21 Sep 2011 /TBS

FYS4220 / 9220

RT-lab no 1 - 2011

The ROBOT

1 Introduction

The documentation for this exercise is structured in two parts.

The first part describes i) the **VxWorks** systems in the lab, ii) organization of user accounts and file space, and iii) introduction to the Wind River Workbench 2.4 Development Suite.

The second part presents the RT-exercise no 1: the ROBOT. A baseline source is given. The code also includes a demonstration of **StethoScope**, which is a real-time graphical monitoring and data collection suite for VxWorks.

Important information for starting Workbench, see chapter 4.

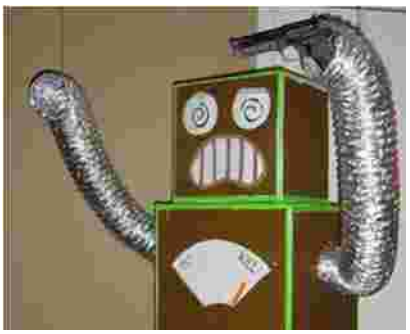
1.1 Lab journal and approval of lab exercises

The lab groups are the same as for the FPGA/VHDL exercises. The lab is located in room 318V.

The lab journal shall contain the source code plus any user defined header files, such that the lab supervisor can have the pleasure of running the program. Output (for instance a screen dump) from the program shall be included, together with a short note explaining the choice of implementation.

Deliver electronically to t.b.skaali@fys.uio.no . Do not forget the group ID for identification.

The deadline for approval of all exercises is 1 December.



PART 1

2 VxWorks and Workbench 2.4

VxWorks is an embedded Real-Time Operating System (RTOS) from Wind River http://www.windriver.com/ Based on the Eclipse platform, the Wind River **Workbench** is a collection of tools for embedded software development for VxWorks.

Workbench 2.4 with **VxWorks 6.2** is installed on two Windows XP host PCs in the lab room 318V. The computer names are *fys-lab-sci-01* and *fys-lab-sci-02*. Due to security restrictions both PCs are placed behind a firewall blocking traffic initiated from outside. However, normal terminal and file access from the PCs to external computers are allowed.

VxWorks is described in the RTOS lecture. S/W development for VxWorks is done on a "host" machine running Linux, Unix, or Microsoft Windows, cross-compiling target software to run on various "target" CPU architectures. The target architecture specific code is called a Board Specific Package (BSP). At boot time the VxWorks kernel is downloaded to the target. User code is dynamically linked to the kernel at download time. Host and target are connected over Ethernet, but also a serial link can be used.
In addition to architecture dependent BSPs, a generic target simulator **vxsim** is included in the Workbench suite. Software which is not architecture specific can be executed under **vxsim**.

FYS 4220 Workbench configurations:

fys-lab-sci-01 ***target #1***: vxsim0
fys-lab-sci-02 ***target #1***: vxsim0
target #2: M5000 VME module, "tgt_192.168.0.12" (IP-address)
M5000 is a VME Single Board Computer (SBC) from VMETRO (now Curtiss Wright) with a PPC440 processor.
The BSP for this target is has been supplied by VMETRO.
The terminal port of M5000 is connected to VxWorks COM1.

Additional target simulators can be added, they will be named vxsim1, vxsim2 and so on.

For lab exercises which does include the M5000 module it is recommended to use the **vxsim** simulator. (A bonus is that one does not have to listen to the fan noise from the VME crate).

3 User accounts

Individual group accounts for the lab course have been defined on the two host PC's. The groups are the same as for the FPGA/VHDL lab.

fys-lab-sci-01: accounts GR1, GR2, GR3, GR4, GR5, GR11
fys-lab-sci-02: accounts GR6, GR7, GR8, GR9, GR10, GR12

Passwords are distributed to the groups. Only the owner of a group account can access their own files under ***My Documents*** (assuming that you don't give your password away).

An identical baseline directory structure has been set up for all accounts, shown below is the structure for GR5. (Ignore that disk address is D: and not C: and that the first part of the path may differ from that on the host PCs, because the structure has been generated on another XP platform).

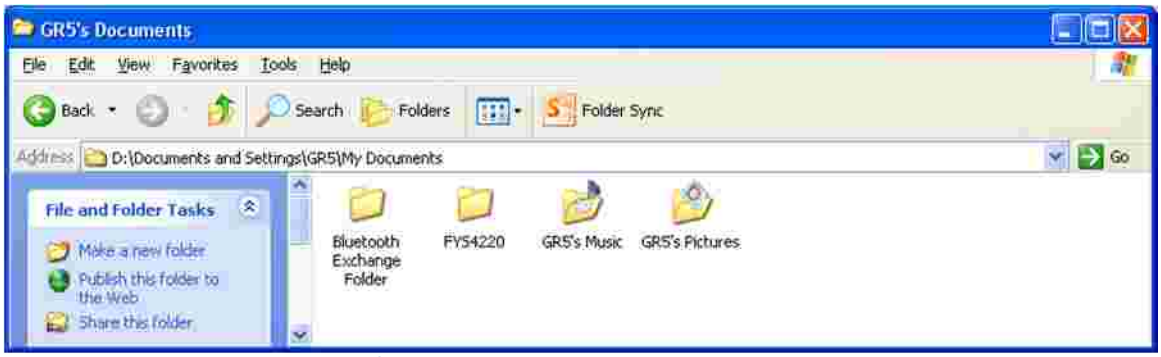


Figure 1 Baseline directory structure for FYS4220 lab group

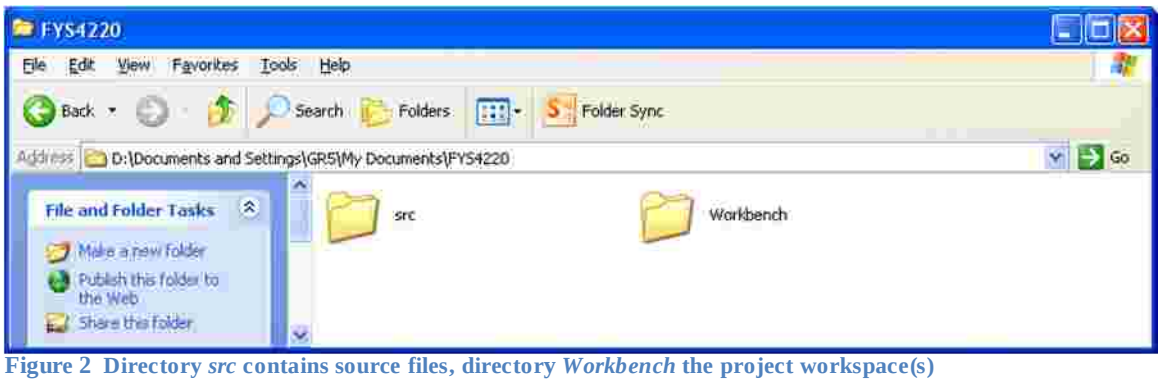


Figure 2 Directory *src* contains source files, directory *Workbench* the project workspace(s)

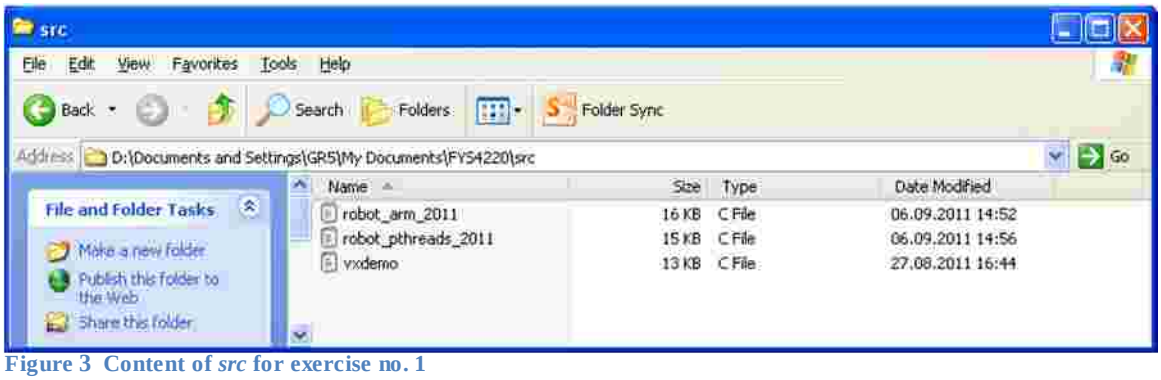


Figure 3 Content of *src* for exercise no. 1

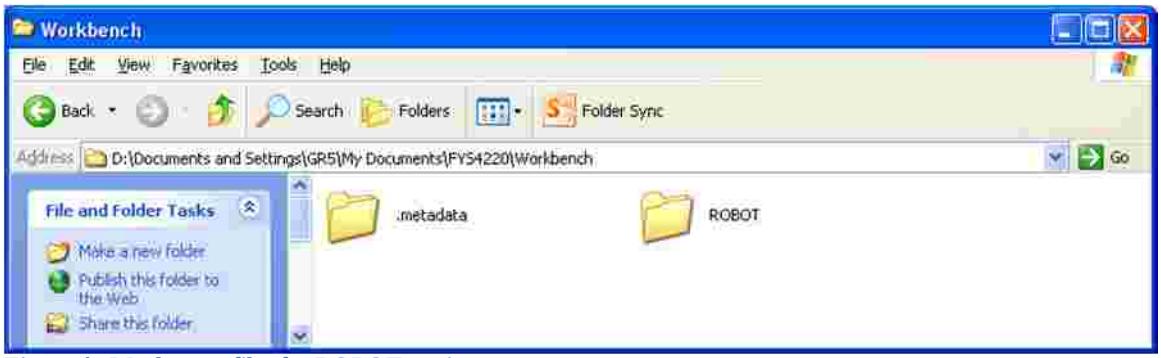


Figure 4 Workspace files for ROBOT project

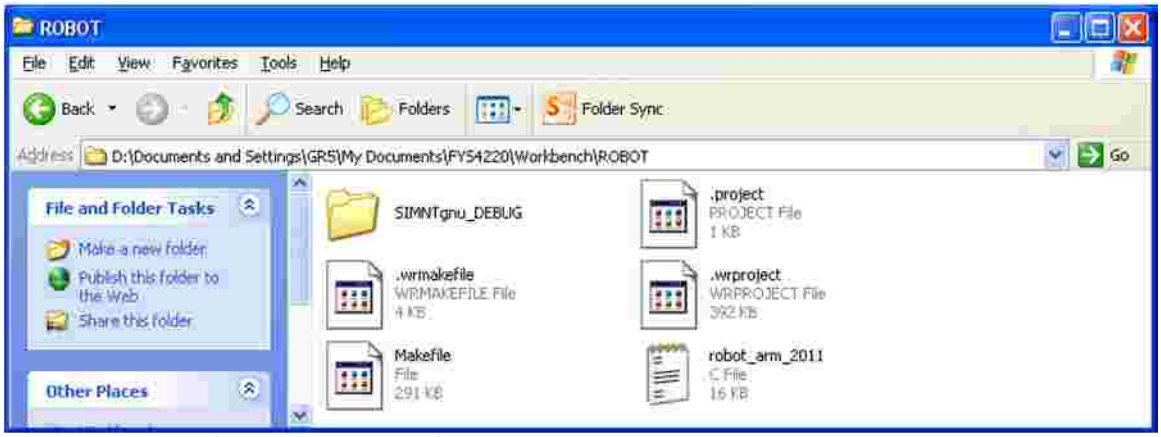


Figure 5 Content of ROBOT workspace for vxsim target with gnu compiler

Note! The source code (C File) stored in a workspace is an imported copy of the original stored elsewhere. All editing under Workbench is done on the workspace copy only. There is no automatic write-back of source files that have been modified. When a project is deleted, one can specify if the workspace files shall be deleted or not.

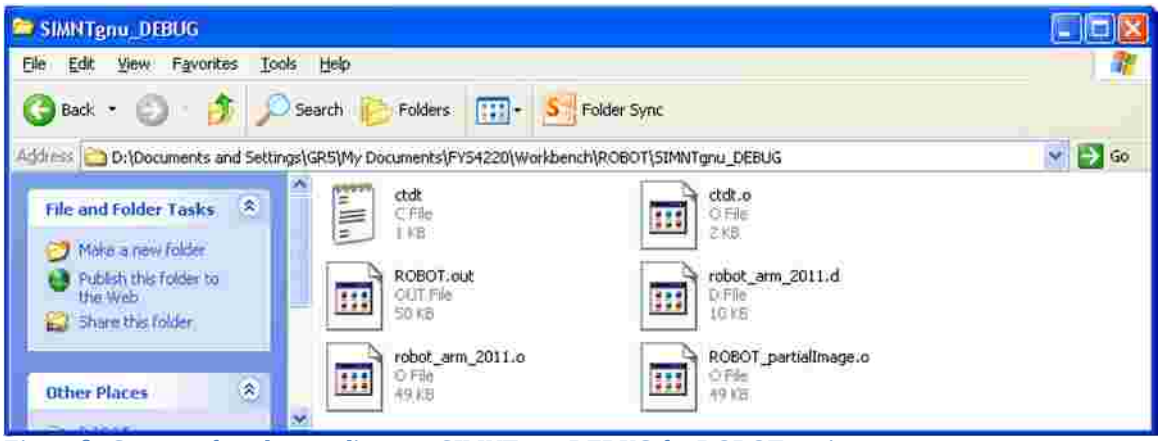


Figure 6 Content of workspace directory SIMNTgnu_DEBUG for ROBOT project

4 Workbench 2.4

The Workbench GUI is based on the ECLIPSE software development environment, and most students are familiar with this environment.

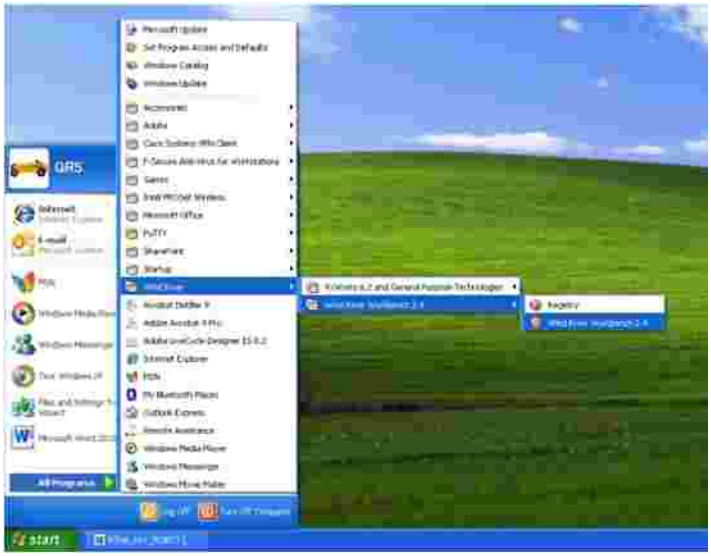


Figure 7 Startup of Workbench 2.4

Workbench is started from the Wind River program menu, see Figure 7. Start first ***Registry*** for host-target communication if it is not already running. The ***Registry*** symbol on the Taskbar Registry is

shown here.

***Registry* must be started manually since the lab accounts GR1 ,---- GR12 are of type *Limited*, i.e. no administrative privileges. Logout from a lab account will shutdown *Registry*. However, an exit from Workbench will not stop *Registry*, so one can restart Workbench with a running *Registry*. When the session is over terminate with *Log Off* and not *Switch User* !**

Skip Chapter 4.1 if you are only going to work on **vxsim**.

4.1 Booting the VME Single Board Computer M5000 target

On **fys-lab-sci-02,** follow the link *VxWorks 6.2 and General Purpose Technologies* for starting COM1 if it is not already active. The serial port of M5000 is connected to COM1.

Power up the VME crate, or RESET the M5000 module. The boot COM1 screen is shown on Figure 8.

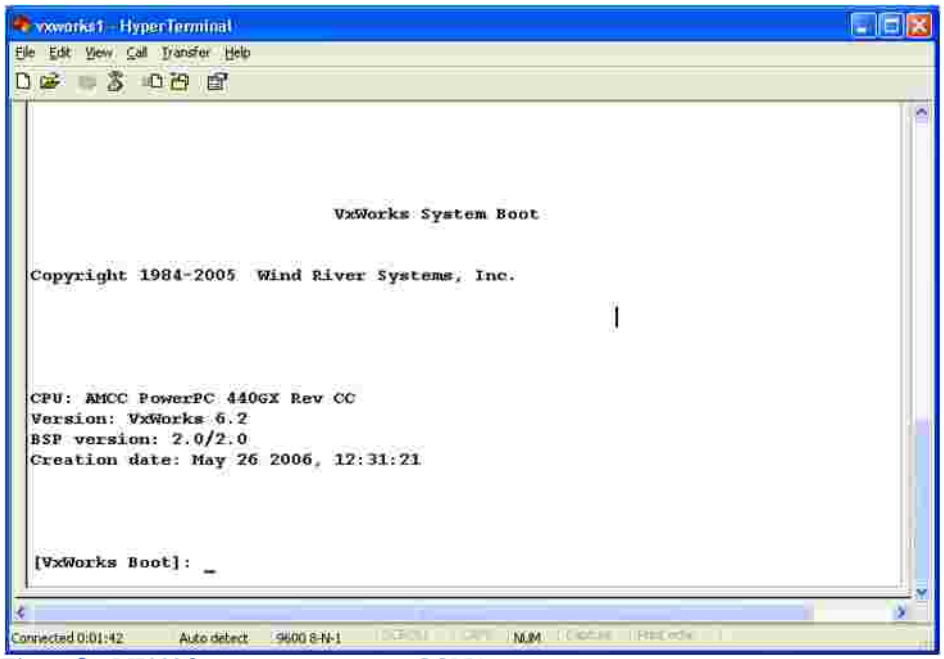


Figure 8 M5000 bootstrap message on COM1

Type @ to download VxWorks kernel from host **fys-lab-sci-02**. If everything goes OK, the boot messages are shown on Figure 9 and Figure 10.

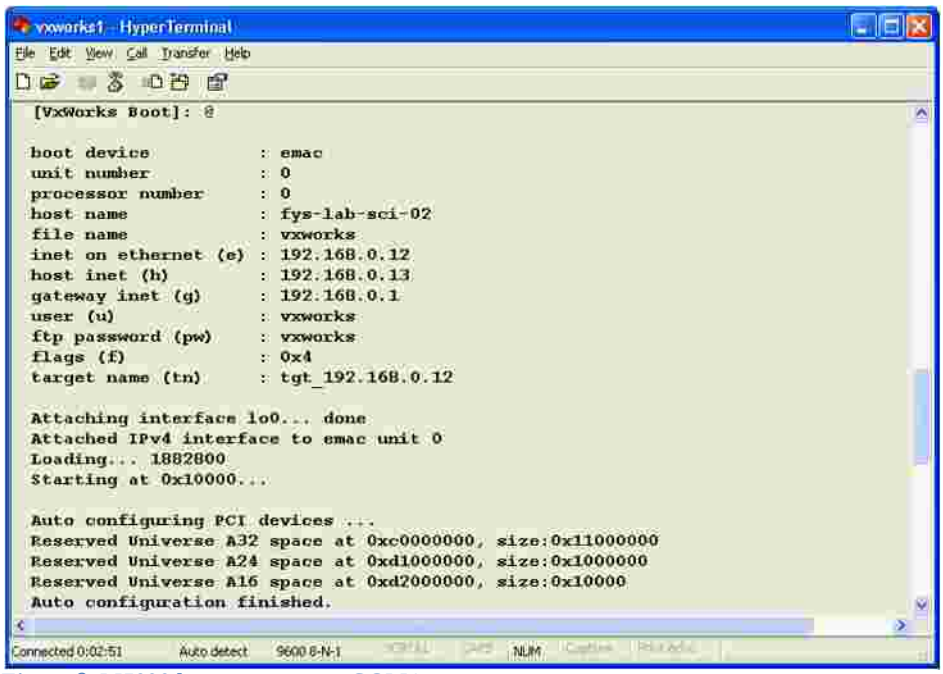


Figure 9 M5000 boot message on COM1

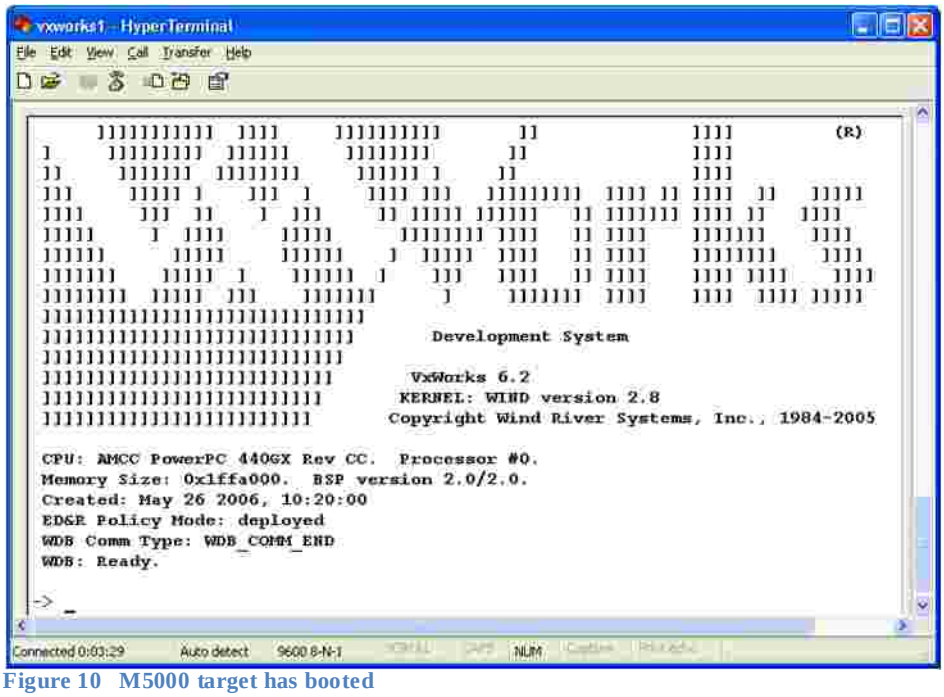


Figure 10 M5000 target has booted

4.2 Your first Workbench session

The very first time Workbench is run, the start page contains links to various help facilities. Even if it is a sign of weakness to read documentation, I suggest that one starts with the Wind River Workbench USER'S GUIDE, which can also be downloaded from the FYS4220 home page Lab Documentation.

At start-up, Workbench asks for **Select a workspace**, see Figure 11. Use the Workbench sub-directory, but on can of course define another path.

The Workbench sub-directory has been set up as default workspace for all groups!

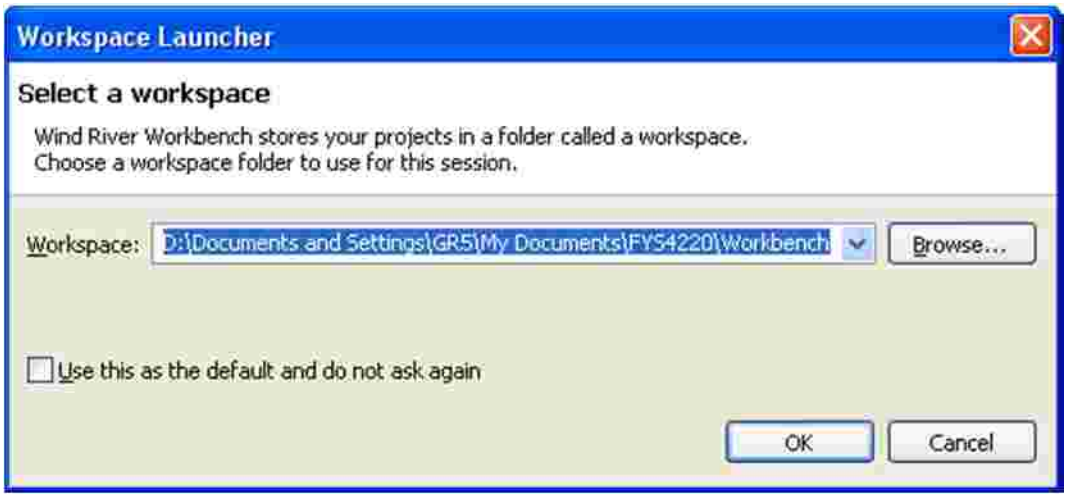


Figure 11 Workbench start-up, select workspace

The Workbench screen at first start-up is shown in Figure 12. The perspective (see upper menu in upper right corner) is “Application Development”. No projects are defined, and the **vxsim** target has not been connected.

Shown below is the procedure for creating a Workbench project which is given the name ROBOT. Select **New VxWorks Downloadable Kernel Module Project**, give it the name ROBOT, and **Create project in Workspace**, see Figure 13 and Figure 14. Use the default **Build Support**, Figure 15.

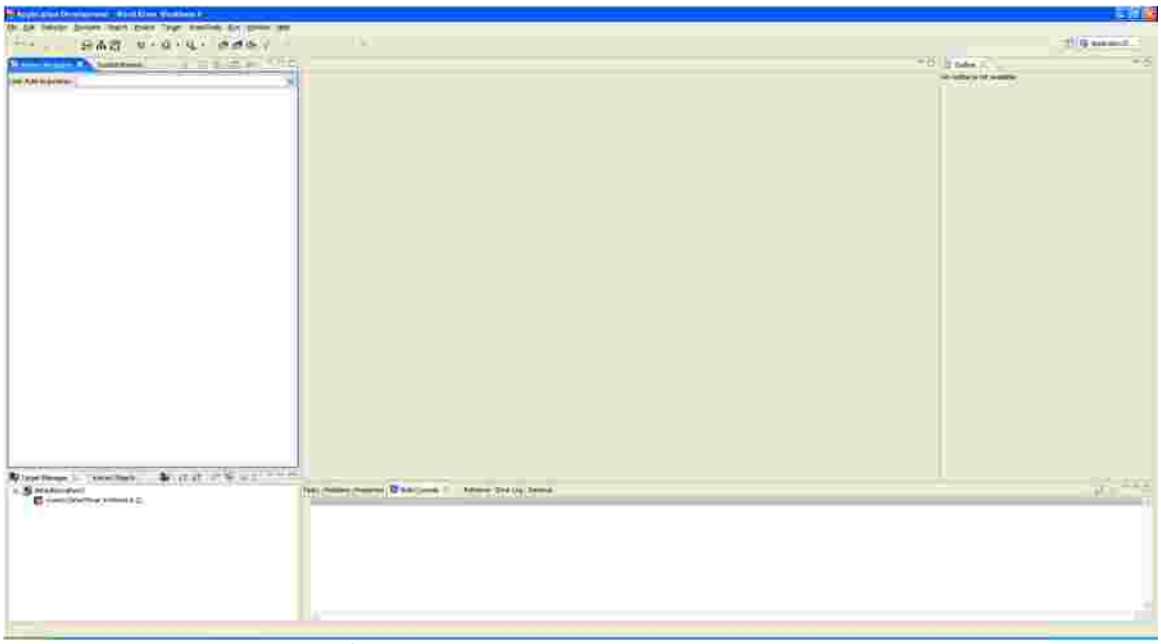


Figure 12 Workbench screen at first start-up

When Figure 16 Active build spec SIMNT gnu (for vxsim) pops up, first **Deselect All** and then select Active Build Specs **SIMNT gnu** (Wind River GNU Compiler 3.3.2) for **vxsim**. (An alternative is SIMNT diab using the Wind River compiler 5.3).

The two last steps are shown in Figure 17 and Figure 18, use default settings.

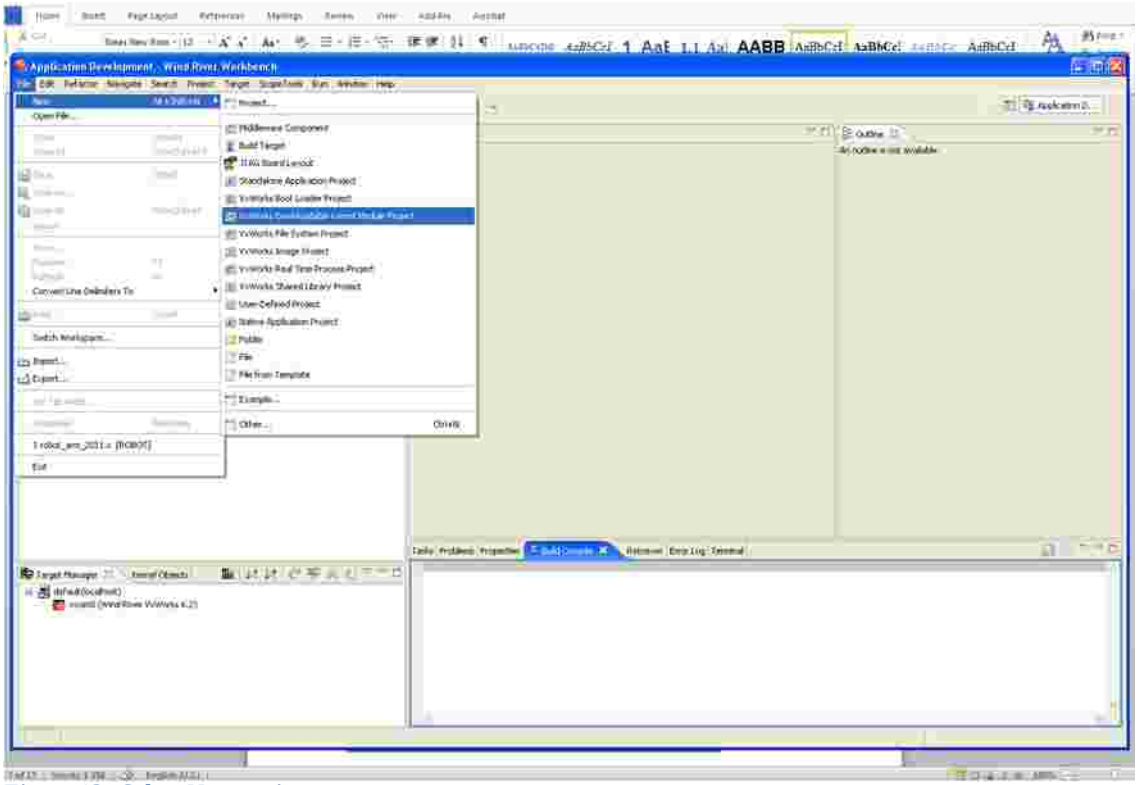


Figure 13 Select New project

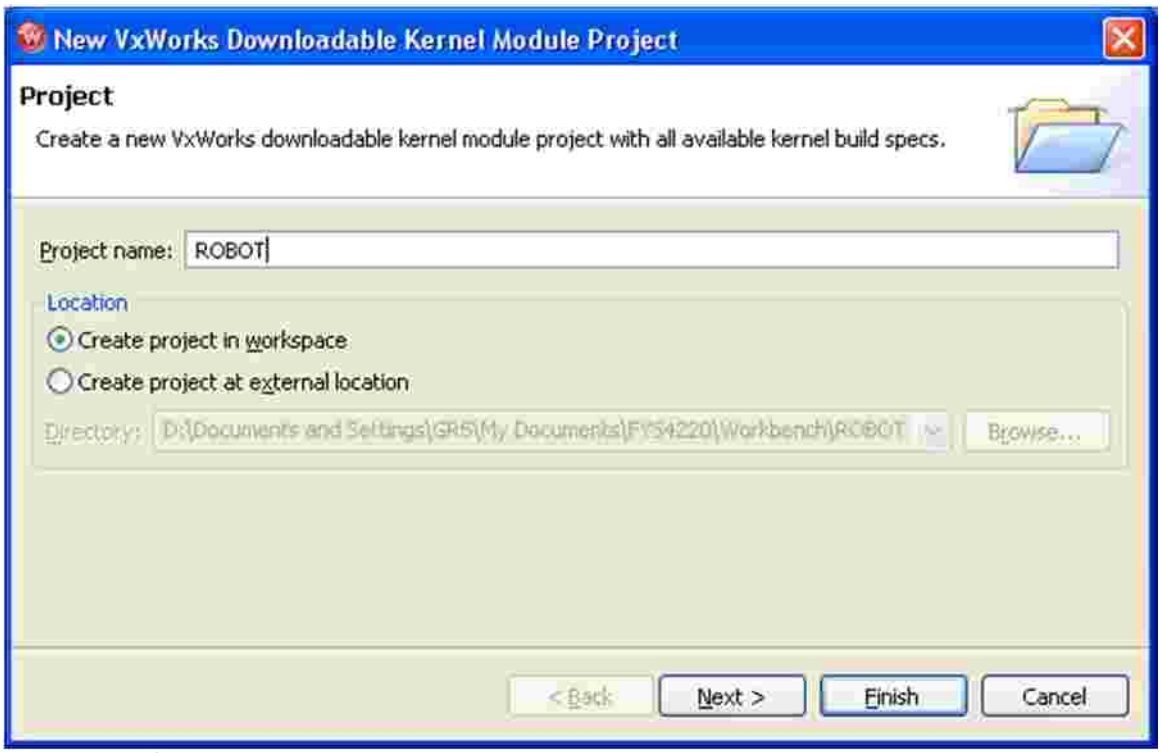


Figure 14 Define Project name ROBOT

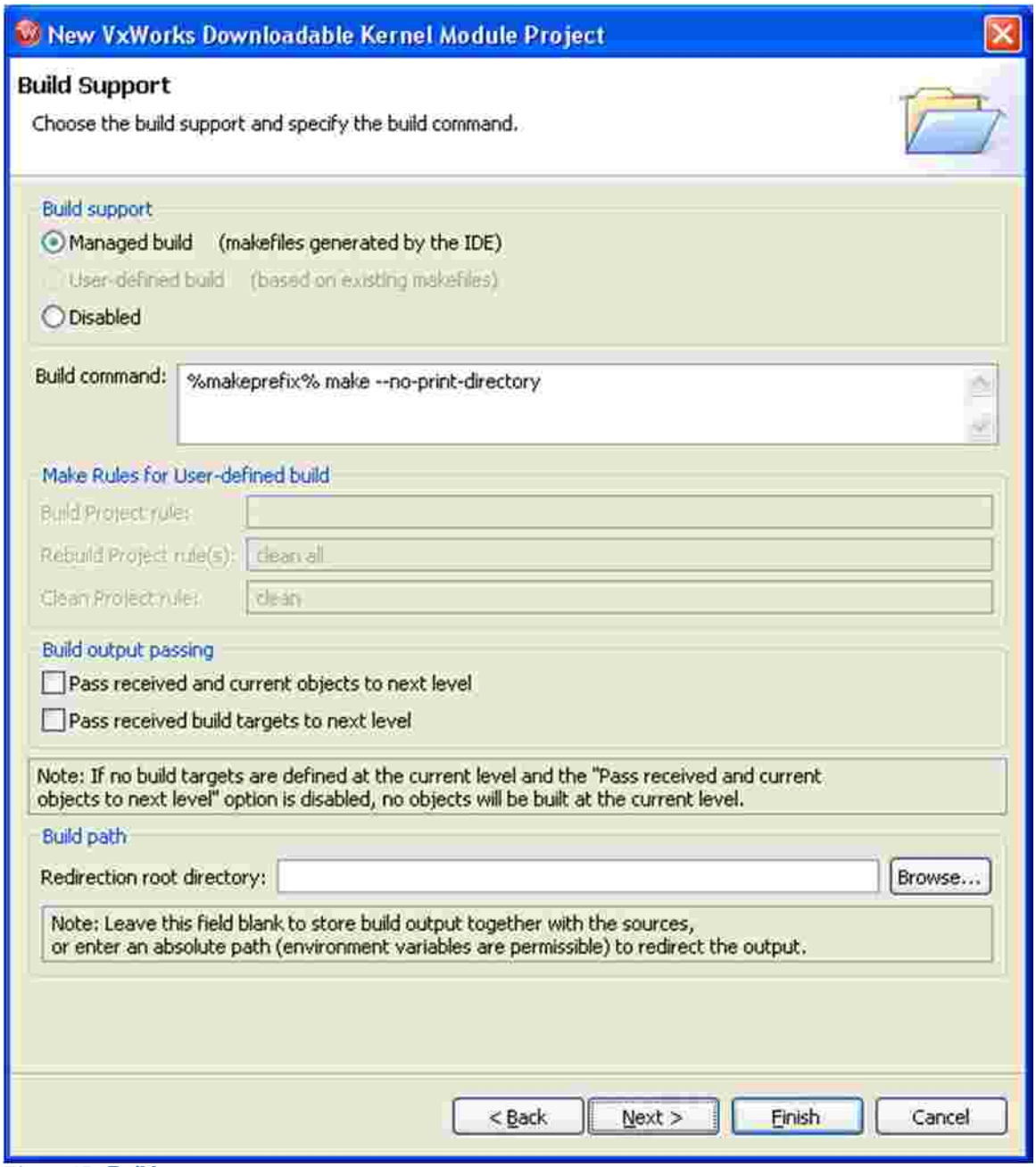


Figure 15 Build support

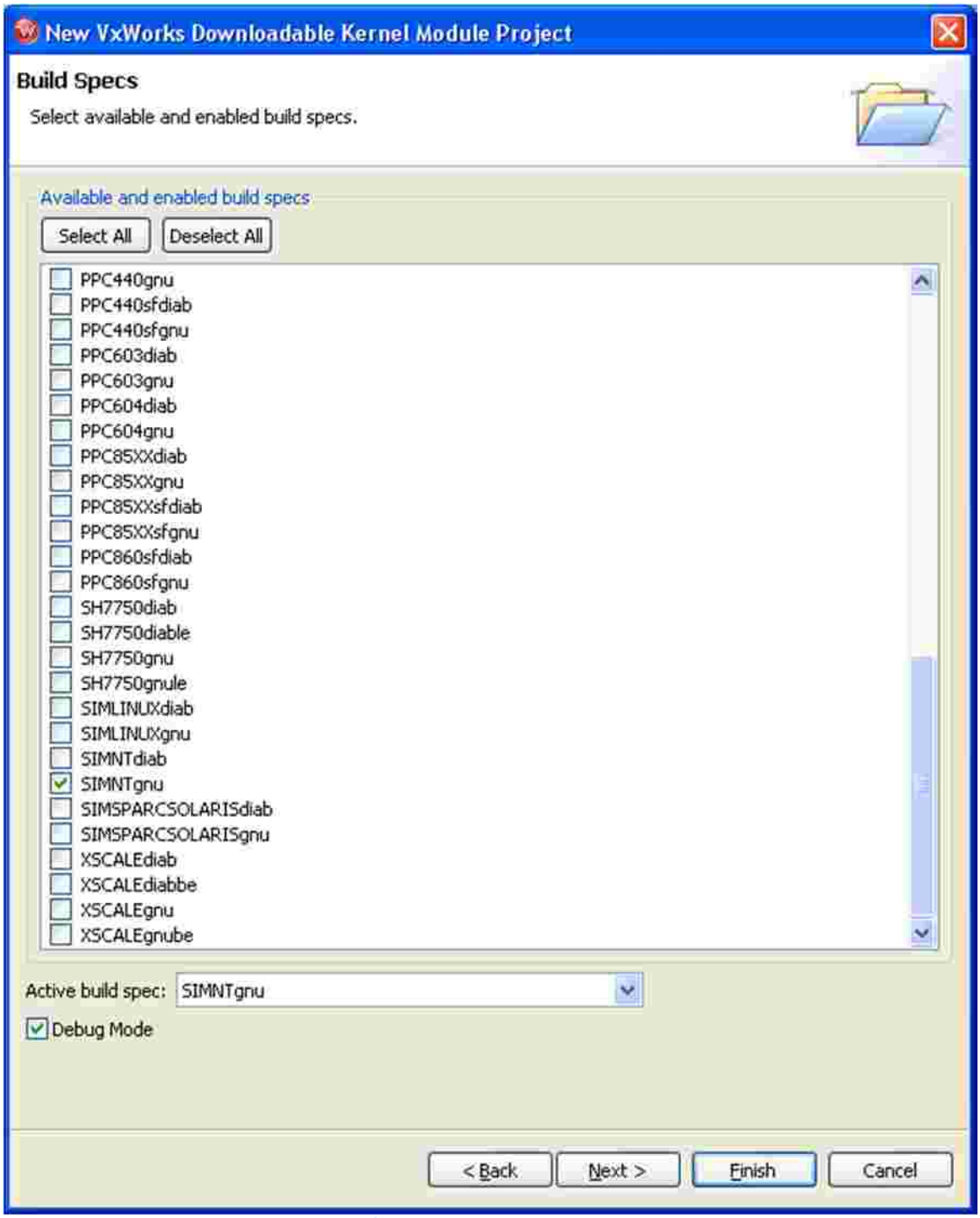


Figure 16 Active build spec SIMNT gnu (for vxsim)

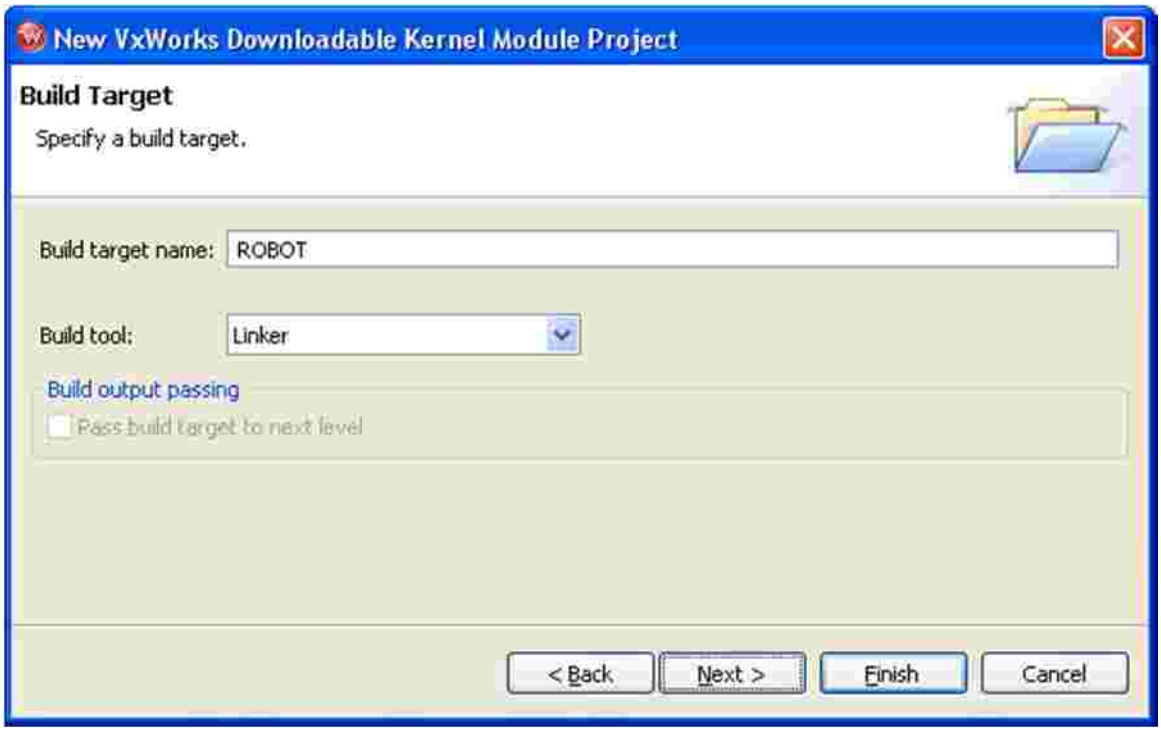


Figure 17 Build Target tool

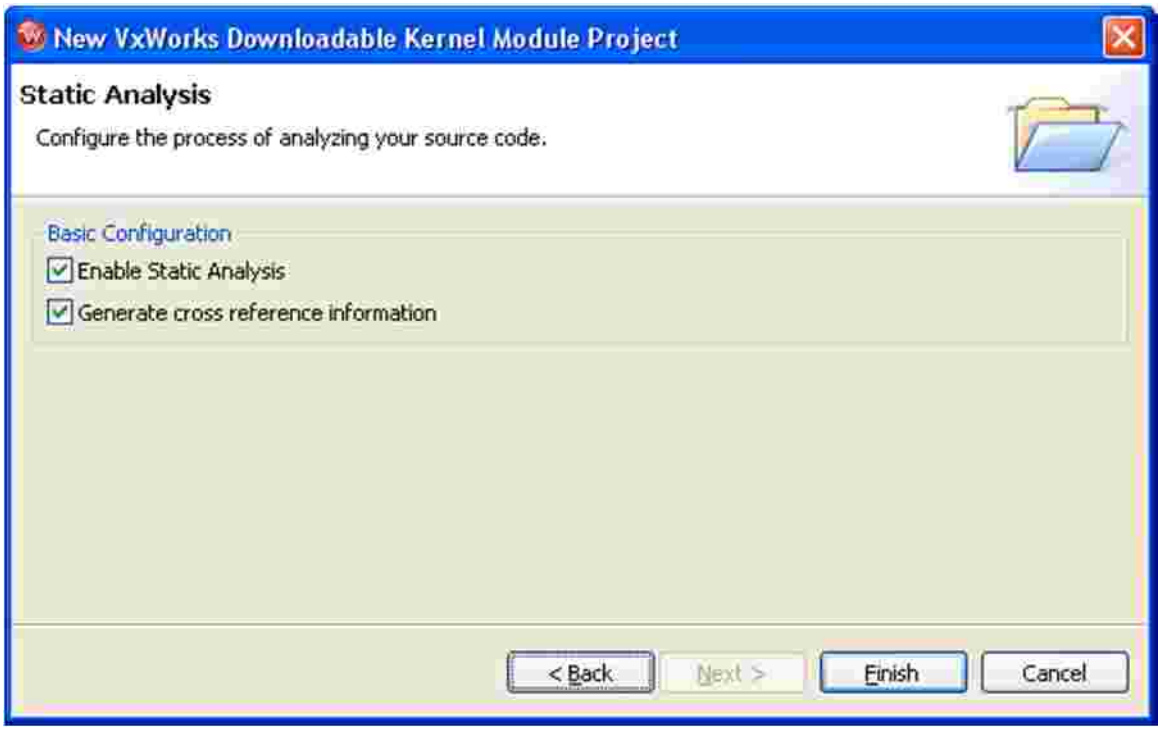


Figure 18 Static Analysis and the Finish

After Finish the Workbench screen looks like Figure 19.

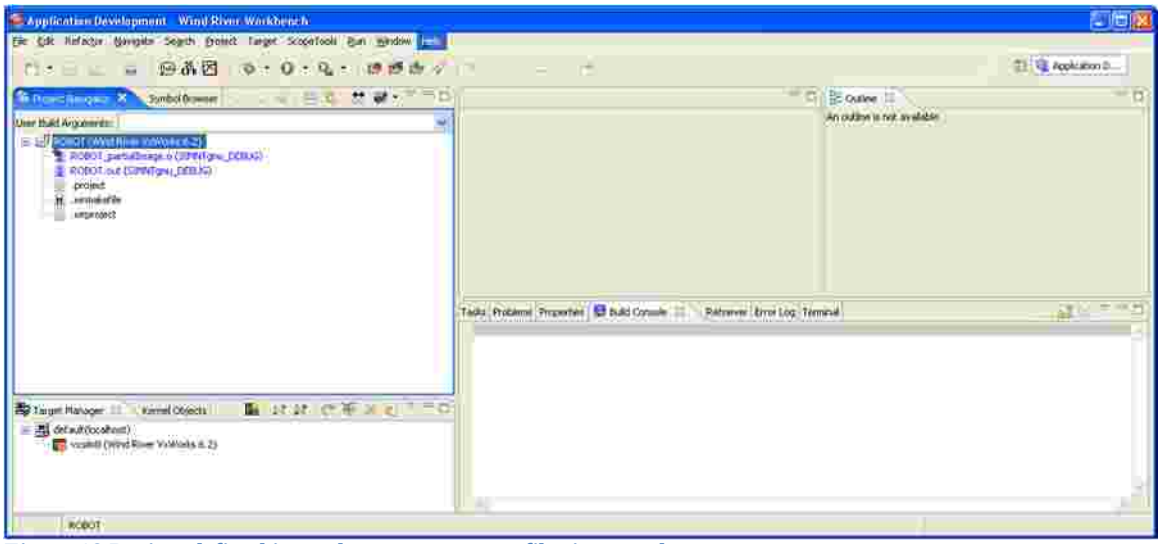


Figure 19 Project defined in workspace, no source files imported

The next step is to import the source file(s) for the project.

Under the **File** menu select **Import** which pops up Figure 20**,** and choose **File system.**

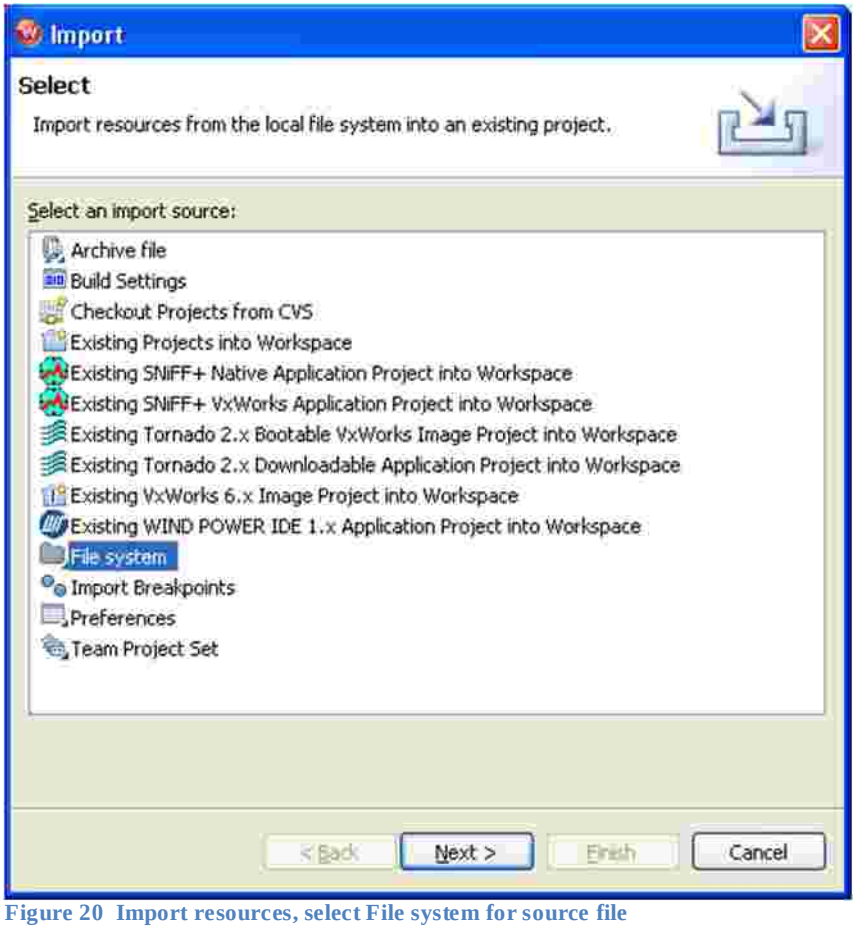


Figure 20 Import resources, select File system for source file

Then browse to the directory with the source file, in this case **…\FYS4220\src** and tick off ***robot_arm_2011.c* ,** and Finish. Now this source file has been imported.

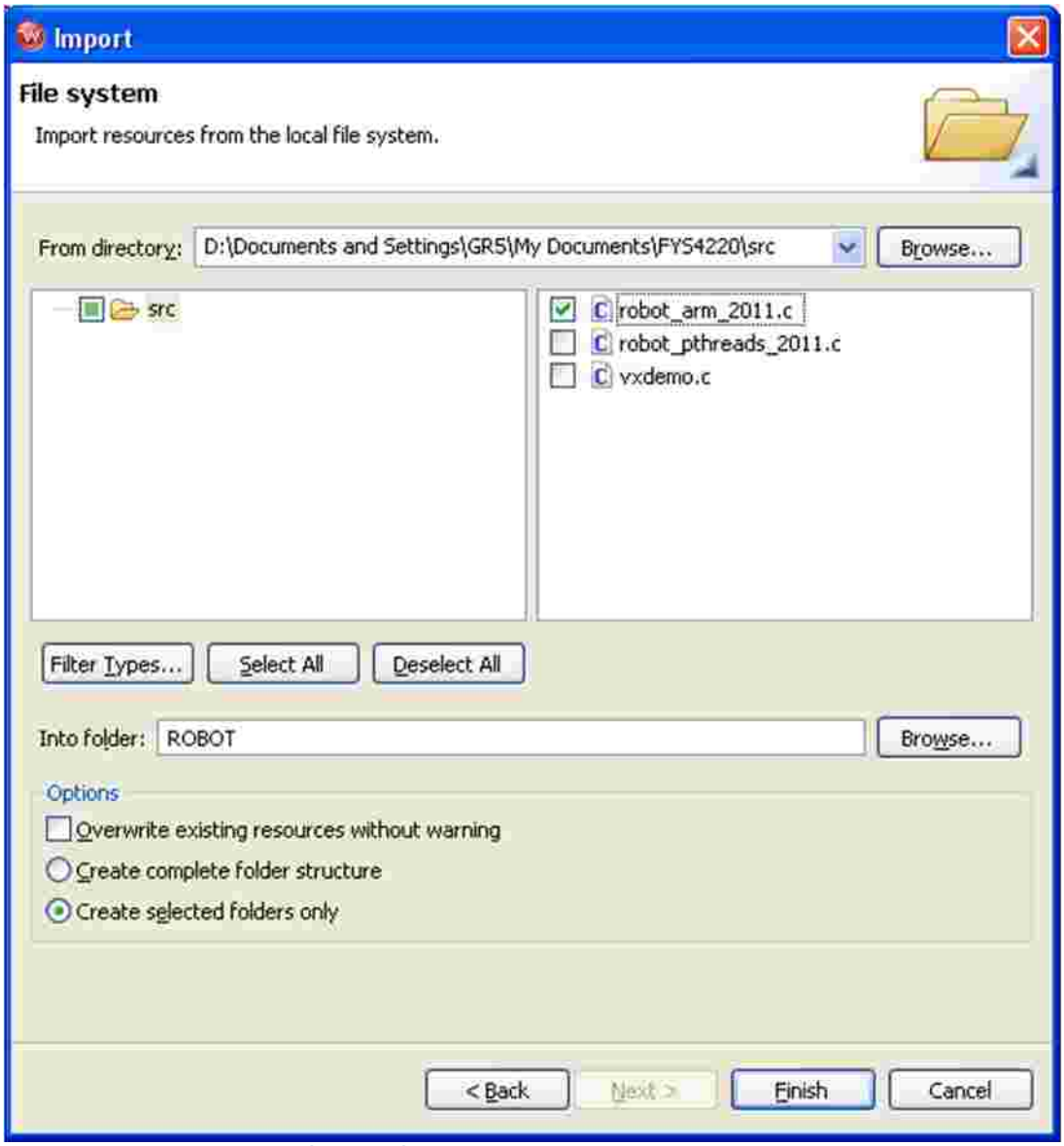


Figure 21 Import source code from the file system

Double-click on the file name in the Workbench project, and a source listing with Outline information is shown, Figure 22.

Note that this copy of the source file is stored in the workspace, and any editing on the file is not reflected in the original version. Therefore, if one wants to backup the workspace copy this must be done manually!

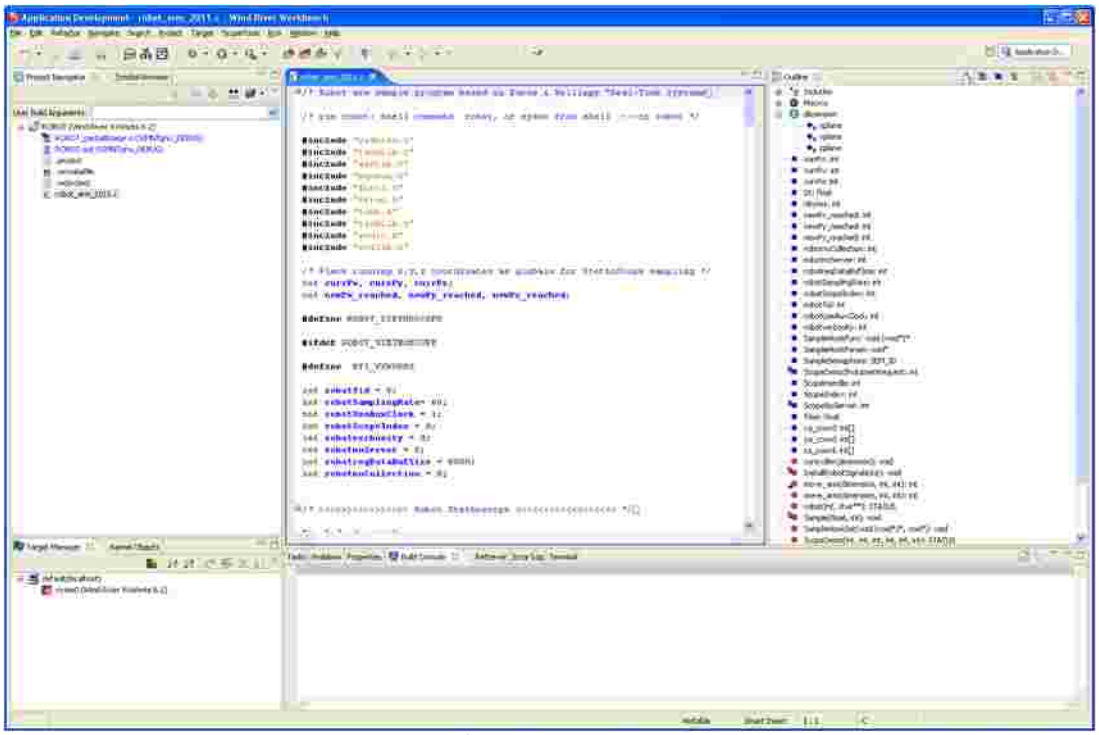


Figure 22 Workbench with imported source file

Great, now the stage is set for building and maybe even running this piece of software. Click the project to be built, and select **Build Project** under the **Project** menu.

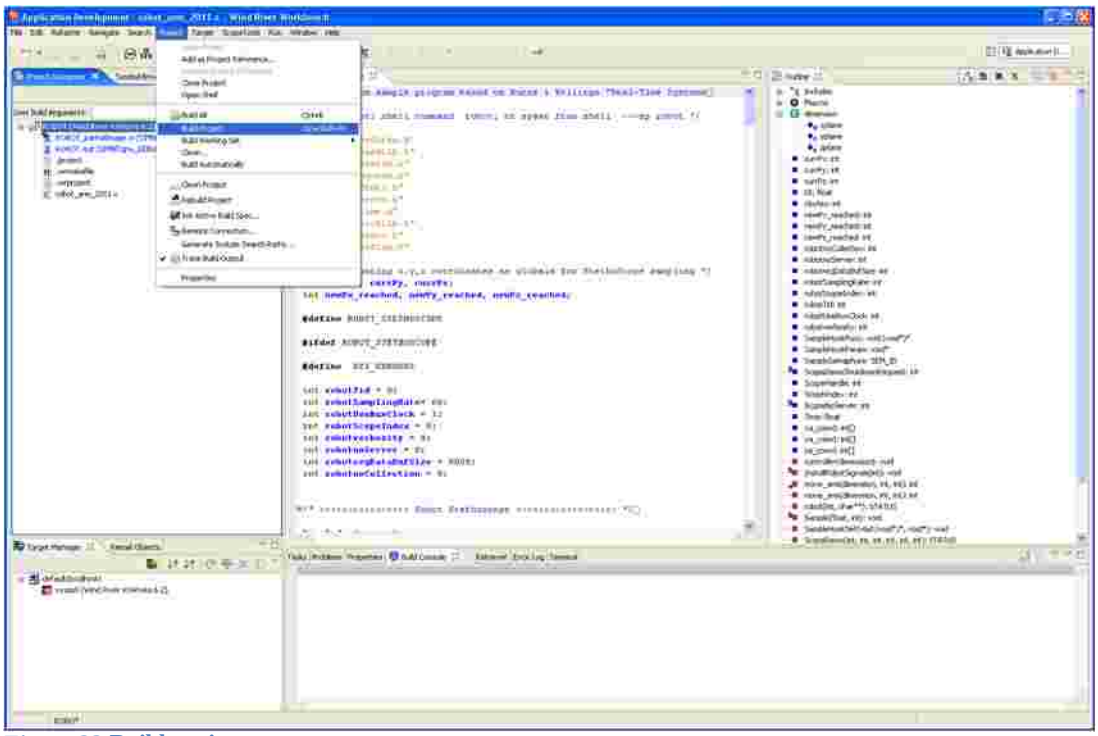


Figure 23 Build project

Probably you will have to define the include search paths, see Figure 24, Figure 27, Figure 28 and Figure 29.

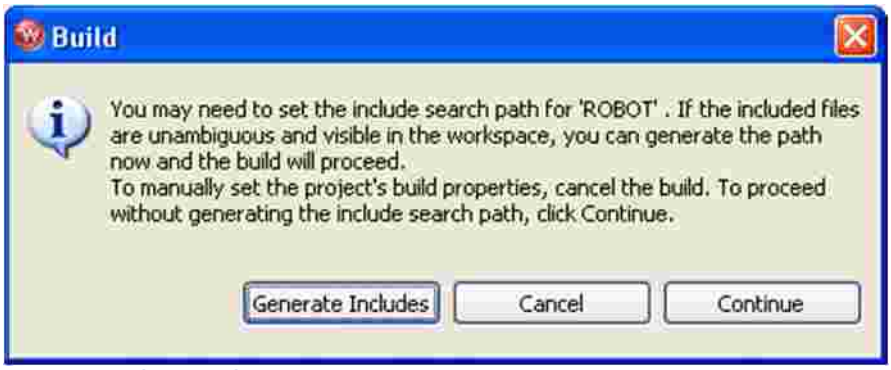


Figure 24 Definition of Include paths

The very first time Build is selected, Figure 25 and Figure 26 may pop up.

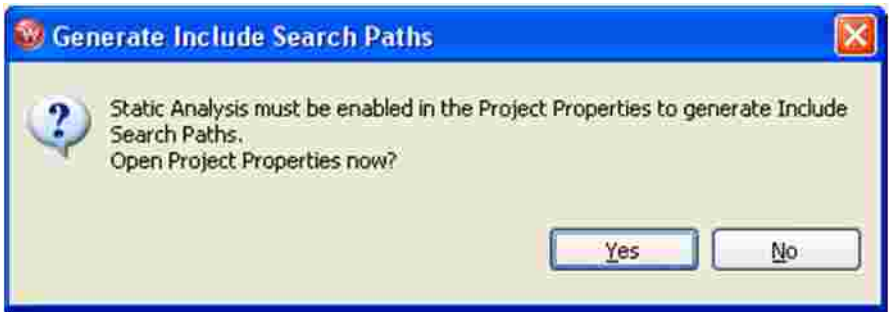


Figure 25 Select YES to continue

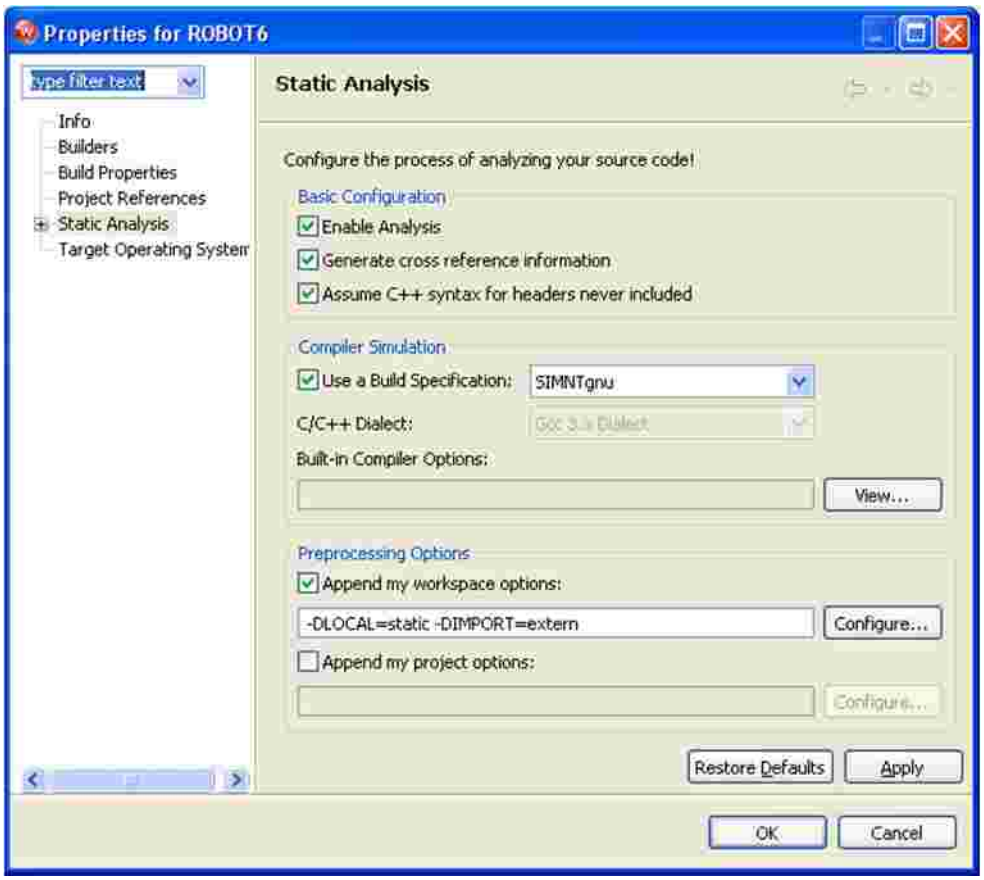


Figure 26 Select OK

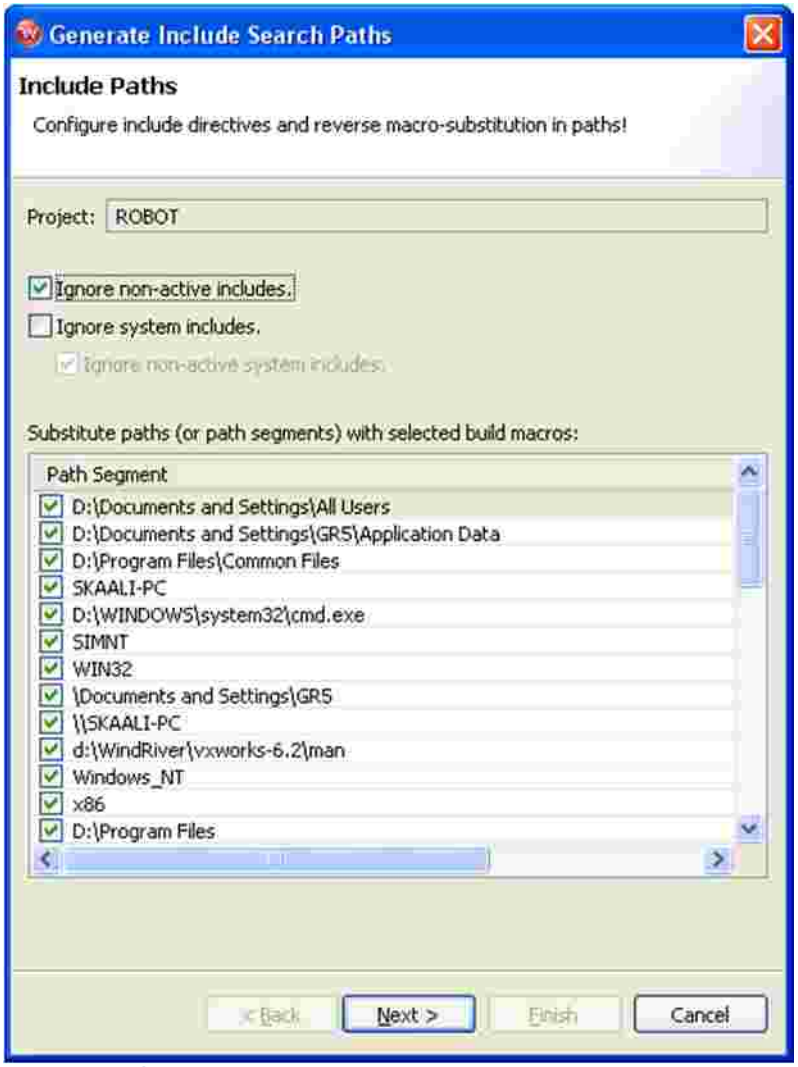


Figure 27 Default Include paths, just take all

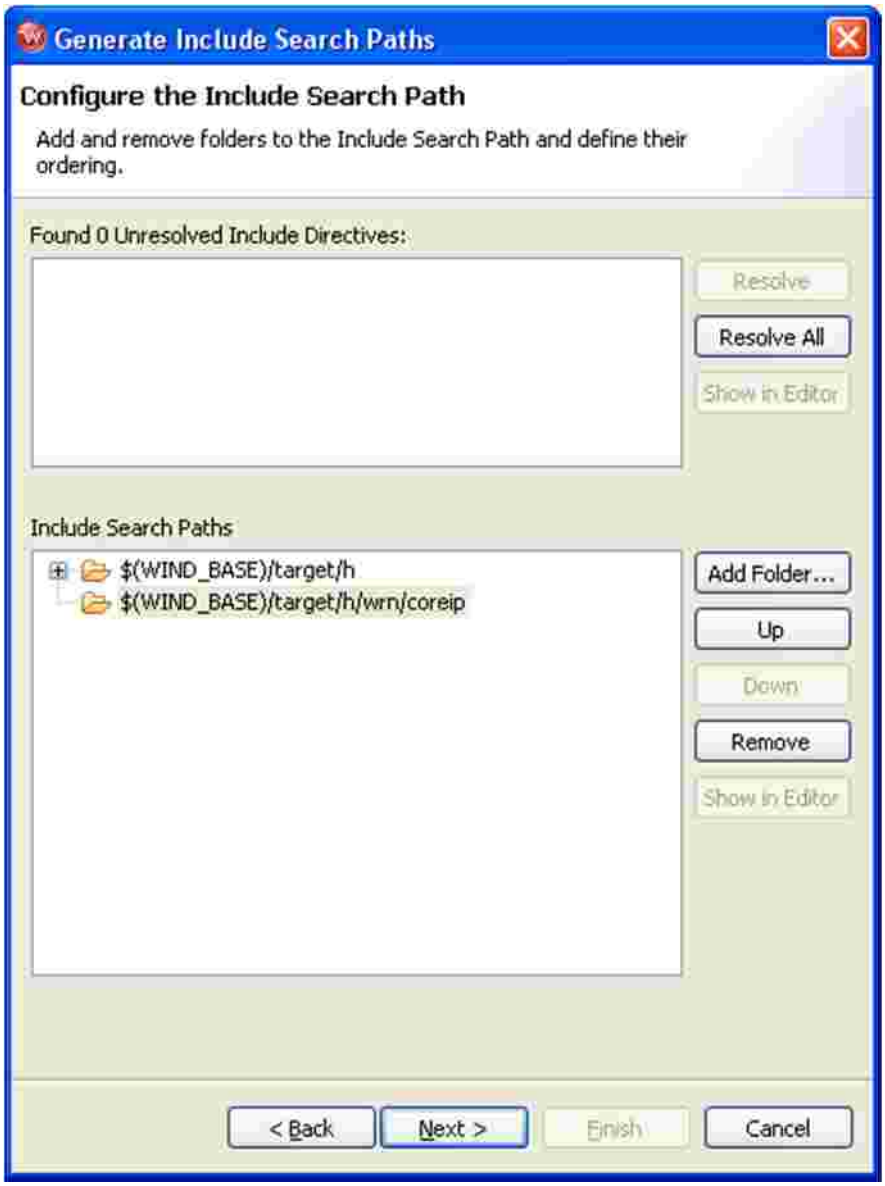


Figure 28 Any unresolved Include paths can now be repaired

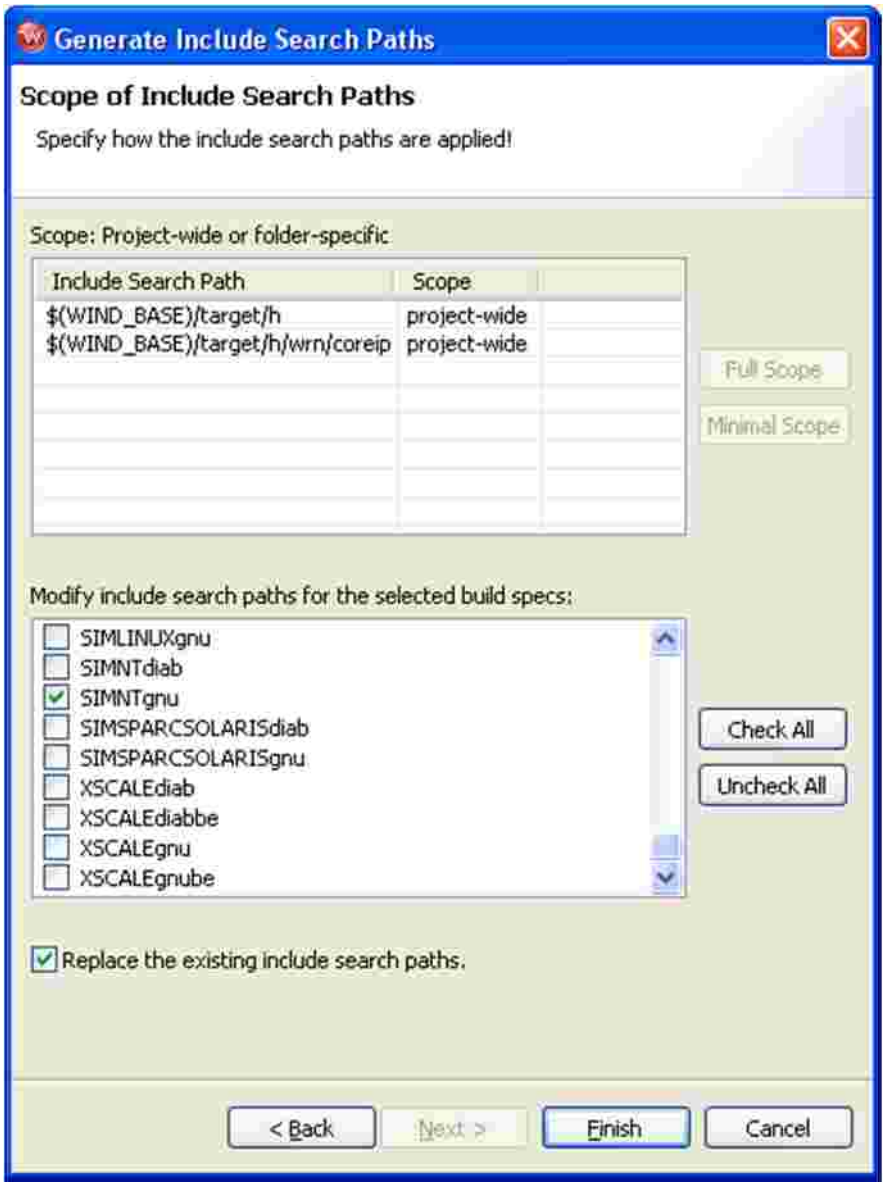


Figure 29 Uncheck All and Check SIMNT

Since the code is perfect, the compilation goes through with flying colours with no error messages, Figure 30.

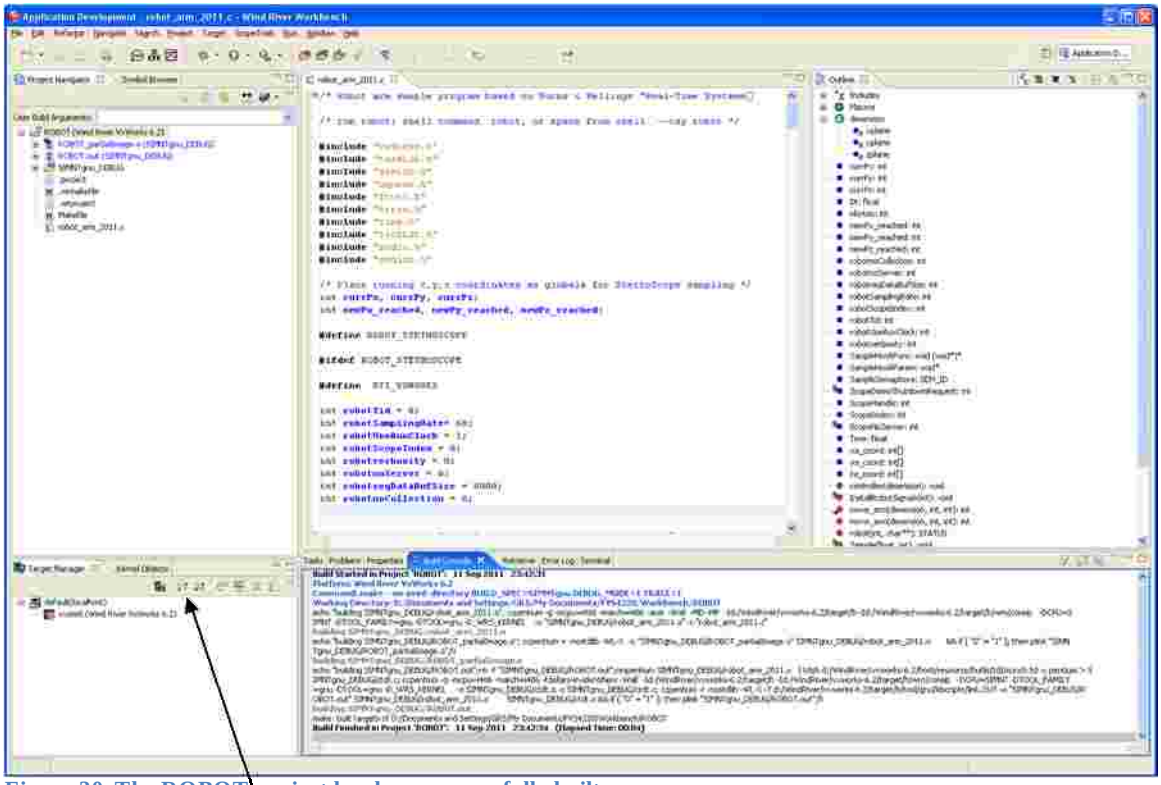


Figure 30 The ROBOT project has been successfully built.

The next step is now to launch the target **vxsim**. The arrow points to the **Connect** symbol in the Target Manager. Click first on ***vxsim0***. If the target Connect succeeds, the content of the Target window is shown in Figure 31. The next symbol stands for **Disconnect** target. A terminal window to vxsim0 is opened, see Figure 32.

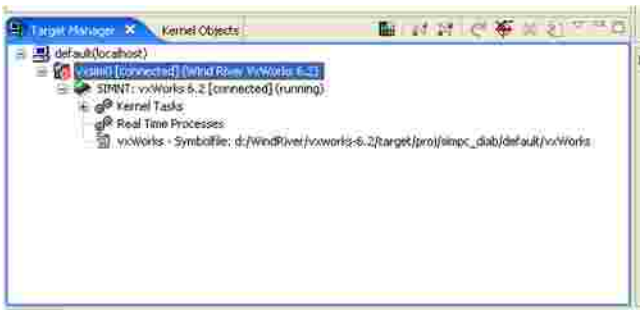


Figure 31 target vxsim0 connected

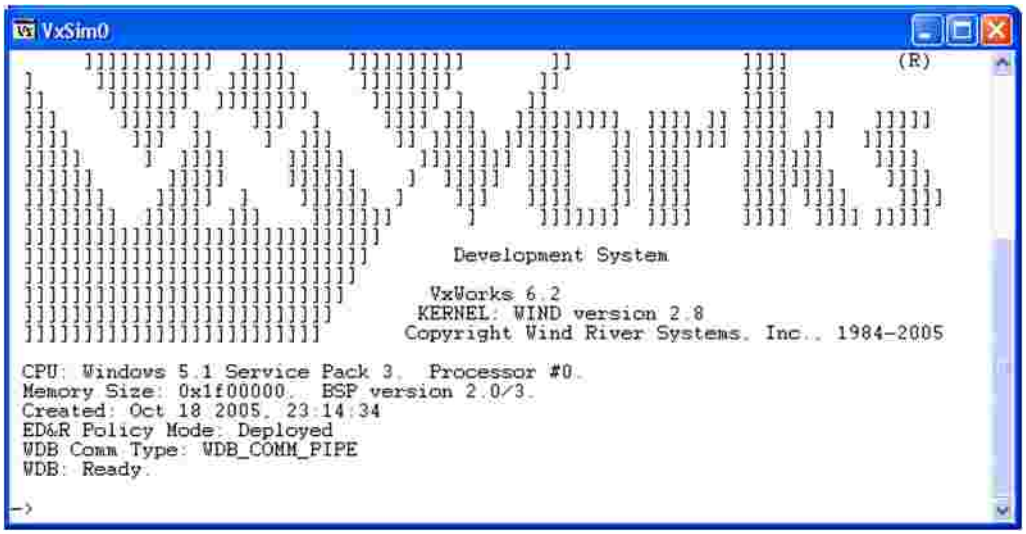


Figure 32 target Vxsim0 terminal window

Continue from section 4.3 if **vxsim** is used.

4.2.1 Connecting the M5000 target to host fys-lab-sci-02

This target name is ***tgt_192.168.0.12***, see Figure 33.

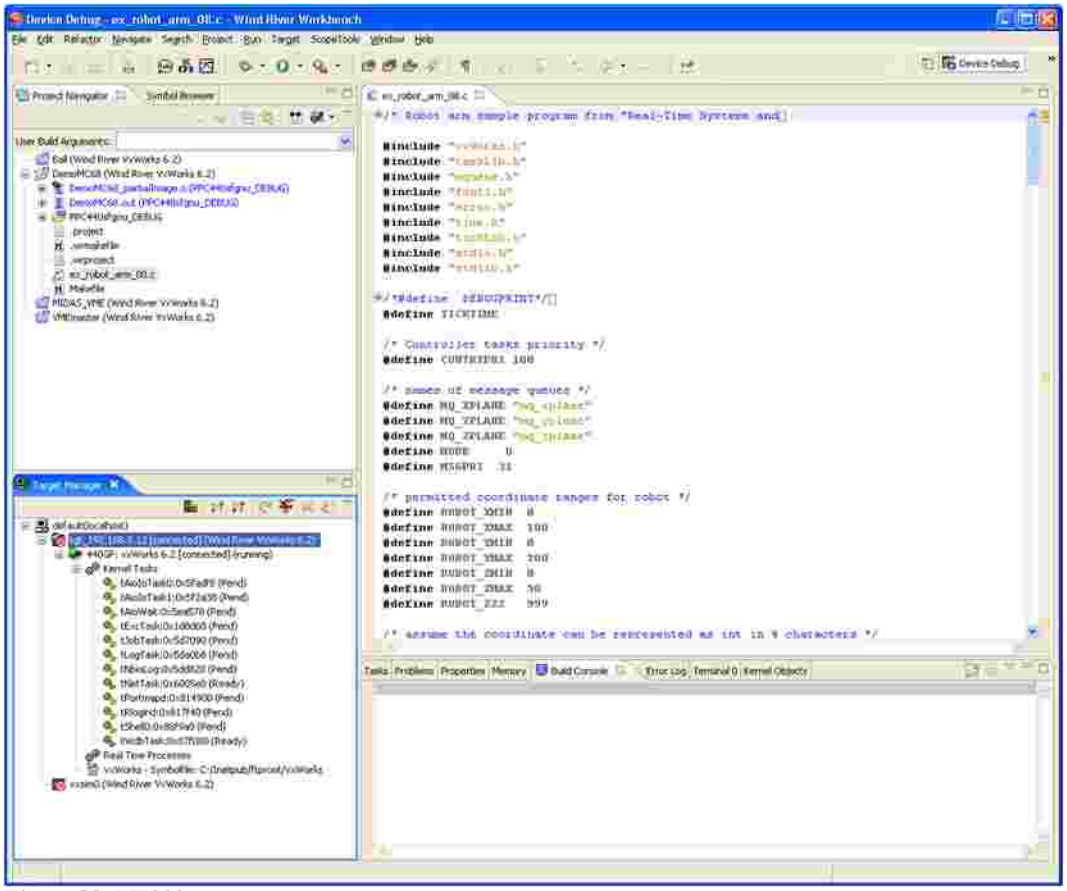


Figure 33 M5000 target

4.2.2 Building a project for the M5000 target

The procedure is the same as for **vxsim**, as **Build Specs** one chooses **PPC440sfgnu.**

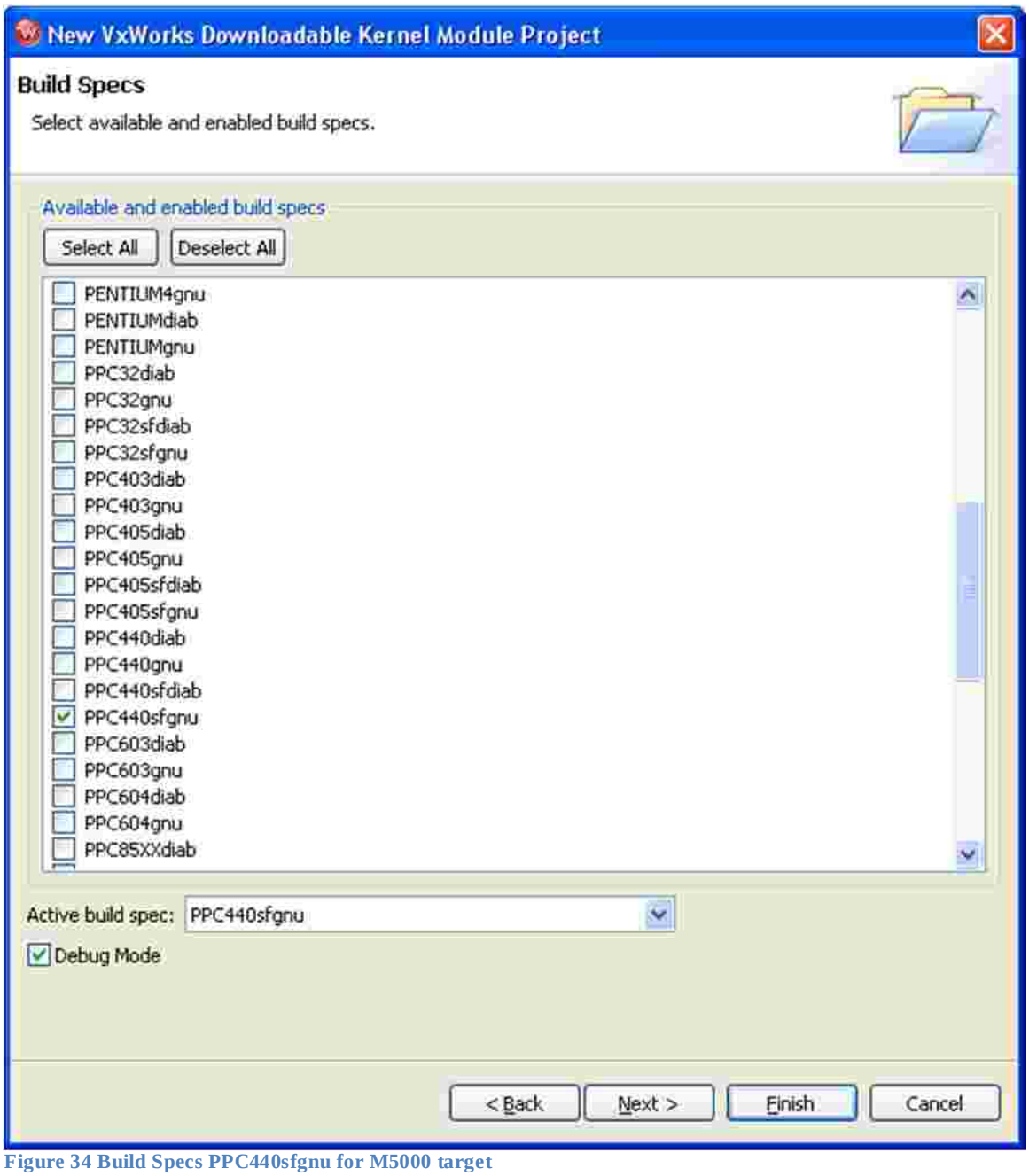


Figure 34 Build Specs PPC440sfgnu for M5000 target

4.3 Downloading the project kernel module

Click on the Download symbol as marked in Figure 35.

Browse the workspace and select the wanted **.out** file as shown in Figure 36. Click OK.

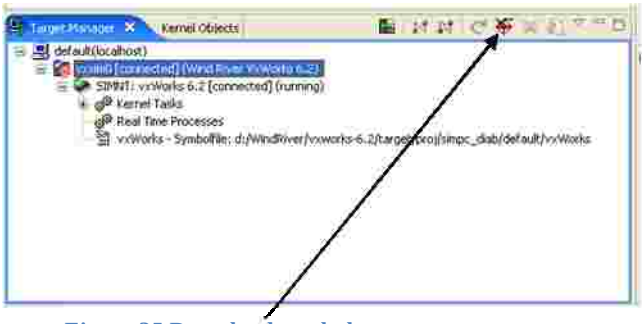


Figure 35 Download symbol

After download the **.out** file will show up in the Target Manager window.

If there are unresolved symbols in the object file an error message will appear. This will happen is a header file is missing, or if a library has to be loaded prior to the object file.

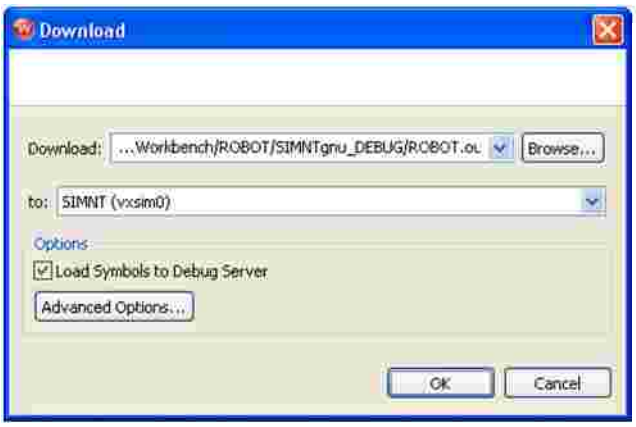


Figure 36 Download of project .out file

4.4 Executing the kernel module

Type the start address in the target terminal window, or COM1 window for M5000 target, or a host shell window (see below). If one has to abort the module, disconnect the target. After a fatal error one will usually have to reboot the target.

One can also open another host shell from the Workbench Target menu with the ***Host Shell*** command.

A task can be deleted with the shell command td <name>. Do not delete system tasks!
The shell command ***->help*** displays the available commands. The command ***->i*** list all kernel tasks.

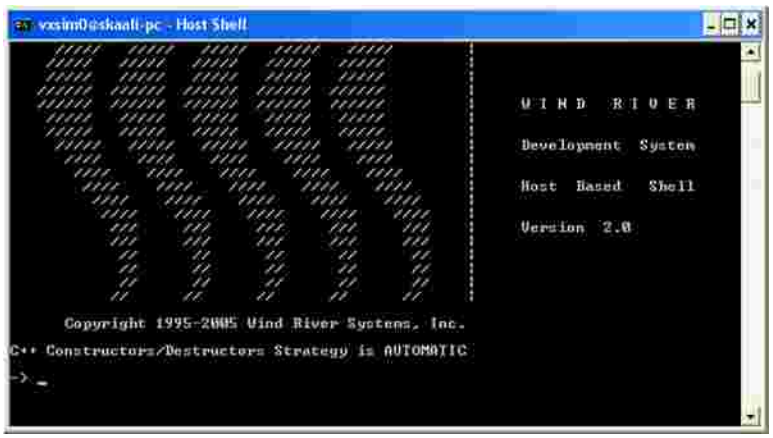


Figure 37 VxWorks host shell

PART 2

5 The ROBOT

The source code which is the basis for the exercise is stored under **…\FYS4220\src**, see Figure 3. One can choose between two versions, **robot_arm_2011.c** based on VxWorks tasks and **robot_pthreads_2011.c** based on POSIX pthreads. Furthermore the directory contains **vxdemo.c** which demonstrates various ***StethoScope*** facilities.

5.1 What to do

The program mimics the X, Y and Z movements of a robot. The coordinate sets are stored in an array. The output in Figure 38 shows that the movements along the three axis's are completely independent, which is hardly a good marketing argument for a robot. (Note that if the program is started from a target shell the "RT-clock" and "tickGet" printouts are not mixed.)

The goal of the exercise is to synchronize the X, Y and Z movements such that all three reach a coordinate waypoint before continuing. Methods for synchronization between tasks / threads have been presented in the lectures. There is no requirement that X_i, Y_i and Z_i are reached at the same time.

Furthermore, the code demonstrates:

- Concurrency
- Communication via POSIX Message Queues
- A half-hearted attempt to handle error conditions
- POSIX pthreads if one prefers that version

Included in the source code is the use of **StethoScope** for visual tracking of the movements in real time. This tool can be disabled by commenting out the declaration #define ROBOT_STETHOSCOPE, see Figure 39.

The building and download of project ROBOT have been explained in PART 1.

If the project is built without StethoScope, the status of **vxsim** after ROBOT download is displayed in Figure 40.

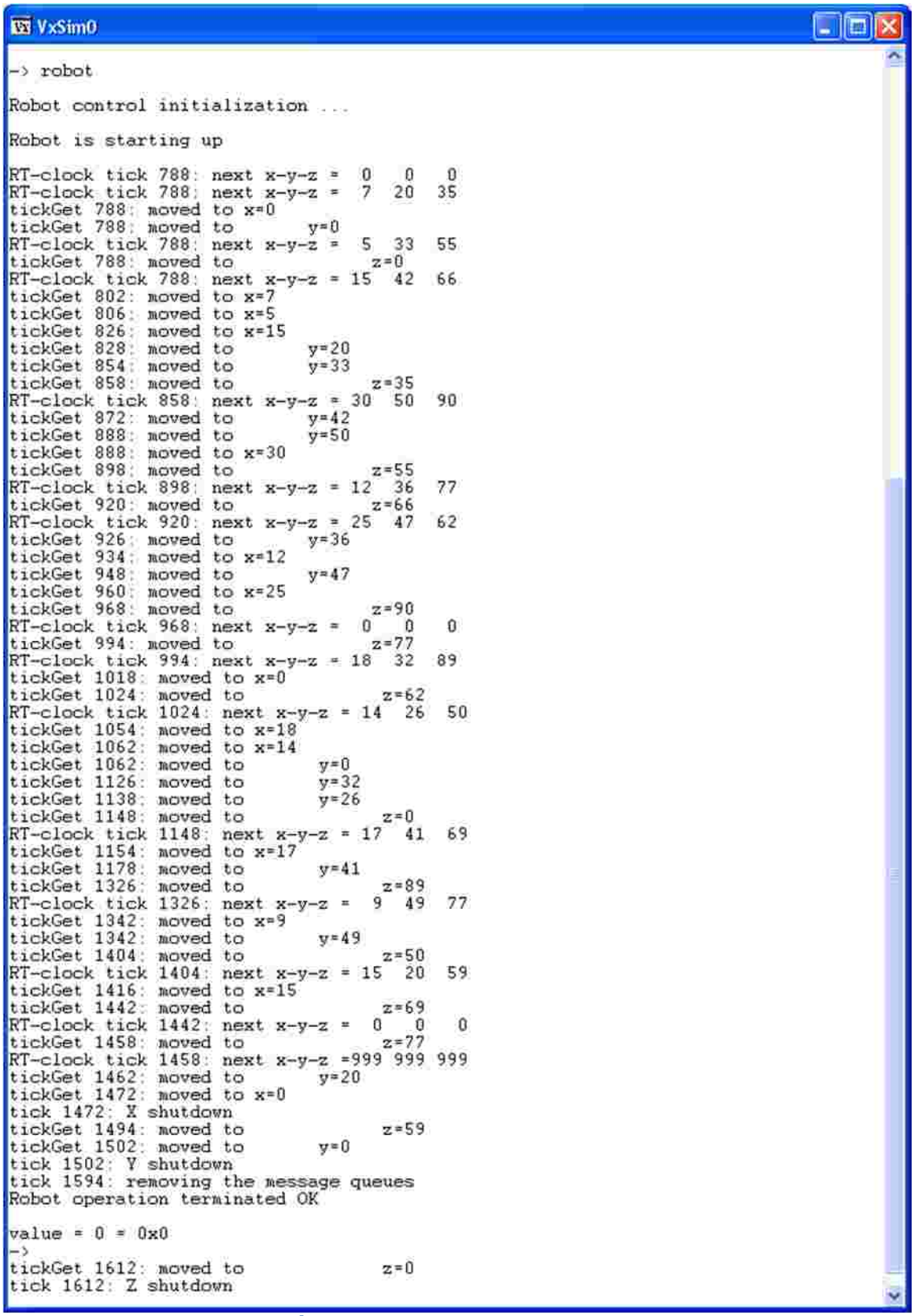
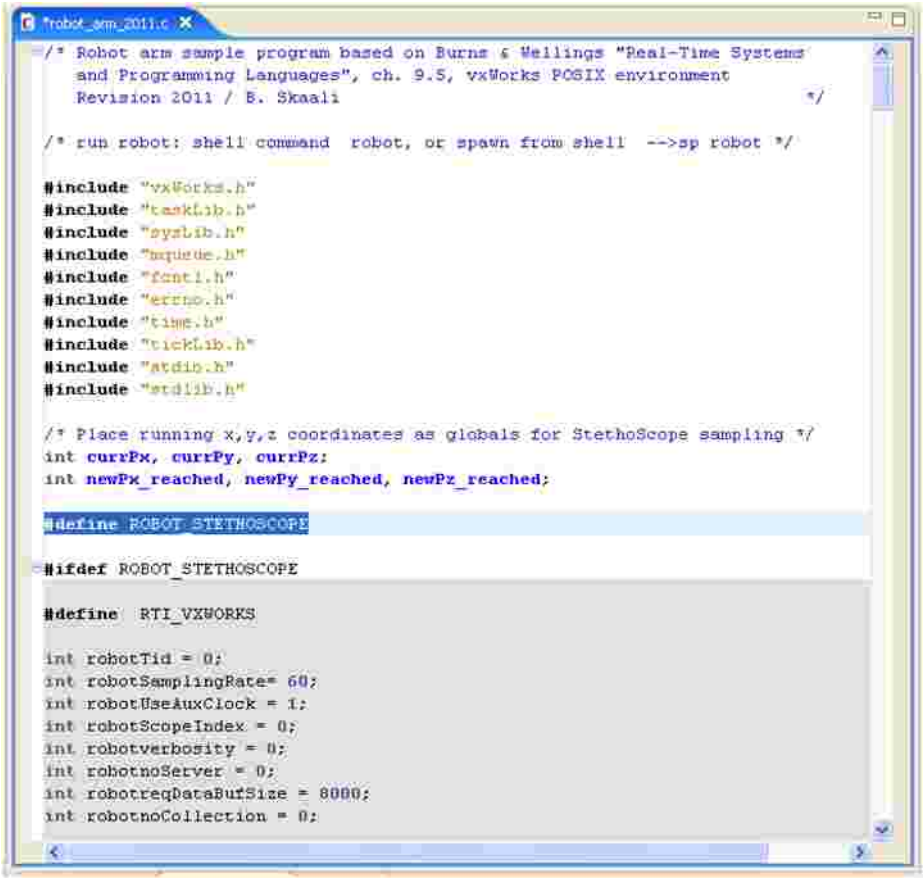


Figure 38 Non-coordinated X, Y and Z robot movements



```
/* Robot arm sample program based on Burns & Wellings "Real-Time Systems
   and Programming Languages", ch. 9.5, vxWorks POSIX environment
   Revision 2011 / B. Skaali                                          */

/* run robot: shell command  robot, or spawn from shell  -->sp robot */

#include "vxWorks.h"
#include "taskLib.h"
#include "sysLib.h"
#include "unistd.h"
#include "fcntl.h"
#include "errno.h"
#include "time.h"
#include "tickLib.h"
#include "stdio.h"
#include "stdlib.h"

/* Place running x,y,z coordinates as globals for StethoScope sampling */
int currPx, currPy, currPz;
int newPx_reached, newPy_reached, newPz_reached;

#define ROBOT_STETHOSCOPE

#ifdef ROBOT_STETHOSCOPE

#define  RTI_VXWORKS

int robotTid = 0;
int robotSamplingRate= 60;
int robotUseAuxClock = 1;
int robotScopeIndex = 0;
int robotverbosity = 0;
int robotnoServer = 0;
int robotreqDataBufSize = 8000;
int robotnoCollection = 0;
```

Figure 39 robot_arm_2011.c source

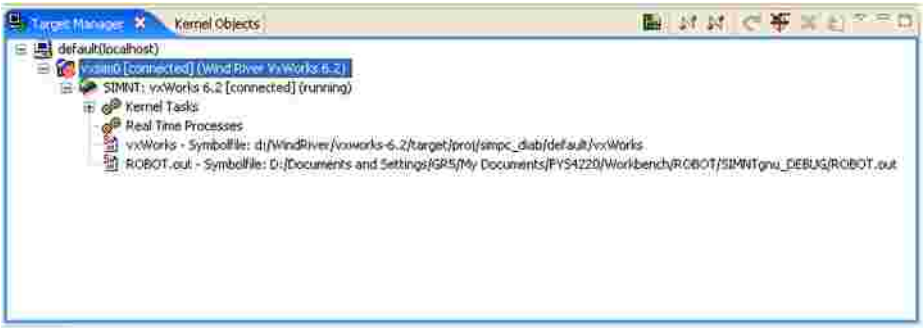


Figure 40 vxsim0 status after download of project ROBOT

5.2 StethoScope

Workbench supplies three **ScopeTools** for monitoring of VxWorks tasks / pthreads in real time: **MemScope**, **ProfileScope** and **StethoScope**. To quote from the **StethoScope** Users Guide:

> StethoScope is the user-friendly, real-time graphical-monitoring and data-collection tool from Wind River. It lets you monitor and analyze the values of variables in your real-time application while the application is running. StethoScope is more than an easy-to-use data-collection tool—it is a powerful debugging aid for both hardware and software. You can use its multi-window environment to track down performance problems, "glitches," and program errors.

The Guide covers 283 pages, is available in the lab, and can be downloaded from the FYS4220 home page.

5.2.1 StethoScope with the ROBOT module

Compile with

#define ROBOT_STETHOSCOPE

Inspect the ROBOT source code and see where the StethoScope information is collected.

Before downloading the .out file, the StethoScope library must be downloaded to the target.

Select **ScopeTools** -> **StethoScope,** specify Target Type VxWorks, see Figure 41.

Figure 41 StethoScope Target Type

Specify the Setup Options as shown in Figure 42. If more printout is wanted the Verbosity value can be increased. Do NOT use Aux Clock.

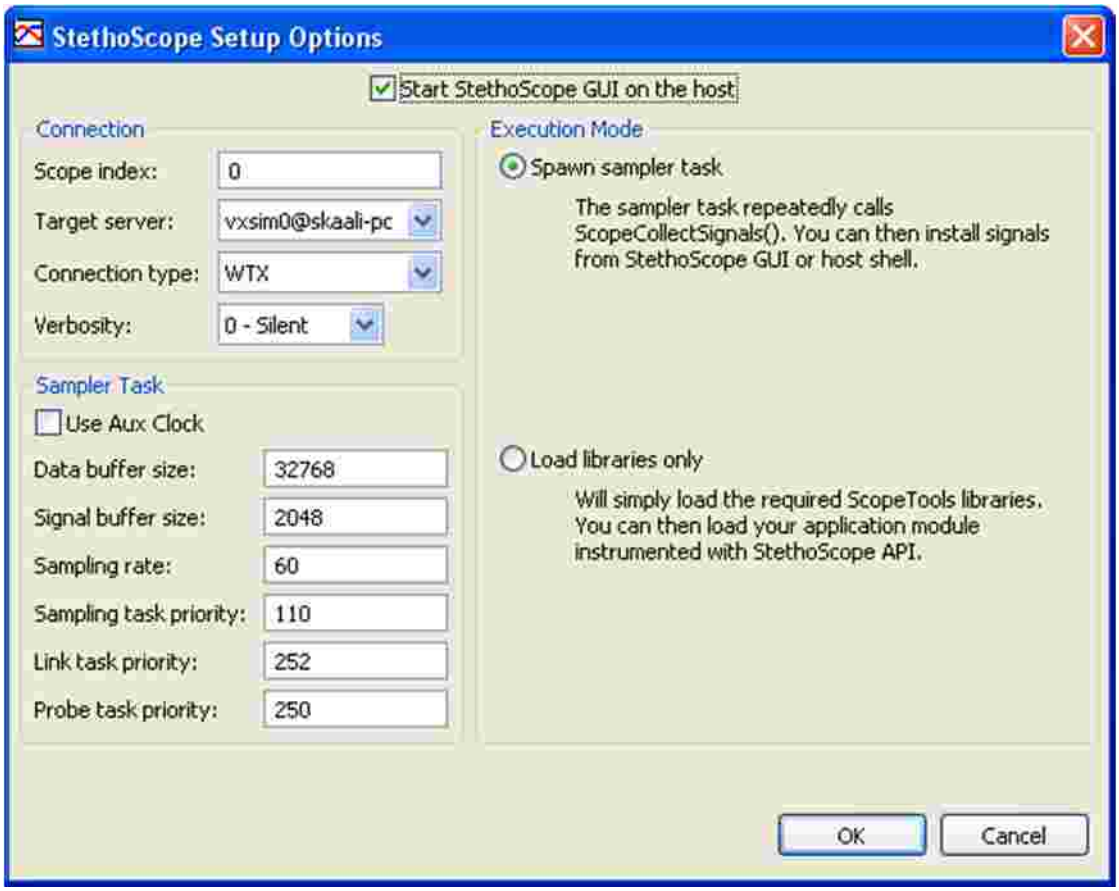


Figure 42 StethoScope Setup Options

The Workbench Target Manager window after download of the StethoScope library and GUI is shown in Figure 43, and the initial GUI plot window in Figure 44.

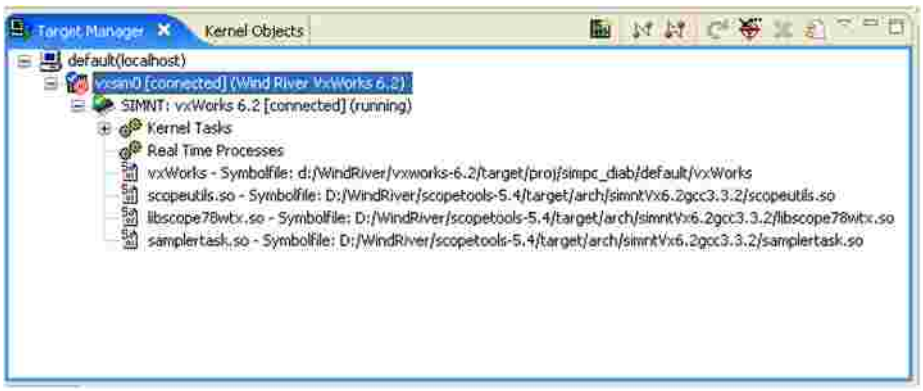


Figure 43 StethoScope library and GUI

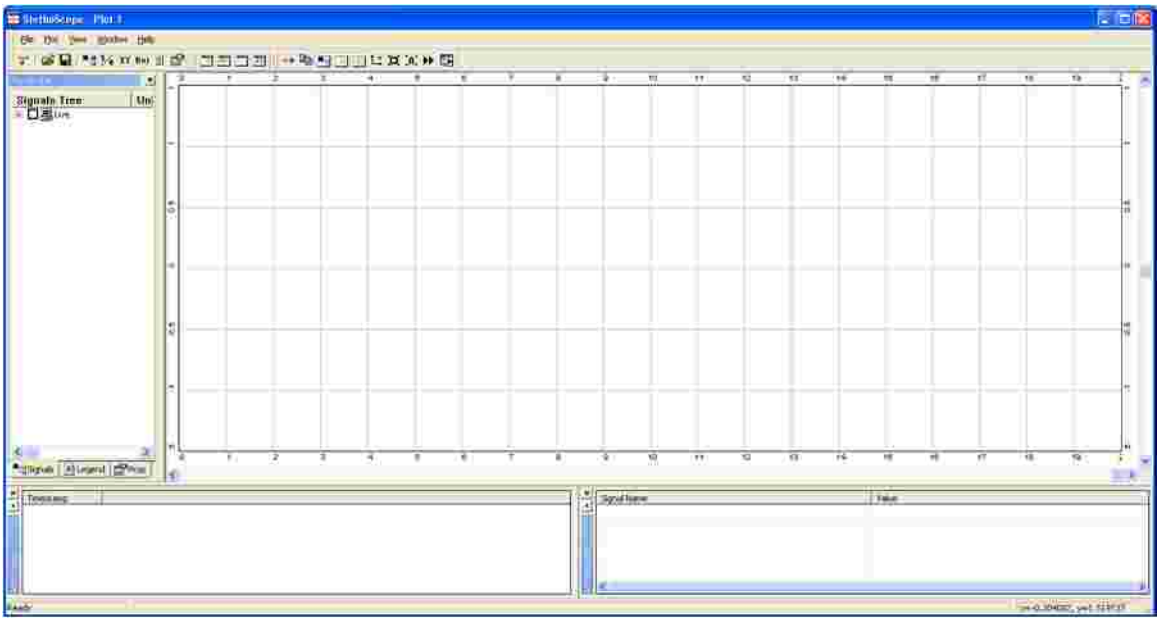


Figure 44 StethoScope GUI Plot window

Now the ROBOT module can be downloaded, and it is linked automatically to StethoScope. Start the ROBOT. In parallel with terminal printout the StethoScope window displays the progress of the robot, according to the Signals Tree window options. Figure 45 shows how **currX**, **currY** and **currZ** values change with time when the associated task executes. In Figure 46 the **Event** information has been included as well.

Options for presenting the information are chosen by means of mouse clicks on the monitored values displayed in the Signals Tree window.

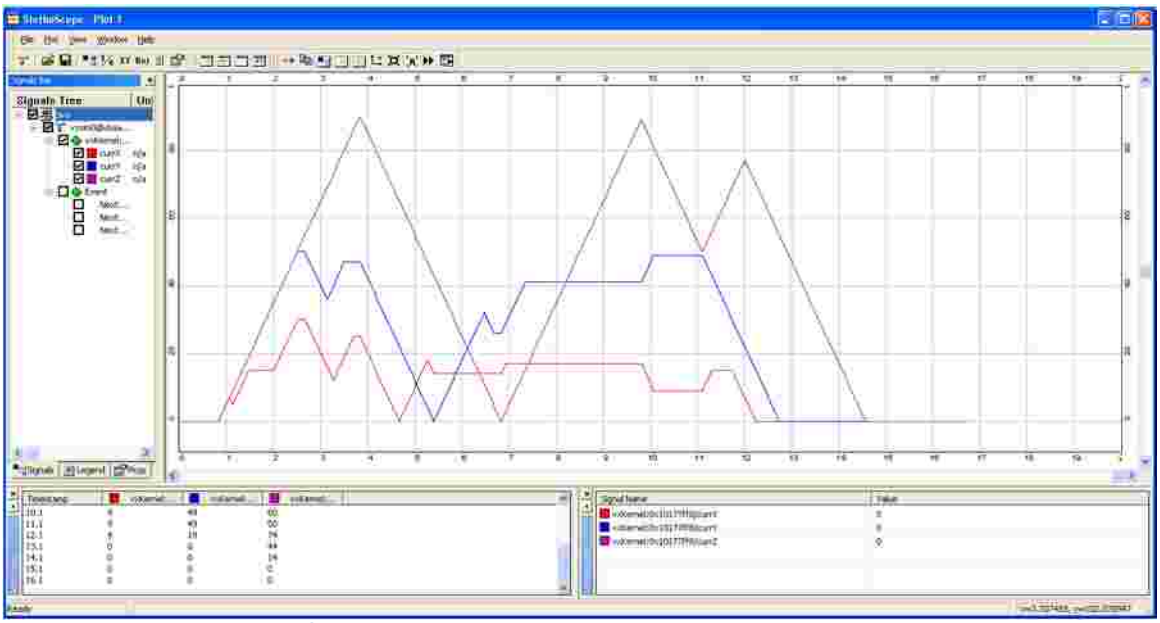


Figure 45 StethoScope Plot of currX, currY and currZ coordinates

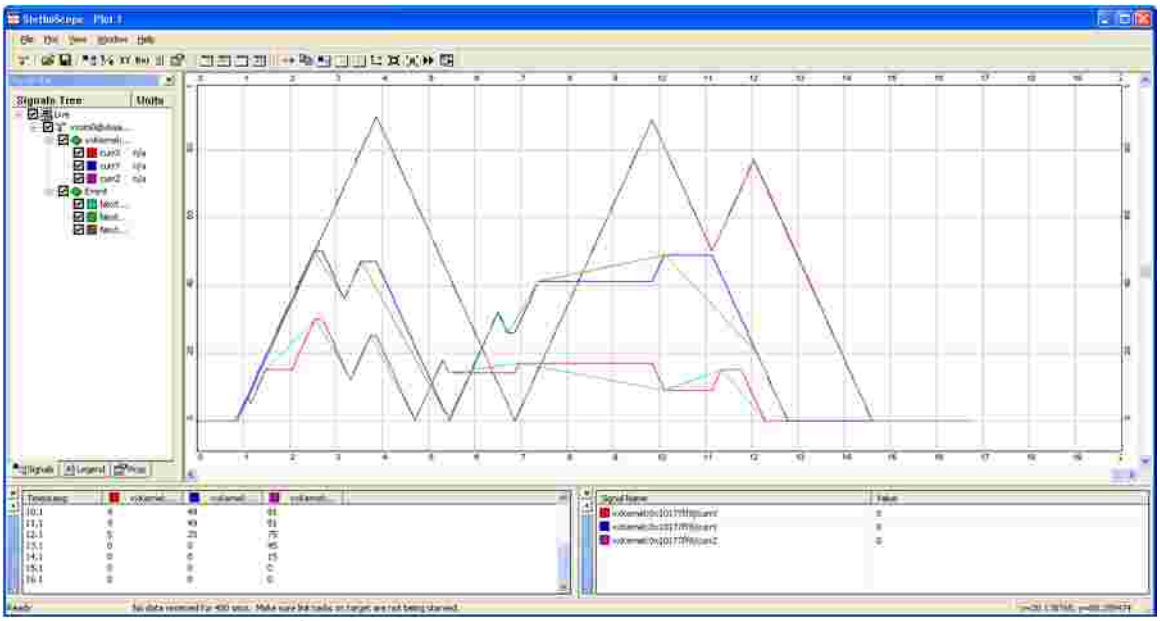


Figure 46 StethoScop Plot of coordinates and Events

5.2.2 Errors during download of project .out file

It is observed that the system sometimes crashes when a project **.out** file is downloaded again after modification and relinked with StethoScope. The reason for this not known. Disconnect the target and then reconnect again, and start from scratch.

5.3 StethoScope demonstrator

Build a project with the …\src\vxdemo.c source file, and execute it with

-> ScopeDemo 1,0,3,0,0,0