Digital Electronics 2 (Brno University of Technology)

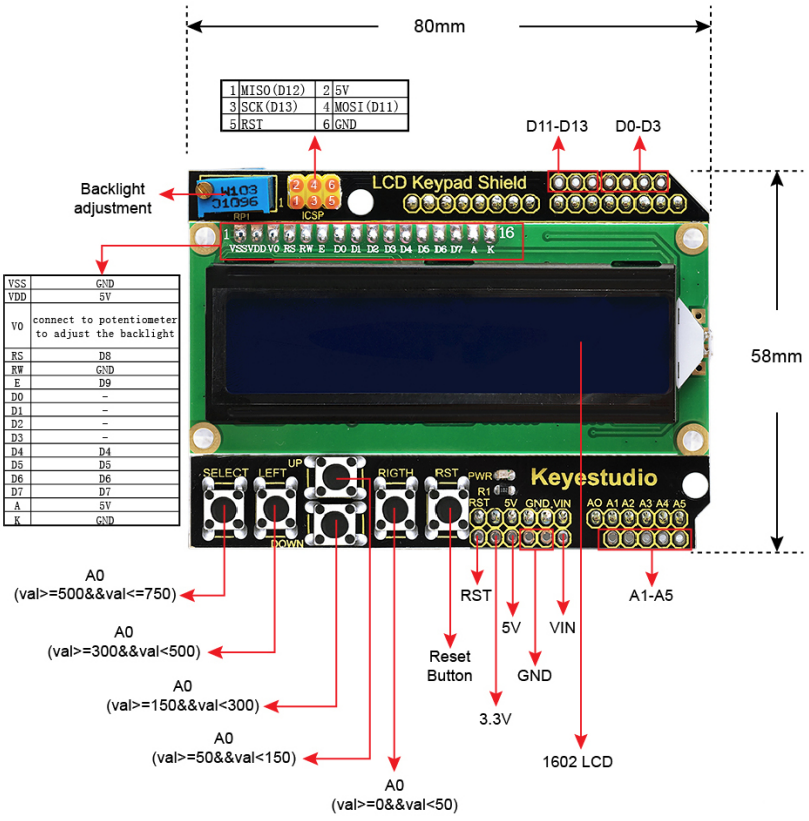
Assignment of lab #6

2019
October

Synchronize Git and create a new project

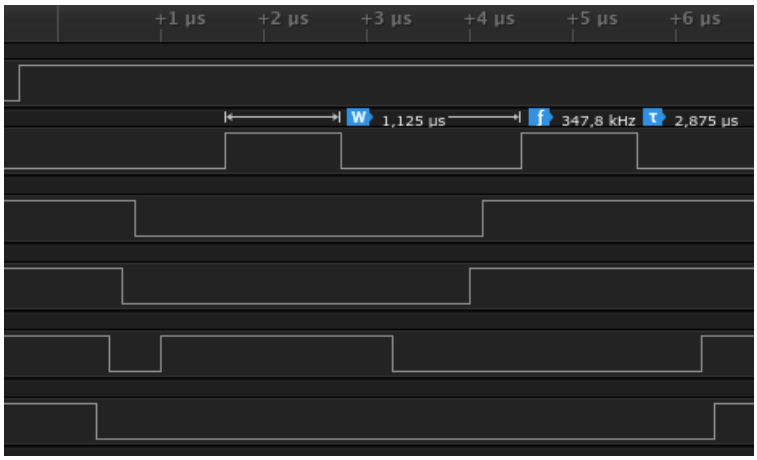
- In VS Code open your Digital-electronics-2 working directory and synchronize the contents of a repository with a single git command (`git pull`) or a sequence of two commands (`git fetch` followed by `git merge`).
- Create a new folder `project/06-lcd` and copy three files from the last project: `main.c`, `Makefile`, `README.md`.

LCD Keypad Shield



- In `docs/hw/` folder, see schematic of LCD Keypad Shield and find out the connection of LCD display. Which data and control signals are used? What is the meaning of these signals?

- Let the following image shows the communication between ATmega328P and LCD display in 4-bit mode. The order of the displayed signals is as follows: RS, E, $D4$, $D5$, $D6$, and $D7$. How does control circuit HD44780 understand the sequence of these signals?



LCD library

In the lab, we are using LCD library for HD44780 based LCDs developed by Peter Fleury [1]

- Use online manual of LCD library and add the input parameters and description of the functions in the following table.

Function	Parameter(s)	Description
`lcd_init`	`LCD_DISP_OFF`	Display off
`lcd_clrscr`		
`lcd_home`		
`lcd_gotoxy`		
`lcd_putc`		
`lcd_puts`		

- Use template `project/06-lcd/main.c` [2] and test all functions mentioned in table above. Comment/uncomment library source files you are using within the list of compiled files in `06-lcd/Makefile`.

Decimal counter

- According to the listing bellow, verify how you can include the value of a variable in text string and then display it on LCD. Display variable value in decimal, binary, and hexadecimal. What is the parameters of standard C function `itoa()`?

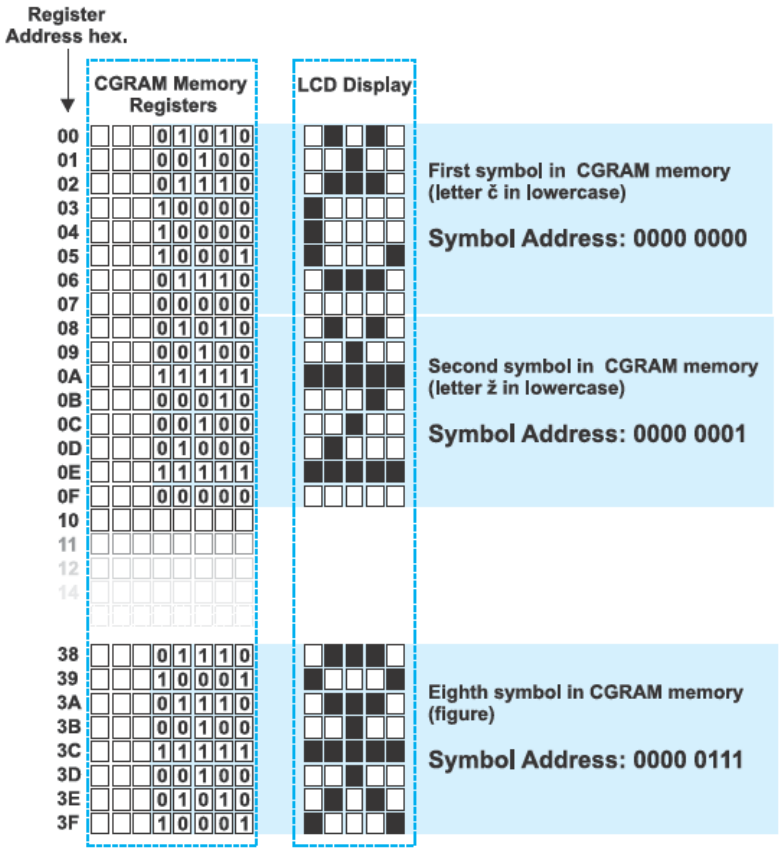
```
uint8_t value = 31;
char lcd_string[3];
...

itoa(value, lcd_string, 16);
lcd_putc('$');
lcd_puts(lcd_string);
```

- Configure Timer1 clock source, enable its overflow interrupt, create a decimal counter from 0 to 255, and show the value on LCD display.

User-defined symbols

- User-defined symbols are represented by eight bytes (lines) and they are stored in the beginning of CGRAM display memory according to the following figure [3].



- Design at least two user characters, store them in the display memory according to the following code and display them on LCD.

```
/* Global variables ---------------------------------------------------------*/
uint8_t lcd_user_symbols[8*2] = {0x0a, ...};
...

    // Set pointer to beginning of CG RAM memory
    lcd_command(1<<LCD_CGRAM);
    // Store new characters line by line
    lcd_data(lcd_user_symbols[0]);
    ...
    // Clear display and set cursor to home position
    lcd_clrscr();
    ...
    // Display first user-defined character
    lcd_putc(0x00);
```

Clean folder and synchronize git

- Remove all binaries and object files from the working directory. Then use git commands [4], commit all modified/created files to your local repository and push them to remote repository. Use VS Code editor options to perform these operations.

Ideas for other tasks

- Create a bar graph at LCD. Let the bar state corresponds to decimal counter value, similar to the following figure.

- Complete `README.md` file.

References

[1] AVR-GCC Source Examples,
`http://homepage.hispeed.ch/peterfleury/avr-software.html`

[2] AVR repository for DE2 course at Brno University of Technology,
`https://github.com/tomas-fryza/Digital-electronics-2`

[3] Book: PIC Microcontrollers - Programming in C,
https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c/additional-components

[4] Git Commands,
https://github.com/joshnh/Git-Commands