```
/*****************************************************************************
*********
*
* MSP430 microcontroller,MSP430F2274 devices.
*
* Lut from otp,test program
*
* Texas Instruments, Version 1.0
*
* Rev. 1.0, Setup,2022.12.07
*
*Author & origin:
*
*****************************************************************************
********/


/*************************************************************
*Header File
*************************************************************/

#include "msp430x22x4.h"

/*************************************************************
* IO Port Define
*************************************************************/
#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)



/*************************************************************
*Image-Display define
*
*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


/*************************************************************
```

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


/************************************************************
*MCU delay configuration
*************************************************************/
void Delay_ms(int delaytime)  // delaytime=1=1ms
{
  int i,j;
  for(i=0;i<delaytime;i++)
     for(j=0;j<200;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<200;k++);
}


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



/************************************************************
*IC reset
*************************************************************/
void RESET()
{
  nRST_H;
  Delay_ms(10);
  nRST_L;
  Delay_ms(10);
  nRST_H;
  Delay_ms(10);
}

/************************************************************
*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_temperture(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;
 }


/************************************************************
* 1685 EPD INIT
************************************************************/

void INIT_SSD1685()
{

  SPI4W_WRITECOM(0x01);
//Driver output control
  SPI4W_WRITEDATA(0x2B);
  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(0x18);                                        // End
0x14=20 (20+1)x8=168

  SPI4W_WRITECOM(0x45);
//set Ram y -address
  SPI4W_WRITEDATA(0x2B);
  SPI4W_WRITEDATA(0x01);
  SPI4W_WRITEDATA(0x00);                                         // RAM y
address end at 00h;
  SPI4W_WRITEDATA(0x00);

  SPI4W_WRITECOM(0x21);
//set Ram y -address
  SPI4W_WRITEDATA(0x40);
  SPI4W_WRITEDATA(0x00);

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

}


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


/***********************************************************
*
*showImage
*
***********************************************************/
void Dis_img_partial(unsigned char num)
{
  unsigned int row, col;
  unsigned int pcnt;

  SPI4W_WRITECOM(0x44);      // Set Ram X -address
  SPI4W_WRITEDATA(0x0C);     // Start 0
  SPI4W_WRITEDATA(0x0D);     // End   0x14=20 (20+1)x8=168

  SPI4W_WRITECOM(0x45);      //set Ram y -address
  SPI4W_WRITEDATA(0x6D);
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITEDATA(0x64);     // RAM y address end at 00h;
```

```
  SPI4W_WRITEDATA(0x00);

  SPI4W_WRITECOM(0x4E);
  SPI4W_WRITEDATA(0x0C);
  SPI4W_WRITECOM(0x4F);
  SPI4W_WRITEDATA(0x6D);
  SPI4W_WRITEDATA(0x00);

  SPI4W_WRITECOM(0x24);
  pcnt = 0;

  for(col=0; col<10; col++)
  {
    for(row=0; row<2; row++)
    {
      switch(num)
        {
        case 0:
        SPI4W_WRITEDATA(gImage_T_0[pcnt]);
        break;

        case 1:
        SPI4W_WRITEDATA(gImage_T_1[pcnt]);
        break;

        case 2:
        SPI4W_WRITEDATA(gImage_T_2[pcnt]);
        break;

        case 3:
        SPI4W_WRITEDATA(gImage_T_3[pcnt]);
        break;

        case 4:
        SPI4W_WRITEDATA(gImage_T_4[pcnt]);
        break;

        case 5:
        SPI4W_WRITEDATA(gImage_T_5[pcnt]);
        break;

        default:
        break;
        }
      pcnt++;
    }
  }

  SPI4W_WRITECOM(0x3C);      // Set border
  SPI4W_WRITEDATA(0xC1);

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

  SPI4W_WRITECOM(0x21);
//set Ram y -address
  SPI4W_WRITEDATA(0x00);
```

```
  SPI4W_WRITEDATA(0x00);

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

}

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

  SPI4W_WRITECOM(0x21);
  SPI4W_WRITEDATA(0x40);
  SPI4W_WRITEDATA(0x00);          // 80:168X384   00:200X384

  SPI4W_WRITECOM(0x4E);
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITECOM(0x4F);
  SPI4W_WRITEDATA(0x2B);
  SPI4W_WRITEDATA(0x01);

  SPI4W_WRITECOM(0x24);
  pcnt = 0;

  for(col=0; col<300; col++)
  {
    for(row=0; row<25; row++)
    {
      switch(num)
      {
        case PIC_A:
          SPI4W_WRITEDATA(gImage_2wx_886[pcnt]);
        break;

        case PIC_White:
          SPI4W_WRITEDATA(0xff);
        break;

        case PIC_Black:
          SPI4W_WRITEDATA(0x00);
        break;

        case PIC_HLINE:
          if(col < 150)
            SPI4W_WRITEDATA(0xff);
          else
            SPI4W_WRITEDATA(0x00);
        break;

        case PIC_VLINE:
          if(row < 13)
            SPI4W_WRITEDATA(0xff);
          else
```

```
            SPI4W_WRITEDATA(0x00);
        break;

        default:
        break;
      }
    pcnt++;
    }
  }

  SPI4W_WRITECOM(0x26);
  pcnt = 0;

  for(col=0; col<300; col++)
  {
    for(row=0; row<25; row++)
    {
      switch(num)
      {
        case PIC_A:
          SPI4W_WRITEDATA(gImage_2wx_886[pcnt]);
        break;

        case PIC_White:
          SPI4W_WRITEDATA(0xff);
        break;

        case PIC_Black:
          SPI4W_WRITEDATA(0x00);
        break;

        case PIC_HLINE:
          if(col < 150)
            SPI4W_WRITEDATA(0xff);
          else
            SPI4W_WRITEDATA(0x00);
        break;

        case PIC_VLINE:
          if(row < 13)
            SPI4W_WRITEDATA(0xff);
          else
            SPI4W_WRITEDATA(0x00);
        break;

        default:
        break;
      }
    pcnt++;
    }
  }

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

  SPI4W_WRITECOM(0x21);
//set Ram y -address
```

```
  SPI4W_WRITEDATA(0x40);
  SPI4W_WRITEDATA(0x00);

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

}

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

  SPI4W_WRITECOM(0x44);
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITEDATA(0x18);
  SPI4W_WRITECOM(0x45);
  SPI4W_WRITEDATA(0x2B);
  SPI4W_WRITEDATA(0x01);
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITEDATA(0x00);

  SPI4W_WRITECOM(0x4E);
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITECOM(0x4F);
  SPI4W_WRITEDATA(0x2B);
  SPI4W_WRITEDATA(0x01);

  SPI4W_WRITECOM(0x24);
  pcnt = 0;

  for(col=0; col<300; col++)
  {
    for(row=0; row<25; row++)
    {
      switch(num)
      {
      case PIC_A:
        SPI4W_WRITEDATA(gImage_2wx_886[pcnt]);
      break;

      case PIC_White:
        SPI4W_WRITEDATA(0xff);
      break;

      case PIC_Black:
        SPI4W_WRITEDATA(0x00);
      break;

      case PIC_HLINE:
        if(col < 150)
          SPI4W_WRITEDATA(0xff);
        else
          SPI4W_WRITEDATA(0x00);
```

```
      break;

      case PIC_VLINE:
        if(row < 13)
          SPI4W_WRITEDATA(0xff);
        else
          SPI4W_WRITEDATA(0x00);
      break;

      default:
      break;
      }

    pcnt++;
    }
  }

  SPI4W_WRITECOM(0x3C);      // Set border
  SPI4W_WRITEDATA(0xC1);

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

  SPI4W_WRITECOM(0x21);
//set Ram y -address
  SPI4W_WRITEDATA(0x00);
  SPI4W_WRITEDATA(0x00);

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

}




/************************************************************
* main to driver ic
************************************************************/
void main( void )
{

    unsigned int i;
    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
    P4DIR |= 0X01;// set P4.1   output


//slow refresh
    RESET();
    SPI4W_WRITECOM(0x12);
```

```
    Read_BUSY();
    INIT_SSD1685();
    Dis_img(PIC_White);
    Enter_deep_sleep();
    Delay_s(1);
    _NOP();

    RESET();
    Read_BUSY();
    Dis_img_fast(PIC_A);
    Enter_deep_sleep();
    Delay_s(1);
    _NOP();

//slow refresh
    RESET();
    SPI4W_WRITECOM(0x12);
    Read_BUSY();
    INIT_SSD1685();
    Dis_img(PIC_A);
    Enter_deep_sleep();
    Delay_s(1);
    _NOP();

//partial refresh
    RESET();
    Read_BUSY();
    Dis_img_partial(1);
    Enter_deep_sleep();

    RESET();
    Read_BUSY();
    Dis_img_partial(2);
    Enter_deep_sleep();

    RESET();
    Read_BUSY();
    Dis_img_partial(3);
    Enter_deep_sleep();

    RESET();
    Read_BUSY();
    Dis_img_partial(4);
    Enter_deep_sleep();

    RESET();
    Read_BUSY();
    Dis_img_partial(5);
    Enter_deep_sleep();

    while(1){}


}
```