

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

```
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//  Program:      EPD
//  Author:       XING TAI
//  Date:         2025
//  Rev:          1.0
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

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

```
#include "hw_intrfc.h"
#include "image.h"
```

```
/******EPD Resolution*****/ //EPD
```

Resolution Settings

```
#define GATES          384
#define SOURCES        184
```

```
#define EINK_SDA_PIN    GPIO_PIN_6
#define EINK_SDA_PORT   GPIOA
#define EINK_SCL_PIN    GPIO_PIN_8
#define EINK_SCL_PORT   GPIOB
#define EINK_NCS_PIN    GPIO_PIN_9
#define EINK_NCS_PORT   GPIOB
#define EINK_NDC_PIN    GPIO_PIN_1
#define EINK_NDC_PORT   GPIOB
#define EINK_RST_PIN    GPIO_PIN_2
#define EINK_RST_PORT   GPIOB
#define EINK_BUSY_PIN   GPIO_PIN_3
#define EINK_BUSY_PORT  GPIOB
```

```
#define SDA_H           GPIO_SetBits    (EINK_SDA_PORT,  EINK_SDA_PIN)
#define SDA_L           GPIO_ResetBits  (EINK_SDA_PORT,  EINK_SDA_PIN)
#define SCLK_H          GPIO_SetBits    (EINK_SCL_PORT,  EINK_SCL_PIN)
#define SCLK_L          GPIO_ResetBits  (EINK_SCL_PORT,  EINK_SCL_PIN)
#define nCS_H           GPIO_SetBits    (EINK_NCS_PORT,  EINK_NCS_PIN)
#define nCS_L           GPIO_ResetBits  (EINK_NCS_PORT,  EINK_NCS_PIN)
```

```

#define nDC_H          GPIO_SetBits    (EINK_NDC_PORT,   EINK_NDC_PIN)
#define nDC_L          GPIO_ResetBits  (EINK_NDC_PORT,   EINK_NDC_PIN)
#define nRST_H         GPIO_SetBits    (EINK_RST_PORT,    EINK_RST_PIN)
#define nRST_L         GPIO_ResetBits  (EINK_RST_PORT,    EINK_RST_PIN)

#define SDA_LEVEL      GPIO_ReadInputDataBit(EINK_SDA_PORT, EINK_SDA_PIN)
#define BUSY_LEVEL     GPIO_ReadInputDataBit(EINK_BUSY_PORT, EINK_BUSY_PIN)

#define EINK_VOLO_PIN   GPIO_PIN_2
#define EINK_VOLO_PORT  GPIOA

#define EINK_VOL1_PIN   GPIO_PIN_3
#define EINK_VOL2_PORT  GPIOA

#define VDD_OFF         GPIO_ResetBits  (EINK_VDD_PORT,    EINK_VDD_PIN)
#define VDD_ON          GPIO_SetBits     (EINK_VDD_PORT,    EINK_VDD_PIN)

#define EINK_VDD_PIN    GPIO_PIN_12
#define EINK_VDD_PORT   GPIOB

#define VDD_OFF         GPIO_ResetBits  (EINK_VDD_PORT,    EINK_VDD_PIN)
#define VDD_ON          GPIO_SetBits     (EINK_VDD_PORT,    EINK_VDD_PIN)

#define PIC_YELLOW      253
#define PIC_RED          254
#define PIC_WHITE       255
#define PIC_BLACK       0
#define PIC_NULL         1
#define PIC_A            2
#define PIC_B            3
#define PIC_HLINE        4
#define PIC_HLINE1      10
#define PIC_VLINE        5
#define PIC_C            6
#define PIC_D            7
#define PIC_E            8
#define PIC_DOT          9

enum Vol_Value_e
{
    _2_3V = 0,
    _2_4V,
    _2_5V,
    _3_0V,
    _3_6V
};

```

```

#define SOURCE_SCAN_NUM                (SOURCES/2)

#define GATE_DIV_SET(x, num)           (GATES*x/num)
#define SOURCE_DIV_SET(y, num)         (SOURCE_SCAN_NUM*y/num)

static unsigned char Temptr = 0;

/*****
*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<5375; 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<800; j++)
            for(k=0; k<5375; k++);
}

/*****
*Function-ICCheck busy Signal
*Parameters-void
*Back-void
*****/
void ReadBusy(void)
{
    while(1)
    {
        __NOP();
        if(BUSY_LEVEL != 0){
            break;
        }
    }
}

```

```

    }
}

/*****
*Function-IC Reset Signal
*Parameters-void
*Back-void
*****/
void Reset(void)
{
    nRST_L;
    Delay_ms(20);
    nRST_H;
    Delay_ms(10);
    nRST_L;
    Delay_ms(20);
    nRST_H;
    Delay_ms(10);
    ReadBusy();
    Delay_ms(10);
}

/*****
*Funcio-IC_SPI Enter Command
*Parameters-init_com: Command Code
*Back-void
*****/
void SPI4W_WriteCom(unsigned char init_com)
{
    unsigned char temp_com;
    unsigned char scnt;

    temp_com = init_com;
    nCS_H;
    nCS_L; //SPI
    chip select the signal, It will be active at a low level.
    SCLK_L;
    nDC_L;
    for(scnt=0; scnt<8; scnt++)
    {
        if(temp_com & 0x80)
            SDA_H;
        else
            SDA_L;
        SCLK_H;
        SCLK_L;
        temp_com = temp_com << 1;
    }
    nCS_H;

```

```

}

/*****
*Function-IC_SPI Enter Data
*Parameters-init_data: Data Content
*Back-void
*****/
void SPI4W_WriteData(unsigned char init_data)
{
    unsigned char temp_data;
    unsigned char scnt;

    temp_data = init_data;
    nCS_H;
    nCS_L; //SPI
    chip select the signal, It will be active at a low level.
    SCLK_L;
    nDC_H;
    for(scnt=0;scnt<8;scnt++)
    {
        if(temp_data & 0x80)
            SDA_H;
        else
            SDA_L;
        SCLK_H;
        SCLK_L;
        temp_data = temp_data<<1;
    }
    nCS_H;
}

/*****
*Function-IC_SPI Read Data
*Parameters-void
*Back-unsigned char one byte
*****/
unsigned char SPI4W_ReadData(void)
{
    unsigned char scnt,temp;

    gpio_cfg_input(EINK_SDA_PORT, EINK_SDA_PIN, PULLUP);

    nCS_H;
    nCS_L; //SPI
    chip select the signal, It will be active at a low level.
    //SCLK_H;
    SCLK_L;
    nDC_H;

```

```

    for(scnt=0; scnt<8; scnt++)
    {
        //SCLK_H;
        if(SDA_LEVEL)
            temp = (temp<<1) | 0x01;
        else
            temp = temp << 1;
        //SCLK_H;
        SCLK_H;
        SCLK_L;
    }
    nCS_H;

    gpio_cfg_output(EINK_SDA_PORT, EINK_SDA_PIN, NOPULL);

    return temp;
}

```

```

/*****
*Function-Read the IC Temperature
*Parameters-void
*Back-unsigned char one byte
*****/
unsigned char ReadTemptr(void)
{
    unsigned char tempr_intgr = 0;
    unsigned char tempr_decml = 0;

    Reset();

    SPI4W_WriteCom(0x40);
    ReadBusy();
    tempr_intgr = SPI4W_ReadData();
    tempr_decml = SPI4W_ReadData();

    Temptr = tempr_intgr;

    return tempr_intgr;
}

```

```

/*****
*Function-Read IC's ChipID
*Parameters-void
*Back-unsigned char one byte
*****/
unsigned char Read_Revision(void)
{

```

```

    unsigned char rever[4]= {0};

    Reset();

    SPI4W_WriteCom(0x70);
    rever[0] = SPI4W_ReadData(); //chip
id
    rever[1] = SPI4W_ReadData();
//lut_ver[0:7]
    rever[2] = SPI4W_ReadData();
//lut_ver[8:15]
    rever[3] = SPI4W_ReadData();
//lut_ver[23:16]

    return rever[0];
}

/*****
*Function-Send Display Command
*Parameters-None
*Back-None
*****/
void Display_Cmd(void)
{
    SPI4W_WriteCom(0x04); //
    Power ON
    ReadBusy();

    SPI4W_WriteCom(0x12); //
    Display Refresh
    SPI4W_WriteData(0x00);
    ReadBusy();

    SPI4W_WriteCom(0x02); //
    Power OFF
    SPI4W_WriteData(0x00);
    ReadBusy();
}

/*****
*Function-Enter Deep Sleep Mode
*Parameters-None
*Back-None
*****/
void EnterDeepsleep(void)
{
    SPI4W_WriteCom(0x07); //
    Deep sleep
    SPI4W_WriteData(0xA5);

```

```

}

/*****
*Functions—Basic Displays (the most common display content, such as font charts,
horizontal bars, vertical bars, full-color displays and others)
*Parameters- dis_type: Display Item
*Back- void
*****/
void Basic_Display(unsigned char dis_type)
{
    unsigned int col = 0, row = 0;
    unsigned int pcnt = 0;

    SPI4W_WriteCom(0x10); //
    DTMI_Write

    for(col=0; col<GATES; col++)
    {
        for(row=0; row<SOURCE_SCAN_NUM; row++)
        {
            switch(dis_type)
            {
            case PIC_A:
                SPI4W_WriteData(gImage_Font[pcnt]);
                break;
            case PIC_VLINE:
                if(col < GATE_DIV_SET(1, 4))
                    SPI4W_WriteData(0x00);
                else if(col < GATE_DIV_SET(2, 4))
                    SPI4W_WriteData(0x11);
                else if(col < GATE_DIV_SET(3, 4))
                    SPI4W_WriteData(0x33);
                else
                    SPI4W_WriteData(0x22);
                break;
            case PIC_YELLOW:
                SPI4W_WriteData(0x22);
                break;
            case PIC_RED:
                SPI4W_WriteData(0x33);
                break;
            case PIC_BLACK:
                SPI4W_WriteData(0x00);
                break;
            default:
            case PIC_WHITE:
                SPI4W_WriteData(0x11);
                break;
            }
        }
    }
}

```

```

        pent++;
    }
}

Display_Cmd();
return;
}

/*****
*Functions - System Initialization
*Parameters-void
*Back-void
*****/
void Sys_init(void)
{
    RCC_EnableAPB2PeriphClk(RCC_APB2_PERIPH_GPIOA, ENABLE);
    RCC_EnableAPB2PeriphClk(RCC_APB2_PERIPH_GPIOB, ENABLE);

    gpio_cfg_output(EINK_SDA_PORT, EINK_SDA_PIN, NOPULL);
    //SDA
    gpio_cfg_output(EINK_SCL_PORT, EINK_SCL_PIN, NOPULL);
    //SCL
    gpio_cfg_output(EINK_NCS_PORT, EINK_NCS_PIN, NOPULL);           //CS
    gpio_cfg_output(EINK_NDC_PORT, EINK_NDC_PIN, NOPULL);           //DC
    gpio_cfg_output(EINK_RST_PORT, EINK_RST_PIN, NOPULL);
    //RST

    gpio_cfg_input(EINK_BUSY_PORT, EINK_BUSY_PIN, PULLUP);
    //BUSY
}

/*****
*Function-IC Initialization Settings
*Parameters-void: None
*Back- void
*****/
void Init_IC(void)
{
    //FITI cmd.
    SPI4W_WriteCom(0x4D);
    SPI4W_WriteData(0x78);

    SPI4W_WriteCom(0xE9);
    SPI4W_WriteData(0x01);
}

```

```
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//xx    Main Function
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

int main(void)
{
    Sys_init();

    Delay_ms(2000);           //System delay; waiting for VDD (3.0 V) to power on

    ReadTemptr();

    Reset();
    Init_IC();
    Basic_Display(PIC_VLINE);
    EnterDeepsleep();
    Delay_S(5);

    while (1)
    {

    };

}
```