```
//Note: For the best effect, we suggest you view the program with the Source Insight
software.

//
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx




//xx Includes
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxx
#include "msp430x22x4.h"
#include "image.h"



//xx Private macro
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxx

#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)

#define R_SDA         0x80   //P1.7

#define DELAY_TIME 1                                            // The Time after image
remains on screen (unit: seconds)

#define MODE1                                                       // panel scan
direction

unsigned char tempvalue;
unsigned char temp1,temp2;
unsigned char temp3,temp4;

#define PIC_BLACK       252
#define PIC_WHITE       255
#define PIC_A           1
#define PIC_B          2
```

```
#define PIC_HLINE       3
#define PIC_VLINE      4
#define PIC_C           5
#define PIC_D          6
#define PIC_E          7
#define PIC_R         8
//xx Private functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxx


//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
//xx   Delay Function
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxx
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
void DELAY_100nS(int delaytime)                                // 30us
{
     int i,j;

     for(i=0;i<delaytime;i++)
          for(j=0;j<1;j++);
}

void DELAY_mS(int delaytime)                              // 1ms
{
     int i,j;

     for(i=0;i<delaytime;i++)
          for(j=0;j<200;j++);
}

void DELAY_S(int delaytime)                               // 1s
{
     int i,j,k;

     for(i=0;i<delaytime;i++)
          for(j=0;j<1000;j++)
               for(k=0;k<200;k++);
}



//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
//xx   E-paper Driver Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx

// Reset
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxx
void RESET()
{
     nRST_L;
     DELAY_mS(1);
     nRST_H;
     DELAY_mS(1);
}

// Check busy
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxx
void READBUSY()
{
     while(1)
     {
          _NOP();
          if((P1IN & 0x04)==0)
          break;
     }
}

// Enter command xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void SPI4W_WRITECOM(unsigned char INIT_COM)
{
     unsigned char TEMPCOM;
     unsigned char scnt;
   P1DIR |= R_SDA;
     TEMPCOM=INIT_COM;
     nCS_H;
     nCS_L;
     SCLK_L;
     nDC_L;
     for(scnt=0;scnt<8;scnt++)
     {
          if(TEMPCOM&0x80)
               SDA_H;
          else
               SDA_L;
          SCLK_H;
          SCLK_L;
          TEMPCOM=TEMPCOM<<1;
     }
     //nCS_H;
```

```
}

// Enter data
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
void SPI4W_WRITEDATA(unsigned char INIT_DATA)
{
     unsigned char TEMPCOM;
     unsigned char scnt;
     P1DIR |= R_SDA;

     TEMPCOM=INIT_DATA;
     //nCS_H;
     //nCS_L;
     SCLK_L;
     nDC_H;
     for(scnt=0;scnt<8;scnt++)
     {
          if(TEMPCOM&0x80)
          SDA_H;
          else
          SDA_L;
          SCLK_H;
          SCLK_L;
          TEMPCOM=TEMPCOM<<1;
     }
     //nCS_H;
}


// Read Data
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
 unsigned char SPI4W_READDATA1()
 {
      P1DIR &=~ R_SDA;

      unsigned char scnt,temp;
         temp=0;

     // nCS_H;
     // nCS_L;
     // SCLK_H;
      nDC_H;
      for(scnt=0;scnt<8;scnt++)
      {

           SCLK_L;
           if(P1IN&R_SDA)
```

```
        temp=(temp<<1)|0x01;
        else
        temp=temp<<1;
        SCLK_H;
        SCLK_L;
    }
   // nCS_H;
       tempvalue= temp;
    return temp;
}

void read_temperture(void)
{
  //The "busy" line must be low before calling this function
  unsigned char temp1,temp2;
  SPI4W_WRITECOM(0x18);
  SPI4W_WRITEDATA(0X80);
  SPI4W_WRITECOM(0x22);
  SPI4W_WRITEDATA(0XB1);
  SPI4W_WRITECOM(0x20);
    nCS_H;
  READBUSY();

  SPI4W_WRITECOM(0x1B);
   temp1=SPI4W_READDATA1();
   temp2=SPI4W_READDATA1();

  tempvalue=temp1;
  tempvalue=temp2;
    nCS_H;
}

// E-Paper Driver Initialization
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxx

void INIT_SSD1680()
{

        SPI4W_WRITECOM(0x01);
        SPI4W_WRITEDATA(0xD3);
        SPI4W_WRITEDATA(0x00);
        SPI4W_WRITEDATA(0x00);

        SPI4W_WRITECOM(0x11);       // data enter mode
        SPI4W_WRITEDATA(0x01);

        SPI4W_WRITECOM(0x44);       // set RAM x address start/end, in page 36
        SPI4W_WRITEDATA(0x00);
```

```
        SPI4W_WRITEDATA(0x11);          // RAM x address start at 104/8=13-1=12

        SPI4W_WRITECOM(0x45);
        SPI4W_WRITEDATA(0xC7);
        SPI4W_WRITEDATA(0x00);
        SPI4W_WRITEDATA(0x00);       // RAM y address end at 00h;
        SPI4W_WRITEDATA(0x00);

        SPI4W_WRITECOM(0x3C);        // board
        SPI4W_WRITEDATA(0x05);       //GS1-->GS1

        SPI4W_WRITECOM(0x21);        //  Display update control
        SPI4W_WRITEDATA(0x00);
        SPI4W_WRITEDATA(0x80);

              nCS_H;

}


//
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxx
void enterdeepsleep()
{

       SPI4W_WRITECOM(0x10);
     SPI4W_WRITEDATA(0x01);
        nCS_H;

}

void dis_img_fast(unsigned char num)
{
     unsigned int row, col;
     unsigned int pcnt;
/*****************************************************bw
image*****************************************************/
     SPI4W_WRITECOM(0x4E);
     SPI4W_WRITEDATA(0x00);
     SPI4W_WRITECOM(0x4F);
     SPI4W_WRITEDATA(0xC7);
     SPI4W_WRITEDATA(0x00);

     SPI4W_WRITECOM(0x24);

     pcnt = 0;                                          // Reset or Save Prompt
Byte Number
```

```
    for(col=0; col<200; col++)                                  // 172 columns in total
    // send 128x252bits ram 2D13
    {
        for(row=0; row<18; row++)                               // A total of 72 lines,
with 2 bits per pixel, or 72/4 bytes
        {
                switch (num)
                {
                case PIC_A:

                                    SPI4W_WRITEDATA(gImage_1D3BW[pcnt]);
                    break;


                case PIC_VLINE:
                     if(col<66)
                    SPI4W_WRITEDATA(0x00);
                        else
                      SPI4W_WRITEDATA(0xff);

                 break;

                   case PIC_HLINE:
                 if(row<6)
                     SPI4W_WRITEDATA(0x00);
                    else
                     SPI4W_WRITEDATA(0xff);
                 break;

                case PIC_WHITE:
                    SPI4W_WRITEDATA(0xff);
                    break;

                case PIC_BLACK:
                    SPI4W_WRITEDATA(0x00);
                    break;
                default:
                    break;
                }
            pcnt++;
        }
    }

    SPI4W_WRITECOM(0x18);        //  Display update control
    SPI4W_WRITEDATA(0x80);


    SPI4W_WRITECOM(0x22);
    SPI4W_WRITEDATA(0xFF);       //Load LUT from MCU(0x32), Display update
```

```c
	SPI4W_WRITECOM(0x20);
	DELAY_mS(1);
	READBUSY();


}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
//xx   Image Display Function
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxx
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
void dis_img(unsigned char num)
{
	unsigned int row, col;
	unsigned int pcnt;

/***************************************************bw
image*****************************************************/
	SPI4W_WRITECOM(0x4E);
	SPI4W_WRITEDATA(0x00);
	SPI4W_WRITECOM(0x4F);
	SPI4W_WRITEDATA(0xC7);
	SPI4W_WRITEDATA(0x00);

	SPI4W_WRITECOM(0x24);

    pcnt = 0;                                                     // Reset or Save Prompt
Byte Number
	for(col=0; col<200; col++)                                // 172 columns in total
	// send 128x252bits ram 2D13
	{
		for(row=0; row<18; row++)                             // A total of 72 lines,
with 2 bits per pixel, or 72/4 bytes
		{
			switch (num)
			{
			case PIC_A:

						SPI4W_WRITEDATA(gImage_1D3BW[pcnt]);
				break;


			case PIC_VLINE:
				 if(col<66)
				SPI4W_WRITEDATA(0x00);
				    else
```

```
                    SPI4W_WRITEDATA(0xff);

                 break;

                    case PIC_HLINE:
                 if(row<6)
                     SPI4W_WRITEDATA(0x00);
                    else
                     SPI4W_WRITEDATA(0xff);
                 break;

                case PIC_WHITE:
                    SPI4W_WRITEDATA(0xff);
                    break;

                case PIC_BLACK:
                    SPI4W_WRITEDATA(0x00);
                    break;
                default:
                    break;
                }
            pcnt++;
        }
    }

    SPI4W_WRITECOM(0x26);

   pcnt = 0;                                                    // Reset or Save Prompt
Byte Number
    for(col=0; col<200; col++)                                  // 172 columns in total
    // send 128x252bits ram 2D13
    {
        for(row=0; row<18; row++)                               // A total of 72 lines,
with 2 bits per pixel, or 72/4 bytes
        {
                switch (num)
                {
                case PIC_A:

                 SPI4W_WRITEDATA(gImage_1D3BW[pcnt]);
                    break;


                case PIC_VLINE:
                     if(col<66)
                    SPI4W_WRITEDATA(0x00);
                        else
                      SPI4W_WRITEDATA(0xff);
```

```
                    break;

                       case PIC_HLINE:
                    if(row<6)
                         SPI4W_WRITEDATA(0x00);
                        else
                         SPI4W_WRITEDATA(0xff);
                    break;

                   case PIC_WHITE:
                        SPI4W_WRITEDATA(0xff);
                        break;

                   case PIC_BLACK:
                        SPI4W_WRITEDATA(0x00);
                        break;
                   default:
                        break;
                   }
               pcnt++;
          }
     }

     SPI4W_WRITECOM(0x18);         //  Display update control
     SPI4W_WRITEDATA(0x80);


    SPI4W_WRITECOM(0x22);
     SPI4W_WRITEDATA(0xF7);       //Load LUT from MCU(0x32), Display update
     SPI4W_WRITECOM(0x20);
     DELAY_mS(1);
     READBUSY();


}



//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx
//xx   Main Function
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxx
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx
void main( void )
{
```

```
        int i=10;



        WDTCTL = WDTPW + WDTHOLD;                                   // Stop watchdog
timer to prevent time out reset
        BCSCTL1 = CALBC1_1MHZ;                                      // set DCO frequency
1MHZ
        DCOCTL = CALDCO_1MHZ;


        P1DIR |=0x78;                                                   // set P1.3~7
output
        P3OUT = 0XFF;
        P3DIR = 0xff;                                               // p3.4,p3.5 Set as
Input
        P4DIR |= 0X01;



        //slow  display
        RESET();                                    // E-paper Controller Reset
        SPI4W_WRITECOM(0x12);                       //SWRESET
        READBUSY();
        INIT_SSD1680();
        dis_img(PIC_WHITE);
        enterdeepsleep();
        DELAY_S(5);
        _NOP();


        //fast display
        RESET();                                    // E-paper Controller Reset
        READBUSY();
        dis_img_fast(PIC_BLACK);
        enterdeepsleep();
        DELAY_S(5);
        _NOP();


        while(1);


}
```