```
/*****************************Header
Files*************************************/
#include "msp430G2955.h"

#include "image.h"


/*****************************Note*********************************/
//2.In the code, "2BIT" and "1BIT" refer to the method used to take the mold
when input image data.
//3.bitInterleave the function after comparing image data from the current
and previous frames, using image data mold 1 bit.
/*********************************Macro
definition***********************************/




/*****************************Define I/O
pins*********************************/
#define SDA_H     (P1OUT |=BIT7)     //Define SDA_H as setting P1.7 to a high
level.
#define SDA_L     (P1OUT &=~BIT7)      //Define SDA_L as setting P1.7 to a
low level.
#define SCLK_H    (P1OUT |=BIT6)        //Define SCLK_H as setting P1.6 to
a high level.
#define SCLK_L    (P1OUT &=~BIT6)      //Define SCLK_L as setting P1.6 to
a low level.
#define nCS_H     (P1OUT |=BIT5)     //Define nCS_H as setting P1.5 to a high
level.
#define nCS_L     (P1OUT &=~BIT5)      //Define nCS_L as setting P1.5 to a
low level.
#define nDC_H     (P1OUT |=BIT4)       //Define nDC_H as setting P1.4 to a
high level.
#define nDC_L     (P1OUT &=~BIT4)      //Define nDC_L as setting P1.4 to a
low level.
#define nRST_H    (P1OUT |=BIT3)       //Define nRST_H as setting P1.3 to
a high level.
#define nRST_L    (P1OUT &=~BIT3)      //Define nRST_L as setting P1.3 to
a low level.
#define R_SDA     0x80
/*********************************Macro
definition***********************************/
#define PIC_BLACK   1             //Black
#define PIC_WHITE       2             //White
#define PIC_RED        3             //Red
#define PIC_YELLOW     4             //Horizontal Stripe
#define PIC_BWRY     5             //Vertical Stripe
#define PIC_Image_1      6             //Image
#define PIC_A       7            //Image
#define PIC_R        8             //Red
#define PIC_HLINE       9
#define PIC_VLINE      10

#define PIC_Yellow         11

#define PIC_HLINE_BW           11
```

```
/**********************************Variable
definition************************************/
unsigned char Temperature;    //Define a temperature variable
unsigned char temp1;    //Define a temperature variable.

unsigned char tempvalue,tempvalueA;//Define a temperature variable.

unsigned char temp1_h,temp1_l;



unsigned char VCOM;    //Define VCOM
unsigned char A4_OTP_Byte0;
unsigned char A4_OTP_Byte1;
unsigned char A4_OTP_Byte2;
unsigned char A4_OTP_Byte3;
unsigned char Code_CRC;

unsigned char hi, lo,b;


/*******************************Delay*********************************/
void Delay_100nS(int delaytime)         //Delay 1 pcs delaytime
{
    int i,j;

    for(i=0;i<delaytime;i++)
        for(j=0;j<1;j++);
}
void Delay_mS(int delaytime)       //Delay 200 pcs delaytime
{
    int i,j;

    for(i=0;i<delaytime;i++)
        for(j=0;j<2400;j++);
}
void Delay_S(int delaytime)        //Delay 1000*200 pcs delaytime
{
    int i,j,k;

    for(i=0;i<delaytime;i++)
        for(j=0;j<1000;j++)
            for(k=0;k<2400;k++);
}




/*************************E-Paper
Reset*******************************/
void Reset()
{
   nRST_H;
```

```
    Delay_mS(10);
    nRST_L;
    Delay_mS(20);
    nRST_H;
    Delay_mS(10);
}



/*************************E-Paper Read the status
bit*************************/


void READBUSY()
{
    while(1)
        {
            _NOP();
            if((P1IN & 0x04)==0x04)
            break;
        }
}
/*************************E-Paper Enter
Command*******************************/
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;
        }
}


/*************************E-Paper Enter
Data*******************************/
void SPI4W_WriteDATA(unsigned char INIT_DATA)
{
    unsigned char TEMPCOM;
    unsigned char scnt;
    P1DIR |= R_SDA;
    TEMPCOM=INIT_DATA;
    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;
        }
}

/************************E-Paper Read Data************************/
unsigned char SPI4W_READDATA()              /*Function of E-Paper Read
Data*/
 {
    P1DIR &=~ R_SDA;
    unsigned char scnt,temp;
    temp=0;
    nCS_H;
    nDC_H;
    nCS_L;
    for(scnt=0;scnt<8;scnt++)
        {
            SCLK_L;
            if(P1IN&R_SDA)
                temp=(temp<<1)|0x01;
            else
                temp=temp<<1;
                SCLK_H;
                SCLK_L;
        }
    return temp;
}
/***************************E-Paper Dormant
Functions*****************************/

void enterdeepsleep()
{


    SPI4W_WriteCOM(0x07);  // Deep sleep
    SPI4W_WriteDATA(0xA5);




}




void JD79665_Init_For_OTP()
{

   SPI4W_WriteCOM(0x00);     //PSR Register
```

```
    SPI4W_WriteDATA(0x2F);
    SPI4W_WriteDATA(0x0E);      //High 8bit be used in conjunction with the
E9 command
    READBUSY();

    SPI4W_WriteCOM(0x50);     //CDI Booder Settings   POR:97
    SPI4W_WriteDATA(0x77);

    SPI4W_WriteCOM(0xE9);     //PST
    SPI4W_WriteDATA(0x01);

}
```

tft

```
/************************************************************************
***************************************
* Read Revision function
************************************************************************
***************************************/
void bitInterleave(unsigned char bytes1, unsigned char bytes2) {

   unsigned short result=0;


   for (int i = 0; i < 8; i++) {


       result |= ((bytes1 >> (7 - i)) & 1) << (2 * (7-i)+1);
       result |= ((bytes2 >> (7 - i)) & 1) << (2 * (7-i));


       if(i == 3)
         SPI4W_WriteDATA(result >> 8);


       if(i == 7)
         SPI4W_WriteDATA(result);


   }


}


void Fast_WHITE_TO_A()
{
  unsigned int i,j,pcnt,pcnt1;
```

```
    SPI4W_WriteCOM(0x10);   // DTM1 Write
          READBUSY();

  pcnt = 0;

  for(i=0; i <384; i++)
  {

         for(j=0; j<21; j++)
         {


             bitInterleave(0xFF,gImage_250_112_R[pcnt]);   //The first
parameter represents the current displayed image data, and the second
parameter represents the image data to be refreshed.

            pcnt++;
           }


  }



    SPI4W_WriteCOM(0x04); // Power ON
    READBUSY();
    Delay_mS(10);



    SPI4W_WriteCOM(0x12);  // Display Refresh
    SPI4W_WriteDATA(0x00);
    Delay_mS(10);
    READBUSY();

    SPI4W_WriteCOM(0x02);  // Power OFF
    SPI4W_WriteDATA(0x00);
    READBUSY();
    Delay_mS(20);



}
```

```
void EPD_Slow_Display_1BIT()
{
  unsigned int row, col;
  unsigned int pcnt;
  unsigned char temp1,temp2,tempvalue;



 SPI4W_WRITECOM(0x40);
   READBUSY();
   temp1=SPI4W_READDATA();


   if(temp1<=0)
     tempvalue=232;  // -24


   else if(temp1<=10)
     tempvalue=235;   // -21


   else if(temp1<=20)
     tempvalue=238;   // -18


   else if(temp1<=30)
     tempvalue=241;   // -15


   else if(temp1<=127)
     tempvalue=244;    // -12

   else
     tempvalue=232;




   SPI4W_WRITECOM(0xE0);
   SPI4W_WRITEDATA(0x02);
   SPI4W_WRITECOM(0xE6);
   SPI4W_WRITEDATA(tempvalue);

   SPI4W_WRITECOM(0xA5);
   READBUSY();
   DELAY_mS(10);



   SPI4W_WriteCOM(0x10);
   pcnt=0;

   for(col=0; col<384; col++)
   {
     for(row=0; row<21; row++)
     {
```

```
    unsigned char combin_byte0=0, combin_byte1=0;
    unsigned char data = gImage_250_112_R[pcnt++];

         for(unsigned char i=0; i<8; i++)
         {
             if(i<4)
             {
                 combin_byte0 |= (((data>>(7-i))&0x01))<<(8-2*(i+1));
             }
             else
             {
                 combin_byte1 |= (((data>>(7-i))&0x01))<<(14-2*i);
             }

         }

        SPI4W_WriteDATA(combin_byte0);
        SPI4W_WriteDATA(combin_byte1);


      }




    }







    SPI4W_WriteCOM(0x04); // Power ON
    READBUSY();
    Delay_mS(10);



    SPI4W_WriteCOM(0x12);  // Display Refresh
    SPI4W_WriteDATA(0x00);
    Delay_mS(10);
    READBUSY();


    SPI4W_WriteCOM(0x02);  // Power OFF
    SPI4W_WriteDATA(0x00);
    READBUSY();
    Delay_mS(20);


}
```

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



    SPI4W_WriteCOM(0x10);   // DTM1 Write
    READBUSY();
    pcnt = 0;
    for(col=0; col<384; col++)
    {
           for(row=0; row<42; row++)
           {

             SPI4W_WriteDATA(0x00);

           }

    }



   SPI4W_WriteCOM(0x83);
   SPI4W_WriteDATA(0x00);//HRST[9:8] Horizontal start address Source:250
   SPI4W_WriteDATA(0x20);//HRST[7:2] 32

   SPI4W_WriteDATA(0x00);//HRED[9:8] Horizontal end address. Source:250
   SPI4W_WriteDATA(0x3F);//HRED[7:2] 63

   SPI4W_WriteDATA(0x00);//VRST[9:8] Vertical start address. Gate:128
   SPI4W_WriteDATA(0x0E);//VRST[7:2] 15

   SPI4W_WriteDATA(0x00);//VRST[9:8] Vertical end address. Gate:128
   SPI4W_WriteDATA(0x1D);//VRST[7:2] 29


   SPI4W_WriteDATA(0x01);



    SPI4W_WriteCOM(0x10);   // DTM1 Write
        READBUSY();
        pcnt = 0;
        for(col=0; col<16; col++)
        {
            for(row=0; row<4; row++)
            {

                      bitInterleave(0xFF,gImage_55[pcnt]);   //The first
parameter represents the current displayed image data, and the second
parameter represents the image data to be refreshed.



                        pcnt++;
```

```
            }

        }

  SPI4W_WriteCOM(0x83);
  SPI4W_WriteDATA(0x00);//HRST[9:8] Horizontal start address Source:250
  SPI4W_WriteDATA(0x20);//HRST[7:2] 32

  SPI4W_WriteDATA(0x00);//HRED[9:8] Horizontal end address. Source:250
  SPI4W_WriteDATA(0x3F);//HRED[7:2] 63

  SPI4W_WriteDATA(0x00);//VRST[9:8] Vertical start address. Gate:128
  SPI4W_WriteDATA(0x0E);//VRST[7:2] 15

  SPI4W_WriteDATA(0x00);//VRST[9:8] Vertical end address. Gate:128
  SPI4W_WriteDATA(0x1D);//VRST[7:2] 29

  SPI4W_WriteDATA(0x00);//VRST[9:8] Vertic



   SPI4W_WriteCOM(0x04); // Power ON
   READBUSY();
   Delay_mS(10);



   SPI4W_WriteCOM(0x12);  // Display Refresh
   SPI4W_WriteDATA(0x00);
   READBUSY();

   SPI4W_WriteCOM(0x02);  // Power OFF
   SPI4W_WriteDATA(0x00);
   READBUSY();


   Delay_mS(20);

}


void EPD_Slow_Display_2BIT(unsigned char num)
{
  unsigned int row, col;
  unsigned int pcnt;
 unsigned char temp1,temp2,tempvalue;



 SPI4W_WRITECOM(0x40);
   READBUSY();
   temp1=SPI4W_READDATA();


   if(temp1<=0)
     tempvalue=232;  // -24
```

```
else if(temp1<=10)
  tempvalue=235;   // -21


else if(temp1<=20)
  tempvalue=238;   // -18


else if(temp1<=30)
  tempvalue=241;   // -15


else if(temp1<=127)
  tempvalue=244;    // -12

else
  tempvalue=232;




SPI4W_WRITECOM(0xE0);
SPI4W_WRITEDATA(0x02);
SPI4W_WRITECOM(0xE6);
SPI4W_WRITEDATA(tempvalue);

SPI4W_WRITECOM(0xA5);
READBUSY();
DELAY_mS(10);



SPI4W_WriteCOM(0x10);
pcnt=0;

for(col=0; col<384; col++)
{
  for(row=0; row<42; row++)
  {


    switch(num){

       case PIC_A:

        SPI4W_WriteDATA(gImage_Font[pcnt++]);

      break;

      case PIC_VLINE:
           if(col < 261)
               SPI4W_WriteDATA(0xFF);
          else
               SPI4W_WriteDATA(0x00);
```

```
            break;

      case PIC_HLINE:

        if(row < 18)
           SPI4W_WriteDATA(0x00);
        else
           SPI4W_WriteDATA(0xFF);

       break;



    case PIC_WHITE:
      SPI4W_WriteDATA(0xFF);
      break;
    case PIC_BLACK:
      SPI4W_WriteDATA(0x00);
      break;



    }



  }



}
```

```
SPI4W_WriteCOM(0x04); // Power ON
READBUSY();
Delay_mS(10);



SPI4W_WriteCOM(0x12);  // Display Refresh
SPI4W_WriteDATA(0x00);
Delay_mS(10);
READBUSY();


SPI4W_WriteCOM(0x02);  // Power OFF
SPI4W_WriteDATA(0x00);
READBUSY();
Delay_mS(20);
```

```
}



void tempetrue_value()
{


      SPI4W_WriteCOM(0xE0);
      SPI4W_WriteDATA(0x00);

      SPI4W_WriteCOM(0xA5);
      READBUSY();
      Delay_mS(10);



}



/**********************************Main 函数
***********************************/
int main()
{

    unsigned int i;

    WDTCTL = WDTPW + WDTHOLD;                 //停止看门狗定时器，防止超时复位
    BCSCTL1 = CALBC1_8MHZ;
    DCOCTL = CALDCO_8MHZ;


  P1DIR |=0x78;                                                          //
set P1.3~7 output
  P4DIR |=0x08;                                                          //
set  P4.3 output


//Slow  refresh
    Reset();
    READBUSY();
    JD79665_Init_For_OTP();
    tempetrue_value();
    EPD_Slow_Display_2BIT(PIC_WHITE);
    enterdeepsleep();
    Delay_S(5);


// fast  refresh
    Reset();
    READBUSY();
    JD79665_Init_For_OTP();
```

```
    Fast_WHITE_TO_A();
    enterdeepsleep();
    Delay_S(5);


//Slow  refresh
    Reset();
    READBUSY();
    JD79665_Init_For_OTP();
    tempetrue_value();
    EPD_Slow_Display_2BIT(PIC_WHITE);
    enterdeepsleep();
    Delay_S(5);


//  partial refresh
    Reset();
    READBUSY();
    JD79665_Init_For_OTP();
    dis_partial_img();
    enterdeepsleep();
    Delay_S(5);



//  Slow  refresh   Use 1-bit image data
    Reset();
    READBUSY();
    JD79665_Init_For_OTP();
    tempetrue_value();
    EPD_Slow_Display_1BIT();
    enterdeepsleep();
    Delay_S(5);


    while(1){};


}
```