

HowTo... configure the AVR32 Studio to run programs on a Windows Host System (DOS Box) for test and transfer them to the NGW100

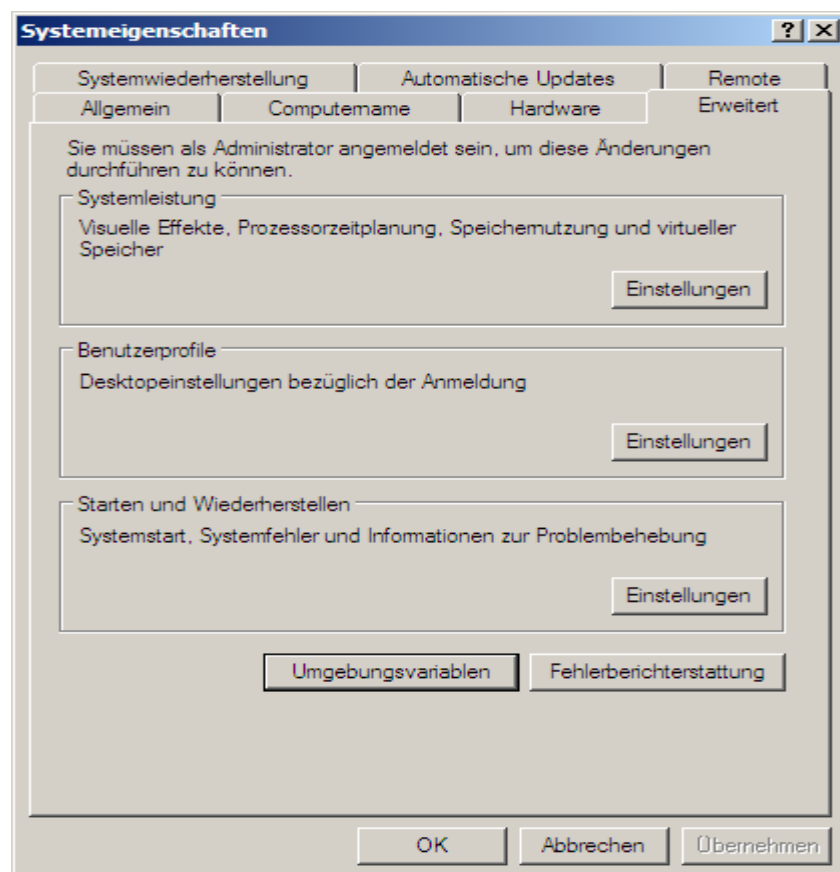
This HowTo describes how to configure the AVR32 Studio to run programs on a Windows Host System in a DOS Box. This configuration is useful to test programs and algorithms on a windows machine before transferring the program to the NGW100.

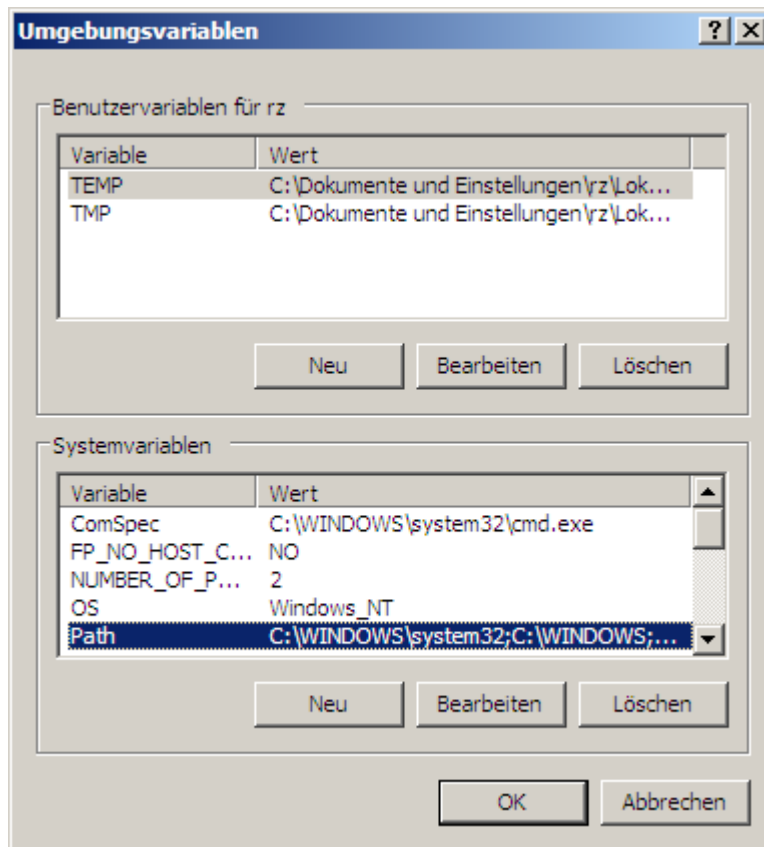
The procedure to configuring the Host system in a proper way is as follows:

- Set the PATH Variable of the windows machine to find the necessary Libraries (e.g. cygwin.dll)
- Create a project type "Managed Make C Project"
- Create a project "Managed Make AVR32 C Project"
- Write and evaluate (compile) and run the source code as "Managed Make C Project"
- Transfer the tested Source Codes to the "Managed Make AVR32 C Project"
- Compile and build the .elf file
- Transfer the .elf file to NGW100

Set the PATH Variable of the windows machine to find the necessary Libraries (e.g. cygwin.dll)

For setting up the right PATH variable on your windows machine you have to find the point where to configure it. As I'm a "Sauerkraut" (German guy), the screen shots are based on a german language setting. I work on Windows XP so the screenshot will reflect this. Here we go:





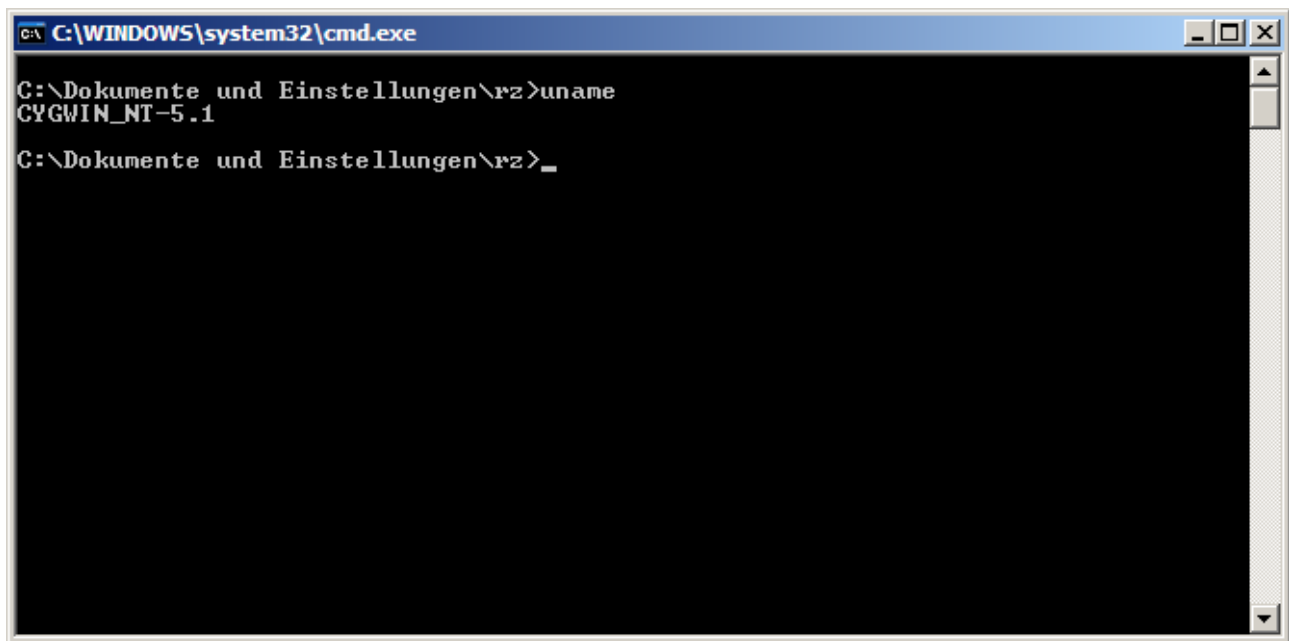
Find the PATH Settings on your System and add – for example - :

;C:\Programme\FlexBison;C:\cygwin\bin

PATH setting “C:\cygwin\bin”:

To explain, the Path **C:\cygwin\bin** shows to a directory where all the Cygwin .dlls and unix programs (the C-compiler, linker and so on) are installed on my machine (I think to remember this is the installation default path).

Also you find some other (useful) unix programs in this directory which can be run on Windows now. For example, the unix command “uname” replies in the DOS box:



```
C:\WINDOWS\system32\cmd.exe
C:\Dokumente und Einstellungen\rz>uname
CYGWIN_NT-5.1
C:\Dokumente und Einstellungen\rz>_
```

If you are ready with this you are able to compile and run your own C- Programs on the windows machine.

PATH setting “C:\Programme\FlexBison”:

Important note: To get your C Programs run on windows you don't need to install Flex++ and Bison++.

For interest and to explain what the setting does, the Path **C:\Programme\FlexBison** shows to a directory, where i put my “Lex” and “Yacc” compatible program generators in.

“Flex++” corresponds to Unix “Lex” Program, which is able to run on native Windows machines.

“Bison++” corresponds to Unix “Yacc” Program, which is able to run on native Windows machines.

It can be downloaded here: http://alzt.tau.ac.il/~michaelg/Downloads_Win/Development/

The file set is zipped as file flex++bison++.zip.

Flex++ is able to generate C and C++ code, which scans input data files based on regular expression you programmed. To say it simple you are able to scan a text file and make something with it (find or count words, or sentences, transfer html statements into letters – turn “#;65” into “A” and so).

Yacc++ is able to generate C and C++ code, which is able to perform a program sequence based on grammar rules (semantic scan). To say it simple you are able to define your own language like:

```
FOR i = 1 TO 10 { NGWPRINT “Hello”};
```

or to give a more sophisticated example:

meters = 2,0

GO FORWARD meters;

TURN RIGHT 90;

FOR degree = 0 TO 360 STEP 10

{MAKE Photo;

BLINK};

TURN LEFT 90;

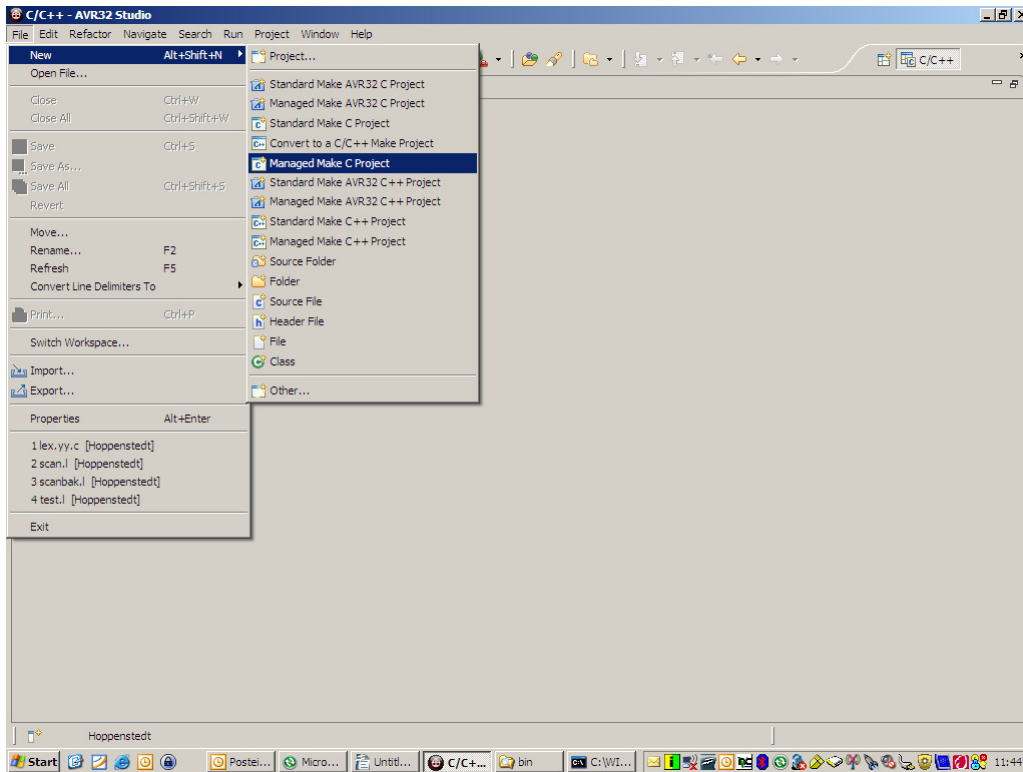
GO BACKWARD meters;

(for an NGW100 which was set up automotive with a controlled photo camera and LEDs)

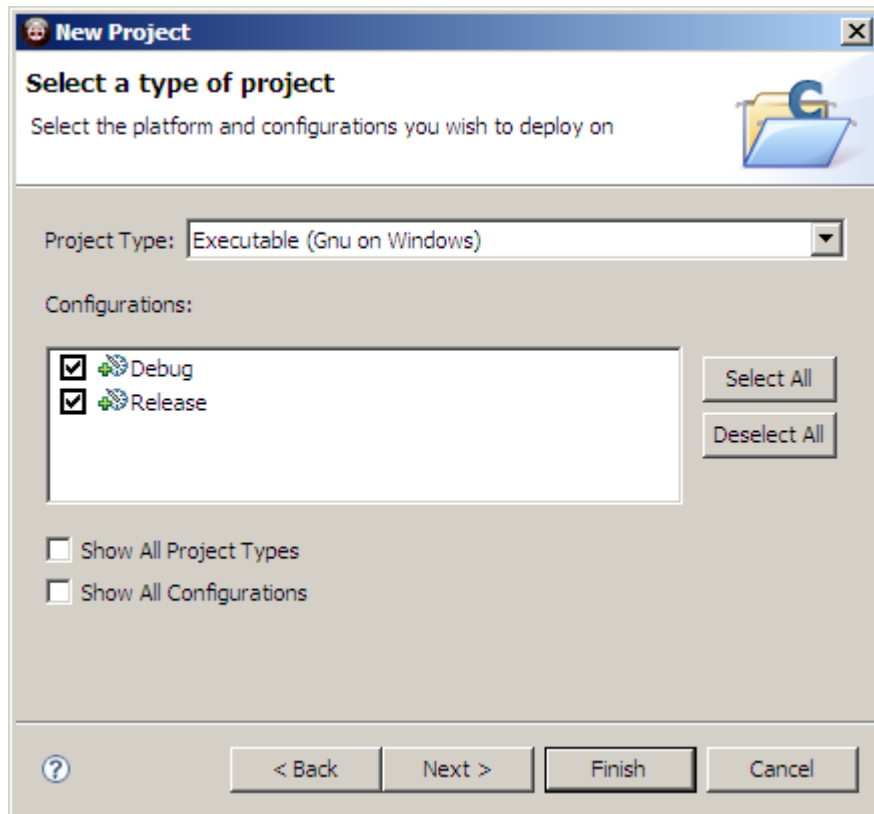
Creating a project “Managed Make C Project”

To show how we can create a program which runs on a windows machine (with a unix compiler/Linker!). We can arrange this by setting up the AVR32 Studio.

The next steps are pretty easy. Set up Project Type and name the Project like “PName_win”:



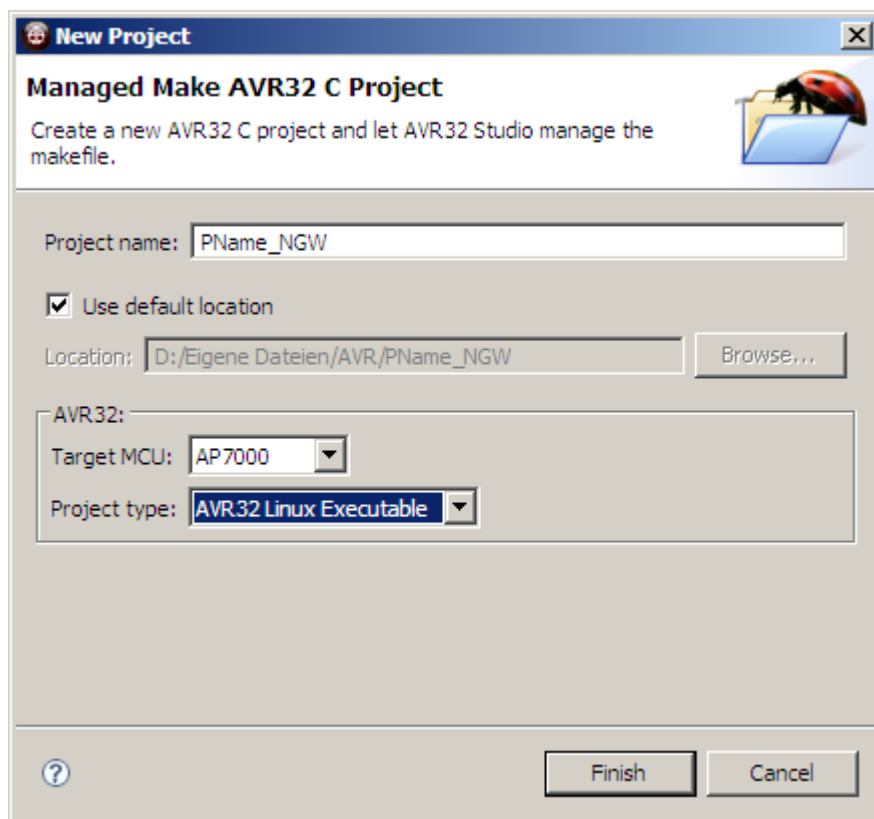
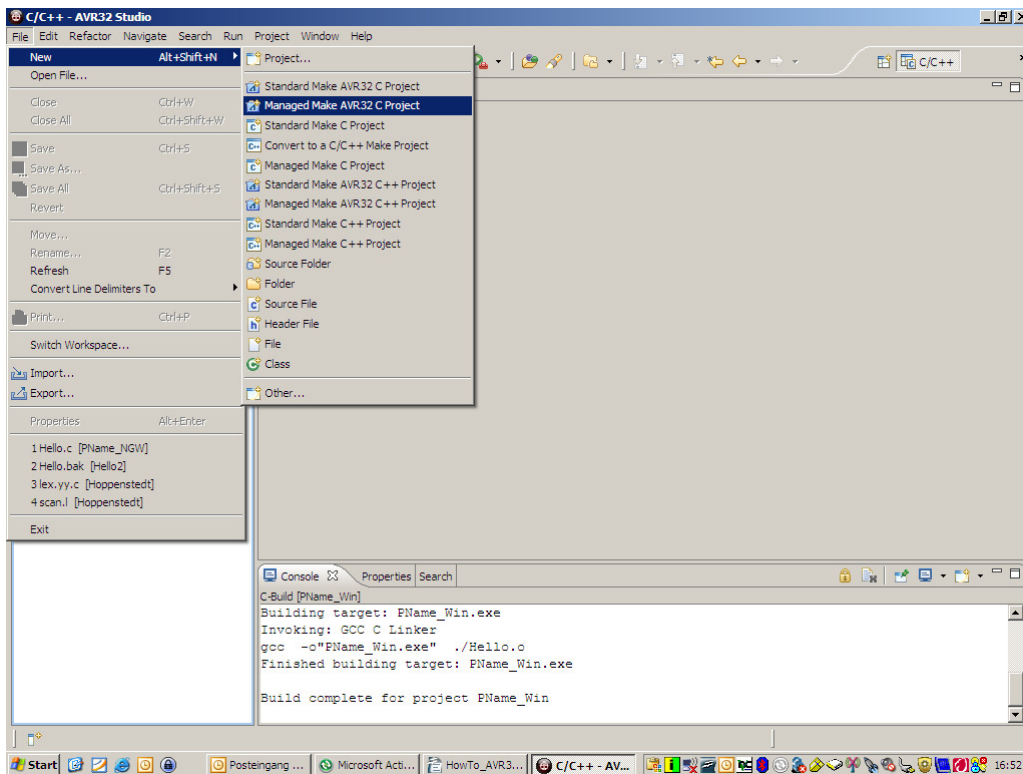
This should be the default setting:



Creating a project “Managed Make AVR32 C Project”

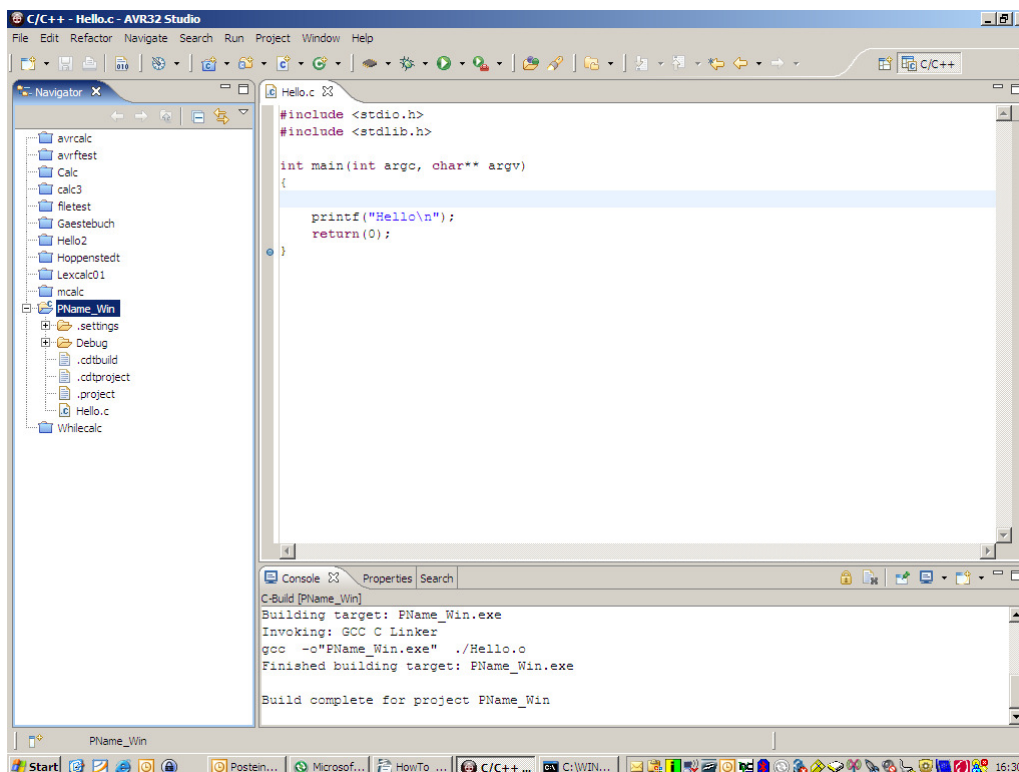
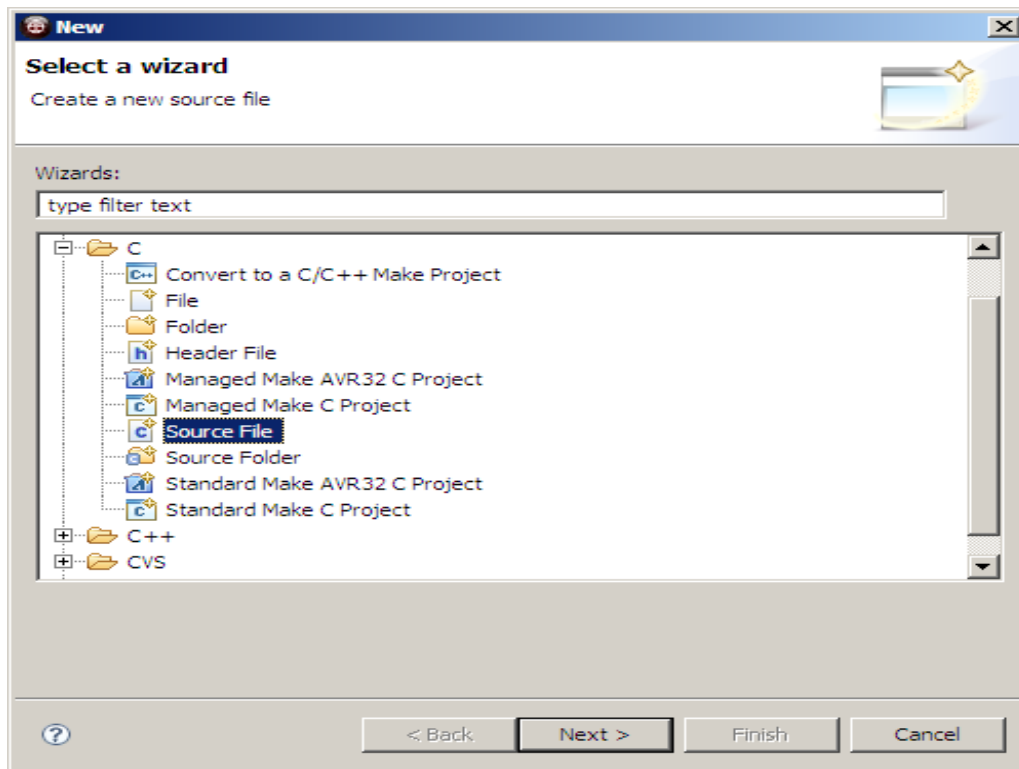
The same procedure we do now with the “Managed Make AVR32 C Project”:

Name the Project like “PName_NGW”:

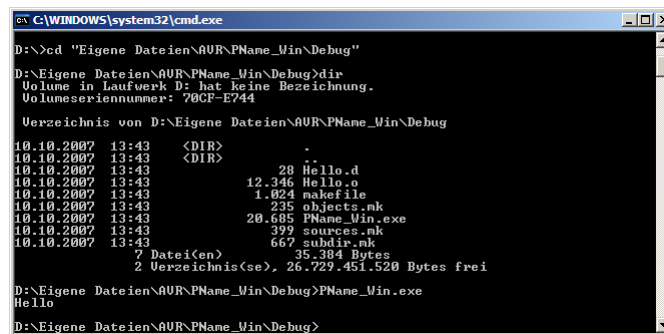


Write and evaluate the source code in “Managed Make C Project”

After setting up a New C source file you can compile the program via the build project feature (example “Hello.c”).



After Compilation we test the executable by opening a command line, change to the directory the program is resided in and call it:



```
C:\WINDOWS\system32\cmd.exe
D:\>cd "Eigene Dateien\AUR\PName_Win\Debug"
D:\Eigene Dateien\AUR\PName_Win\Debug>dir
Volume in Laufwerk D: hat keine Bezeichnung.
Volumennummer: 70CF-E744

Verzeichnis von D:\Eigene Dateien\AUR\PName_Win\Debug

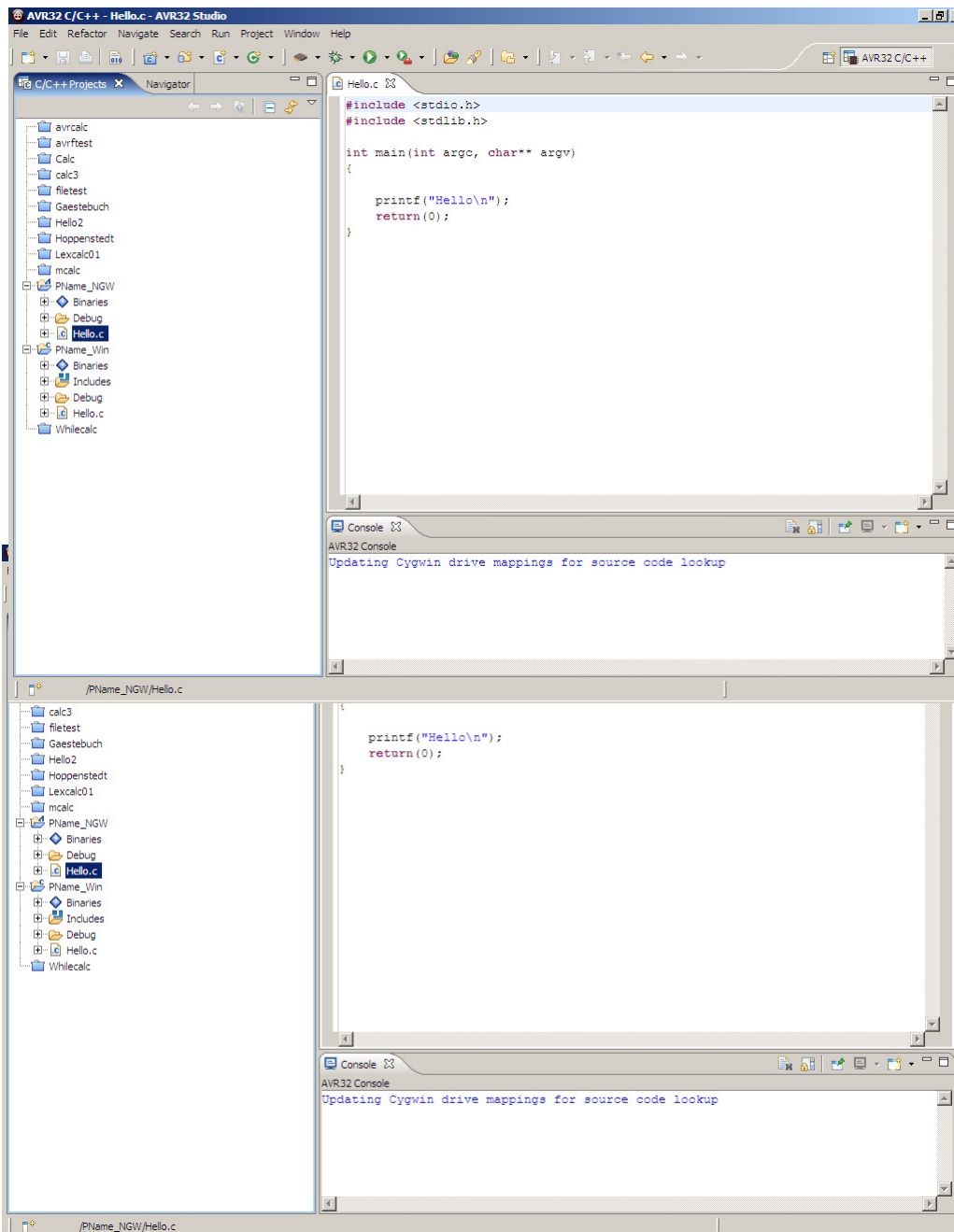
10.10.2007  13:43    <DIR>          .
10.10.2007  13:43    <DIR>          ..
               28 Hello.d
            12.346 Hello.o
               1.024 makefile
            235 objects.mk
            20.605 PName_Win.exe
               399 sources.mk
               667 subdir.mk
               7 Dateien
       2 Verzeichnis(se), 26.729.451.520 Bytes frei

D:\Eigene Dateien\AUR\PName_Win\Debug>PName_Win.exe
Hello

D:\Eigene Dateien\AUR\PName_Win\Debug>
```

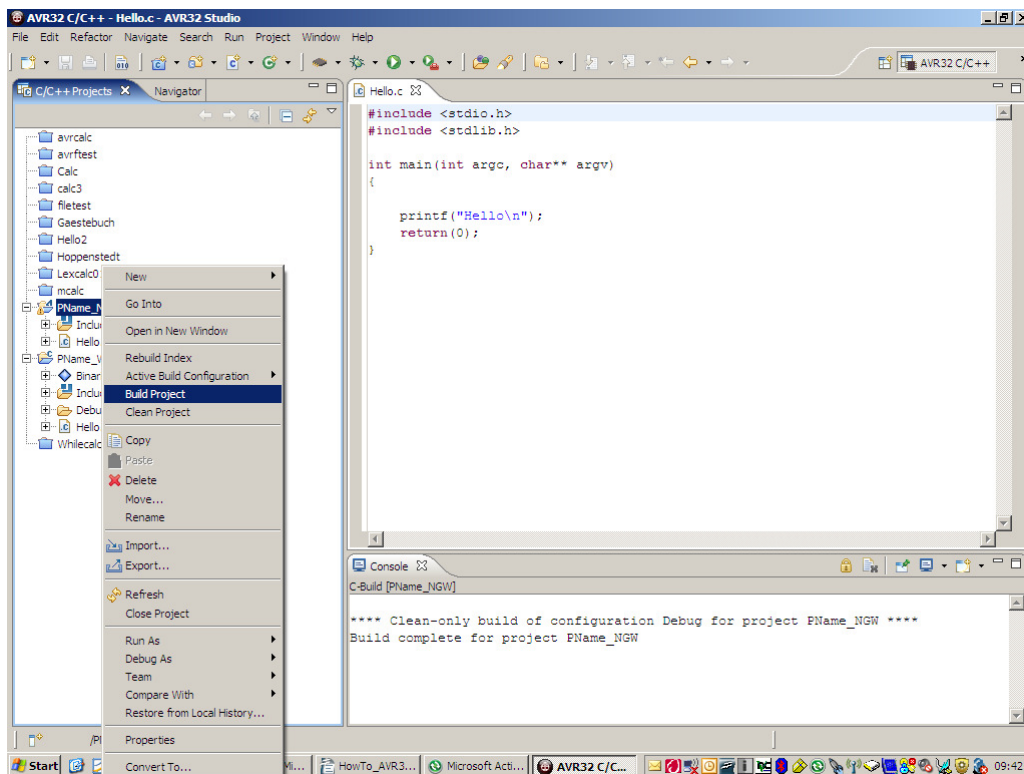
Hey, it works! Transfer the tested Source Codes to “Managed Make AVR32 C Project”

To transfer the Source File to the NGW directory just pick it via mouse and draw it from the Win – to the NGW directory or use the copy paste action. In the screen shot below I used copy/paste:

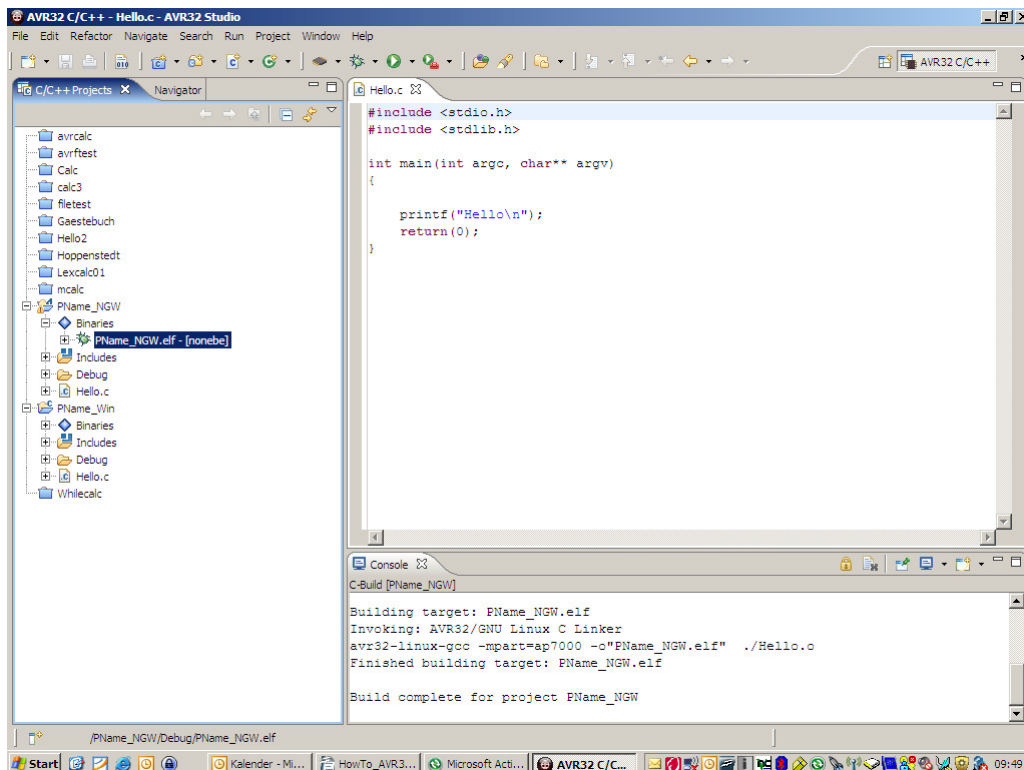


Compile and build the .elf file

Just build the project:



There is the .elf file:



Transfer the .elf file to NGW100

Here I need your help:

During my vacations I was not able to upload the files (.elf) to the NGW via the TFTP Server daemon. It just didn't work. Because this would be a very elegant way to transfer data between the NGW and the windows machine I would like to get the information from you, how I can set up the system.

I used a SD Card instead which I put in my PC, copied .elf on it and put it back to NGW.

I found that auto mount feature didn't work always so I had to reset NGW sometimes.

Over all I'm not happy with this way to transfer data, so I would be happy if you could help me, here.