Lab 8: C programming with AVR Studio 4

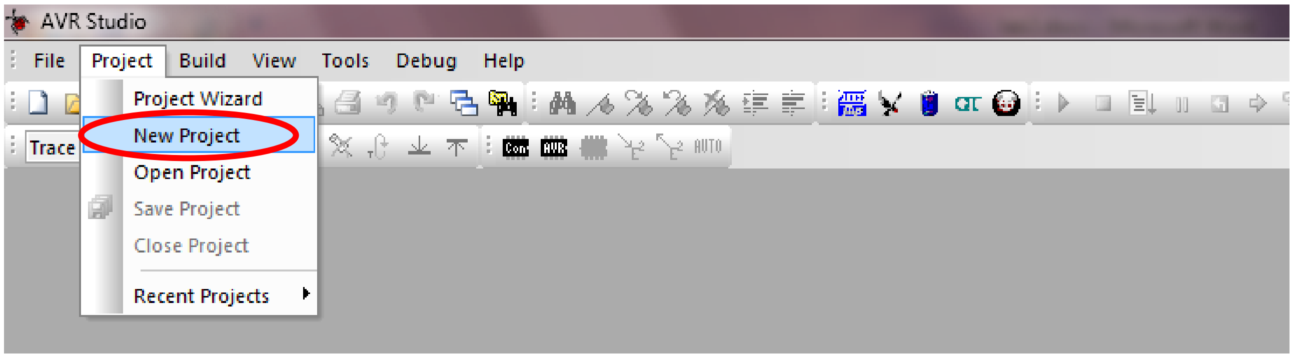
The goal of this lab is to introduce you to “C” programming language. We will repeat the concepts from previous labs and use “C” instead of assembly programming. There are two parts to this lab. The first one introduces you to a blink program and the second one is the LCD display program. **Submit lab8.c at the end of your lab.**

PART1: LED blinking

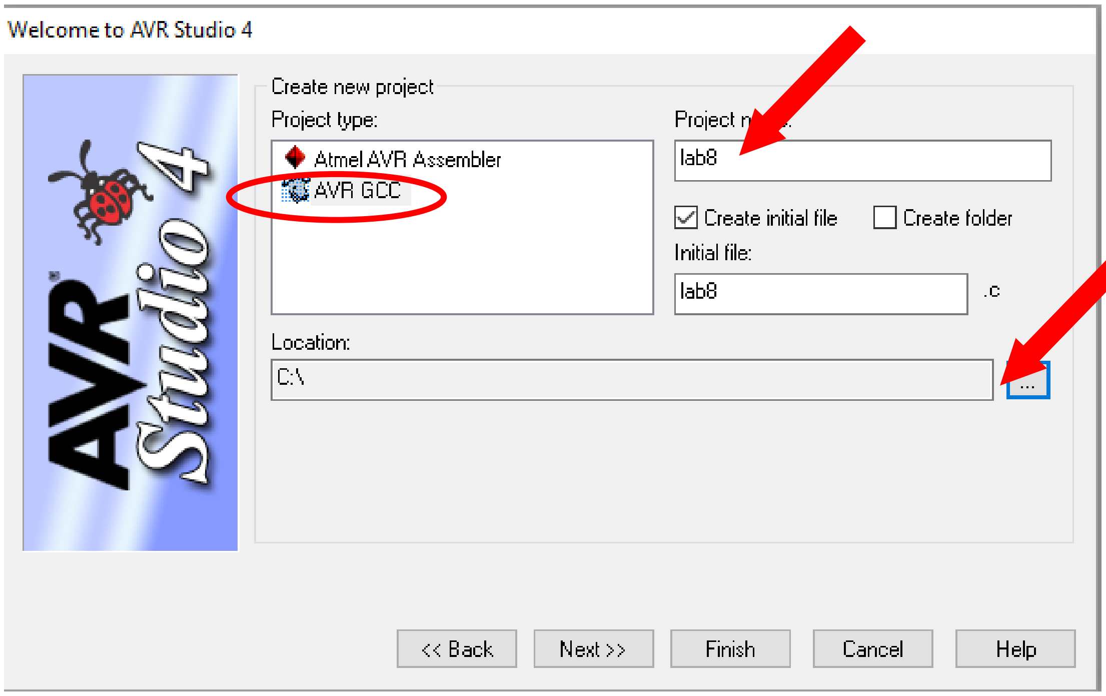
I. Create a C project in AVR Studio 4

Start the AVR Studio 4.

- Select “New Project”.

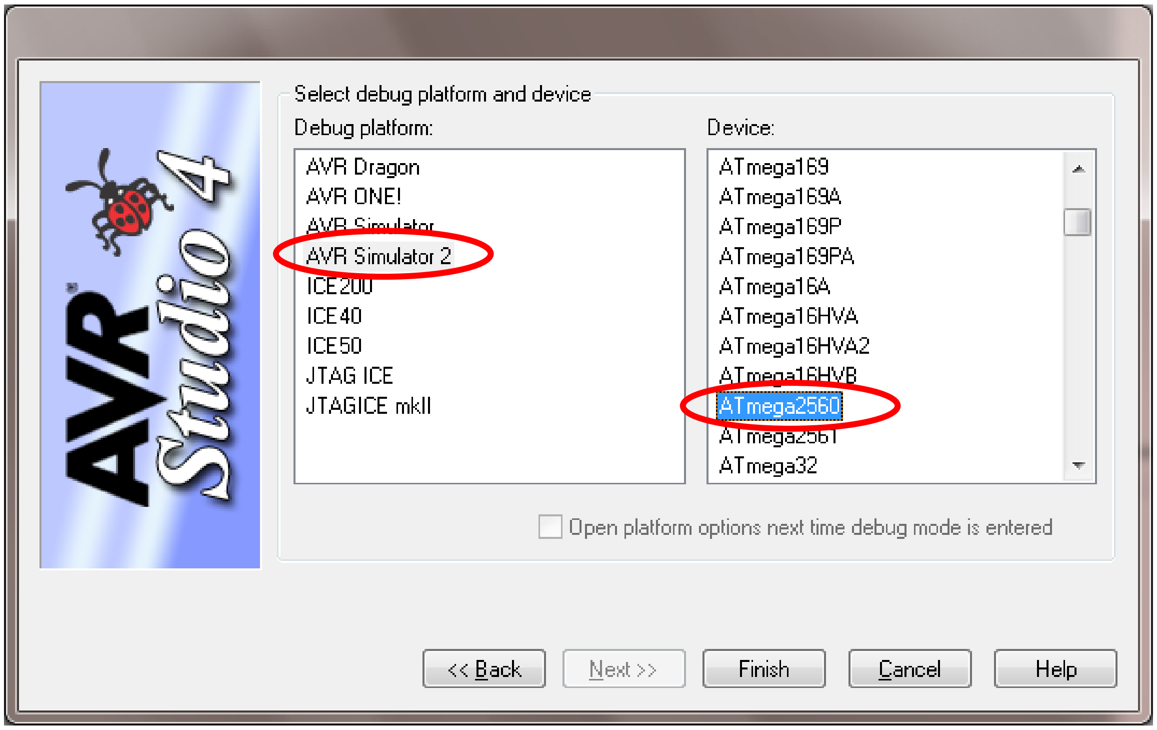


- Select “Project Type” → AVR GCC, name the project "lab8", click "…" button to choose a location (H drive)



- Go to → Next.

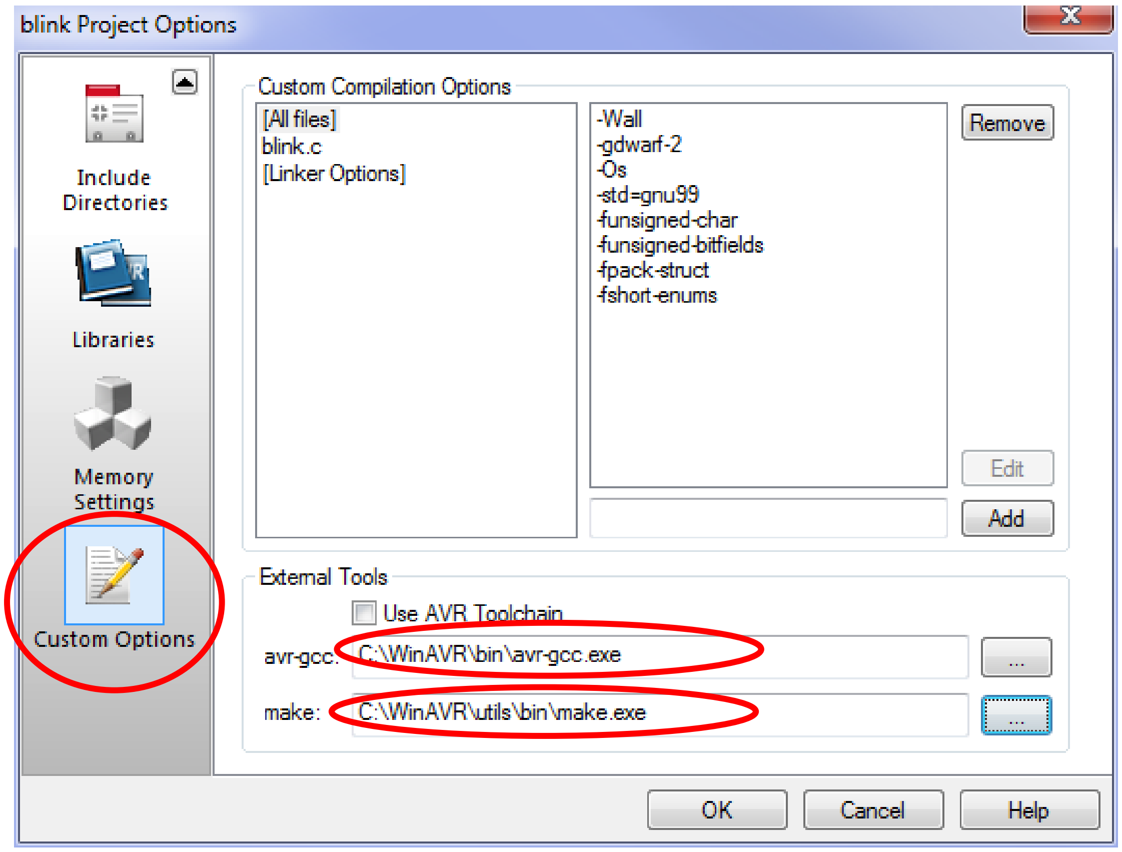
- Select a platform: Debug Platform → AVR Simulator 2
- Select a Device: Device → ATmega2560. Click the "Finish" button.



II. Compile C project

After creating a C project, it opens up the project space. There are issues with the **avr-gcc** compiler within the AVR Studio. Hence we need to replace this with a **gcc** compiler located in the WinAVR directory. **This step needs to be done for every C project you create.**

- Go to Project → Configuration Options
- Go to the last item "Custom Options" in the left plane.
- Uncheck "Use AVR Toolchain" under External Tools at the bottom
- Select the button for "avr-gcc" to point to avr-gcc.exe in the folder "C:\WinAVR\bin\avr-gcc.exe"
- Select the button for "make" to point to make.exe in the folder "C:\WinAVR\utils\bin\make.exe"



Custom Compilation Options

[All files]
blink.c
[Linker Options]

-Wall
-gdwarf-2
-Os
-std=gnu99
-funsigned-char
-funsigned-bitfields
-fpack-struct
-fshort-enums

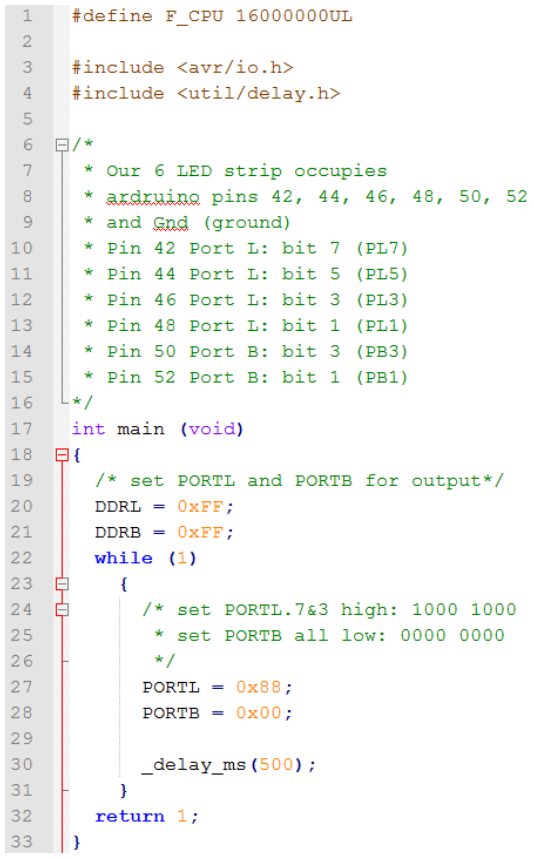
External Tools

[ ] Use AVR Toolchain

avr-gcc: C:\WinAVR\bin\avr-gcc.exe

make: C:\WinAVR\utils\bin\make.exe

- Copy the contents of lab8.c provided to you in the Lab8 folder on connex
- Read the code and understand

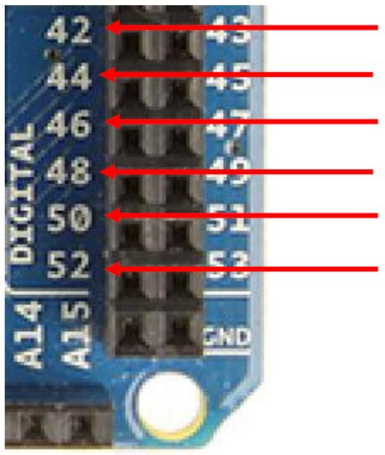


```
#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>

/*
 * Our 6 LED strip occupies
 * ardruino pins 42, 44, 46, 48, 50, 52
 * and Gnd (ground)
 * Pin 42 Port L: bit 7 (PL7)
 * Pin 44 Port L: bit 5 (PL5)
 * Pin 46 Port L: bit 3 (PL3)
 * Pin 48 Port L: bit 1 (PL1)
 * Pin 50 Port B: bit 3 (PB3)
 * Pin 52 Port B: bit 1 (PB1)
*/
int main (void)
{
  /* set PORTL and PORTB for output*/
  DDRL = 0xFF;
  DDRB = 0xFF;
  while (1)
    {
      /* set PORTL.7&3 high: 1000 1000
       * set PORTB all low: 0000 0000
       */
      PORTL = 0x88;
      PORTB = 0x00;

      _delay_ms(500);
    }
  return 1;
}
```



```
Port L: bit 7 PL7

Port L: bit 5 PL5

Port L: bit 3 PL3

Port L: bit 1 PL1

Port B: bit 3 PL3

Port B: bit 1 PL1
```

- Build the code (F7) or click on the Build
- See the build window below to see if there any possible errors
- If the program compiled with no errors it is time to load the binary hex file (lab8.hex) to the board using avrdude.

III. Exercises:

Download lab8.c.

Realize the following Pseudo-code:

```
turn the first led on;
1 second delay;
while(1)
{
   turn the current led off, turn the next led on; //wrap around when appropriate
   1 second delay;
}
```

Submit lab8.c at the end of your lab.

PART2: LCD Display

Following the steps similar to blink project, create an AVR-gcc project for display. Import the file **main.c**. For this purpose we have provided you complete LCD driver software. This driver is obtained online from http://www.avrfreaks.net.

a. Create your **avr-gcc** project in the AVR Studio 4.

b. Make sure you use WinAVR tools and not the AVR Toolchain as explained above

c. Import the lcd_driver files (lcd_drv.h, main.h, mydefs.h, lcd_drv.c and main.c) into your project space.

d. Our Lab LCD is a 2x16 (i.e., 2 line display with 16 characters on each line) display. Therefore the first thing you have to do is that make sure (i.e., uncomment the 2x16 line) the LCD is set to 2x16 in "main.h" and comment others in the "define wanted LCD type" section of main.h

e. Our LAB cpu uses a clock rate of 16MHz. Therefore you need to also make sure that the correct value for F_CPU in "main.h" file. Otherwise the timing used in the code will not work.

f. The LCD is connected to the board through specific set of digital pins as outlined in the HD44780 tutorial. These are already defined under "define the LCD connections" section of "main.h" with the appropriate port pins.

g. Study the simple "main.c" program. It initializes the LCD by calling lcd_init(); sets the cursor to position (0,0) (first character, first line) using lcd_xy() and puts a string "Welcome to" using lcd_puts(). Since ours is a two line display, it will continue with displaying the second line by setting the cursor to (0,1) (first character, second line) and writes a string "CSc 230".

h. Compile your project and upload your hex file to the board by using “avrdude” program if there are no errors.
i. If everything goes well, you should see “Welcome to CSc 230” on the display.

Modify the code so that you display your name on the second line. Wait for 500ms, then erase it and display it again in a loop creating a flashing effect.