```
/*
* lcd_dog_AVR128_driver.h
*
* Created: 3/19/2022 8:09:13 PM
* Author : jason
*/

#include <avr/io.h>
#include <stdio.h>

/*
Is what is responsible for transmitting the command for the LCD
*/
void lcd_spi_transmit_CMD (unsigned char cmd);

/*
Is what is responsible for transmitting the data for the LCD into what is to be displayed
*/
void lcd_spi_transmit_DATA (unsigned char cmd);

/*
Initialize the SPI LCD to being blank
*/
void init_spi_lcd (void);

/*
Initialize the LCD Dog to being blank
*/
void init_lcd_dog (void);

/*
Function prototype for 40 ms
*/
void delay_40mS(void);

/*
Function prototype for 40 microsecond
*/
void delay_30uS(void);

/*
To update on the LCD for something to be displayed
*/
void update_lcd_dog(void);


// Display buffer for DOG LCD using sprintf()
char dsp_buff1[17];
char dsp_buff2[17];
char dsp_buff3[17];
```

```
void lcd_spi_transmit_CMD (unsigned char cmd) {
    //Poll until ready to send the command
    //while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTC_OUT &= ~PIN0_bm;  //Clear PC0 = RS = 0 = command
    PORTA_OUT &= ~PIN7_bm;  //clear PA7 = /SS = selected
    SPI0_DATA = cmd;
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTA_OUT |= PIN7_bm;   //clear PA7 = /SS = selected

}

void lcd_spi_transmit_DATA (unsigned char cmd) {
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTC_OUT |= PIN0_bm;   //PC0 = RS = 1 = command
    PORTA_OUT &= ~PIN7_bm;  //clear PA7 = /SS = selected
    SPI0_DATA = cmd;
    //Poll until ready to send the command
    while(!(SPI0_INTFLAGS & SPI_IF_bm)){}
    PORTA_OUT |= PIN7_bm;   //clear PA7 = /SS = selected

}

void init_spi_lcd (void) {
    PORTA_DIR |= PIN4_bm | PIN6_bm | PIN7_bm;  //Set MOSI, SCK and //SS as output
      while MISO as input

    PORTC_DIR |= PIN0_bm;  //Set RS of LCD as output

    SPI0_CTRLA |= SPI_ENABLE_bm | SPI_MASTER_bm; //Enable the SPI and make it in the
      Master Mode

    SPI0_CTRLB |= SPI_SSD_bm | SPI_MODE1_bm | SPI_MODE0_bm;   //Put the SPI with slave
       select (/SS) to be enabled and be in SPI Mode 3 (CPOL = 1 and CPHA = 1)

    //Wait to clears the IF flag in the INTFLAG meaning there no serial data yet to be
       transferred
    //while(SPI0_INTFLAGS & SPI_IF_bm){}
    PORTC_OUT &= ~PIN0_bm;  //PC0 = RS = 0 = command



}


void init_lcd_dog (void) {

    init_spi_lcd();     //Initialize mcu for LCD SPI
```

```
    //start_dly_40ms:
    delay_40mS();    //startup delay.


    //func_set1:
    lcd_spi_transmit_CMD(0x39);   // sedn function set #1
    delay_30uS();   //delay for command to be processed


    //func_set2:
    lcd_spi_transmit_CMD(0x39); //send fuction set #2
    delay_30uS();   //delay for command to be processed


    //bias_set:
    lcd_spi_transmit_CMD(0x1E); //set bias value.
    delay_30uS();   //delay for command to be processed


    //power_ctrl:
    lcd_spi_transmit_CMD(0x55); //~ 0x50 nominal for 5V
    //~ 0x55 for 3.3V (delicate adjustment).
    delay_30uS();   //delay for command to be processed


    //follower_ctrl:
    lcd_spi_transmit_CMD(0x6C); //follower mode on...
    delay_40mS();   //delay for command to be processed


    //contrast_set:
    lcd_spi_transmit_CMD(0x7F); //~ 77 for 5V, ~ 7F for 3.3V
    delay_30uS();   //delay for command to be processed


    //display_on:
    lcd_spi_transmit_CMD(0x0c); //display on, cursor off, blink off
    delay_30uS();   //delay for command to be processed


    //clr_display:
    lcd_spi_transmit_CMD(0x01); //clear display, cursor home
    delay_30uS();   //delay for command to be processed


    //entry_mode:
    lcd_spi_transmit_CMD(0x06); //clear display, cursor home
    delay_30uS();   //delay for command to be processed
}


void delay_40mS(void) {
```

```
    int i;
    for (int n = 40; n > 0; n--)
    for (i = 0; i < 800; i++)
    __asm("nop");
}

void delay_30uS(void) {
    int i;
    for (int n = 1; n > 0; n--)
    for (i = 0; i < 2; i++)
    __asm("nop");
}


// Updates the LCD display lines 1, 2, and 3, using the
// contents of dsp_buff_1, dsp_buff_2, and dsp_buff_3, respectively.
void update_lcd_dog(void) {

    init_spi_lcd();     //init SPI port for LCD.

    // send line 1 to the LCD module.
    lcd_spi_transmit_CMD(0x80); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff1[i]);
        delay_30uS();
    }

    // send line 2 to the LCD module.
    lcd_spi_transmit_CMD(0x90); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff2[i]);
        delay_30uS();
    }

    // send line 3 to the LCD module.
    lcd_spi_transmit_CMD(0xA0); //init DDRAM addr-ctr
    delay_30uS();
    for (int i = 0; i < 16; i++) {
        lcd_spi_transmit_DATA(dsp_buff3[i]);
        delay_30uS();
    }
}
```