LAB6 home assignment **Lukáš Kaleta**

1. Preparation tasks (done before the lab at home). Submit:
    - Table with LCD signals,

LCD signal(s)	AVR pin(s)	Description
RS	PB0	Register selection signal. Selection between Instruction register (RS=0) and Data register (RS=1)
R/W	GND	Choosing between reading (R/W = 1) or writing (R/W = 0 or GND)
E	PB1	On the falling edge of E signal data transfers ito the LCD
D[3:0]	-	Lower 4 data pins. Used only for 8bit communication
D[7:4]	PD[7:4]	Upper 4 data pins. Usable for both 4bit and 8bit communication

Function name	Function parameters	Description	Example

lcd_init	LCD_DISP_OFF
	LCD_DISP_ON
	LCD_DISP_ON_CURSOR
	LCD_DISP_ON_CURSOR_BLINK
		display off
		display on, cursor off
		display on, cursor on
		disp on, cursor flashing
			`lcd_init(LCD_DISP_OFF);`
			`lcd_init(LCD_DISP_ON);`
			`lcd_init(LCD_DISP_ON_CURSOR);`
			`lcd_init(LCD_DISP_ON_CURSOR_BLINK);`

lcd_clrscr	-	clear screen	`lcd_clrscr();`
lcd_gotoxy	x, y	Set cursor to specified position.	`lcd_gotoxy(1,0);`
lcd_putc	c	Wtrite one symbol on current position	`lcd_putc(a);`
lcd_puts	“string“	Wtrite given string on current position	`lcd_putc(“DE_2“);`
lcd_command	cmd	Send LCD controller instruction command	`lcd_command(uint8_t cmd);`
lcd_data	data	Send data byte to LCD controller	`lcd_data(uint8_t data)`

- ASCII values.

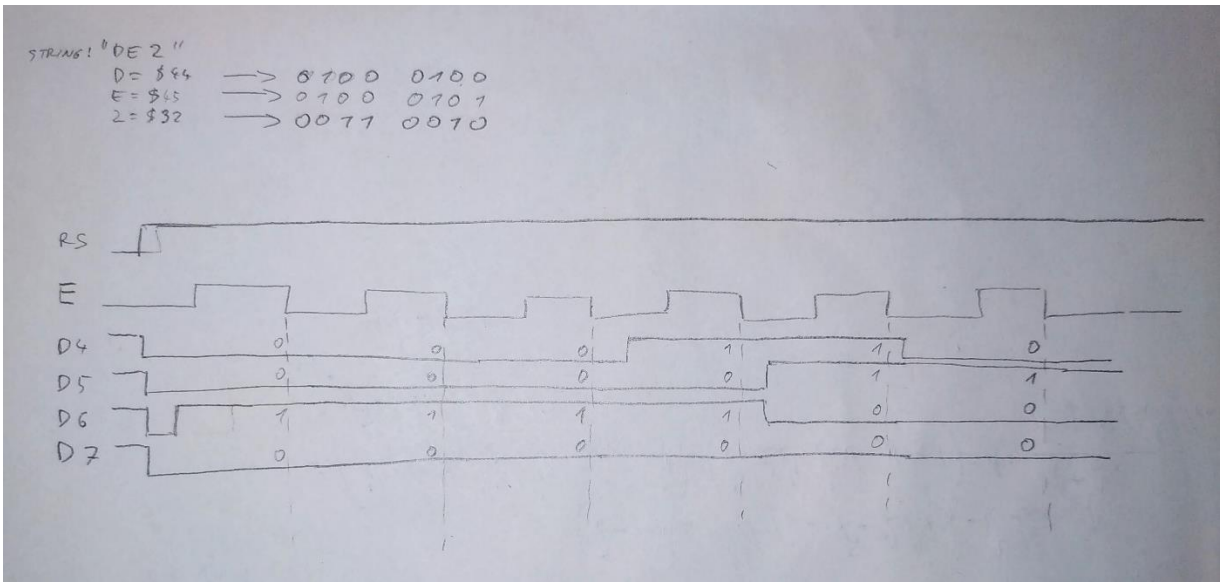
A = $41
B = $42
…
Z = $5A

a = $61
…
z = $7A

0 = $30
1 = $31
…
9 = $39

2. HD44780 communication. Submit:

   - Picture of time signals between ATmega328P and HD44780 (LCD keypad shield) when transmitting data DE2.



3. Stopwatch. Submit:

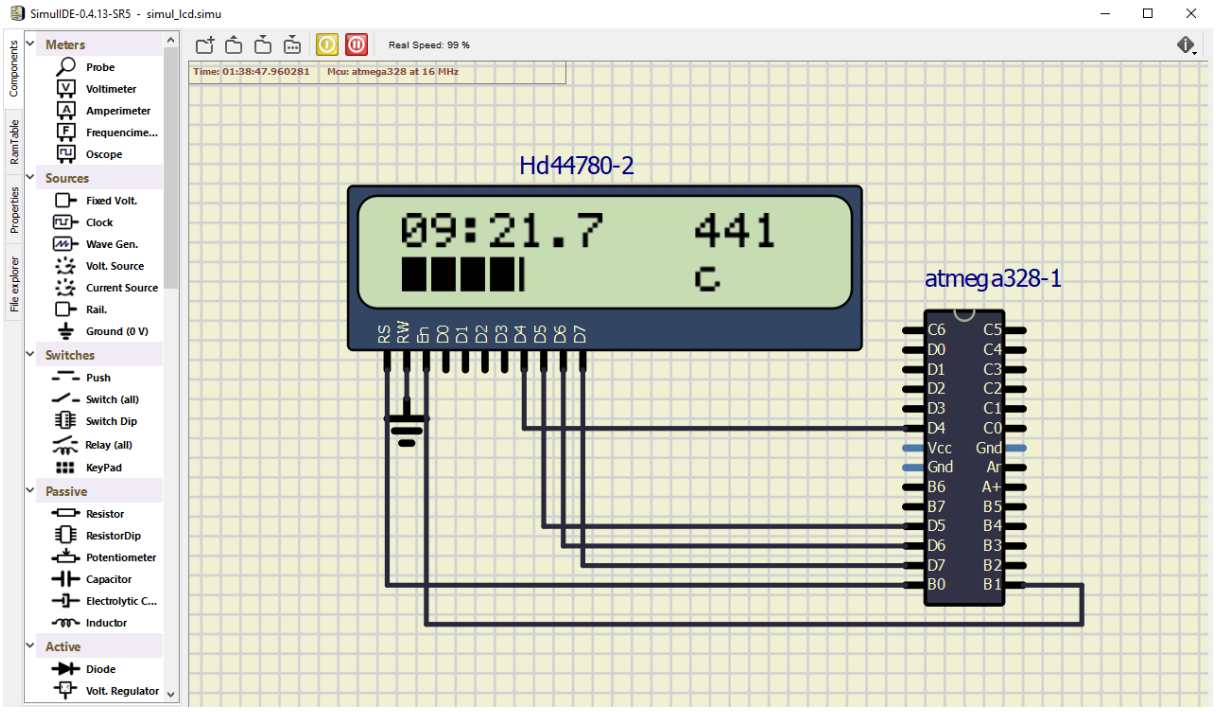
   - Listing of TIMER2_OVF_vect interrupt routine with complete stopwatch code (minutes:seconds.tenths) and square value computation,

```
/* Interrupt service routines ------------------------------------------*/
/**
 * ISR starts when Timer/Counter2 overflows. Update the stopwatch on
 * LCD display every sixth overflow, ie approximately every 100 ms
 * (6 x 16 ms = 100 ms).
 */
```

```
ISR(TIMER2_OVF_vect)
{
    static uint8_t number_of_overflows = 0;
    static uint8_t tens = 0;        // Tenths of a second
    static uint8_t secs = 0;        // Seconds
    static uint8_t mins = 0;        // Minutes

    static uint16_t secs_square = 0; // squared number of secs on position A

    char lcd_string[] = "  ";      // String for converting numbers by itoa()

    number_of_overflows++;
    if (number_of_overflows >= 5)
    {
        // Do this every 6 x 16 ms = cca 100 ms
        number_of_overflows = 0;

        // WRITE YOUR CODE HERE
        tens++;
        if(tens > 9)
        {
              tens = 0;
              secs++;
              if(secs > 9)
              {
                    if(secs > 59)
                    {
                          secs = 0;
                          lcd_gotoxy(5, 0);
                          lcd_puts("0");

                          mins++;
                          if(mins > 9)
                          {
                                if(mins > 59)
                                {
                                      mins = 0;
                                      lcd_gotoxy(2, 0);
                                      lcd_puts("0");
                                }
                                // displaying > 9 minutes
                                itoa(mins, lcd_string, 10);
                                lcd_gotoxy(1, 0);
                                lcd_puts(lcd_string);
                          }

                          else
                          {
                                // displaying < 10 minutes
                                itoa(mins, lcd_string, 10);
                                lcd_gotoxy(2, 0);
                                lcd_puts(lcd_string);
                          }

                          // clearing square of secs
                          lcd_gotoxy(11, 0);
                          lcd_puts("    ");

                    }
                    // displaying > 9 seconds
                    itoa(secs, lcd_string, 10);
                    lcd_gotoxy(4, 0);
```

```
                lcd_puts(lcd_string);

            }
            else
            {
                // displaying < 10 seconds
                itoa(secs, lcd_string, 10);
                lcd_gotoxy(5, 0);
                lcd_puts(lcd_string);
            }

            // computing square of secs
            secs_square = secs * secs;
            itoa(secs_square, lcd_string, 10);
            lcd_gotoxy(11, 0);
            lcd_puts(lcd_string)
        }
        // displaying tenths of seconds
        itoa(tens, lcd_string, 10);
        lcd_gotoxy(7, 0);
        lcd_puts(lcd_string);
    }
}
```

- Screenshot of SimulIDE circuit when "Power Circuit" is applied.



4. Progress bar. Submit:

- Listing of TIMER0_OVF_vect interrupt routine with a progress bar,

```
/*-------------------------------------------------------------------*/
/**
 * ISR starts when Timer/Counter0 overflows. Update the progress bar on
 * LCD display every 16 ms.
 */
```

```
ISR(TIMER0_OVF_vect)
{
	static uint8_t number_of_overflows = 0;
	static uint8_t symbol = 0;
	static uint8_t position = 1;


	number_of_overflows++;

	if (number_of_overflows >= 6)
	{
		number_of_overflows = 0;
		symbol++;
		if (symbol > 5)                                  // one bar is full
		{
			symbol = 1;
			position++;
			if (position > 9)
			{
				position = 1;                            // clear position
				lcd_gotoxy(1, 1);                        // clear all bars
				lcd_puts("          ");
			}
		}

		// display bar
		lcd_gotoxy(position, 1);
		lcd_putc(symbol);
	}
}
```

- Screenshot of SimulIDE circuit when "Power Circuit" is applied.