WINSTAR Character Application Note

Character type

Last part number “W” for RockWorks

Confidential

Version 0.1

Date: 2013/03/12

WINSTAR Character Application Note

WINSTAR Character Application Note

1 RECORD OF REVISION

Revision Date	Page	Contents	Editor
2012/02/24	-	New Release	Lisa
2013/03/12	p.22~ p.25	Update program	Austin

2. DESCRIPTION

WINSTAR character WH1602B 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

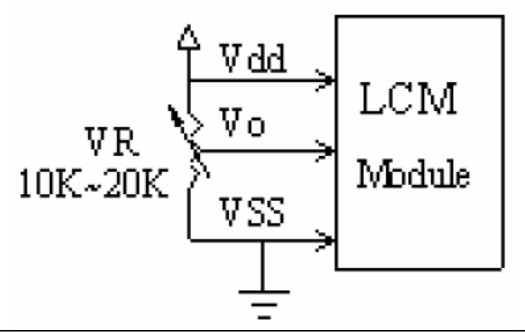
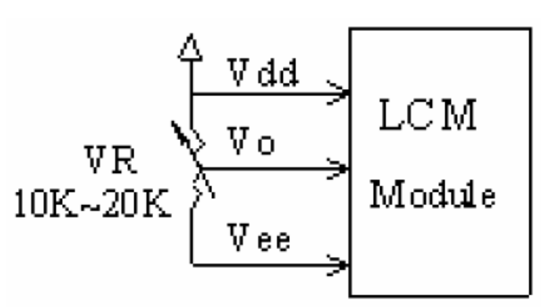
- Character type dot matrix LCD driver & controller
- Internal drivers: 16common and 40 segment for 2 line display, 1/16 duty (N=1, F=0).
  11 common and 40 segment for 1 line display, 1/11 duty (N=0, F=1).
- 8 common and 40 segment for 1 line display, 1/8 duty (N=0, F=0).
- Easy interface with 4-bit or 8-bit 68 series MPU or 4 lines SPI / IIC serial peripheral Interface.
  5 x 8 dot matrix font or 5 x 11 dot matrix font
- Various instruction functions.
- Automatic power on reset
- Double Frequency, half instruction time

2.2 FEATURES

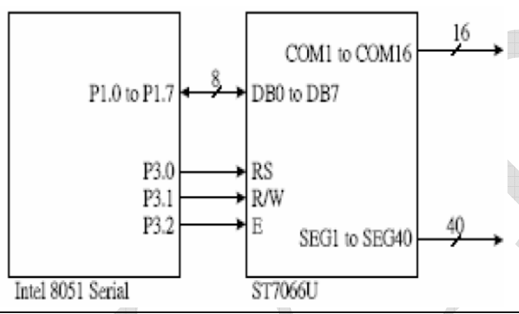
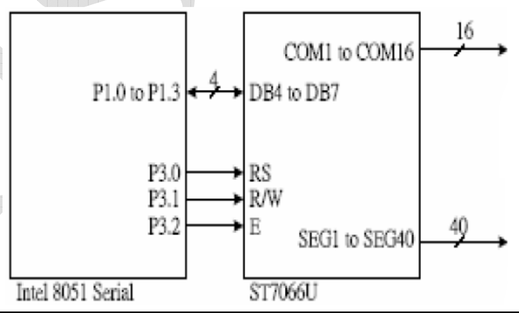
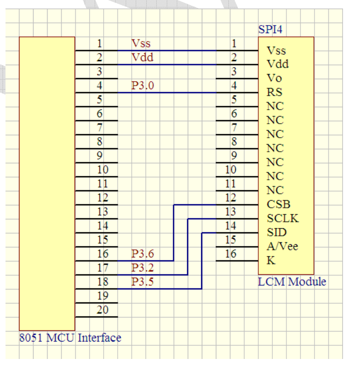
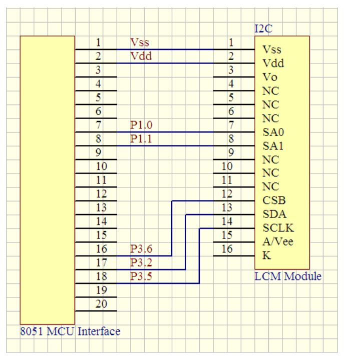
- Internal Memory
- Character Generator ROM (CGROM): 13200 bits (240 characters: 5 x 8 dot or 5 x 11 dot)
- Character Generator RAM (CGRAM): 64 x 8 bits (8 characters 5 x 8 dot or 4 characters 5 x 11)
- Display Data RAM (DDRAM): 80 x 8 bits (80 characters max).
- Power supply voltage range: 2.7 ~ 5.5 V (VDD)
- LCD Drive voltage range: 3.0 ~ 7V (V0 – VSS)
- CMOS process
- Programmable duty cycle: 1/16, 1/11, 1/8.
- Low power consumption
- Bare chip available

3. APPLICATION OF INPUT POWER

3.1 Input Power

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

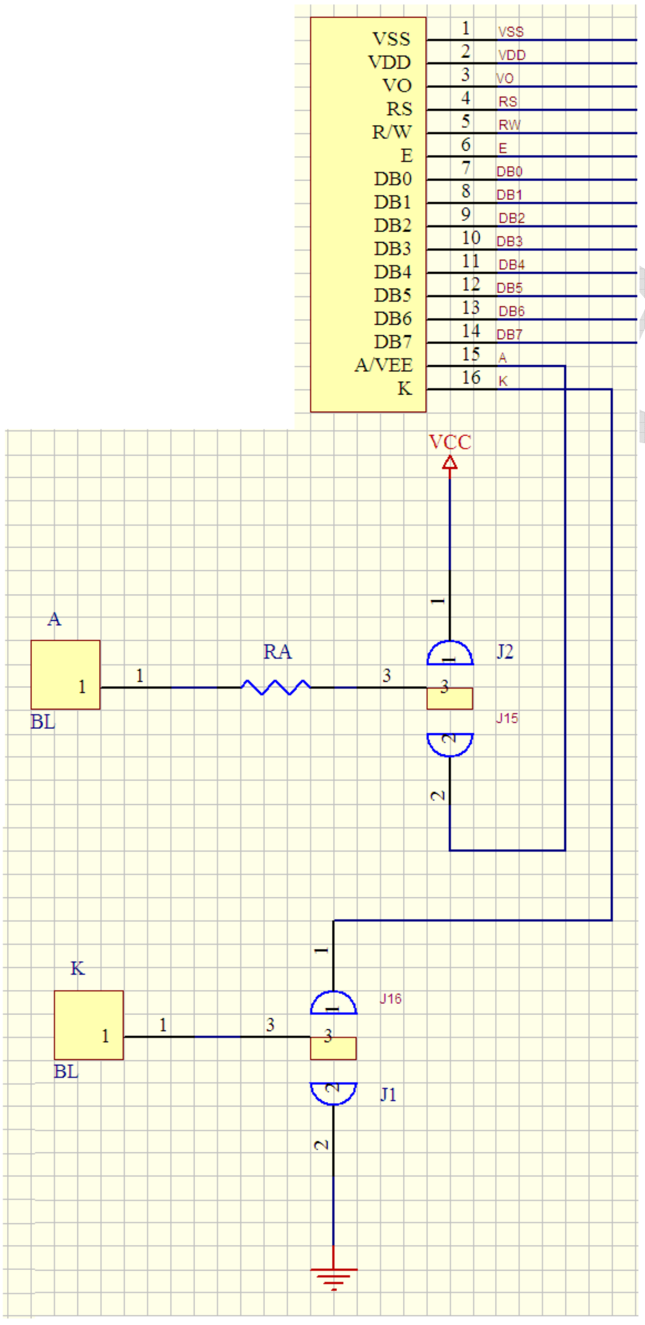
3.2 How to connect

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
SPI4	**I2C**
 SPI4 Vss Vdd P3.0 Vss Vdd Vo RS NC CSB SCLK SID A/Vee K P3.6 P3.2 P3.5 LCM Module 8051 MCU Interface	 I2C Vss Vdd P1.0 P1.1 Vss Vdd Vo NC SA0 SA1 CSB SDA SCLK A/Vee K P3.6 P3.2 P3.5 LCM Module 8051 MCU Interface Note

Note：
1. CSB also could be fixed to Vss.
2. SA0 and SA1 also could be fixed to Vss.

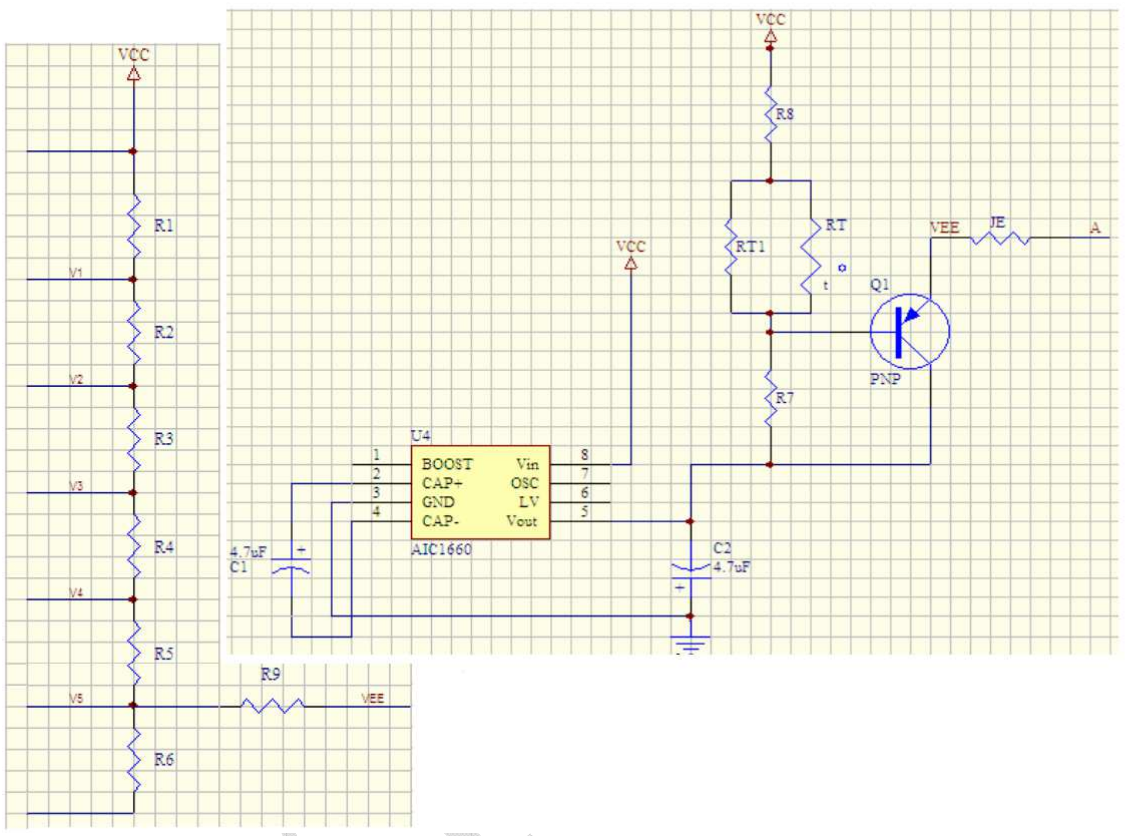
3. IF MCU I/O port w/o pull high resistor, I/O ports which connect SDA and SCLK needs 2.2K ohms to Vdd.

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	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Description Time (540KHz)
Read display data	1	1	Read data								Read data from DDRAM/CGRAM	18.5us
Write display data	1	0	Write data								Write data into DDRAM/CGRAM	18.5us
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM, and set DDRAM address to "00H" from AC	0.76ms
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.	0.76ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Assign cursor moving direction and specify display shift. These operations are performed during data read and write. I/D="1": increment I/D="0": decrement	18.5us
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	Set Display /Cursor/Blink On/OFF D="1": display on D="0": display off C="1": cursor on C="0": cursor off B="1": blink on B="0": blink off	18.5us
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	X	X	Cursor or display shift S/C="1": display shift S/C="0": cursor shift R/L="1": shift to right R/L="0": shift to left	18.5us
Function Set	0	0	0	0	1	DL	N	F	X	X	Set Interface Data Length DL= 8-bit interface/ 4-bit interface N = 2-line/1-line display F= 5x8 Font Size / 5x11Font Size	18.5us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	18.5us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	18.5us
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Can know internal operation is ready or not by reading BF. The contents of address counter can also be read. BF="1": busy state BF="0": ready state	0us

Note：

1. When an MPU program with Busy Flag(DB7) checking is made, 1/ 2 FOSC (is necessary) for

executing the next instruction by the “ E “ signal after the Busy Flag (DB7) goes to “ Low ”.
2. “X” Don’t care

4.2 INSTRUCTION DESCRIPTION

- **Clear Display**

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

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. A content of DDRAM does not change.

- **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

Set the moving direction of cursor and display.

I/D: Increment/decrement of DDRAM address (cursor or blink)
I/D = 1: cursor/blink moves to right and DDRAM address is increased by 1.
I/D = 0: cursor/blink moves to left and DDRAM address is decreased by 1.
* CGRAM operates the same as DDRAM, when read/write from or 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	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
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.
D = 1: entire display is turned on.
D = 0: display is turned off, but display data is remained in DDRAM.

C: Cursor ON/OFF control bit.
C = 1: cursor is turned on.
C = 0: cursor is disappeared in current display, but I/D register remains its data.

B: Cursor Blink ON/OFF control bit.
B = 1: cursor blink is on, that performs alternate between all the high data and display character at the cursor position. If fosc has 540 kHz frequency, blinking has 185 ms interval.
B = 0: blink is off.

- **Cursor or Display Shift**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	-	-

Without writing or reading of display data, shift right/left cursor position or display. This instruction is used to correct or search display data (refer to Table1). During 2-line mode display, cursor moves to the 2nd line after 40th digit of 1st line.

Note that display shift is performed simultaneously by the shift enable instruction. When displayed data is shifted repeatedly, all display lines shifted simultaneously. When display shift is performed, the contents of address counter are not changed.

Table1 Shift Patterns According to S/C and R/L Bits

S/C	R/L	Operation
0	0	Shift cursor to the left, address counter is decreased by 1
0	1	Shift cursor to the right, address counter is increased by 1
1	0	Shift all the display to the left, cursor moves according to the display
1	1	Shift all the display to the right, cursor moves according to the display

- **Function Set**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
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.
IF using IIC and 4-SPI interface、DL bit must be setting to “1”

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	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
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	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
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 (NW = 0), DDRAM address in the 1st line is from "00H" - "27H", and DDRAM address in the 2nd line is from "40H" - "67H".

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

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

This instruction shows whether WS0063 is in internal operation or not. If the resultant BF is “high”, it means the internal operation is in progress and you have to wait until BF to be Low, and then the next instruction can be performed. In this instruction you can read also the value of address counter.

- **Write Data to RAM**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
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 RAM (only support parallel 8-bit bus and 4 bit bus)**

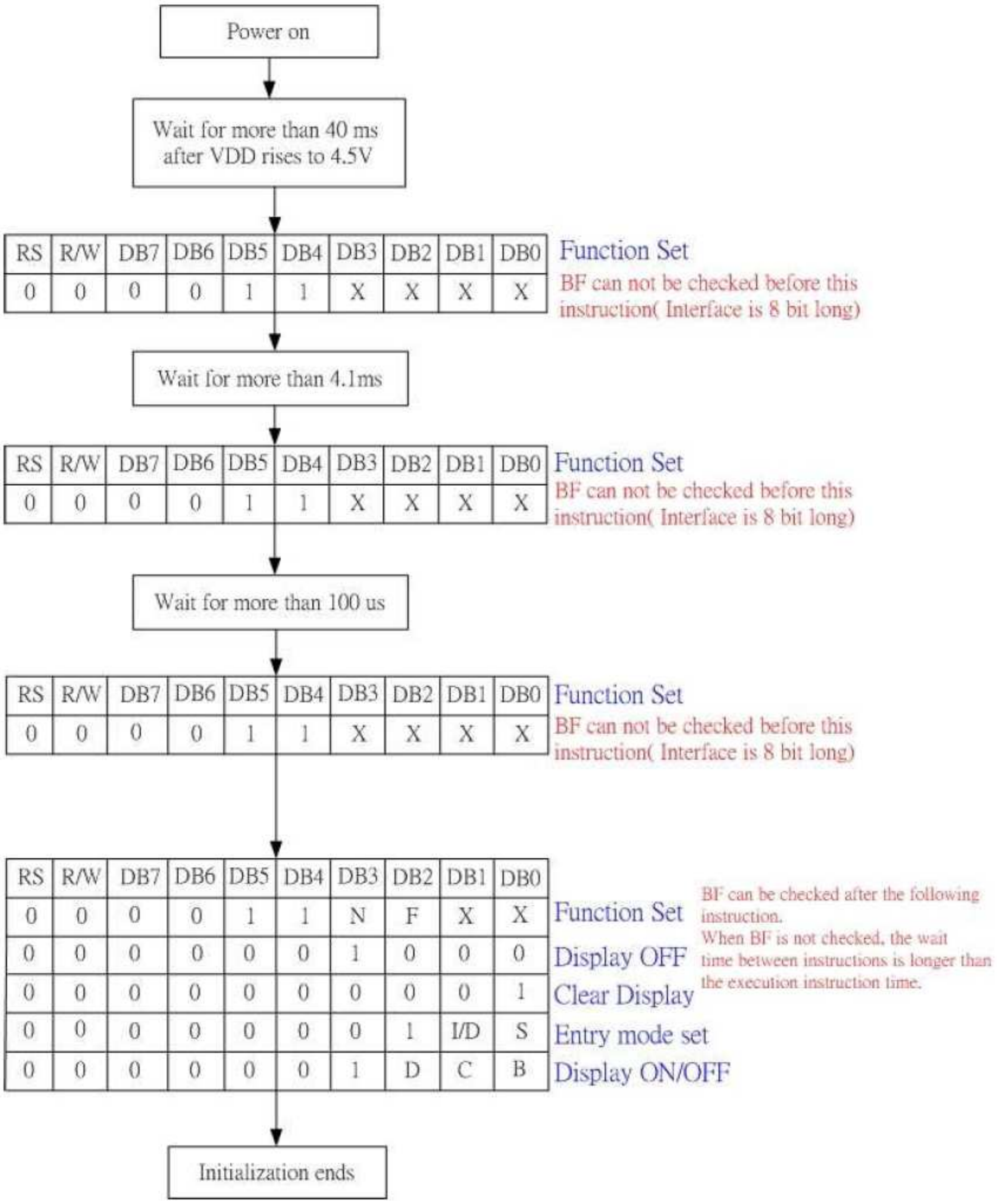
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

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


Power on
Wait for more than 40 ms after VDD rises to 4.5V
Wait for more than 4.1ms
Wait for more than 100 us
Initialization ends

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	X	X	X	X	Function Set BF can not be checked before this instruction(Interface is 8 bit long)

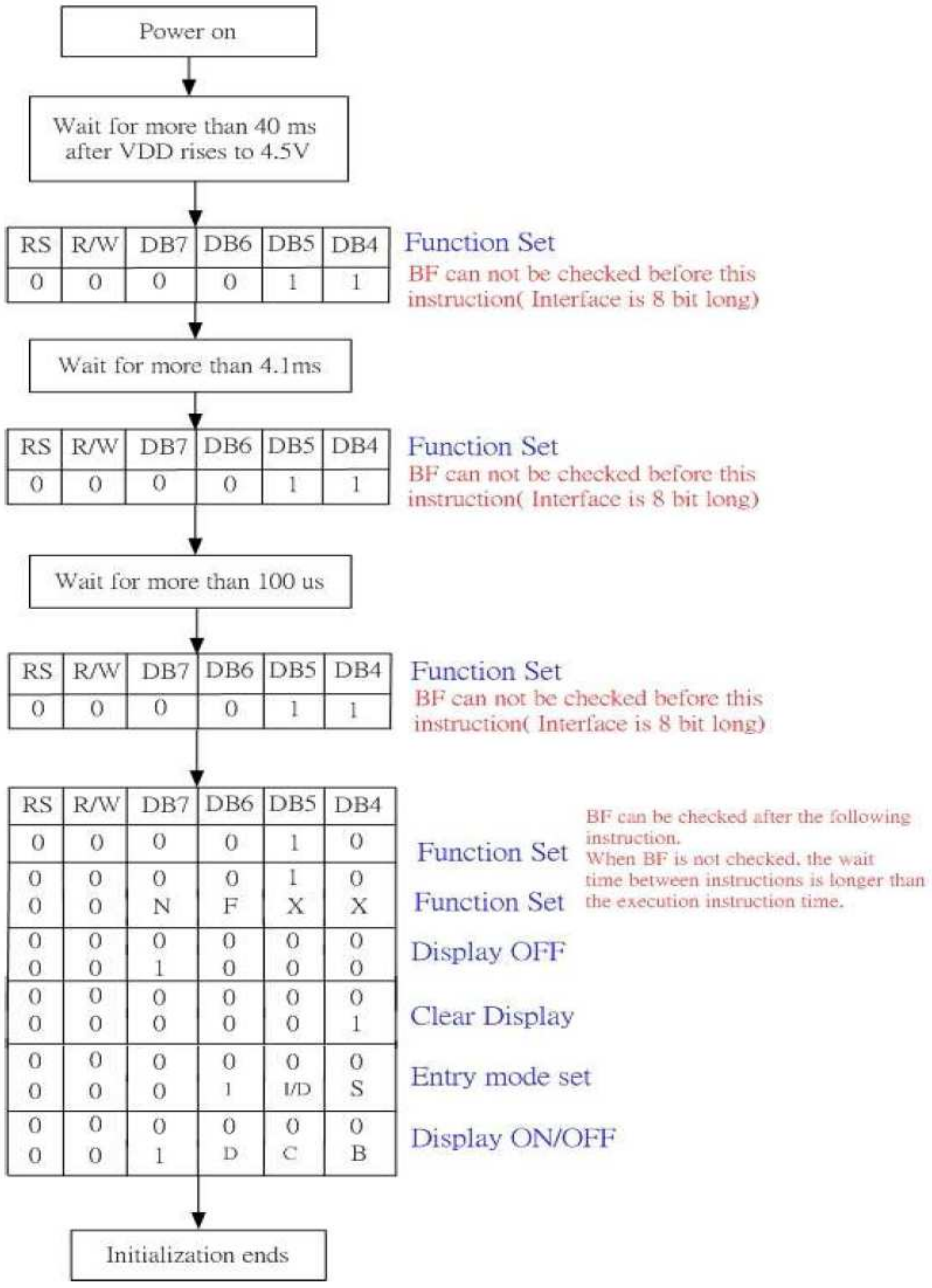
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	X	X	X	X	Function Set BF can not be checked before this instruction(Interface is 8 bit long)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	X	X	X	X	Function Set BF can not be checked before this instruction(Interface is 8 bit long)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	0	1	1	N	F	X	X	Function Set
0	0	0	0	0	0	1	0	0	0	Display OFF
0	0	0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	0	1	I/D	S	Entry mode set
0	0	0	0	0	0	1	D	C	B	Display ON/OFF

BF can be checked after the following instruction.
When BF is not checked, the wait time between instructions is longer than the execution instruction time.

5.2 4-bit Interface



Power on

Wait for more than 40 ms after VDD rises to 4.5V

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

Function Set
BF can not be checked before this instruction(Interface is 8 bit long)

Wait for more than 4.1ms

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

Function Set
BF can not be checked before this instruction(Interface is 8 bit long)

Wait for more than 100 us

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

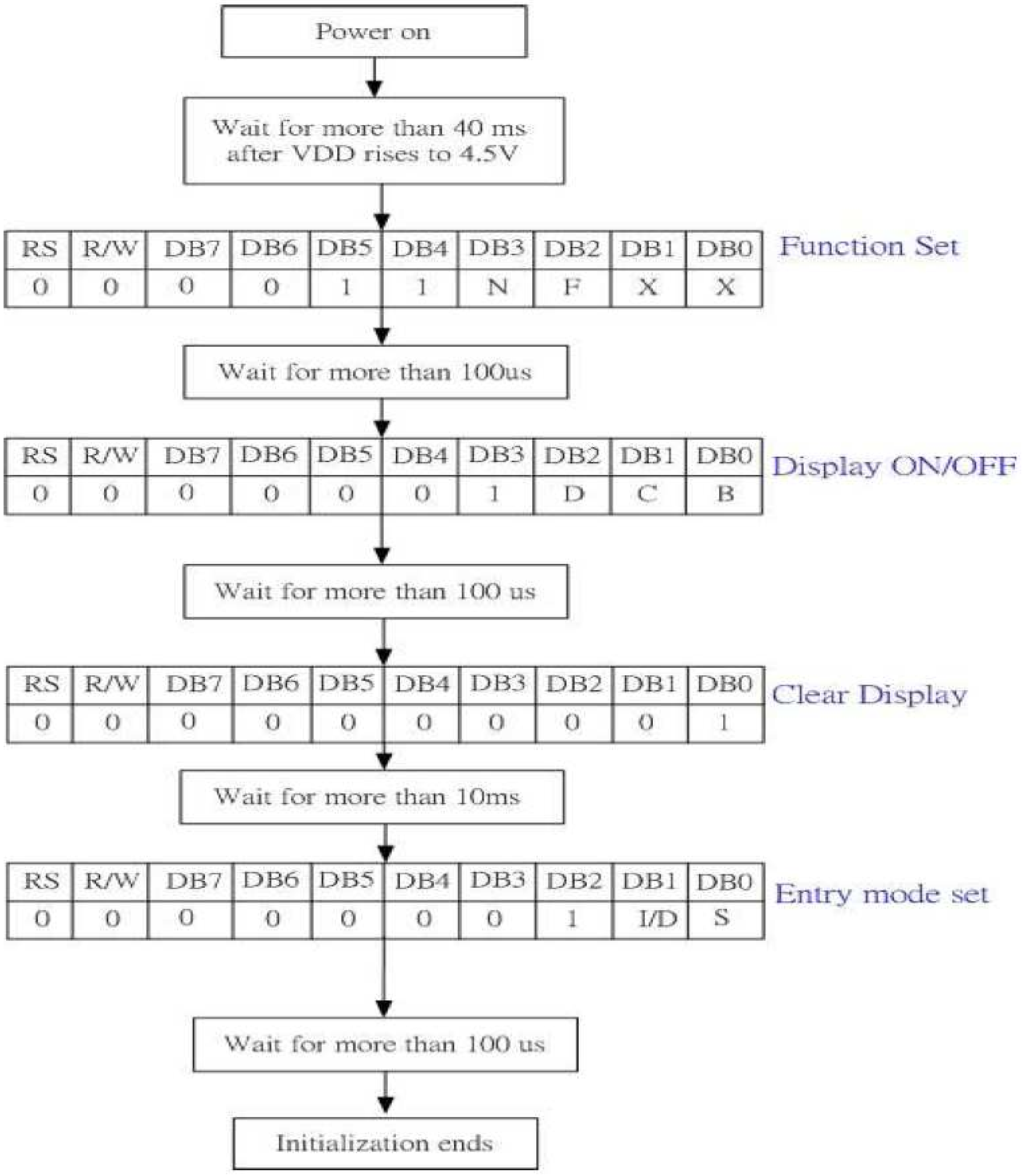
Function Set
BF can not be checked before this instruction(Interface is 8 bit long)

RS	R/W	DB7	DB6	DB5	DB4	
0	0	0	0	1	0	Function Set
0	0	0	0	1	0	
0	0	N	F	X	X	Function Set
0	0	0	0	0	0	
0	0	1	0	0	0	Display OFF
0	0	0	0	0	0	
0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	
0	0	0	1	I/D	S	Entry mode set
0	0	0	0	0	0	
0	0	1	D	C	B	Display ON/OFF

BF can be checked after the following instruction.
When BF is not checked, the wait time between instructions is longer than the execution instruction time.

Initialization ends

5.3 Serial Interface


Power on
Wait for more than 40 ms after VDD rises to 4.5V

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

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

Wait for more than 100 us

Clear Display

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

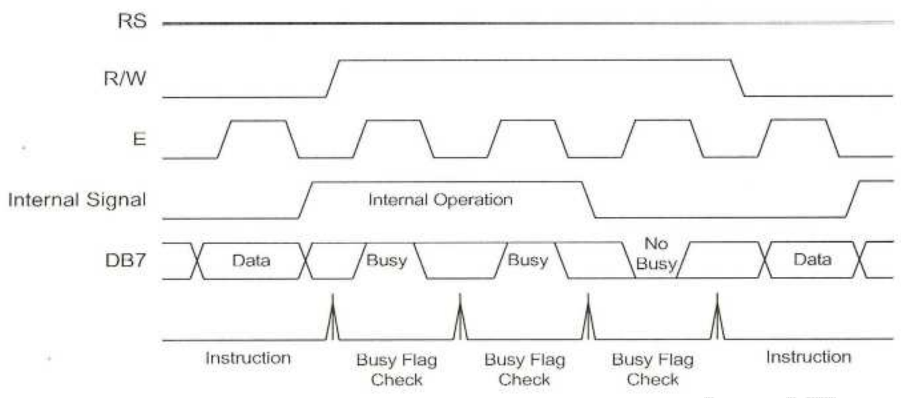
Entry mode set

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

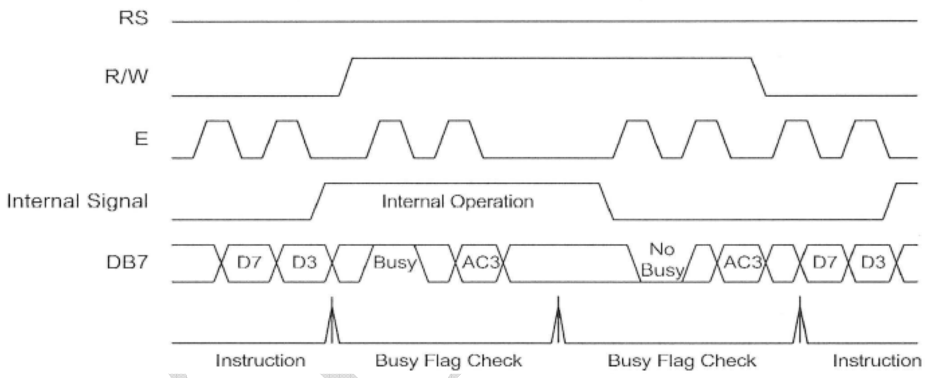
Wait for more than 100 us
Initialization ends

6 MCU INTERFACE

6.1 8-bit

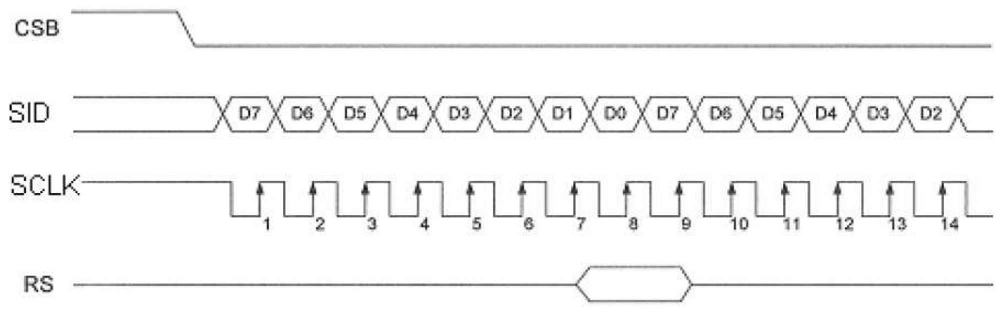


6.2 4-bit



6.3 SPI4

If 4-Pin SPI mode is used, CSB (DB5), SID (DB7), SCLK (DB6), and RS are used.



Note: Following is the master SPI clock mode of MPU.
Idle state for clock is a high level, data transmitted on rising edge of SCLK, and data is hold during low level.

6.4 I2C

Serial data line SDA (DB6) and a Serial clock line SCL (DB7) must be connected to a positive supply via a pull-up resistor.
Data transfer may be initiated only when the bus is not busy.
*The CSB (DB5) Pin must be setting to “VSS”.

- Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse because changes in the data line at this time will be interpreted as a control signal. Bit transfer is illustrated in Fig.1

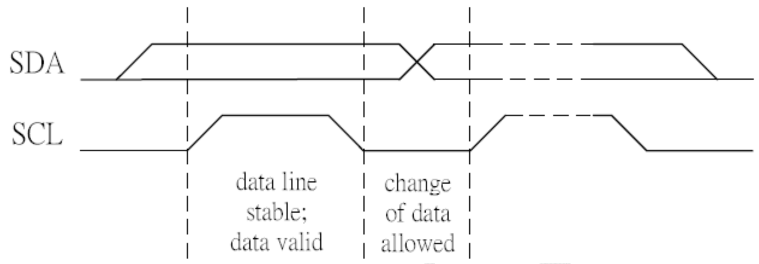


Fig.1 Bit transfer

- Start and Stop conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH is defined as the START condition (S). A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the STOP condition (P). The START and STOP conditions are illustrated in Fig.2.

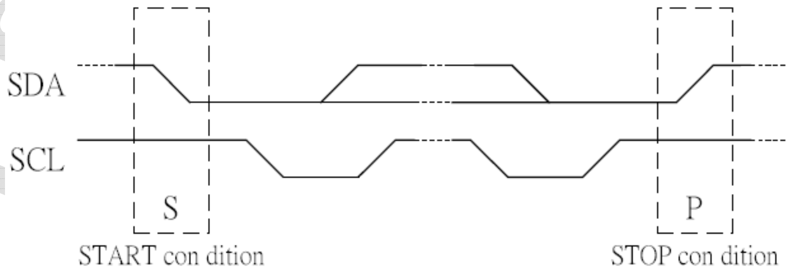


Fig.2 Start and Stop conditions

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_1063();
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_1063(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_1063();
    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

```
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_1063()
{
    WriteIns(0x20);
    delay(2);
    WriteIns(0x28);
    delay(2);
    WriteIns(0x28);
    delay(2);
    WriteIns(0x0c);
    delay(2);
    WriteIns(0x01);
    delay(2);
    WriteIns(0x06);
    delay(2);
}
```

7.3 4-line Serial SPI Interface

```
//============================================================
//    I/O define
//============================================================
sbit            RS              =P3^0;
sbit            CSB     =P3^6;
sbit            SCL     =P3^2;
sbit            SDA     =P3^5;
//================================================================
void WriteIns(unsigned char command){
        unsigned char bMask;
        CSB = 0;
        RS = 0;
        for (bMask = 0x80; bMask; bMask >>= 1){
                SCL = 0;
                _nop_();_nop_();
                SDA = command & bMask;
                SCL = 1;
                _nop_();_nop_();
        }
        RS = 1;
        CSB = 1;
}
//================================
//================================
void WriteData(unsigned char datap){
        unsigned char bMask;
        CSB = 0;
        RS = 1;
        for (bMask = 0x80; bMask; bMask >>= 1){
                SCL = 0;
                _nop_();_nop_();
                SDA = datap & bMask;
                SCL = 1;
                _nop_();_nop_();
        }
        RS = 1;
        CSB = 1;
}
//=================================
//=================================
void Initial(){

        RS              = 0;
        CSB             = 0;
        SCL             = 1;
        SDA             = 1;
        WriteIns(0x38);         //Function Set
        WriteIns(0x08);         //Display off
        WriteIns(0x01);         //Display Clear
        delay(1);
        WriteIns(0x06);         //Entry mode set
        WriteIns(0x0c);         //Display On
}
```

7.4 I2C Interface

```
//=============================================================
//    I/O define
//=============================================================
#define              I2CSA0 2
#define              I2CSA1 4

sbit                 RS                  =P3^0;
sbit                 CSB       =P3^6;
sbit                 SA0       =P1^0;
sbit                 SA1       =P1^1;
sbit                 SCL                 =P3^5;
sbit                 SDA         =P3^2;
//=============================================================================
//       IIC write command
//=============================================================================
void WRITE_CODE(unsigned char I2C_CONTROL, unsigned char I2C_data){
            unsigned char adds = 0x78;
            START_CON();
            Write_control(adds + I2CSA0 + I2CSA1)
            Write_control(I2C_CONTROL);
            Write_data(I2C_data) ;
            STOP_CON();
}

//-----------------------------------------------------------------------------
void START_CON(){
            SDA = 1;
            SCL = 1;
            SDA = 0;
            SCL = 0;
}

//-----------------------------------------------------------------------------
void Write_data(unsigned char data1){
            unsigned char count=0;

            for(count=0;count<8;count++){
                        if(data1&0x80)
                                    SDA=1;
                        else
                                    SDA=0;

                        data1 <<= 1;
                        SCL=1;
                        SCL=0;
            }
            Ack_Check();
}
```

```
//--------------------------------------------------------------------------------
void Write_control(unsigned char C_CONTROL){
            unsigned char count=0;

            for(count=0;count<8;count++){
                        if(C_CONTROL&0x80)
                                    SDA=1;
                        else
                                    SDA=0;

                        C_CONTROL<<=1;
                        SCL=1;
                        SCL=0;
            }
            Ack_Check();
}

//--------------------------------------------------------------------------------
void STOP_CON(){
            SCL = 1;
            SDA = 1;
}

void Ack_Check(){
            SDA = 1;
            SCL = 1;

            do{
                        SDA = 0;
                        SCL = 0;
            }while(SDA);
}
//================================
//INITIALIZE
//============================================
void Initial_1063(){
            RS                    = 0;
            CSB                   = 0;
            SA0                   = I2CSA0;
            SA1                   = I2CSA1;
            SCL         = 1;
            SDA         = 1;


            WRITE_CODE(0x00,0x38);          //SET 2 LINE,5*8 FONT
            CGRAM();
            WRITE_CODE(0x00,0x08);          //Display off
            WRITE_CODE(0x00,0x06);          //Entry mode set
            WRITE_CODE(0x00,0x01);          //CLEAR DISPLAY
            delay(1);
            WRITE_CODE(0x00,0x0c);          //DISPLAY ON,Cursor OFF,Cursor Blink OFF
}
```