**EE084IU**

**Micro-processing Systems Lab**

**Lab 2**

**LCD interface using C and Assembly Programming Language**

**Student name: Đỗ Minh Duy**

**Student ID: ITITSB22029**

**Class: Micro-processingSystems Laboratory\_S1\_2024-25-Group01**

**Date: 13/11/2024**

# **OBJECTIVES**

# Understand Microcontroller Basics:

# Familiarize students with the architecture and functionalities of the AVR ATmega32 microcontroller, including its I/O ports, timers, and interrupts, to prepare them for practical applications.

# Learn LCD Operation and Interfacing:

# Teach students the principles of how LCDs operate, including command and data modes, and guide them through the process of interfacing an LCD with the ATmega32 using appropriate wiring and programming techniques.

# Develop Programming Skills:

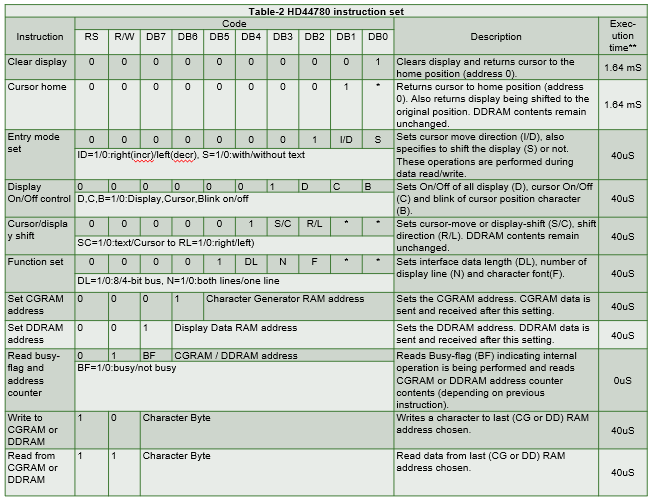
# Enhance students' programming abilities in C (or assembly language) by having them write and debug code to control the LCD display, enabling them to display characters and messages based on user inputs or sensor data.

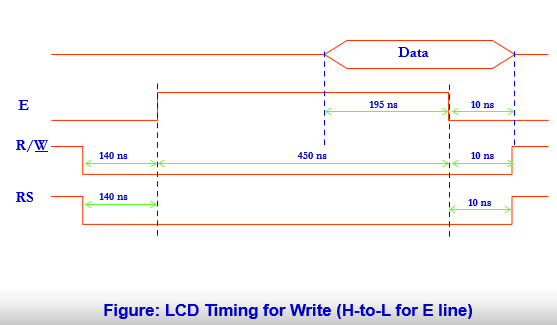
# **EQUIPMENTS/SOFTWARE**

* Computer, MPS kit
* Atmel Studio, Simulation software
* AVR Instruction set manual – PDF file.

# **THEORY REVIEW**

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLE-1 44780 Based LCD Pin out** | | | |
| PIN | SYMBOL | I/O | DESCRIPTION |
| 1 | VSS | - | Power supply (GND) |
| 2 | VCC | - | Power supply (VDD) (+5V) |
| 3 | VEE | - | Contrast Settings (0~2V) |
| 4 | RS | I | 0 = Select command reg. 1 = Select data reg. of LCD |
| 5 | R/W | I | 0 = Write to LCD  1 = Read from LCD |
| 6 | E | I | The Enable (E) line allows access to the display through R/W and RS lines. 0 = Access to LCD disabled, 1 = Access to LCD enabled |
| 7 | DB0 | I/O | Data bit line 0 (LSB) |
| 8 | DB1 | I/O | Data bit line 1 |
| 9 | DB2 | I/O | Data bit line 2 |
| 10 | DB3 | I/O | Data bit line 3 |
| 11 | DB4 | I/O | Data bit line 4 |
| 12 | DB5 | I/O | Data bit line 5 |
| 13 | DB6 | I/O | Data bit line 6 |
| 14 | DB7 | I/O | Data bit line 7 (MSB) |

****

****

**LAB PROCEDURE**

**Exercise 1**

Create a project to interface the ATMEGA32 with LCD 8-bit data interface. Any LCD pin can be connected to port PORTC and PORTBD as shown in Figure 1. Write a C program to declare a string

“ Your Name” in the first line and the display the string “study at School of Computer Engineering”, then display in the second line this string in LCD with 16 characters at a time, delay 1 second then scroll to next 16 characters, repeat until the end of the string. Run this program in Proteus then Implement this circuit in practical AVR Experiment KIT.

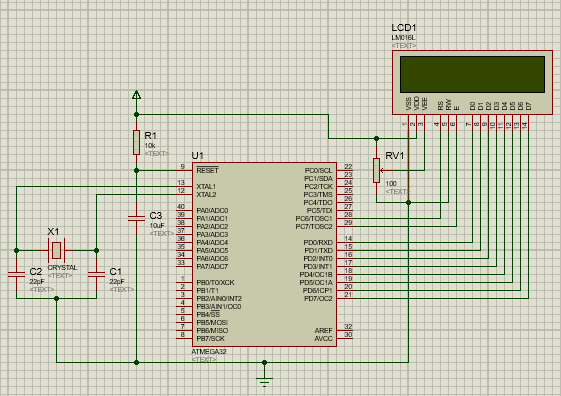


Figure 1. ATMEGA32 Interface to LCD in 8 bit mode

**Exercise 2**

Create a project to interface the ATMEGA32 with LCD 4-bit data interface. Any LCD pin can be connected to port PORTD as shown in Figure 2. Write a C program to declare a string “School of Computer Engineering International University”, then display this string in LCD with 16 characters at a time, delay 1 second then scroll to next 16 characters, repeat until the end of the string. Run this program in Proteus then Implement this circuit in practical AVR Experiment KIT.

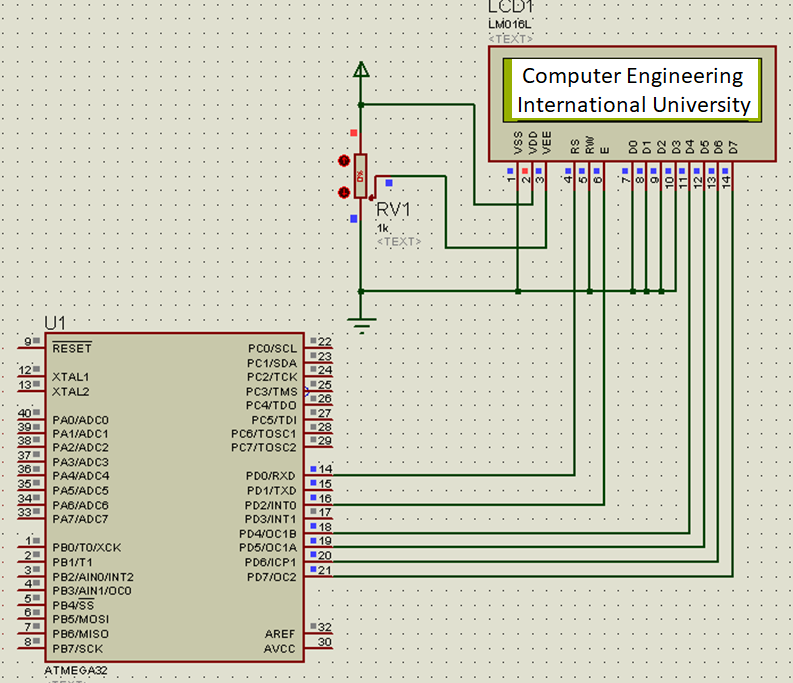
****

Figure 2. ATMEGA32 Interface to LCD in 4 bit mode

**Exercise 3**

Create a project to interface the ATMEGA32 with LCD 4-bit data interface. Any LCD pin can be connected to port PORTD as shown in Figure 3. Write a C program to to display the “Digital clock” string in first line and the current clock data in second line of LCD with format “hh:mm:ss’ using delay function \_delay\_ms() .Run this program in Proteus then Implement this circuit in practical AVR Experiment KIT.

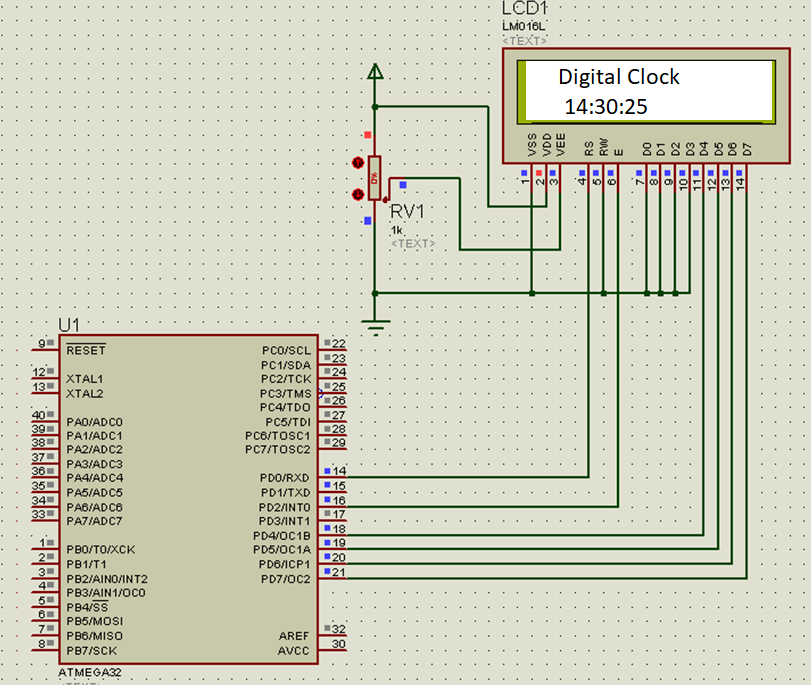


Figure 3. ATMEGA32 Interface to LCD in 4 bit mode

**Exercise 4**

Create a project to interface the ATMEGA32 with LCD 8-bit data interface. Any LCD pin can be connected to port PORTA and PORTB as shown in Figure 5. Write an AVR Assembly program to display the string “ Your Name” in first line, and the string “International University” in the second line. Run this program in Proteus and in the MPS Experiment KIT.

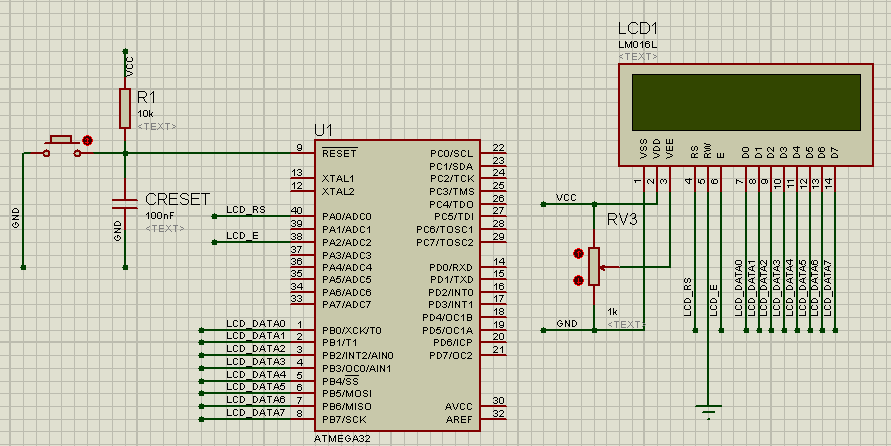


Figure 5. ATMEGA32 Interface to LCD in 8 bit mode

**Exercise 5**

Create a project to interface the ATMEGA32 with LCD 8-bit data interface. Any LCD pin can be connected to port PORTA and PORTB as shown in Figure 5. Write an AVR Assembly program to display the string “ Your Name” in first line, and the string “International University” in the second line. Run this program in Proteus and in the MPS Experiment KIT.

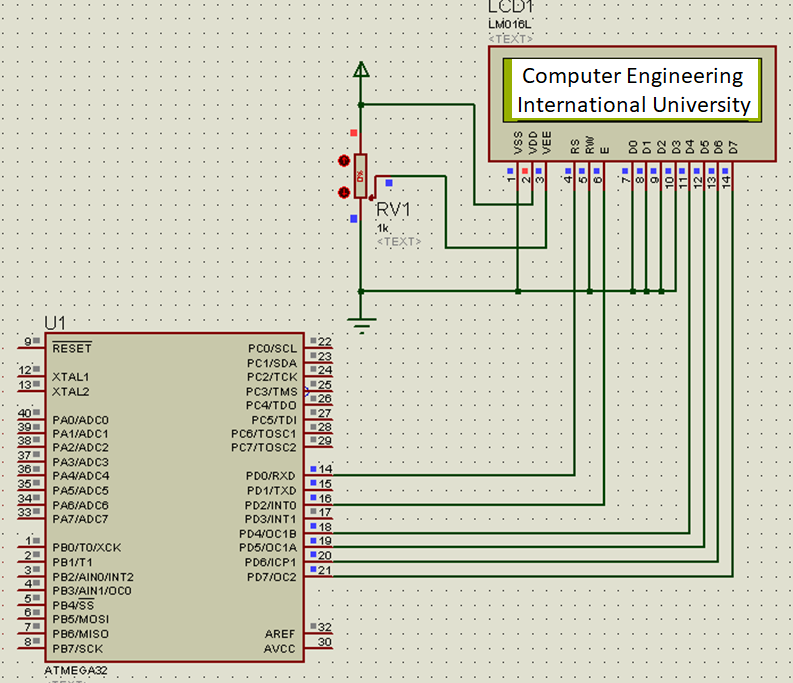
****

Figure 6. ATMEGA32 Interface to LCD in 4 bit mode

Lab 2 : Report

Draw the flow chart, Analyze the function of the code, create an AVR Studio Project to Compile this Code, Design an appropriate Schematic to simulate and test this code.

For Exercise 1 and 2 run the code in Experiment KIT, show the evidences of Practice figures.

Submit all Lab experiment evidences in the report.

**Exercise 1**

Flowchart:

+-------------------------------+

| Start Program |

+-------------------------------+

|

v

+-------------------------------+

| Initialize Ports (DDRC, DDRD)| <- Set DDRC and DDRD to output for control and data lines

+-------------------------------+

|

v

+-------------------------------+

| Initialize LCD | <- Call LCD\_Initialize() to configure the LCD

+-------------------------------+

|

v

+-------------------------------+

| Display static text (Hello) | <- Display the static message "Do Minh Duy" on the LCD

+-------------------------------+

|

v

+-------------------------------+

| Start Infinite Loop | <- Loop forever to display scrolling text

+-------------------------------+

|

v

+-------------------------------+

| Call LCD\_Scroll function | <- Scroll the string "study at School of Computer Engineer"

+-------------------------------+

|

v

+-------------------------------+

| Scroll the message | <- Move characters from left to right with spaces

+-------------------------------+

|

v

+-------------------------------+

| Delay and repeat | <- Delay for smooth scrolling and repeat until message finished

+-------------------------------+

|

v

+-------------------------------+

| End of Program | <- Program will keep running indefinitely

+-------------------------------+

Schematic:

+-------------------------+

| AVR Microcontroller |

| |

| PORTC (Control Pins) |-------------> RS (pin 0), RW (pin 1), E (pin 2)

| PORTD (Data Pins) |-------------> D0-D7 (pins 0-7)

| |

+-------------------------+

+-------------------------+

| 16x2 LCD Display |

| |

| RS (Register Select) |<--------------> PC0

| RW (Read/Write) |<--------------> PC1

| E (Enable) |<--------------> PC2

| D0-D7 (Data Pins) |<--------------> PD0-PD7

| VSS (Ground) |<--------------> GND

| VDD (Power) |<--------------> VCC

| V0 (Contrast) |<--------------> Potentiometer

+-------------------------+

Function:

**Analysis of the Code Functionality**

The code is designed to drive an LCD display using an AVR microcontroller. The core functionality is as follows:

1. **Hardware Setup:**
   * **PORTC** is used to control the LCD's control pins (RS, RW, and E).
   * **PORTD** is used to send the data (characters) to the LCD.
2. **LCD Commands:**
   * The LCD commands are sent via the LCD\_Command() function. Each command controls the LCD's behavior (e.g., clearing the screen, setting the cursor position, turning the display on/off, etc.).
3. **Displaying Text:**
   * **LCD\_Show()** is used to display individual characters on the LCD. This is used both for static text (like "Do Minh Duy") and for the scrolling text.
4. **Scrolling Text:**
   * The LCD\_Scroll() function handles the scrolling text. It scrolls the string "study at School of Computer Engineer" by shifting each character to the left with spaces filled in at the right. The scrolling continues indefinitely.
5. **Delay Function:**
   * A delay() function is used throughout the program to provide timing for scrolling and pulse timing for the E pin.

#include <avr/io.h>

#include <util/delay.h>

#define RS 0 // bit 0 of Port

#define RW 1 // bit 1 of Port

#define E 2 // bit 2 of Port

#define DATA\_BUS PORTD

#define DATA\_DDR DDRD

#define CTRL\_BUS PORTC

#define CTRL\_DDR DDRC

void delay(unsigned int z) {

unsigned int x;

for(x = 0; x < z; x++)

\_delay\_ms(10);

}

int ready() {

delay(10);

return 1;

}

void LCD\_Pulse\_E(int t) {

CTRL\_BUS |= 0b00000100; // E = 1;

delay(t);

CTRL\_BUS &= 0b11111011; // E = 0;

delay(t);

}

int LCD\_Command(unsigned char COMMAND) {

ready();

DATA\_BUS = COMMAND;

CTRL\_BUS = 0b11111000; // RS=0, RW=0

LCD\_Pulse\_E(1);

return 1;

}

int LCD\_Show(unsigned char CHARACTER) {

ready();

DATA\_BUS = CHARACTER;

CTRL\_BUS = 0b11111001; // RS=1, RW=0

LCD\_Pulse\_E(1);

return 1;

}

int LCD\_Initialize() {

LCD\_Command(0x38); // 8 data lines, two lines, Font 5x7

LCD\_Command(0x0F); // Display ON, Cursor ON, Cursor Blinking ON

LCD\_Command(0x01); // Clear display and return cursor to home position

LCD\_Command(0x06); // Cursor moves right

LCD\_Command(0x80); // Cursor at Line 1, Position 0

return 1;

}

void LCD\_Scroll(char\* str) {

int len = 0;

while (str[len] != '\0') len++; // Calculate the length of the string

for (int i = 0; i <= len; i++) {

LCD\_Command(0xC0); // Move cursor to Line 2

for (int j = 0; j < len; j++) {

if (j + i < len) {

LCD\_Show(str[j + i]); // Display the character

} else {

LCD\_Show(' '); // Fill with spaces

}

}

delay(500); // Delay for scroll speed

}

}

int main() {

DATA\_DDR = 0xFF;

CTRL\_DDR = 0xFF;

CTRL\_BUS = 0;

DATA\_BUS = 0;

LCD\_Initialize();

LCD\_Show('D'); // D

LCD\_Show('o'); // o

LCD\_Show(' '); // (space)

LCD\_Show('M'); // M

LCD\_Show('i'); // i

LCD\_Show('n'); // n

LCD\_Show('h'); // h

LCD\_Show(' '); // (space)

LCD\_Show('D'); // D

LCD\_Show('u'); // u

LCD\_Show('y'); // y

char message[] = "study at School of Computer Engineer";

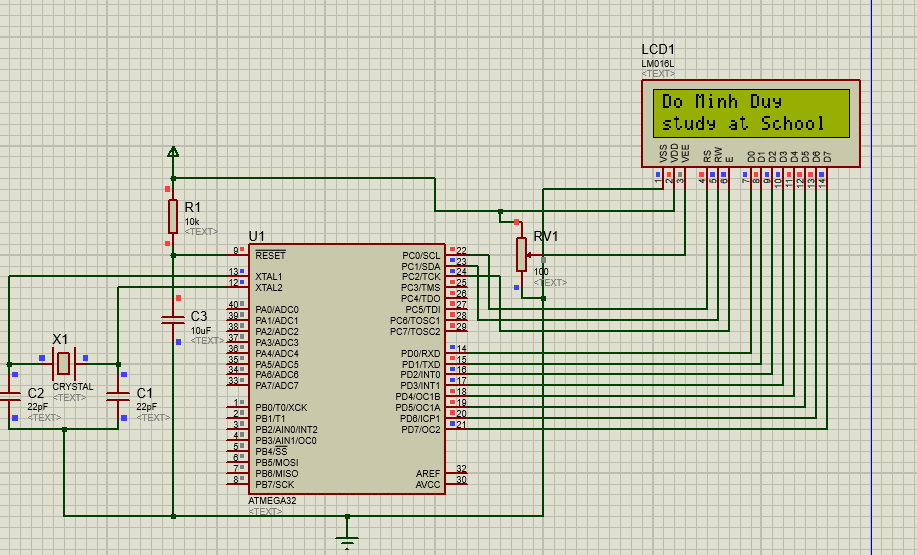
for (;;) {

LCD\_Scroll(message);

}

return 0;

}



**Exercise 2**

Flowchart:

+-----------------------------------------------+

| Start Program |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Initialize LCD and Ports (DDRx, DDRy) |

| Set up LCD pins as output |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| LCD Initialization (send command) |

| - Set LCD to 4-bit mode |

| - Set the cursor behavior and display mode|

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Main Loop: Display Message |

| - Clear LCD screen |

| - Display "Computer Engineering" |

| - Display "International University" |

| - Delay between each display cycle |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Continue repeating the message display |

| in an infinite loop |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| End Program |

| (Infinite Loop) |

+-----------------------------------------------+

Schematic:

+-------------------------+

| AVR Microcontroller |

| |

| PORTC (Control Pins) |--------------------> RS (PC0), RW (PC1), E (PC2)

| PORTD (Data Pins) |--------------------> D0 (PD0), D1 (PD1), ..., D7 (PD7)

| |

+-------------------------+

+-------------------------+

| 16x2 LCD |

| |

| RS (Register Select) |<------------------> PC0

| RW (Read/Write) |<------------------> PC1

| E (Enable) |<------------------> PC2

| D0-D7 (Data Pins) |<------------------> PD0-PD7

| VSS (Ground) |<------------------> GND

| VDD (Power) |<------------------> VCC

| V0 (Contrast) |<------------------> Potentiometer (adjustable)

| A (Anode) |<------------------> VCC

| K (Cathode) |<------------------> GND

+-------------------------+

Function:

**1. Port Initialization**

* The program configures PORTD as the data bus and PORTC as the control bus, setting them to output mode. This is done by configuring the direction registers DATA\_DDR and CTRL\_DDR.

**2. LCD Initialization (LCD\_init)**

* The LCD is first initialized in 4-bit mode, and several LCD commands are sent to configure the display settings:
  + 0x30: Set the LCD to 4-bit mode (sending the upper nibble first).
  + 0x20: Switch to 4-bit mode.
  + 0x28: Set 2-line mode, 5x7 character font.
  + 0x0C: Display on, cursor off.
  + 0x80: Move the cursor to the beginning of the first line.

**3. LCD Command and Data Display**

* LCD\_Command() sends commands to the LCD by writing to the LCD\_DATA bus and controlling the LCD\_E, LCD\_RS, and LCD\_RW pins.
* LCD\_Show() sends data (characters) to the LCD. It first sends the upper 4 bits, pulses the E pin (Enable), and then sends the lower 4 bits.
* LCD\_String() iterates through a string of characters and sends each one to the LCD using LCD\_Show().

**4. Main Loop**

* The main program enters an infinite loop where it:
  + Clears the display (LCD\_Command(0x01)).
  + Displays the first message on the first line (LCD\_Command(0x80)).
  + Displays the second message on the second line (LCD\_Command(0xC0)).
  + Waits for a second before clearing the screen and starting the process again.

**5. Delays**

* The delay1s() function creates a delay of 1 second using a simple loop that calls \_delay\_ms(10) 100 times.

#define F\_CPU 16000000UL // XTAL = 16MHZ = 8000000Hz

#include <avr/io.h>

#include <util/delay.h>

#define LCD\_DATA PORTD // port connected to LCD data pins

#define DATA\_DDR DDRD // direction register for data pins

#define LCD\_CTRL PORTD // port connected to LCD control pins

#define CTRL\_DDR DDRD // direction register for control pins

#define LCD\_RS 0 // define MCU pin connected to LCD RS

#define LCD\_RW 1 // define MCU pin connected to LCD R/W

#define LCD\_E 2 // define MCU pin connected to LCD E

#define LCD\_D0 0 // define MCU pin connected to LCD D0

#define LCD\_D1 1 // define MCU pin connected to LCD D1

#define LCD\_D2 2 // define MCU pin connected to LCD D1

#define LCD\_D3 3 // define MCU pin connected to LCD D2

#define LCD\_D4 4 // define MCU pin connected to LCD D3

#define LCD\_D5 5 // define MCU pin connected to LCD D4

#define LCD\_D6 6 // define MCU pin connected to LCD D5

#define LCD\_D7 7 // define MCU pin connected to LCD D6

const unsigned char message1[] ="Computer Engineering ";

const unsigned char message2[] ="International University";

void LCD\_Command(unsigned char cmd){ //Sends Command to LCD

//4 bit part

LCD\_DATA=(cmd&0b11110000); // send upper 4-bits

LCD\_CTRL|=1<<LCD\_E; // E=1 ,RS=0, RW=0

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0 ,RS=0, RW=0

\_delay\_ms(1); // keep E=0 for some time

LCD\_DATA=((cmd&0b00001111)<<4); // send lover 4-bits

LCD\_CTRL|=1<<LCD\_E; // E=1,RS=0,RW=0

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0,RS=0,RW=0

\_delay\_ms(1); // keep E=0 for some time

}

void delay1s(void){ //delay 1s

unsigned char i;

for(i=0;i<100;i++){

\_delay\_ms(10);

}

}

void LCD\_Show(uint8\_t ch)

{ //Sends Char to LCD

LCD\_DATA=(ch&0b11110000); // send upper 4-bits

LCD\_CTRL|=(1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~((1<<LCD\_E)); // E=0

\_delay\_ms(1); // keep E=0 for some time

LCD\_DATA=((ch&0b00001111)<<4); // send lower 4-bits

LCD\_CTRL|= (1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

}

void LCD\_init(void){ //Initializes LCD

\_delay\_ms(15);

LCD\_DATA=0x00; // data = 0

LCD\_CTRL=0x00; // RS = RW = E = 0

DATA\_DDR|=1<<LCD\_D7|1<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;

// Set bits 4 to 7 as output pins for data out

CTRL\_DDR|=1<<LCD\_E|1<<LCD\_RW|1<<LCD\_RS;

// Set bit 0 to 2 as output pins

//---------one------ // DATA = 0x30 ;

LCD\_DATA = 0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4; //4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|= 1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

// E=0;

LCD\_CTRL&=~(1<<LCD\_E);

\_delay\_ms(1); // keep E=0 for some time

//-----------two----------- // DATA = 0x30 ;

LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;

//4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

//-------three-------------

// DATA = 0x20 ;

LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|0<<LCD\_D4; //4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

//--------4 bit--dual line---------------

LCD\_Command(0b00101000); // 0x28

//-----increment address, invisible cursor shift------

LCD\_Command(0b00001100); // 0x0C

LCD\_Command(0b10000000); // 0x80

}

void LCD\_String (char \*str)

{

int i;

for(i=0;str[i]!=0;i++) /\* send each char of string till the NULL \*/

{

LCD\_Show(str[i]); /\* call LCD data write \*/

}

}

int main(void){

unsigned char i;

LCD\_init();//init LCD bit, dual line, cursor right

LCD\_Command(0x01); //clears LCD, Cursor at Home

while(1){ //loop for ever

delay1s(); delay1s();

LCD\_Command(0x80); // cursor at upper line most left

LCD\_String (message1) ;

LCD\_Command(0xC0); // cursor at lower line most left

LCD\_String (message2);

delay1s();

delay1s();

LCD\_Command(0x01); //clears LCD, Cursor at Home

}

return 0;

}

A computer circuit board with many wires

Description automatically generated

**Exercise 3**

Flowchart:

+-----------------------------------------------+

| Start Program |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Initialize LCD and Ports (DDRx, DDRy) |

| Set up LCD pins as output |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| LCD Initialization (send command) |

| - Set LCD to 4-bit mode |

| - Set display settings (cursor, etc.) |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Display "Digital Clock" on LCD |

| - Send the message to the top line |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| Main Loop: Update Clock |

| - Format time (hour:minute:second) |

| - Display time on second line of LCD |

| - Delay 1 second |

| - Loop back to update time again |

+-----------------------------------------------+

|

v

+-----------------------------------------------+

| End Program |

| (Infinite Loop) |

+-----------------------------------------------+

Schematic:

+-------------------------+

| AVR Microcontroller |

| |

| PORTC (Control Pins) |--------------------> RS (PC0), RW (PC1), E (PC2)

| PORTD (Data Pins) |--------------------> D0 (PD0), D1 (PD1), ..., D7 (PD7)

| |

+-------------------------+

+-------------------------+

| 16x2 LCD |

| |

| RS (Register Select) |<------------------> PC0

| RW (Read/Write) |<------------------> PC1

| E (Enable) |<------------------> PC2

| D0-D7 (Data Pins) |<------------------> PD0-PD7

| VSS (Ground) |<------------------> GND

| VDD (Power) |<------------------> VCC (5V)

| V0 (Contrast) |<------------------> Potentiometer (adjustable)

| A (Anode) |<------------------> VCC (5V)

| K (Cathode) |<------------------> GND

+-------------------------+

Function:

**Port Initialization (LCD\_init):**

* The program starts by setting up the LCD's control and data pins as output.
* It then sends initialization commands to the LCD in 4-bit mode. The sequence is essential for the LCD to enter the correct mode for data display.

**Sending Commands and Data to the LCD:**

* **LCD\_Command(cmd)**: This function sends a command to the LCD. It uses 4-bit data mode, first sending the upper nibble of the command, then the lower nibble. It also controls the E, RS, and RW pins to ensure correct timing and behavior of the LCD.
* **LCD\_Show(ch)**: This function sends a character to the LCD in a similar way to LCD\_Command, but with the RS pin set to 1, indicating that the data being sent is character data, not a command.
* **LCD\_String(str)**: This function sends a string of characters to the LCD by calling LCD\_Show() for each character in the string until the null terminator ('\0') is reached.

**Displaying the Time:**

* **sprintf(message2, "%0.2d:%0.2d:%0.2d", hour, minute, second)**: This formats the current time into a string in the format hh:mm:ss. The %0.2d ensures that each time component (hour, minute, second) is always two digits, padding with zeros if necessary (e.g., 08:05:03).
* The formatted time string is then displayed on the second line of the LCD using LCD\_String(message2).

**Time Variables:**

* The program uses three integer variables (hour, minute, second) to keep track of the time. Currently, they are set to 16:50:00 as an initial value.

**Main Loop:**

* The program enters an infinite loop where it continuously updates and displays the time. The LCD is updated every 1 second (delay1s()).

**Delays:**

* The delay1s() function creates a 1-second delay using \_delay\_ms(10) in a loop that repeats 100 times. This is not precise but provides enough delay for the digital clock to be visible to the user.

#define F\_CPU 16000000UL // XTAL = 16MHZ = 8000000Hz

#include <avr/io.h>

#include <util/delay.h>

#define LCD\_DATA PORTD // port connected to LCD data pins

#define DATA\_DDR DDRD // direction register for data pins

#define LCD\_CTRL PORTD // port connected to LCD control pins

#define CTRL\_DDR DDRD // direction register for control pins

#define LCD\_RS 0 // define MCU pin connected to LCD RS

#define LCD\_RW 1 // define MCU pin connected to LCD R/W

#define LCD\_E 2 // define MCU pin connected to LCD E

#define LCD\_D0 0 // define MCU pin connected to LCD D0

#define LCD\_D1 1 // define MCU pin connected to LCD D1

#define LCD\_D2 2 // define MCU pin connected to LCD D1

#define LCD\_D3 3 // define MCU pin connected to LCD D2

#define LCD\_D4 4 // define MCU pin connected to LCD D3

#define LCD\_D5 5 // define MCU pin connected to LCD D4

#define LCD\_D6 6 // define MCU pin connected to LCD D5

#define LCD\_D7 7 // define MCU pin connected to LCD D6

const unsigned char message1[] ="Didital Clock";

const unsigned char message2[] ="00:00:00 ";

void LCD\_Command(unsigned char cmd){ //Sends Command to LCD

//4 bit part

LCD\_DATA=(cmd&0b11110000); // send upper 4-bits

LCD\_CTRL|=1<<LCD\_E; // E=1 ,RS=0, RW=0

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0 ,RS=0, RW=0

\_delay\_ms(1); // keep E=0 for some time

LCD\_DATA=((cmd&0b00001111)<<4); // send lover 4-bits

LCD\_CTRL|=1<<LCD\_E; // E=1,RS=0,RW=0

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0,RS=0,RW=0

\_delay\_ms(1); // keep E=0 for some time

}

void delay1s(void){ //delay 1s

unsigned char i;

for(i=0;i<100;i++){

\_delay\_ms(10);

}

}

void LCD\_Show(uint8\_t ch)

{ //Sends Char to LCD

LCD\_DATA=(ch&0b11110000); // send upper 4-bits

LCD\_CTRL|=(1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~((1<<LCD\_E)); // E=0

\_delay\_ms(1); // keep E=0 for some time

LCD\_DATA=((ch&0b00001111)<<4); // send lower 4-bits

LCD\_CTRL|= (1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

}

void LCD\_init(void){ //Initializes LCD

\_delay\_ms(15);

LCD\_DATA=0x00; // data = 0

LCD\_CTRL=0x00; // RS = RW = E = 0

DATA\_DDR|=1<<LCD\_D7|1<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;

// Set bits 4 to 7 as output pins for data out

CTRL\_DDR|=1<<LCD\_E|1<<LCD\_RW|1<<LCD\_RS;

// Set bit 0 to 2 as output pins

//---------one------ // DATA = 0x30 ;

LCD\_DATA = 0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4; //4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|= 1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

// E=0;

LCD\_CTRL&=~(1<<LCD\_E);

\_delay\_ms(1); // keep E=0 for some time

//-----------two----------- // DATA = 0x30 ;

LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;

//4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

//-------three-------------

// DATA = 0x20 ;

LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|0<<LCD\_D4; //4 bit mode

// E=1, RW=0, RS=0 for command mode

LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;

\_delay\_ms(1); // keep E=1 for some time

LCD\_CTRL&=~(1<<LCD\_E); // E=0

\_delay\_ms(1); // keep E=0 for some time

//--------4 bit--dual line---------------

LCD\_Command(0b00101000); // 0x28

//-----increment address, invisible cursor shift------

LCD\_Command(0b00001100); // 0x0C

LCD\_Command(0b10000000); // 0x80

}

void LCD\_String (char \*str)

{

int i;

for(i=0;str[i]!=0;i++) /\* send each char of string till the NULL \*/

{

LCD\_Show(str[i]); /\* call LCD data write \*/

}

}

int hour=16,minute=50,second=0;

int main(void){

unsigned char i;

LCD\_init();//init LCD bit, dual line, cursor right

LCD\_Command(0x01); //clears LCD, Cursor at Home

delay1s(); delay1s();

LCD\_Command(0x80); // cursor at upper line most left

LCD\_String (message1) ;

while(1)

{ //loop for ever

sprintf(message2, "%0.2d:%0.2d:%0.2d",hour,minute,second);

LCD\_Command(0xC0); // cursor at lower line most left

LCD\_String (message2);

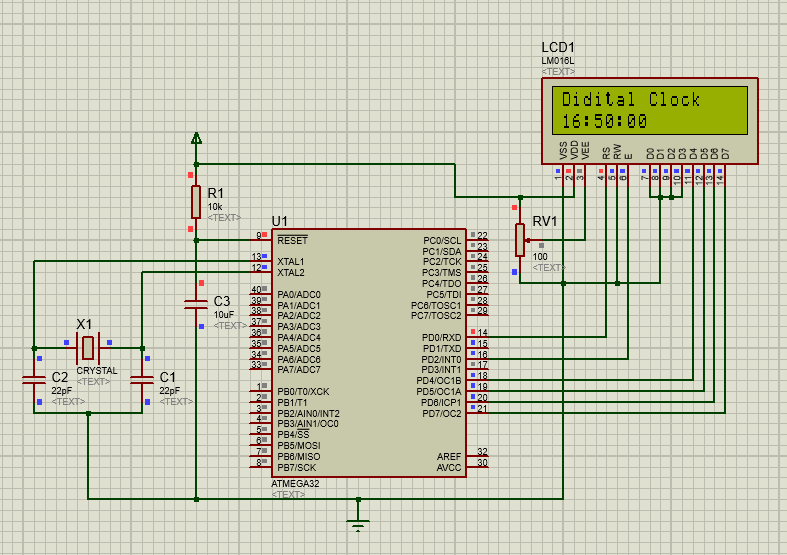
delay1s();

delay1s();

}

return 0;

}



**Exercise 4**

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; LCD-AVR-8bit.asm - Use an HD44780U based LCD with an Atmel ATmega processor

;

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; File: LCD-AVR-8bit.asm

; Date: March 9, 2022

;

; Target: ATmega32

; Assembler: Atmel AvrAssembler2 (AVR Studio 6)

;

; Summary: 8-bit data interface, busy flag not implemented.

; Any LCD pin can be connected to any available I/O port.

; Includes a simple write string routine.

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program Notes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; This program uses an 8-bit data interface but does not use the

; busy flag to determine when the LCD controller is ready. The

; LCD RW line (pin 5) is not connected to the uP and it must be

; connected to GND for the program to function.

;

; All time delays are longer than those specified in most datasheets

; in order to accommodate slower than normal LCD modules. This

; requirement is well documented but almost always ignored. The

; information is in a note at the bottom of the right hand

; (Execution Time) column of the instruction set.

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; The eight data lines as well as the two control lines may be

; implemented on any available I/O pin of any port. These are

; the connections used for this program:

;

; ----------- ----------

; | ATmega32 | | LCD |

; | | | |

; | PB7|---------------->|D7 |

; | PB6|---------------->|D6 |

; | PB5|---------------->|D5 |

; | PB4|---------------->|D4 |

; | PB3|---------------->|D3 |

; | PB2|---------------->|D2 |

; | PB1|---------------->|D1 |

; | PB0|---------------->|D0 |

; | | | |

; | PA1|---------------->|E |

; | | GND --->|RW |

; | PA0|---------------->|RS |

; ----------- ----------

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "m32def.inc"

.equ fclk = 8000000 ; system clock frequency (for delays)

; register usage

.def temp = R16 ; temporary storage

; LCD interface (should agree with the diagram above)

; make sure that the LCD RW pin is connected to GND

.equ lcd\_D7\_port = PORTB ; lcd D7 connection

.equ lcd\_D7\_bit = PORTB7

.equ lcd\_D7\_ddr = DDRB

.equ lcd\_D6\_port = PORTB ; lcd D6 connection

.equ lcd\_D6\_bit = PORTB6

.equ lcd\_D6\_ddr = DDRB

.equ lcd\_D5\_port = PORTB ; lcd D5 connection

.equ lcd\_D5\_bit = PORTB5

.equ lcd\_D5\_ddr = DDRB

.equ lcd\_D4\_port = PORTB ; lcd D4 connection

.equ lcd\_D4\_bit = PORTB4

.equ lcd\_D4\_ddr = DDRB

.equ lcd\_D3\_port = PORTB ; lcd D3 connection

.equ lcd\_D3\_bit = PORTB3

.equ lcd\_D3\_ddr = DDRB

.equ lcd\_D2\_port = PORTB ; lcd D2 connection

.equ lcd\_D2\_bit = PORTB2

.equ lcd\_D2\_ddr = DDRB

.equ lcd\_D1\_port = PORTB ; lcd D1 connection

.equ lcd\_D1\_bit = PORTB1

.equ lcd\_D1\_ddr = DDRB

.equ lcd\_D0\_port = PORTB ; lcd D0 connection

.equ lcd\_D0\_bit = PORTB0

.equ lcd\_D0\_ddr = DDRB

.equ lcd\_E\_port = PORTA ; lcd Enable pin

.equ lcd\_E\_bit = PORTA2

.equ lcd\_E\_ddr = DDRA

.equ lcd\_RS\_port = PORTA ; lcd Register Select pin

.equ lcd\_RS\_bit = PORTA0

.equ lcd\_RS\_ddr = DDRA

; LCD module information

.equ lcd\_LineOne = 0x00 ; start of line 1

.equ lcd\_LineTwo = 0x40 ; start of line 2

;.equ lcd\_LineThree = 0x14 ; start of line 3 (20x4)

;.equ lcd\_lineFour = 0x54 ; start of line 4 (20x4)

;.equ lcd\_LineThree = 0x10 ; start of line 3 (16x4)

;.equ lcd\_lineFour = 0x50 ; start of line 4 (16x4)

; LCD instructions

.equ lcd\_Clear = 0b00000001 ; replace all characters with ASCII 'space'

.equ lcd\_Home = 0b00000010 ; return cursor to first position on first line

.equ lcd\_EntryMode = 0b00000110 ; shift cursor from left to right on read/write

.equ lcd\_DisplayOff = 0b00001000 ; turn display off

.equ lcd\_DisplayOn = 0b00001100 ; display on, cursor off, don't blink character

.equ lcd\_FunctionReset = 0b00110000 ; reset the LCD

.equ lcd\_FunctionSet8bit = 0b00111000 ; 8-bit data, 2-line display, 5 x 7 font

.equ lcd\_SetCursor = 0b10000000 ; set cursor position

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Reset Vector \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0000

jmp start ; jump over Interrupt Vectors, Program ID etc.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program ID \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x30

program\_author:

.db "Do Minh Duy",0

program\_version:

.db "International University",0,0

program\_date:

.db "March,2022",0

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Program Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

start:

; initialize the stack pointer to the highest RAM address

ldi temp,low(RAMEND)

out SPL,temp

ldi temp,high(RAMEND)

out SPH,temp

; configure the microprocessor pins for the data lines

sbi lcd\_D7\_ddr, lcd\_D7\_bit ; 8 data lines - output

sbi lcd\_D6\_ddr, lcd\_D6\_bit

sbi lcd\_D5\_ddr, lcd\_D5\_bit

sbi lcd\_D4\_ddr, lcd\_D4\_bit

sbi lcd\_D3\_ddr, lcd\_D3\_bit

sbi lcd\_D2\_ddr, lcd\_D2\_bit

sbi lcd\_D1\_ddr, lcd\_D1\_bit

sbi lcd\_D0\_ddr, lcd\_D0\_bit

; configure the microprocessor pins for the control lines

sbi lcd\_E\_ddr, lcd\_E\_bit ; E line - output

sbi lcd\_RS\_ddr, lcd\_RS\_bit ; RS line - output

; initialize the LCD controller as determined by the equates (LCD instructions)

call lcd\_init\_8d ; initialize the LCD display for an 8-bit interface

; display the first line of information

ldi ZH, high(program\_author) ; point to the information that is to be displayed

ldi ZL, low(program\_author)

ldi temp, lcd\_LineOne ; point to where the information should be displayed

call lcd\_write\_string\_8d

; display the second line of information

ldi ZH, high(program\_version) ; point to the information that is to be displayed

ldi ZL, low(program\_version)

ldi temp, lcd\_LineTwo ; point to where the information should be displayed

call lcd\_write\_string\_8d

; endless loop

here:

rjmp here

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Main Program Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; ============================== 8-bit LCD Subroutines ======================

; Name: lcd\_init\_8d

; Purpose: initialize the LCD module for a 8-bit data interface

; Entry: equates (LCD instructions) set up for the desired operation

; Exit: no parameters

; Notes: uses time delays instead of checking the busy flag

lcd\_init\_8d:

; Power-up delay

ldi temp, 100 ; initial 40 mSec delay

call delayTx1mS

; Reset the LCD controller.

ldi temp, lcd\_FunctionReset ; first part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 10 ; 4.1 mS delay (min)

call delayTx1mS

ldi temp, lcd\_FunctionReset ; second part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 200 ; 100 uS delay (min)

call delayTx1uS

ldi temp, lcd\_FunctionReset ; third part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 200 ; this delay is omitted in the data sheet

call delayTx1uS

; Function Set instruction

ldi temp, lcd\_FunctionSet8bit ; set mode, lines, and font

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; The next three instructions are specified in the data sheet as part of the initialization routine,

; so it is a good idea (but probably not necessary) to do them just as specified and then redo them

; later if the application requires a different configuration.

; Display On/Off Control instruction

ldi temp, lcd\_DisplayOff ; turn display OFF

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; Clear Display instruction

ldi temp, lcd\_Clear ; clear display RAM

call lcd\_write\_instruction\_8d

ldi temp, 4 ; 1.64 mS delay (min)

call delayTx1mS

; Entry Mode Set instruction

ldi temp, lcd\_EntryMode ; set desired shift characteristics

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; This is the end of the LCD controller initialization as specified in the data sheet, but the display

; has been left in the OFF condition. This is a good time to turn the display back ON.

; Display On/Off Control instruction

ldi temp, lcd\_DisplayOn ; turn the display ON

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_string\_8d

; Purpose: display a string of characters on the LCD

; Entry: ZH and ZL pointing to the start of the string

; (temp) contains the desired DDRAM address at which to start the display

; Exit: no parameters

; Notes: the string must end with a null (0)

; uses time delays instead of checking the busy flag

lcd\_write\_string\_8d:

; preserve registers

push ZH ; preserve pointer registers

push ZL

; fix up the pointers for use with the 'lpm' instruction

lsl ZL ; shift the pointer one bit left for the lpm instruction

rol ZH

; set up the initial DDRAM address

ori temp, lcd\_SetCursor ; convert the plain address to a set cursor instruction

call lcd\_write\_instruction\_8d ; set up the first DDRAM address

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; write the string of characters

lcd\_write\_string\_8d\_01:

lpm temp, Z+ ; get a character

cpi temp, 0 ; check for end of string

breq lcd\_write\_string\_8d\_02 ; done

; arrive here if this is a valid character

call lcd\_write\_character\_8d ; display the character

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

rjmp lcd\_write\_string\_8d\_01 ; not done, send another character

; arrive here when all characters in the message have been sent to the LCD module

lcd\_write\_string\_8d\_02:

pop ZL ; restore pointer registers

pop ZH

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_character\_8d

; Purpose: send a byte of information to the LCD data register

; Entry: (temp) contains the data byte

; Exit: no parameters

; Notes: does not deal with RW (busy flag is not implemented)

lcd\_write\_character\_8d:

sbi lcd\_RS\_port, lcd\_RS\_bit ; select the Data Register (RS high)

cbi lcd\_E\_port, lcd\_E\_bit ; make sure E is initially low

call lcd\_write\_8 ; write the data

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_instruction\_8d

; Purpose: send a byte of information to the LCD instruction register

; Entry: (temp) contains the data byte

; Exit: no parameters

; Notes: does not deal with RW (busy flag is not implemented)

lcd\_write\_instruction\_8d:

cbi lcd\_RS\_port, lcd\_RS\_bit ; select the Instruction Register (RS low)

cbi lcd\_E\_port, lcd\_E\_bit ; make sure E is initially low

call lcd\_write\_8 ; write the instruction

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_8

; Purpose: send a byte of information to the LCD module

; Entry: (temp) contains the data byte

; RS is configured for the desired LCD register

; E is low

; RW is low

; Exit: no parameters

; Notes: use either time delays or the busy flag

lcd\_write\_8:

; set up the data bits

sbi lcd\_D7\_port, lcd\_D7\_bit ; assume that the data bit is '1'

sbrs temp, 7 ; check the actual data value

cbi lcd\_D7\_port, lcd\_D7\_bit ; arrive here only if the data was actually '0'

sbi lcd\_D6\_port, lcd\_D6\_bit ; repeat for each data bit

sbrs temp, 6

cbi lcd\_D6\_port, lcd\_D6\_bit

sbi lcd\_D5\_port, lcd\_D5\_bit

sbrs temp, 5

cbi lcd\_D5\_port, lcd\_D5\_bit

sbi lcd\_D4\_port, lcd\_D4\_bit

sbrs temp, 4

cbi lcd\_D4\_port, lcd\_D4\_bit

sbi lcd\_D3\_port, lcd\_D3\_bit

sbrs temp, 3

cbi lcd\_D3\_port, lcd\_D3\_bit

sbi lcd\_D2\_port, lcd\_D2\_bit

sbrs temp, 2

cbi lcd\_D2\_port, lcd\_D2\_bit

sbi lcd\_D1\_port, lcd\_D1\_bit

sbrs temp, 1

cbi lcd\_D1\_port, lcd\_D1\_bit

sbi lcd\_D0\_port, lcd\_D0\_bit

sbrs temp, 0

cbi lcd\_D0\_port, lcd\_D0\_bit

; write the data

; 'Address set-up time' (40 nS)

sbi lcd\_E\_port, lcd\_E\_bit ; Enable pin high

call delay1uS ; implement 'Data set-up time' (80 nS) and 'Enable pulse width' (230 nS)

cbi lcd\_E\_port, lcd\_E\_bit ; Enable pin low

call delay1uS ; implement 'Data hold time' (10 nS) and 'Enable cycle time' (500 nS)

ret

; ============================== End of 8-bit LCD Subroutines ===============

; ============================== Time Delay Subroutines =====================

; Name: delayYx1mS

; Purpose: provide a delay of (YH:YL) x 1 mS

; Entry: (YH:YL) = delay data

; Exit: no parameters

; Notes: the 16-bit register provides for a delay of up to 65.535 Seconds

; requires delay1mS

delayYx1mS:

call delay1mS ; delay for 1 mS

dec YL ; update the the delay counter

brne delayYx1mS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delayTx1mS

; Purpose: provide a delay of (temp) x 1 mS

; Entry: (temp) = delay data

; Exit: no parameters

; Notes: the 8-bit register provides for a delay of up to 255 mS

; requires delay1mS

delayTx1mS:

call delay1mS ; delay for 1 mS

dec temp ; update the delay counter

brne delayTx1mS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delay1mS

; Purpose: provide a delay of 1 mS

; Entry: no parameters

; Exit: no parameters

; Notes: chews up fclk/1000 clock cycles (including the 'call')

delay1mS:

push YL ; [2] preserve registers

push YH ; [2]

ldi YL, low (((fclk/1000)-18)/4) ; [1] delay counter

ldi YH, high(((fclk/1000)-18)/4) ; [1]

delay1mS\_01:

dec YL ; [2] update the the delay counter

brne delay1mS\_01 ; [2] delay counter is not zero

; arrive here when delay counter is zero

pop YH ; [2] restore registers

pop YL ; [2]

ret ; [4]

; ---------------------------------------------------------------------------

; Name: delayTx1uS

; Purpose: provide a delay of (temp) x 1 uS with a 16 MHz clock frequency

; Entry: (temp) = delay data

; Exit: no parameters

; Notes: the 8-bit register provides for a delay of up to 255 uS

; requires delay1uS

delayTx1uS:

call delay1uS ; delay for 1 uS

dec temp ; decrement the delay counter

brne delayTx1uS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delay1uS

; Purpose: provide a delay of 1 uS with a 16 MHz clock frequency

; Entry: no parameters

; Exit: no parameters

; Notes: add another push/pop for 20 MHz clock frequency

delay1uS:

push temp ; [2] these instructions do nothing except consume clock cycles

pop temp ; [2]

push temp ; [2]

pop temp ; [2]

ret ; [4]

; ============================== End of Time Delay Subroutines ==============

Flowchart:

[Start]

|

V

[Initialize Stack Pointer]

|

V

[Set Up I/O Pins for LCD]

|

V

[Call lcd\_init\_8d] ---> [LCD Initialized?] --- Yes --> [Continue]

|

V

[Display First Line of Info]

|

V

[Display Second Line of Info]

|

V

[Endless Loop (Wait)]

|

V

[End]

Schematic:

+-----------------+

| |

+--------| ATmega32 |

| | |

| | PB0 (D0) ----|-----> D0 (LCD)

| | PB1 (D1) ----|-----> D1 (LCD)

| | PB2 (D2) ----|-----> D2 (LCD)

| | PB3 (D3) ----|-----> D3 (LCD)

| | PB4 (D4) ----|-----> D4 (LCD)

| | PB5 (D5) ----|-----> D5 (LCD)

| | PB6 (D6) ----|-----> D6 (LCD)

| | PB7 (D7) ----|-----> D7 (LCD)

| | PA0 (RS) ----|-----> RS (LCD)

| | PA1 (E) ----|-----> E (LCD)

| | GND ----|-----> RW (LCD) (connected to GND)

| | |

| | |

| +-----------------+

|

| +--------------+

| | |

+----------| HD44780U |

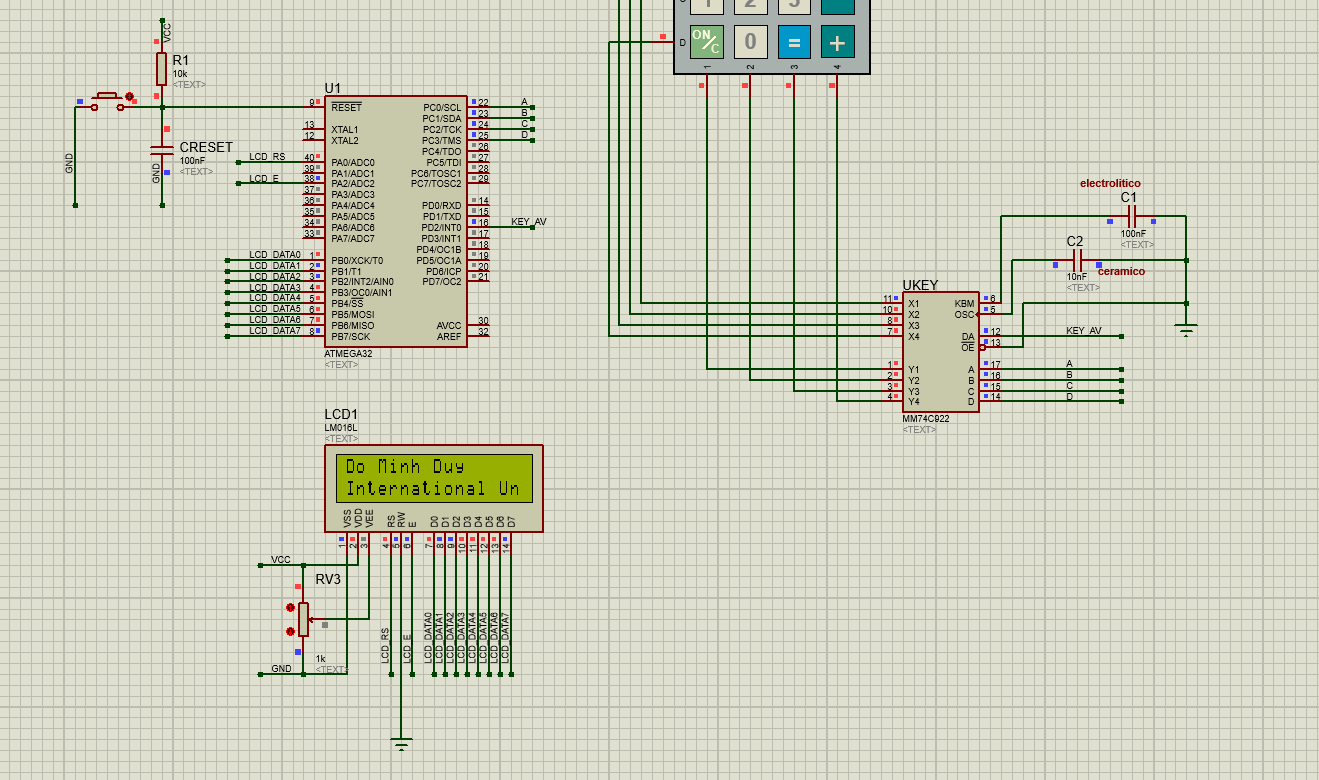
| LCD |

| |

| |

| |

+--------------+

****

**Exercise 5**

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; LCD-AVR-8bit.asm - Use an HD44780U based LCD with an Atmel ATmega processor

;

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; File: LCD-AVR-8bit.asm

; Date: March 9, 2022

;

; Target: ATmega32

; Assembler: Atmel AvrAssembler2 (AVR Studio 6)

;

; Summary: 8-bit data interface, busy flag not implemented.

; Any LCD pin can be connected to any available I/O port.

; Includes a simple write string routine.

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program Notes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; This program uses an 8-bit data interface but does not use the

; busy flag to determine when the LCD controller is ready. The

; LCD RW line (pin 5) is not connected to the uP and it must be

; connected to GND for the program to function.

;

; All time delays are longer than those specified in most datasheets

; in order to accommodate slower than normal LCD modules. This

; requirement is well documented but almost always ignored. The

; information is in a note at the bottom of the right hand

; (Execution Time) column of the instruction set.

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; The eight data lines as well as the two control lines may be

; implemented on any available I/O pin of any port. These are

; the connections used for this program:

;

; ----------- ----------

; | ATmega32 | | LCD |

; | | | |

; | PB7|---------------->|D7 |

; | PB6|---------------->|D6 |

; | PB5|---------------->|D5 |

; | PB4|---------------->|D4 |

; | PB3|---------------->|D3 |

; | PB2|---------------->|D2 |

; | PB1|---------------->|D1 |

; | PB0|---------------->|D0 |

; | | | |

; | PA1|---------------->|E |

; | | GND --->|RW |

; | PA0|---------------->|RS |

; ----------- ----------

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "m32def.inc"

.equ fclk = 8000000 ; system clock frequency (for delays)

; register usage

.def temp = R16 ; temporary storage

; LCD interface (should agree with the diagram above)

; make sure that the LCD RW pin is connected to GND

.equ lcd\_D7\_port = PORTD ; lcd D7 connection

.equ lcd\_D7\_bit = PORTD7

.equ lcd\_D7\_ddr = DDRD

.equ lcd\_D6\_port = PORTD ; lcd D6 connection

.equ lcd\_D6\_bit = PORTD6

.equ lcd\_D6\_ddr = DDRD

.equ lcd\_D5\_port = PORTD ; lcd D5 connection

.equ lcd\_D5\_bit = PORTD5

.equ lcd\_D5\_ddr = DDRD

.equ lcd\_D4\_port = PORTD ; lcd D4 connection

.equ lcd\_D4\_bit = PORTD4

.equ lcd\_D4\_ddr = DDRD

.equ lcd\_E\_port = PORTD ; lcd Enable pin

.equ lcd\_E\_bit = PORTD2

.equ lcd\_E\_ddr = DDRD

.equ lcd\_RS\_port = PORTD ; lcd Register Select pin

.equ lcd\_RS\_bit = PORTD0

.equ lcd\_RS\_ddr = DDRD

; LCD module information

.equ lcd\_LineOne = 0x00 ; start of line 1

.equ lcd\_LineTwo = 0x40 ; start of line 2

;.equ lcd\_LineThree = 0x14 ; start of line 3 (20x4)

;.equ lcd\_lineFour = 0x54 ; start of line 4 (20x4)

;.equ lcd\_LineThree = 0x10 ; start of line 3 (16x4)

;.equ lcd\_lineFour = 0x50 ; start of line 4 (16x4)

; LCD instructions

.equ lcd\_Clear = 0b00000001 ; replace all characters with ASCII 'space'

.equ lcd\_Home = 0b00000010 ; return cursor to first position on first line

.equ lcd\_EntryMode = 0b00000110 ; shift cursor from left to right on read/write

.equ lcd\_DisplayOff = 0b00001000 ; turn display off

.equ lcd\_DisplayOn = 0b00001100 ; display on, cursor off, don't blink character

.equ lcd\_FunctionReset = 0b00110000 ; reset the LCD

.equ lcd\_FunctionSet4bit = 0x02 ; 8-bit data, 2-line display, 5 x 7 font

.equ lcd\_CMD\_4BIT\_MODE = 0x28 ; = 0x30 ; 4-bit data, 2 lines, 5x7 matrix

.equ lcd\_SetCursor = 0b10000000 ; set cursor position

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Reset Vector \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x0000

jmp start ; jump over Interrupt Vectors, Program ID etc.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program ID \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org 0x30

program\_author:

.db "Do Minh Duy",0

program\_version:

.db "International University",0,0

program\_date:

.db "March,2022",0

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main Program Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

start:

; initialize the stack pointer to the highest RAM address

ldi temp,low(RAMEND)

out SPL,temp

ldi temp,high(RAMEND)

out SPH,temp

; configure the microprocessor pins for the data lines

LDI R16,0xFF

OUT DDRD,R16

; configure the microprocessor pins for the control lines

sbi lcd\_E\_ddr, lcd\_E\_bit ; E line - output

sbi lcd\_RS\_ddr, lcd\_RS\_bit ; RS line - output

; initialize the LCD controller as determined by the equates (LCD instructions)

call lcd\_init\_8d ; initialize the LCD display for an 8-bit interface

; display the first line of information

ldi ZH, high(program\_author) ; point to the information that is to be displayed

ldi ZL, low(program\_author)

ldi temp, lcd\_LineOne ; point to where the information should be displayed

call lcd\_write\_string\_8d

; display the second line of information

ldi ZH, high(program\_version) ; point to the information that is to be displayed

ldi ZL, low(program\_version)

ldi temp, lcd\_LineTwo ; point to where the information should be displayed

call lcd\_write\_string\_8d

; endless loop

here:

rjmp here

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Main Program Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; ============================== 8-bit LCD Subroutines ======================

; Name: lcd\_init\_8d

; Purpose: initialize the LCD module for a 8-bit data interface

; Entry: equates (LCD instructions) set up for the desired operation

; Exit: no parameters

; Notes: uses time delays instead of checking the busy flag

lcd\_init\_8d:

; Power-up delay

ldi temp, 100 ; initial 40 mSec delay

call delayTx1mS

; Reset the LCD controller.

ldi temp, lcd\_FunctionReset ; first part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 10 ; 4.1 mS delay (min)

call delayTx1mS

ldi temp, lcd\_FunctionReset ; second part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 200 ; 100 uS delay (min)

call delayTx1uS

ldi temp, lcd\_FunctionReset ; third part of reset sequence

call lcd\_write\_instruction\_8d

ldi temp, 200 ; this delay is omitted in the data sheet

call delayTx1uS

; Function Set instruction

ldi temp, lcd\_FunctionSet4bit ; set mode 4 bit, lines, and font

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; Function Set instruction

ldi temp, lcd\_CMD\_4BIT\_MODE ; set mode 4 bit, lines, and font

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; The next three instructions are specified in the data sheet as part of the initialization routine,

; so it is a good idea (but probably not necessary) to do them just as specified and then redo them

; later if the application requires a different configuration.

; Display On/Off Control instruction

ldi temp, lcd\_DisplayOff ; turn display OFF

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; Clear Display instruction

ldi temp, lcd\_Clear ; clear display RAM

call lcd\_write\_instruction\_8d

ldi temp, 4 ; 1.64 mS delay (min)

call delayTx1mS

; Entry Mode Set instruction

ldi temp, lcd\_EntryMode ; set desired shift characteristics

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; This is the end of the LCD controller initialization as specified in the data sheet, but the display

; has been left in the OFF condition. This is a good time to turn the display back ON.

; Display On/Off Control instruction

ldi temp, lcd\_DisplayOn ; turn the display ON

call lcd\_write\_instruction\_8d

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_string\_8d

; Purpose: display a string of characters on the LCD

; Entry: ZH and ZL pointing to the start of the string

; (temp) contains the desired DDRAM address at which to start the display

; Exit: no parameters

; Notes: the string must end with a null (0)

; uses time delays instead of checking the busy flag

lcd\_write\_string\_8d:

; preserve registers

push ZH ; preserve pointer registers

push ZL

; fix up the pointers for use with the 'lpm' instruction

lsl ZL ; shift the pointer one bit left for the lpm instruction

rol ZH

; set up the initial DDRAM address

ori temp, lcd\_SetCursor ; convert the plain address to a set cursor instruction

call lcd\_write\_instruction\_8d ; set up the first DDRAM address

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

; write the string of characters

lcd\_write\_string\_8d\_01:

lpm temp, Z+ ; get a character

cpi temp, 0 ; check for end of string

breq lcd\_write\_string\_8d\_02 ; done

; arrive here if this is a valid character

call lcd\_write\_character\_8d ; display the character

ldi temp, 80 ; 40 uS delay (min)

call delayTx1uS

rjmp lcd\_write\_string\_8d\_01 ; not done, send another character

; arrive here when all characters in the message have been sent to the LCD module

lcd\_write\_string\_8d\_02:

pop ZL ; restore pointer registers

pop ZH

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_character\_8d

; Purpose: send a byte of information to the LCD data register

; Entry: (temp) contains the data byte

; Exit: no parameters

; Notes: does not deal with RW (busy flag is not implemented)

lcd\_write\_character\_8d:

sbi lcd\_RS\_port, lcd\_RS\_bit ; select the Data Register (RS high)

cbi lcd\_E\_port, lcd\_E\_bit ; make sure E is initially low

call LCD\_4bit\_write\_Data ; write the data

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_instruction\_8d

; Purpose: send a byte of information to the LCD instruction register

; Entry: (temp) contains the data byte

; Exit: no parameters

; Notes: does not deal with RW (busy flag is not implemented)

lcd\_write\_instruction\_8d:

cbi lcd\_RS\_port, lcd\_RS\_bit ; select the Instruction Register (RS low)

cbi lcd\_E\_port, lcd\_E\_bit ; make sure E is initially low

call LCD\_4bit\_write\_Data ; write the instruction

ret

; ---------------------------------------------------------------------------

; Name: lcd\_write\_8

; Purpose: send a byte of information to the LCD module

; Entry: (temp) contains the data byte

; RS is configured for the desired LCD register

; E is low

; RW is low

; Exit: no parameters

; Notes: use either time delays or the busy flag

LCD\_4bit\_write\_Data:

; Send upper nibble

ldi r20, 0xF0 ; Mask for upper nibble

and r20, temp

sbi lcd\_D7\_port, lcd\_D7\_bit ; assume that the data bit is '1'

sbrs r20, 7 ; check the actual data value

cbi lcd\_D7\_port, lcd\_D7\_bit ; arrive here only if the data was actually '0'

sbi lcd\_D6\_port, lcd\_D6\_bit ; repeat for each data bit

sbrs r20, 6

cbi lcd\_D6\_port, lcd\_D6\_bit

sbi lcd\_D5\_port, lcd\_D5\_bit

sbrs r20, 5

cbi lcd\_D5\_port, lcd\_D5\_bit

sbi lcd\_D4\_port, lcd\_D4\_bit

sbrs r20, 4

cbi lcd\_D4\_port, lcd\_D4\_bit

; write the data

; 'Address set-up time' (40 nS)

sbi lcd\_E\_port, lcd\_E\_bit ; Enable pin high

call delay1uS ; implement 'Data set-up time' (80 nS) and 'Enable pulse width' (230 nS)

cbi lcd\_E\_port, lcd\_E\_bit ; Enable pin low

call delay1uS ; implement 'Data hold time' (10 nS) and 'Enable cycle time' (500 n

nop

; Send lower nibble

ldi r20, 0x0F ; Mask for lower nibble

and r20, temp

LSL R20

LSL R20

LSL R20

LSL R20

sbi lcd\_D7\_port, lcd\_D7\_bit ; assume that the data bit is '1'

sbrs r20, 7 ; check the actual data value

cbi lcd\_D7\_port, lcd\_D7\_bit ; arrive here only if the data was actually '0'

sbi lcd\_D6\_port, lcd\_D6\_bit ; repeat for each data bit

sbrs r20, 6

cbi lcd\_D6\_port, lcd\_D6\_bit

sbi lcd\_D5\_port, lcd\_D5\_bit

sbrs r20, 5

cbi lcd\_D5\_port, lcd\_D5\_bit

sbi lcd\_D4\_port, lcd\_D4\_bit

sbrs r20, 4

cbi lcd\_D4\_port, lcd\_D4\_bit

; write the data

; 'Address set-up time' (40 nS)

sbi lcd\_E\_port, lcd\_E\_bit ; Enable pin high

call delay1uS ; implement 'Data set-up time' (80 nS) and 'Enable pulse width' (230 nS)

cbi lcd\_E\_port, lcd\_E\_bit ; Enable pin low

call delay1uS ; implement 'Data hold time' (10 nS) and 'Enable cycle time' (500 n

nop

nop

ret

; ============================== End of 8-bit LCD Subroutines ===============

; ============================== Time Delay Subroutines =====================

; Name: delayYx1mS

; Purpose: provide a delay of (YH:YL) x 1 mS

; Entry: (YH:YL) = delay data

; Exit: no parameters

; Notes: the 16-bit register provides for a delay of up to 65.535 Seconds

; requires delay1mS

delayYx1mS:

call delay1mS ; delay for 1 mS

dec YL ; update the the delay counter

brne delayYx1mS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delayTx1mS

; Purpose: provide a delay of (temp) x 1 mS

; Entry: (temp) = delay data

; Exit: no parameters

; Notes: the 8-bit register provides for a delay of up to 255 mS

; requires delay1mS

delayTx1mS:

call delay1mS ; delay for 1 mS

dec temp ; update the delay counter

brne delayTx1mS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delay1mS

; Purpose: provide a delay of 1 mS

; Entry: no parameters

; Exit: no parameters

; Notes: chews up fclk/1000 clock cycles (including the 'call')

delay1mS:

push YL ; [2] preserve registers

push YH ; [2]

ldi YL, low (((fclk/1000)-18)/4) ; [1] delay counter

ldi YH, high(((fclk/1000)-18)/4) ; [1]

delay1mS\_01:

dec YL ; [2] update the the delay counter

brne delay1mS\_01 ; [2] delay counter is not zero

; arrive here when delay counter is zero

pop YH ; [2] restore registers

pop YL ; [2]

ret ; [4]

; ---------------------------------------------------------------------------

; Name: delayTx1uS

; Purpose: provide a delay of (temp) x 1 uS with a 16 MHz clock frequency

; Entry: (temp) = delay data

; Exit: no parameters

; Notes: the 8-bit register provides for a delay of up to 255 uS

; requires delay1uS

delayTx1uS:

call delay1uS ; delay for 1 uS

dec temp ; decrement the delay counter

brne delayTx1uS ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)

ret

; ---------------------------------------------------------------------------

; Name: delay1uS

; Purpose: provide a delay of 1 uS with a 16 MHz clock frequency

; Entry: no parameters

; Exit: no parameters

; Notes: add another push/pop for 20 MHz clock frequency

delay1uS:

push temp ; [2] these instructions do nothing except consume clock cycles

pop temp ; [2]

push temp ; [2]

pop temp ; [2]

ret ; [4]

; ============================== End of Time Delay Subroutines ==============

Schematic:

+--------------------+ +------------------+

| | | |

| ATmega32 | | HD44780U LCD |

| | | |

| +------------+ | | +----------+ |

| | | | | | | |

| | PORTD |---|-> D0-D7 | | D0-D7 | |

| | (PB0-PB7)| | | | | |

| | | | | | | |

| | PA0(RS) |---|-> RS | | RS | |

| | PA1(E) |---|-> E | | E | |

| | GND |---|-> RW | | RW (GND) | |

| +------------+ | | +----------+ |

+--------------------+ +------------------+

Flowchart:

+--------------------------+

| Start |

+--------------------------+

|

+--------------------------+

| Initialize Stack Pointer |

+--------------------------+

|

+--------------------------+

| Configure Data Lines |

| (PORTD as output) |

+--------------------------+

|

+--------------------------+

| Configure Control Lines |

| (RS, E as output) |

+--------------------------+

|

+--------------------------+

| LCD Initialization |

| (Send reset commands) |

+--------------------------+

|

+--------------------------+

| Display First Line (Name) |

+--------------------------+

|

+--------------------------+

| Display Second Line |

| (University Name) |

+--------------------------+

|

+--------------------------+

| Endless Loop (here) |

+--------------------------+

|

+--------------------------+

| Subroutines |

| (lcd\_write\_string\_8d, |

| lcd\_write\_instruction\_8d, |

| lcd\_write\_character\_8d) |

+--------------------------+

**END**