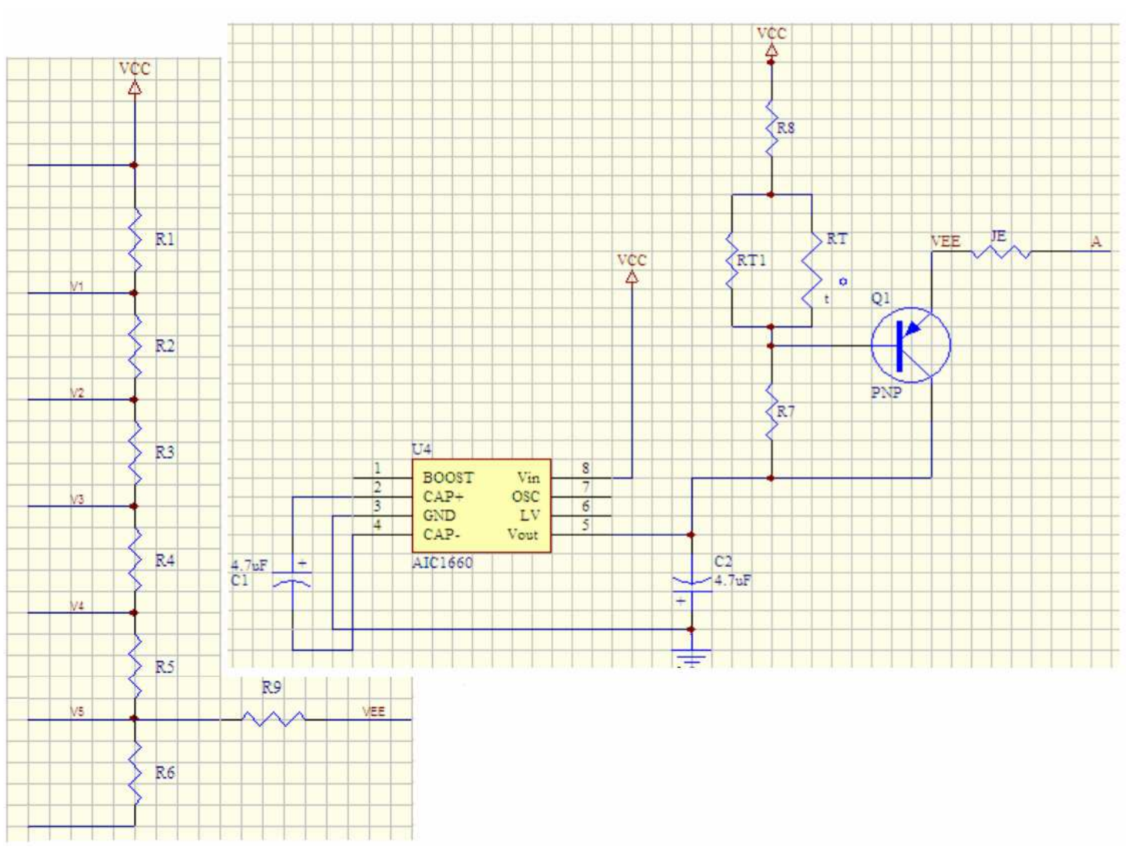
8-bit Interface	4-bit Interface
 P1.0 to P1.7 8 DB0 to DB7 COM1 to COM16 16 P3.0 P3.1 P3.2 RS R/W E SEG1 to SEG40 40 Intel 8051 Serial ST7066U	 P1.0 to P1.3 4 DB4 to DB7 COM1 to COM16 16 P3.0 P3.1 P3.2 RS R/W E SEG1 to SEG40 40 Intel 8051 Serial ST7066U

3.3 Backlight circuit



3.4 Negative circuit

When Input power is DC 3.3V, the negative circuit is needed.



4. INSTRUCTIONS

4.1 INSTRUCTIONS TABLE

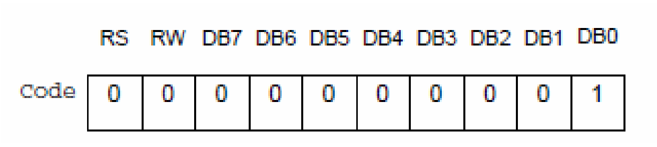
Instruction	Instruction Code										Description	Description Time (270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.52 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.52 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	37 us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	37 us
Function Set	0	0	0	0	1	DL	N	F	x	x	DL:interface data is 8/4 bits N:number of line is 2/1 F:font size is 5x11/5x8	37 us
Set CGRAM address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	37 us
Set DDRAM address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	37 us
Read Busy flag and address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 us
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	37 us
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	37 us

Note：

Be sure the ST7066U is not in the busy state (BF = 0) before sending an instruction from the MPU to the ST7066U. If an instruction is sent without checking the busy flag, the time between the first instruction and next instruction will take much longer than the instruction time itself. Refer to Instruction Table for the list of each instruction execution time.

4.2 INSTRUCTION DESCRIPTION

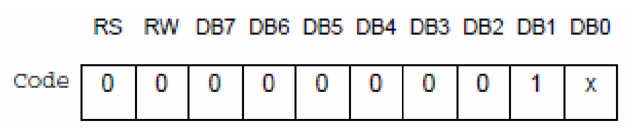
- **Clear Display**



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

Clear all the display data by writing "20H" (space code) to all DDRAM address, and set DDRAM address to "00H" into AC (address counter). Return cursor to the original status, namely, bring the cursor to the left edge on first line of the display. Make entry mode increment (I/D = "1").

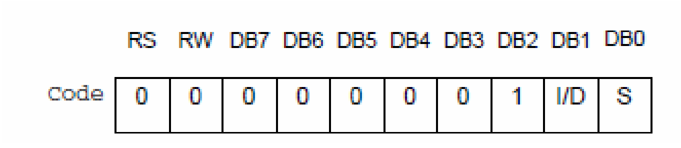
- **Return Home**



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

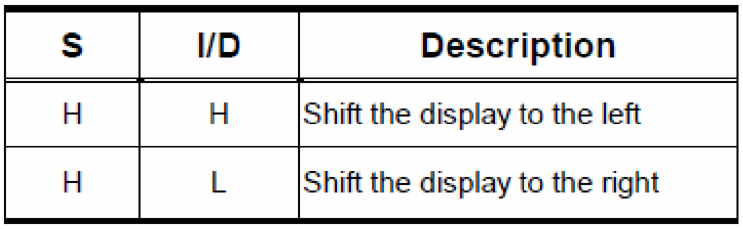
Return Home is cursor return home instruction. Set DDRAM address to "00H" into the address counter. Return cursor to its original site and return display to its original status, if shifted. Contents of DDRAM does not change.

- **Entry 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 moving direction of cursor and display.
I/D : Increment / decrement of DDRAM address (cursor or blink)
When I/D = "High", cursor/blink moves to right and DDRAM address is increased by 1.
When I/D = "Low", cursor/blink moves to left and DDRAM address is decreased by 1.
* CGRAM operates the same as DDRAM, when read from or write to CGRAM.
S: Shift of entire display
When DDRAM read (CGRAM read/write) operation or S = "Low", shift of entire display is not performed. If S = "High" and DDRAM write operation, shift of entire display is performed according to I/D value (I/D = "1" : shift left, I/D = "0" : shift right).



S	I/D	Description
H	H	Shift the display to the left
H	L	Shift the display to the right

Display ON/OFF

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

Control display/cursor/blink ON/OFF 1 bit register.

D : Display ON/OFF control bit

When D = "High", entire display is turned on.
When D = "Low", display is turned off, but display data is remained in DDRAM.

C : Cursor ON/OFF control bit

When C = "High", cursor is turned on.
When C = "Low", cursor is disappeared in current display, but I/D register remains its data.

B : Cursor Blink ON/OFF control bit

When B = "High", cursor blink is on, that performs alternate between all the high data and display character at the cursor position.
When B = "Low", blink is off.

Cursor or Display Shift

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

Without writing or reading of display data, shift right/left cursor position or display. This instruction is used to correct or search display data. During 2-line mode display, cursor moves to the 2nd line after 40th digit of 1st line. Note that display shift is performed simultaneously in all the line. When displayed data is shifted repeatedly, each line shifted individually. When display shift is performed, the contents of address counter are not changed.

S/C	R/L	Description	AC Value
L	L	Shift cursor to the left	AC=AC-1
L	H	Shift cursor to the right	AC=AC+1
H	L	Shift display to the left. Cursor follows the display shift	AC=AC
H	H	Shift display to the right. Cursor follows the display shift	AC=AC

Function Set

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	N	F	x	x

DL : Interface data length control bit
When DL = "High", it means 8-bit bus mode with MPU.
When DL = "Low", it means 4-bit bus mode with MPU. So to speak, DL is a signal to select 8-bit or 4-bit bus mode.
When 4-bit bus mode, it needs to transfer 4-bit data by two times.
N : Display line number control bit
When N = "Low", it means 1-line display mode.
When N = "High", 2-line display mode is set.
F : Display font type control bit
When F = "Low", it means 5 x 8 dots format display mode
When F = "High", 5 x11 dots format display mode.

N	F	No. of Display Lines	Character Font	Duty Factor
L	L	1	5x8	1/8
L	H	1	5x11	1/11
H	x	2	5x8	1/16

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 to AC.
This instruction makes CGRAM data available from MPU.

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 to AC.
This instruction makes DDRAM data available from MPU.
When 1-line display mode (N = 0), DDRAM address is from "00H" to "4FH".

In 2-line display mode (N = 1), DDRAM address in the 1st line is from "00H" to "27H", and DDRAM address in the 2nd line is from "40H" to "67H".

- **Read Busy Flag and Address (only support parallel 8-bit bus and 4 bit bus)**

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

When BF = "High", indicates that the internal operation is being processed.So during this time the next instruction cannot be accepted.
The address Counter (AC) stores DDRAM/CGRAM addresses, transferred from IR.
After writing into (reading from) DDRAM/CGRAM, AC is automatically increased (decreased) by 1.

- **Write Data to CGRAM or DDRAM**

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

Write binary 8-bit data to DDRAM/CGRAM.
The selection of RAM from DDRAM, CGRAM, is set by the previous address set instruction DDRAM address set, CGRAM address set. RAM set instruction can also determine the AC direction to RAM.
After write operation, the address is automatically increased/decreased by 1, according to the entry mode.

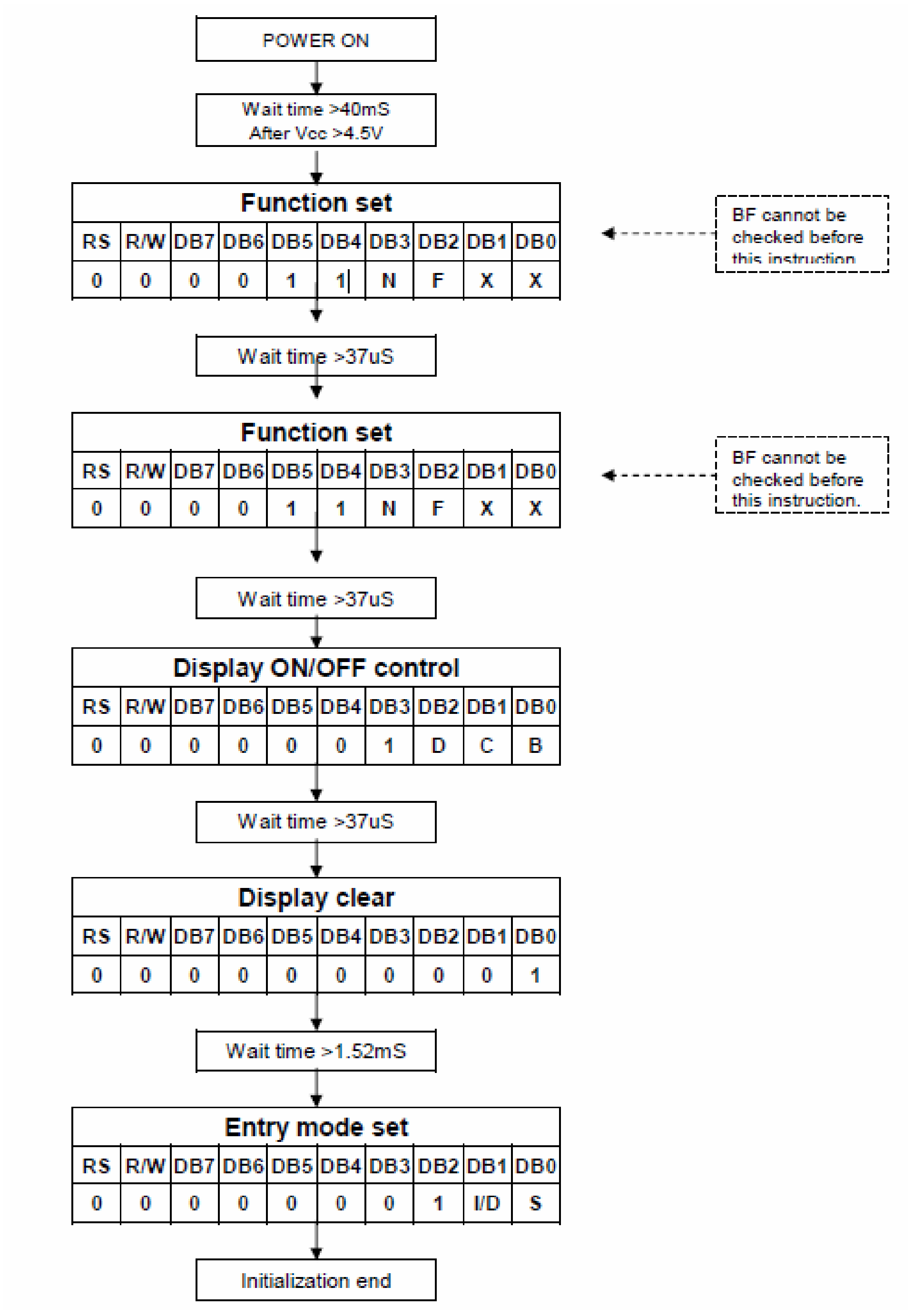
- **Read Data from CGRAM or DDRAM**

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

Read binary 8-bit data from DDRAM/CGRAM.
The selection of RAM is set by the previous address set instruction. If address set instruction of RAM is not performed before this instruction, the data that read first is invalid, because the direction of AC is not determined. If you read RAM data several times without RAM address set instruction before read operation, you can get correct RAM data from the second, but the first data would be incorrect, because there is no time margin to transfer RAM data.
In case of DDRAM read operation, cursor shift instruction plays the same role as DDRAM address set instruction : it also transfer RAM data to output data register. After read operation address counter is automatically increased/decreased by 1 according to the entry mode. After CGRAM read operation, display shift may not be executed correctly.
* In case of RAM write operation, after this AC is increased/decreased by 1 like read operation. In this time, AC indicates the next address position, but you can read only the previous data by read instruction.

5. INITIALIZATION BY INSTRUCTION

5.1 8-bit Interface



Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	N	F	X	X

Function set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	N	F	X	X

Display ON/OFF control

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

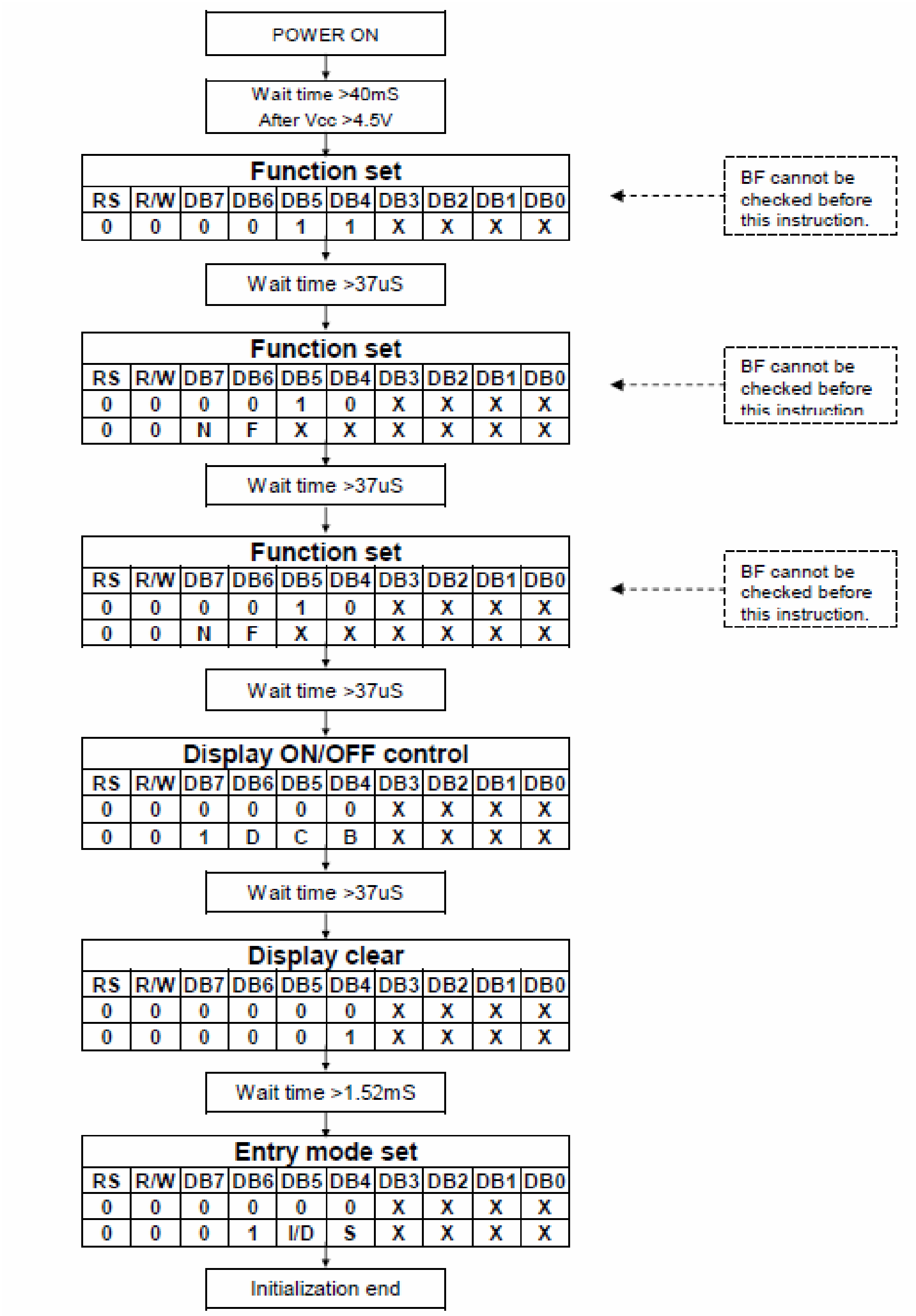
Display clear

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Entry mode set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

5.2 4-bit Interface



POWER ON

Wait time >40mS
After Vcc >4.5V

Function set									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	X	X	X	X

BF cannot be checked before this instruction.

Wait time >37uS

Function set									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	0	X	X	X	X
0	0	N	F	X	X	X	X	X	X

BF cannot be checked before this instruction

Wait time >37uS

Function set									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	0	X	X	X	X
0	0	N	F	X	X	X	X	X	X

BF cannot be checked before this instruction.

Wait time >37uS

Display ON/OFF control									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	X	X	X	X
0	0	1	D	C	B	X	X	X	X

Wait time >37uS

Display clear									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	X	X	X	X
0	0	0	0	0	1	X	X	X	X

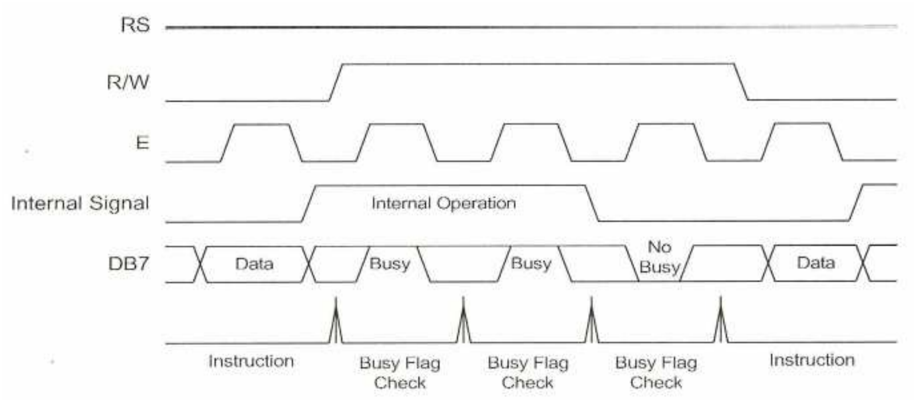
Wait time >1.52mS

Entry mode set									
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	X	X	X	X
0	0	0	1	I/D	S	X	X	X	X

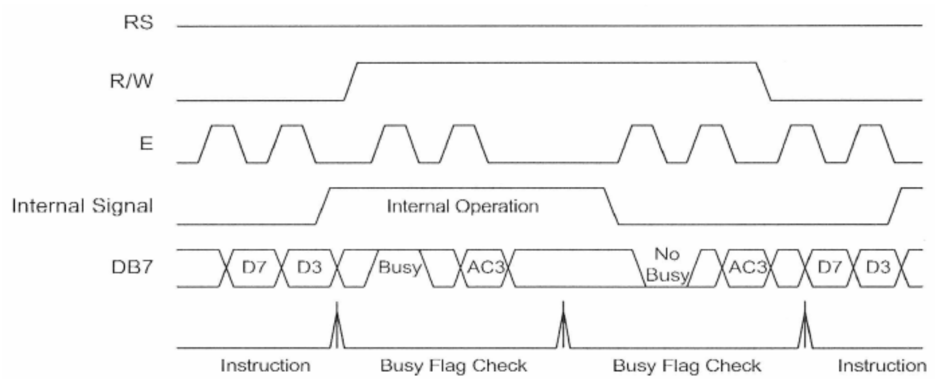
Initialization end

6 MCU INTERFACE

6.1 8-bit



6.2 4-bit



7 REFERENCE INITIAL CODE

7.1 8-bit Interface

```
#include    <reg51.h>
#include    <stdio.h>                // define I/O functions
#include    <INTRINS.H>              // KEIL FUNCTION
#define     Cword   0x10 //16
#define             one              0x80
#define             two              0xc0
#define             Data_BUS P1
sbit                busy      =P1^7;
sbit                RS         =P3^0;
sbit                RW          =P3^7;
sbit                Enable    =P3^4;

char bdata    flag;
sbit busy_f    = flag^0;


void CheckBusy();
void WriteIns(char);
void WriteData(char);
void WriteString(char,char *);
void Initial_ks0066();
void delay(char);

unsigned char code MSG1[Cword]    ="Winstar display ";
unsigned char code MSG2[Cword]    ="       WH1602W        ";
unsigned char code CGRAM1[8] ={0x04,0x0E,0x15,0x04,0x04,0x04,0x04,0x04,};    //  ↑
unsigned char code CGRAM2[8] ={0x04,0x04,0x04,0x04,0x04,0x15,0x0e,0x04,};    //  ↓

void CheckBusy(void)
{
    Data_BUS = 0xff;.
    RS = 0;
    RW = 1;
    do
    {
       Enable = 1;
       busy_f = busy;
       Enable = 0;
     }while(busy_f);
}
```

```
//=================================
void WriteIns(char instruction)
{
      RS = 0;
      RW = 0;
      Data_BUS = instruction;
      Enable = 1;                    //1us
      _nop_();                       //1us
      Enable = 0;                    //1us
      CheckBusy();
}
//=================================
//=================================
void WriteData(char data1)
{
      RS = 1;
      RW = 0;
      Data_BUS = data1;
      Enable = 1;
      _nop_();
      Enable = 0;
      CheckBusy();
}
//=================================
//=================================
void WriteString(count,MSG)
char count;
char *MSG;
{
    char i;
    for(i = 0; i<count;i++)
            WriteData(MSG[i]);
}
//=================================
//=================================
void Initial_ST66(void)
{
    WriteIns(0x38);
    WriteIns(0x38);
    WriteIns(0x0c);
    WriteIns(0x01);
    WriteIns(0x06);
}
//==========================================
//=================================
void delay(char m)
{
    unsigned char i,j,k;
      for(j = 0;j<m;j++)
    {for(k = 0; k<200;k++)
      {for(i = 0; i<200;i++)
                        {}
     }}
}
```

```
//==================================
void CGRAM(void)
{
    unsigned char i,j;
    WriteIns(0x40);

    for(i = 0;i<8;i++)
    {
            WriteData(CGRAM1[i]);
    }
    WriteIns(0x48);

    for(j = 0;j<8;j++)
    {
            WriteData(CGRAM2[j]); //data write to CGRAM
    }
}


void main(void)
{

    Initial_ST7066();
    CGRAM();                        //<== write data to CGRAM

    WriteIns(0x81);
    WriteData(0x00);
    WriteIns(0xc1);
    WriteData(0x01);
    delay(30);

    WriteIns(one);
    WriteString(Cword,MSG1);
    WriteIns(two);
    WriteString(Cword,MSG2);
    delay(30);
}
```

7.2 4-bit Interface

```
//        hi 4 bit
void WriteIns(char instruction)
{
      RS = 0;
      RW = 0;
      Data_BUS = instruction;
      Enable = 1;                    //1us
      _nop_();                      //1us
      Enable = 0;                   //1us

      instruction <<=4;

      Data_BUS = instruction;
      Enable = 1;                    //1us
      _nop_();                      //1us
      Enable = 0;                   //1us

    delay(1);
}
//================================
void WriteData(char data1)
{
      RS = 1;
      RW = 0;
      Data_BUS = data1;
      Enable = 1;
      _nop_();
      Enable = 0;

      data1 <<= 4;

      Data_BUS = data1;
      Enable = 1;
      _nop_();
      Enable = 0;
      delay(1);
      CheckBusy();
}
```

```
//================================
  void Initial ST7066()
  {
      WriteIns(0x20);
      delay(2);
      WriteIns(0x28);
      delay(2);
      WriteIns(0x28);
      delay(2);
      WriteIns(0x0c);
      delay(2);
      WriteIns(0x01);
      delay(2);
      WriteIns(0x06);
      delay(2);
  }
  //==========================================

  main()
  {
      unsigned char i;

      Initial_ST7066();

      while(1)
      {
              WriteIns(0x20);
              WriteIns(one);
              for(i = 0; i<Cword;i++)
              {
                      WriteData(MSG1[i]);
              }
              WriteIns(0x28);
              WriteIns(one);
              for(i = 0; i<Cword;i++)
              {
                      WriteData(MSG2[i]);
              }
              WriteIns(two);
              for(i = 0; i<Cword;i++)
              {
                      WriteData(MSG1[i]);
              }
      }
  }
```