AMS LAB exercise 1

”Hands On” STK500 and AVR Studio 5

Purpose

The purpose of this exercise is to have ”Hands On” the ”STK500 Starter kit” and the AVR Studio 5 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 has a serial COM-port).
- STK500 users guide (AMS Campusnet file sharing).

The exercise

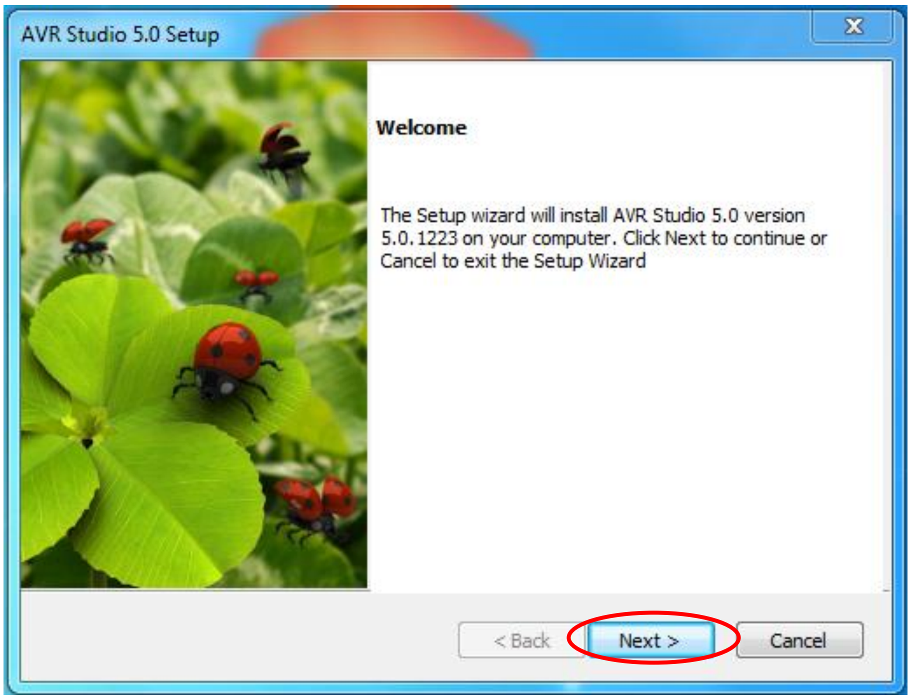
In this exercise we will:

- Install AVR Studio 5.
- 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 AVR Studio 5.
- Create an AVR 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 AVR Studio AVR GCC 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 AVR Studio 5
Check out to see, if your PC allready has AVR Studio 5 installed.
If this is not the case, AVR Studio 5 has to be installed.

Run the setup file "as5installer-5.0.1223-full.exe" to be found at AMS Campusnet file sharing (the "AVR Studio" folder):

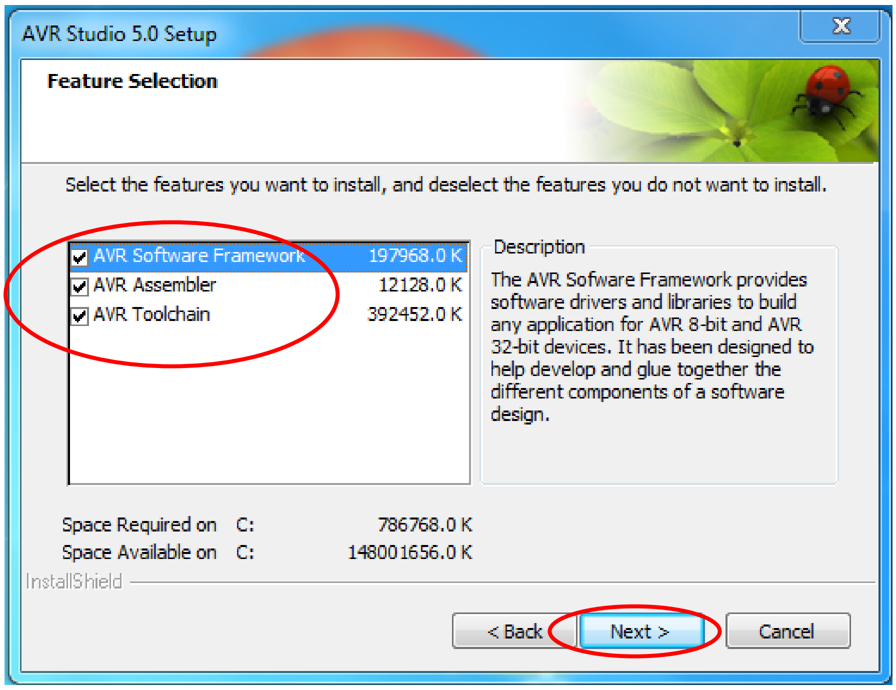


Welcome

The Setup wizard will install AVR Studio 5.0 version 5.0.1223 on your computer. Click Next to continue or Cancel to exit the Setup Wizard

Click "Next" all through the installation.

Accept the licens terms and mark all 3 items at this window:



Feature Selection

Select the features you want to install, and deselect the features you do not want to install.

☑ AVR Software Framework	197968.0 K
☑ AVR Assembler	12128.0 K
☑ AVR Toolchain	392452.0 K

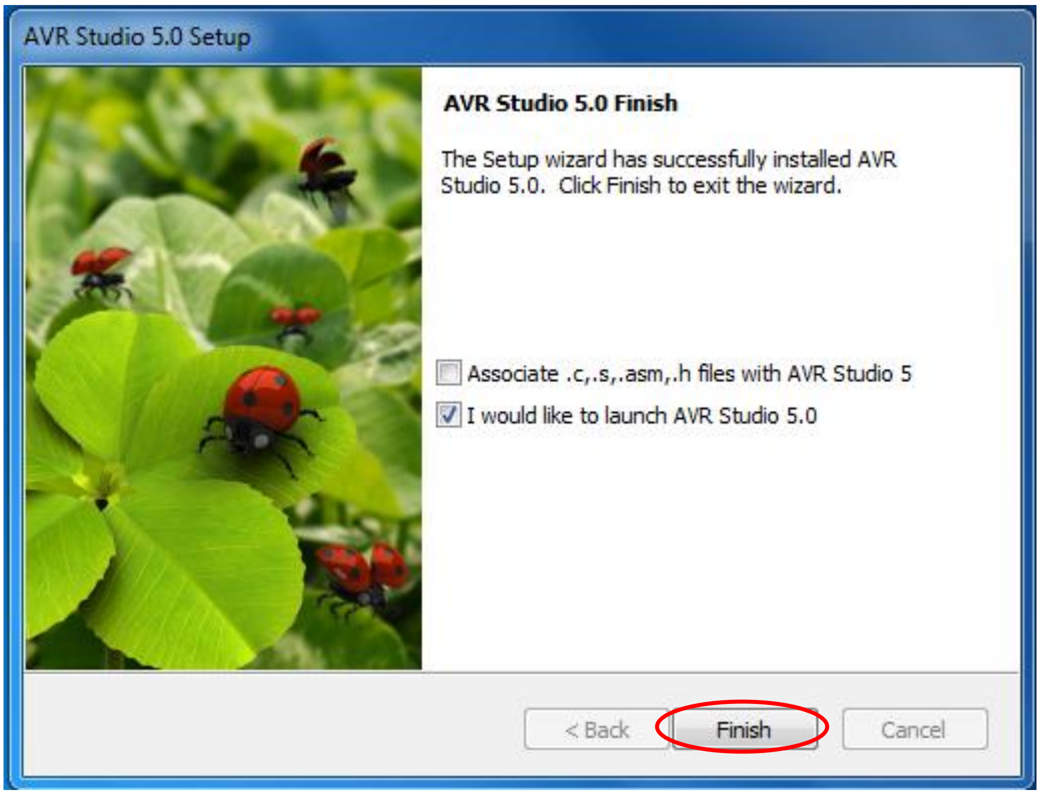
Description

The AVR Sofware Framework provides software drivers and libraries to build any application for AVR 8-bit and AVR 32-bit devices. It has been designed to help develop and glue together the different components of a software design.

Space Required on C: 786768.0 K
Space Available on C: 148001656.0 K

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

Click ”Finish”, when this window shows up:

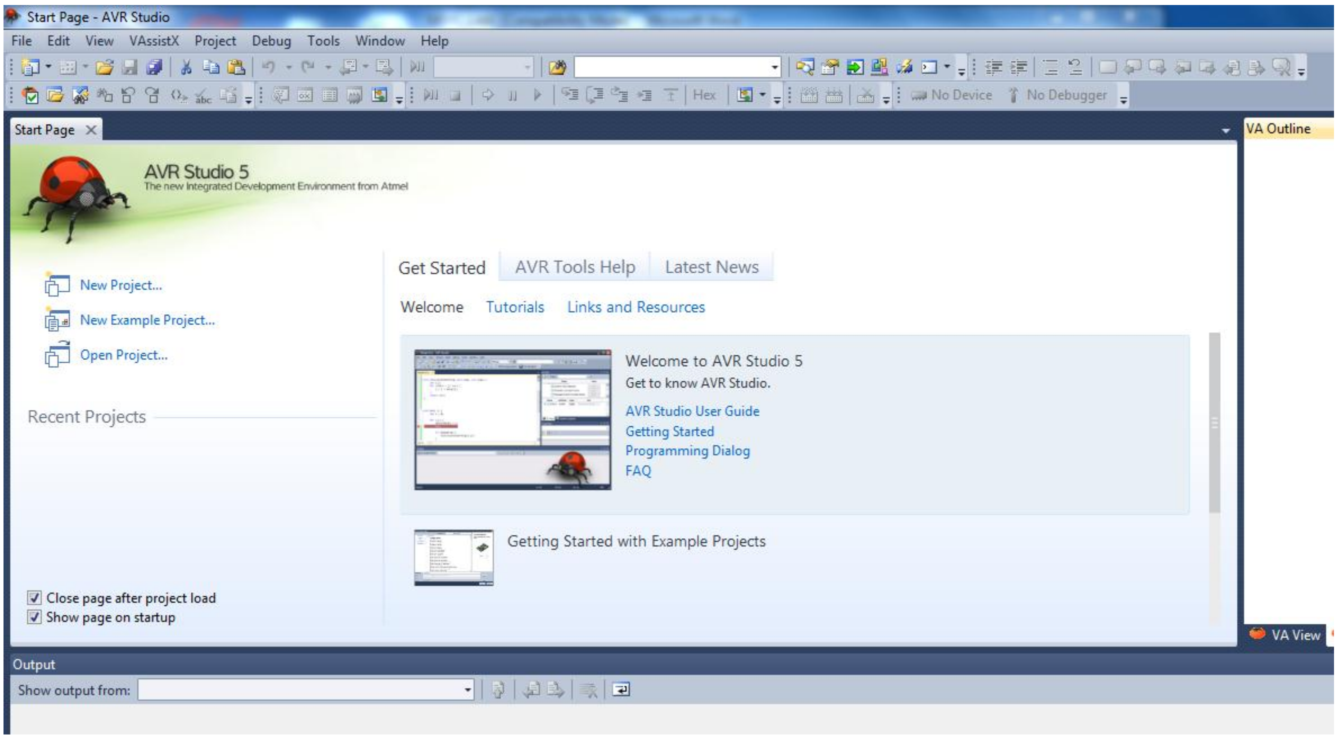

AVR Studio 5.0 Setup

AVR Studio 5.0 Finish

The Setup wizard has successfully installed AVR Studio 5.0. Click Finish to exit the wizard.

☐ Associate .c,.s,.asm,.h files with AVR Studio 5
☑ I would like to launch AVR Studio 5.0

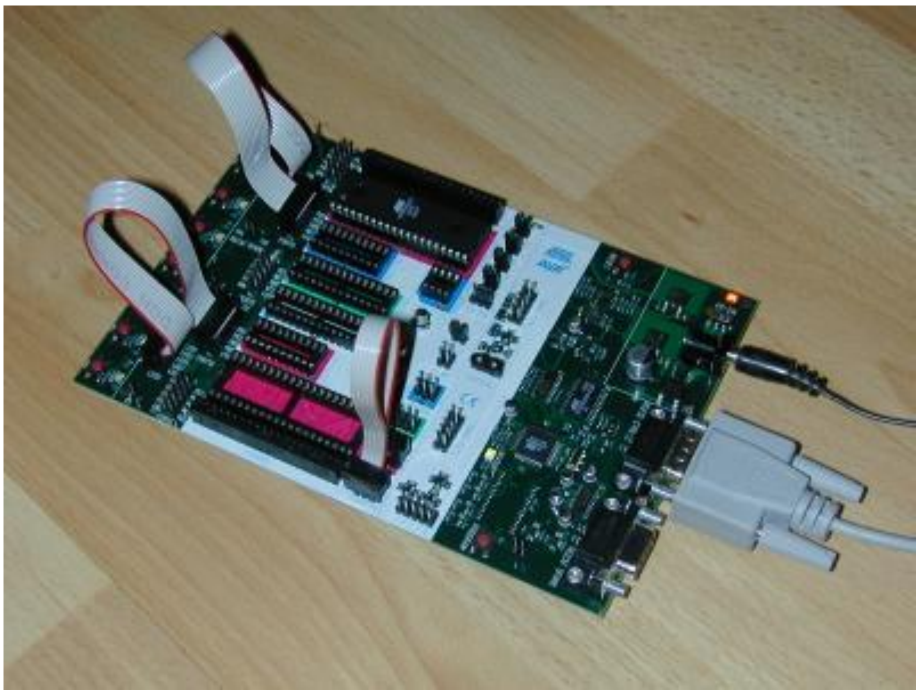
This will cause AVR Studio 5 to start:



Step 2: Exchange the microcontroller to a Mega32
For all exercises in this course we will use the microcontroller ATMega32 that comes as an extra device within the STK500 package (if bought at the IHA book store).

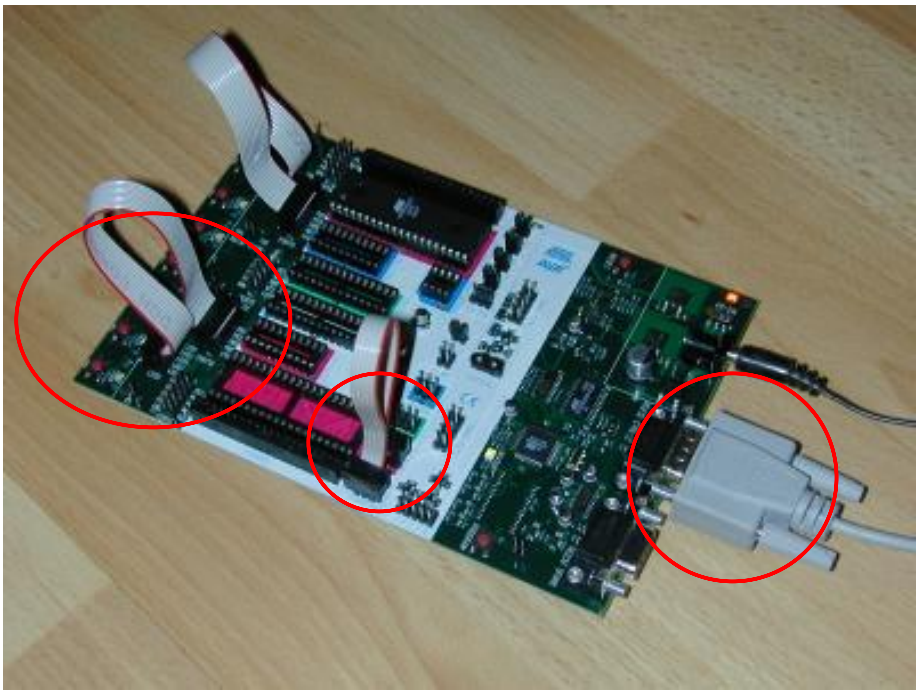
The STK500 kit is factory delivered with another (outdated) microcontroller (typically a Mega8515 or a 90S8515).
Since only **one** microcontroller is allowed to be mounted at a time, we have to dismount the original controller chip. Since it is very tight fit in the socket, it might be necessary to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



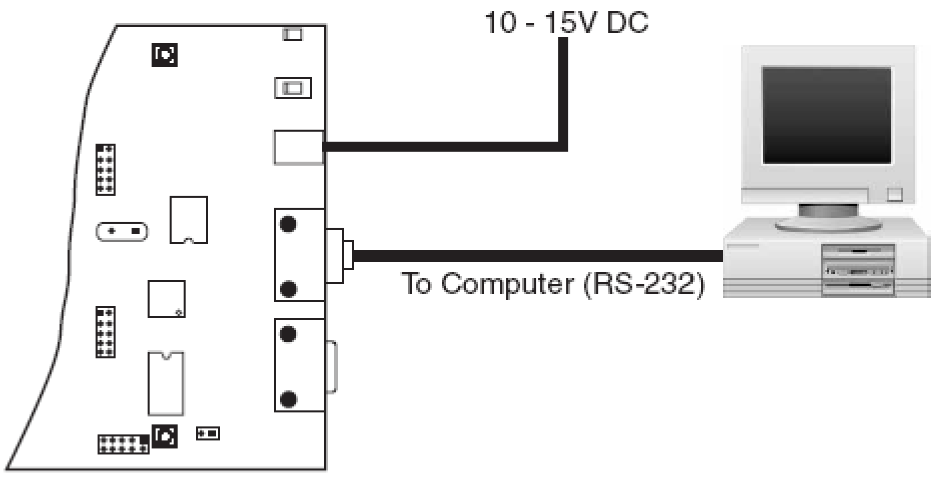
All the cables come with the kit (plastic bag).

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).

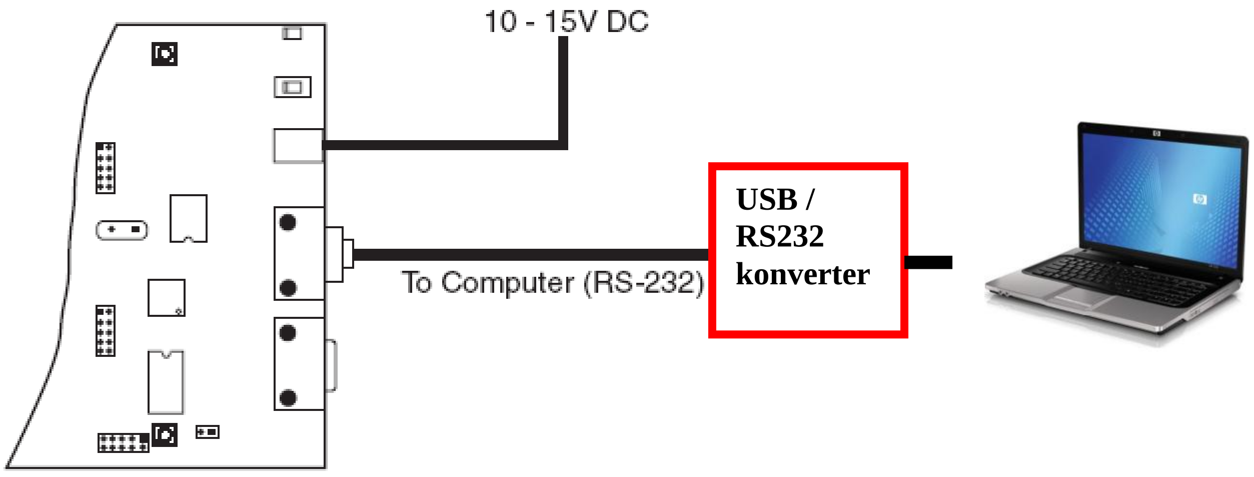


*If your PC **has no** COM port* (the most likely):

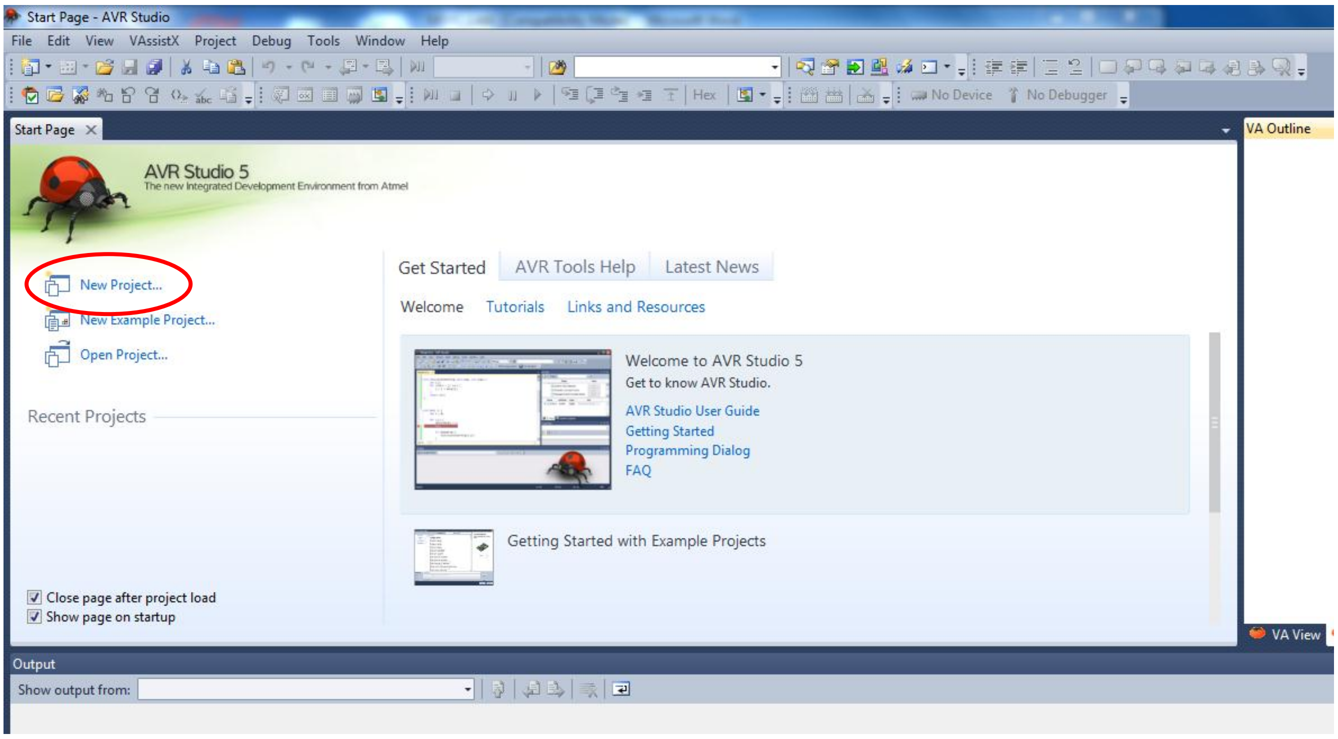
a. Connect the STK500 to a USB/RS232 converter using the serial cable. Connect the USB/RS232 converter to the PC (an arbitrary USB connector).

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.

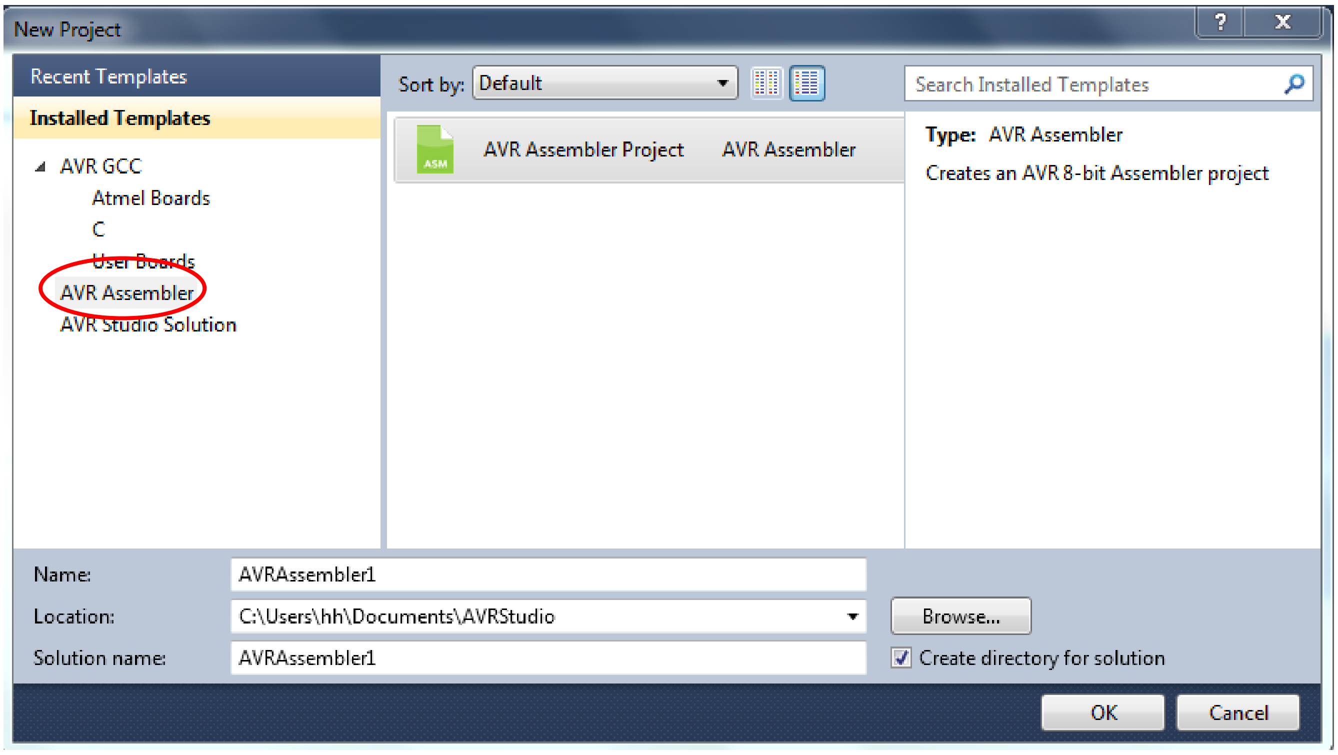
c. Apply power (10-15 volts, minimum 500 mA), and switch on the STK500 (power button).



Step 5: Start AVR Studio and create a new ASSEMBLY project
When AVR Studio starts up, this window is shown:



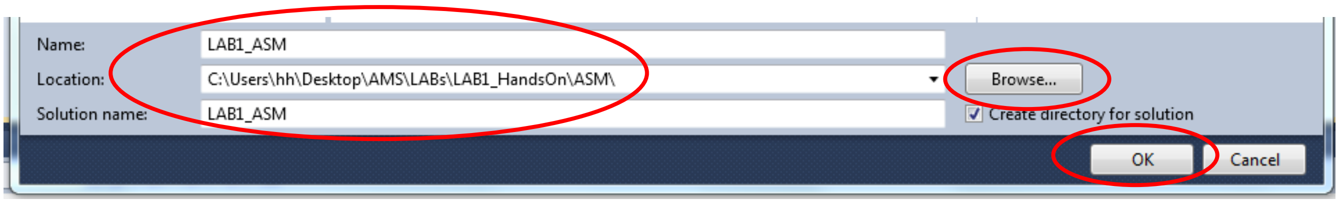
Click "New Project" and then click "ASM Assembler" (to the left):



Type: AVR Assembler
Creates an AVR 8-bit Assembler project

Name: AVRAssembler1
Location: C:\Users\hh\Documents\AVRStudio
Solution name: AVRAssembler1
☑ Create directory for solution

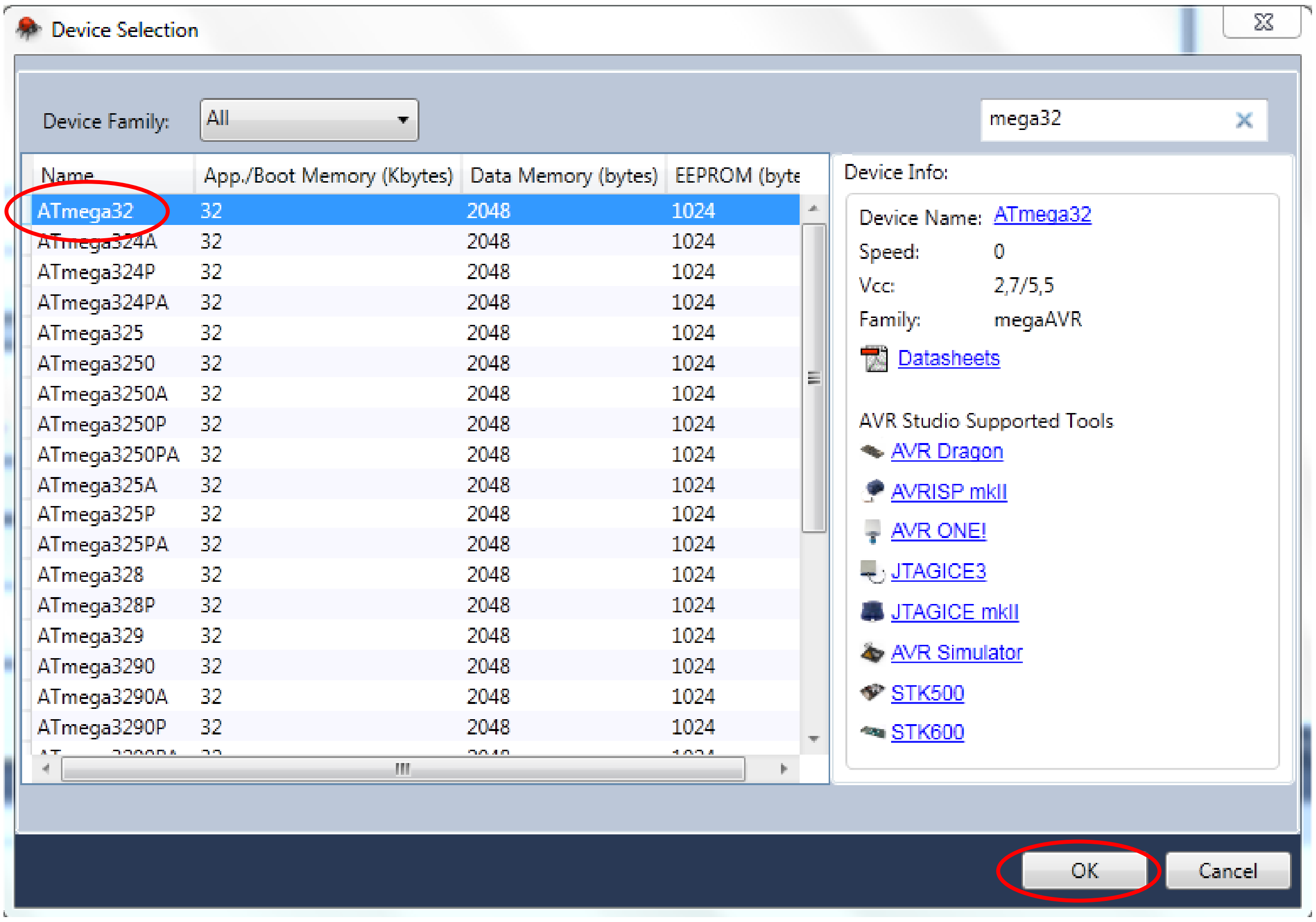
Step 6: Configure the project
In the fields for ”Name” and ”Solution Name” write ”**LAB1_ASM**”.
For the field ”**Location**” browse to the folder you want the project files to be stored.



Name: LAB1_ASM
Location: C:\Users\hh\Desktop\AMS\LABs\LAB1_HandsOn\ASM\
Solution name: LAB1_ASM
✓ Create directory for solution

Click ”OK”.

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



Device Selection

Device Family: All

mega32

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (byte
ATmega32	32	2048	1024
ATmega324A	32	2048	1024
ATmega324P	32	2048	1024
ATmega324PA	32	2048	1024
ATmega325	32	2048	1024
ATmega3250	32	2048	1024
ATmega3250A	32	2048	1024
ATmega3250P	32	2048	1024
ATmega3250PA	32	2048	1024
ATmega325A	32	2048	1024
ATmega325P	32	2048	1024
ATmega325PA	32	2048	1024
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega329	32	2048	1024
ATmega3290	32	2048	1024
ATmega3290A	32	2048	1024
ATmega3290P	32	2048	1024

Device Info:

Device Name:	ATmega32
Speed:	0
Vcc:	2,7/5,5
Family:	megaAVR

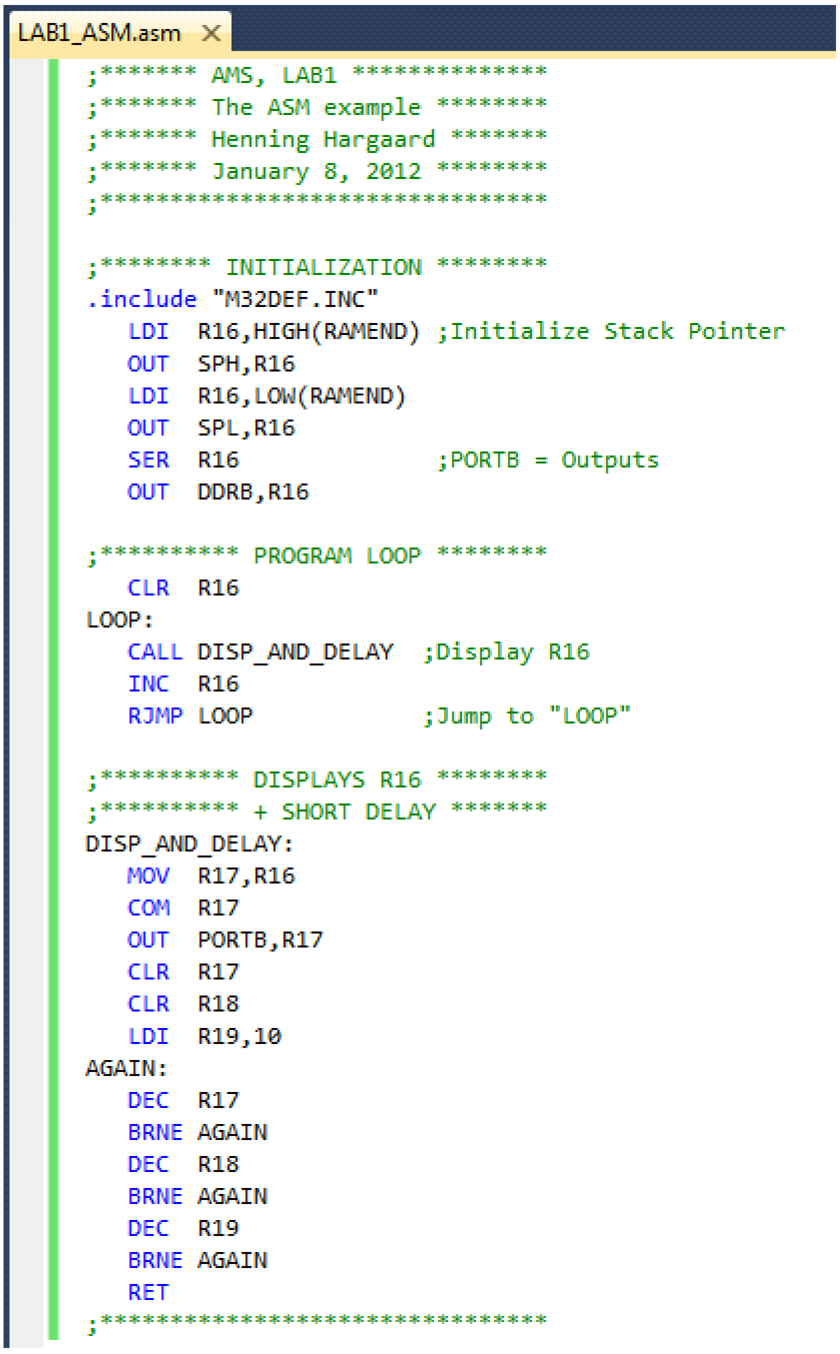
Datasheets

AVR Studio Supported Tools
- AVR Dragon
- AVRISP mkII
- AVR ONE!
- JTAGICE3
- JTAGICE mkII
- AVR Simulator
- STK500
- STK600

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 Campusnet (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 the Campusnet).


LAB1_ASM.asm

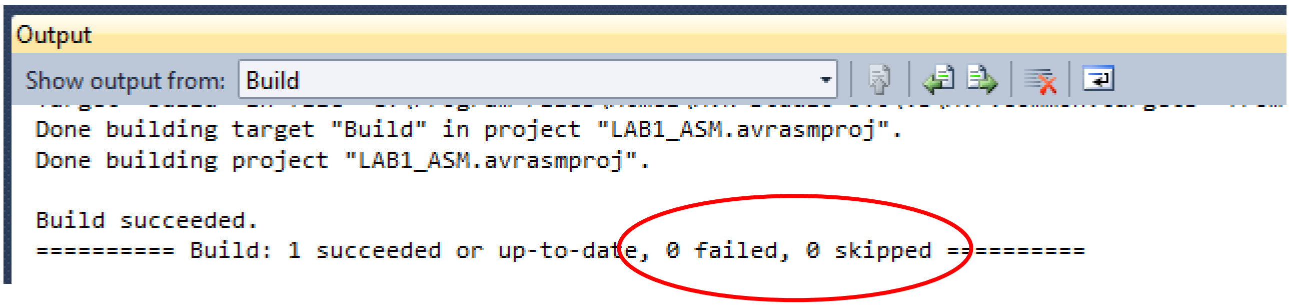
```
;******* AMS, LAB1 **************
;******* The ASM example ********
;******* Henning Hargaard *******
;******* January 8, 2012 ********
;********************************

;******** 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).

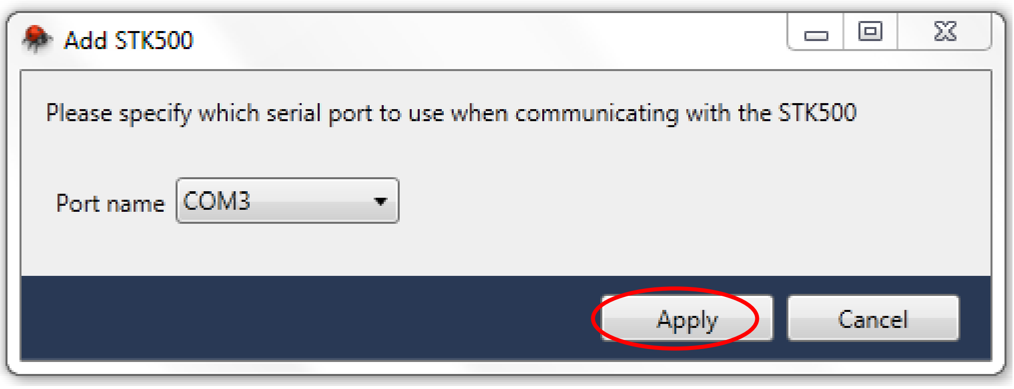
If there are no errors, the Output window will appear like this:



```
Done building target "Build" in project "LAB1_ASM.avrasmproj".
Done building project "LAB1_ASM.avrasmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

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".

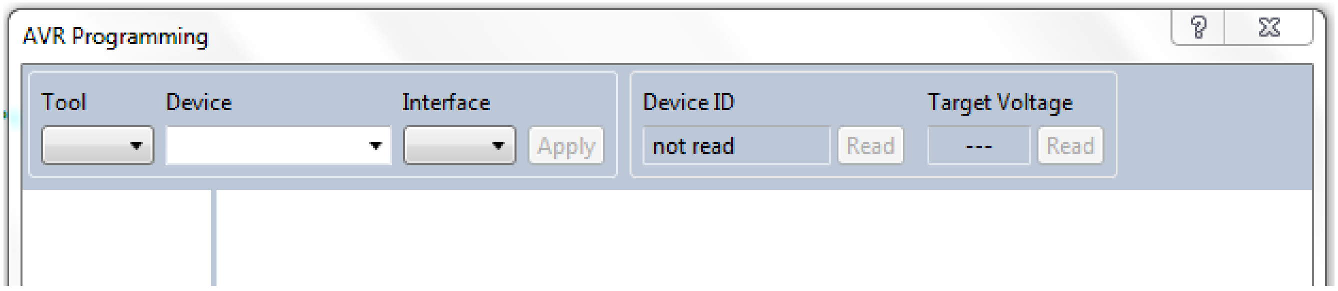

Add STK500

Please specify which serial port to use when communicating with the STK500

Port name COM3

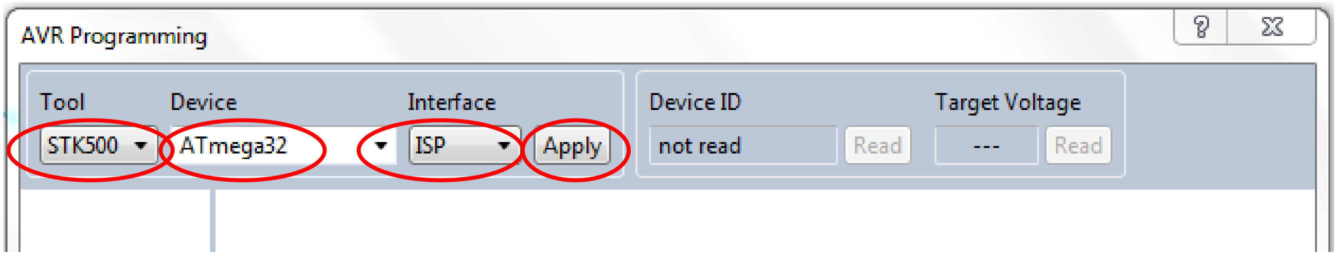
"Port Name" is the COM port connected to the STK500 kit (eventually use "Device Manager" to determine the COM port number used by your USB/serial adaptor).
Click "Apply".

Select "Tools" -> "AVR Programming":


AVR Programming

Tool	Device	Interface		Device ID		Target Voltage	
			Apply	not read	Read	---	Read

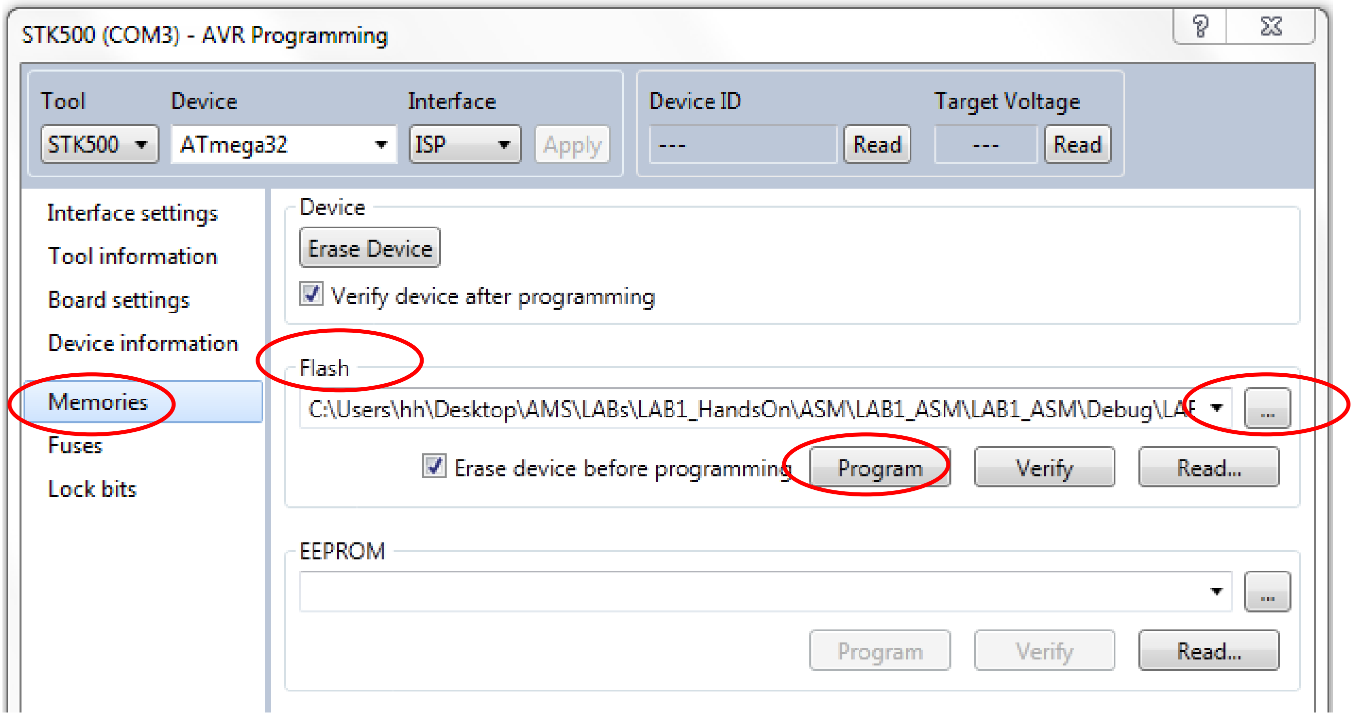
Now set "Tool" to "STK500" and "Device" to "ATmega32". "Interface" must be "ISP":


AVR Programming

Tool	Device	Interface		Device ID		Target Voltage	
STK500	ATmega32	ISP	Apply	not read	Read	---	Read

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

(To be continued next page)

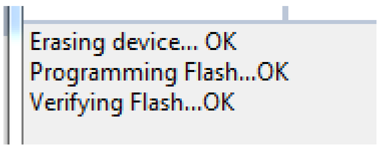


```
C:\Users\hh\Desktop\AMS\LABs\LAB1_HandsOn\ASM\LAB1_ASM\LAB1_ASM\Debug\LAE
```

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_ASM.HEX”. The file will normally be located in the folder named ”Debug”.

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

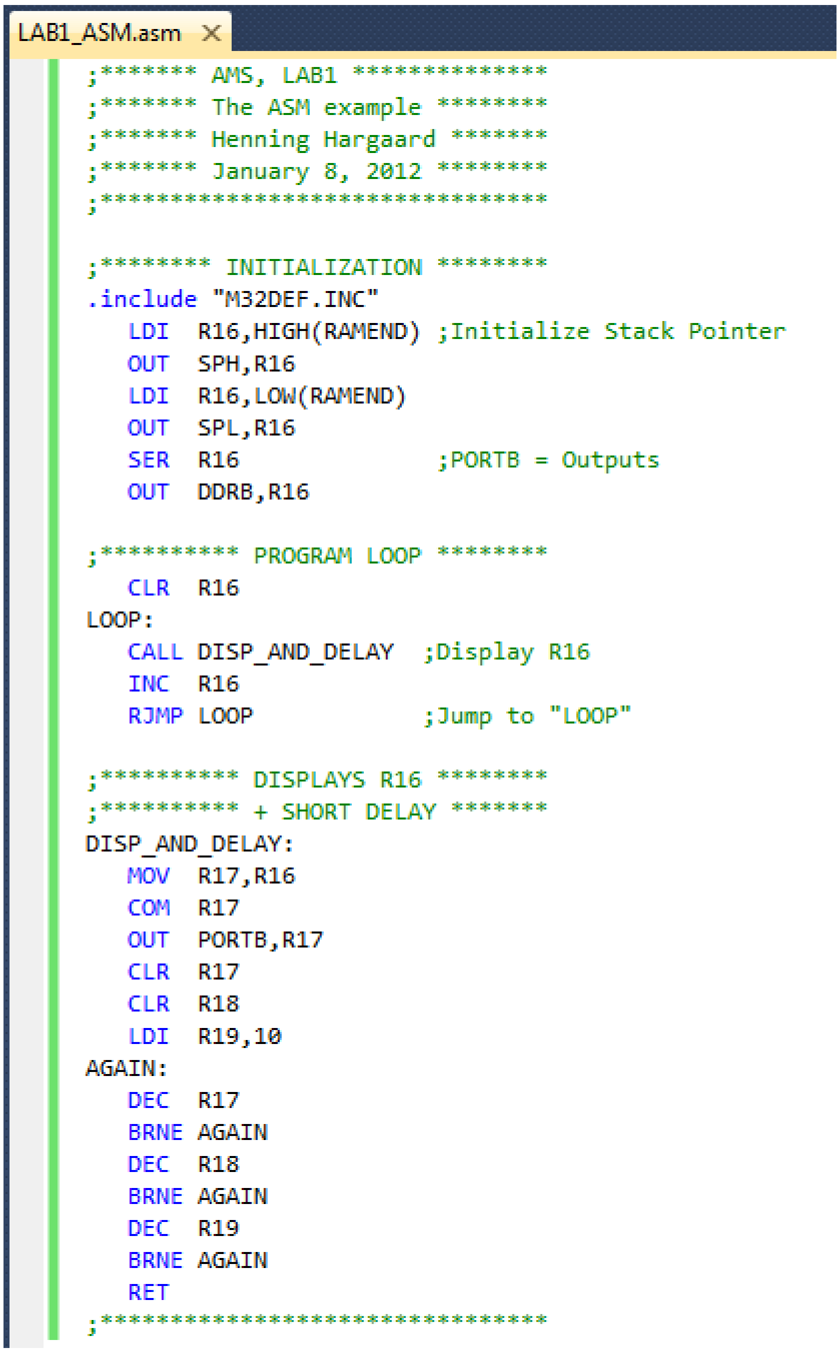
Now the program will be downloaded to the microcontroller at the STK500 (and the program automatically starts up):



```
Erasing device... OK
Programming Flash...OK
Verifying Flash...OK
```

Close the window ”AVR Programming”.

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



```
;******* 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
;********************************
```

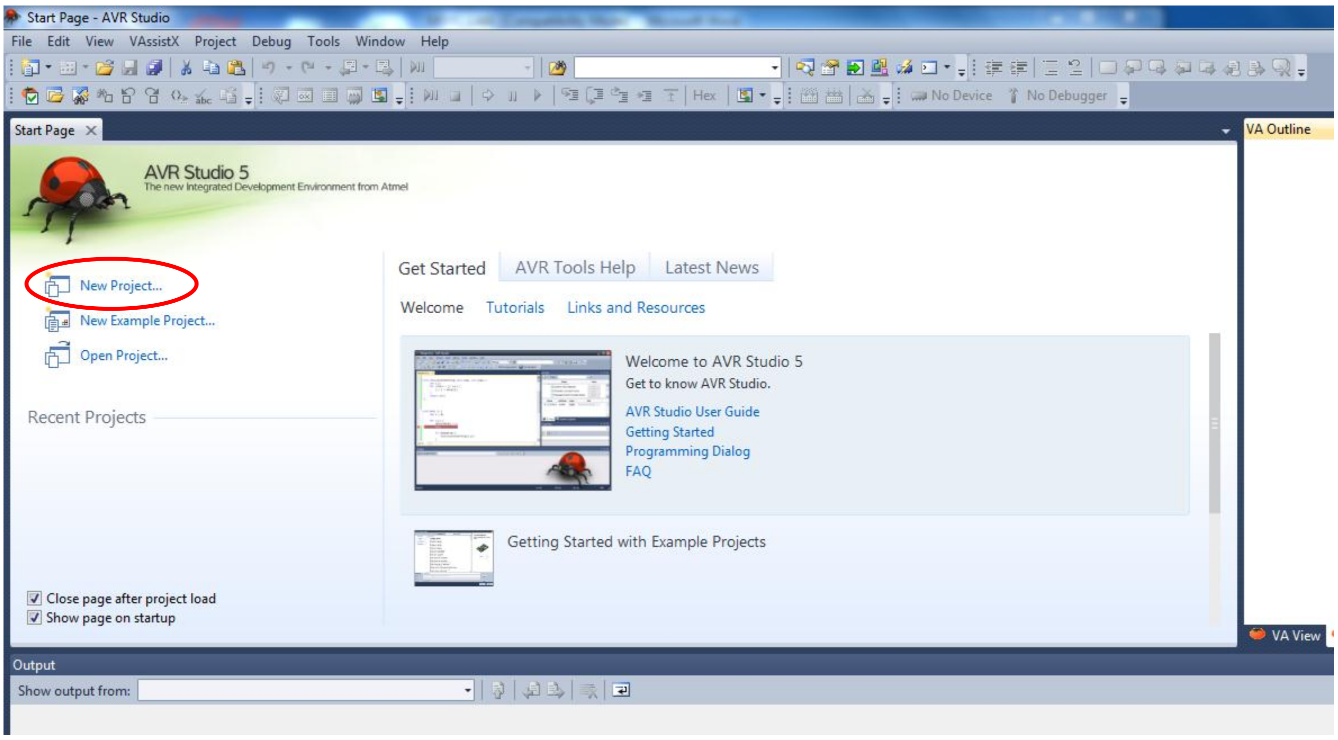
Step 11: Make changes to the program
When you have a good understanding of how the program functions, try doing minor changes to the program. Recompile ("Build") the program, download the code ("Tools" -> "AVR Programming") and test the changes.

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

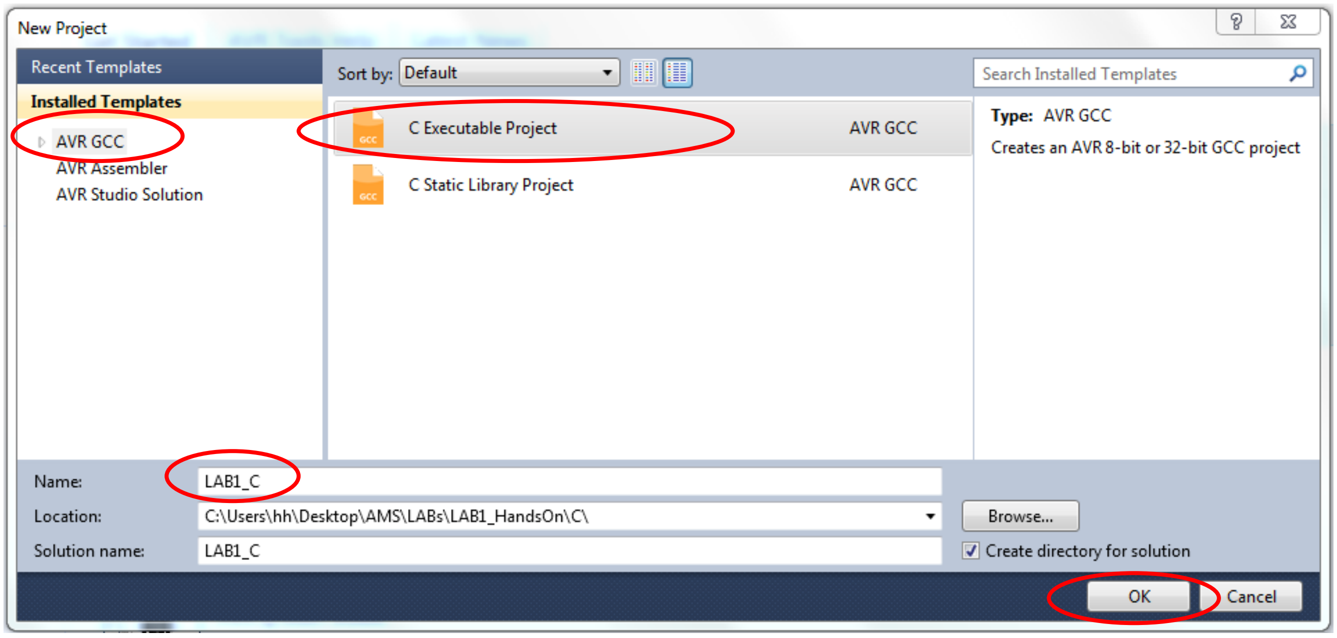
Step 12: Start AVR Studio and create a new AVR GCC project

In the following steps we will demonstrate how to create an AVR GCC project for a C program.

Restart AVR Studio:



Click "New Project" and then select "AVR GCC" (to the left).
Name the project "LAB1_C" and select a proper location.



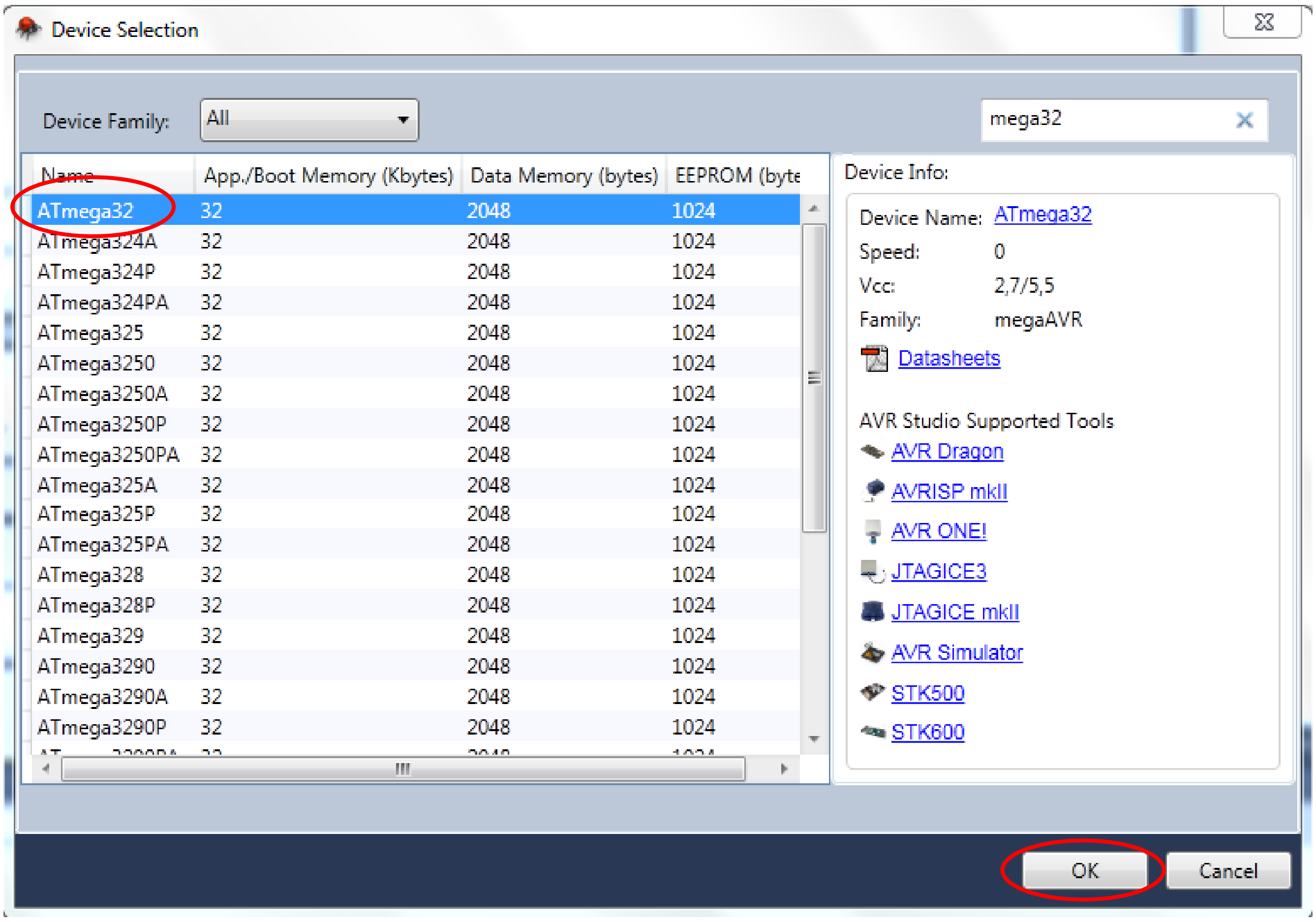
Type: AVR GCC
Creates an AVR 8-bit or 32-bit GCC project

Name: LAB1_C
Location: C:\Users\hh\Desktop\AMS\LABs\LAB1_HandsOn\C\
Solution name: LAB1_C
☑ Create directory for solution

Click "OK".

Step 13: Select device

Select the microcontroller ”**ATmega32**”.



Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (byte
ATmega32	32	2048	1024
ATmega324A	32	2048	1024
ATmega324P	32	2048	1024
ATmega324PA	32	2048	1024
ATmega325	32	2048	1024
ATmega3250	32	2048	1024
ATmega3250A	32	2048	1024
ATmega3250P	32	2048	1024
ATmega3250PA	32	2048	1024
ATmega325A	32	2048	1024
ATmega325P	32	2048	1024
ATmega325PA	32	2048	1024
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega329	32	2048	1024
ATmega3290	32	2048	1024
ATmega3290A	32	2048	1024
ATmega3290P	32	2048	1024

Device Info:

Device Name:	ATmega32
Speed:	0
Vcc:	2,7/5,5
Family:	megaAVR

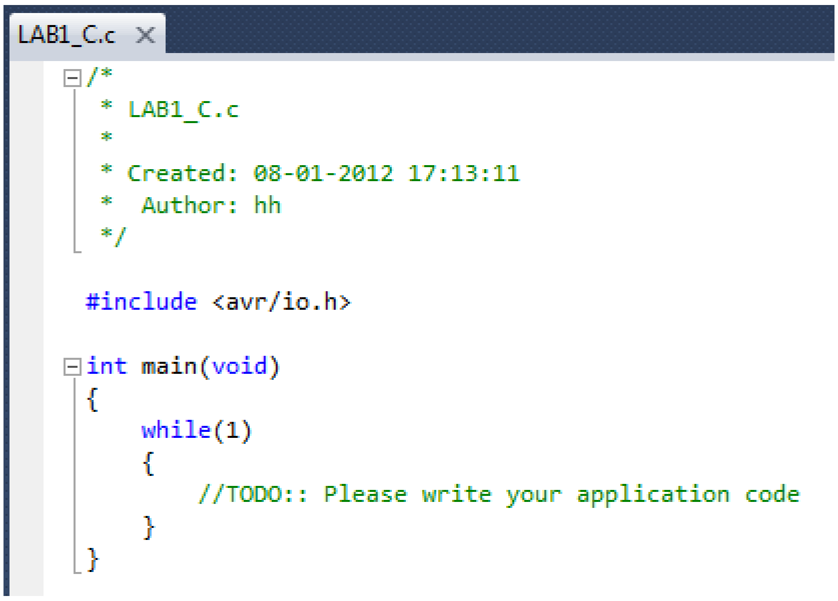
Datasheets

AVR Studio Supported Tools
- AVR Dragon
- AVRISP mkII
- AVR ONE!
- JTAGICE3
- JTAGICE mkII
- AVR Simulator
- STK500
- STK600

Click ”OK”.

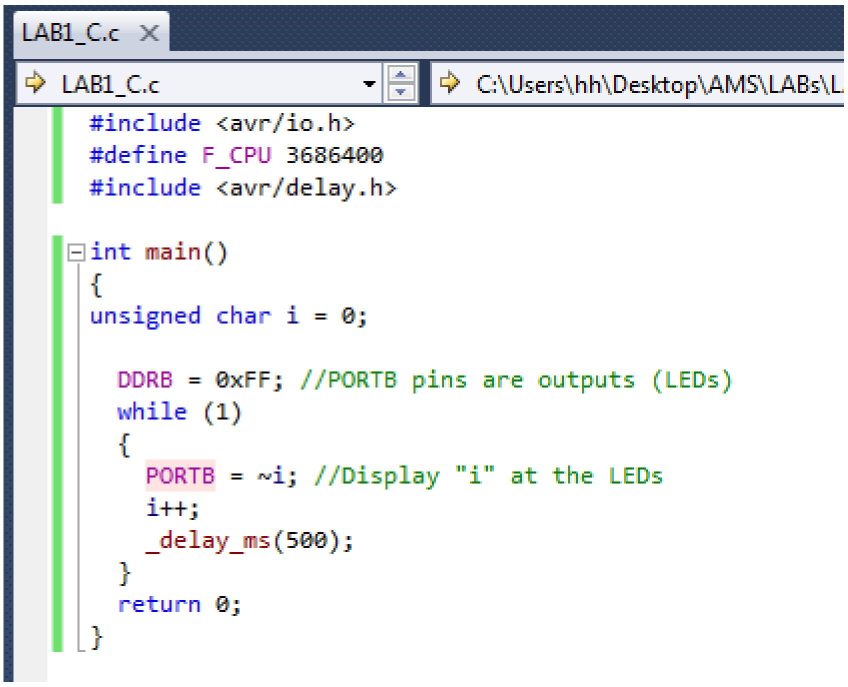
The project will be created in the folder, you selected.

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



```
/*
 * LAB1_C.c
 *
 * Created: 08-01-2012 17:13:11
 *  Author: hh
 */

#include <avr/io.h>

int main(void)
{
    while(1)
    {
        //TODO:: Please write your application code
    }
}
```

Step 14: Write the C program
At the AMS Campusnet (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 the Campusnet).



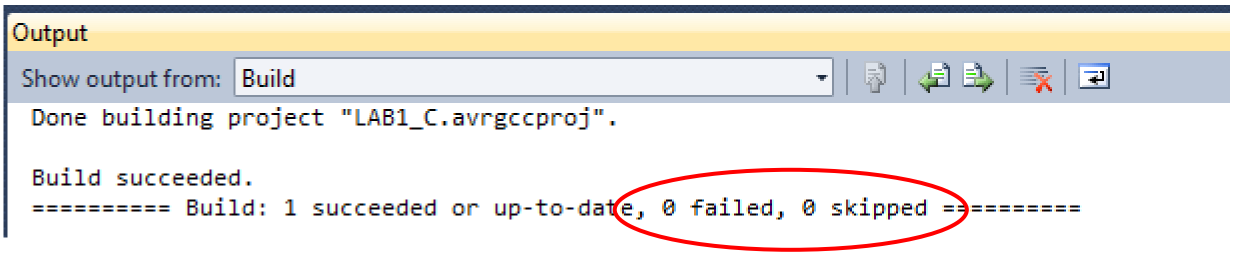
```c
#include <avr/io.h>
#define F_CPU 3686400
#include <avr/delay.h>

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:

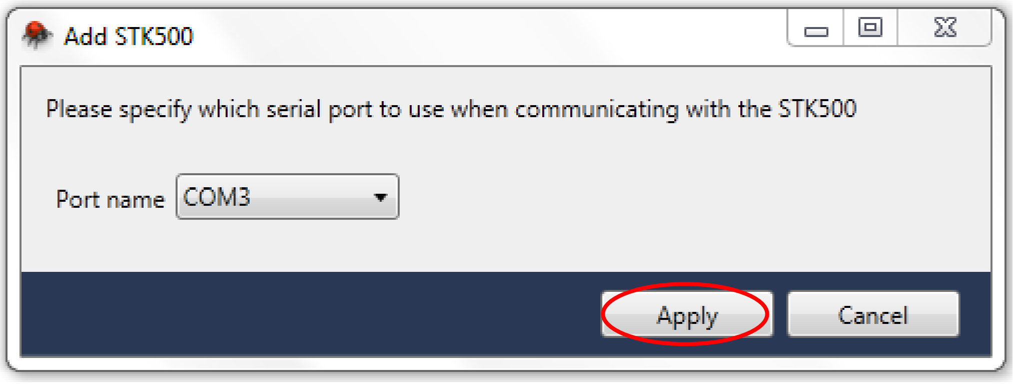


```
Done building project "LAB1_C.avrgccproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

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

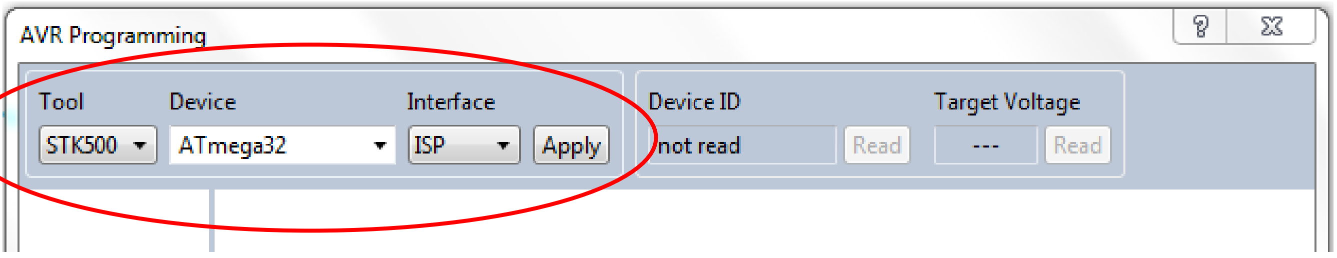
Step 16: Connect to STK500 and download the program
This process is similar to the instructions given in step 9 (eventually read this again).

<u>VERY IMPORTANT</u>: 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").

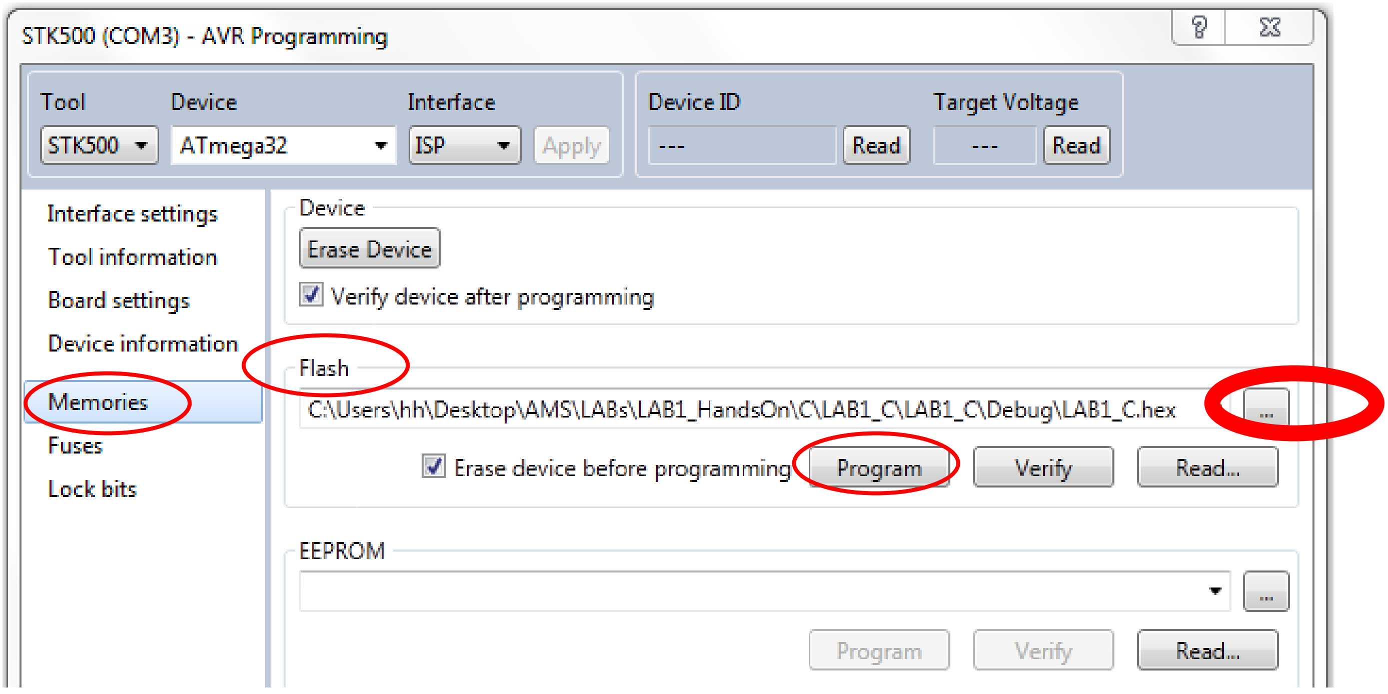

Add STK500

Please specify which serial port to use when communicating with the STK500

Port name: COM3


AVR Programming

Tool: STK500 | Device: ATmega32 | Interface: ISP | Device ID: not read | Target Voltage: ---


STK500 (COM3) - AVR Programming

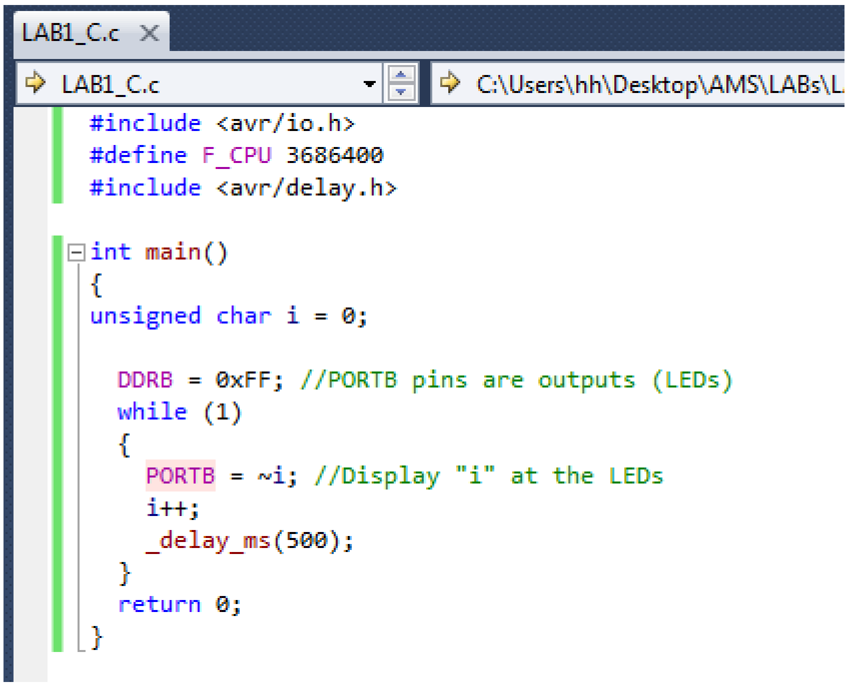
Tool: STK500 | Device: ATmega32 | Interface: ISP | Device ID: --- | Target Voltage: ---

☑ Verify device after programming

Flash: C:\Users\hh\Desktop\AMS\LABs\LAB1_HandsOn\C\LAB1_C\LAB1_C\Debug\LAB1_C.hex

☑ Erase device before programming

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



```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.