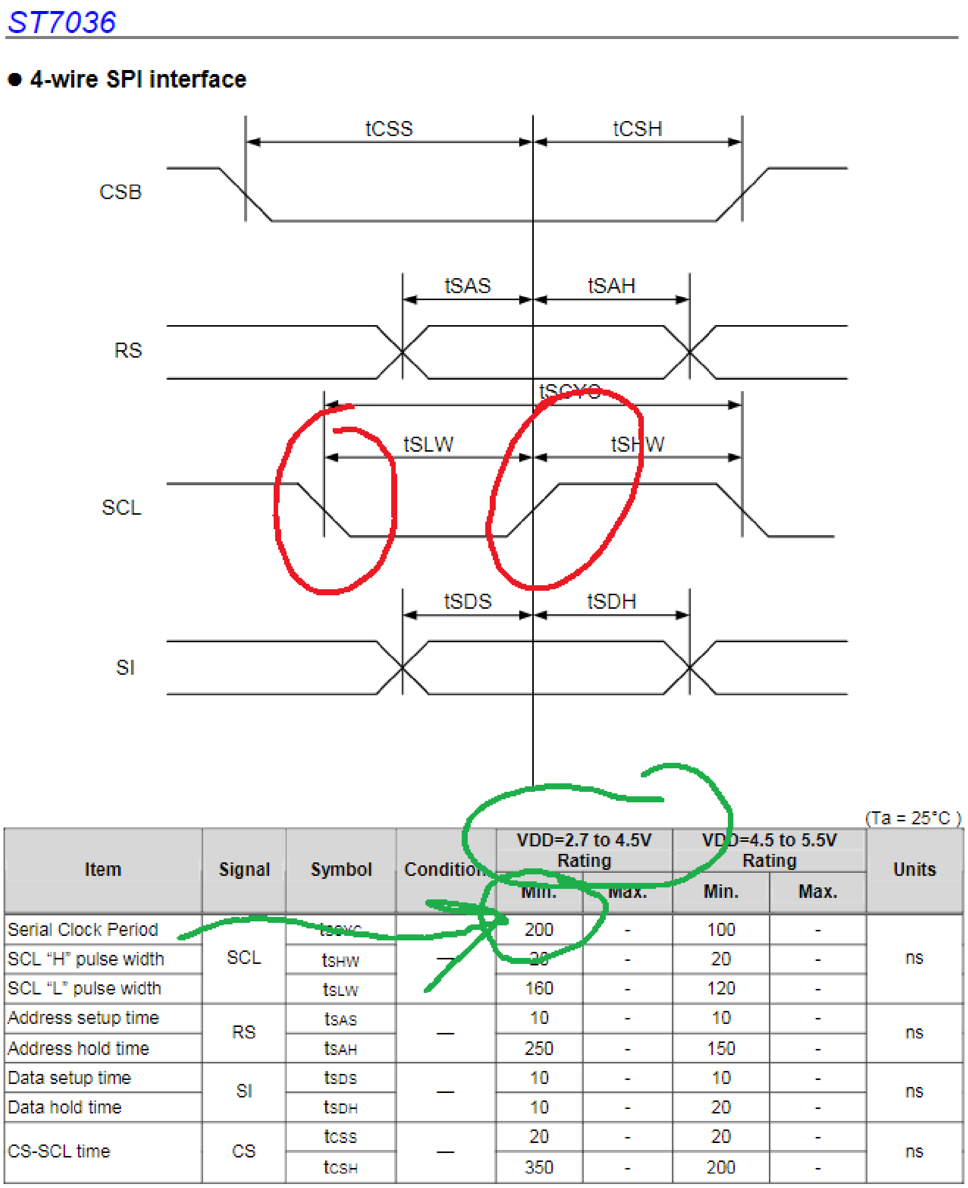
Laboratory 08: AVR128DB48 C Driver for DOGM163W-A LCD

CHRISTOPHER NIELSEN + CHRISTOPHER SHAMAH
KENNETH SHORT
Bench No: 17
ID#114211318
ID#112229076
Lab Section L01

```
/*
 * DOG_LCD_BasicTest.c
 *
 * Created: 3/25/2024 11:12:25 PM
 * Author : MysticOwl
 */
#include <avr/io.h>
#define F_CPU 4000000 //freq
#include <util/delay.h>
#include <stdio.h>

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

void lcd_spi_transmit_CMD (char cmd);  //macro for multiple write spi
functions for a setup command
void lcd_spi_transmit_DATA (char cmd); //macro for multiple write spi
functions for any data to be sent
void init_spi_lcd (void); //init spi0 settings for avr
void init_lcd_dog (void); //finish init commands for dog
void update_lcd_dog(void); //send buffer for line data

void init_spi_lcd (){

        PORTA.DIR |= PIN4_bm; /* Set MOSI pin direction to output */
        PORTA.DIR &= ~PIN5_bm; /* Set MISO pin direction to input */
        PORTA.DIR |= PIN6_bm; /* Set SCK pin direction to output */
        PORTA.DIR |= PIN7_bm; /* Set SS pin direction to output */
        PORTA.OUT |= PIN7_bm; /* Set SS pin direction to output */

        PORTC.DIR |= PIN0_bm; //Reg select output to the display
memory
        PORTC.OUT &= ~PIN0_bm;

        SPI0.CTRLB |= (SPI_SSD_bm | 0x03 ); // mode 3 as per the dog
waveforms

        SPI0.CTRLA = SPI_ENABLE_bm | SPI_MASTER_bm;

        init_lcd_dog();
}


void lcd_spi_transmit_CMD (char data){

        PORTA_OUT &= ~PIN7_bm; //Slave select ON
        PORTC.OUT &= ~PIN0_bm; // register select 0, command setting
        SPI0.DATA = data;
        while (!(SPI0.INTFLAGS & SPI_IF_bm))  /* waits until data is 
exchanged*/
        {
                asm volatile ("nop");
        }
        volatile uint8_t dummy;
        dummy = SPI0_DATA;


        PORTF_OUT = PIN7_bm; //Slave select OFF

}

void lcd_spi_transmit_DATA (char data){

        PORTA_OUT &= PIN7_bm; //Slave select ON
        PORTC.OUT = PIN0_bm;  // register select 1, data setting
        SPI0.DATA = data;

        while (!(SPI0.INTFLAGS & SPI_IF_bm))  /* waits until data is 
exchanged*/
        {
        asm volatile ("nop");
        }


        PORTF_OUT = PIN7_bm; //Slave select OFF

}



void init_lcd_dog(){

                //start_dly_40ms:
                _delay_ms(90);    //startup delay.

                //func_set1:
                lcd_spi_transmit_CMD(0x39);   // send function set 
#1 //tell for 3 lines and data interface at 8 bits
                _delay_us(30);  //delay for command to be processed


                //func_set2:
                lcd_spi_transmit_CMD(0x39);     //send function set 
#2 // again??
                _delay_us(30);  //delay for command to be processed
                //bias_set:
                lcd_spi_transmit_CMD(0x1E);     //set bias value.
                _delay_us(30);  //delay for command to be processed


                //power_ctrl:
                lcd_spi_transmit_CMD(0x55);     //~ 0x50 nominal for
5V
                //~ 0x55 for 3.3V (delicate adjustment).
                _delay_us(30);  //delay for command to be processed


                //follower_ctrl:
                lcd_spi_transmit_CMD(0x6C);     //follower mode on...
                _delay_ms(220); //delay for command to be processed
SPECIAL CASE


                //contrast_set:
                lcd_spi_transmit_CMD(0x7F);     //~ 77 for 5V, ~ 7F
for 3.3V
                _delay_us(30);  //delay for command to be processed


                //display_on:
                lcd_spi_transmit_CMD(0x0c);     //display on, cursor
off, blink off
                _delay_us(30);  //delay for command to be processed


                //clr_display:
                lcd_spi_transmit_CMD(0x01);     //clear display,
cursor home
                _delay_us(420); //delay for command to be processed


                //entry_mode:
                lcd_spi_transmit_CMD(0x06);     //clear display,
cursor home
                _delay_us(30);  //delay for command to be processed


}



// 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_us(30);  //delay for command to be processed
        for (int i = 0; i < 16; i++) {
                lcd_spi_transmit_DATA(dsp_buff1[i]);
                _delay_us(30);  //delay for command to be processed
        }

        // send line 2 to the LCD module.
        lcd_spi_transmit_CMD(0x90);     //init DDRAM addr-ctr
        _delay_us(30);  //delay for command to be processed
        for (int i = 0; i < 16; i++) {
                lcd_spi_transmit_DATA(dsp_buff2[i]);
                _delay_us(30);  //delay for command to be processed
        }

        // send line 3 to the LCD module.
        lcd_spi_transmit_CMD(0xA0);     //init DDRAM addr-ctr
        _delay_us(30);  //delay for command to be processed
        for (int i = 0; i < 16; i++) {
                lcd_spi_transmit_DATA(dsp_buff3[i]);
                _delay_us(30);  //delay for command to be processed
        }
}



int main(void)
{

        init_spi_lcd();

    while (1)
    {

                sprintf(dsp_buff1, "     ESE 381    ");
                sprintf(dsp_buff2, "Variable Voltage");
                sprintf(dsp_buff3, "   MAX5402 Lab  ");
                update_lcd_dog();

    }
}
```



ST7036

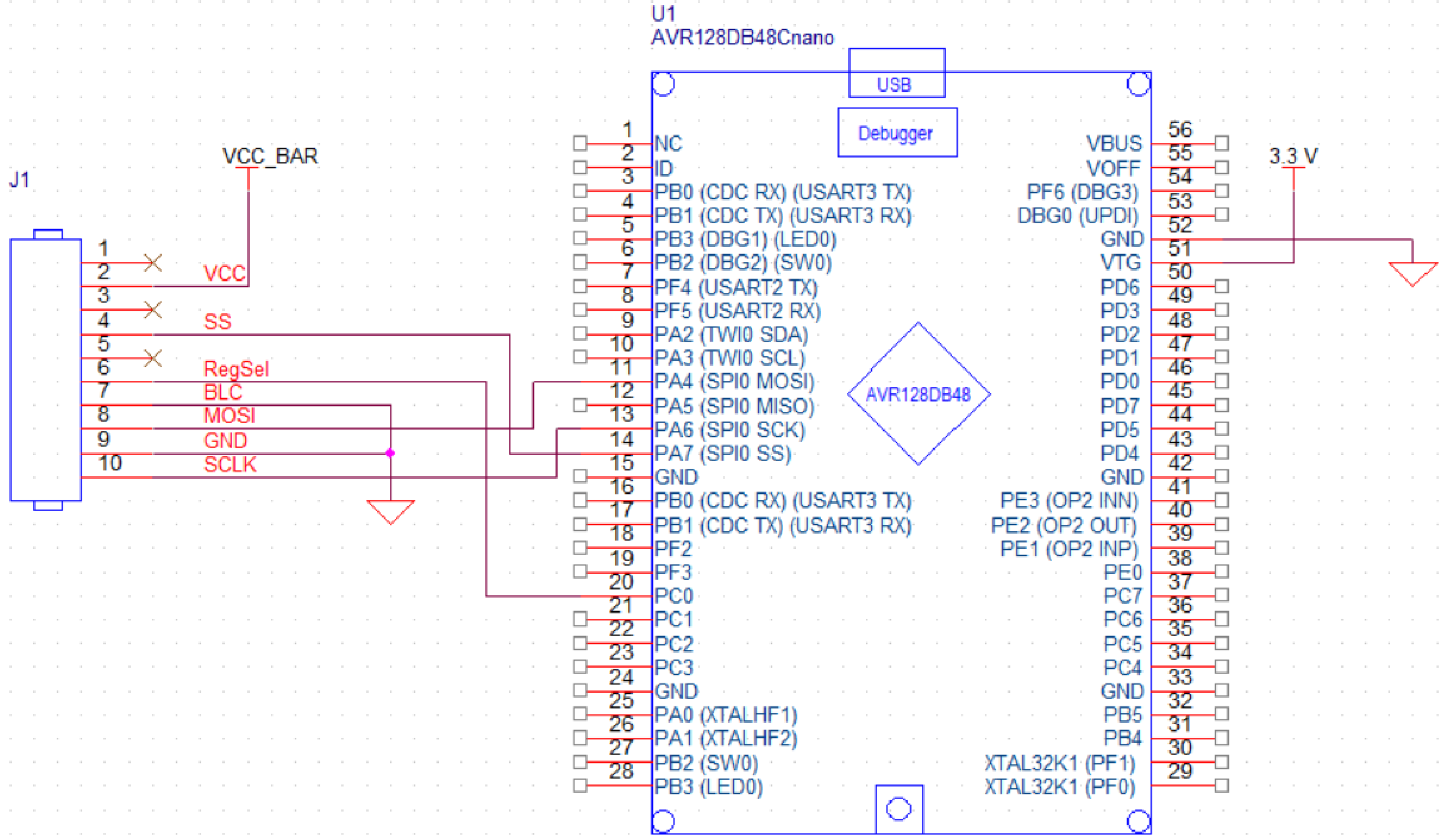
● 4-wire SPI interface

(Ta = 25°C)

Item	Signal	Symbol	Condition	VDD=2.7 to 4.5V Rating		VDD=4.5 to 5.5V Rating		Units
				Min.	Max.	Min.	Max.	
Serial Clock Period	SCL	tSCYC	—	200	-	100	-	ns
SCL "H" pulse width		tSHW		20	-	20	-	
SCL "L" pulse width		tSLW		160	-	120	-	
Address setup time	RS	tSAS	—	10	-	10	-	ns
Address hold time		tSAH		250	-	150	-	
Data setup time	SI	tSDS	—	10	-	10	-	ns
Data hold time		tSDH		10	-	20	-	
CS-SCL time	CS	tCSS	—	20	-	20	-	ns
		tCSH		350	-	200	-	

1) The SPI mode is mode 3, because looking at the red circles above, the clock is inactive at the first circle on a one, meaning CPOL is 1, and the data is sampled in the second circle, on the trailing edge, meaning CPHA is also 1. This corresponds to mode 3.

2) The bitrate can be calculated as one over the minimum serial clock period, the Min SCP is ns as seen circled in green above, this means the bit rate is 5Mhz


U1
AVR128DB48Cnano
USB
Debugger
AVR128DB48
J1
VCC_BAR
3.3 V
1
2
3
4
5
6
7
8
9
10
VCC
SS
RegSel
BLC
MOSI
GND
SCLK
1 NC
2 ID
3 PB0 (CDC RX) (USART3 TX)
4 PB1 (CDC TX) (USART3 RX)
5 PB3 (DBG1) (LED0)
6 PB2 (DBG2) (SW0)
7 PF4 (USART2 TX)
8 PF5 (USART2 RX)
9 PA2 (TWI0 SDA)
10 PA3 (TWI0 SCL)
11 PA4 (SPI0 MOSI)
12 PA5 (SPI0 MISO)
13 PA6 (SPI0 SCK)
14 PA7 (SPI0 SS)
15 GND
16 PB0 (CDC RX) (USART3 TX)
17 PB1 (CDC TX) (USART3 RX)
18 PF2
19 PF3
20 PC0
21 PC1
22 PC2
23 PC3
24 GND
25 PA0 (XTALHF1)
26 PA1 (XTALHF2)
27 PB2 (SW0)
28 PB3 (LED0)
56 VBUS
55 VOFF
54 PF6 (DBG3)
53 DBG0 (UPDI)
52 GND
51 VTG
50 PD6
49 PD3
48 PD2
47 PD1
46 PD0
45 PD7
44 PD5
43 PD4
42 GND
41 PE3 (OP2 INN)
40 PE2 (OP2 OUT)
39 PE1 (OP2 INP)
38 PE0
37 PC7
36 PC6
35 PC5
34 PC4
33 GND
32 PB5
31 PB4
30 XTAL32K1 (PF1)
29 XTAL32K1 (PF0)