```
/*************************************Copyright (c)*****************************************************
**
**                                 http://www.powermcu.com
**
**--------------File Info-----------------------------------------------------------------------------
** File name:                   GLCD.c
** Descriptions:          Have been tested HY28A-LCDB HY28B
**
**----------------------------------------------------------------------------------------------------
** Created by:                  AVRman
** Created date:          2013-12-4
** Version:                             2.1
** Descriptions:          The original version
**
**----------------------------------------------------------------------------------------------------
** Modified by:
** Modified date:
** Version:
** Descriptions:
****************************************************************************************************/

/* Includes ------------------------------------------------------------------*/
#include "GLCD.h"
#include "AsciiLib.h"

/*****************************************************************************
* Function Name  : Lcd_Configuration
* Description    : Configures LCD Control lines
* Input          : None
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
static void LCD_Configuration(void)
{
        PINSEL_CFG_Type PinCfg;
        SSP_CFG_Type SSP_ConfigStruct;
        /*
         * Initialize SPI pin connect
         * P0.15 - LCD_CSB - used as GPIO
         * P0.16 - LCD_SCK
         * P0.17 - LCD_MISO
         * P0.18 - LCD_MOSI
         */
        PinCfg.Funcnum = 2;
        PinCfg.OpenDrain = 0;
        PinCfg.Pinmode = 0;
        PinCfg.Portnum = 0;
        PinCfg.Pinnum = 15;
        PINSEL_ConfigPin(&PinCfg);
        PinCfg.Pinnum = 17;
        PINSEL_ConfigPin(&PinCfg);
        PinCfg.Pinnum = 18;
        PINSEL_ConfigPin(&PinCfg);
        PinCfg.Funcnum = 0;
        PinCfg.Portnum = 0;
        PinCfg.Pinnum = 16;
        PINSEL_ConfigPin(&PinCfg);

        /* P1.31 LCD_CSB is output */
        GPIO_SetDir(CSB_PORT_NUM, ( (uint32_t) 1 << CSB_PIN_NUM ), 1);
        GPIO_SetValue(CSB_PORT_NUM, ( (uint32_t) 1 << CSB_PIN_NUM ) );

        /* initialize SSP configuration structure to default */
        SSP_ConfigStructInit(&SSP_ConfigStruct);

        SSP_ConfigStruct.CPHA = SSP_CPHA_SECOND;
        SSP_ConfigStruct.CPOL = SSP_CPOL_LO;
        SSP_ConfigStruct.ClockRate = 25000000;

        /* Initialize SSP peripheral with parameter given in structure above */
        SSP_Init(LPC_SSP0, &SSP_ConfigStruct);
        /* Enable SSP peripheral */
        SSP_Cmd(LPC_SSP0, ENABLE);
}
/*****************************************************************************
* Function Name  : LPC17xx_SPI_SendRecvByte
* Description    : Send one byte then recv one byte of response
* Input          : - byte_s: byte_s
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
unsigned char LPC17xx_SPI_SendRecvByte (unsigned char byte_s)
{
        /* wait for current SSP activity complete */
        while (SSP_GetStatus(LPC_SSP0, SSP_STAT_BUSY) ==  SET);

        SSP_SendData(LPC_SSP0, (unsigned short) byte_s);

        while (SSP_GetStatus(LPC_SSP0, SSP_STAT_RXFIFO_NOTEMPTY) == RESET);
        return (SSP_ReceiveData(LPC_SSP0));
}
```

```c
/*****************************************************************************
* Function Name  : LCD_WriteIndex
* Description    : LCD write register address
* Input          : - index: register address
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
void LCD_WriteIndex(unsigned char index)
{
        SPI_CS_LOW;

        /* SPI write data */
        LPC17xx_SPI_SendRecvByte(SPI_START | SPI_WR | SPI_INDEX);   /* Write : RS = 0, RW = 0 */
        LPC17xx_SPI_SendRecvByte(0);
        LPC17xx_SPI_SendRecvByte(index);

        SPI_CS_HIGH;
}

/*****************************************************************************
* Function Name  : LCD_WriteData
* Description    : LCD write register data
* Input          : - data: register data
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
void LCD_WriteData( unsigned short data)
{
        SPI_CS_LOW;

        LPC17xx_SPI_SendRecvByte(SPI_START | SPI_WR | SPI_DATA);    /* Write : RS = 1, RW = 0 */
        LPC17xx_SPI_SendRecvByte((data >>   8));                    /* Write D8..D15          */
        LPC17xx_SPI_SendRecvByte((data & 0xFF));                    /* Write D0..D7           */

        SPI_CS_HIGH;
}
/*****************************************************************************
* Function Name  : LCD_Write_Data_Start
* Description    : Start of data writing to the LCD controller
* Input          : None
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
void LCD_Write_Data_Start(void)
{
  LPC17xx_SPI_SendRecvByte(SPI_START | SPI_WR | SPI_DATA);    /* Write : RS = 1, RW = 0       */
}
/*****************************************************************************
* Function Name  : LCD_Write_Data_Only
* Description    : Data writing to the LCD controller
* Input          : - data: data to be written
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
void LCD_Write_Data_Only( unsigned short data)
{
  LPC17xx_SPI_SendRecvByte((data >>   8));      /* Write D8..D15                */
  LPC17xx_SPI_SendRecvByte((data & 0xFF));      /* Write D0..D7                 */
}
/*****************************************************************************
* Function Name  : LCD_ReadData
* Description    : LCD read data
* Input          : None
* Output         : None
* Return         : return data
* Attention       : None
*****************************************************************************/
unsigned short LCD_ReadData(void)
{
        unsigned short value;

        SPI_CS_LOW;

        LPC17xx_SPI_SendRecvByte(SPI_START | SPI_RD | SPI_DATA);    /* Read: RS = 1, RW = 1         */
        LPC17xx_SPI_SendRecvByte(0);                                /* Dummy read 1                 */
        value   = LPC17xx_SPI_SendRecvByte(0);                      /* Read D8..D15                 */
        value <<= 8;
        value  |= LPC17xx_SPI_SendRecvByte(0);                      /* Read D0..D7                  */

        SPI_CS_HIGH;

        return value;
}
```

```
/*******************************************************************************
* Function Name  : LCD_WriteReg
* Description    : Writes to the selected LCD register.
* Input          : - LCD_Reg: address of the selected register.
*                  - LCD_RegValue: value to write to the selected register.
* Output         : None
* Return         : None
* Attention              : None
*******************************************************************************/
void LCD_WriteReg( unsigned short LCD_Reg, unsigned short LCD_RegValue)
{
        /* Write 16-bit Index, then Write Reg */
        LCD_WriteIndex(LCD_Reg);
        /* Write 16-bit Reg */
        LCD_WriteData(LCD_RegValue);
}

/*******************************************************************************
* Function Name  : LCD_WriteReg
* Description    : Reads the selected LCD Register.
* Input          : None
* Output         : None
* Return         : LCD Register Value.
* Attention              : None
*******************************************************************************/
unsigned short LCD_ReadReg( unsigned short LCD_Reg)
{
        unsigned short LCD_RAM;

        /* Write 16-bit Index (then Read Reg) */
        LCD_WriteIndex(LCD_Reg);
        /* Read 16-bit Reg */
        LCD_RAM = LCD_ReadData();

        return LCD_RAM;
}

/*******************************************************************************
* Function Name  : LCD_SetCursor
* Description    : Sets the cursor position.
* Input          : - Xpos: specifies the X position.
*                  - Ypos: specifies the Y position.
* Output         : None
* Return         : None
* Attention              : None
*******************************************************************************/
static void LCD_SetCursor( unsigned short Xpos, unsigned short Ypos )
{
        /* 0x9320 */
        LCD_WriteReg(0x0020, Xpos );
        LCD_WriteReg(0x0021, Ypos );
}
/*******************************************************************************
* Function Name  : LCD_Delay
* Description    : Delay Time
* Input          : - nCount: Delay Time
* Output         : None
* Return         : None
* Return         : None
* Attention              : None
*******************************************************************************/
static void delay_ms( unsigned short ms)
{
        unsigned short i,j;
        for( i = 0; i < ms; i++ )
        {
                for( j = 0; j < 1141; j++ );
        }
}
/*******************************************************************************
* Function Name  : LCD_Initializtion
* Description    : Initialize TFT Controller.
* Input          : None
* Output         : None
* Return         : None
* Attention              : None
*******************************************************************************/
void LCD_Initializtion(void)
{
        unsigned short DeviceCode;

        LCD_Configuration();
        DeviceCode = LCD_ReadReg(0x0000);          /* Read ID       */
        /* Different driver IC initialization*/
        if( DeviceCode == 0x9320 || DeviceCode == 0x9300 )
        {
                LCD_WriteReg(0x00,0x0000);
                LCD_WriteReg(0x01,0x0100); /* Driver Output Contral */
                LCD_WriteReg(0x02,0x0700); /* LCD Driver Waveform Contral */
                LCD_WriteReg(0x03,0x1038); /* Set the scan mode */
                LCD_WriteReg(0x04,0x0000); /* Scalling Contral */
                LCD_WriteReg(0x08,0x0202); /* Display Contral 2 */
                LCD_WriteReg(0x09,0x0000); /* Display Contral 3 */
```

```
        LCD_WriteReg(0x0a,0x0000); /* Frame Cycle Contal */
        LCD_WriteReg(0x0c,(1<<0)); /* Extern Display Interface Contral 1 */
        LCD_WriteReg(0x0d,0x0000); /* Frame Maker Position */
        LCD_WriteReg(0x0f,0x0000); /* Extern Display Interface Contral 2 */
        delay_ms(50);
        LCD_WriteReg(0x07,0x0101); /* Display Contral */
        delay_ms(50);
        LCD_WriteReg(0x10,(1<<12)|(0<<8)|(1<<7)|(1<<6)|(0<<4)); /* Power Control 1 */
        LCD_WriteReg(0x11,0x0007);                              /* Power Control 2 */
        LCD_WriteReg(0x12,(1<<8)|(1<<4)|(0<<0));                /* Power Control 3 */
        LCD_WriteReg(0x13,0x0b00);                              /* Power Control 4 */
        LCD_WriteReg(0x29,0x0000);                              /* Power Control 7 */
        LCD_WriteReg(0x2b,(1<<14)|(1<<4));

        LCD_WriteReg(0x50,0);       /* Set X Start */
        LCD_WriteReg(0x51,239);      /* Set X End */
        LCD_WriteReg(0x52,0);        /* Set Y Start */
        LCD_WriteReg(0x53,319);      /* Set Y End */
        delay_ms(50);

        LCD_WriteReg(0x60,0x2700); /* Driver Output Control */
        LCD_WriteReg(0x61,0x0001); /* Driver Output Control */
        LCD_WriteReg(0x6a,0x0000); /* Vertical Srcoll Control */

        LCD_WriteReg(0x80,0x0000); /* Display Position? Partial Display 1 */
        LCD_WriteReg(0x81,0x0000); /* RAM Address Start? Partial Display 1 */
        LCD_WriteReg(0x82,0x0000); /* RAM Address End-Partial Display 1 */
        LCD_WriteReg(0x83,0x0000); /* Displsy Position? Partial Display 2 */
        LCD_WriteReg(0x84,0x0000); /* RAM Address Start? Partial Display 2 */
        LCD_WriteReg(0x85,0x0000); /* RAM Address End? Partial Display 2 */

        LCD_WriteReg(0x90,(0<<7)|(16<<0)); /* Frame Cycle Contral */
        LCD_WriteReg(0x92,0x0000);         /* Panel Interface Contral 2 */
        LCD_WriteReg(0x93,0x0001);         /* Panel Interface Contral 3 */
        LCD_WriteReg(0x95,0x0110);         /* Frame Cycle Contral */
        LCD_WriteReg(0x97,(0<<8));
        LCD_WriteReg(0x98,0x0000);         /* Frame Cycle Contral */
        LCD_WriteReg(0x07,0x0133);
}
if( DeviceCode == 0x9325 || DeviceCode == 0x9328 )
{
        LCD_Code = ILI9325;
        LCD_WriteReg(0x00e7,0x0010);
        LCD_WriteReg(0x0000,0x0001);      /* start internal osc */
        LCD_WriteReg(0x0001,0x0100);
        LCD_WriteReg(0x0002,0x0700);      /* power on sequence */
        LCD_WriteReg(0x0003,(1<<12)|(1<<5)|(1<<4)|(0<<3) );        /* importance */
        LCD_WriteReg(0x0004,0x0000);
        LCD_WriteReg(0x0008,0x0207);
        LCD_WriteReg(0x0009,0x0000);
        LCD_WriteReg(0x000a,0x0000);      /* display setting */
        LCD_WriteReg(0x000c,0x0001);      /* display setting */
        LCD_WriteReg(0x000d,0x0000);
        LCD_WriteReg(0x000f,0x0000);
        /* Power On sequence */
        LCD_WriteReg(0x0010,0x0000);
        LCD_WriteReg(0x0011,0x0007);
        LCD_WriteReg(0x0012,0x0000);
        LCD_WriteReg(0x0013,0x0000);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0010,0x1590);
        LCD_WriteReg(0x0011,0x0227);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0012,0x009c);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0013,0x1900);
        LCD_WriteReg(0x0029,0x0023);
        LCD_WriteReg(0x002b,0x000e);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0020,0x0000);
        LCD_WriteReg(0x0021,0x0000);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0030,0x0007);
        LCD_WriteReg(0x0031,0x0707);
        LCD_WriteReg(0x0032,0x0006);
        LCD_WriteReg(0x0035,0x0704);
        LCD_WriteReg(0x0036,0x1f04);
        LCD_WriteReg(0x0037,0x0004);
        LCD_WriteReg(0x0038,0x0000);
        LCD_WriteReg(0x0039,0x0706);
        LCD_WriteReg(0x003c,0x0701);
        LCD_WriteReg(0x003d,0x000f);
        delay_ms(50);  /* delay 50 ms */
        LCD_WriteReg(0x0050,0x0000);
        LCD_WriteReg(0x0051,0x00ef);
        LCD_WriteReg(0x0052,0x0000);
        LCD_WriteReg(0x0053,0x013f);
        LCD_WriteReg(0x0060,0xa700);
        LCD_WriteReg(0x0061,0x0001);
        LCD_WriteReg(0x006a,0x0000);
        LCD_WriteReg(0x0080,0x0000);
        LCD_WriteReg(0x0081,0x0000);
        LCD_WriteReg(0x0082,0x0000);
```

```
                LCD_WriteReg(0x0083,0x0000);
                LCD_WriteReg(0x0084,0x0000);
                LCD_WriteReg(0x0085,0x0000);

                LCD_WriteReg(0x0090,0x0010);
                LCD_WriteReg(0x0092,0x0000);
                LCD_WriteReg(0x0093,0x0003);
                LCD_WriteReg(0x0095,0x0110);
                LCD_WriteReg(0x0097,0x0000);
                LCD_WriteReg(0x0098,0x0000);
                /* display on sequence */
                LCD_WriteReg(0x0007,0x0133);

                LCD_WriteReg(0x0020,0x0000);
                LCD_WriteReg(0x0021,0x0000);
        }
    delay_ms(100);   /* delay 50 ms */
}
/*****************************************************************************
* Function Name  : LCD_Clear
* Description    : Fill the screen as the specified color
* Input          : - Color: Screen Color
* Output         : None
* Return         : None
* Attention              : None
*****************************************************************************/
void LCD_Clear( unsigned short Color)
{
        unsigned int index=0;

        LCD_SetCursor(0,0);

        LCD_WriteIndex(0x0022);

        SPI_CS_LOW;
        LCD_Write_Data_Start();

        for( index = 0; index < MAX_X * MAX_Y; index++ )
        {
                LCD_Write_Data_Only(Color);
        }
        SPI_CS_HIGH;
}
/*****************************************************************************
* Function Name  : LCD_BGR2RGB
* Description    : RRRRRGGGGGGBBBBB To BBBBBGGGGGGRRRRR
* Input          : - color: BRG Color value
* Output         : None
* Return         : RGB Color value
* Attention              : None
*****************************************************************************/
static unsigned short LCD_BGR2RGB( unsigned short color)
{
        unsigned short  r, g, b, rgb;

        b = ( color>>0 )  & 0x1f;
        g = ( color>>5 )  & 0x3f;
        r = ( color>>11 ) & 0x1f;

        rgb =  (b<<11) + (g<<5) + (r<<0);

        return( rgb );
}
/*****************************************************************************
* Function Name  : LCD_GetPoint
* Description    : Get color value for the specified coordinates
* Input          : - Xpos: Row Coordinate
*                  - Xpos: Line Coordinate
* Output         : None
* Return         : Screen Color
* Attention              : None
*****************************************************************************/
unsigned short LCD_GetPoint( unsigned short Xpos, unsigned short Ypos)
{
        unsigned short dummy;

        LCD_SetCursor(Xpos,Ypos);

        LCD_WriteIndex(0x0022);

        dummy = LCD_ReadData();   /* An empty read */
        dummy = LCD_ReadData();

        return  LCD_BGR2RGB( dummy );
}
/*****************************************************************************
* Function Name  : LCD_SetPoint
* Description    : Drawn at a specified point coordinates
* Input          : - Xpos: Row Coordinate
*                  - Ypos: Line Coordinate
* Output         : None
* Return         : None
* Attention              : None
```

```
*****************************************************************************/
void LCD_SetPoint( unsigned short Xpos, unsigned short Ypos, unsigned short point)
{
        if( Xpos >= MAX_X || Ypos >= MAX_Y )
        {
                return;
        }
        LCD_SetCursor(Xpos,Ypos);
        LCD_WriteReg(0x0022,point);
}

/*****************************************************************************
* Function Name  : LCD_DrawLine
* Description    : Bresenham's line algorithm
* Input          : - x1: A point line coordinates
*                  - y1: A point column coordinates
*                   - x2: B point line coordinates
*                   - y2: B point column coordinates
*                   - color: Line color
* Output         : None
* Return         : None
* Attention               : None
*****************************************************************************/
void LCD_DrawLine( uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1 , uint16_t color )
{
    short dx,dy;      /* The definition of the X Y axis increase the value of the variable */
    short temp;

    if( x0 > x1 )
    {
            temp = x1;
            x1 = x0;
            x0 = temp;
    }
    if( y0 > y1 )
    {
            temp = y1;
            y1 = y0;
            y0 = temp;
    }

        dx = x1-x0;
        dy = y1-y0;

    if( dx == 0 )
    {
        do
        {
            LCD_SetPoint(x0, y0, color);
            y0++;
        }
        while( y1 >= y0 );
                return;
    }
    if( dy == 0 )
    {
        do
        {
            LCD_SetPoint(x0, y0, color);
            x0++;
        }
        while( x1 >= x0 );
                return;
    }

    if( dx > dy )
    {
            temp = 2 * dy - dx;
        while( x0 != x1 )
        {
                LCD_SetPoint(x0,y0,color);
                x0++;
                if( temp > 0 )
                {
                    y0++;
                    temp += 2 * dy - 2 * dx;
                     }
            else
            {
                                temp += 2 * dy;
                        }
        }
        LCD_SetPoint(x0,y0,color);
    }
    else
    {
            temp = 2 * dx - dy;
        while( y0 != y1 )
        {
                     LCD_SetPoint(x0,y0,color);
            y0++;
            if( temp > 0 )
            {
```

```
                x0++;
                temp+=2*dy-2*dx;
            }
            else
                        {
                temp += 2 * dy;
                        }
        }
        LCD_SetPoint(x0,y0,color);
         }
}
/*****************************************************************************
* Function Name  : PutChar
* Description    : Lcd screen displays a character
* Input          : - Xpos: Horizontal coordinate
*                  - Ypos: Vertical coordinate
*                                  - ASCI: Displayed character
*                                  - charColor: Character color
*                                  - bkColor: Background color
* Output         : None
* Return         : None
* Attention               : None
*****************************************************************************/
void PutChar( uint16_t Xpos, uint16_t Ypos, uint8_t ASCI, uint16_t charColor, uint16_t bkColor )
{
        uint16_t i, j;
    uint8_t buffer[16], tmp_char;
    GetASCIICode(buffer,ASCI);  /* get font data */
    for( i=0; i<16; i++ )
    {
        tmp_char = buffer[i];
        for( j=0; j<8; j++ )
        {
            if( (tmp_char >> 7 - j) & 0x01 == 0x01 )
            {
                LCD_SetPoint( Xpos + j, Ypos + i, charColor );  /* Character color */
            }
            else
            {
                LCD_SetPoint( Xpos + j, Ypos + i, bkColor );  /* Background color */
            }
        }
    }
}
/*****************************************************************************
* Function Name  : GUI_Text
* Description    : Displays the string
* Input          : - Xpos: Horizontal coordinate
*                  - Ypos: Vertical coordinate
*                                  - str: Displayed string
*                                  - charColor: Character color
*                                  - bkColor: Background color
* Output         : None
* Return         : None
* Attention                : None
*****************************************************************************/
void GUI_Text(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor)
{
    uint8_t TempChar;
    do
    {
        TempChar = *str++;
        PutChar( Xpos, Ypos, TempChar, Color, bkColor );
        if( Xpos < MAX_X - 8 )
        {
            Xpos += 8;
        }
        else if ( Ypos < MAX_Y - 16 )
        {
            Xpos = 0;
            Ypos += 16;
        }
        else
        {
            Xpos = 0;
            Ypos = 0;
        }
    }
    while ( *str != 0 );
}

/*****************************************************************************************************************
      END FILE
*****************************************************************************************************************/
```