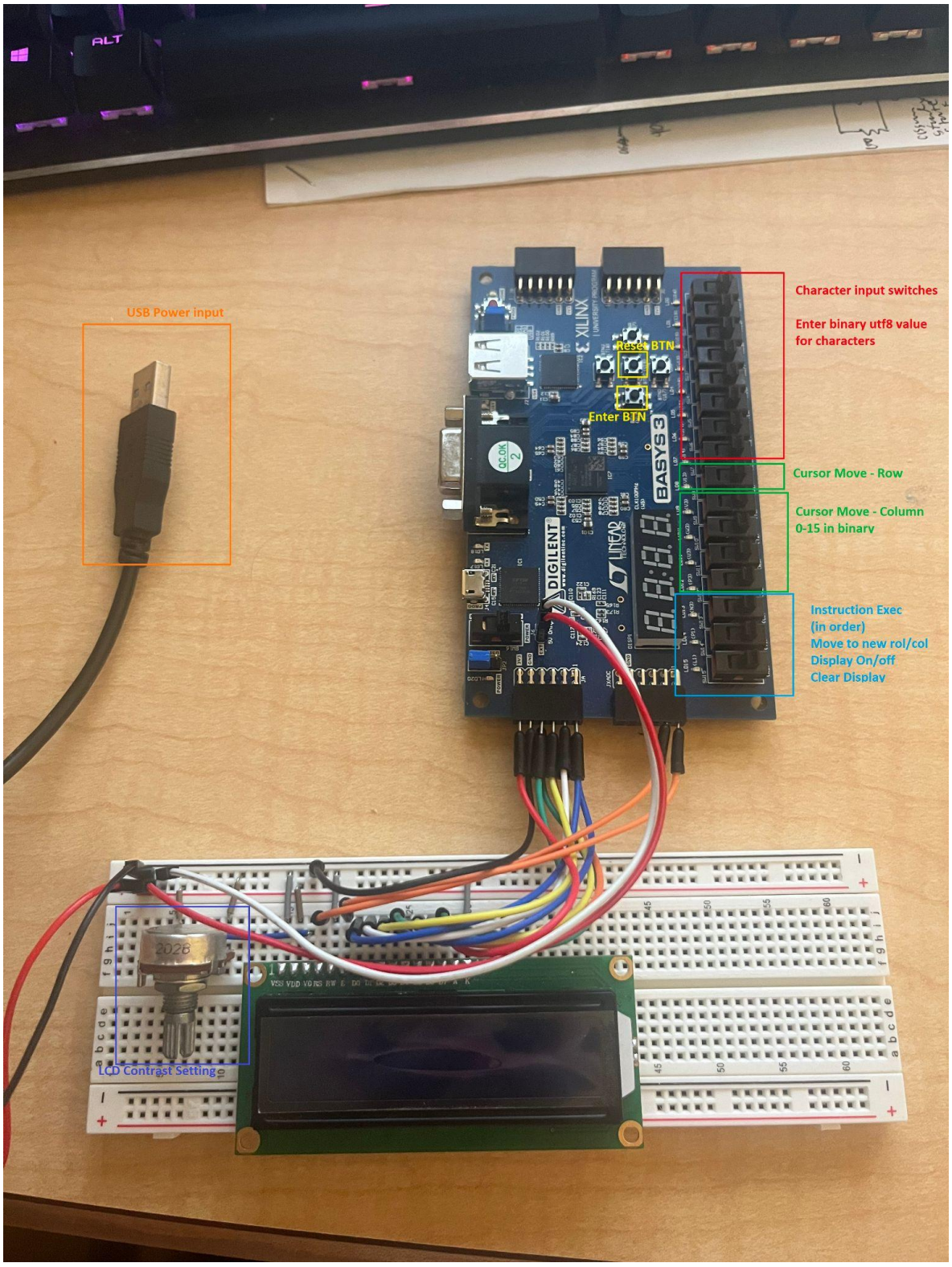
LCD Demo Project

Introduction:

The device designed uses the OTTER MCU to provide a demo of a Liquid Crystal Display. The device lets you demo the basic features of the LCD such as printing characters, moving cursors, clearing, and display on/off. The contrast on the display can also be controlled by an external potentiometer. The input switches are only checked when there is an interrupt button press. Any utf8 character can be displayed on the lcd. The device needs a usb plug to connect to in order to provide power for the peripheral circuit and Basys3 board.

Operating Instructions:



Controls Overview

Setup

To setup the device the usb cable needs to be plugged into a usb port to supply power to the device. When properly powered on the device will display a welcome message of “CPE233” on the first line and “Fortnite” on the second line. Additionally the LCD Contrast potentiometer can be adjusted to find the right contrast setting for the user. The welcome message can be useful for finding the right contrast setting.

The Buttons

The Buttons control when a function is actually executed. When the Reset BTN is pressed the device will reset to its original state with the default message on the LCD. When the Enter BTN is pressed the device will execute the current instruction specified by the Switches.

The Switches

The Switches determine the function that will be executed by the OTTER MCU and the LCD. There are 3 sets of switches: the instruction switches, cursor row/col switches, and the character input switches.

Instruction Switches

The instruction switches are checked from bottom to top in the overview picture, checking clear display first and move row/col last. Make sure to only have the instruction switch on for the function you want. If none of the instruction switches are flipped on then on a enter button press the device will print the character specified by the character switches on the display at the current row/column.

Clear Display

If the clear display is flipped the display will completely clear its memory on the press of the Enter button press and reset the cursor to the top left of the LCD.

Display On/off

If the display on/off switch is flipped on then the display will turn off at the press of the Enter button. This does not clear the display or move the cursor. When the switch is returned to

the off position the display will turn back on with the current data on the screen and correct cursor position.

Move Cursor Row/Column

If the move cursor row/column switch is on then on the next press of the Enter button the cursor for inputting characters will move the position specified by the row/col switches.

The Column Switches are the 4 switches above the move cursor switch in the Overview diagram. Enter the binary value of the new column position with the lsb at the top. The LCD defines cursor column zero as the furthest left column.

The Row switch controls whether the new cursor position is in row 1 or row 2. When the row switch is in the off position the new cursor position will be on the top line. While in the on position the new cursor position will be on the bottom line.

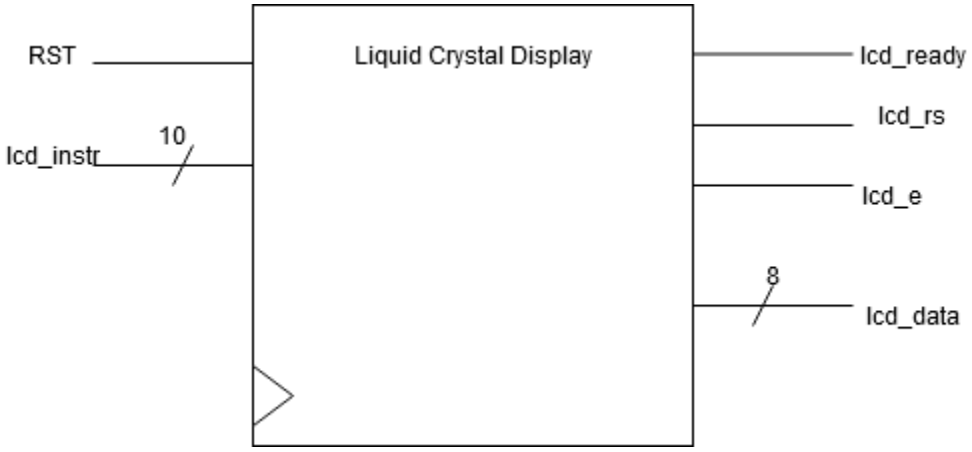
Character Switches

The Character Switches are checked if none of the instruction switches are in the on position. To print a character to the screen enter the utf8 coding of the character in binary into the switches with the least significant bit at the top in the overview diagram. Then press the enter button and the character will be displayed on the LCD screen at the current row/column. The column is automatically incremented by 1 when a character is entered, but the row does not automatically wrap around.

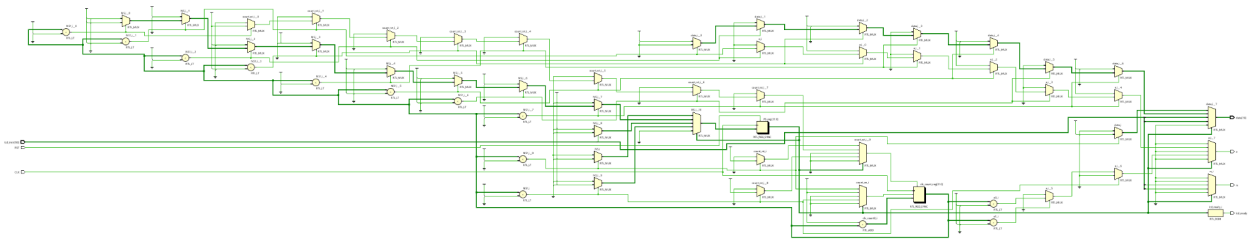
Peripheral Details:

Liquid Crystal Display

The liquid crystal display hardware on the Basys3 takes a 10 bit instruction from the IO Bus with the 10th bit being the send signal. It outputs the e, rs, and data signals to the external lcd and a lcd_ready signal back into the Basys3 board. The hardware includes a reset sequence so reset does not have to be done through software. This was done because there is lots of extra space on the basys3 for the initialization hardware to be included. The addition of the lcd_ready signal allows for the enable pulse to be sent by the hardware as well instead of having to determine clock cycle waiting time in software. It takes the same 50Mhz clock as the OTTER MCU to run.

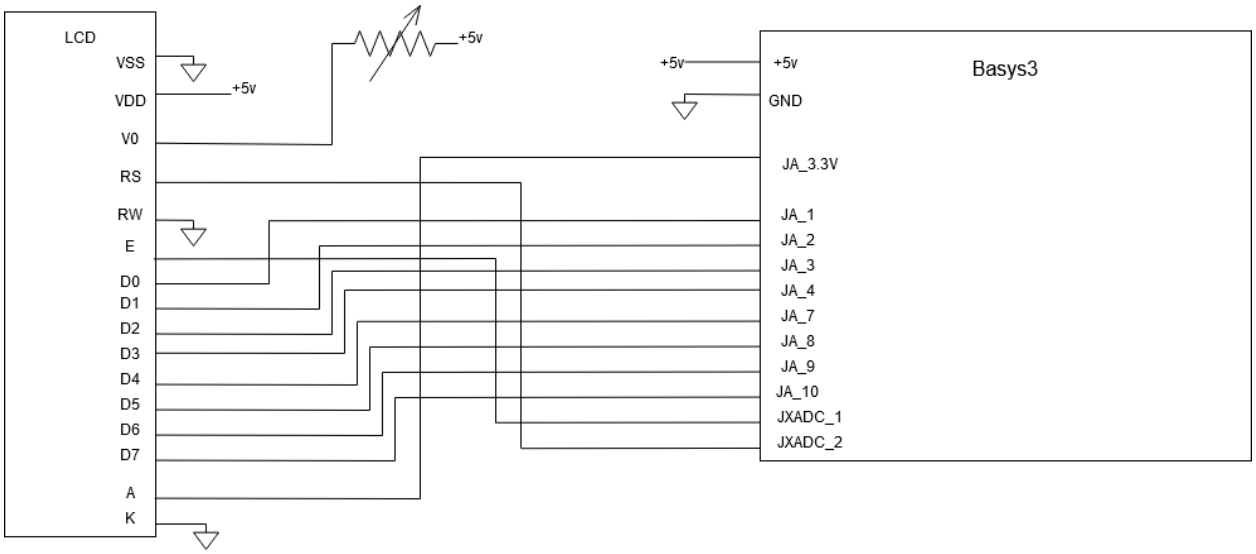


Block Diagram

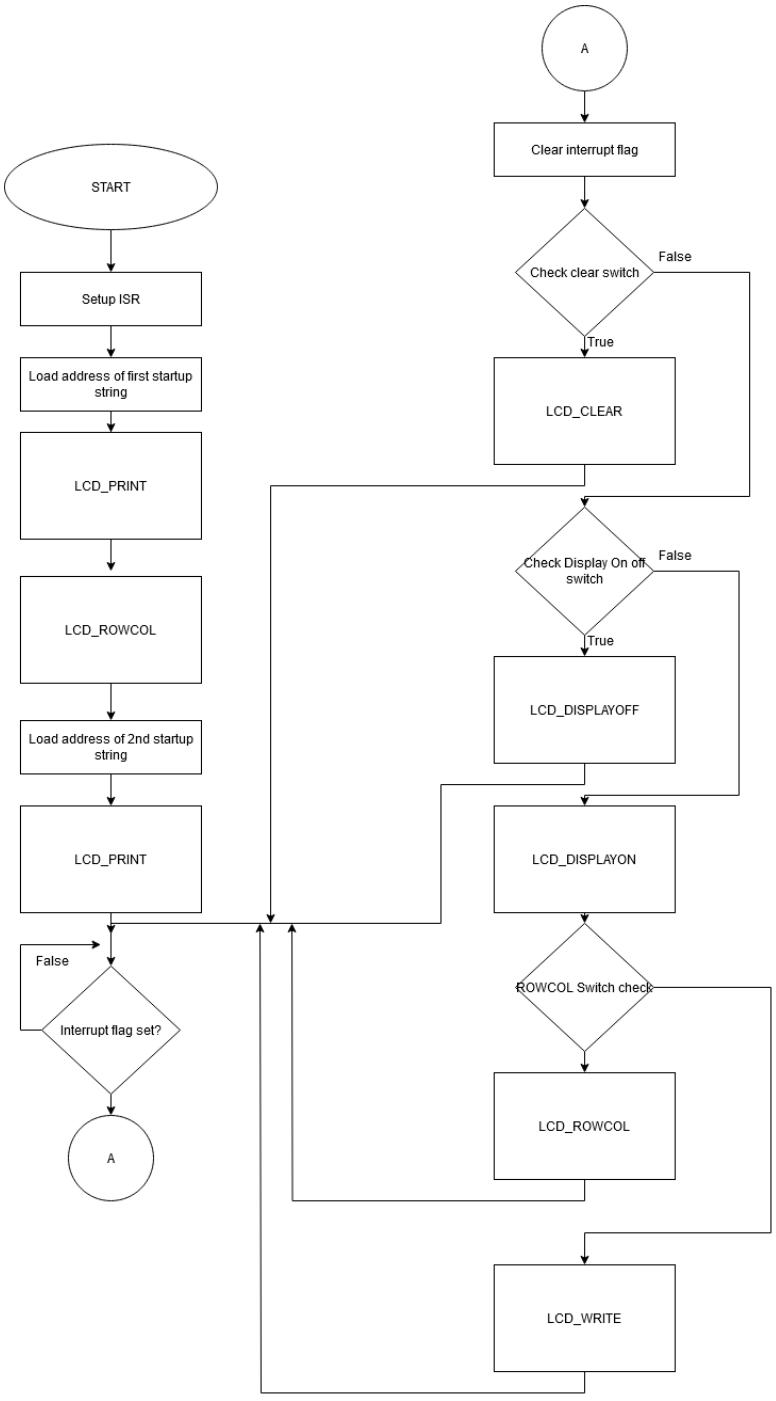


Elaborated Design

External Circuit Peripheral:

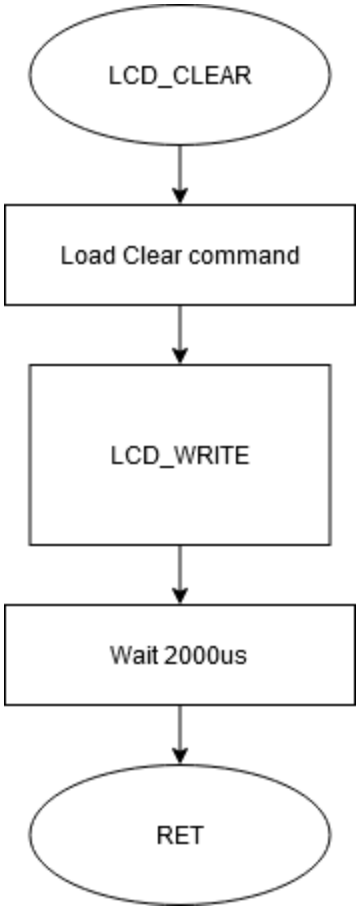


Software Design:



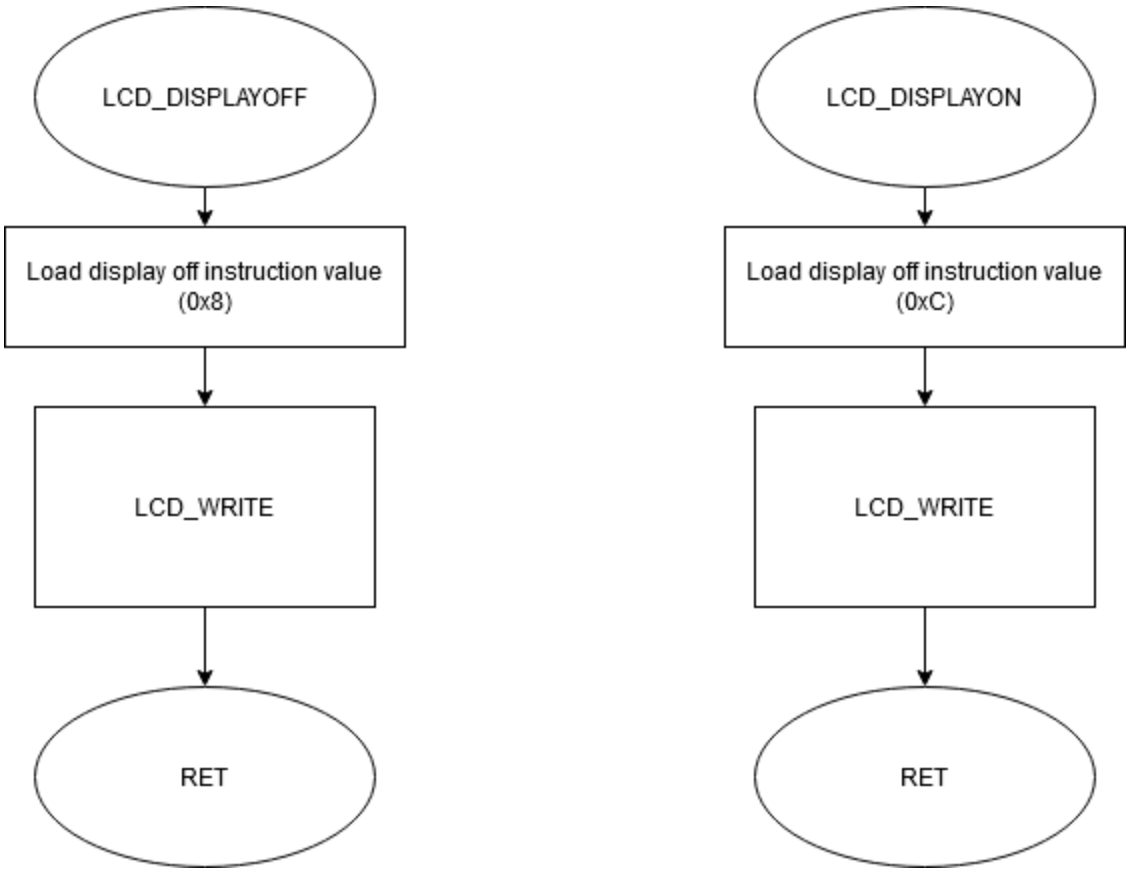
Main Program

Waits for an interrupt then cycles through the instruction switches to determine what instruction to run on the lcd. If no instructions switches then writes the current character switches to the lcd.



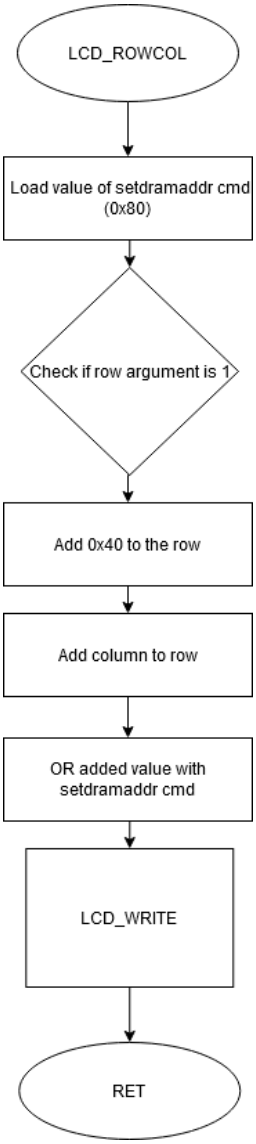
LCD clear subroutine

The LCD Clear subroutine loads the value of the clear instruction(0x1) and then calls LCD_Write. It then waits 2000 us because the clear instruction takes a while to finish.



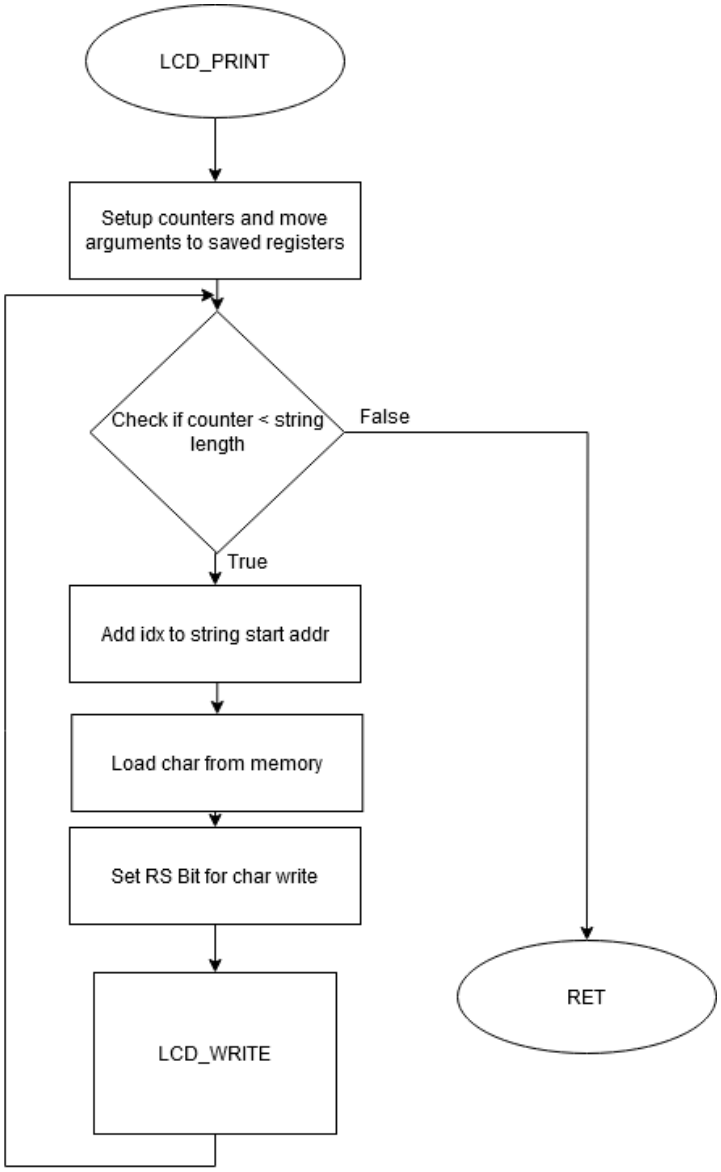
LCD Display On/Off subroutines

The Display on off subroutines load the value for their respective functions and then call the LCD_WRITE subroutine.



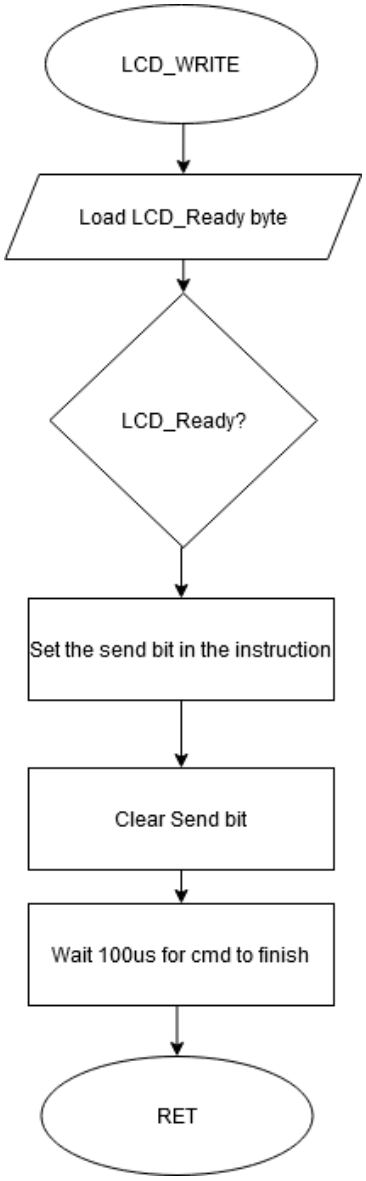
Move LCD Row/col subroutine

The subroutine first loads the value of the setdramaddr command which will tell the lcd to move the cursor. Then it checks if the row argument is 0 or 1. Row 0 starts at 0x0 while row 1 starts at 0x40 so 0x40 is added to value if row 1. Then the row and column are added and or'd with the original command to get the full command with the new position. LCD_WRITE is then called to execute the instruction.



LCD_PRINT subroutine

The LCD Print subroutine takes two arguments the address of the start of the string and the length of the string. It loops through loading each character from memory and calling LCD_WRITE to print it to the lcd.



LCD_WRITE subroutine

This subroutine waits until the LCD is ready for a command. Then it sets the send bit in the argument and writes it to the LCD. Immediately after it clears the send bit. The short pulse is enough for the lcd hardware to go into the correct state and send the correct length pulse to the actual display. Lastly it waits 100us for commands to finish executing.

Appendix

LCD_Demo.s

```
.eqv MMIO, 0x11000000
.eqv STACK, 0x10000

.data
STRING:   .byte 0x46, 0x4f, 0x52, 0x54, 0x4e, 0x49, 0x54, 0x45
#FORTNITE
STRING2: .byte 0x43, 0x50, 0x45, 0x32, 0x33, 0x33 #CPE233


.text

MAIN:          li sp, STACK        #Stack address
          li s0, MMIO         #MMIO address
          li s1, 0       #interrupt flag

          #Setup interrupts
          la    t0, ISR
               csrrw x0, mtvec, t0      # setup ISR address
          li    t0, 8          #INT EN
               csrrs x0, mstatus, t0  # enable interrupts

          la a0, STRING2

          li t1, 1
          sw t1, 0x20(s0)

          addi a1, zero, 6

          call LCD_PRINT

          li a0, 1
          li a1, 4
          call LCD_ROWCOL
          la a0, STRING
          addi a1, zero, 8
          call LCD_PRINT

LOOP:          beqz s1, LOOP
          addi s1, zero, 0    #Reset interrupt flag
```

```
          lhu s2, (s0)        #Read Switches

          #Check for display clear (Switch 15)
CLEAR_CHECK:   li t1, 0x8000
          and t0, s2, t1
          beqz t0, ONOFF_CHECK
          call LCD_CLEAR
          j LOOP

          #Check for display on off (Switch 14)
ONOFF_CHECK:   li t1, 0x4000
          and t0, s2, t1
          beqz t0, POS_CHECK
          call LCD_NODISPLAY
          j LOOP


          #Check for ROW/COL change (Switch 13)
POS_CHECK:     call LCD_DISPLAY

          li t1, 0x2000
          and t0, s2, t1
          beqz t0, WRITE_CHAR

          #Switches 12-9 determin col switch 8 determines row
          li t1, 0x1E00
          and a1, s2, t1
          srli a1, a1, 9
          li t1, 0x100
          and a0, s2, t1
          srli a0, a0, 8
          call LCD_ROWCOL
          j LOOP

          #Check lower 8 switches for utf 8 char to display
WRITE_CHAR:    andi a0, s2, 0xFF
          li t1, 0x100
          or a0, a0, t1  #Setup Char cmd
          call LCD_WRITE
          j LOOP
```

```
#Shouldn't reach this
END:      j END


ISR:        addi sp, sp, -8       # push t1, t2 to stack
               sw   t1, 4(sp)
               sw   t2, 0(sp)
               addi s1, x0, 1       # set interrupt flag

ISR_RET:  lw   t2, 0(sp)        # pop t1, t2 from stack
               lw   t1, 4(sp)
               addi sp, sp, 8
               mret

#################################################################
############
# LCD CLEAR -  Clears the LCD and resets cursor
#
#################################################################
############
LCD_CLEAR:     addi sp, sp, -4
          sw ra, 0(sp)

          li a0, 0x1          #LCD Clear display cmd

          call LCD_WRITE

          #Wait 2000us for command to finish
          addi t1, zero, 840
CLEAR_WAIT:    beqz t1, LCD_CLEAR_RET
          addi t1, t1, -1
          j CLEAR_WAIT

LCD_CLEAR_RET: lw ra, 0(sp)
          addi sp, sp, 4
          ret

#################################################################
############
# LCD HOME -   Returns cursor to zero
#
```

```
#################################################################
############
LCD_HOME: addi sp, sp, -4
          sw ra, 0(sp)

          li a0, 0x2          #LCD Return Home

          call LCD_WRITE

          #Wait 2000us for command to finish
          addi t1, zero, 840
HOME_WAIT:     beqz t1, LCD_HOME_RET
          addi t1, t1, -1
          j HOME_WAIT

LCD_HOME_RET:  lw ra, 0(sp)
          addi sp, sp, 4
          ret

#################################################################
############
# LCD DISPLAY OFF  -  Turns off the Display
#
#################################################################
############
LCD_NODISPLAY: addi sp, sp, -4
          sw ra, 0(sp)

          li a0, 0x8          #LCD Display OFF

          call LCD_WRITE

          lw ra, 0(sp)
          addi sp, sp, 4
          ret

#################################################################
############
# LCD DISPLAY ON  -  Turns On the Display
#
```

```
#################################################################
############
LCD_DISPLAY:   addi sp, sp, -4
          sw ra, 0(sp)

          li a0, 0xC          #LCD Display On

          call LCD_WRITE

          lw ra, 0(sp)
          addi sp, sp, 4
          ret

#################################################################
############
# LCD SET ROW/COL -      Move the cursor to the new Row/Column
# Inputs:
#    a0 - Row
#    a1 - Column
#
#################################################################
############
LCD_ROWCOL:    addi sp, sp, -4
          sw ra, 0(sp)

          mv t0, a0
          mv t1, a1

          li a0, 0x80         #SETDRAMADDR CMD
          beqz t0, ROW0       #Which row?
          addi t0, zero, 0x40
ROW0:          add t0, t0, t1
          or a0, a0, t0       #Setup row/col command
          call LCD_WRITE      #Send command to LCD

          lw ra, 0(sp)
          addi sp, sp, 4
          ret

#################################################################
############
```

```
# LCD Print -  Prints the given string to the LCD starting at
the current
#         row/column
# Inputs:
#    a0 - start of string in data segment
#    a1 - length of string
#
##################################################################
############
LCD_PRINT:     addi sp, sp, -16
          sw ra, 12(sp)
          sw s1, 8(sp)
          sw s2, 4(sp)
          sw s3, 0(sp)

          mv s1, a0      #Start of string in data segment
          mv s2, a1      #Number of iterations/len of string
          addi s3, zero, 0    #counter for print loop

          li t1, 2
          sw t1, 0x20(s0)

PRINT_CHAR:    beq s2, s3, LCD_PRINT_RET
          add t2, s1, s3      #Address for loading is start +
current char idx
          lbu a0, (t2)        #Load char from data
          li t0, 0x100        #Set RS Bit for char write
          or a0, a0, t0

          li t1, 3
          sw t1, 0x20(s0)

          call LCD_WRITE
          addi s3, s3, 1
          j PRINT_CHAR

LCD_PRINT_RET: lw s3, 0(sp)
          lw s2, 4(sp)
          lw s1, 8(sp)
          lw ra, 12(sp)
          addi sp, sp, 16
```

```
          ret

#################################################################
############
# LCD Write -  Waits for LCD_ready then writes a single CMD to
the LCD
# Inputs:
#    a0 - 9 bit lcd commmand
#
#################################################################
############
LCD_WRITE:     addi sp, sp, -4
          sw ra, 0(sp)

          #Wait for LCD_READY
          li t0, 0x11000140
LCD_WAIT: lbu t1, (t0)
          addi t2, zero, 1

          li t3, 7
          sw t3, 0x20(s0)

          beq t1, t2, LCD_READY
          j LCD_WAIT


          #Write cmd to LCD
LCD_READY:     li t0, 0x11000160

          li t2, 0xf
          sw t2, 0x20(s0)

          sw a0, (t0)
          li t1, 0x200
          or a0, a0, t1
          sw a0, (t0)

          #Clear send bit
          li t2, 0x1FF
          and a0, a0, t2
          sw a0, (t0)
```

```
          #Wait min 100us for cmd to finish
          addi t1, zero, 42
CMD_WAIT: beqz t1, LCD_WRITE_RET
          addi t1, t1, -1
          j CMD_WAIT

LCD_WRITE_RET: lw ra, (sp)
          addi sp, sp, 4
          ret
```

LiquidCrystal.sv

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////////////////
// Company:
// Engineer: Ryken Thompson
//
// Create Date: 12/07/2023 01:57:32 PM
// Design Name:
// Module Name: LiquidCrystal
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////


module LiquidCrystal(
    input CLK,
    input RST,
    input [9:0] lcd_instr,
    output logic lcd_ready = 0,
    output logic rs,
    output logic e,
    output logic [7:0] data
    );

    localparam freq = 50; //Freq in MHz

    logic [31:0] clk_count = 0;
    logic count_en = 0;
    logic count_rst = 0;
```

```
    enum {POWER_ON, INIT, IDLE, SEND} PS, NS;

    //assign rs = lcd_instr[8];
    //  assign data = lcd_instr[7:0];

    always_ff @(posedge CLK) begin
        if (RST) begin
            PS <= POWER_ON;
        end
        else PS <= NS;
    end

    always_comb begin
        data = 8'b00000000;
        rs = 1'b0;
        e = 1'b0;
        count_en = 0;
        count_rst = 0;
        lcd_ready = 0;

        case (PS)

            //Wait for LCD to properly power up
            POWER_ON: begin
                if (clk_count < (50000 * freq)) begin //Wait
50ms for LCD to properly power on
                    count_en = 1;
                    NS = POWER_ON;
                end
                else begin
                    count_rst = 1;
                    data = 8'b00110000;
                    NS = INIT;
                end
            end

            //Begin LCD initialization sequence
            INIT: begin
                count_en = 1;
                if (clk_count < (10 * freq)) begin //Set the LCD
Function
```

```
                    data = 8'b00111100; //2 line mode, display
on
                    e = 1'b1;
                    NS = INIT;
                end
                else if (clk_count < (60*freq)) begin // Wait 50
us
                    data = 8'b00000000;
                    e = 1'b0;
                    NS = INIT;
                end
                else if (clk_count < (70*freq)) begin //Display
On, Cursor off, blink off
                    data = 8'b00001100;
                    e = 1'b1;
                    NS = INIT;
                end
                else if (clk_count < (120*freq)) begin //Wait 50
us
                    data = 8'b00000000;
                    e = 1'b0;
                    NS = INIT;
                end
                else if (clk_count < (130*freq)) begin //Display
Clear
                    data = 8'b00000001;
                    e = 1'b1;
                    NS = INIT;
                end
                else if (clk_count < (2130*freq)) begin //Wait 2
ms for proper clear
                    data = 8'b00000000;
                    e = 1'b0;
                    NS = INIT;
                end
                else if (clk_count < (2140*freq)) begin //Entry
mode set: increment mode, entire shift off
                    data = 8'b00000110;
                    e = 1'b1;
                    NS = INIT;
                end
```

```
            else if (clk_count < (2200*freq)) begin // Wait
60 us
                data = 8'b00000000;
                e = 1'b0;
                NS = INIT;
            end
            else begin // Init complete
                count_rst = 1;
                NS = IDLE;
            end
        end

        //Wait for send signal before sending enable
        IDLE: begin
            lcd_ready = 1;
            data = lcd_instr[7:0];
            rs = lcd_instr[8];
            if (lcd_instr[9]) begin
                NS = SEND;
            end
            else begin
                NS = IDLE;
            end
        end
        SEND: begin
            lcd_ready = 0;
            data = lcd_instr[7:0];
            rs = lcd_instr[8];
            if (clk_count < (50 * freq)) begin
                count_en = 1;
                NS = SEND;
                if (clk_count < freq) begin
                    e = 1'b0;
                end
                else if (clk_count < (14*freq)) begin
                    e = 1'b1;
                end
                else if (clk_count < (27*freq)) begin
                    e = 1'b0;
                end
            end
```

```
                else begin
                    count_rst = 1;
                    NS = IDLE;
                end
            end
            default: begin
                NS = POWER_ON;
            end
        endcase
    end


    always_ff @(posedge CLK) begin
        if (count_rst) begin
            clk_count = 0;
        end
        else if (count_en) begin
            clk_count = clk_count + 1;
        end
    end

endmodule
```