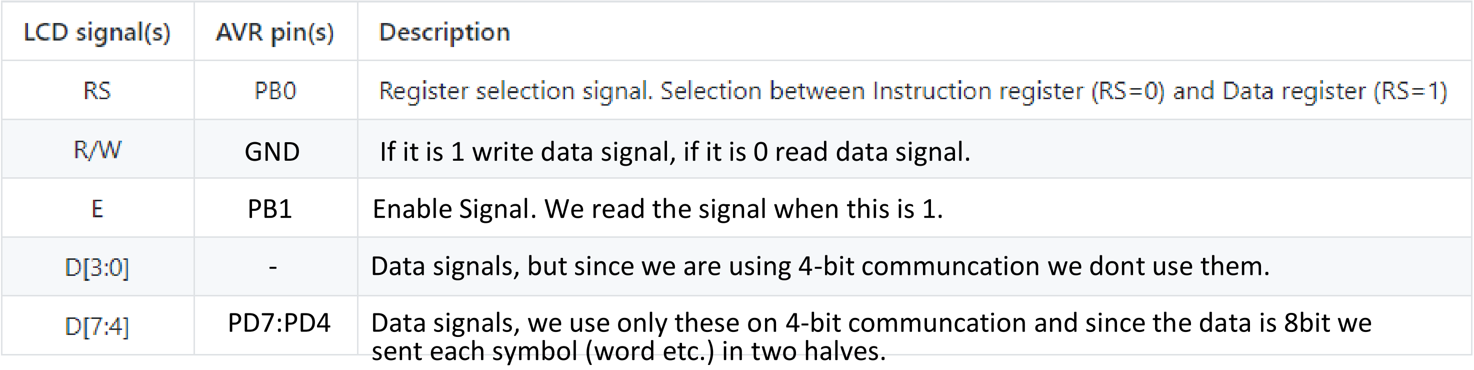
DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 6

Name: Demirkan Korbey Baglamac
Github Repository Link: Click Here

1) What data and control signals are used? What is the meaning of these 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	If it is 1 write data signal, if it is 0 read data signal.
E	PB1	Enable Signal. We read the signal when this is 1.
D[3:0]	-	Data signals, but since we are using 4-bit communcation we dont use them.
D[7:4]	PD7:PD4	Data signals, we use only these on 4-bit communcation and since the data is 8bit we sent each symbol (word etc.) in two halves.

What is the ASCII table? What are the values for uppercase letters A to Z, lowercase letters a to z, and numbers 0 to 9 in this table?

ASCII is American Standard Code for Information Interchange.

HEXADECIMAL	BINARY	SYMBOL
30	00110000	0
31	00110001	1
32	00110010	2
33	00110011	3
34	00110100	4
35	00110101	5
36	00110110	6
37	00110111	7
38	00111000	8
39	00111001	9
41	01000001	A
42	01000010	B
43	01000011	C
44	01000100	D
45	01000101	E
46	01000110	F
47	01000111	G

48	01001000	H
49	01001001	I
4A	01001010	J
4B	01001011	K
4C	01001100	L
4D	01001101	M
4E	01001110	N
4F	01001111	O
50	01010000	P
51	01010001	Q
52	01010010	R
53	01010011	S
54	01010100	T
55	01010101	U
56	01010110	V
57	01010111	W
58	01011000	X
59	01011001	Y
5A	01011010	Z
61	01100001	a
62	01100010	b
63	01100011	c
64	01100100	d
65	01100101	e
66	01100110	f
67	01100111	g
68	01101000	h
69	01101001	i
6A	01101010	j
6B	01101011	k
6C	01101100	l
6D	01101101	m
6E	01101110	n
6F	01101111	o

70	01110000	p
71	01110001	q
72	01110010	r
73	01110011	s
74	01110100	t
75	01110101	u
76	01110110	v
77	01110111	w
78	01111000	x
79	01111001	y
7A	01111010	z

2)

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
		Display on, Cursor on 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`	void	Clear display and set cursor to home pos.	`lcd_clrscr();`
`lcd_gotoxy`	uint8_t x, uint8_t y	Set cursor to specified position	lcd_gotoxy(0,0);
		x: horizontal position, y: vertical pos.	
`lcd_putc`	char c	Display character at current cursor.	lcd_putc(“c”);
`lcd_puts`	const char * c	Display string without auto linefeed.	lcd_puts(“hello”);
`lcd_command`	uint8_t cmd	Send LCD controller instruction command.	lcd_command(1<<LCD_CGRAM);
`lcd_data`	uint8_t data	Send data byte to LCD controller.	lcd_data(customChar[i]);

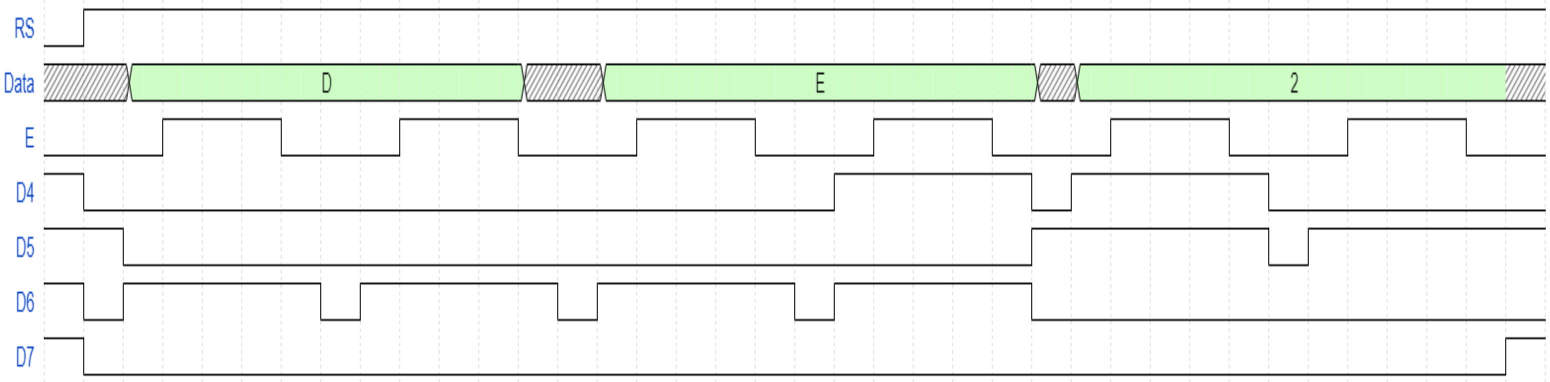
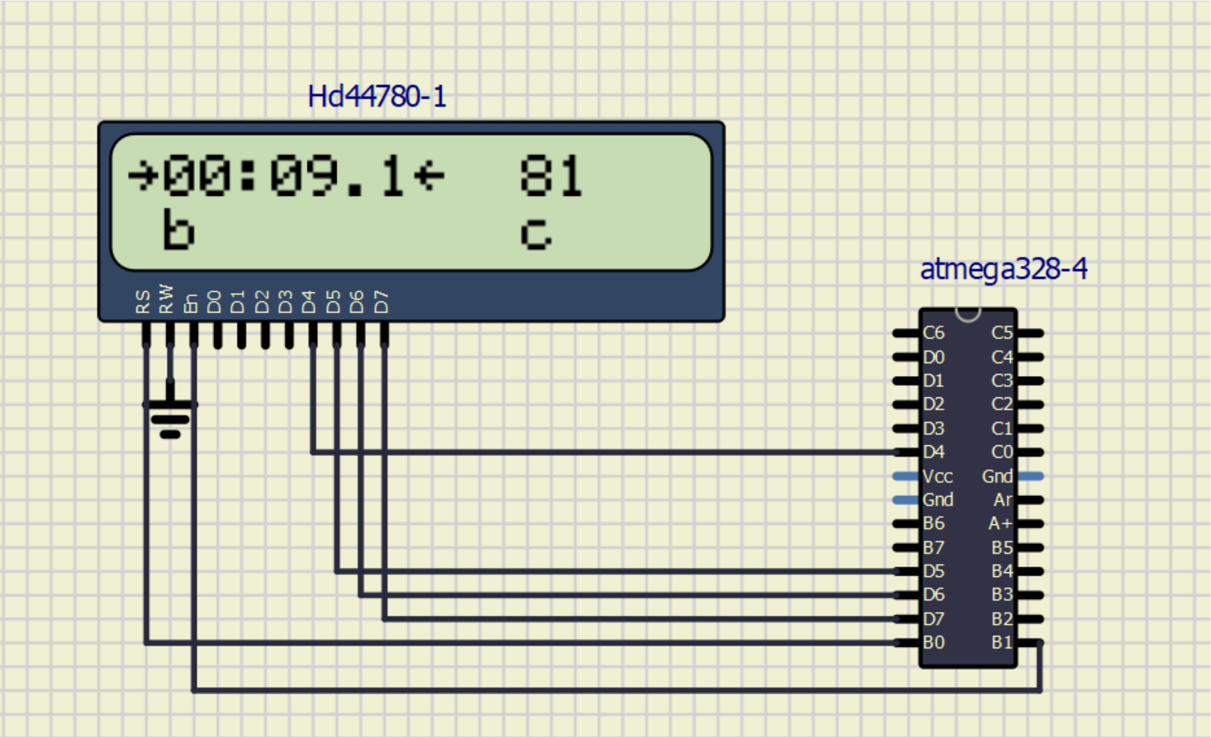


Table 1: Waveform for DE2

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

```c
/**
 * 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
    char lcd_string[2] = "  ";      // String for converting numbers by itoa()
    char lcd_longstring[4];

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

        // Update the tenths of a second
        tens++;
        if (tens >= 10)
        {
            tens = 0;

            //Update the seconds
            secs++;
            if(secs >= 60)
            {
                secs = 0;

                // Update the minutes
                mins++;
                if(mins >= 60)
                    mins = 0;

                // Display minutes
                lcd_gotoxy(1,0);
                if(mins < 10)
                    lcd_putc('0');
                itoa(mins, lcd_string, 10);
                lcd_puts(lcd_string);

            }

            //Display Seconds
            lcd_gotoxy(4,0);
            if(secs < 10)
                lcd_putc('0');
            itoa(secs, lcd_string, 10);
            lcd_puts(lcd_string);

            // Display the square value of the Seconds
            lcd_gotoxy(11, 0);
            itoa((secs * secs), lcd_longstring, 10);
            lcd_puts(lcd_longstring);
            if(secs == 0)
                lcd_puts("   ");
                }

                // Display hundredths of seconds
                lcd_gotoxy(7,0);
                // Converting cnt0 in decimal to string
                itoa(tens, lcd_string, 10);
                lcd_puts(lcd_string);

        }
}
```

Screenshot of the circuit:



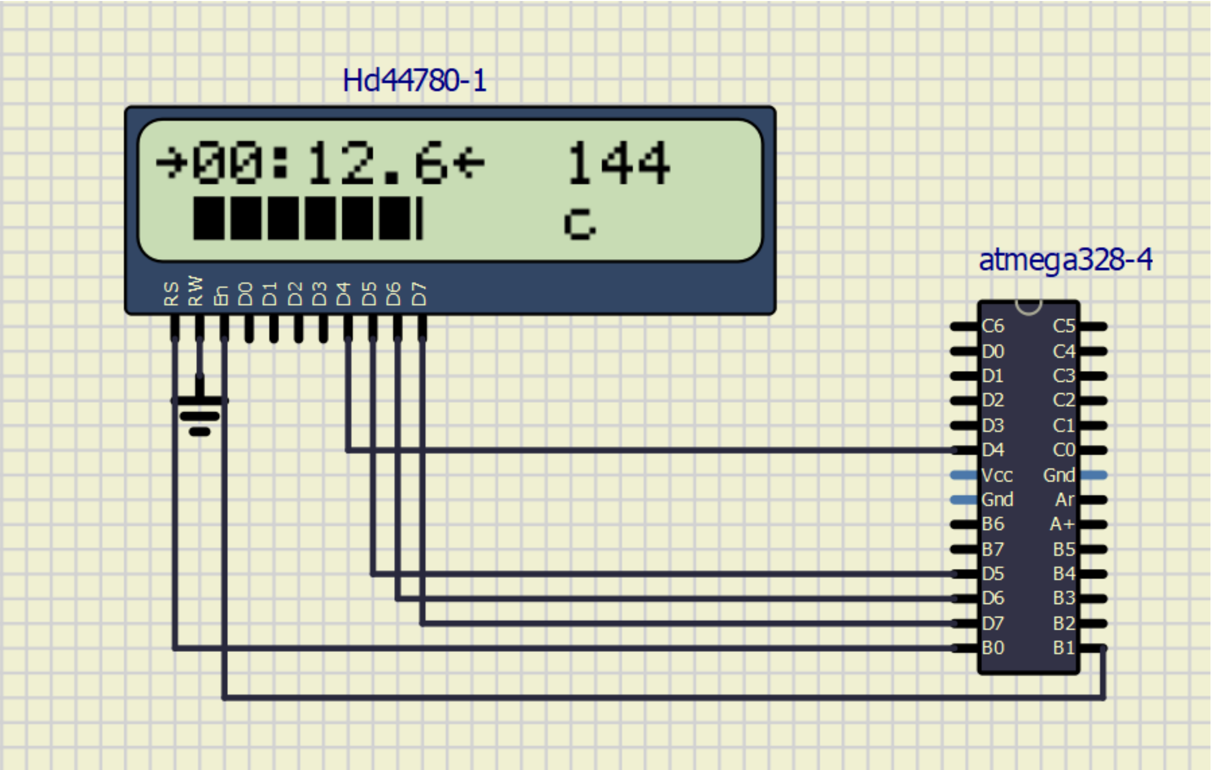
```
→00:09.1←   81
 b          c
```

4) 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 symbol = 0;
        static uint8_t position = 0;

        lcd_gotoxy(1 + position, 1);
        lcd_putc(symbol);
```

```
    symbol++;
    if(symbol >= 6)
    {
        symbol = 0;
        position++;
        if(position >= 10) {
            position = 0;
            lcd_gotoxy(1,1);
            for(uint8_t i; i < 10; i++) {
                lcd_putc(0);
            }
        }
    }
}
```

Screenshot of the circuit:


Hd44780-1
→00:12.6←   144
c
RS
RW
En
D0
D1
D2
D3
D4
D5
D6
D7
atmega328-4
C6
D0
D1
D2
D3
D4
Vcc
Gnd
B6
B7
D5
D6
D7
B0
C5
C4
C3
C2
C1
C0
Gnd
Ar
A+
B5
B4
B3
B2
B1