DE2 – project 6

Václav Pastušek

xpastu02

204437

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	Read/Write
E	PB1	Enable loads the data into the HD44780 on the falling edge.
D[3:0]	X	Used in 8-bit mode.
D[7:4]	D7-D4	Upper nibble used in 4-bit mode.

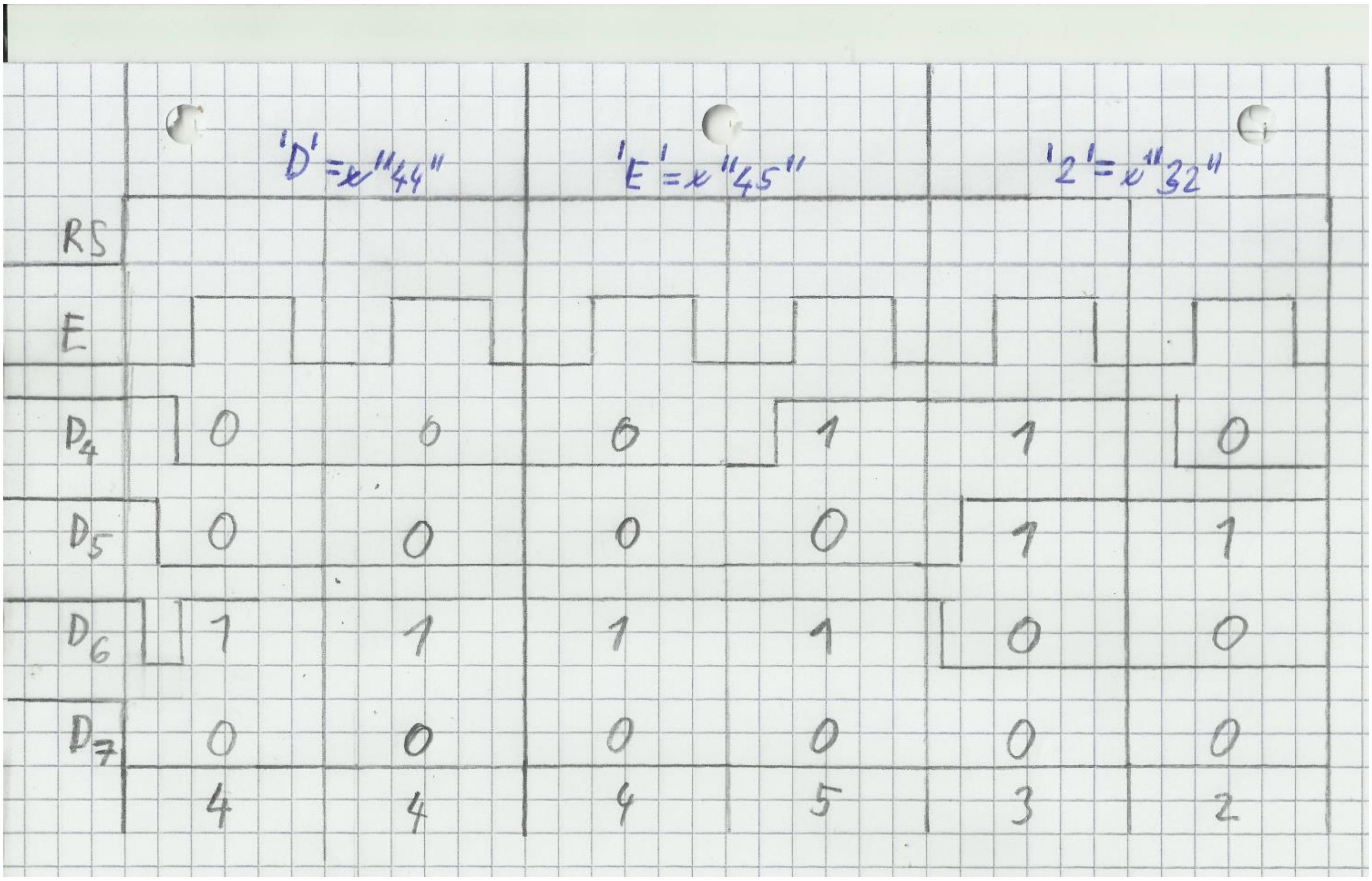
What is the ASCII table? What are the values for uppercase letters A to Z, lowercase letters a to z, and numbers 0 to 9 in this table?

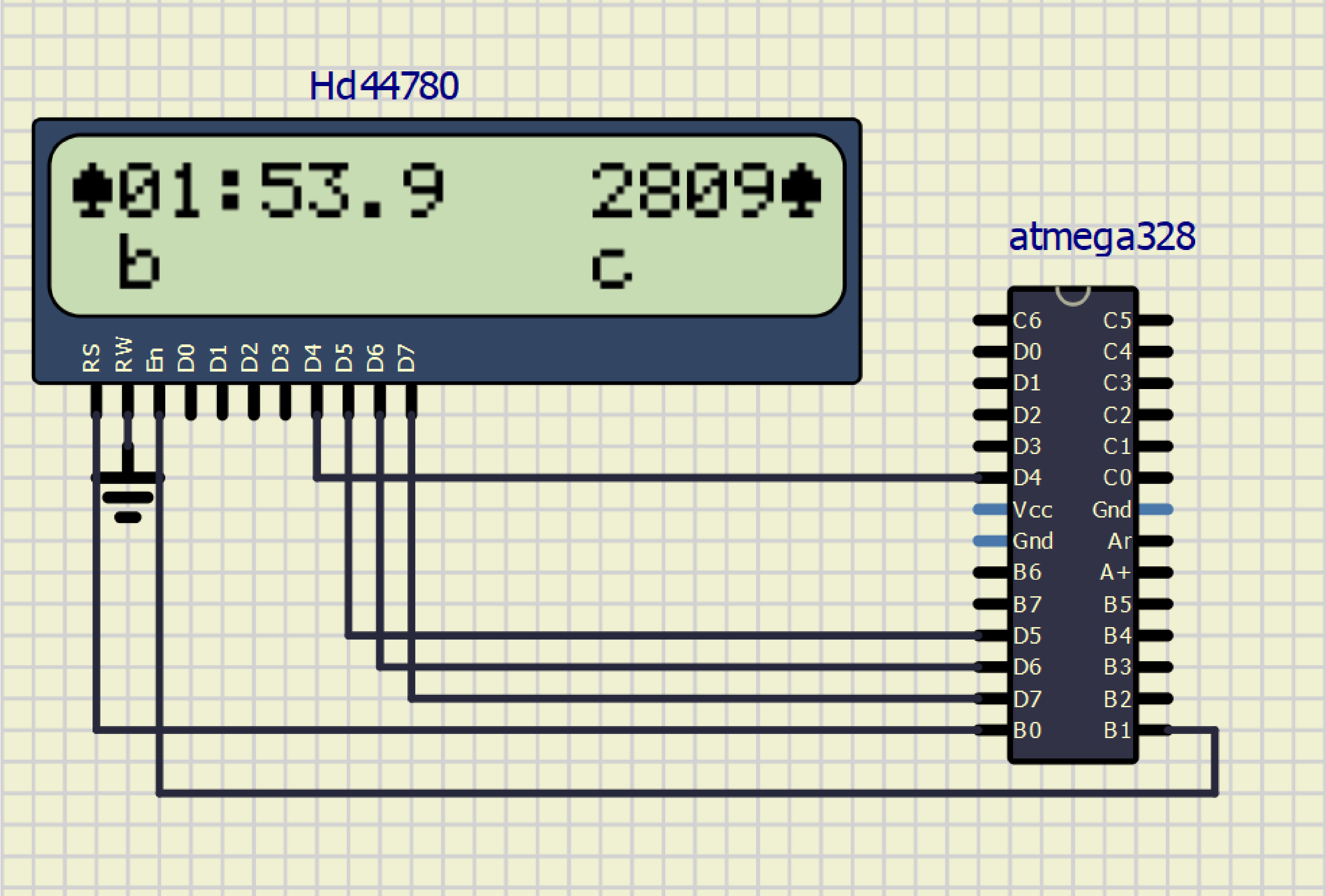
ASCII table contains ASCII code.

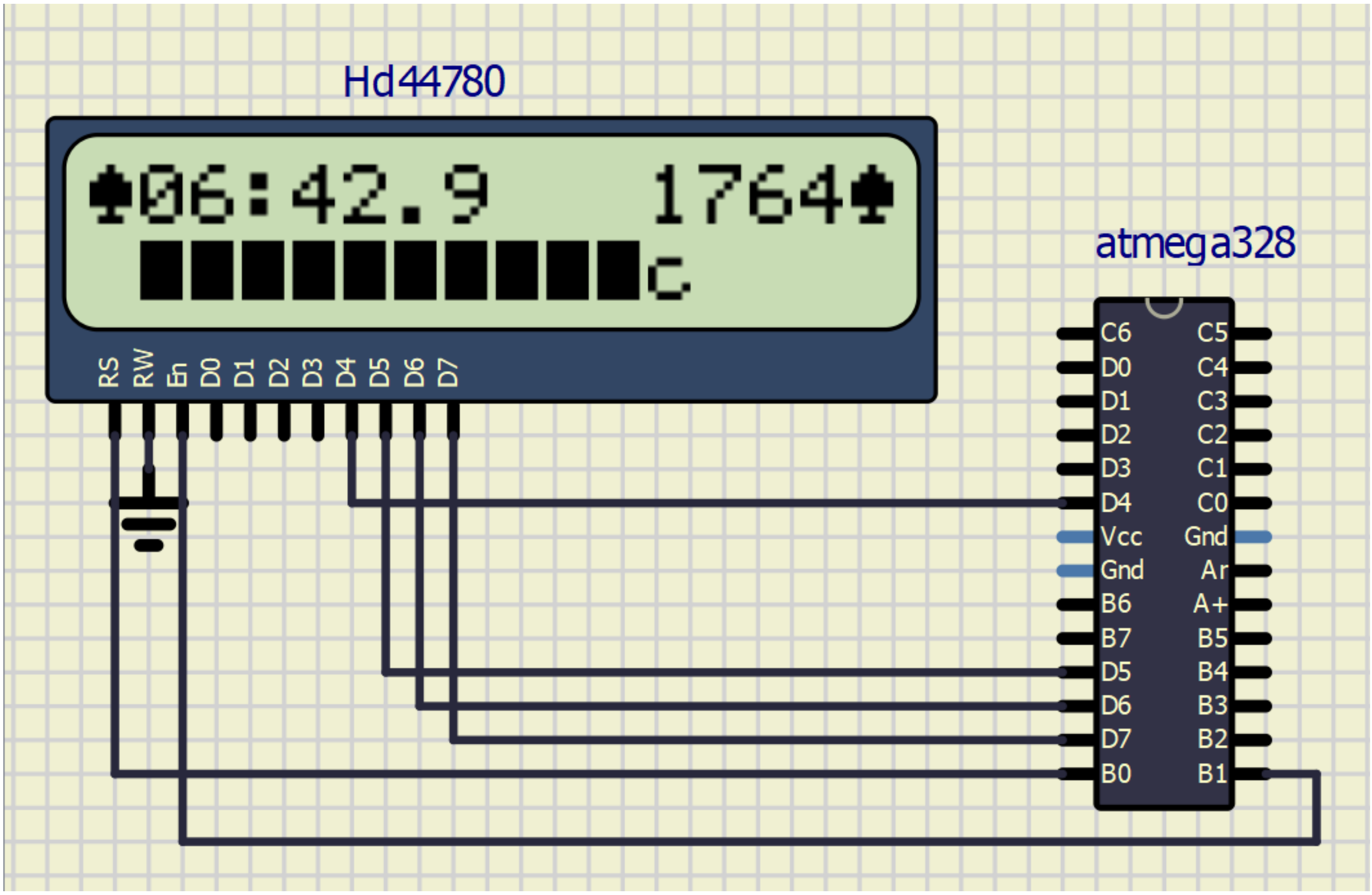
ASCII code is the numerical representation of a character.

Char	**→**	**Dec**
A-Z	**→**	**65-90**
a-z	**→**	**97-122**
0-9	**→**	**48-57**

Function name	Function parameters	Description	Example
lcd_init	uint8_t LCD_DISP_OFF	Display off	lcd_init(LCD_DISP_OFF);
	uint8_t LCD_DISP_ON	Display on, cursor off	lcd_init(LCD_DISP_ON);
	uint8_t LCD_DISP_ON_CURSOR	Display on, cursor on	lcd_init(LCD_DISP_ON_CURSOR);
	uint8_t LCD_DISP_ON_CURSOR_BLINK	Display on, cursor on flashing	lcd_init(LCD_DISP_OON CURSOR_BLINK);
lcd_clrscr	void	Clear display and set cursor to home position	lcd_clrscr();
lcd_gotoxy	uint8_t x, uint8_t y	Set cursor to specified position, x horizontal (left most), y vertical (first line) positions	lcd_gotoxy(x, y);
lcd_putc	char c	Display character at current cursor position.	lcd_putc(c);
lcd_puts	const char* s	Display string without auto linefeed.	lcd_puts(s);
lcd_command	uint8_t cmd	Send LCD controller instruction command.	lcd_command(cmd);
lcd_data	uint8_t data	Send data byte to LCD controller.	lcd_data(data);


'D'=x"44"
'E'=x"45"
'2'=x"32"
RS
E
D4
D5
D6
D7
0 0 0 1 1 0
0 0 0 0 1 1
1 1 1 1 0 0
0 0 0 0 0 0
4 4 4 5 3 2


Hd44780
♠01:53.9   2809♠
b   c
RS
RW
En
D0
D1
D2
D3
D4
D5
D6
D7
atmega328
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


Hd44780
♠06:42.9   1764♠
██████████c
RS
RW
En
D0
D1
D2
D3
D4
D5
D6
D7
atmega328
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

```c
/***********************************************************************
 * 
 * Stopwatch with LCD display output.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8 -bit Toolchain 3.6.2
 *
 * Copyright (c) 2017-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 * 
 **********************************************************************/

/* Includes ----------------------------------------------------------*/
#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

/* Variables ---------------------------------------------------------*/
// Custom character definition using https://omerk.github.io/lcdchargen/
uint8_t customChar[8*6] = {
    0b00100,
    0b01110,
    0b11111,
    0b11111,
    0b11111,
    0b00100,
    0b01110,
    0b11111,
    
    //progressBar1
    0b10000,
    0b10000,
    0b10000,
    0b10000,
    0b10000,
    0b10000,
    0b10000,
    0b10000,
    
    //progressBar2
    0b11000,
    0b11000,
    0b11000,
    0b11000,
    0b11000,
    0b11000,
    0b11000,
    0b11000,
    
    //progressBar3
    0b11100,
    0b11100,
    0b11100,
    0b11100,
    0b11100,
    0b11100,
    0b11100,
    0b11100,

    //progressBar4
    0b11110,
    0b11110,
    0b11110,
    0b11110,
    0b11110,
    0b11110,
    0b11110,
    0b11110,

    //progressBar5
    0b11111,
    0b11111,
    0b11111,
    0b11111,
    0b11111,
    0b11111,
    0b11111,
    0b11111
};

uint8_t running_text[] = "   I like Digital electronics!\n";

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Update stopwatch
 * value on LCD display when 8-bit Timer/Counter2 overflows.
 */
int main(void)
{
    // Initialize LCD display
    lcd_init(LCD_DISP_ON);

    // Set pointer to beginning of CGRAM memory
    lcd_command(1 << LCD_CGRAM);
    for (uint8_t i = 0; i < 8*6; i++)
    {
        // Store all new chars to memory line by line
        lcd_data(customChar[i]);
    }
    // Set DDRAM address
    lcd_command(1 << LCD_DDRAM);

    // Display custom characters
    lcd_putc(0);
    lcd_gotoxy(15, 0);
    lcd_putc(0);

    // Put string(s) at LCD display
    lcd_gotoxy(1, 0);
    lcd_puts("00:00.0");
    lcd_gotoxy(11, 0);
    lcd_putc('a');
    lcd_gotoxy(1, 1);
    lcd_putc('b');
    lcd_gotoxy(11, 1);
    lcd_putc('c');

    // Configure 8-bit Timer/Counter2 for Stopwatch
    // Enable interrupt and set the overflow prescaler to 4 ms
    TIM2_overflow_4ms();
    TIM2_overflow_interrupt_enable();


    // Configure 8-bit Timer/Counter0 for Stopwatch
    // Enable interrupt and set the overflow prescaler to 4 ms
    TIM0_overflow_4ms();
    TIM0_overflow_interrupt_enable();


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

    // Infinite loop
    while (1)
    {
        /* Empty loop. All subsequent operations are performed exclusively
         * inside interrupt service routines ISRs */
    }

    // Will never reach this
    return 0;
}

/* Interrupt service routines ----------------------------------------*/
/**
 * ISR starts when Timer/Counter0 overflows.
 * 5 x 50 x 4ms = 1s
 * (5 x 4ms = 20ms) for one part of bar
 */
ISR(TIMER0_OVF_vect)
{
    static uint8_t number_of_overflows = 0;
    static uint8_t nth_cell = 0;    // nth cell bar
    static uint8_t part_of_bar = 0; // part of bar

    number_of_overflows++;
    if (number_of_overflows >= 5)
    {
        // Do this every 5 x 4 ms = 20 ms
        number_of_overflows = 0;

        part_of_bar++;
        if(part_of_bar >= 5)
        {
            part_of_bar = 0;
            nth_cell++;
            if(nth_cell > 9)
            {
                nth_cell = 0;
                lcd_gotoxy(1, 1);
                lcd_puts("          "); //reset
            }
        }
        lcd_gotoxy(nth_cell+1, 1);  //position of bar
        lcd_putc(part_of_bar+1);    //part of bar from memory
    }
}


/**
 * ISR starts when Timer/Counter2 overflows.
 * 10 x 25 x 4ms = 1s
 * (25 x 4 ms = 100 ms) for one tenth of a second
 */
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 uint32_t secs_secpow = 0;        // Seconds to the second power
    static uint8_t mins = 0;        // Minutes
    char lcd_string[5] = "  ";      // String for converting numbers by itoa()


    number_of_overflows++;
    if (number_of_overflows > 25)
    {
        // Do this every 25 x 4 ms = 100 ms
        number_of_overflows = 0;

        // WRITE YOUR CODE HERE
        tens++;
        if(tens > 9){ //10 x 0.1s = 1.0s
            tens = 0;
            secs++;
        }
        if(secs > 59){ //60 x 1s = 1min 0s
            secs = 0;
            mins++;
        }
        if(mins > 59) //60 x 1min = (1h) 0min
            mins = 0;

        secs_secpow = secs*secs;


        // tenths of seconds
        itoa(tens, lcd_string, 10); // Convert decimal value to string
        lcd_gotoxy(7, 0);
        lcd_puts(lcd_string);

        // seconds
        if(secs < 10){
            lcd_gotoxy(4, 0);
            lcd_putc('0');
            lcd_gotoxy(5, 0);
        }else
            lcd_gotoxy(4, 0);
        itoa(secs, lcd_string, 10); // Convert decimal value to string
        lcd_puts(lcd_string);

        // minutes
        if(mins < 10){
            lcd_gotoxy(1, 0);
            lcd_putc('0');
            lcd_gotoxy(2, 0);
        }else
            lcd_gotoxy(1, 0);
        itoa(mins, lcd_string, 10); // Convert decimal value to string
        lcd_puts(lcd_string);

        // seconds to the second power
        if(secs == 0){
            lcd_gotoxy(11, 0);
            lcd_puts("    ");   //reset
        }
        lcd_gotoxy(11, 0);
        itoa(secs_secpow, lcd_string, 10); // Convert decimal value to string
        lcd_puts(lcd_string);
    }
}
```