Lab 7: Using the LCD Display

Submit numDisplay.asm from your project at the end of your lab.

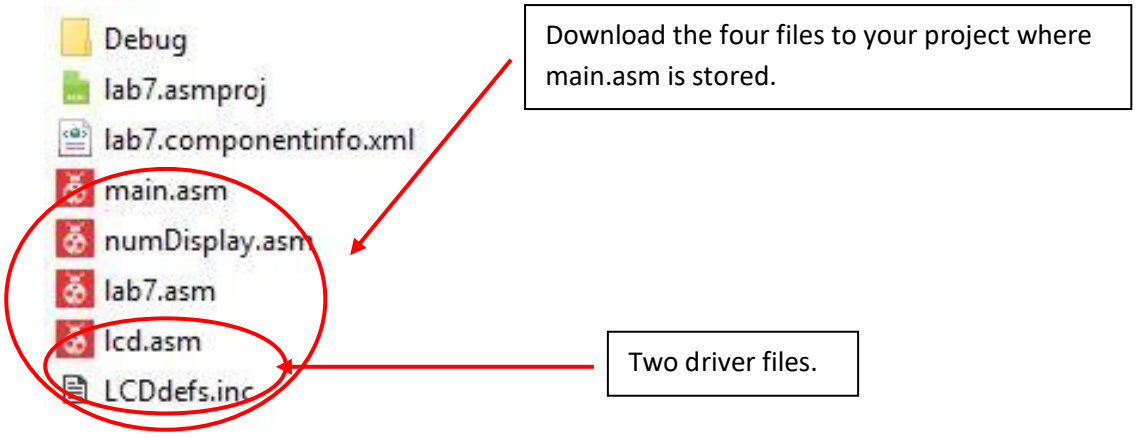
I. LCD Display

In Lab 6, we implemented the *unsigned int strLength(char* str)* subroutine, but we had to use the debugger and manually examine the memory in order to see the result. It would be much easier if we could display the result on an LCD[1] display. The one that we use has its own, built-in controller and driver chips and it is capable of displaying some ASCII text characters and some special symbols. Our LCD display consists of two rows, each can display up to 16 characters.

Let's see how we can interface with the LCD. Create a new project named lab7, and then download the two HD44780 LCD Driver files into the newly created project directory, which already contains the main.asm file:

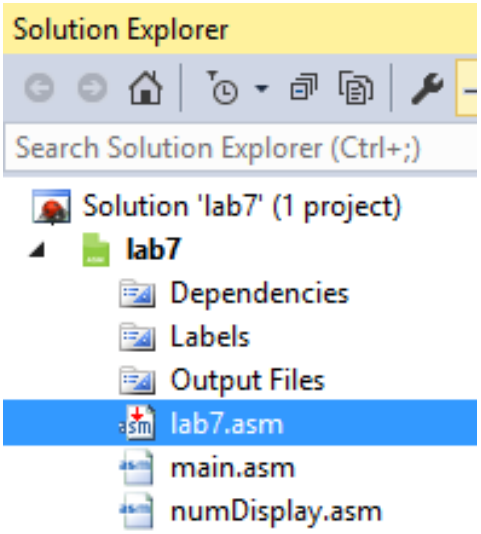
LCDdefs.inc (LCD driver)
lcd.asm (LCD driver)

Download *lab7.asm* and *numDisplay.asm* into the same place as above. The directory of your project should look like this:

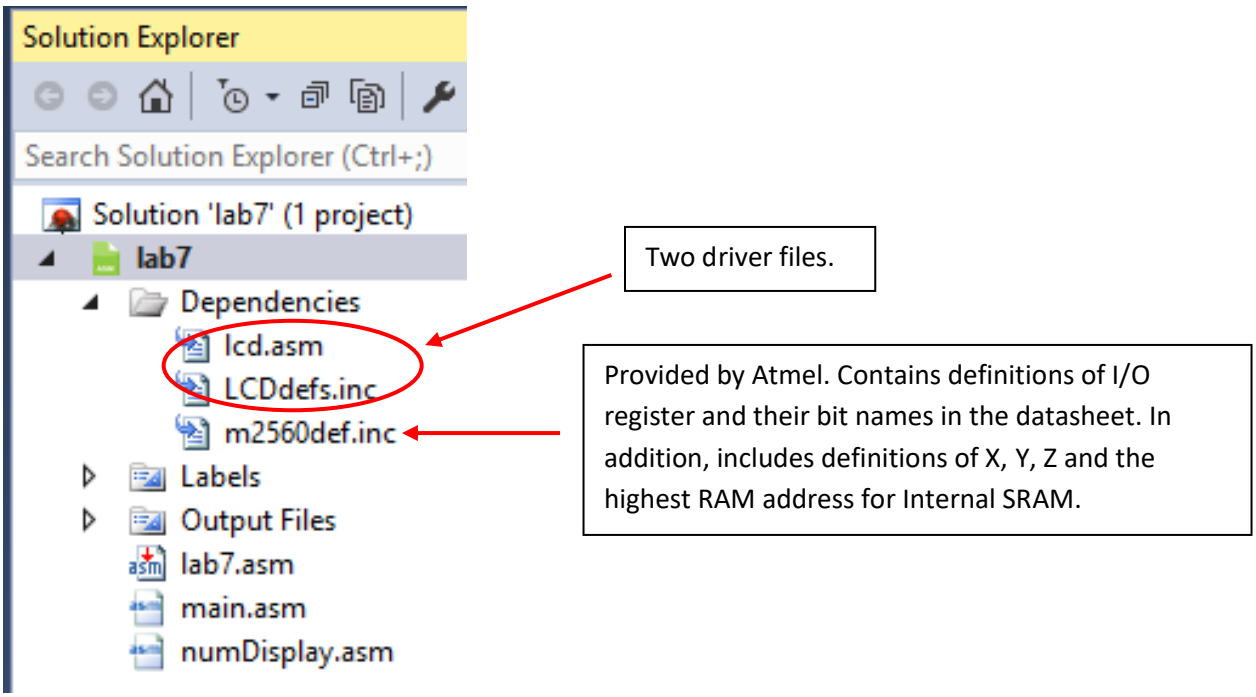


[1] LCD stands for Liquid Crystal Display.

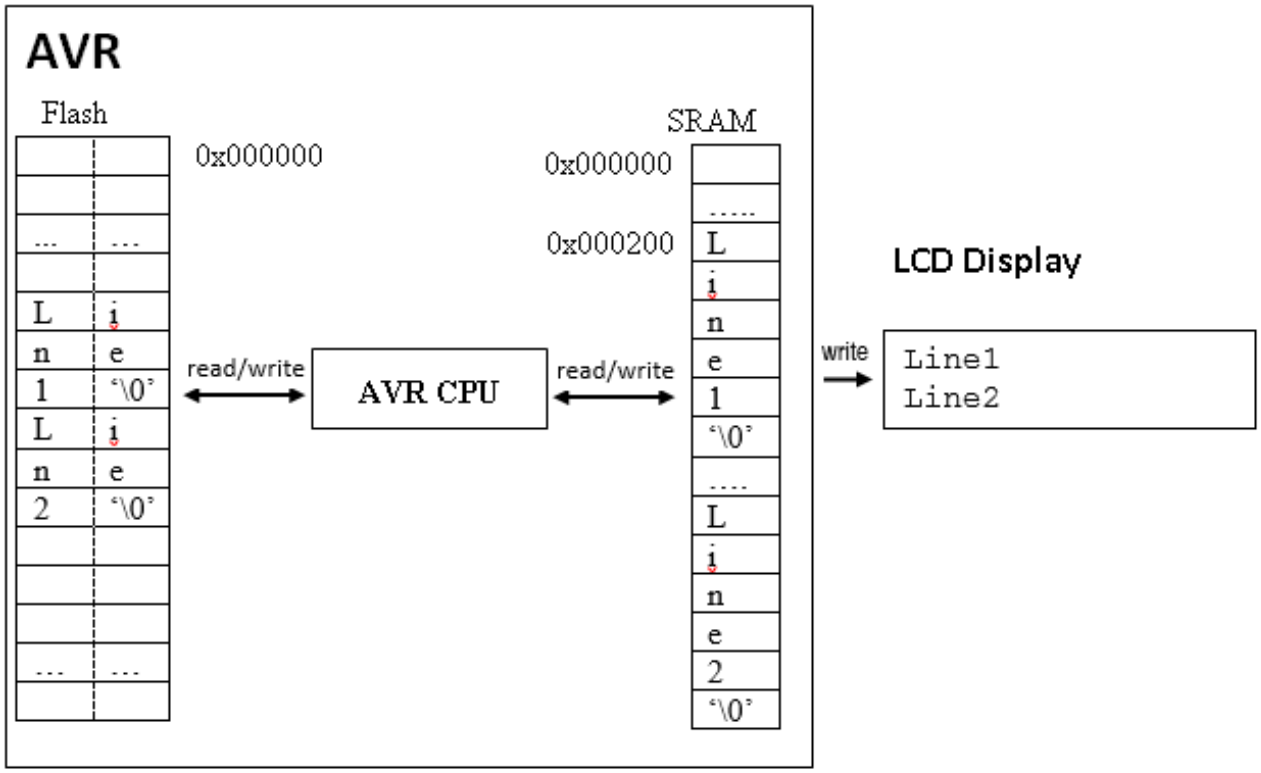
In the *Solution Explorer* of the *Atmel Studio 7.0* project, add *lab7.asm* and *numDisplay.asm* to your project and set *lab7.asm* as entry file. The project Solution Explorer will then look like this:



Build the project. In *Solution Explorer* of the *Atmel Studio 7.0* project, expand the *Dependencies* folder. The *Solution Explorer* should look like this:



The following diagram demonstrates the relationships between the program (Flash) memory, data (SRAM) memory, the AVR CPU of the AVR board and the LCD display. In lab7.asm, the two strings "Line1" and "Line2" are copied from the program memory to the data memory. Then the two strings are displayed from the data memory to the LCD display. Notice that the program memory is word (2 bytes per word) addressable and the data memory is byte addressable.



Note that *str_init()* function is defined in *lcd.asm*. It copies a string from the program memory (flash) to the data memory (SRAM), like what we did in previous labs and assignments. "SP_OFFSET" is defined in LCDdefs.inc (line 42 and 43). It is the number of bytes of the return address for a subroutine call, which is 3 bytes in the case of ATMEGA 2560.

Open lab7.asm. Read and understand the code.

In the *display_strings* subroutine, the algorithm is doing the following:

Clear the LCD display (*lcd_clr*)
Move the cursor to the desired location (row 0, column 0) (2X16 display) (*lcd_gotoxy*)
Display "Line1" stored in SDRAM (*lcd_puts*)
Move the cursor to the desired location (row 1, column 0) (2X16 display) (*lcd_gotoxy*)
Display "Line2" stored in SDRAM (*lcd_puts*)

II. Exercises.

Set *numDisplay.asm* as entry file. Finish implementing the main program to display the number. Modify the subroutine *void int_to_string()* such that it accepts two parameters passed on the stack, which are the number to be displayed and the address where to store the string. The string is to be stored in c-string format (zero-terminated). It is up to you whether to make the string left-justified or right-justified within the allocated space. The function can assume that enough space is available at the address that is passed to it. If time permits, modify the function *int_to_string()* to accept a 24-bit number and produce a maximum of 7 least significant digits of the given number.

The C equivalent code is available in *divide.c* file:

```c
#include <stdio.h>
/*division, using repeated subtractions.
Convert integer 123 to a character array: '1' '2' '3' '\0'
The last character '\0' indicates the end of the character array.
*/
int main()
{
    int dividend=123;
    int quotient=0;
    int divisor=10;
    char num_in_char_array[4];
    num_in_char_array[3]='\0';
    int i=2;
    do{
        while(dividend>=divisor)
        {
            quotient++;
            dividend -= divisor;
        }
            num_in_char_array[i--]=dividend+'0';
            dividend=quotient;
            quotient=0;
    }while(dividend>=divisor);
    num_in_char_array[i]=dividend+'0';
    printf ("%s\n",num_in_char_array);
    return 0;
}
```

You may download *divide.c*, compile and run the code from the DOS command window:

```
H:\201809\230\lab7>gcc -o divide -Wall divide.c

H:\201809\230\lab7>divide
123
```

Submit numDisplay.asm from your project at the end of your lab.