Purpose

The purpose of this exercise is to have ”Hands On” the ”STK500 Starter kit” and the Atmel Studio assembler + C compiler. It’s important to put focus on the development tools – and less important to understand the program examples used in the exercise.

Material

- STK500 starter kit + power supply + USB/RS232 converter (unless your PC happens to have a serial COM-port).
- STK500 users guide (AMS Blackboard “file sharing”).

The exercise

In this exercise we will:

- Install Atmel Studio 6.
- Mount the microcontroller Mega32 at the STK500 board.
- Mount various cables at the STK500 and connect to the PC.
- At the PC: Start and prepare Atmel Studio 6.
- Create an Atmel Studio ASSEMBLY project.
- Write a simple assembly program.
- Build (assemble) the project.
- Download the program to STK500.
- Test the program.
- Modify the program and test the changes.
- Create an Atmel Studio AVR GCC C project.
- Write a simple C program.
- Build (compile) the project.
- Download the program to STK500.
- Test the program.
- Modify the program and test the changes.

Follow the step-by-step guidance (next pages).

Step 1: Install Atmel Studio 6
Check out to see, if your PC allready has Atmel Studio 6 installed (Start->All Programs->Atmel AVR Tools->Atmel Studio6).
If this is not the case, Atmel Studio 6 has to be installed.

Run the setup file "as6installer-6.0.1843" to be found at AMS Blackboard:

Click "Next" all through the installation and accept the license terms.

While waiting for the installation to end, carry out steps 2 to 4 (the following pages).

Click "Finish", when this window shows up:

The installation now has to upgraded:
Run the file " as6installer-patch-6.0.1882" also to be found at the AMS Blackbord (folder "Atmel Studio 6.0"):

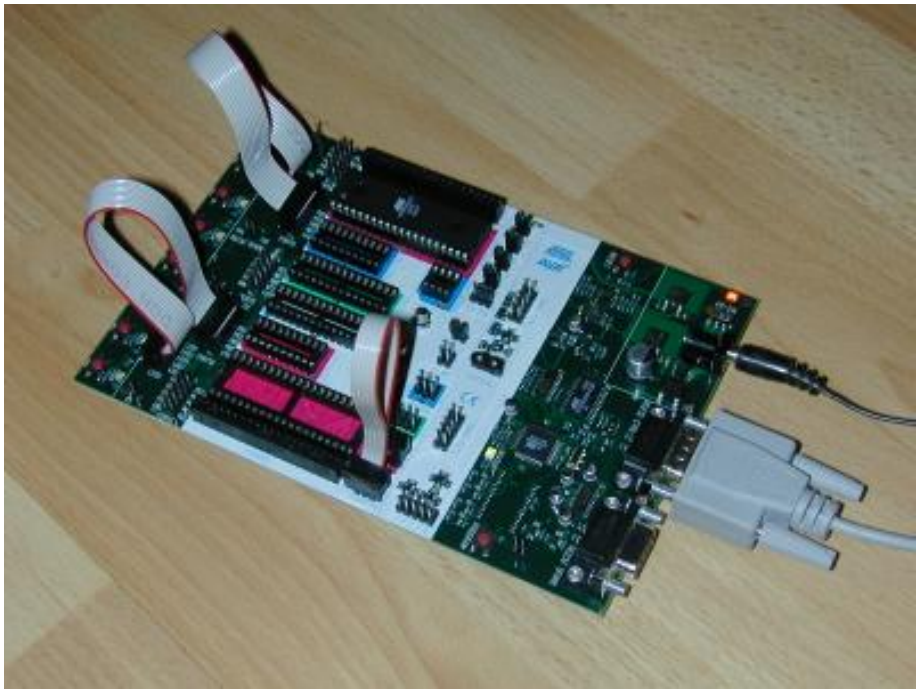
Click "Next" and then "Finish".

NOTICE: If your PC runs Windows 8, also run the file "as6installer-6.0.1996.exe" !

Step 2: Exchange the microcontroller to a Mega32 (if not already done)
For many exercises in this course we will use the microcontroller ATMega32 that comes as an extra device within the STK500 package (if bought at the ASE book store).

The STK500 kit is factory delivered with another (outdated) microcontroller (typically a Mega8515 or a 90S8515).
Since only **one** microcontroller is allowed to be mounted at a time, we have to dismount the original controller chip. Since it is very tight fit in the socket, it might be necessary to use a special tool for the removal (eventually ask the teacher for a little help).

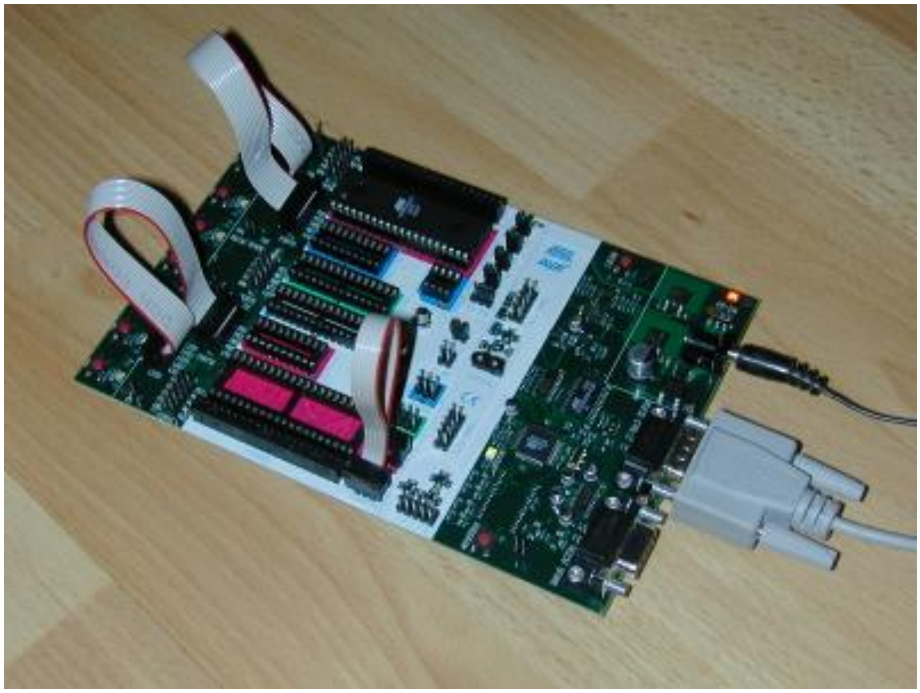
Then mount the Mega32 controller at the STK500 board (OBS: In the socket marked ”SCKT3100A3”, **not** being the same socket used for the original controller chip).
Mega32 has to be mounted as shown below.
Be careful to orientate it properly, and take care not to damage the chip pins.



Step 3: Mount the relevant cables
To be able to download programs to the microcontroller, you will have to mount a 6-wire cable between the connectors ”ISP6PIN” and ”SPROG3” (see the figure).

In this exercise we will connect the LEDs to the PORTB pins.
Therefore also mount a 10-wire cable as shown at the figure (”PORTB” til ”LEDS”).

You don’t have to mount the third cable shown at the figure.
Be aware not to mount the cable ”twisted”.



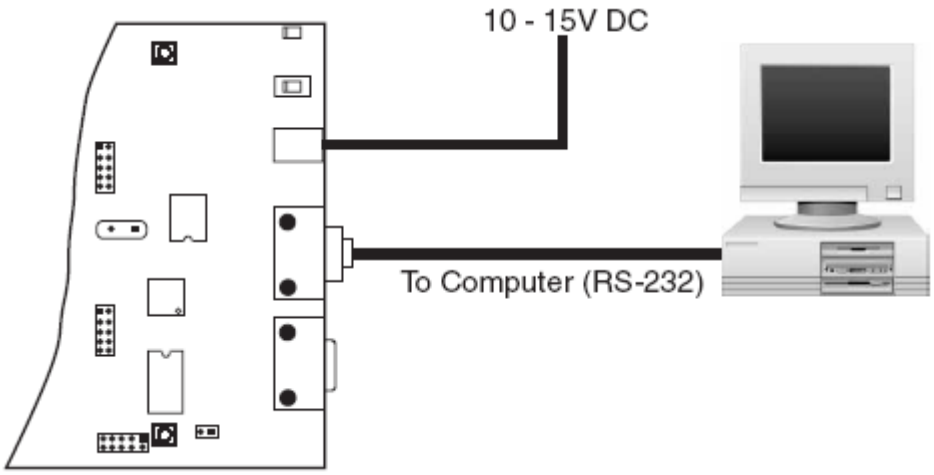
All the cables come with the kit (plastic bag).

Step 4: Connect to PC and power supply

If your PC has a serial port (a COM port), unfortunately not being the fact in most cases for a modern laptop computer:

a. Connect the STK500 to the PC COM port using the serial cable and apply power (10-15 volts, minimum 500 mA).

b. Switch on the STK500 (power button).

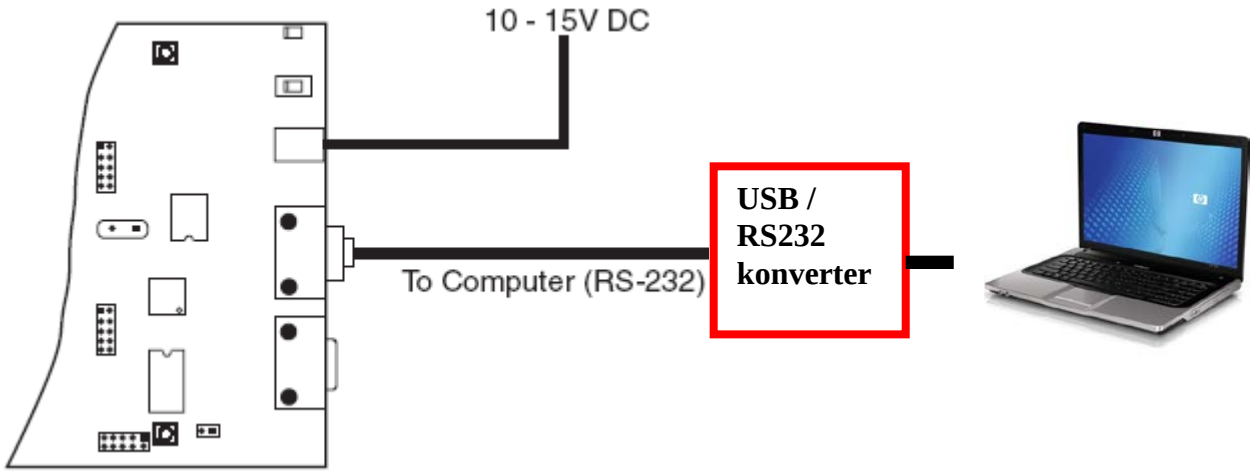


*If your PC **has no** COM port* (the most likely):

a. Connect the STK500 to a USB/RS232 converter using the serial cable. Connect the USB/RS232 converter to the PC (an arbitrary USB connector).

b. Normally there will delivered a driver on a CD when you buy the USB/RS232 converter. This driver has to be installed on the PC before you can use the converter.
Install the driver, if your PC demands it.

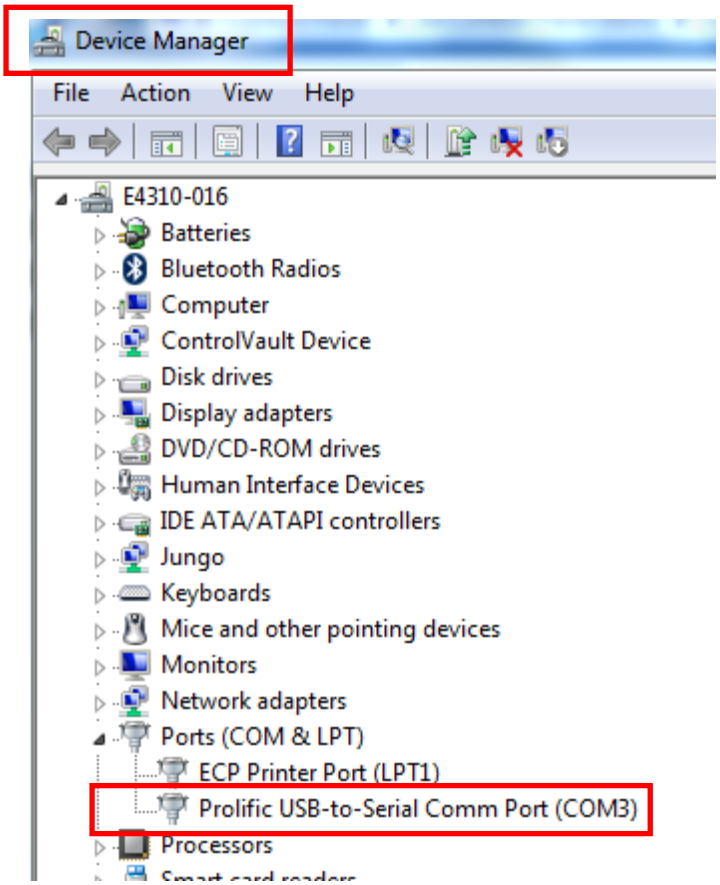
c. Apply power (10-15 volts, minimum 500 mA), and switch on the STK500 (power button).



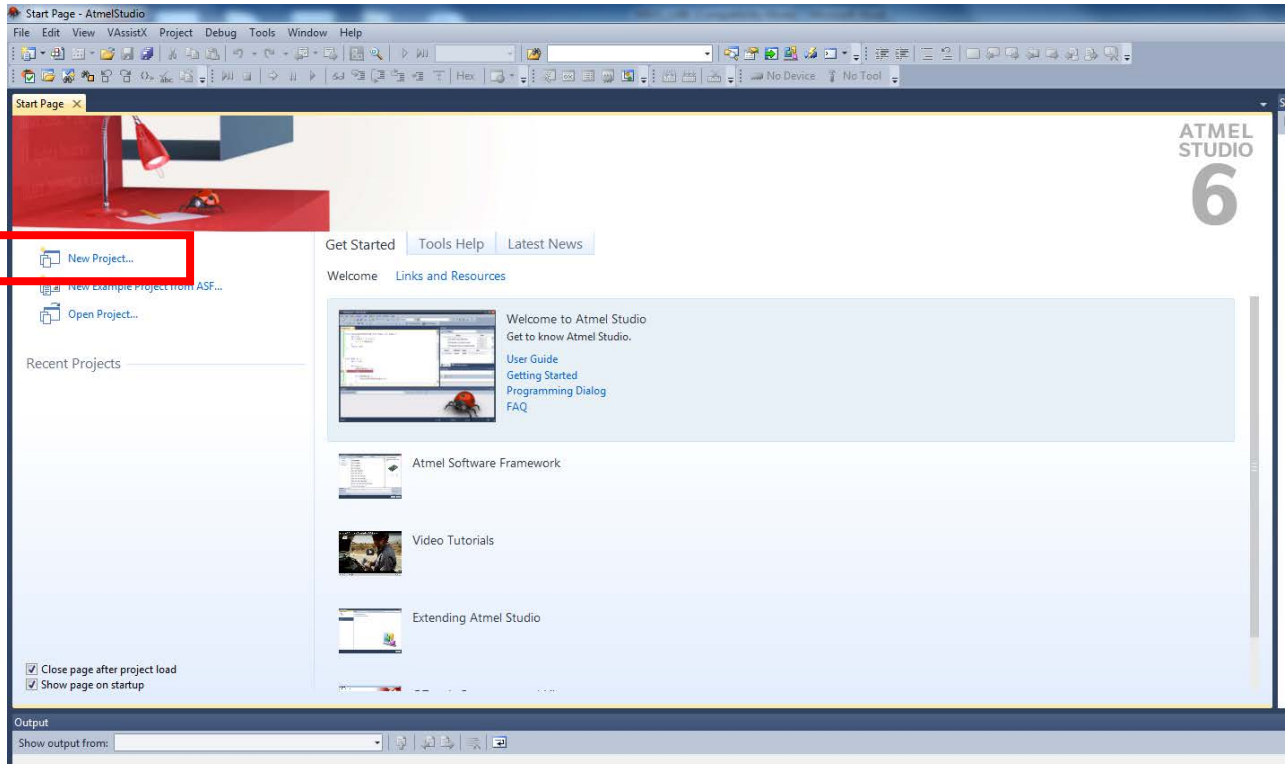
Comment and hint regarding the USB/RS232 converter

* When you the first time connect the USB/RS232 converter, a driver for this unit has to be installed. Often this happens automatically, but otherwise you might use the driver on the CD that comes with the converter (normally it is well hidden in the wrapping of the unit).
Since the CD physically is very small, don’t put it in the disc drive of a MAC computer. Maybe it can’t be ejected again!
Another quite save method is to use the USB/RS232 driver to be found at IECA Blackboard -> File sharing -> USB/RS232 driver. Download the file, unzip and install.

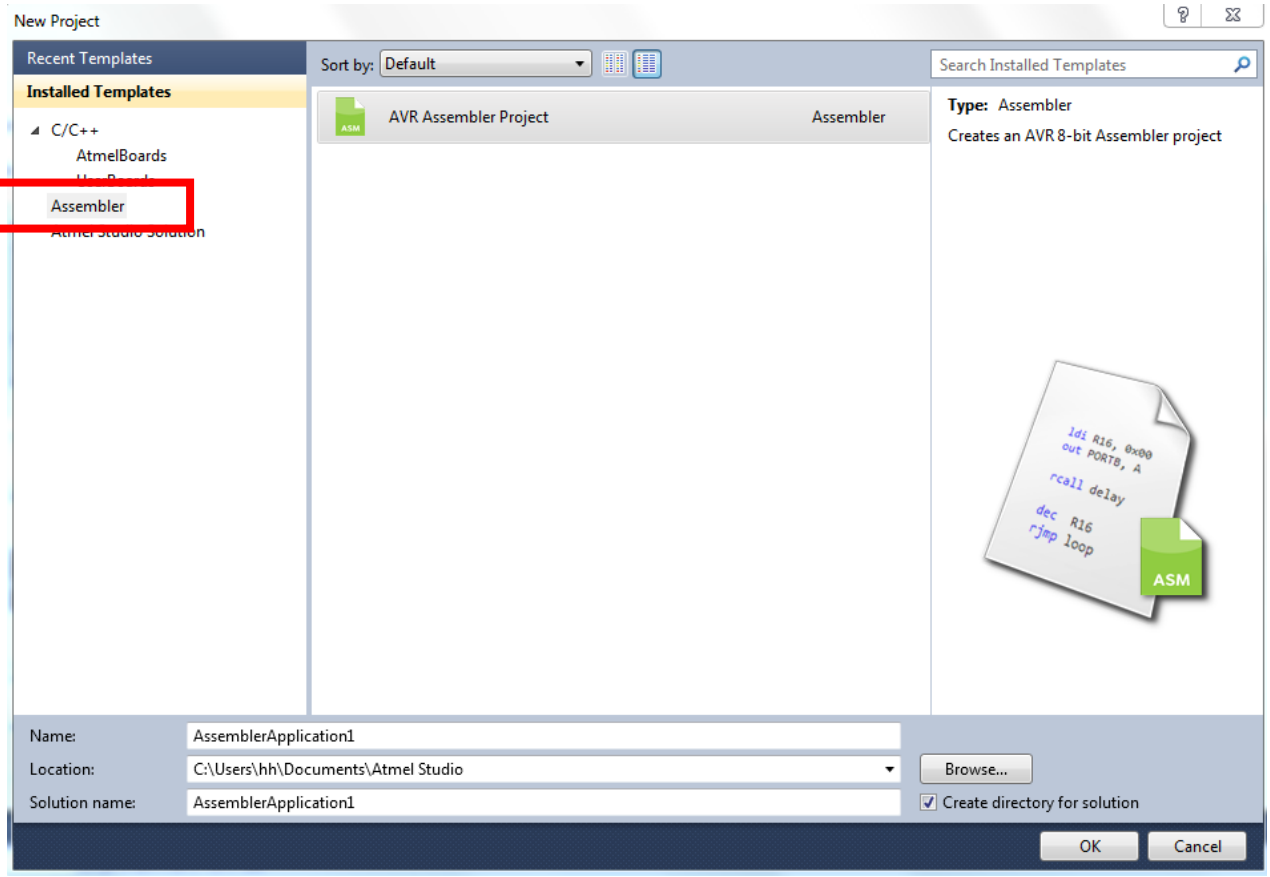
* The operating system of your PC automatically assigns a ”COM”-port number (e.g. COM4). Check “Device manager” to see the assigned port number.
For COM 1 - COM 9 there is no problems, but COM numbers higher than 9 (e.g. COM11), will not be accepted by Atmel Studio. In this case, another COM number can often be manually chosen using “Device Manager”.



Step 5: Start Atmel Studio and create a new ASSEMBLY project
When Atmel Studio starts up, this window shows up:



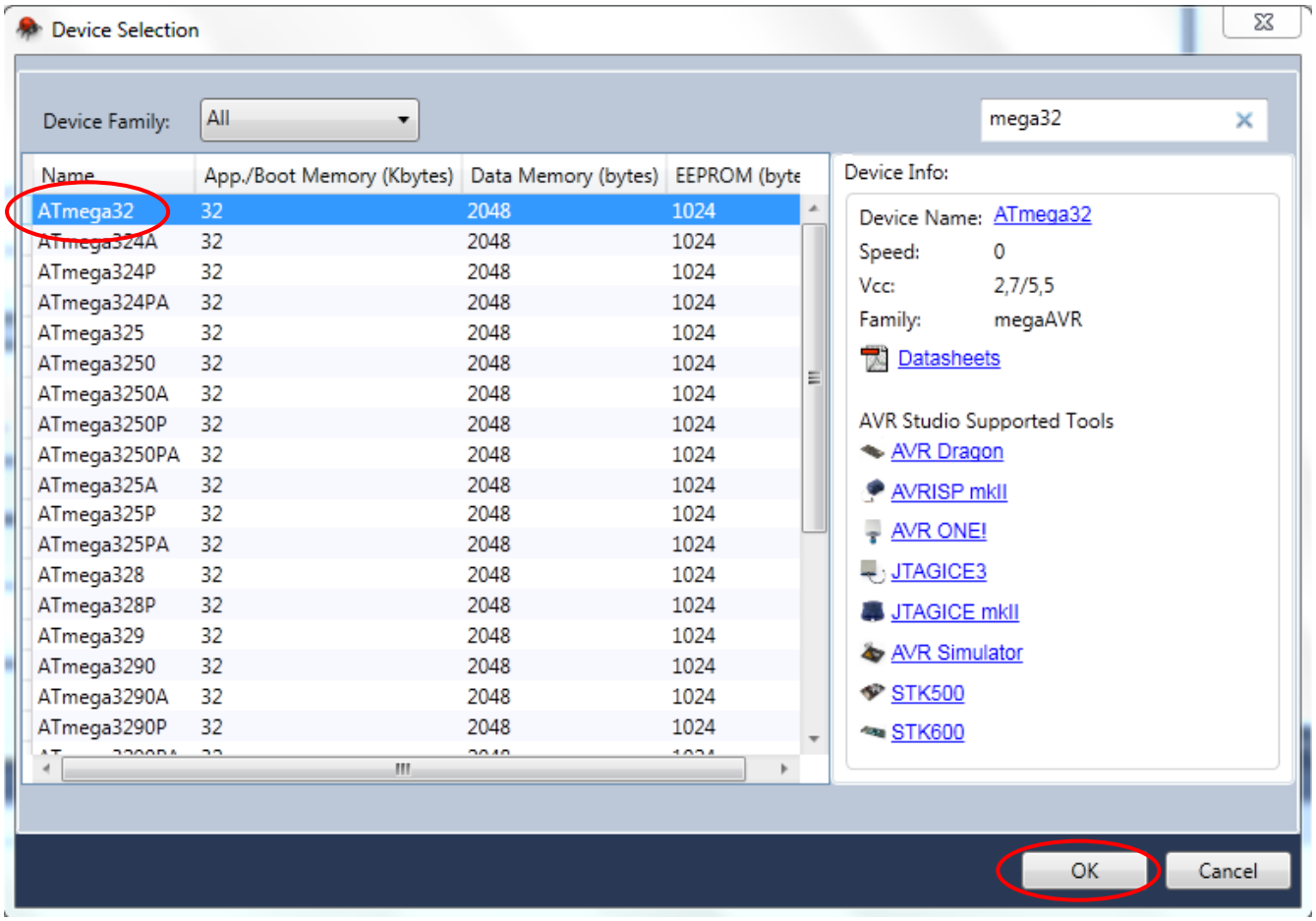
Click "New Project" and then click "ASM Assembler" (to the left):



Step 6: Configure the project
In the fields for "Name" and "Solution Name" write "**LAB1_ASM**".
For the field "**Location**" browse to the folder you want the project files to be stored.

Click "OK".

Now a dialog box appears, telling you to mark what "Device" (e.g. Microcontroller) to use.
In this case select "**ATmega32**".



Click "OK".

The project will be created in the folder, you selected.
The editor will open, enabling us to white our assembly program.

Step 7: Write the assembly program
At the AMS Blackboard (LAB1 folder) you will find the simple assembly program, which we are going to use.
You can write this in the editor window "LAB1_ASM.asm" – or simply copy-paste from the file (called "LAB1_ASM.txt" at Blackboard).

LAB1_ASM.asm

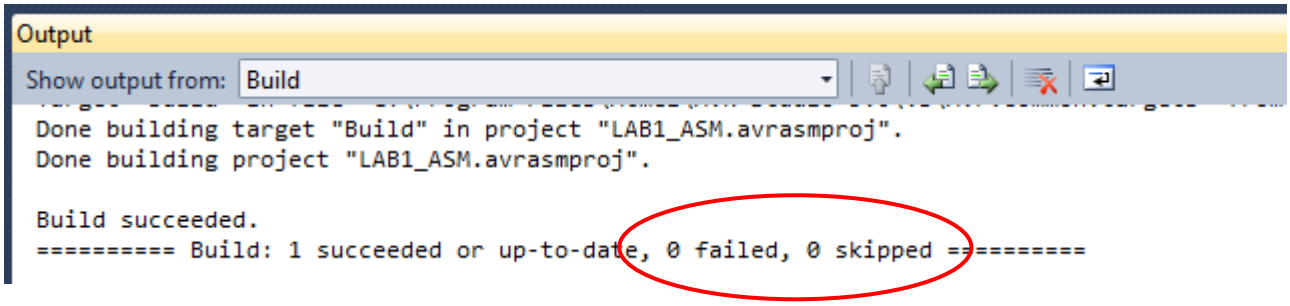
```
;******* AMS, LAB1 **************
;******* The ASM example ********
;******* Henning Hargaard *******
;******* January 8, 2012 ********
;*******************************

;******** INITIALIZATION ********
.include "M32DEF.INC"
   LDI  R16,HIGH(RAMEND) ;Initialize Stack Pointer
   OUT  SPH,R16
   LDI  R16,LOW(RAMEND)
   OUT  SPL,R16
   SER  R16              ;PORTB = Outputs
   OUT  DDRB,R16

;********** PROGRAM LOOP ********
   CLR  R16
LOOP:
   CALL DISP_AND_DELAY  ;Display R16
   INC  R16
   RJMP LOOP            ;Jump to "LOOP"

;********** DISPLAYS R16 ********
;********** + SHORT DELAY *******
DISP_AND_DELAY:
   MOV  R17,R16
   COM  R17
   OUT  PORTB,R17
   CLR  R17
   CLR  R18
   LDI  R19,10
AGAIN:
   DEC  R17
   BRNE AGAIN
   DEC  R18
   BRNE AGAIN
   DEC  R19
   BRNE AGAIN
   RET
;*******************************
```

Notice the automatic "syntax highlighting", e.g. the reserved words gets special colors, making the code better readable.

The program initializes the stack pointer and configures PORTB as output pins.
Then in an infinite loop register R16 is incremented and displayed at PORTB (followed by some delay). Study the code to understand how it works.
Select "File"->"Save" to save "LAB1_ASM.asm".

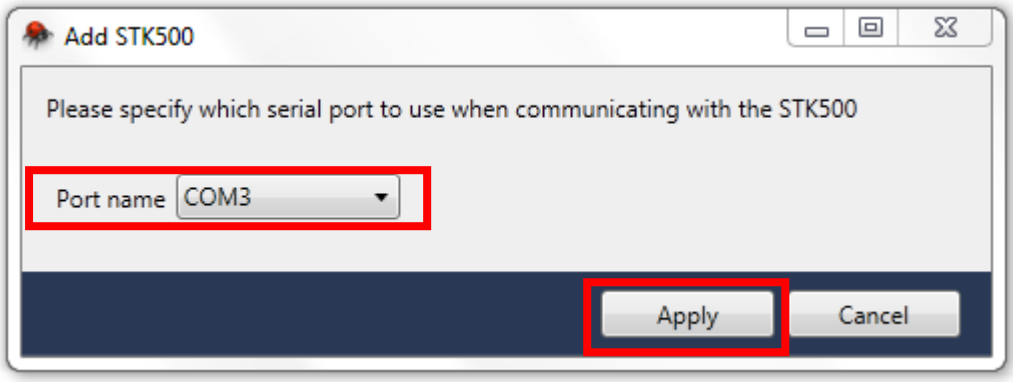
Step 8: Assemble (Build)
Select "Build" -> "Build Solution" or simply press **F7**.
Now the program will be compiled (creating a hex file ready for STK500 download).

If there are no errors, the Output window will appear like this:



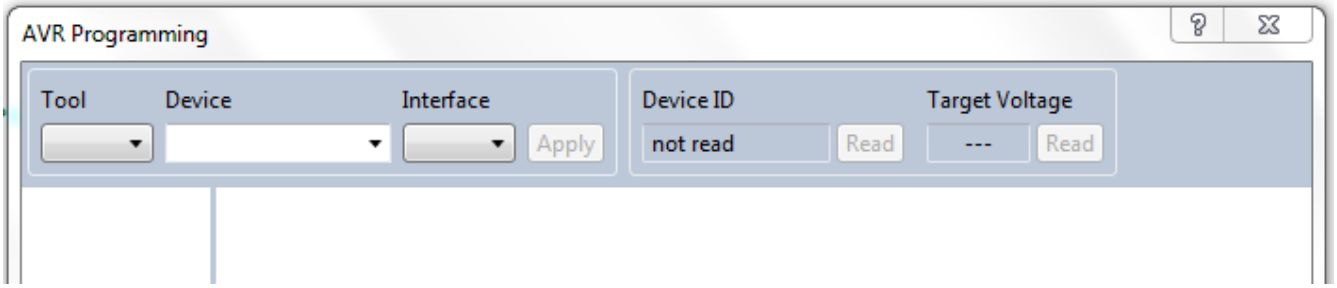
In case of compile errors: Locate them, correct and rebuild.

Step 9: Connect to STK500 and download the program
Having STK500 connected to your PC (refer to step 4), select "Tools" -> "Add STK500".

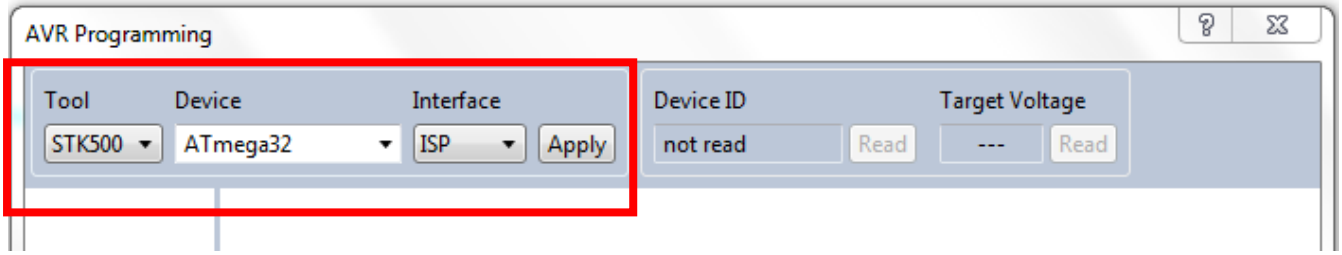


"Port Name" is the COM port connected to the STK500 kit.
Click "Apply".

Select "Tools" -> "AVR Programming":



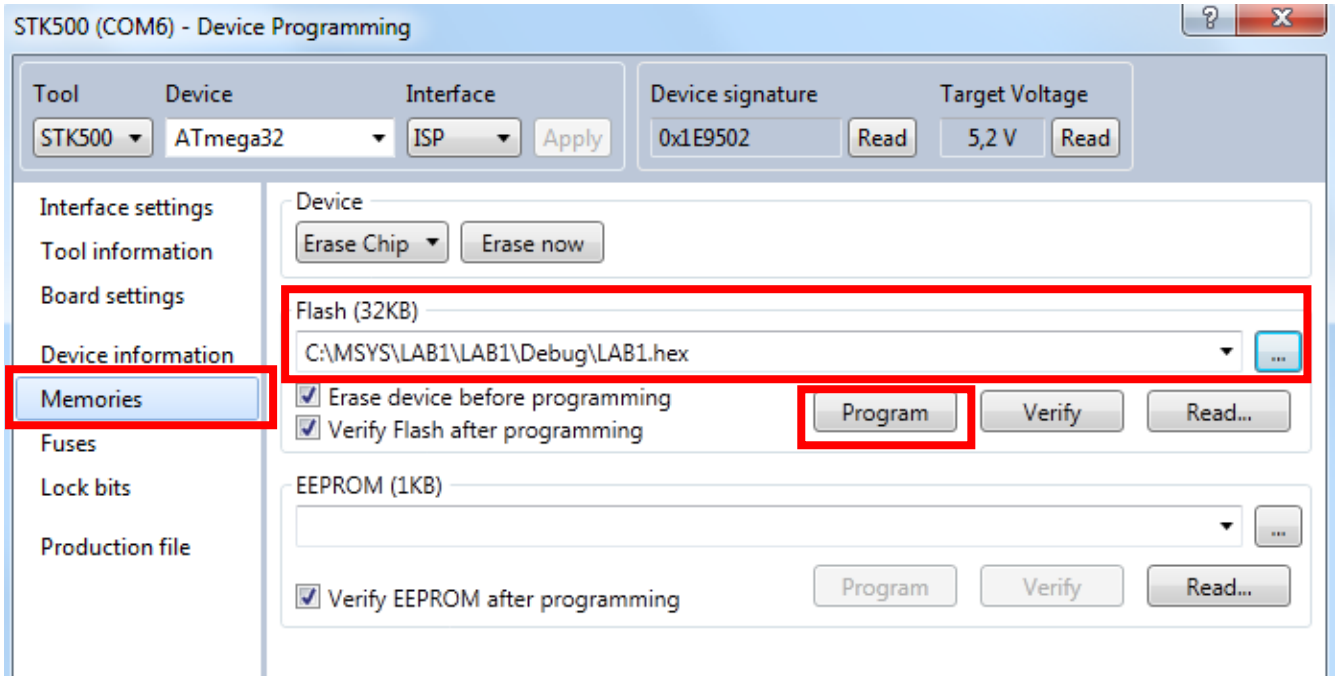
Now set "Tool" to "STK500" and "Device" to "ATmega32". "Interface" must be "ISP":



Click "Apply" and then click at "Memories" (to the left at the window).

If you are prompted to upgrade the STK500 firmware, follow the guidelines at page 14.

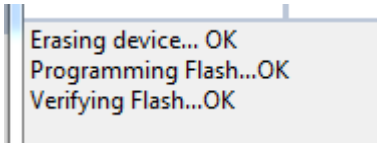
(To be continued next page)



In the "Flash" field: Browse to the "HEX file" to be downloaded (the HEX file will contain the machine codes of our program. After browsing, double click at "LAB1.hex". The file will be located in the folder named "Debug".

Then click "**Program**" (NOTICE: In the "Flash" field).

Now the program will be downloaded to the microcontroller at the STK500 (and the program automatically starts up):



Close the window "AVR Programming".

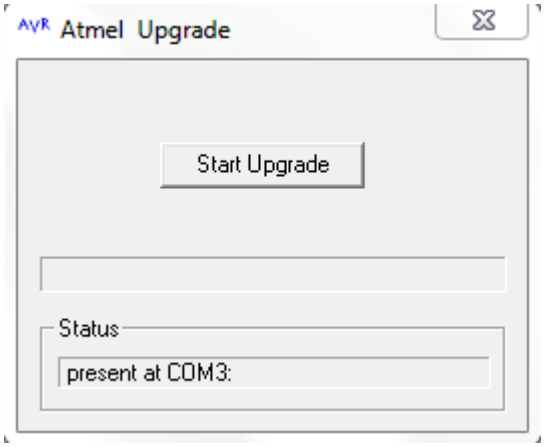
Comments regarding firmware update

If you, when you connect to STK500, are asked to update the firmware, this means that the firmware present at the STK500 board is outdated compared to what is required by your version of Atmel Studio 6.
Unfortunately, the method suggested by Atmel Studio 6 does not work out all right.

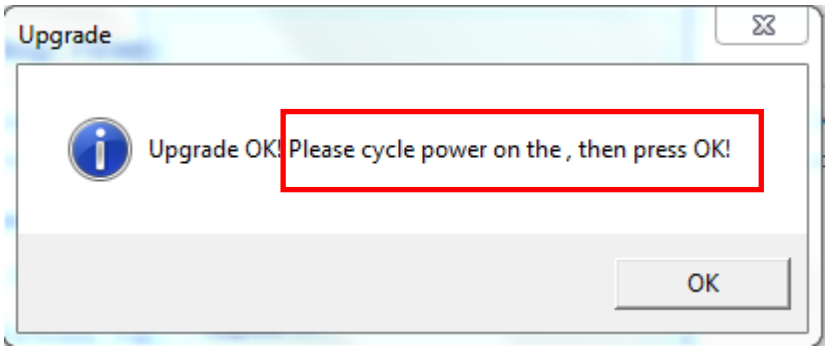
Instead use the program "Upgrade.exe", to be found (zipped) at IECA Blackboard -> File sharing -> STK500 -> Firmware upgrade.

This method will upgrade your firmware:

1. Power off STK500.
2. Keep pressing the button "PROGRAM" at STK500.
3. Power on STK500.
4. Release the button "PROGRAM".
5. Run the program "Upgrade.exe" and click "Start Upgrade":



6. Follow the guidelines given ("Cycle Power" means "power off, then power on").



NOTICE: Don't switch off STK500, while the upgrading goes on.

Step 10: Test the program
Study the STK500 LEDs and test that the program behavior is as expected!

LAB1_ASM.asm

```
;******* AMS, LAB1 **************
;******* The ASM example ********
;******* Henning Hargaard *******
;******* January 8, 2012 ********
;*******************************

;******** INITIALIZATION ********
.include "M32DEF.INC"
   LDI  R16,HIGH(RAMEND) ;Initialize Stack Pointer
   OUT  SPH,R16
   LDI  R16,LOW(RAMEND)
   OUT  SPL,R16
   SER  R16              ;PORTB = Outputs
   OUT  DDRB,R16

;********** PROGRAM LOOP ********
   CLR  R16
LOOP:
   CALL DISP_AND_DELAY  ;Display R16
   INC  R16
   RJMP LOOP            ;Jump to "LOOP"

;********** DISPLAYS R16 ********
;********** + SHORT DELAY *******
DISP_AND_DELAY:
   MOV  R17,R16
   COM  R17
   OUT  PORTB,R17
   CLR  R17
   CLR  R18
   LDI  R19,10
AGAIN:
   DEC  R17
   BRNE AGAIN
   DEC  R18
   BRNE AGAIN
   DEC  R19
   BRNE AGAIN
   RET
;*******************************
```

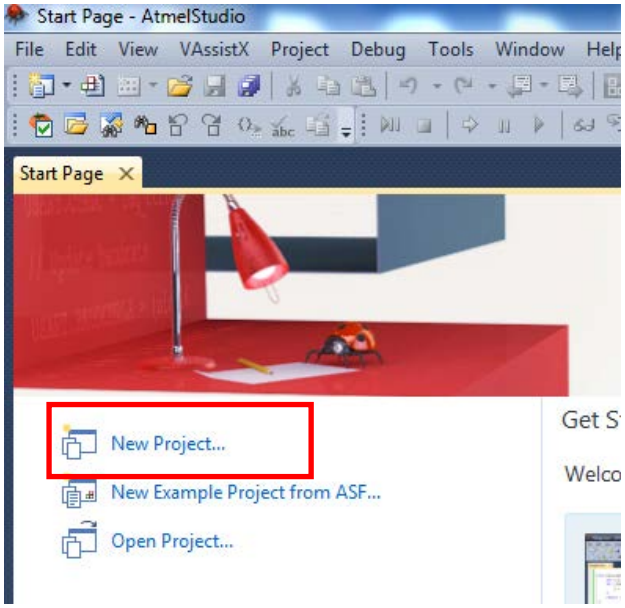
Step 11: Make changes to the program
When you have a good understanding of how the program functions, try doing minor changes to the program. Recompile ("Build") the program, download the code ("Tools" -> "AVR Programming") and test the changes.

End up by closing the assembly project: "File" -> "Close Solution".

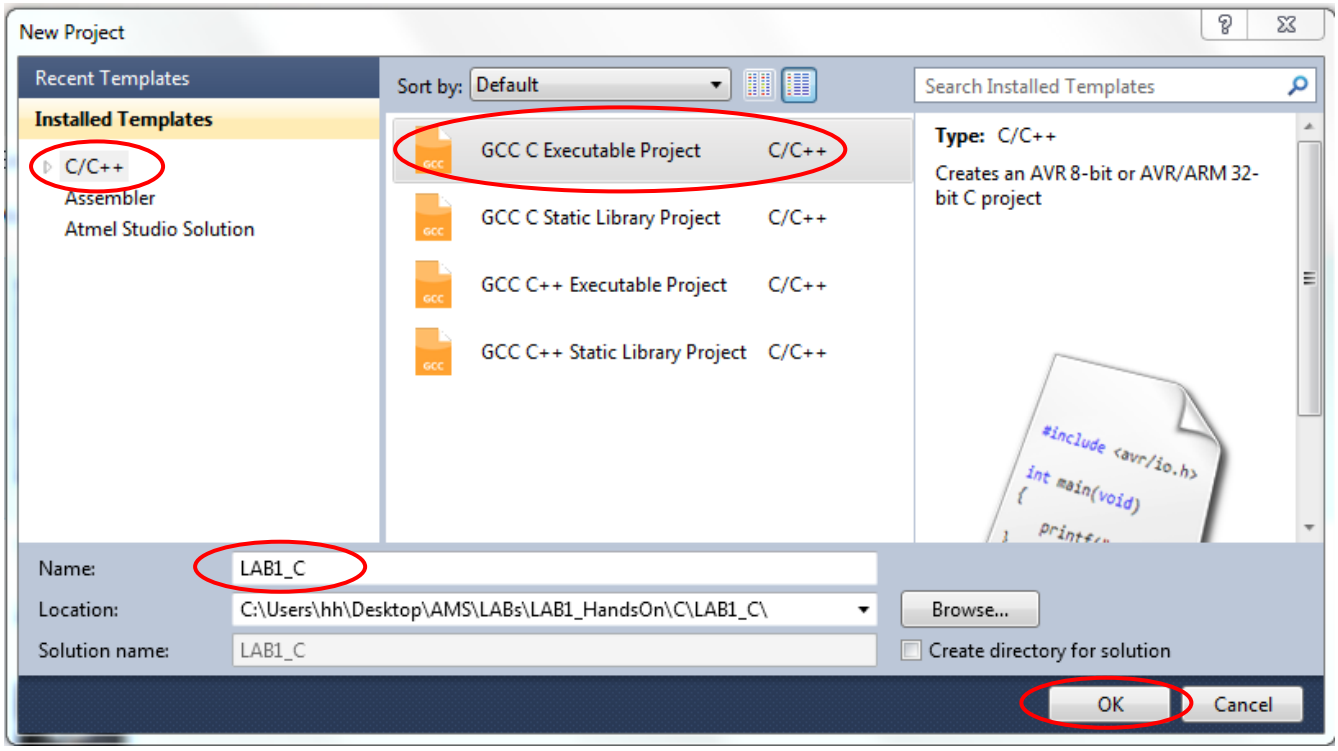
Step 12: Start Atmel Studio and create a new AVR GCC project
In the following steps we will demonstrate how to create an AVR GCC project for a C program.

Embedded in Atmel Studio is an integrated "AVR GCC" C compiler, and in creating a project the main difference is that you should select "AVR GCC" instead of "Assembler".

Start Atmel Studio 6 and click "New Project":

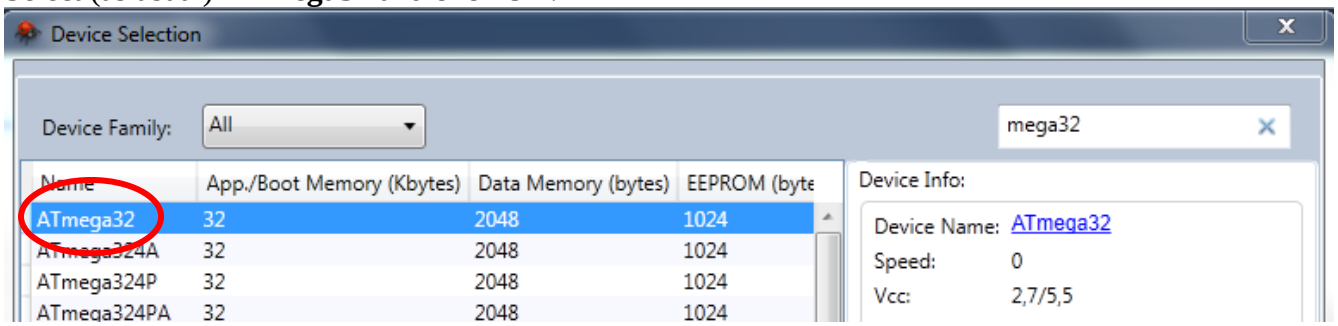


Select "AVR GCC".
Name the project "LAB1_C" and select a proper location.



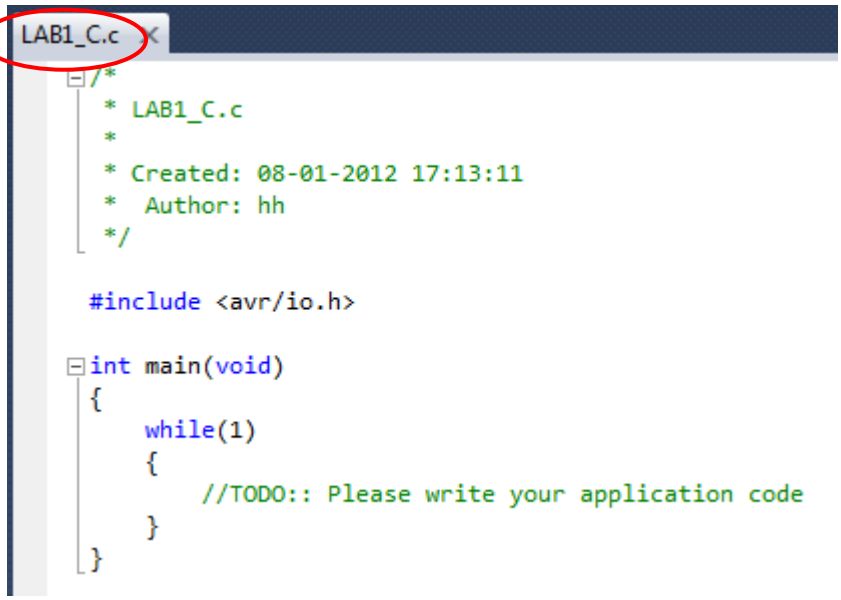
Click "OK".

Step 13: Select device
Select (as usual) **ATmega32** and click OK:



Click ”OK”.

The project now opens with a ”standard” C code skeleton in the file ”LAB1_C.c” :



Step 14: Write the C program
At the AMS Blackboard (LAB1 folder) you will find the simple C program, which we are going to use.
You can write this in the editor window "LAB1_C.c" – or simply copy-paste from the file (called "LAB1_C.txt" at Blackboard).

LAB1_C.c

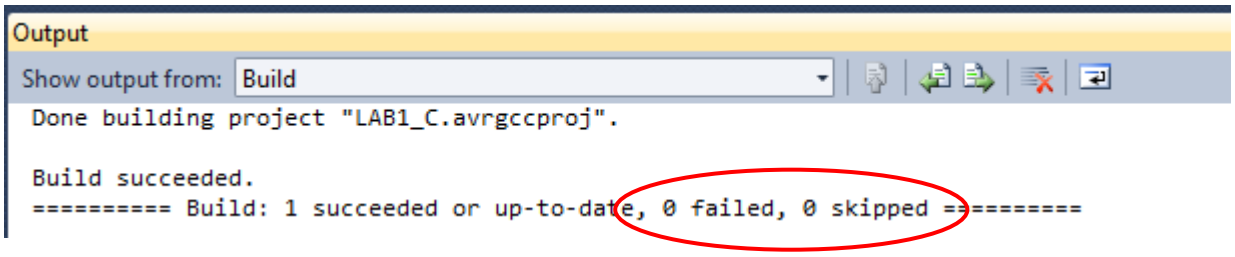
```
#include <avr/io.h>
#define F_CPU 3686400
#include <avr/delay.h>

int main()
{
unsigned char i = 0;

  DDRB = 0xFF; //PORTB pins are outputs (LEDs)
  while (1)
  {
    PORTB = ~i; //Display "i" at the LEDs
    i++;
    _delay_ms(500);
  }
  return 0;
}
```

Study the code to understand how it works.

Select "File"->"Save" to save "LAB1_C.c".

Step 15: Compile (Build)
Select "Build" -> "Build Solution" or simply press **F7**.
Now the program will be compiled (creating a hex file ready for STK500 download).

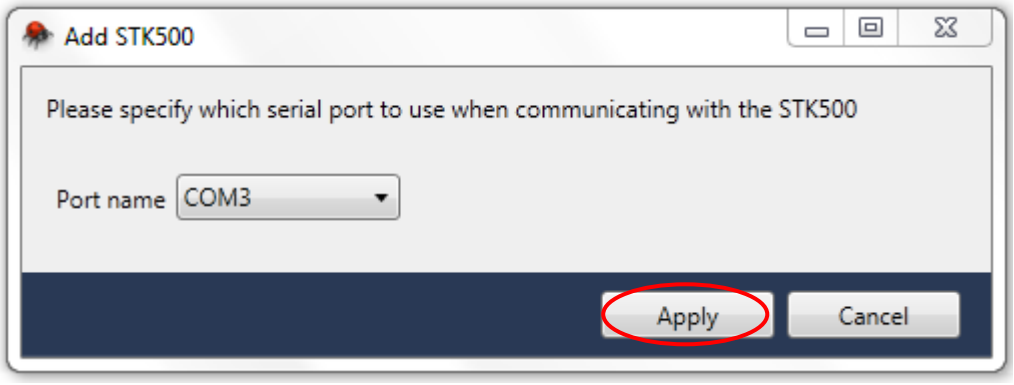
If there are no errors, the Output window will appear like this:

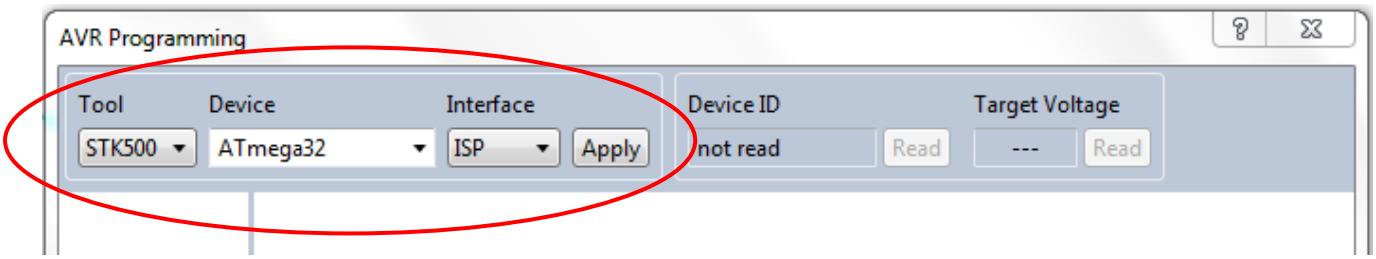


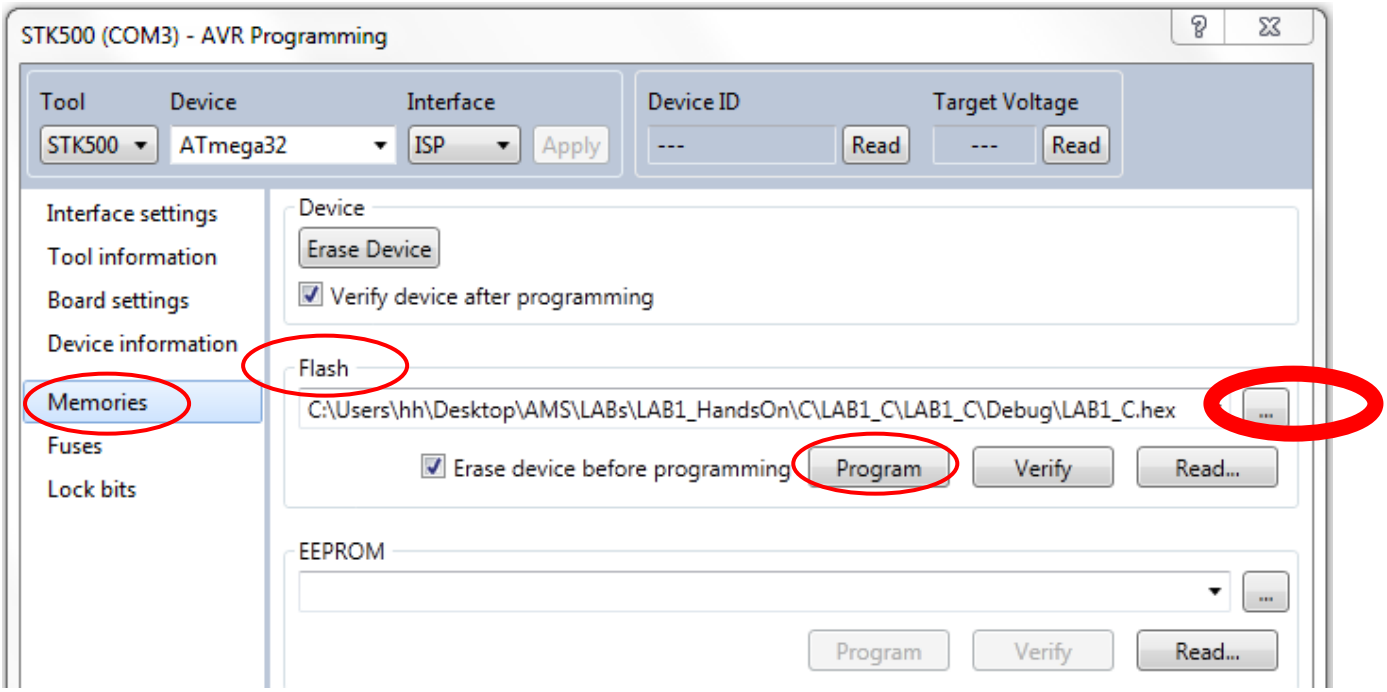
In case of compile errors: Locate them, correct and rebuild.

Step 16: Connect to STK500 and download the program
This process is similar to the instructions given in step 9 (eventually read this again).

VERY IMPORTANT: Remember to browse to the right HEX file ("LAB1_C.hex").
The default HEX file will be the last one used (probably "LAB1_ASM.hex").







Step 17: Test the program
Study the STK500 LEDs and test that the program behavior is as expected!

LAB1_C.c

LAB1_C.c C:\Users\hh\Desktop\AMS\LABs\L

```c
#include <avr/io.h>
#define F_CPU 3686400
#include <avr/delay.h>

int main()
{
unsigned char i = 0;

  DDRB = 0xFF; //PORTB pins are outputs (LEDs)
  while (1)
  {
    PORTB = ~i; //Display "i" at the LEDs
    i++;
    _delay_ms(500);
  }
  return 0;
}
```

Step 18: Make changes to the program
Do minor changes to the program. Recompile (”Build”) the program, download the code (”Tools” -> ”AVR Programming”) and test your changes.