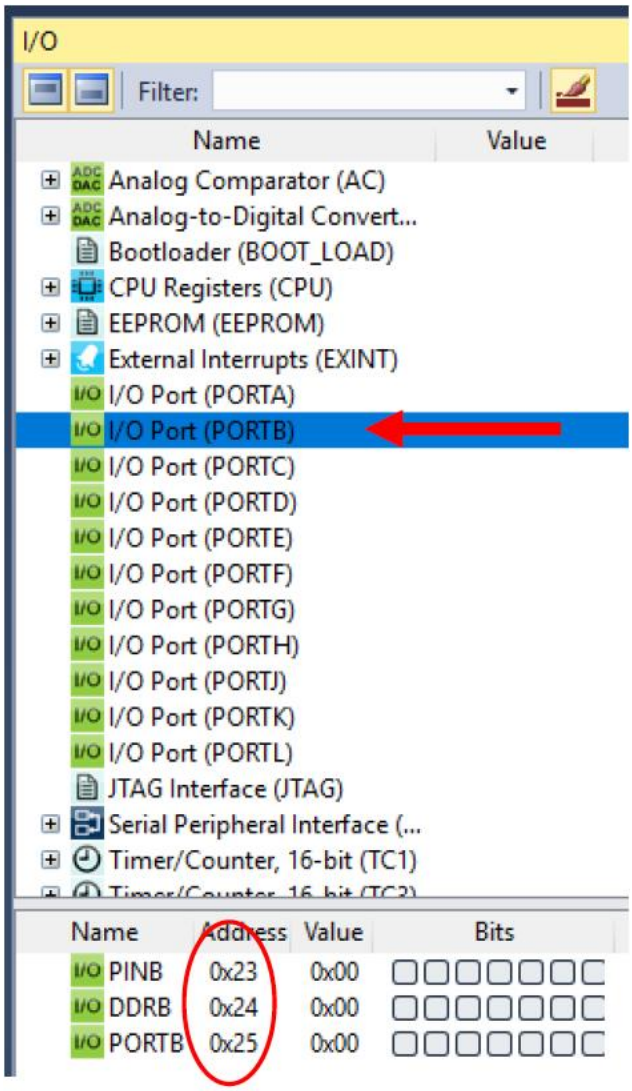
Lab 3 Introduction to MEGA 2560 Board and I/O Instructions

Submit main.asm from the lab3 project at the end of your lab, and not your project file.

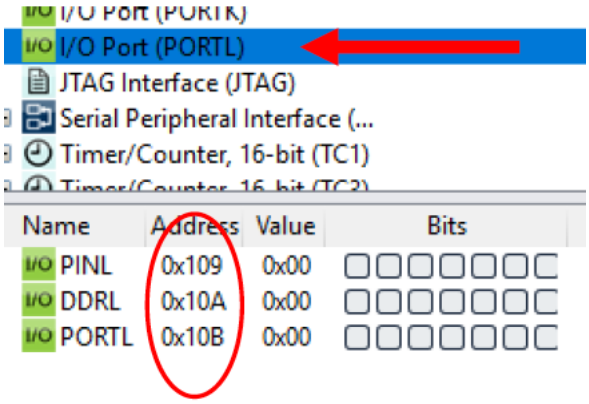
I. I/O Instructions

In the AVR MEGA 2560 microcontroller, each Input/Output (I/O) port has three associated special purpose registers, the data direction register (DDRx), output register (PORTx), and an input register (PINx). These registers correspond to addresses in the data space accessible by the processor (the first 0x200 bytes in SRAM).

Create a new project named lab3 and build the program. From the menu, click on "Debug" -> "Start Debugging and Break". On the right-hand side, observe the I/O View. To see the memory address of PORTB and PORTL, click on PORTB or PORTL:



The memory address of PORTL:

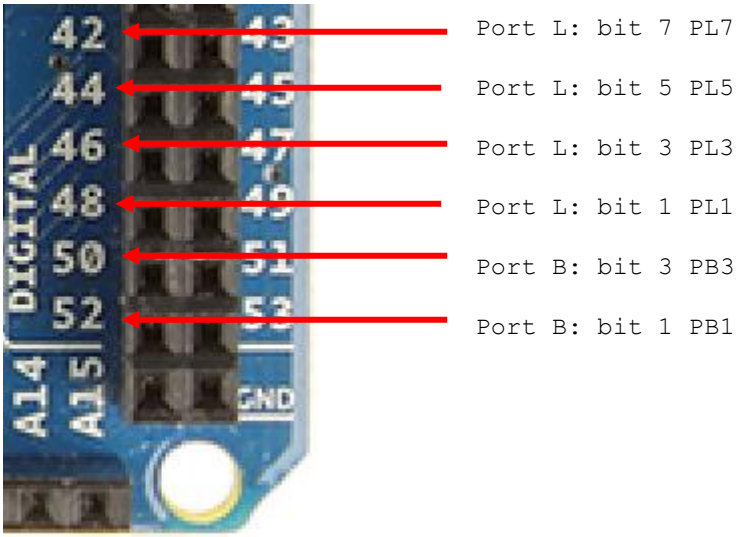


Some ports are mapped to an address less than or equal to 0x3F. For example, PORTB is mapped to 0x25. For these ports, you may use IN/OUT instructions to transfer data between registers and SRAM (data memory). However, some ports, such as PORTL, have a mapping to an address which is greater than 0x3F and they cannot be accessed using the IN/OUT instructions. What can we do in this case? We can use LDS and STS instructions instead.

In AVR, ports A to G use Port Mapped I/O (separate addresses from memory) and Ports H to K use Memory Mapped I/O (usage is similar to any memory location). Port Mapped addresses use IN/OUT instructions while Memory Mapped ones use LDS and STS.

II. Pins and Ports

The six LEDs are associated with six pins and the pins are mapped to some bits of two ports, PORTB and PORTL:



In the project named lab3 which you created above, write a small program to turn a specific LED on (the code is provided below).

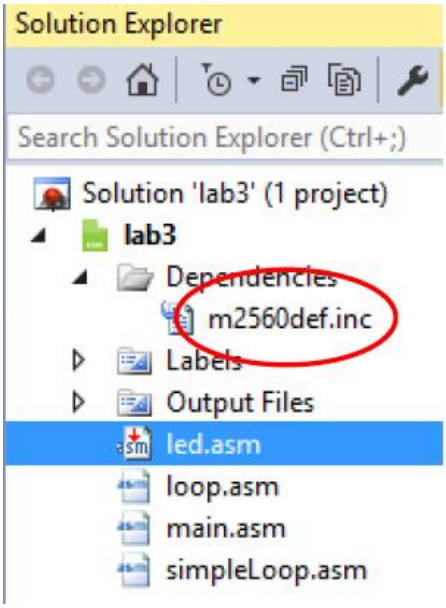
```
.cseg
    ldi r16, 0xFF
    sts DDRL, r16 ;PORTL and PORTB as output
    out DDRB, r16

    ;set the top and the bottom lights on
    ldi r16, 0b10000000
    sts PORTL, r16
    ldi r16, 0b00000010
    out PORTB, r16

done:jmp done
```

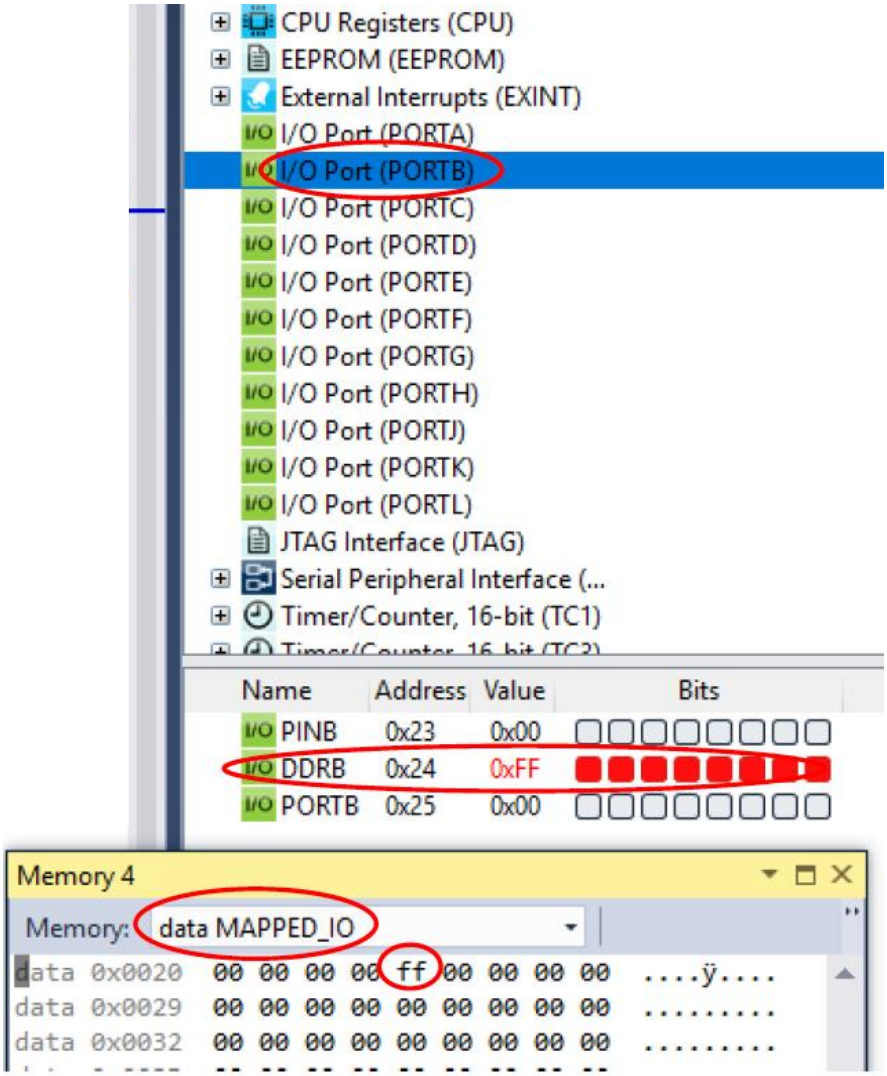
The memory addresses of DDRL, DDRB, PORTL, and PORTB are defined in a file named "m2560def.inc". To read the file go to the "Solution Explorer" and click on "Dependencies" -> "m2560def.inc" (see the screenshot below).

Note that the I/O registers - 0x10B and 0x10A - are specified in hexadecimal numbers. The two I/O registers are given names, PORTL and DDRL, by using the .equ directives to represent their memory addresses in data memory, which allows us to refer to the I/O registers by names instead of numbers when we program.



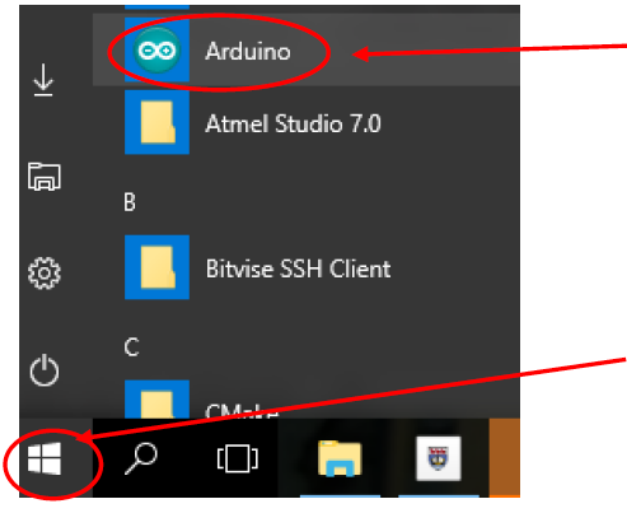
III. Build and upload the .hex file to the board:

Build and run the program above in Atmel Studio. Observe the changes of the registers and the I/O registers.

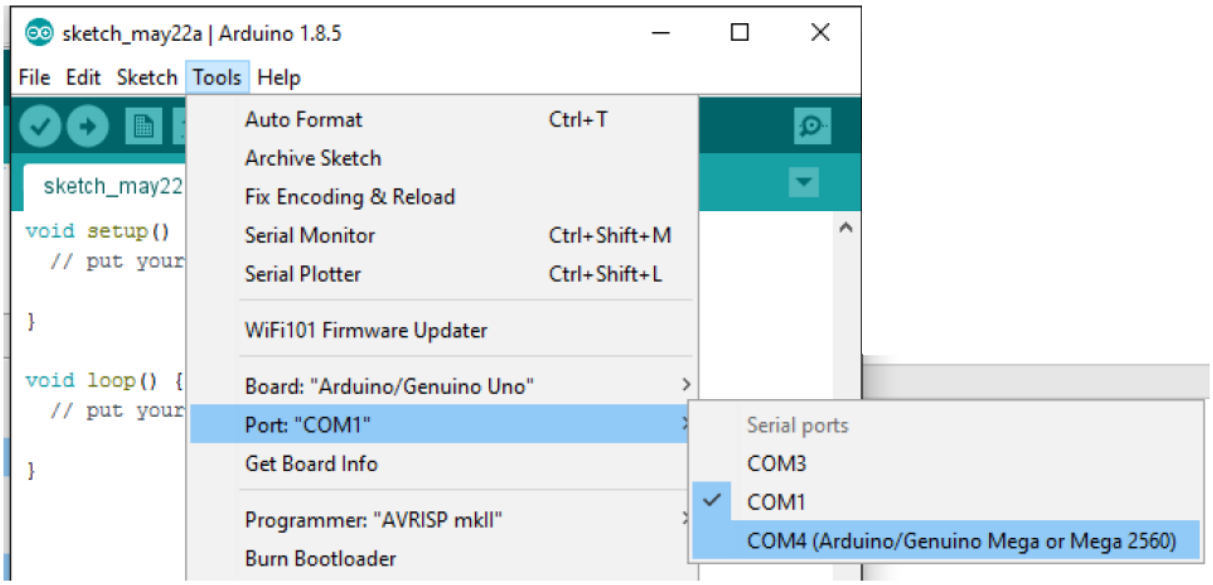


Determine which COM port the lab AVR board is connected to. The port can be different on each machine each time. You can check the COM port number by launching the Arduino IDE, see the instructions below.

1. Launch Arduino IDE by clicking the Start button, then Arduino.



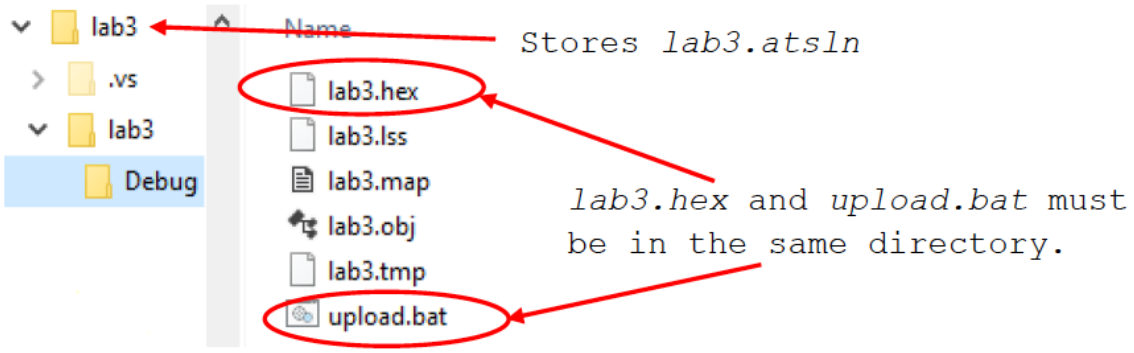
2. In the Arduino IDE program, from the menu at the top select “Tools”, then “Port”, then see which COM port lists “Arduino/Genuino Mega or Mega 2560” next to it. In the screenshot below, COM4 is the one.



Upload the lab3.hex file to the board by typing the following command (as one long line) at the command window:
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe" -C "C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -p atmega2560 -c wiring -P COM4 -b 115200 -D -F -U flash:w:lab3.hex

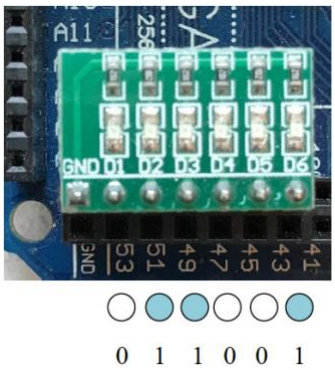
Instead of typing such a long command, you can use upload.bat file by double-clicking it. This file contains the abovementioned command. Note that you may need to modify the contents of upload.bat first. The relevant components are highlighted above in red.

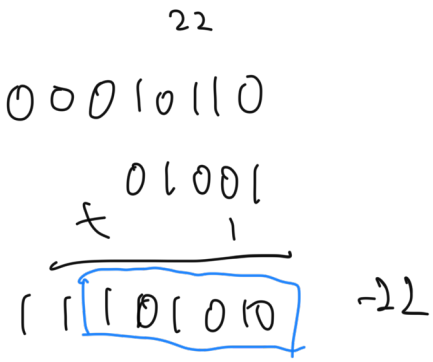
Additionally, upload.bat file must be located in the same directory (folder) as the .hex file that you wish to upload to the AVR board. To find the location of lab3.hex file, refer to the following screenshot:



IV. Exercises:

1. You can use the LED lights to display a binary number. If you tilt your LED lights (or your head) by 90 degrees, each can represent a bit of that binary number (see the picture below). Let pin 52 represent the most significant bit and pin 42 represent the least significant bit.





Change the values sent to PORTB and PORTL via r16 to reflect the picture above. Rebuild the program and upload it to the AVR board to verify that the correct LEDs are on. Now display -22 using two's complement representation.

2. Modify your code to make the LED lights blink by turning them on for a while and then off for a while. Hint: use nested loops with NOP (no operation) to use up **eight million** cycles, then turn the lights on or off (XOR gate?). Remember, you can count the exact number of cycles your program requires, refer to the AVR Instruction Set to determine each operation's cycle count. For instance, DEC and BRNE require 2n+3, where n is the starting number. Adding a NOP will make it 3n+3. Since the ATMEGA2560 processor runs at 16MHz clock speed, if your nested loop uses 8M cycles, then the lights would turn on for half a second, then turn off for half a second. Build and upload your code and verify that it runs correctly.

Submit main.asm from the lab3 project at the end of your lab, and not your project file.