master Digital-Electronics-2 / labs / lab6 / Go to file Add file

FilipPaul No commit message b8ba843 32 seconds ago History

..		
.vscode	sync	18 hours ago
include	sync	18 hours ago
lib	sync	18 hours ago
pictures	No commit message	4 minutes ago
src	No commit message	4 minutes ago
test	sync	18 hours ago
.gitignore	sync	18 hours ago
platformio.ini	sync	18 hours ago
readme.md	No commit message	32 seconds ago
simulation.simu	sync	18 hours ago
simulation_backup.simu	No commit message	4 minutes ago

readme.md

Lab 6

Preparation Task

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	LOW - WRITE to the display, HIGH READ - read from display
E	PB1	enable. This loads the data into the HD44780 on the falling edge
D[3:0]	PD3 - PD0	data pins for custom characters
D[7:4]	PD7 - PD4	Upper nibble used in 4-bit mode

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 table is a way how to describe characters with numbers in DEC, BIN, HEX etc. Each special charracter or letter is assigned to a number. For example 'A' is represented by: DEC 65, HEX 41 and bin 100 0001.

ASCII table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Lab results

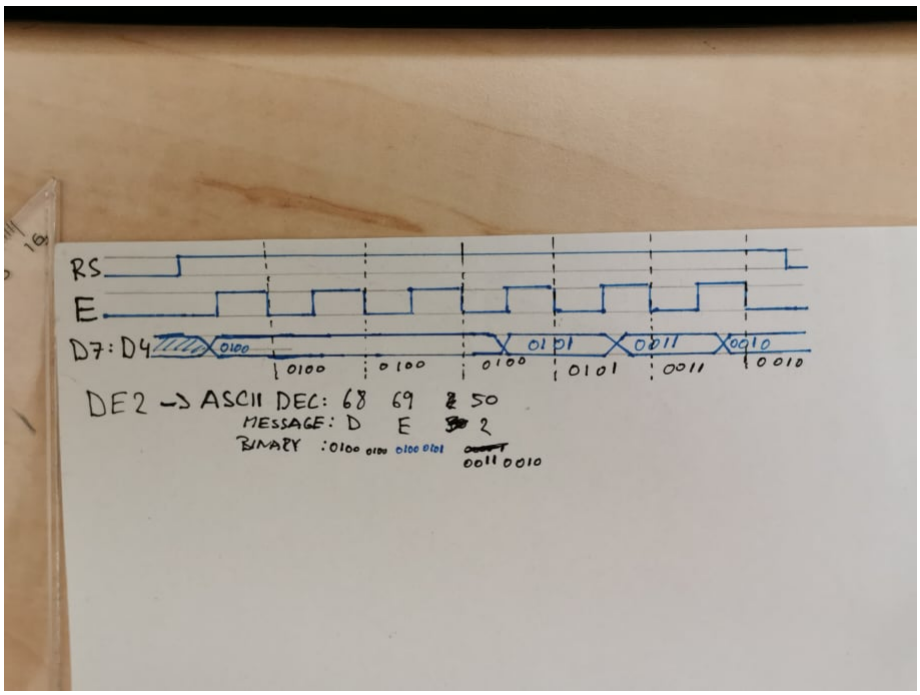
Table of functions from LCD library

Function name	Function parameters	Description	Example

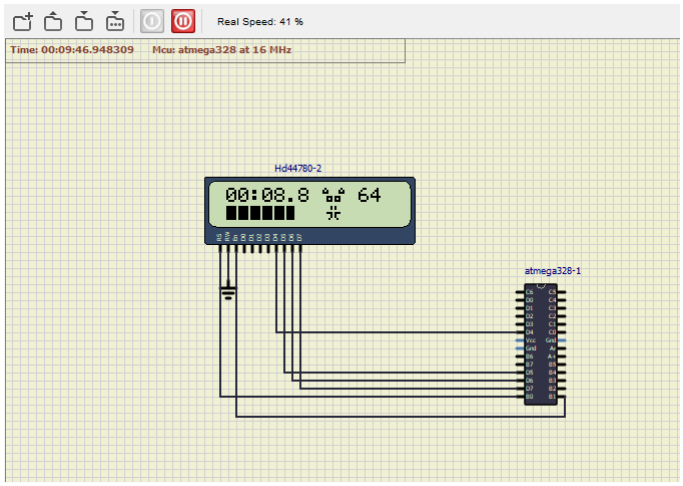
`lcd_init`	`LCD_DISP_OFF`
	`LCD_DISP_ON`
	`LCD_DISP_ON_CURSOR`
	`LCD_DISP_ON_CURSOR_BLINK`
		Display off
		Display on, cursor off
		Display on, cursor on
		Display on, cursor on, blink char
			`lcd_init(LCD_DISP_OFF);`

`lcd_clrscr`	none	Clear display and set cursor to home position.	`lcd_clrscr();`
`lcd_gotoxy`	uint8_t x, uint8_t y	set cursor's position	`lcd_gotoxy(x,y);`
`lcd_putc`	char c	Display character at current cursor position.	`lcd_putc(c);`
`lcd_puts`	string s	Display string without auto linefeed.	`lcd_puts(s);`
`lcd_command`	uint8_t cmd instruction to send to LCD controller	Send LCD controller instruction command.	`lcd_command(cmd)`
`lcd_data`	uint8_t data	Send data byte to LCD controller.	`lcd_data(data);`

HD44780 communication, message: DE2



Simulation



main.cpp

```
/* Includes ------------------------------------------------------------*/
#include <Arduino.h>
#include <avr/io.h>         // AVR device-specific IO definitions
#include <avr/interrupt.h>  // Interrupts standard C library for AVR-GCC
#include "timer.h"          // Timer library for AVR-GCC
#include "lcd.h"            // Peter Fleury's LCD library
#include <stdlib.h>         // C library. Needed for conversion function

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Update stopwatch
```

```
 * value on LCD display when 8-bit Timer/Counter2 overflows.
 */
//custom character
uint8_t customChar[8*8] = { //adress 0
       0b01000,0b10100,0b11100,0b00000,0b00111,0b00101,0b00111,0b10000,
//adress 1
    0b00010,0b00101,0b00111,0b00000,0b11100,0b10100,0b11100,0b00001,
//adress 2
    0b00001,0b00001,0b00000,0b00111,0b00001,0b00001,0b00010,0b00100,
// adress 3
       0b10000,0b10000,0b00000,0b11100,0b10000,0b10000,0b01000,0b00100,
// adress 4
       0b10000,0b10000,0b10000,0b10000,0b10000,0b10000,0b10000,0b10000,
// adress 5
    0b11000,0b11000,0b11000,0b11000,0b11000,0b11000,0b11000,0b11000,
// adress 6
    0b11100,0b11100,0b11100,0b11100,0b11100,0b11100,0b11100,0b11100,
// adress 7
    0b11110,0b11110,0b11110,0b11110,0b11110,0b11110,0b11110,0b11110
};

int main(void)
{
    // Initialize LCD display
    lcd_init(LCD_DISP_ON);
    lcd_gotoxy(4-1,0);
    lcd_puts(":");
    lcd_gotoxy(7-1,0);
    lcd_puts(".");
    lcd_gotoxy(10-1,0);
    lcd_puts("a");

// display custom character
 lcd_command(1 << LCD_CGRAM);
    for (uint8_t i = 0; i < 8*8; i++)
    {
        // Store all new chars to memory line by line
        lcd_data(customChar[i]);
    }
    // Set DDRAM address
    lcd_command(1 << LCD_DDRAM);

    // Display first custom character
    lcd_gotoxy(10-1,0);
    lcd_putc(0);
    lcd_gotoxy(11-1,0);
    lcd_putc(1);
    lcd_gotoxy(10-1,1);
    lcd_putc(2);
    lcd_gotoxy(11-1,1);
    lcd_putc(3);
    lcd_gotoxy(2-1,1);

    // Configure 8-bit Timer/Counter2 for Stopwatch
    // Set prescaler and enable overflow interrupt every 16 ms
    TIM2_overflow_16384us();
    TIM2_overflow_interrupt_enable();
    //timer for "progress bar"
    TIM0_overflow_1024us();
    TIM0_overflow_interrupt_enable();

    // Enables interrupts by setting the global interrupt mask
    sei();

    // Infinite loop
    while (1)
    {
    }
    // Will never reach this
    return 0;
}

/* Interrupt service routines ----------------------------------------*/
ISR(TIMER0_OVF_vect)
// progress bar
{
    static uint16_t n = 0;
    static uint8_t c = 0; //character to display
    static uint8_t pos = 0; //character to display
    n++;
    if (n >= 28) //1024 us *28 = 28.7ms --> aprox 35 times per second (display 8x5 chars per second)
    {
        c++;
        if (c > 4){ // c = 5, move cursor and reset C
            c = 0;
            pos++;
            if (pos >7-1){ // pos = 7 reset position and delete content
                pos = 0;
                lcd_gotoxy(1,1);
                lcd_puts("        "); // delete progress bar
            }
            lcd_gotoxy(pos+1,1);
            lcd_putc(c+4); // offset to correct character
        }
        else if (c == 4 ){ //this symbol was already defined
            lcd_gotoxy(pos+1,1);
            lcd_putc(0xff); // all black
        }
        else{ // c = 0,1,2
            lcd_gotoxy(pos+1,1);
            lcd_putc(c+4); // offset to correct character
        }
      n = 0;
    }

}


/**
 * ISR starts when Timer/Counter2 overflows. Update the stopwatch on
```

```
 * LCD display every sixth overflow, ie approximately every 100 ms
 * (6 x 16384us = cca 100 ms).
 */
ISR(TIMER2_OVF_vect)
{
    static uint8_t number_of_overflows = 0;
    static uint8_t tens = 0;
    static uint8_t secs = 0;
    static uint8_t secs10 = 0;
    static uint8_t min = 0;
    static uint8_t min10 = 0;
    char lcd_string[2]  = " ";
    number_of_overflows++;

    if (number_of_overflows >= 6)
    {
        // Do this every 6 x 16 ms = cca 100 ms
        number_of_overflows = 0;
        tens++;
        if (tens > 9)
        {
            secs++;
            tens = 0;
        }
        if (secs > 9){
            secs = 0;
            tens = 0;
            secs10++;
        }
        if (secs10 > 5){
            secs = 0;
            tens = 0;
            secs10 = 0;
            min++;
            lcd_gotoxy(13-1,0);
            lcd_puts("    "); //4 blank characters to reset
        }
        if (min > 9){
            secs = 0;
            tens = 0;
            secs10=0;
            min = 0;
            min10++;
        }
            if (min10 > 5){
            secs = 0;
            tens = 0;
            secs10=0;
            min = 0;
            min10 = 0;
        }

// display values
        itoa(tens,lcd_string,10);
        lcd_gotoxy(8-1,0);
        lcd_puts(lcd_string);

        itoa(secs,lcd_string,10);
        lcd_gotoxy(6-1,0);
        lcd_puts(lcd_string);

        itoa(secs10,lcd_string,10);
        lcd_gotoxy(5-1,0);
        lcd_puts(lcd_string);

        itoa(min,lcd_string,10);
        lcd_gotoxy(3-1,0);
        lcd_puts(lcd_string);

        itoa(min10,lcd_string,10);
        lcd_gotoxy(2-1,0);
        lcd_puts(lcd_string);
        //squared sec
        itoa((secs+secs10*10)*(secs+secs10*10),lcd_string,10);
        lcd_gotoxy(13-1,0);
        lcd_puts(lcd_string);
        number_of_overflows = 0;
    }
}
```