

```

-----*/

/* Includes
-----*/
#include "msp430x22x4.h"

/*
IO_Configuration-----
----- */
#define SDA_H          (P1OUT |=BIT7)                // P1.7
#define SDA_L          (P1OUT &=~BIT7)
#define SCLK_H         (P1OUT |=BIT6)                // P1.6
#define SCLK_L         (P1OUT &=~BIT6)
#define nCS_H          (P1OUT |=BIT5)                // P1.5
#define nCS_L          (P1OUT &=~BIT5)
#define nDC_H          (P1OUT |=BIT4)                // P1.4
#define nDC_L          (P1OUT &=~BIT4)
#define nRST_H         (P1OUT |=BIT3)                // P1.3
#define nRST_L         (P1OUT &=~BIT3)

/* Variable
definitions-----
-*/
#define R_SDA           0x80
unsigned char tempvalue;
unsigned char temp1,temp2;

/* Display
function-----
*/
#define PIC_Black       252                          //Disply black
#define PIC_White      255                          //Disply white
#define PIC_A           1                          //Disply char
#define PIC_B           2                          //Disply lebel
#define PIC_C           3                          //Disply QR code
#define PIC_HLINE       4                          //Disply hline
#define PIC_VLINE       5                          //Disply vline
#define PIC_R           6                          //Disply rad

/* MCU delay configuration
-----*/
void Delay_ms(int delaytime)                          // delaytime=1=1ms
{
    int i,j;
    for(i=0;i<delaytime;i++)
        for(j=0;j<1600;j++);
}

void Delay_s(int delaytime)                          // delaytime=1=1s
{
    int i,j,k;
    for(i=0;i<delaytime;i++)

```

```

        for(j=0;j<1000;j++)
            for(k=0;k<1600;k++) ;
    }

/*-----*/
/* EPD
function-----*/
/*-----*/

/*IC reset
-----*/
*/
void RESET()
{
    nRST_H;
    Delay_ms(20);
    nRST_L;
    Delay_ms(20);
    nRST_H;
    Delay_ms(20);
}

/* IC read BUSY
-----*/
void Read_BUSY()
{
    while(1)
    {
        _NOP();
        if((P1IN & 0x04)==0) // Read BUSY electrica
level
        break;
    }
}

/* 4line SPI Write Command
-----*/
void SPI4W_WRITECOM(unsigned char INIT_COM)
{
    unsigned char scnt;
    P1DIR |= R_SDA;
    nCS_L;
    SCLK_L;
    nDC_L;
    for(scnt=0;scnt<8;scnt++)
    {
        if(INIT_COM&0x80)
            SDA_H;
        else
            SDA_L;
        SCLK_H;
        SCLK_L;
        INIT_COM=INIT_COM<<1;
    }
}

```

```

    }
    nCS_H;
}

/* 4line SPI Write Data
-----*/
void SPI4W_WRITEDATA(unsigned char INIT_DATA)
{
    unsigned char scnt;
    P1DIR |= R_SDA;
    nCS_L;
    SCLK_L;
    nDC_H;
    for(scnt=0;scnt<8;scnt++)
    {
        if(INIT_DATA&0x80)
            SDA_H;
        else
            SDA_L;
        SCLK_H;
        SCLK_L;
        INIT_DATA=INIT_DATA<<1;
    }
    nCS_H;
}

/* 4line SPI Read Data
-----*/
unsigned char SPI4W_READDATA()
{
    P1DIR &=~ R_SDA;
    unsigned char scnt,temp=0;
    nCS_L;
    nDC_H;
    SCLK_L;
    for(scnt=0;scnt<8;scnt++)
    {
        if(P1IN&R_SDA)
            temp=(temp<<1)|0x01;
        else
            temp=temp<<1;
        SCLK_H;
        SCLK_L;
    }
    nCS_H;
    tempvalue= temp;
    return temp;
}

/* Read EPD temperture
-----*/
void Read_temperature(void)
{
    SPI4W_WRITECOM(0x18);           // Temperature sensor control
    SPI4W_WRITEDATA(0x80);
    SPI4W_WRITECOM(0x22);           // Display update
    SPI4W_WRITEDATA(0xB1);
}

```

```

    SPI4W_WRITECOM(0x20);           // Master activation
    Read_BUSY();

    SPI4W_WRITECOM(0x1B);           // Read Temperature
    temp1=SPI4W_READDATA();
    temp2=SPI4W_READDATA();

    tempvalue=temp1;
    tempvalue=temp2;
}

/* EPD INIT
-----*/

void INIT_SSD1685()
{

    SPI4W_WRITECOM(0x01);
    //Driver output control
    SPI4W_WRITEDATA(0x7F);
    SPI4W_WRITEDATA(0x01);
    SPI4W_WRITEDATA(0x00);

    SPI4W_WRITECOM(0x11);           //
    Data entry mode setting
    SPI4W_WRITEDATA(0x01);

    SPI4W_WRITECOM(0x44);           //
    Set Ram X -address
    SPI4W_WRITEDATA(0x00);           // Start 0

    SPI4W_WRITEDATA(0x16);           // End
    0x16=22 (22+1)x8=184

    SPI4W_WRITECOM(0x45);
    //set Ram y -address
    SPI4W_WRITEDATA(0x7F);
    SPI4W_WRITEDATA(0x01);
    SPI4W_WRITEDATA(0x00);           // RAM y
    address end at 00h;
    SPI4W_WRITEDATA(0x00);

    SPI4W_WRITECOM(0x3C);           //
    Set border
    SPI4W_WRITEDATA(0x01);           // 0x01
    border white 0x00 black 0x07 rad

    SPI4W_WRITECOM(0x21);           //
    Display update control
    SPI4W_WRITEDATA(0x00);
    SPI4W_WRITEDATA(0x40);

```

```

}

/* IC Sleep
-----*/
void Enter_deep_sleep()
{
    SPI4W_WRITECOM(0x10); //
    Deep sleep mode
    SPI4W_WRITEDATA(0x01);
}

/* Display configuration
-----*/

void Dis_img(unsigned char num)
{
    unsigned int row, col;
    unsigned int pcnt;

/* write bw
data-----*/

    SPI4W_WRITECOM(0x4E);
    SPI4W_WRITEDATA(0x00);
    SPI4W_WRITECOM(0x4F);
    SPI4W_WRITEDATA(0x7F);
    SPI4W_WRITEDATA(0x01);
    SPI4W_WRITECOM(0x24);
    pcnt = 0;
    for(col=0; col<384; col++)
    {
        for(row=0; row<23; row++)
        {
            switch (num)
            {
                case PIC_A:
                    SPI4W_WRITEDATA(Image_BW[pcnt]);
                    break;

                case PIC_White:
                    SPI4W_WRITEDATA(0xff);
                    break;
                case PIC_Black:
                    SPI4W_WRITEDATA(0x00);
                    break;

                default:
                    break;
            }
            pcnt++;
        }
    }
}

```

```

SPI4W_WRITECOM(0x4E);
SPI4W_WRITEDATA(0x00);
SPI4W_WRITECOM(0x4F);
SPI4W_WRITEDATA(0x7F);
SPI4W_WRITEDATA(0x01);
SPI4W_WRITECOM(0x26);
pcnt = 0;
for(col=0; col<384; col++)
{
    for(row=0; row<23; row++)
    {
        switch (num)
        {
            case PIC_A:
                SPI4W_WRITEDATA(Image_BW[pcnt]);
                break;
            case PIC_White:
                SPI4W_WRITEDATA(0xff);
                break;
            case PIC_Black:
                SPI4W_WRITEDATA(0x00);
                break;
            default:
                break;
        }
        pcnt++;
    }
}

SPI4W_WRITECOM(0x18);
SPI4W_WRITEDATA(0x80);

SPI4W_WRITECOM(0x22);
SPI4W_WRITEDATA(0xF7); //Load LUT from MCU(0x32), Display update
SPI4W_WRITECOM(0x20);
Delay_ms(10);
Read_BUSY();

Delay_ms(20);
}

void Dis_img_fast(unsigned char num)
{
    unsigned int row, col;
    unsigned int pcnt;

    /* write bw
data-----*/

    SPI4W_WRITECOM(0x4E);
    SPI4W_WRITEDATA(0x00);
    SPI4W_WRITECOM(0x4F);
    SPI4W_WRITEDATA(0x7F);
    SPI4W_WRITEDATA(0x01);
    SPI4W_WRITECOM(0x24);
    pcnt = 0;

```

```

for(col=0; col<384; col++)
{
    for(row=0; row<23; row++)
    {
        switch (num)
        {
            case PIC_B:
                SPI4W_WRITEDATA(Image_BW1[pcnt]);
                break;

            case PIC_White:
                SPI4W_WRITEDATA(0xff);
                break;
            case PIC_Black:
                SPI4W_WRITEDATA(0x00);
                break;

            default:
                break;
        }
        pcnt++;
    }
}

SPI4W_WRITECOM(0x3C); // Set border
SPI4W_WRITEDATA(0xC0); // 0x01 border white 0x00 border black
0x07 border rad

SPI4W_WRITECOM(0x18);
SPI4W_WRITEDATA(0x80);

SPI4W_WRITECOM(0x22);
SPI4W_WRITEDATA(0xFF); //Load LUT from MCU(0x32), Display update
SPI4W_WRITECOM(0x20);
Delay_ms(10);
Read_BUSY();

Delay_ms(20);
}

void dis_partial_img(unsigned char num)
{
    unsigned int row, col;
    unsigned int pcnt;

    //-----Test bw
    image-----

    SPI4W_WRITECOM(0x44); //
    set RAM x address start/end,
    SPI4W_WRITEDATA(0x0D); // RAM x
    address start at 00h;
    SPI4W_WRITEDATA(0x13); // RAM x
    address end at 18h(24+1)*8->200

```

```

    SPI4W_WRITECOM(0x45); //
    set RAM y address start/end, in page 37
    SPI4W_WRITEDATA(0x2C); // RAM y
    address start at 200h;
    SPI4W_WRITEDATA(0x00);
    SPI4W_WRITEDATA(0x0A); // RAM y
    address end at 00h;
    SPI4W_WRITEDATA(0x00);

    SPI4W_WRITECOM(0x4E); //
    set RAM x address
    SPI4W_WRITEDATA(0x0D);
    SPI4W_WRITECOM(0x4F); //
    set RAM y address
    SPI4W_WRITEDATA(0x2C);
    SPI4W_WRITEDATA(0x00);

    SPI4W_WRITECOM(0x24);

    pcnt = 0;
    for(col=0; col<35; col++)
    {
        for(row=0; row<7; row++)
        {
            switch (num)
            {
                case 2:
                    SPI4W_WRITEDATA(~gImage_102[pcnt]);
                    break;

                case 3:
                    SPI4W_WRITEDATA(~gImage_103[pcnt]);
                    break;

                case 4:
                    SPI4W_WRITEDATA(~gImage_104[pcnt]);
                    break;

                case 5:
                    SPI4W_WRITEDATA(~gImage_105[pcnt]);
                    break;

                default:
                    break;
            }
            pcnt++;
        }
    }

    SPI4W_WRITECOM(0x3C); // Set border
    SPI4W_WRITEDATA(0xC0); // 0x01 border white 0x00 border black 0x07
    border rad

    //LED_ON;

```

```
SPI4W_WRITECOM(0x18);  
SPI4W_WRITEDATA(0x80); //Select  
internal temperature sensor  
  
SPI4W_WRITECOM(0x22);  
SPI4W_WRITEDATA(0xFF); //Load LUT  
from OTP  
SPI4W_WRITECOM(0x20);  
DELAY_ms(10);  
READBUSY();  
  
Delay_ms(20);  
}  
  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
//xx Main Function  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
void main( void )  
{  
    // int i=0;  
  
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog  
timer to prevent time out reset  
    BCSCTL1 = CALBC1_8MHZ; // set DCO  
frequency 1MHZ  
    DCOCTL = CALDCO_8MHZ;  
  
    P1DIR |=0x78; // set P1.3~7  
output  
    P3DIR = 0xff; // set P3  
output  
    P4DIR |= 0X01; // set P4.1  
output  
  
    //SLOW DISPLAY  
    RESET();  
    SPI4W_WRITECOM(0x12);  
    Read_BUSY();  
    INIT_SSD1685();  
    Dis_img(PIC_A);  
    Enter_deep_sleep();  
    Delay_s(2);  
    _NOP();  
  
    //FAST DISPLAY  
    RESET();  
    Read_BUSY();  
    Dis_img_fast(PIC_B);  
    Enter_deep_sleep();  
    Delay_s(2);  
    NOP();
```

```
//SLOW DISPLAY
    RESET();
    SPI4W_WRITECOM(0x12);
    Read_BUSY();
    INIT_SSD1685();
    Dis_img(PIC_A);
    Enter_deep_sleep();
    Delay_s(2);
    _NOP();

//partial
    RESET();
    Read_BUSY();
    dis_partial_img(PIC_1);
    Enter_deep_sleep();
    Delay_s(2);
    _NOP();

    RESET();
    Read_BUSY();
    dis_partial_img(PIC_2);
    Enter_deep_sleep();
    Delay_s(2);
    _NOP();
}
```