

```

#include "msp430x22x4.h"
#include "image.h"

#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 nRST_H     (P1OUT |=BIT3)           //
P1.3
#define nRST_L     (P1OUT &=~BIT3)
#define nDC_H      (P1OUT |=BIT4)           //
P1.4
#define nDC_L      (P1OUT &=~BIT4)

#define CSB_H      (P1OUT |=BIT5)           //
P1.5
#define CSB_L      (P1OUT &=~BIT5)
#define CSB2_H     (P1OUT |=BIT1)           //
P1.1
#define CSB2_L     (P1OUT &=~BIT1)

#define R_SDA      0x80
//P1.7

#define MS_H       (P2OUT |= BIT3)
// P2.3
#define MS_L       (P2OUT &= ~BIT3)

#define MASTER_SLAVE      0
#define MASTER_ONLY       1
#define SLAVE_ONLY        2

#define TEMPTR_ON         0xFF
#define TEMPTR_OFF        0

#define WHITE              0x11
#define BLACK              0x00
#define RED                0x33
#define YELLOW             0x22
#define BLUE               0x55
#define GREEN              0x66

```

```

#define PIC_HALF          0xFC          //Half an image
#define PIC_A             0xFD          //A whole image
#define STRIPE            0xFE
#define IMAGE              0xFF

```

```

unsigned char User_Read_Data[128] = {0};
unsigned char Temptr_Cur = 0;
unsigned char OTP_PWR[5] = {0};
    unsigned char temp, temp1, temp2;

```

```

/*****
*Function-Delay function (mS)
*Parameters-time: mS
*Back-void
*****/
void Delay_ms(int time)
// 1ms
{
    int i, j;

    for(i=0; i<time; i++)
        for(j=0; j<1600; j++);
}

```

```

/*****
*Function-Delay function(S)
*Parameters-time: S
*Back-void
*****/
void Delay_S(int time)
// 1s
{
    int i, j, k;

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

```

```

/*****
*Functions - System Initialization
*Parameters-void
*Back-void
*****/
void Sys_init(void)

```

```

{
    WDTCTL = WDTPW + WDTHOLD;
    // Stop watchdog timer to prevent time out reset
    BCSCTL1 = CALBC1_8MHZ;
    set DCO frequency 8MHZ
    DCOCTL = CALDCO_8MHZ;

    P1DIR |=0xFA;
    // set P1.3~7 output
    P2DIR |=0x08;
    // set P1.3~7 output
    P4DIR |=0x0A;
    P4.3 P4.1 output    0A should berevised
}

void ReadBusy(void)
{
    while(1)
    {
        Delay_ms(2);
        if(P1IN & 0x04)    // It will indicate idle at a high level
            break;
    }
}

void Reset(void)
{
    nRST_H;
    Delay_ms(30);
    nRST_L;
    Delay_ms(30);
    nRST_H;
    Delay_ms(10);
}

void SPI3W_WriteCom(unsigned char init_com)
{
    unsigned char scnt;

    SCLK_L;
    _NOP();_NOP();
    SDA_L;
    _NOP();_NOP();
    SCLK_H;
    _NOP();_NOP();
    SCLK_L;

```

```

_NOP();_NOP();
for(scnt=0; scnt<8; scnt++)
{
    if(init_com & 0x80)
        SDA_H;
    else
        SDA_L;
    _NOP();
    SCLK_H;
    _NOP();
    SCLK_L;
    init_com = init_com << 1;
}
SDA_L;
_NOP();
}

void SPI3W_WriteData(unsigned char init_data)
{
    unsigned char scnt;

    SCLK_L;
    _NOP();_NOP();
    SDA_H;
    _NOP();_NOP();
    SCLK_H;
    _NOP();_NOP();
    SCLK_L;
    _NOP();
    for(scnt=0; scnt<8; scnt++)
    {
        if(init_data & 0x80)
            SDA_H;
        else
            SDA_L;
        _NOP();
        SCLK_H;
        _NOP();
        SCLK_L;
        init_data = init_data << 1;
    }
    SDA_L;
    _NOP();
}

unsigned char SPI3W_ReadData(void)
{
    unsigned char scnt,temp = 0x00;

```

```

    SCLK_L;
    _NOP();_NOP();
    SDA_H;
    _NOP();_NOP();
    SCLK_H;
    _NOP();_NOP();
    SCLK_L;
    _NOP();
    P1DIR &= ~R_SDA;
    _NOP();_NOP();_NOP();
    for(scnt=0; scnt<8; scnt++)
    {
        temp = temp << 1;
        SCLK_H;
        if(P1IN&R_SDA)
            temp |= 0x01;
        SCLK_L;
        _NOP();
    }
    P1DIR |= R_SDA;
    SDA_L;

    _NOP();

    return temp;
}

void MsDev_WriteCom(unsigned char MS_opt, unsigned char init_com)
{
    if(1 == MS_opt)
    {
        CSB_L; CSB2_H;
    }
    else if(2 == MS_opt)
    {
        CSB_H; CSB2_L;
    }
    else
    {
        CSB_L; CSB2_L;
    }

    _NOP();

    SPI3W_WriteCom(init_com);

    CSB_H; CSB2_H;

```

```
}
```

```
void MsDev_WriteData(unsigned char MS_opt, unsigned char init_data)
```

```
{
```

```
    if(1 == MS_opt)
```

```
    {
```

```
        CSB_L; CSB2_H;
```

```
    }
```

```
    else if(2 == MS_opt)
```

```
    {
```

```
        CSB_H; CSB2_L;
```

```
    }
```

```
    else
```

```
    {
```

```
        CSB_L; CSB2_L;
```

```
    }
```

```
    SPI3W_WriteData(init_data);
```

```
    CSB_H; CSB2_H;
```

```
}
```

```
unsigned char MsDev_ReadData(unsigned char MS_opt)
```

```
{
```

```
    unsigned char temp = 0x00;
```

```
    if(1 == MS_opt)
```

```
    {
```

```
        CSB_L; CSB2_H;
```

```
    }
```

```
    else if(2 == MS_opt)
```

```
    {
```

```
        CSB_H; CSB2_L;
```

```
    }
```

```
    else
```

```
    {
```

```
        CSB_L; CSB2_L;
```

```
    }
```

```
    temp = SPI3W_ReadData();
```

```
    CSB_H; CSB2_H;
```

```
    return temp;
```

```
}
```

```
unsigned char ReadTemptr(void)
```

```
{
```

```
    unsigned char temptr_intgr = 0;
```

```

    unsigned char temptr_decml = 0;

    MsDev_WriteCom(MASTER_ONLY, 0x40);
    Delay_ms(100);
    ReadBusy();
    temptr_intgr = MsDev_ReadData(MASTER_ONLY);
    temptr_decml = MsDev_ReadData(MASTER_ONLY);

    Temptr_Cur = temptr_intgr;

    return temptr_intgr;
}

void WriteTemptr(unsigned char temptr_lock)
{
    MsDev_WriteCom(MASTER_SLAVE, 0xE0);
    MsDev_WriteData(MASTER_SLAVE, 0x03);

    MsDev_WriteCom(MASTER_SLAVE, 0xE5);
    MsDev_WriteData(MASTER_SLAVE, temptr_lock);
    ReadBusy();
}

void Read_OTP_PWR(unsigned char temptr_opt)
{
    unsigned long i, j;
    unsigned char OTP_VCOM;
    unsigned char temptr_val = 0;

    MS_H;
    Reset();
    MS_L;

    MsDev_WriteCom(MASTER_SLAVE, 0x00);
    MsDev_WriteData(MASTER_SLAVE, 0x0F);
    MsDev_WriteData(MASTER_SLAVE, 0x69);

    ReadTemptr();

    if(temptr_opt > 0 && temptr_opt != TEMPTR_ON)
    {
        temptr_val = temptr_opt;
    }
}

```

```

}
else
{
    temptr_val = Temptr_Cur;
}

MS_H;
_NOP();

MsDev_WriteCom(MASTER_SLAVE, 0x00);
MsDev_WriteData(MASTER_SLAVE, 0x0F);
MsDev_WriteData(MASTER_SLAVE, 0x69);

MsDev_WriteCom(MASTER_SLAVE, 0x01);
MsDev_WriteData(MASTER_SLAVE, 0x00);

WriteTemptr(temptr_val);

MsDev_WriteCom(MASTER_SLAVE, 0x04);
ReadBusy();
Delay_ms(10);

MsDev_WriteCom(MASTER_SLAVE, 0x02);
MsDev_WriteData(MASTER_SLAVE, 0x00);
ReadBusy();
Delay_ms(10);

MS_L;
_NOP();

MsDev_WriteCom(MASTER_SLAVE, 0xF0);
MsDev_ReadData(MASTER_ONLY);

for(j=0; j<207; j++)
    MsDev_ReadData(MASTER_ONLY);

OTP_VCOM = MsDev_ReadData(MASTER_ONLY);

MsDev_WriteCom(MASTER_SLAVE, 0xF5);
MsDev_WriteData(MASTER_SLAVE, 0xA5);

MsDev_WriteCom(MASTER_SLAVE, 0x94);
MsDev_ReadData(MASTER_ONLY);

for(i=0; i<5; i++)
{
    OTP_PWR[i] = MsDev_ReadData(MASTER_ONLY);
}

```

```
MsDev_WriteCom(MASTER_SLAVE, 0xF5);  
MsDev_WriteData(MASTER_SLAVE, 0x00);
```

```
MS_H;  
Reset();  
MS_L;
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x66);  
MsDev_WriteData(MASTER_SLAVE, 0x49);  
MsDev_WriteData(MASTER_SLAVE, 0x55);  
MsDev_WriteData(MASTER_SLAVE, 0x13);  
MsDev_WriteData(MASTER_SLAVE, 0x5D);  
MsDev_WriteData(MASTER_SLAVE, 0x05);  
MsDev_WriteData(MASTER_SLAVE, 0x10);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x13);  
MsDev_WriteData(MASTER_SLAVE, 0x00);  
MsDev_WriteData(MASTER_SLAVE, 0x00);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0xE0);  
MsDev_WriteData(MASTER_SLAVE, 0x01);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x00);  
MsDev_WriteData(MASTER_SLAVE, 0x13);  
MsDev_WriteData(MASTER_SLAVE, 0xE9);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x01);  
MsDev_WriteData(MASTER_SLAVE, 0x0F);  
MsDev_WriteData(MASTER_SLAVE, OTP_PWR[0]);  
MsDev_WriteData(MASTER_SLAVE, OTP_PWR[1]);  
MsDev_WriteData(MASTER_SLAVE, OTP_PWR[2]);  
MsDev_WriteData(MASTER_SLAVE, OTP_PWR[3]);  
MsDev_WriteData(MASTER_SLAVE, OTP_PWR[4]);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x06);  
MsDev_WriteData(MASTER_SLAVE, 0xD7);  
MsDev_WriteData(MASTER_SLAVE, 0xDE);  
MsDev_WriteData(MASTER_SLAVE, 0x12);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x61);  
MsDev_WriteData(MASTER_SLAVE, 0x00);  
MsDev_WriteData(MASTER_SLAVE, 0xC8);  
MsDev_WriteData(MASTER_SLAVE, 0x01);  
MsDev_WriteData(MASTER_SLAVE, 0x90);
```

```
MsDev_WriteCom(MASTER_SLAVE, 0x82);  
MsDev_WriteData(MASTER_SLAVE, OTP_VCOM);
```

```

    MsDev_WriteCom(MASTER_SLAVE, 0xE3);
    MsDev_WriteData(MASTER_SLAVE, 0x01);

    MsDev_WriteCom(MASTER_SLAVE, 0xE9);
    MsDev_WriteData(MASTER_SLAVE, 0x01);

}

void EnterDeepsleep(void)
{
    MsDev_WriteCom(MASTER_SLAVE, 0x07);
    MsDev_WriteData(MASTER_SLAVE, 0xA5);
}

unsigned char Read_OtpData(unsigned char MS_opt)
{
    unsigned int i;

    Reset();
    ReadBusy();

    MsDev_WriteCom(MS_opt, 0x92);
    MsDev_ReadData(MS_opt);
    for(i=0; i<sizeof(User_Read_Data); i++)
    {
        User_Read_Data[i] = MsDev_ReadData(MS_opt);
    }

    return User_Read_Data[1];
}

static void Send_Image_Data(void)
{
    unsigned int i, j, num, cnt, data;

    //    MsDev_WriteCom(MASTER_ONLY, 0x00);
    //    MsDev_WriteData(MASTER_ONLY, 0x13);
    //    MsDev_WriteData(MASTER_ONLY, 0xE9);
    //
    //    MsDev_WriteCom(MASTER_ONLY, 0x10);
    //    Delay_ms(10);
    //
    //    for(num =0, i=0; i<sizeof(Decrypt_Master_Image); i++, num++)
    //    {
    //        cnt = Decrypt_Master_Cnt[num];
    //        data = Decrypt_Master_Image[num];

```

```

//      for(j=0; j<cnt; j++)
//      {
//          MsDev_WriteData(MASTER_ONLY, data);
//      }
//
//
//
//
//
//      MsDev_WriteCom(SLAVE_ONLY, 0x00);
//      MsDev_WriteData(SLAVE_ONLY, 0x17);
//      MsDev_WriteData(SLAVE_ONLY, 0xE9);
//
//      MsDev_WriteCom(SLAVE_ONLY, 0x10);
//      Delay_ms(10);
//
//      for(num =0, i=0; i<sizeof(Decrypt_Slave_Image); i++, num++)
//      {
//
//          cnt = Decrypt_Slave_Cnt[num];
//          data = Decrypt_Slave_Image[num];
//          for(j=0; j<cnt; j++)
//          {
//              MsDev_WriteData(SLAVE_ONLY, data);
//          }
//      }
//
}

static void Send_HV_Stripe_Data(void)
{
    unsigned int col, row;

    MsDev_WriteCom(MASTER_ONLY, 0x00);
    MsDev_WriteData(MASTER_ONLY, 0x13);
    MsDev_WriteData(MASTER_ONLY, 0xE9);

    MsDev_WriteCom(MASTER_ONLY, 0x10);
    Delay_ms(10);
    for(col=0; col<400; col++)
    {
        for(row=0; row<100; row++)
        {
            if(col >= 82 && col < 200 && row >= 10 && row <= 36)
            {
                MsDev_WriteData(MASTER_ONLY, 0x11);
            }
            else if(col >= 82 && col < 200 && row >36 && row <= 62)
            {
                MsDev_WriteData(MASTER_ONLY, 0x22);
            }
        }
    }
}

```

```

    }
    else if(col >= 82 && col < 200 && row > 62 && row <= 89)
    {
        MsDev_WriteData(MASTER_ONLY, 0x66);
    }
    else if(col >= 200 && col < 318 && row >= 10 && row <= 36)
    {
        MsDev_WriteData(MASTER_ONLY, 0x00);
    }
    else if(col >= 200 && col < 318 && row > 36 && row <= 62)
    {
        MsDev_WriteData(MASTER_ONLY, 0x55);
    }
    else if(col >= 200 && col < 318 && row > 62 && row <= 89)
    {
        MsDev_WriteData(MASTER_ONLY, 0x33);
    }
    else
    {
        MsDev_WriteData(MASTER_ONLY, 0x11);
    }
}
}

```

```

MsDev_WriteCom(SLAVE_ONLY, 0x00);
MsDev_WriteData(SLAVE_ONLY, 0x17);
MsDev_WriteData(SLAVE_ONLY, 0xE9);

```

```

MsDev_WriteCom(SLAVE_ONLY, 0x10);
Delay_ms(10);
for(col=0; col<400; col++)
{
    for(row=0; row<100; row++)
    {
        if(col >= 82 && col < 200 && row >= 10 && row <= 36)
        {
            MsDev_WriteData(SLAVE_ONLY, 0x11); //1 WHITE
        }
        else if(col >= 82 && col < 200 && row > 36 && row <= 62)
        {
            MsDev_WriteData(SLAVE_ONLY, 0x22); // 2 YELLOW
        }
        else if(col >= 82 && col < 200 && row > 62 && row <= 89)
        {
            MsDev_WriteData(SLAVE_ONLY, 0x66); //6 GREEN
        }
        else if(col >= 200 && col < 318 && row >= 10 && row <= 36)
        {
            MsDev_WriteData(SLAVE_ONLY, 0x00); // 0 BLACK
        }
    }
}

```

```

    }
    else if(col >= 200 && col < 318 && row > 36 && row <= 62)
    {
        MsDev_WriteData(SLAVE_ONLY, 0x55); // 5 BLUE
    }
    else if(col >= 200 && col < 318 && row > 62 && row <= 89)
    {
        MsDev_WriteData(SLAVE_ONLY, 0x33); // 3 RED
    }
    else
    {
        MsDev_WriteData(SLAVE_ONLY, 0x11);
    }
}
}
}

```

```

static void Send_Color_Data(unsigned char color)
{
    unsigned int col, row;

    MsDev_WriteCom(MASTER_ONLY, 0x00);
    MsDev_WriteData(MASTER_ONLY, 0x13);
    MsDev_WriteData(MASTER_ONLY, 0xE9);

    MsDev_WriteCom(MASTER_ONLY, 0x10);
    Delay_ms(10);
    for(col=0; col<400; col++)
    {
        for(row=0; row<100; row++)
        {
            MsDev_WriteData(MASTER_ONLY, color);
        }
    }

    MsDev_WriteCom(SLAVE_ONLY, 0x00);
    MsDev_WriteData(SLAVE_ONLY, 0x17);
    MsDev_WriteData(SLAVE_ONLY, 0xE9);

    MsDev_WriteCom(SLAVE_ONLY, 0x10);
    Delay_ms(10);
    for(col=0; col<400; col++)
    {
        for(row=0; row<100; row++)
        {
            MsDev_WriteData(SLAVE_ONLY, color);
        }
    }
}

```

```
}
```

```
static void Send_Color_Data_half_A(unsigned char color)
{
```

```
    unsigned int col, row;
    unsigned int pcnt;
```

```
    MsDev_WriteCom(MASTER_ONLY, 0x00);
    MsDev_WriteData(MASTER_ONLY, 0x13);
    MsDev_WriteData(MASTER_ONLY, 0xE9);
```

```
    MsDev_WriteCom(MASTER_ONLY, 0x10);
    Delay_ms(10);
    pcnt = 0;
```

```
    for(col=0; col<400; col++)
    {
```

```
        for(row=0; row<50; row++)
        {
```

```
            temp1 = (DTM1[col*100+row*2]&0xF0);
            temp2 = (DTM1[col*100+row*2+1]>>4);
```

```
            temp = temp1|temp2;
```

```
            MsDev_WriteData(MASTER_ONLY, temp);
```

```
        }
```

```
    for(row=0; row<50; row++)
    {
```

```
        temp1 = (DTM1[col*100+row*2]&0xF0);
        temp2 = (DTM1[col*100+row*2+1]>>4);
```

```
        temp = temp1|temp2;
```

```
        MsDev_WriteData(MASTER_ONLY, temp);
```

```
    }
```

```
}
```

```
MsDev_WriteCom(SLAVE_ONLY, 0x00);
MsDev_WriteData(SLAVE_ONLY, 0x17);
MsDev_WriteData(SLAVE_ONLY, 0xE9);
```

```
MsDev_WriteCom(SLAVE_ONLY, 0x10);
Delay_ms(10);
for(col=0; col<400; col++)
```

```

{
    for(row=0; row<50; row++)
    {
        temp1 = ((DTM1[col*100+row*2]&0x0F)<<4);
        temp2 = (DTM1[col*100+row*2+1]&0x0F);

        temp = temp1|temp2;

        MsDev_WriteData(SLAVE_ONLY, temp);
    }

    for(row=0; row<50; row++)
    {
        temp1 = ((DTM1[col*100+row*2]&0x0F)<<4);
        temp2 = (DTM1[col*100+row*2+1]&0x0F);

        temp = temp1|temp2;

        MsDev_WriteData(SLAVE_ONLY, temp);
    }
}

static void Send_Color_Data_A(unsigned char color)
{
    unsigned int col, row;
    unsigned int pcnt;

    MsDev_WriteCom(MASTER_ONLY, 0x00);
    MsDev_WriteData(MASTER_ONLY, 0x13);
    MsDev_WriteData(MASTER_ONLY, 0xE9);

    MsDev_WriteCom(MASTER_ONLY, 0x10);
    Delay_ms(10);
    pcnt = 0;
    for(col=0; col<400; col++)
    {
        for(row=0; row<100; row++)
        {
            temp1 = (DTM1[col*200+row*2]&0xF0);
            temp2 = (DTM1[col*200+row*2+1]>>4);

            temp = temp1|temp2;

            MsDev_WriteData(MASTER_ONLY, temp);
        }
    }
}

```

```

    }

}

MsDev_WriteCom(SLAVE_ONLY, 0x00);
MsDev_WriteData(SLAVE_ONLY, 0x17);
MsDev_WriteData(SLAVE_ONLY, 0xE9);

MsDev_WriteCom(SLAVE_ONLY, 0x10);
Delay_ms(10);
for(col=0; col<400; col++)
{
    for(row=0; row<100; row++)
    {
        temp1 = ((DTM1[col*200+row*2]&0x0F)<<4);
        temp2 = (DTM1[col*200+row*2+1]&0x0F);

        temp = temp1|temp2;

        MsDev_WriteData(SLAVE_ONLY, temp);
    }
}

}

void EPD_Display(unsigned char display_bkg)
{
    unsigned int i, j, num, cnt, data;

    if(PIC_A == display_bkg)
    {
        Send_Color_Data_A(display_bkg);
    }
    else if(STRIPE == display_bkg)
    {
        Send_HV_Stripe_Data();
    }
    else if(PIC_HALF == display_bkg)
    {
        Send_Color_Data_half_A(display_bkg);
    }
    else
    {
        Send_Color_Data(display_bkg);
    }
}

```

```

    MsDev_WriteCom(MASTER_SLAVE, 0x04);
    ReadBusy();
    Delay_ms(10);

    MsDev_WriteCom(MASTER_SLAVE, 0x12);
    MsDev_WriteData(MASTER_SLAVE, 0x00);
    Delay_ms(10);
    ReadBusy();

    MsDev_WriteCom(MASTER_SLAVE, 0x02);
    MsDev_WriteData(MASTER_SLAVE, 0x00);
    ReadBusy();
    Delay_ms(20);
}

void main (void)
{
    // Sys_init();

    //Spiciling together from half an image
    Reset();
    ReadBusy();
    Read_OTP_PWR(TEMPTR_ON);
    EPD_Display(PIC_HALF);
    EnterDeepsleep();
    Delay_S(5);

    //A whole image
    Reset();
    ReadBusy();
    Read_OTP_PWR(TEMPTR_ON);
    EPD_Display(PIC_A);
    EnterDeepsleep();
    Delay_S(5);

    while(1)
    {
    }
}

```