

# AVR Studio Tutorial for ELEN90066

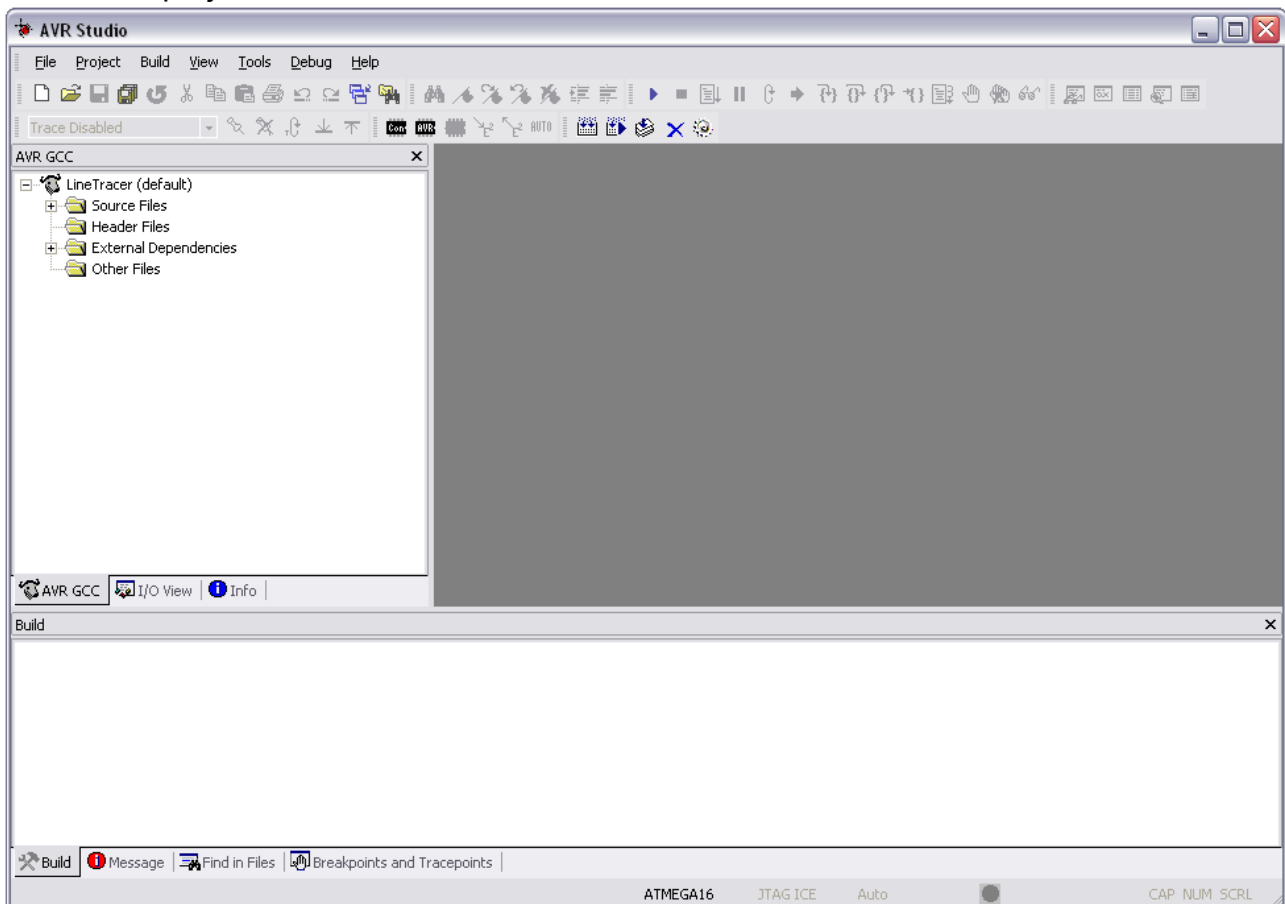
David Jahshan 2012 Version 0.3

## Installing AVR Studio

1. Download AVR studio ZIP from LMS under Assignment → AstudioandWinAVR.zip
2. Unzip AstudioandWinAVR.zip to a convenient location
3. Install WinAVR-20060421-install.exe
4. Install astudio4b460.exe
5. Patch (install) aStudio412SP4b498.exe.
6. In Windows 7 you will be informed there was a problem with installation. Click fix the problem and then repair.
7. If your OS does not support this version newer versions are available from [www.atmel.com](http://www.atmel.com) and [winavr.sourceforge.net/download.html](http://winavr.sourceforge.net/download.html).

## Opening the Template

1. Download the template from LMS under documents → Project Material → game\_consol.zip
2. Unzip game\_consol.zip in a convenient location
3. Run AVR Studio
4. When the dialog pops up, click on Open and browse to the location you unzipped the files to and click open.
5. The project should load.



## Editing Source Files

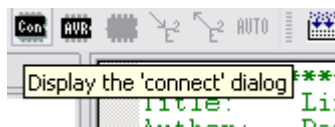
1. Click on the little + sign next to Source Files in the AVR GCC pane. Double click on game\_consol.c.
2. Click on the little + sign next to Header Files and double click on game\_consol.h
3. You will need to edit the header file to make it match your PCB.
4. Look up the port that your LED is connected to in your schematic. Edit the header file. In Line:  
`LED_DIR(DIR) SET(DDR?,(BV(P?#)|BV(P?#)|BV(P?#)),DIR)`  
replace ? With the letter of the port and # with the pins the LED is on
5. If your LED is active high, remove the ~ from before STATE in the LED macros.

## Connecting up the programmer

1. Step 1 is to assemble the programmer and have the firmware installed on it and tested (there will be arranged times when I'll be in the lab to upload firmware for you)
2. Plug the short span of ribbon cable into your game console.
3. Wire some temporary power leads to your game console and connected it to a lab power source. Set the power source to 1.5V and limit the current to 100mA. Do not turn on the power at this stage
4. Plug the programmer into the USB port of the computer. Download and install the FTDI driver if necessary ([www.ftdichip.com](http://www.ftdichip.com) and download VCP driver for your platform.)
5. Turn on the power supply. The LED of your programmer should turn on.

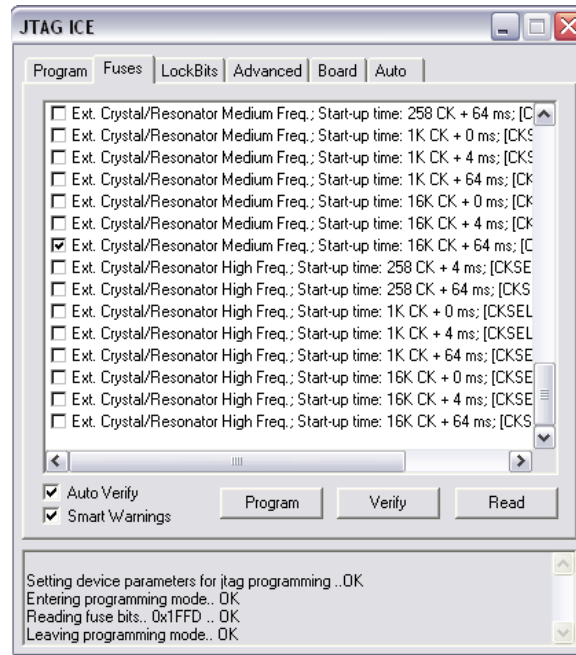
## Test the connection and set the fuses

1. Click on the con button on the top tool bar:



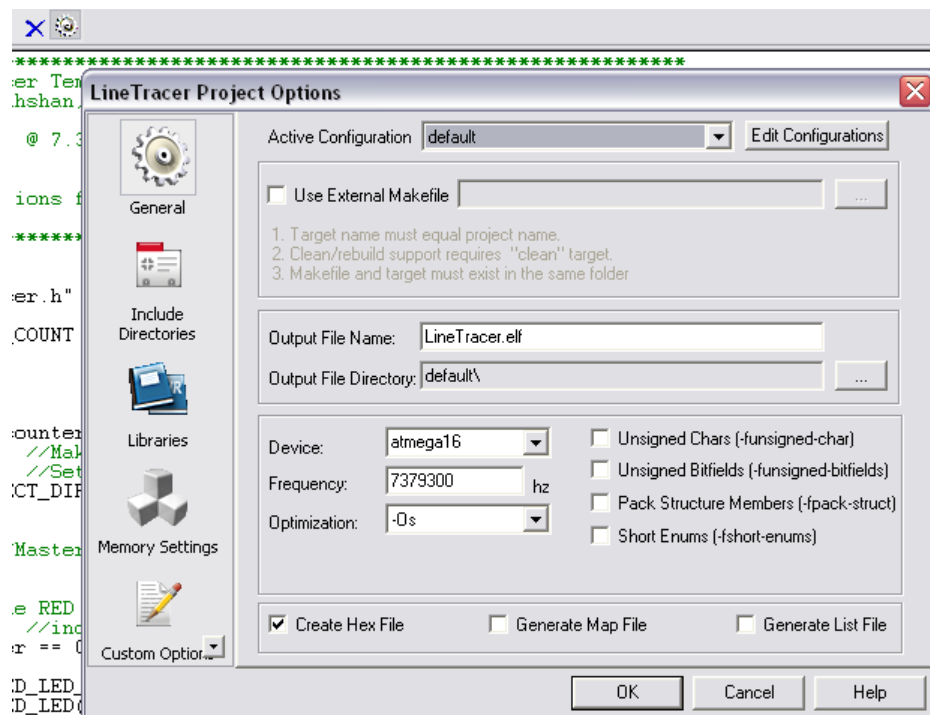
2. Make sure the connection is set on JTAG ICE and Auto and click connect
3. Click on the Advanced Tab and click on Read next to Signature Byte. You should get 0x1E 0x94 0x03 and on the line below it should say Signature matches selected device. If the signature is 0x1E 0x94 0x03 but does not match the selected device, then go to the Program tab and change the device to ATMEGA16.
4. If you get an error message or 3 similar bytes as the signature there is a problem with your circuit. Make sure you have power to your uC. Make sure your programmer is plugged in, and that TDI, TDO, TMS and TCK are connected to the programmer and try again.
5. Click on the Fuses tab and turn on your external crystal oscillator. This is done by selecting Ext. Crystal/Resonator Medium Freq; Start-up time:16K CK + 64ms; Be very careful when programming Fuses, if you do the wrong thing you may brick your chip (breaking it and all it will be good for is a brick). You should always check that JTAG Interface Enabled and Serial Programming Download are always ticked.
6. Once you are satisfied that the correct fuses are selected click on Program.

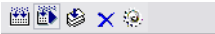
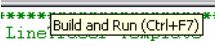
7. To make sure you are still working repeat step 3. Then exit the programmer.



## Compile your code and upload

1. Click on the Edit Current Configuration button and make sure that the project is set up as below

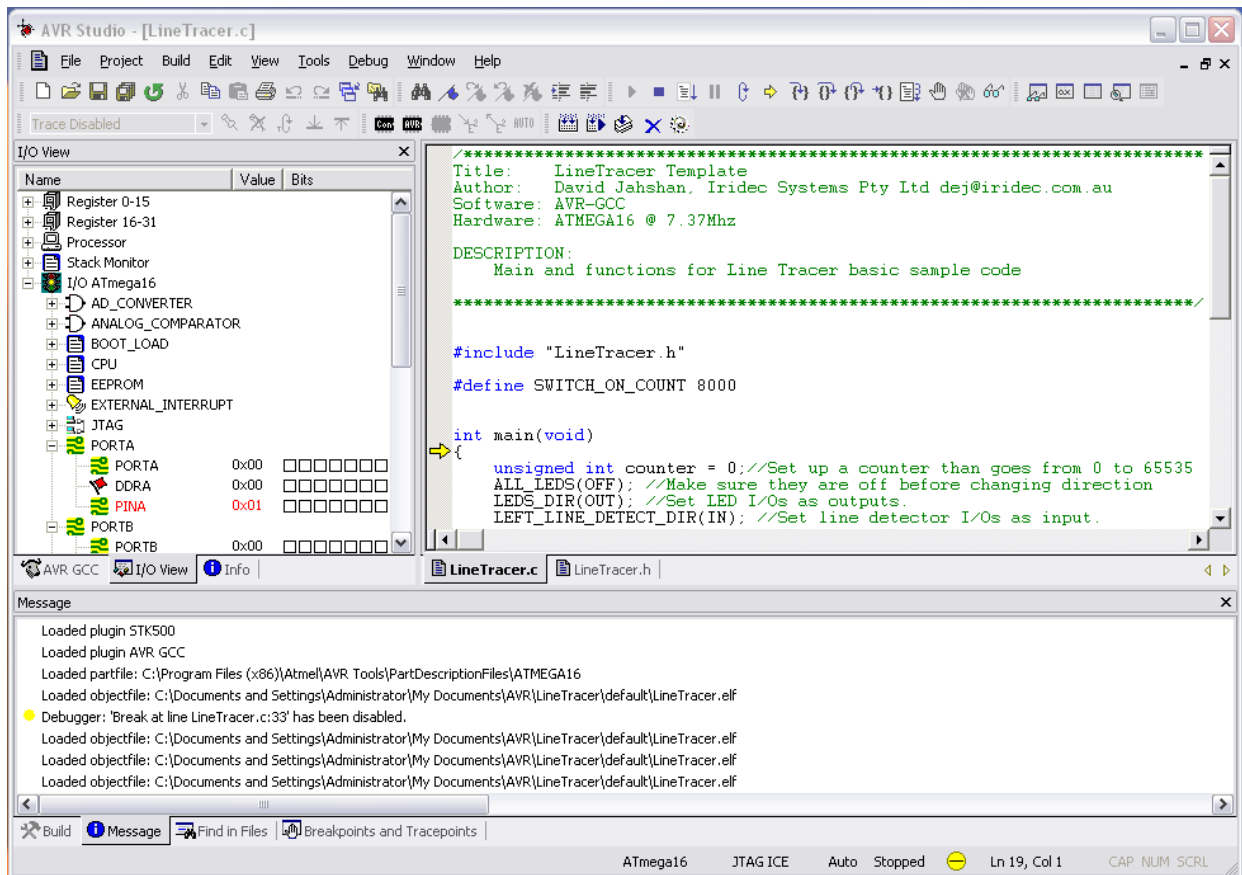


2. Click ok.
3. In the menu, select Debug → Select Platform and Device and Make sure JTAG ICE, Atmega16 and Port is set to Auto. Click Finish. If it tries connecting cancel.
4. Click on the Build and Run button    
 
5. The code should compile upload and the debugger will be launched.
6. If the code fails to compile fix the bugs and repeat from step 4. If you modify code, click on 4 and it will auto save and build and run.

7. If you need to clean the code, (you have made major changes in multiple header files) you can click on the X two across from Build And Run and all object files will be deleted and rebuilt.

## Debugging

1. The debugging screen will now be displayed. The Yellow pointer shows you the line at which the program is at. On the left, if you click on the I/O View tab, you can see what the current internal states the processor is at. Click on I/O Atmega16 and expand the ports A to D.



2. To debug toolbar controls the debugging process:



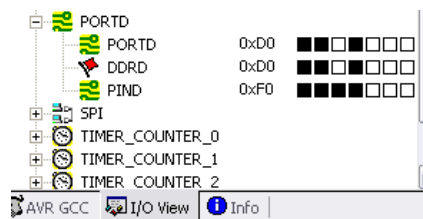
3. The run button starts the program and runs it indefinitely.
4. The pause button stops the execution.
5. The restart button (next to the pause button) takes the program to the start of the program.
6. The step into button (next to the yellow arrow button) executes the line you are currently at.
7. Step over (next to the step into button) will execute a command that is a function or a loop without entering the function or loop.
8. The step out of button (next to step over) will run the loop or function till it exits.
9. Run to cursor (next to step out of) is useful when you do not want to step through

hundreds of line of code to get to a particular line. Place your cursor on the line you are interested in then click on this button

10. The auto step button is (next to run to cursor) is useful when you want the program to step through the program reasonably quickly. (watch out, this sometimes crashes the program)
11. A breakpoint is a place where the program will pause when it is reached. To activate a break point click on the break point button (next to auto step button has a drawing of a hand). When you click on the run button, it will run till the breakpoint is reached. You can have a maximum of 3 breakpoints are permitted at any one time.
12. To remove all breakpoints click on the multiple hands with a X across it.
13. To watch a variable (to keep track of the current value that variable has) Click on the quick watch button (the glasses next to remove all breakpoints) and select the variable you are interested in. A watch window will appear with that variable.
14. To view and step through the disassembled code click on the Toggle Disassembler Window button. Click on this button again to return to source.



15. Right clicking on variable or on lines provide a context menu that is sometimes quicker than the buttons described above.



16. To change the values of the I/O pins of the micro, under the left pane I/O view, go to the port and click on the boxes to turn signals on or off. You can see what is being received by the pin on the bottom row PINx. Remember the input only gets updated when you step, the output is updated immediately. You can edit all values appearing in the I/O pane, including registers,
17. If you want to make changes to your code, click on the stop button (next to run button) and make changes. Click on the compile and run button to reload your work. The play button (next to the stop button) only reloads the current code, and does not compile. If you are running the program, editing is limited (you can not select an area), so if you need to do extensive editing it is advisable to stop first.