

```
//=====
// This library will offer help programming help for the base display
// functions used by the Noritake 7000 and 3900 Series display modules.
// This library is programmed in ANSI C.
// This material is the intellectual property of Noritake Technology Center
// and Noritake Co. Inc.
// Copyright Jan-10-2006 and Mar-07-2006 and May-11-2006
//=====

//=====
// VFD 7000/3900 source code LIB Version 1.1
// Date: JAN 10 ,2006
// By: Lester Johnson
// MAR 7 ,2006 Lester Johnson modification : Formatting
// MAY 11,2006 Lester Johnson modification : Formatting, Speling , Gramma
// File Name: LIBV11.H
//=====

//=====
// Header for all functions
//=====
void VFDport(unsigned char ThisByte);
void B_S(void);
void H_T(void);
void L_F(void);
void HOM_P(void);
void CLR_D(void);
void C_R(void);
void BrighnessLevelSet( unsigned char n);
void InitializeDisplay();
void CursorSet(unsigned int X, unsigned int Y);
void CursorDisplayONOFFSelect( unsigned char n);
void WriteScreenModeSelect(unsigned char a);
void SpecifiesCharacterCodeType(unsigned char n);
void OverWriteMode();
void VerticalScrolMode();
void HorizontalScrollMode();;
void HorizontalScrollSpeed(unsigned char n);
void FontSizeSelect(unsigned char n);
void SpecifyCancel2ByteCharaterMode(unsigned char m);
void Select2byteCharaterType(unsigned char m);
void FontMagnifiedDisplay(unsigned char x,unsigned char y);
void CharacterBoldDisplay(unsigned char b);
void Wait(unsigned char t);
void ShortWait(unsigned char t);
void ScrollDisplayActon(unsigned int w, unsigned int c, unsigned char s);
void DisplayBlink(unsigned char p, unsigned char t1, unsigned char t2, unsigned char c);
void CurtainDisplayAction(unsigned char v, unsigned char s, unsigned char p);
```

```

void SpringDisplayAction(unsigned char s,unsigned int p);
void RandomDisplayAction(unsigned char s,unsigned int p);
void DisplayPowerONOFF(unsigned char p);
void DotPatternDrawing(unsigned char pen, unsigned int x, unsigned int y);
void LineBoxPatterndrawing(unsigned char mode, unsigned char pen,unsigned int x1,unsigned int y1,unsigned int x2,
    unsigned int y2);
void RealTimeBitImageDisplay(unsigned int x, unsigned int y, unsigned char *data2);
void RAMBitImageDefinition(unsigned int A, unsigned char AE, unsigned int S,unsigned char SE, unsigned char *data);
void FROMBitImageDefinition(unsigned int A, unsigned char AE, unsigned int S,unsigned char SE, unsigned char *data);
void DownloadBitImageDisplay(unsigned char m, unsigned int A, unsigned char AE, unsigned int YS,unsigned int x ,
    unsigned int y );
void DownloadBitImageScroll(unsigned char m, unsigned int A, unsigned char AE, unsigned int YS,unsigned int x , unsigned
    int y ,unsigned char s);
void HorizontalScrollQualitySelect(unsigned char n);
void ReverseDisplay(unsigned char n);
void MixtureMode(unsigned char n);
void CurrentWindowSelect(unsigned char a);
void UserWindowDefineCancel(unsigned char a,unsigned char b ,unsigned int xP,unsigned int yP, unsigned int xS,unsigned
    int yS);
void SpecifyDownloadChar(unsigned char n);
void DownloadCharDef(unsigned char a, unsigned char c1, unsigned char c2, *data);
void DeleteDownloadChar(unsigned char a, unsigned char c);
void C16x16DownloadCharDef(unsigned char c1, unsigned char c2, unsigned char * data);
void C16x16DownloadCharDel(unsigned char c1,unsigned char c2);
void SaveDownloadChar(unsigned char a);
void DownloadCharTranswer(unsigned char a);
void FROMUserFontDef(unsigned char m, unsigned char * data);
void UserSetUPModeStart();
void UserSetUPModeEnd();
void IOPortSeting(unsigned char n, unsigned char a);
void IOOutput(unsigned char n,unsigned char a);
void IOInput(unsigned char n);
void RAMmacroDef(unsigned int p,char *data);
void FROMmacroDEF(unsigned char a , unsigned int p, unsigned char t1, unsigned char t2, char *data);
void RunMacro(unsigned char a, unsigned char t1,unsigned char t2);
void MemorySWSetting(unsigned char a ,unsigned char b);
void MemorySWDataSend(unsigned char a);
void DisplayStatus(unsigned char a, unsigned char b, unsigned char c);
void MemoryRewriteMode();

```

```
/*-----*/
```

```

#define BS 0x08
#define HT 0x09
#define LF 0x0A
#define HOM 0x06

```

```
#define CLR 0x0C
#define CR 0x0D
#define US 0x1F
#define ESC 0x1B

//=====
// Generic port output
//=====
void VFDport(unsigned char ThisByte)
{
    printf("02X\n",ThisByte);
}
//=====

/*****

        Backspace
        Command      :      BS

        The cursor is moved one position to the left

*****/
void B_S(void)
{
    VFDport(BS); /* 0x08 */
}

/*****

        Horizontal tab

        Command      :      HT

        The cursor is moved one position to the right

*****/
void H_T(void)
{
    VFDport(HT); /* 0x09 */
}

/*****
```

Line feed

Command : LF

The cursor is moved down one line

```
*****/
void L_F(void)
{
    VFDport(LF); /* 0x0A */
}
```

```
*****
```

Home position

Command : HOM

The cursor is moved to the top left position (0,0)
of the display area

```
*****/
void HOM_P(void)
{
    VFDport(HOM); /* 0x0B */
}
```

```
*****
```

Display clear

Command : CLR

This command erases a display screen.

```
*****/
void CLR_D(void)
{
    VFDport(CLR); /* 0x0C */
}
```

```
*****
```

Carriage return

Command : CR

The cursor is moved to the left most position of
the line it is on currently onk

```

*****/
void C_R(void)
{
    VFDport(CR); /* 0x0D */
}

//3.7.3 Command Set//=====

// 3.7.3.1 General seting Commands
//=====
//-----

/*
    BrighnessLevelSet    1FH,58H, n    n=00H 0%
                                   n=01H 25%
                                   n=02H 50%
                                   n=03H 75%
    Default n=04H         n=04H 100%
    or depending on       n=10H 0%
    Memory SW.            n=11H 12.5%
                                   n=12H 25%
                                   n=13H 37.5%
                                   n=14H 50%
                                   n=15H 62.5%
                                   n=16H 75%
                                   n=17H 87.5%
                                   n=18H 100%

*/

void BrighnessLevelSet( unsigned char n)
{
    VFDport(0x1F);
    VFDport(0x58);
    VFDport(n);
}

//-----

/*
    InitializeDisplay    1BH,40H

```

Clear all display screen and initial all settings

```
*/
void InitializeDisplay()
{
    VFDport(0x1B);
    VFDport(0x40);
}

//-----

/*
    CursorSet 1FH,24H,xL,xH,yL,yH    The cursor moves to specified X,Y position
                                    on display memory.

                                    xL: Cursor position x lower byte
                                    xH: Cursor position x upper byte
                                    yL: Cursor position y lower byte
                                    yH: Cursor position y upper byte
*/

void CursorSet(unsigned int X, unsigned int Y)
{
    VFDport(0x1F);    /* 0x1F */
    VFDport(0x24);    /* 0x24 */
    VFDport(X % 0x100); /* x lower byte */
    VFDport(X / 0x100); /* x upper byte */
    VFDport(Y % 0x100); /* y lower byte */
    VFDport(Y / 0x100); /* y upper byte */

}

//-----

/*
    CursorDisplayONOFFSelect 1FH,43H,n    Display cursor ON/OFF select
                                        n=00H: Cursor OFF
                                        n=01H: Cursor ON
*/

void CursorDisplayONOFFSelect( unsigned char n)
{
```

```

    VFDport(0x1F);      /* 0x1F */
    VFDport(0x43);      /* 0x43 */
    VFDport(n);         /* Cursor State */

}

//-----

//3.7.3.2 Character display setting commands
//=====
//-----

/*

    WriteScreenModeSelect    1FH,28H,77H,10H,a    Selects the write screen
                                                mode for base window.
                                                a=00H Display screen mode
                                                or depending          a=01H All screen mode
                                                on Memory SW

*/
void WriteScreenModeSelect(unsigned char a)
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x77);      /* 0x77 */
    VFDport(0x10);      /* 0x10 */
    VFDport(a);         /* mode a */
}

//-----

/*

    Specifies International Font Set  1BH,52H,n    Some characters located on 20h-7Fh
                                                are chosen/replaced from 14 types
                                                font set.
                                                n=00H: America
                                                or depending          n=01H: France
                                                on Memory SW          n=02H: Germany
                                                n=03H: England
                                                n=04H: Denmark 1
                                                n=05H: Sweden
                                                n=06H: Italy
                                                n=07H: Spain 1
                                                n=08H: Japan

```

```

n=09H: Norway
n=0AH: Denemark 2
n=0BH: Spain 2
n=0CH: Latin America
n=0DH: Korean

*/

void SpecifiesInternationalFontSet(unsigned char n)
{
    VFDport(0x1B);          /* 0x1B */
    VFDport(0x52);          /* 0x52 */
    VFDport(n);             /* fontSet */
}

//-----

/*
SpecifiesCharacterCodeType    1BH,74H,n    The fonts located on 80h-FFh
                                Default n=00H of font table are chosen/
                                or depending replaced from one of 10 types font tables
                                on Memory SW

                                n=00H: PC437 USA Standard Europe
                                n=01H: Katakana
                                n=02H: PC850 Multilingual
                                n=03H: PC860 Portuguese
                                n=04H: PC863 Canadian-French
                                n=05H: PC865 Nordic
                                n=10H: WPC1252
                                n=11H: PC866 Cyrillic #2
                                n=12H: PC852 Latin 2
                                n=13H: PC858
                                n=FFH: User table

*/

void SpecifiesCharacterCodeType(unsigned char n)
{
    VFDport(0x1B);          /* 0x1B */
    VFDport(0x74);          /* 0x74 */
    VFDport(n);             /* fontTable */
}

```



```
//-----

/*
    OverWriteMode          1FH,01H          Sets to Over-write mode
*/

void  OverWriteMode()
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x01);          /* 0x01 */

}

//-----

/*
    VerticalScrolMode      1FH,02H          Sets to Vertical scroll mode
*/

void  VerticalScrolMode()
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x02);          /* 0x02 */

}

//-----

/*
    HorizontalScrollMode   1FH,03H          Sets selection to Horizontal Scroll Mode
*/

void  HorizontalScrollMode();
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x03);          /* 0x03 */

}

//-----

/*
    HorizontalScrollSpeed   1FH,73H,n       Sets Horizontal Scroll Speed
```

```

                                Default n=00H
                                or depending
                                on Memory SW
*/

void HorizontalScrollSpeed(unsigned char n)
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x73);      /* 0x73 */
    VFDport(n);         /* speed*/
}

//-----

/*
    FontSizeSelect          1FH,28H,67H,      Selects character font size
                           01H,n
                           Default n=00H      n=01H: 6x8 font
                           or depending         n=02H: 8x16 font
                           on Memoery SW       n=03H: 16x32 font
*/
void FontSizeSelect(unsigned char n)
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x67);      /* 0x67 */
    VFDport(0x01);      /* 0x01*/

    VFDport(n);         /* FontSize*/
}

//-----

/*
    SpecifyCancel2ByteCharaterMode
                           1FH,26H,67H,02H, Specifies or cancels 2 byte character m mode.
                           m=01H Specify 2 byte character mode
                           m=00H Cancel 2 byte character mode
*/

void SpecifyCancel2ByteCharaterMode(unsigned char m)

```

```
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x26);          /* 0x26 */
    VFDport(0x67);          /* 0x67 */
    VFDport(0x02);          /* 0x02*/

    VFDport(m);             /* param m*/
}

//-----

/*
    Select2byteCharaterType    1FH,28H,67,03H,  Select 2 byte character type
                                m                m=00H: Japanese
                                Default m=00H   m=01H: Korean
                                or depending    m=02H: Simplified Chinese
                                on Memory SW    m=03H: Traditional Chinese
*/

void    Select2byteCharaterType(unsigned char m)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x67);          /* 0x67 */
    VFDport(0x03);          /* 0x03*/

    VFDport(m);             /* param m*/
}

//-----

/*
    FontMagnifiedDisplay       1FH,28H,67H,41H, Magnify the character by x times
                                x,y             on the right , y times downward
                                Default          x: specify the size of magnification x
                                x=01H,y=01H     y: specify the size of magnification y
                                or depending
                                on Memory SW
*/

void    FontMagnifiedDisplay(unsigned char x,unsigned char y)
{
    VFDport(0x1F);          /* 0x1F */
```

```

    VFDport(0x28);          /* 0x28 */
    VFDport(0x67);          /* 0x67 */
    VFDport(0x41);          /* 0x41*/

    VFDport(x);             /* param x*/
    VFDport(y);             /* param y*/

}

//-----
/*
CharacterBoldDisplay      1FH,28H,67H,41H,b   Specifies or Cancels   boldface character.

                        Default b=00H   b=00H: Cancel Bold
                        or depending    b=01H: Specify Bold
                        on Memory SW

*/
void CharacterBoldDisplay(unsigned char b)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x67);          /* 0x67 */
    VFDport(0x41);          /* 0x41*/

    VFDport(b);             /* param b*/

}

//-----

//3.7.3.3 Display action setting commands
//=====
//-----

/*
Wait                      1FH,28H,61H,01H,t   Waits t time

                                                t: Wait time   approx 0.5 sec by one unit

*/

void Wait(unsigned char t)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x61);          /* 0x61 */

```

```

        VFDport(0x01);          /* 0x01*/

        VFDport(t);             /* param t*/

    }

//-----

/*  ShortWait                1FH,28H,61H,02H,t    Waits t time

                                t: Wait time approx 16 msec  by one unit

*/
void  ShortWait(unsigned char t)
{
    VFDport(0x1F);              /* 0x1F */
    VFDport(0x28);              /* 0x28 */
    VFDport(0x61);              /* 0x61 */
    VFDport(0x02);              /* 0x02*/

    VFDport(t);                 /* param t*/

}

//-----

/*
    ScrollDisplayAction      1FH,28H,61H,10H      Shits the display screen horizontal
                           ,wL,wH,cL,cH,s        by w screen shift with repetition c  and speed s

                                wL: Number of display sreeen shift lower byte
                                wH: Number of display screen shift upper byte
                                cL: Number of repetition lower byte
                                cH: Number of repetition upper byte
                                s:  Scroll speed

*/

void  ScrollDisplayActon(unsigned int w, unsigned int c, unsigned char s)
{
    VFDport(0x1F);              /* 0x1F */
    VFDport(0x28);              /* 0x28 */
    VFDport(0x61);              /* 0x61 */
    VFDport(0x10);              /* 0x10*/
    VFDport(w % 0x100);          /* Display screen shift ,number of lower byte */
    VFDport(w / 0x100);          /* Display screen shift ,number of upper byte */
    VFDport(c % 0x100);          /* Number of repetition lower byte */

```

```

    VFDport(c / 0x100);    /* Number of repetition upper byte */

    VFDport(s);            /* Speed param */

}

//-----
/*
    DisplayBlink          1FH,28H,61H,11H    Blink display action on display screen
                        p,t1,t2,c
                                p: Blink pattern
                                t1: Normal display time
                                t2: Blank or Reverse display time
                                c:  Number of repetitions
*/
void DisplayBlink(unsigned char p, unsigned char t1, unsigned char t2, unsigned char c)
{
    VFDport(0x1F);        /* 0x1F */
    VFDport(0x28);        /* 0x28 */
    VFDport(0x61);        /* 0x61 */
    VFDport(0x11);        /* 0x11*/

    VFDport(p);           /*
                            blink pattern , p=0: Normal display
                            p=1: Repeat blink display with normal and blank display
                            p=2: Repeat blink display with normal and reverse display
                        */
    VFDport(t1);          /* Normal display time */
    VFDport(t2);          /* Blank or Reverse display time */

    VFDport(c);           /* Repetition param */

}

//-----
/*
    CurtainDisplayAction  1FH,28H,61H,12H,    Curtain display action on display screen
                        v,s,p
                                v: Direction of curtain action
                                s: Curtain action speed
                                p: Curtain action pattern
*/

```

```
void CurtainDisplayAction(unsigned char v, unsigned char s, unsigned char p)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x61);          /* 0x61 */
    VFDport(0x12);          /* 0x12*/

    VFDport(v);             /* Direction param */
    VFDport(s);             /* Speed param */
    VFDport(p);             /* Patern param */

}

//-----

/*
    SpringDisplayAction      1FH,28H,61H,13H,      Spring display action on display screen
                           v,s,pL,pH
                           v: Direction of spring action
                           s: Spring action speed
                           pL: Display memory pattern lower byte
                           pH: Display memory pattern upper byte
*/

void SpringDisplayAction(unsigned char s,unsigned int p)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x61);          /* 0x61 */
    VFDport(0x13);          /* 0x13 */

    VFDport(s);             /* Speed parameter*/
    VFDport(p % 0x100);      /* Display memory pattern address lower byte */
    VFDport(p / 0x100);      /* Display memory pattern address upper byte */
}

//-----

/*
    RandomDisplayAction      1FH,28H,61H,14H,      Random display action on display screen
                           s,pL,pH
                           s: random display action speed
                           pL: Display memory pattern address lower byte
*/
```

pH: Display memory pattern address upper byte

```

*/

void RandomDisplayAction(unsigned char s,unsigned int p)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x61);          /* 0x61 */
    VFDport(0x14);          /* 0x14 */

    VFDport(s);             /* Speed parameter*/
    VFDport(p % 0x100);      /* Display memory pattern address lower byte */
    VFDport(p / 0x100);      /* Display memory pattern address upper byte */

}

//-----
/*
DisplayPowerONOFF          1FH,28H,61H,40H,      Set the display power ON or OFF
                           p
                           p=01H: ON
                           p=00H: OFF

*/

void DisplayPowerONOFF(unsigned char p)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x61);          /* 0x61 */
    VFDport(0x40);          /* 0x40 */

    VFDport(p);             /* parameter p */

}
//-----

//3.7.3.4 Setting Bit Inmage Display Command
//=====
//-----
/*
DotPatternDrawting        1FH,28H,64H,10H      Display the dot pattern on a drawing
                           ,pen,xL,xH,yL,yH      position or delete the dot pattern already
                                                displayed.

```


pen: Dot display ON or OFF
 XL: Dot pattern drawing position x lower byte
 xH: Dot pattern drawing position x upper byte
 yL: Dot pattern drawing position y lower byte
 yH: Dot pattern drawing position y upper byte

```

*/
void DotPatternDrawing(unsigned char pen, unsigned int x, unsigned int y)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x64);          /* 0x64 */
    VFDport(0x10);          /* 0x10 */

    VFDport(pen);           /* pen */
    VFDport(x % 0x100);     /* Dot pattern position lower byte */
    VFDport(x / 0x100);     /* Dot pattern position upper byte */
    VFDport(y % 0x100);     /* Dot pattern position lower byte */
    VFDport(y / 0x100);     /* Dot pattern position upper byte */
}
//-----
/*
  LineBoxPatterndrawing    1FH,28H,64H,11H    Display the Line, Box, Box Fill on the drawing area
                        ,mode,pen,x1L,        specified by x1,y1,x2,y2 or delete the dotpattern
                        x1H,y1L,y1H,x2L        already displayed
                        ,x2H,y2L,y2H
                                mode: Drawing mode select
                                pen: Dot ON or OFF
                                x1L: LineBox pattern drawing start position x1, lower byte
                                x1H: LineBox pattern drawing start position x1, upper byte
                                y1L: LineBox pattern drawing start position y1, lower byte
                                y1H: LineBox pattern drawing start position y1, upper byte
                                x2L: LineBox pattern drawing end position x2, lower byte
                                x2H: LineBox pattern drawing end position x2, upper byte
                                y2L: LineBox pattern drawing end position y2, lower byte
                                y2H: LineBox pattern drawing end position y2, upper byte
*/
void LineBoxPatterndrawing(unsigned char mode, unsigned char pen,unsigned int x1,unsigned int y1,unsigned int x2,
    unsigned int y2)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x64);          /* 0x64 */
    VFDport(0x10);          /* 0x11 */
    VFDport(mode);          /* mode */
    VFDport(pen);           /* pen */

```

```

VFDport(x1 % 0x100);    /* x1L: LineBox pattern drawing start position x1, lower byte */
VFDport(x1 / 0x100);    /* x1H: LineBox pattern drawing start position x1, upper byte */
VFDport(y1 % 0x100);    /* y1L: LineBox pattern drawing start position y1, lower byte */
VFDport(y1 / 0x100);    /* y1H: LineBox pattern drawing start position y1, upper byte */
VFDport(x2 % 0x100);    /* x2L: LineBox pattern drawing end position x2, lower byte */
VFDport(x2 / 0x100);    /* x2H: LineBox pattern drawing end position x2, upper byte */
VFDport(y2 % 0x100);    /* y2L: LineBox pattern drawing end position y2, lower byte */
VFDport(y2 / 0x100);    /* y2H: LineBox pattern drawing end position y2, upper byte */

}
//-----

/*
RealTimeBitImageDisplay 1FH,28H,66H,11H      Display the bit image data on the
                        ,xL,xH,yL,yH,01H      Cursor Position real - time
                        ,d(1)...d(S)          xL: Bit image X size lower byte by 1 dot
                                                xH: Bit image X size upper byte by 1 dot
                                                yL: Bit image Y size lower byte by 8 dots
                                                yH: Bit image Y size upper byte by 8 dots
                                                d(1) ... d(S): image data.

*/

void RealTimeBitImageDisplay(unsigned int x, unsigned int y, unsigned char *data2)
{
int i;
    VFDport(0x1F);        /* 0x1F */
    VFDport(0x28);        /* 0x28 */
    VFDport(0x66);        /* 0x66 */
    VFDport(0x11);        /* 0x11 */

    VFDport(x % 0x100);    /* Bit image X size lower byte (by 1dot) */
    VFDport(x / 0x100);    /* Bit image X size upper byte (by 1dot) */
    VFDport(y % 0x100);    /* Bit image Y size lower byte (by 8dots) */
    VFDport(y / 0x100);    /* Bit image Y size upper byte (by 8dots) */
    VFDport(1);           /* image=1 (Fixed) */
    for(i=0; i<(x*y); i++) /* This loop is the loop to read bit image data for 8 dot. */
        VFDport(*data2++); /* d(1) - d(S): Bit image data */

}

//-----

```

/*

```

RAMBitImageDefinition    1FH,28H,66H,01H    Define user bit image to the RAM
                        ,aL,aH,aE,sL,sH,
                        ,sE,d(1)...d(s)
                        aL: Bit image data definition address lower byte
                        aH: Bit image data definition address upper byte
                        aE: Bit image data definition address extension byte
                        sL: Bit image data lenght lower byte
                        sH: Bit image data lenght upper byte
                        sE: Bit image data lenght extension byte
                        d(1) ... d(S): Image data

```

*/

```

void  RAMBitImageDefinition(unsigned int A, unsigned char AE, unsigned int S,unsigned char SE, unsigned char *data)
{
int i;

```

```

    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x66);          /* 0x66 */
    VFDport(0x01);          /* 0x01 */

```

```

    VFDport(A % 0x100);
    VFDport(A / 0x100);
    VFDport(AE);

```

```

    VFDport(S % 0x100);
    VFDport(S / 0x100);
    VFDport(SE);

```

```

    for(i=0; i<(A*S); i++)    /* This loop is the loop to read bit image data for 8 dot. */
        VFDport(*data++);    /* d(1) - d(S): Bit image data */

```

}

//-----

/*

```

FROMBitImageDefinition    1FH,28,66H,10H    Define user bit image to the FROM
                        aL,aH,aE,sL,sH
                        ,sE,d(1)...d(s)
                        aL: Bit image data definition address lower byte
                        aH: Bit image data definition address upper byte
                        aE: Bit image data definition address extension byte
                        sL: Bit image data length lower byte
                        sH: Bit image data length upper byte

```

sE: Bit image data length extension byte
d(1) ... d(S): Image data

*/

```
void FROMBitImageDefinition(unsigned int A, unsigned char AE, unsigned int S, unsigned char SE, unsigned char *data)
{
    int i;
```

```
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x66);          /* 0x66 */
    VFDport(0x10);          /* 0x10 */
```

```
    VFDport(A % 0x100);
    VFDport(A / 0x100);
    VFDport(AE);
```

```
    VFDport(S % 0x100);
    VFDport(S / 0x100);
    VFDport(SE);
```

```
    for(i=0; i<(A*S); i++)    /* This loop is the loop to read bit image data for 8 dot. */
        VFDport(*data++);    /* d(1) - d(S): Bit image data */
```

}

//-----

/*

```
DownloadBitImageDisplay 1FH,28H,65H,10H,
                        m,aL,aH,aE,ySL,
                        ySH,xL,xH,yL,yH,
                        01H
```

Display the RAM or FROM bit image defined on cursor position

m: Select bit image data display memory
aL: Bit image data definition address lower byte
aH: Bit image data definition address upper byte
aE: Bit image data definition address extension byte
ySL: Bit image defined Y size lower byte by 8dots
ySH: Bit image defined Y size upper byte by 8dots
xL: Bit image display X size lower byte by 1dot
xH: Bit image display X size upper byte by 1dot
yL: Bit image display Y size lower byte by 8dots
yH: Bit image display Y size upper byte by 8dots

*/

```
void DownloadBitImageDisplay(unsigned char m, unsigned int A, unsigned char AE, unsigned int YS,unsigned int x , unsigned
int y )
```

```
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x65);          /* 0x65 */
    VFDport(0x10);          /* 0x10 */
```

```
    VFDport(m);            /* m */
```

```
    VFDport(A % 0x100);
    VFDport(A / 0x100);
    VFDport(AE);
```

```
    VFDport(YS % 0x100);
    VFDport(YS / 0x100);
```

```
    VFDport(x % 0x100);
    VFDport(x / 0x100);
```

```
    VFDport(y % 0x100);
    VFDport(y / 0x100);
```

```
    VFDport(0x01);          /* 0x01 */
```

}

//-----

/*

```
DownloadBitImageScroll 1FH,28H,66H,90H
                        m,aL,aH,aE,ySL,
                        ySH,xL,xH,yL,yH
                        ,01H,s
```

```
Scroll display the RAM , FROM or display memory bit
image defined from right end of current window
```

```
m: Select bit image data display memory
aL: Bit image data definition address lower byte
aH: Bit image data definition address upper byte
aE: Bit image data definition address extension byte
ySL: Bit image defined Y size lower byte by 8dots
ySH: Bit image defined Y size upper byte by 8dots
xL: Bit image display X size lower byte by 1dot
xH: Bit image display X size upper byte by 1dot
```

yL: Bit image display Y size lower byte by 8dots
 yH: Bit image display Y size upper byte by 8dots
 s: Scroll speed

*/

void DownloadBitImageScroll(unsigned char m, unsigned int A, unsigned char AE, unsigned int YS, unsigned int x , unsigned int y , unsigned char s)

```
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x66);          /* 0x66 */
    VFDport(0x90);          /* 0x90 */
```

```
    VFDport(m);             /* m */
```

```
    VFDport(A % 0x100);
    VFDport(A / 0x100);
    VFDport(AE);
```

```
    VFDport(YS % 0x100);
    VFDport(YS / 0x100);
```

```
    VFDport(x % 0x100);
    VFDport(x / 0x100);
```

```
    VFDport(y % 0x100);
    VFDport(y / 0x100);
```

```
    VFDport(0x01);          /* 0x01 */
```

```
    VFDport(s);             /* s */
```

```
}
```

```
//-----
```

```
//3.7.3.5 General Display setting commands
```

```
//=====
```

```
//-----
```

```
/*
```

HorizontalScrollQualitySelect Select the visual quality of horizontal scroll

```

        1FH,60H,n          n=00H Scroll speed priority
        Default n=00H      n=01H Visual quality priority
        or depending
        on Memory SW
*/
void HorizontalScrollQualitySelect(unsigned char n)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x60);          /* 0x60 */
    VFDport(n);             /* n */
}

//-----
/*
SpecifiesORCanselReverseDisplay          Specifies or cancels reverse display
Aka:
    ReverseDisplay    1FH,72H,n          n=00H Cancels reverse mode
                        Default n=00H      n=01H Specify reverse mode
                        or depending
                        on Memory SW
*/
void ReverseDisplay(unsigned char n)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x72);          /* 0x72 */
    VFDport(n );           /* n */
}

//-----
/*
SpecifiesWriteMixtureDisplayMode          Specifies write mixture mode
Aka:
    MixtureMode        1FH,77H,n          n=00H Normal display
                        Default n=00H      n=01H OR Display Write
                        or depending        n=02H XOR Display Write
                        on Memory SW
*/
void MixtureMode(unsigned char n)
{
    VFDport(0x1F);          /* 0x1F */

```

```

        VFDport(0x77);          /* 0x77 */
        VFDport(n);             /* n */

}

//-----
//3.7.3.6 Windows display setting commands
//=====
//-----

/*
CurrentWindowSelect    1FH,28H,77H,01H,a    Select current window
                                           a=00H: Base Window
                                           a=01H: User Window1
                                           a=02H: User Window2
                                           a=03H: User Window3
                                           a=04H: User Window4

*/
void CurrentWindowSelect(unsigned char a)
{
    VFDport(0x1F);              /* 0x1F */
    VFDport(0x28);              /* 0x28 */
    VFDport(0x77);              /* 0x77 */
    VFDport(0x01);              /* 0x01 */
    VFDport(a);                 /* a */

}

//-----

/*
UserWindowDefineCancel    1FH,28H,77H,02H,
                          a,b,xPL,xPH,yPL,
                          yPH,xSL,xSH,ySL,
                          ySH    Define or Cancel User window
                                           a: Define window No.
                                           b: Define or Cancel
                                           xPL: Left position of window lower byte by 1dot
                                           xPH: Left position of window upper byte by 1dot
                                           yPL: Top position of window lower byte by 8dots
                                           yPH: Top position of window upper byte by 8dots
                                           xSL: X size of window lower byte by 1 dot
                                           xSH: X size of window upper byte by 1 dot
                                           ySL: Y size of window lower byte by 8 dots
                                           ySH: Y size of window upper byte by 8 dots

*/

```



```

void UserWindowDefineCancel(unsigned char a,unsigned char b ,unsigned int xP,unsigned int yP, unsigned int xS,unsigned
    int yS)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x77);          /* 0x77 */
    VFDport(0x02);          /* 0x02 */

    VFDport(a);             /* a */
    VFDport(b);             /* b */

    VFDport(xP % 0x100);
    VFDport(xP / 0x100);

    VFDport(yP % 0x100);
    VFDport(yP / 0x100);

    VFDport(xS % 0x100);
    VFDport(xS / 0x100);

    VFDport(yS % 0x100);
    VFDport(yS / 0x100);

}

//-----
//3.7.3.7 Download character setting commands
//=====
/*
SpecifyDownloadChar    1BH,25H,n          Enable or Disable for download character
                        Default n=00H      n=01H: Enable
                                           n=00H: Disable
*/

void SpecifyDownloadChar(unsigned char n)
{
    VFDport(0x1B);          /* 0x1B */
    VFDport(0x25);          /* 0x25 */
    VFDport(n );           /* n */

}

```

```
//-----
/*
DownloadCharDef      1BH,26H,a,c1,c2      Defines 6x8 or 8x16 download char into RAM
                    ,x1,d1..dx1,xk      a: Select char Type
                    ,d1...dxk          c1: Start char code
                                       c2: End char code
                                       x: Number of dot for X direction
                                       d1..dxk: Defined data

*/
void DownloadCharDef(unsigned char a, unsigned char c1, unsigned char c2, *data)
{

    unsigned char i,j ;
    unsigned char x_d;

    VFDport(0x1B);          /* 0x1B */
    VFDport(0x26);          /* 0x26 */

    VFDport(a);             /* a */
    VFDport(c1);            /* c2 */
    VFDport(c2);            /* c2 */

    /*****
    There are 2 loops in the following sequence.
    The 1st is a loop command that demonstrates code to acquire and
    store more than one character.
    The 2nd loop shows how to read data for one line.
    *****/

    for(i=0; i<(c2-c1+1); i++)
    {
        x_d=*data++;
        VFDport(x_d);          /* x: Number of dot for X direction */
        for(j=0; j<x_d; j++) VFDport(*data++); /* d1 - dx: Defined data */
    }

}

//-----
/*
DeleteDownloadChar    1BH,3FH,a,c          Deletes a previously downloaded 6x8 dot character
                    a: Select char type

```

c: Character code to delete

*/

void DeleteDownloadChar(unsigned char a, unsigned char c)

```
{
    VFDport(0x1B);      /* 0x1B */
    VFDport(0x3F);      /* 0x3F */
    VFDport(a);         /* a */
    VFDport(c);         /* c */
}
```

//-----

/*

```
16x16DownloadCharDef  1FH,28H,67H,10H      Defines 16x16 downladed character in specified
                      ,c1,c2,d1...dk        code
                                           c1: Char code upper byte
                                           c2: Char code lower byte
                                           d:  Definition data
```

*/

void C16x16DownloadCharDef(unsigned char c1, unsigned char c2, unsigned char * data)

```
{
    unsigned char i ;

    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x67);      /* 0x67 */
    VFDport(0x10);      /* 0x10 */
    VFDport(c1);        /* c1 */
    VFDport(c2);        /* c2 */
    for(i=0; i<=(c2-c1); i++) VFDport(*data++);
}
```

//-----

/*

```
16x16DownloadCharDel  1FH,28H,67H,11H      Deletes the 16x16 downloaded character in specified
                      ,c1,c2                code
                                           c1: Char code upper byte
                                           c2: Char code lower byte
```

*/

void C16x16DownloadCharDel(unsigned char c1,unsigned char c2)

```
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x67);      /* 0x67 */
    VFDport(0x11);      /* 0x11 */
    VFDport(c1);         /* c1 */
    VFDport(c2);         /* c2 */
}

//-----

/*
SaveDownloadChar      1FH,28H,65H,11H,a    Saves the downloaded character already defined
                                         on RAM to FROM
                                         a:Select font type
                                         a=01H: 8x8 dot
                                         a=02H: 8x16 dot
                                         a=03H: 16x16 dot
(valid at the user setup mode)
*/
void SaveDownloadChar(unsigned char a)
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x65);      /* 0x65 */
    VFDport(0x11);      /* 0x11 */
    VFDport(a);         /* a */
}

//-----

/*
DownloadCharTransfer  1FH,28H,65H,21H,a    Transfers the download char defined
                                         in FROM to RAM
                                         a:Select font type
                                         a=01H: 8x8 dot
                                         a=02H: 8x16 dot
                                         a=03H: 16x16 dot
*/
void DownloadCharTransfer(unsigned char a)
{
    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x65);      /* 0x65 */
    VFDport(0x21);      /* 0x21 */
    VFDport(a);         /* a */
}
```

```
}

//-----
/*

FROMUserFontDef      1FH,28H,65H,13H      Define the user font of 1byte code to the
                    ,m,P(80H-1)            user table
                    ,P(80H-2) ...          m: User table
                    ,P(FFH-n)             m=01H: 6x8 dot
                                         m=02H: 8x16 dot
                                         m=04H: 16x16 dot
                                         P: Definition data

*/
void FROMUserFontDef(unsigned char m, unsigned char *data)
{
    int i;
    int bnt;

    VFDport(0x1F);      /* 0x1F */
    VFDport(0x28);      /* 0x28 */
    VFDport(0x65);      /* 0x65 */
    VFDport(0x13);      /* 0x13 */
    VFDport(m);         /* m */

    if(m==0x01)bnt=768;
    if(m==0x02)bnt=2048;
    if(m==0x04)bnt=8192;

    for(i=0; i<bnt; i++) VFDport(*data++);

}

//-----
//3.7.3.8 User setup mode setting commands
//=====
//-----

/*
UserSetUPModeStart    1FH,28H,65H,01H      Start user set up mode
                    ,49H,4EH

*/

void UserSetUPModeStart()
{
```

```

        VFDport(0x1F);          /* 0x1F */
        VFDport(0x28);          /* 0x28 */
        VFDport(0x65);          /* 0x65 */
        VFDport(0x01);          /* 0x01 */
        VFDport(0x49);          /* 0x65 */
        VFDport(0x4E);          /* 0x4E */

}

//-----

/*
UserSetUPModeEnd      1FH,28H,65H,02H      Ends user set up mode
                      ,4FH,55H,54H
*/
void UserSetUPModeEnd()
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x65);          /* 0x65 */
    VFDport(0x02);          /* 0x02 */
    VFDport(0x4F);          /* 0x4F */
    VFDport(0x55);          /* 0x55 */
    VFDport(0x54);          /* 0x54 */

}

//-----

//3.7.3.9 General purpose I/O Port control commands
//=====
//-----

/*
I/O Port Input Output setting
AKA:
IOPortSeting          1FH,28H,70H,01H      Sets  GPIO (Genaral Purpose Input Output) pins  for input or output
                      ,n,a                  n:I/O port number
                                           n=00H: Port0
                                           n=001: Port1
                                           a=00H: Input
                                           a=01H: Output
*/
void IOPortSeting(unsigned char n, unsigned char a)
{
    VFDport(0x1F);          /* 0x1F */

```

```

        VFDport(0x28);          /* 0x28 */
        VFDport(0x70);          /* 0x70 */
        VFDport(0x01);          /* 0x01 */
        VFDport(n);             /* n */
        VFDport(a);             /* a */

}

//-----
/*
IOOutput          1FH,28H,70H,10H,    Outputs data to  General Purpose Input  Output pins  (GPIO)
                  n,a                  n: Port number
                                      a: Value
*/
void IOOutput(unsigned char n,unsigned char a)
{
    VFDport(0x1F);             /* 0x1F */
    VFDport(0x28);             /* 0x28 */
    VFDport(0x70);             /* 0x70 */
    VFDport(0x10);             /* 0x10 */
    VFDport(n);                /* n */
    VFDport(a);                /* a */

}

//-----
/*
IOInput          1FH,28H,70H,20,    Inputs from  (GPIO) General Purpose Input Output pins  and transfer value
                  n                  to RS232 port.
                                      n: port number.
*/
void IOInput(unsigned char n)
{
    VFDport(0x1F);             /* 0x1F */
    VFDport(0x28);             /* 0x28 */
    VFDport(0x70);             /* 0x70 */
    VFDport(0x20);             /* 0x20 */
    VFDport(n);                /* n */

}

//-----

//3.7.3.10 Macro setting commands

```

```
//=====
//-----

/*
    RAMmacroDef          1FH,3AH,pL,pH,    Defines or deletes RAM Macro
                        dl...dk            a: Macro number
                                           pL: RAM Macro length lower byte
                                           pH: RAM Macro length upper byte
                                           dl...dk RAM Macro data
*/

void RAMmacroDef(unsigned int p,char *data)
{
    int i;
        VFDport(0x1F);          /* 0x1F */
        VFDport(0x3A);          /* 0x3A */
        VFDport(p % 0x100);
        VFDport(p / 0x100);
        for(i=0; i<p; i++)
        {
            VFDport(data[i]);    /* data */
        }

}

//-----

/*
    FROMmacroDEF         1FH,28H,65H,12H,  Define or delete FROM Macro
                        a,pL,pH,t1,t2,      a: Macro number
                        d(1)...d(p)        pL:FROM Macro lenght lower byte
                                           pH:FROM Macro lenght upper byte
                                           t1: Display time interval
                                           t2: Idle time of macro repetition
                                           dl...dk RAM Macro data
*/

void FROMmacroDEF(unsigned char a , unsigned int p, unsigned char t1, unsigned char t2, char *data)
{
    int i;
        VFDport(0x1F);          /* 0x1F */
        VFDport(0x28);          /* 0x28 */
        VFDport(0x65);          /* 0x65 */
        VFDport(0x12);          /* 0x12 */
        VFDport(a);             /* a */
        VFDport(p % 0x100);
        VFDport(p / 0x100);
        VFDport(t1);            /* t1 */
        VFDport(t2);            /* t2 */
}
```



```

    for(i=0; i<p; i++)
    {
        VFDport(data[i]);          /* data */
    }

}

//-----
/*
    RunMacro          1FH,5EH,a,t1,t2    Executes Macro continuously , Pluralize execution
                                         a: Macro processing definition number
                                         a=00H: RAM Macro 0
                                         a=01H-04H: FROM Macro 1-4
                                         a=80H: RAM Program Macro
                                         a=81H-84H: FROM Program Macro 1-4
                                         t1: Display time interval
                                         t2: idle time of macro repetition
*/
void RunMacro(unsigned char a, unsigned char t1,unsigned char t2)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x5E);          /* 0x5E */
    VFDport(a);              /* a */
    VFDport(t1);             /* t1 */
    VFDport(t2);             /* t2 */
}

//-----
//3.7.3.11 Other setting commands
//=====
//-----
/*
    MemorySWSetting   1FH,28H,65H,03H,    Sets the contents of b to Memory SW a
                    a,b                    a: Memory SW number
                                         b: Setting data
*/
void MemorySWSetting(unsigned char a ,unsigned char b)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */

```

```
    VFDport(0x65);          /* 0x65 */
    VFDport(0x03);          /* 0x03 */
    VFDport(a);             /* a */
    VFDport(b);             /* b */

}

//-----

/*
    MemorySWDataSend    1FH,28H,65H,04H,    Sends the contents of Memory SW a
                        a                  a: Memory SW number
*/
void MemorySWDataSend(unsigned char a)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x65);          /* 0x65 */
    VFDport(0x04);          /* 0x04 */
    VFDport(a);             /* a */

}

//-----

/*
    DisplayStatus      1FH,28H,65H,40H,    Send each display status information
                        a,b,c              a: Information name
                                           a=01H: Boot verion information
                                           a=02H: Firmware version information
                                           a=10H: Kanji font information
                                           a=20H: Memory Check Sum information
                                           a=30H: Product information
                                           a=40H: Display x dot Information
                                           a=41H: Display y dot Information
                                           b:Start address
                                           c:Data length
*/
void DisplayStatus(unsigned char a, unsigned char b, unsigned char c)
{
    VFDport(0x1F);          /* 0x1F */
    VFDport(0x28);          /* 0x28 */
    VFDport(0x65);          /* 0x65 */
    VFDport(0x40);          /* 0x40 */
    VFDport(a);             /* a */
    VFDport(b);             /* b */
    VFDport(c);             /* c */
}
```

```
}

//-----

/*
MemoryRewriteMode      1CH,7CH,4DH,      Shift to Memory re-write mode
                        D0H,4DH,4FH,      from normal mode
                        44H,45H,49H,
                        4EH

*/

void      MemoryRewriteMode()
{
    VFDport(0x1C);
    VFDport(0x7C);
    VFDport(0x4D);
    VFDport(0xD0);

    VFDport(0x4D);
    VFDport(0x4F);
    VFDport(0x44);
    VFDport(0x45);

    VFDport(0x49);
    VFDport(0x4E);

}

//-----
//      END OF LIB,      TEST I/O
//-----

/*-----*/
/* File : IFtest.c      */
/*-----*/

#include <stdio.h>
#include <string.h>
#include <conio.h>

typedef union bar
```

```
{
    struct
    {
        unsigned int b0:1 ;
        unsigned int b1:1 ;
        unsigned int b2:1 ;
        unsigned int b3:1 ;
        unsigned int b4:1 ;
        unsigned int b5:1 ;
        unsigned int b6:1 ;
        unsigned int b7:1 ;
    } x ;
    unsigned char ch ;
} BAR ;
```

```
int Port=0x378;
void ToDisp(unsigned char What);
void DisplayInit();
unsigned char ch;
```

```
char i;
```

```
void main(argc,argv)
int      argc;
char     **argv;
{

    DisplayInit();

    for(i=48; i<68; i++)
    {
        ToDisp(i);
    }
```

```
}
///-----//
```

```
void ToDisp( unsigned char What)
{
    BAR   Busy ;
    /* Good Practices
       Allways check for a busy signal  before writing to VFD
       if(Busy.x.b7==1) goto Again; This  is good for 3000 series display
       if(Busy.x.b7==0) goto Again; This is good for 7000 series display

    For sending SPI data the procedure is the same  like sending data to parallel port
    but we send 8 times one bit on the time.

    */

    Again:
    Busy.ch =inp(Port+1);
    if(Busy.x.b7==1) goto Again;

    outp(Port+2,0);
    outp(Port,What);
    outp(Port+2,1);
    outp(Port+2,0);
}

///-----//

void DisplayInit()
{
    ToDisp(0x1b);
    ToDisp(0x40);
}
```