Link to my Github

Table 1: Library for HD44780 based LCDs

Function name	Function parameters	Description	Example
lcd_init	LCD_DISP_OFF LCD_DISP_ON LCD_DISP_ON_CURSOR LCD_DISP_ON_CURSOR_BLINK	Initialize display and select type of cursor	lcd_init(LCD_DISP_OFF);
lcd_clrscr	None	Clear display and set cursor to home position	lcd_clrscr();
lcd_gotoxy	x horizontal position (0: left most position) y vertical position (0: first line)	Set cursor to specified position.	lcd_gotoxy(0,0);
lcd_putc	c character to be displayed	Display character at current cursor position	lcd_putc('0');
lcd_puts	s string to be displayed	Display string without auto linefeed	lcd_puts("DE2");
lcd_command	cmd instruction to send to LCD controller, see HD44780 data sheet	Send LCD controller instruction command	lcd_command(1 << LCD_DDRAM);
lcd_data	data byte to send to LCD controller, see HD44780	Send data byte to LCD controller	lcd_data(data);

Table 2: LCD signals

LCD signals	AVR pins	Description
RS	PB0	Register selection signal. Selection between Instruction register (RS=0) and Data register (RS=1)
R/W	GND	Write data signal (R/W=0), read data signal (R/W=1), pin is GND -> only write
E	PB1	Enable signal.
D[3:0]	not used	Data signals, 8 bit mode D[7:0]
D[7:4]	PD7:PD4	Data signals, 4 bit mode (2 E signals needed)

What is the ASCII code?

ASCII is the acronym for the American Standard Code for Information Interchange. It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127. For example, the ASCII code for uppercase M is 77. Most computers use ASCII codes to represent text, which makes it possible to transfer data from one computer to another.

Table 3: ASCII Value

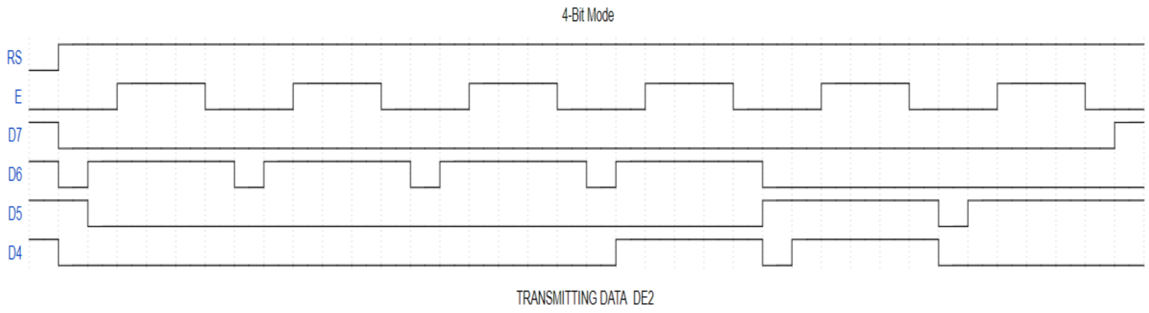
Characters	ASCII Value
A - Z	65 - 95
a - z	97 - 122
0 – 9	48-57
Special Symbol	0 - 47, 58 - 64, 91 - 96, 123 - 127

Table 4: ASCII Table

Decimal	Octal	Hexadecimal	Character
048	060	30	0
049	061	31	1
050	062	32	2
051	063	33	3
052	064	34	4
053	065	35	5
054	066	36	6
055	067	37	7
056	070	38	8
057	071	39	9

Decimal	Octal	Hexadecimal	Character	Decimal	Octal	Hexadecimal	Character
065	101	41	A	097	141	61	a
066	102	42	B	098	142	62	b
067	103	43	C	099	143	63	c
068	104	44	D	100	144	64	d
069	105	45	E	101	145	65	e
070	106	46	F	102	146	66	f
071	107	47	G	103	147	67	g
072	110	48	H	104	150	68	h
073	111	49	I	105	151	69	i
074	112	4A	J	106	152	6A	j
075	113	4B	K	107	153	6B	k
076	114	4C	L	108	154	6C	l
077	115	4D	M	109	155	6D	m
078	116	4E	N	110	156	6E	n
079	117	4F	O	111	157	6F	o
080	120	50	P	112	160	70	p
081	121	51	Q	113	161	71	q
082	122	52	R	114	162	72	r
083	123	53	S	115	163	73	s
084	124	54	T	116	164	74	t
085	125	55	U	117	165	75	u
086	126	56	V	118	166	76	v
087	127	57	W	119	167	77	w
088	130	58	X	120	170	78	x
089	131	59	Y	121	171	79	y
090	132	5A	Z	122	172	7A	z

Figure 1: 4-Bit Data Transfer



Listing of TIMER 2

```
ISR(TIMER2_OVF_vect)
{

        static uint8_t number_of_overflows = 0;
        static uint8_t tens = 0;        // Tenths of a second
        static uint8_t seconds = 0;       // Seconds
        static uint8_t minutes = 0;       // Minutes
        static uint16_t  square_seconds = 0;
        char lcd_string[2] = "00";      // String for converting numbers by itoa()



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

                if(tens > 9)  // If we reach the maximum of the Tenths
                                    // then we have to reset and Update seconds
                {
                        tens = 0;

                        seconds++;


                        if(seconds < 10)
                        {

                                lcd_gotoxy(4, 0);
                                itoa(seconds, lcd_string, 10);
                                lcd_putc('0');
                                lcd_puts(lcd_string);

                        }
                        else
                        {
                                lcd_gotoxy(4, 0);
                                itoa(seconds, lcd_string, 10);
                                lcd_puts(lcd_string);

                        }
```

```
    if (seconds > 59)
    {
        seconds=0;

        lcd_gotoxy(4, 0);
        itoa(seconds, lcd_string, 10);
        lcd_puts(lcd_string);

        // Update minutes
        minutes++;

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

        }
        else
        {
            lcd_gotoxy(1,0);
            itoa(minutes, lcd_string, 10);
            lcd_puts(lcd_string);
        }

        if (minutes > 59)
        {

            minutes = 0;
            lcd_gotoxy(1,0);
            lcd_puts("00");

        }

        // Clearing the square of Second
        lcd_gotoxy(12, 0);
        lcd_putc(' ');
        lcd_gotoxy(13, 0);
        lcd_putc(' ');
        lcd_gotoxy(14, 0);
        lcd_putc(' ');




    }


}

// Displaying the square of seconds
square_seconds= seconds*seconds;
lcd_gotoxy(11, 0);
itoa(square_seconds, lcd_string, 10);
lcd_puts(lcd_string);


// Display hundredths of seconds
lcd_gotoxy(7, 0);
// Convert the value in decimal to string
itoa(tens, lcd_string, 10);
lcd_puts(lcd_string);
// Update the tenths of second
```

```
        }

}
```

Listing of TIMER0

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



        symbol++;
        if(symbol > 5)
        {
                symbol = 0;
                position++;
                if(position > 9)
                {
                        position = 0;
                        lcd_gotoxy(1+i,1);
                        while(i < 10)
                        {

                                lcd_putc(' ');
                                i++;
                         }
                }

        }

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

}
```

Figure 2: Stopwatch and square value

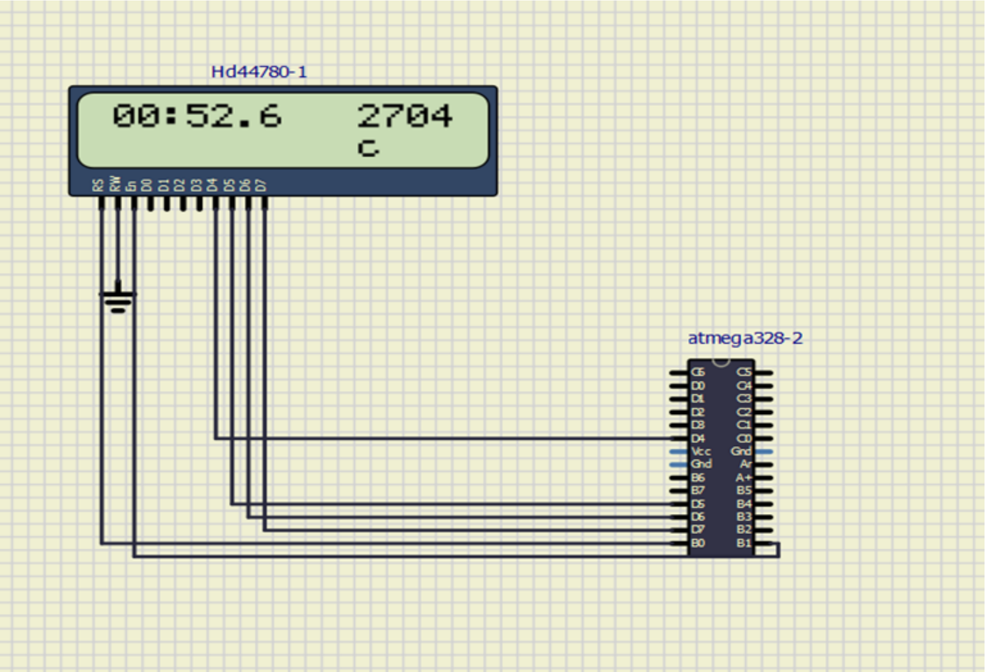
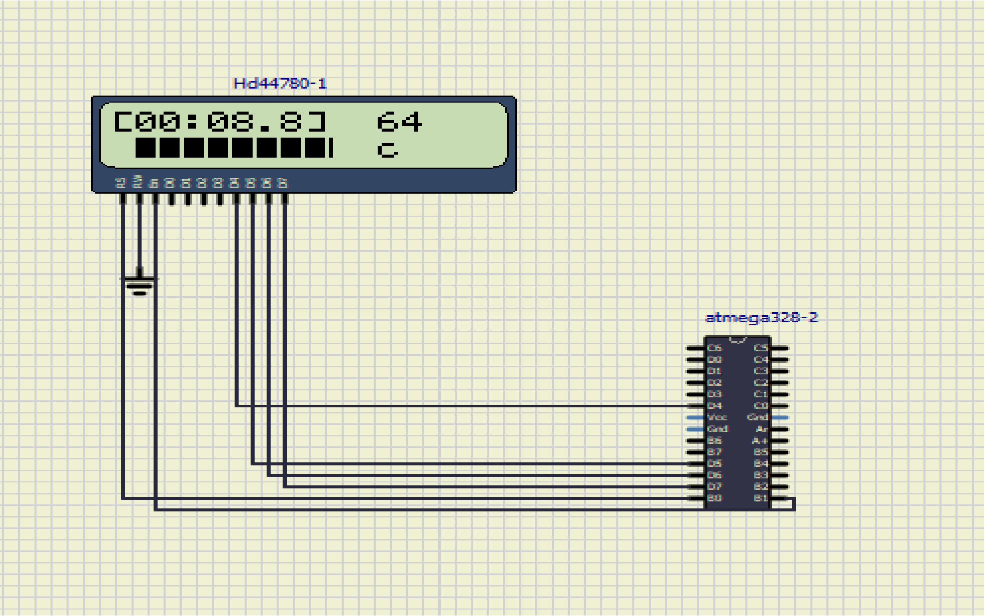

Hd44780-1
00:52.6 2704
c
RS RW En D0 D1 D2 D3 D4 D5 D6 D7
atmega328-2
C6 D0 D1 D2 D3 D4 Vcc Gnd B6 B7 D5 D6 D7 B0
C5 C4 C3 C2 C1 C0 Gnd Ar A+ B5 B4 B3 B2 B1

Figure 3: Stopwatch with a progress bar


Hd44780-1
[00:08.8] 64
█████████▌ c
RS RW En D0 D1 D2 D3 D4 D5 D6 D7
atmega328-2
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