Marine cybernetics laboratory handbook

NTNU – Trondheim
Norwegian University of
Science and Technology

Faculty of Engineering Science and Technology
Department of Marine Technology

Introduction

This handbook is a comprehensive reference for the marine cybernetics laboratory. The laboratory is used in teaching and research on development and real-time testing of marine control systems.

Structure

Part I explains the concepts and motivations for the stepwise controller development.

Part II is a user guide intended for users of the laboratory. Step-by-step instructions for development and deployment of programs to the real-time controller are given.

Lower level details, intended for laboratory assistants and customized use, are given in Part III.

Contents

Chapter 2

Control system development philosophy

As the complexity of marine vessels and operations grows, the need for thorough testing and verification of the vessel real-time control and monitoring systems increases. More advanced integrated functionality relies on many separately designed control and monitoring systems to cooperate on performing common tasks. Regular software simulations cannot cover all aspects of this complexity.

Through steps-wise verification and validation at different levels of fidelity, errors can be discovered at earlier stages thus lowering the total development cost.

In the case of the MC Lab, users qualify their experimental setups before the assigned laboratory time. This reduces debugging time, improves tuning of parameters and test scenarios, thereby increasing efficiency and maximizing the outcome of the experimental work.

2.1 Development Steps

Marine cybernetics deals with control engineering for the vessel mechatronic systems which again interact with the environment. In this section, “the controller” refers to the designed control software and “the plant” to the combination of the mechatronic system and the environment.

2.1.1 Model-in-the-Loop

2.1.1.1 Principle

A model of the controller interconnected with a physical model of the plant, in a control development environment, such as MATLAB Simulink.

2.1.1.2 Aim

Develop control strategies . Test principles.

2.1.1.3 Iteration time

Extremely short, small changes are immediately implemented and tested.

2.1.1.4 Cost

Low

2.1.2 Software-in-the-Loop

The controller is coded in the final language, such as C or C++, and connected to the plant model in a control development environment.

2.1.2.1 Aim

Test of coding system. Reveal coding failures.

2.1.2.2 Iteration time

Slightly longer than MIL.

2.1.3 Processor-in-the-Loop

2.1.3.1 Principle

The controller is deployed to a representative microprocessor, connected to the plant simulation via high speed bus, such as JTAG. The plant must be synchronized with the controller.

2.1.3.2 Aim

Expose problems with execution in the embedded environment, such as insufficient computing resources on the embedded processor.

2.1.3.3 Iteration time

Higher, due to the need to regenerate and deploy code for each run.

2.1.4 Hardware-in-the-Loop

2.1.4.1 Principle

Controller fully installed into the intended final hardware, connected through the plant only through the proper IO. The plant simulator must run on a real-time computer emulating the IO of a real process.

2.1.4.2 Aim

Perform regulation, security and failure tests without risk.

Investigate the interaction between subsystems.

Ensure a high level of robustness and quality.

2.1.5 Scale test

-

2.1.6 Full scale

-

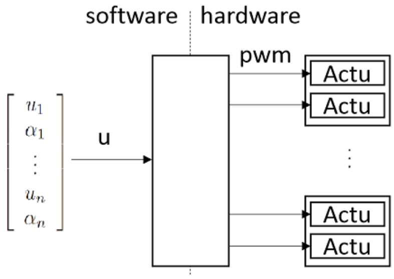

Figure 2.1: Individual actuator control

2.2 Recommended Control Modes

It is favorable to allow for five main control modes:

- Stop all actuators

1. Individual actuator control
2. Generalized force control
3. Regulation
4. Operations

In addition, sub-modes may allow more functionality.

2.2.1 Individual actuator control

The most basic mode allows controlling each thruster separately. Inputs are typically normalized force $u = [-1, 1]$, angle $\alpha = [-\pi, \pi]$, and sometimes normalized rotational speed $\omega = [-1, 1]$. The software computes the corresponding physical signal, for instance a pulse width manipulated (PWM) signal as illustrated in Figure 2.1.

The user interface may be through gamepad, computer, tablet, etc.

Implementation details are discussed in Appendix C.2.1.

2.2.2 Generalized force control

Thrust allocation allows input of the desired generalized force, as seen in Figure 2.2. For six degrees of freedom (6 DOF) control the input is

$$\tau = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix}.$$

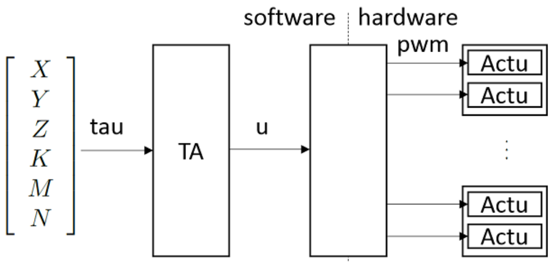

Figure 2.2: Generalized force control

For surface craft, 3 DOF are typically considered:

$$\tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}.$$

The user interface may be through gamepad, computer, tablet, etc. The appropriate reference frame depends on the application.

2.2.2.1 Body frame

Most commonly, the desired thrust is given in the vessel-fixed body frame. This is the intuitive setup for an on-board operator.

Implementation details are discussed in Appendix C.2.2.1.

2.2.2.2 Inertial frame

For remote operation, it may be suitable to input the force with regard to the inertial frame, rather than the vessel orientation.

Implementation details are discussed in Appendix C.2.2.2.

2.2.2.3 User frame

When the operator has eye contact with the vessel, it may be suitable to specify the force with respect to the line of sight between the operator and craft.

Implementation details are discussed in Appendix C.2.2.3.

2.2.3 Regulation

Maintaining a given value in one or several DOFs under the influence of disturbances is the basic automatic control mode. The given value is called setpoint, as illustrated in Figure 2.3. Typical sub-modes are listed in Table 2.1. Reference filters for changing setpoints may or may not be included.

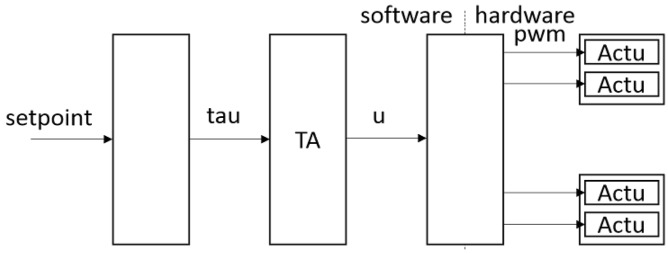

Figure 2.3: Regulation

	x	y	z	ϕ	θ	ψ
Stationkeeping	✓	✓				✓
Heading				✓		
Depth			✓			
Roll/Pitch				✓	✓	

Table 2.1: A selection of regulation modes

The user interface typically allows inputting the setpoint value directly, for instance on a computer or tablet. Alternatively, the setpoint may be translated through gamepad.

Further details are discussed in Appendix C.2.3.

2.2.4 Marine Operations Control

The more complex control modes, typically combined from several of the different submodes, give automatic functions that are important for different marine operations. Input varies depending on the operation. It may be maps or waypoints, as in Figure 2.4.

Further details are discussed in Appendix C.2.5.

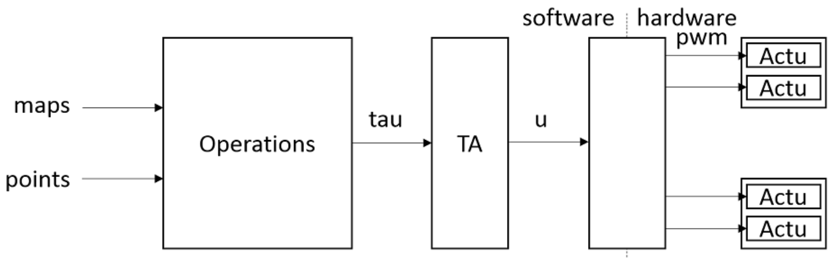

Figure 2.4: Marine Operations Control

Appendix A

Simulation and control with cRIO

A.1 Simulink model adaptation and compilation

Complete the following steps to convert your model you created in Simulink into a compiled model that runs on RT targets.

Version compatibility is an issue for VeriStand-Simulink interaction. Mostly[1] Simulink code may be programmed in any version of the MATLAB, compilation, on the other hand, can only be done in version compatible with the intended VeriStand version. See Section 12.3.1.

A.1.1 Modeling

A.1.1.1 Input and output

In order for the model to interact with VeriStand, special input and output blocks must be added to the block diagram[2]. These are found in the Simulink Library Browser under NI VeriStand Blocks.

A.1.1.2 Initial conditions

If the simulation is to be run with different initial conditions, one possible method is to allow external reset of the integrators. This is done right-click the integrator and selecting Block Parameters (Integrator) in the drop-down

[1] It has been experience that MATLAB function blocks are not compatible across versions. This results in build error message “invalid object ID”. The MATLAB function block code must then be copied and pasted into a new MATLAB function block from the compatible version Simulink Library Browser.

[2] Ordinary input/source and output/sink blocks could be used at the diagram top level. However, subsystem ports are only available when using the VeriStand blocks.

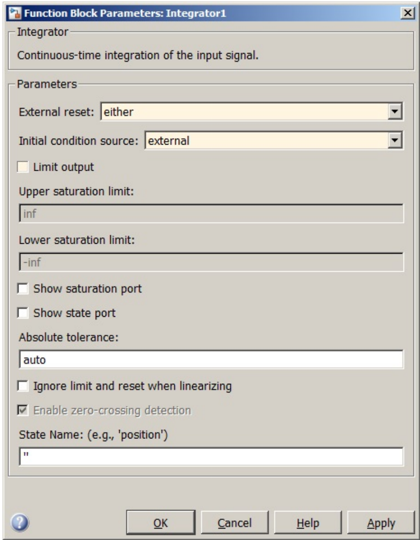

Figure A.1: Integrator function block parameters

menu. Here, the reset condition is set. The initial condition source should be external, as in Figure A.1.

A.1.1.3 Real-time data logging

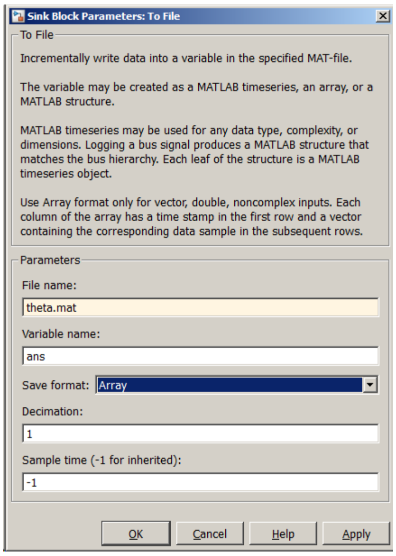

Figure A.2: To File block parameters

Model output can be saved to the cRIO, for later retrieval through FTP, during simulation through a To File block. This block is found in the Simulink Library Browser under Sinks. The output file name is specified under the block parameters, as in Figure A.2. The format should be set to Array, since the cRIO does not support the Timeseries format.

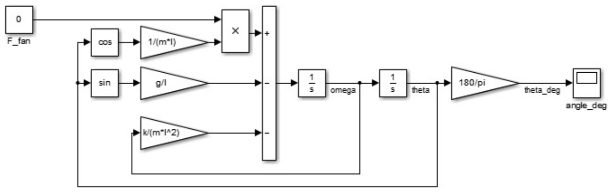

Figure A.3: Simulink model for offline simulation

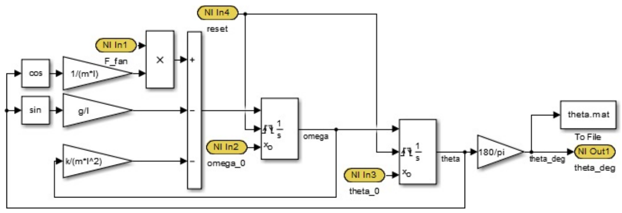

Figure A.4: Simulink model for adjusted for compilation

Example: For a simple pendulum, $\dot{\omega} = -\frac{g}{l}\sin(\theta) - \frac{k}{ml^2}\omega + \frac{F_{\text{fan}}}{ml}\cos(\theta)$, the offline simulation block diagram could look as Figure A.3. Figure A.4 shows the same system adapted for VeriStand input, including reset and initial conditions, and output. The VeriStand blocks are yellow. omega_0 and theta_0 are ports corresponding to the initial conditions $(\omega(0), \theta(0))$. The integrators take these values whenever reset is rising or falling.

A.1.2 Model configuration

The code generation toolbox compiles the Simulink diagram to an output shared library in *.out format[3]. Model configuration parameters must be adjusted before generating, or building, the code.

The solver stop time should be `inf` (infinity) if the model is supposed to run until it is otherwise interrupted. The solver type must be fixed step. If your model only performs arithmetical operations, such as a mapping or transformation module would, the discrete solver should be used. If the model contains continuous states, i.e. if you have integrators, choose some differential equation solver such as ode3 or ode4. See Figure A.5. Finally, the step size can be set: for a target running at 100 Hz, such as the cRIO-9024 default, a 0.01 step size results in the model running in simulating 1 second pr. second[4].

[3]The *.out format is for targets running Wind River VxWorks real-time operating system (RTOS) such as cRIO-9024, while dynamic link libraries in *.dll format are for targets running IntervalZero Phar Lap ETS RTOS such as cRIO-9081.

[4]This can also be achieved by use of decimation, as described in Section A.2.2.

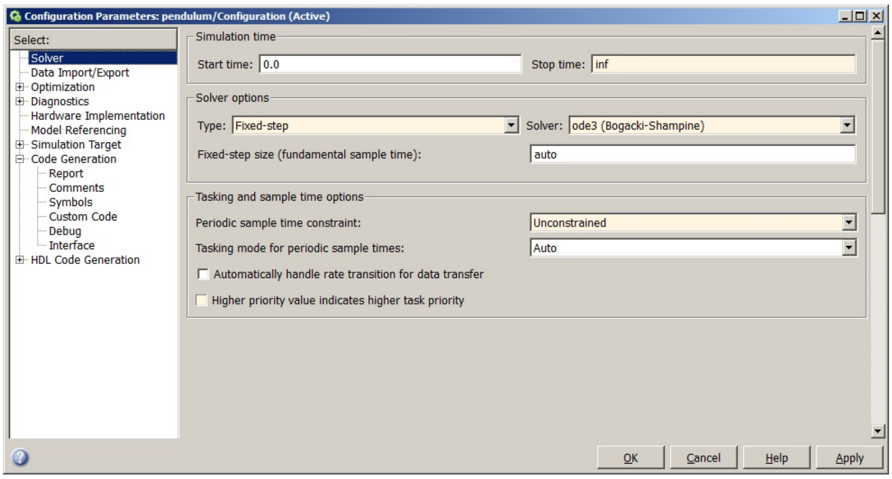

Figure A.5: Simulink configuration parameters - solver

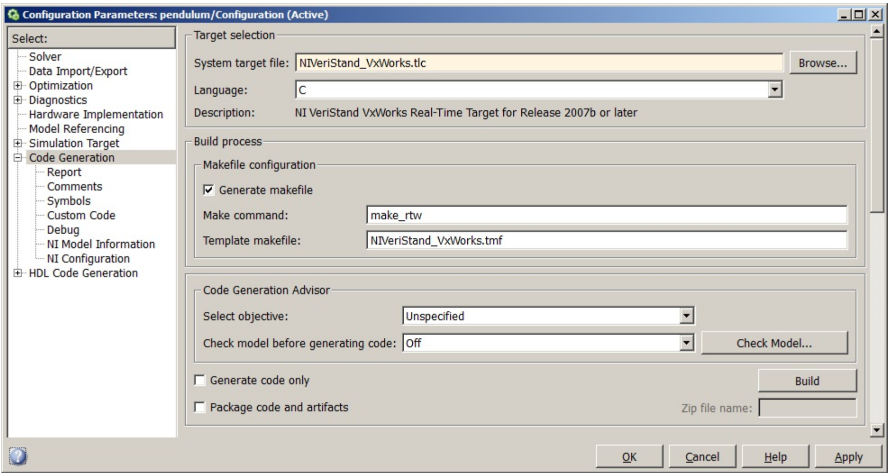

Figure A.6: Simulink configuration parameters - target selection

The correct target file should be selected depending on the target device. Select `NIVeriStand_VxWorks.tlc` for VxWorks targets[5], such as cRIO-9024, as in Figure A.6.

The WindRiver GNU Toolchain must be present in the folder specified under NI Configuration, as in Figure A.7.

A.1.3 Build

The build output is placed in a subfolder in the MATLAB Current Folder. The desired folder must therefore be active in the MATLAB main window, as in

[5]For PharLap targets, select `NIVeriStand.tlc`.

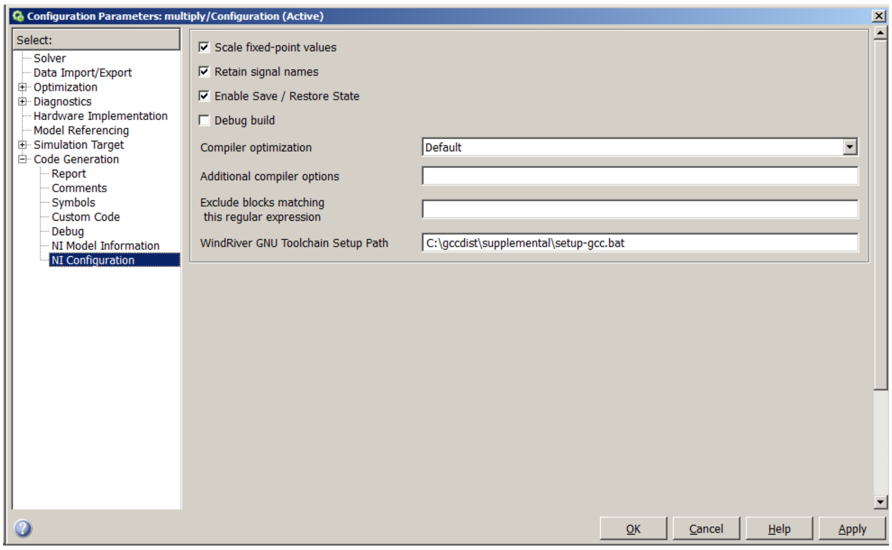

Figure A.7: Simulink model configuration - NI configuration

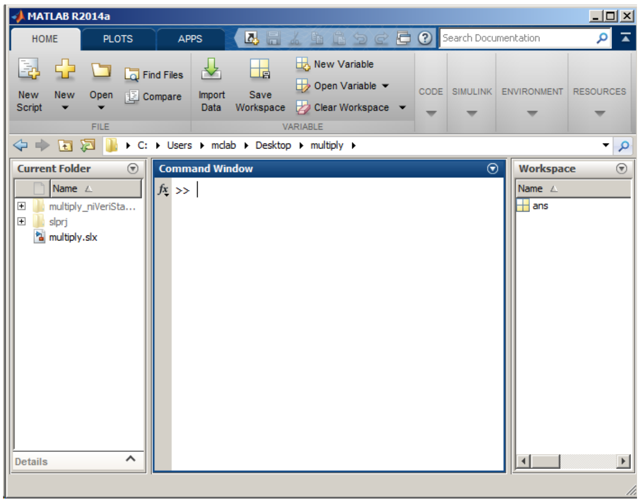

Figure A.8: MATLAB console

Figure A.8, before compiling. The build subfolder name is [simulink model name]_niVeriStand_VxWorks_rtw.

The build is done in in Simulink, either with the Build button in the configuration window, by clicking the button, by the key combination CTRL+B, through the menu Code >C/C++ Code >Build model, or by pushing the icon button.

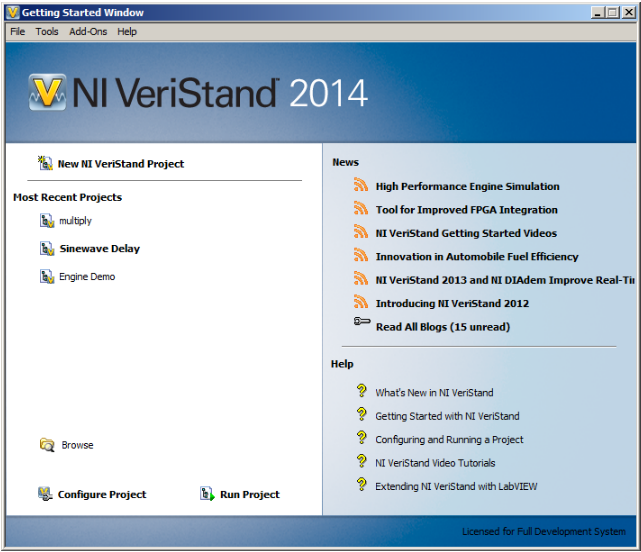

Figure A.9: VeriStand start screen

A.2 Simulation configuration

Simulations are set up, deployed and interfaced through VeriStand. Figure A.9 shows the start screen. Already configured projects can be run directly from here, or reconfigured.

A.2.1 Project creation

To deploy model for the first time, click New NI VeriStand Project. Give your new project a suitable name and location. Clicking OK creates the project files

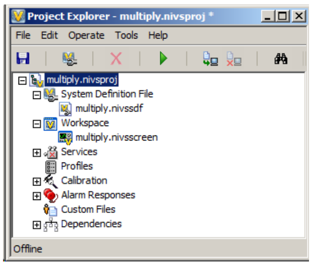

Figure A.10: VeriStand Project Explorer

in given location and opens the Project Explorer, as in Figure A.10. In this section, the example project name is multiply.

A.2.2 System setup

To configure the setup which will run on the cRIO, open the System Explorer by double-clicking the system definition file [project name].nivssdf.

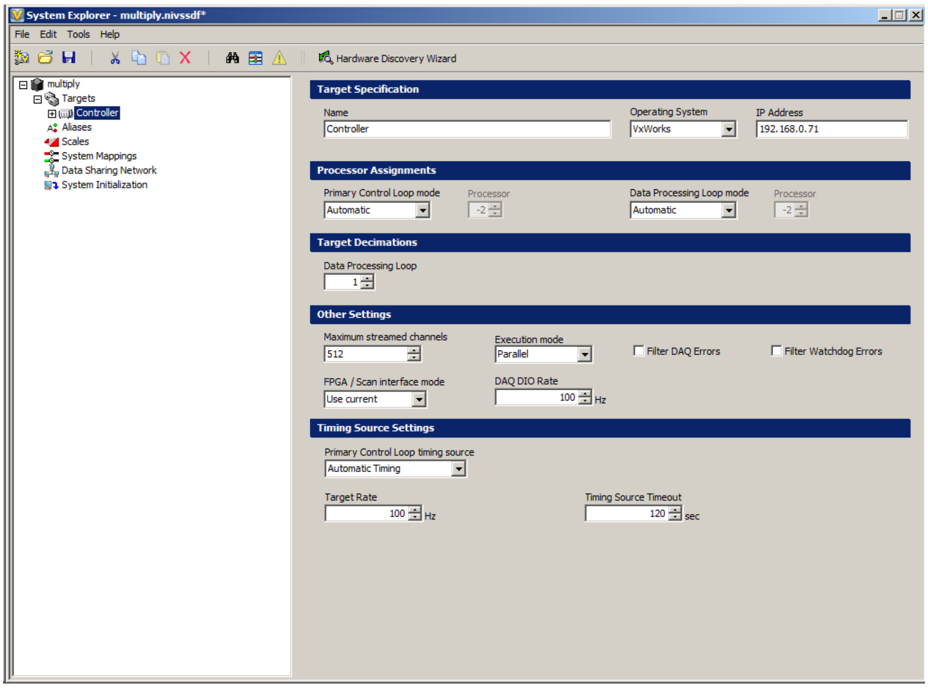

Figure A.11: VeriStand - System Explorer - Controller

1. Set the correct controller operating system and IP address, as in Figure A.11. All HIL and MC lab IP addresses are given in Table B.1. Also, note the target rate.

2. Click Add a Simulation Model, as seen at the top of Figure A.12. Browse to the output of the Simulink compilation, as seen in Figure A.13. Finally, click Auto Select Decimation to make sure the model runs at the intended rate.
   Repeat if several models should run simultaneously.

3. Add custom devices, such as network input, by right clicking the custom device pane and choosing the required device[6]. Figure A.14 shows an example with the Sixaxis (WL_Joystick) device. Upon selection, a subfolder with the device name appears in the tree with signals listed inside it.

4. Configure mappings, by pushing the  icon at the top of the window, to connect signals between custom devices, FPGA and models. Expand the trees to find the desired signals and click Connect, as in Figure A.15.

5. Save and close to return to the Project Explorer.

[6]If the required device is not present, refer to the device driver installation instructions in Section 12.1.4.

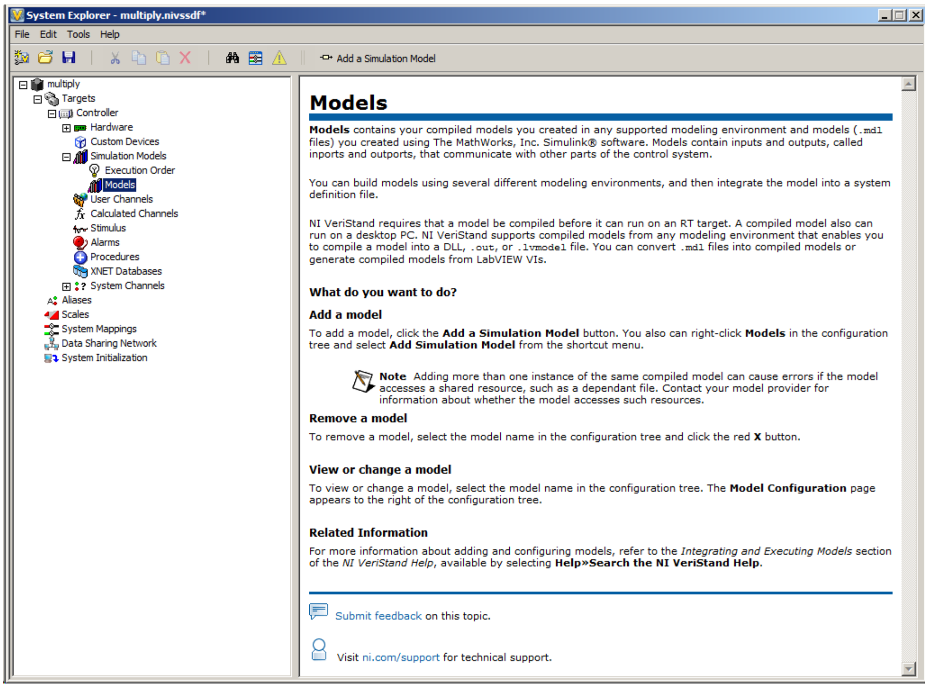

Figure A.12: VeriStand - System Explorer - Models

A.2.3 Create computer interface

To configure the computer interface, open the Workspace editor by double-clicking the workspace file [project name].nivsscreen. The blank workspace pops up.

1. Enter Edit mode by Ctrl+M or Screen >Edit Mode.
2. Click the Workspace Control pane on the left side to access indicators, controls and such.
3. Drag and drop the desired item to the desired position in the workspace. Select the corresponding signal in the pop-up dialog.
4. Close the Workspace editor.

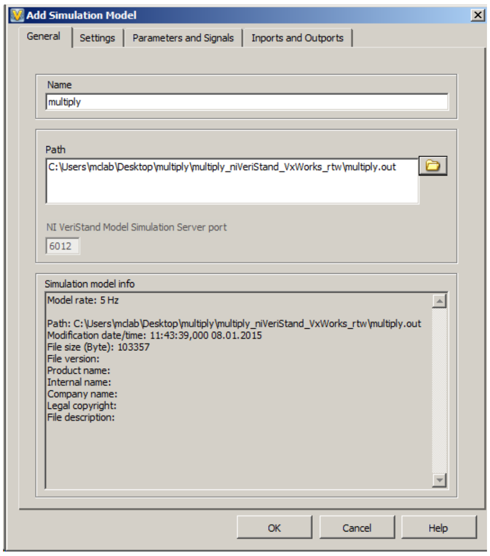

Figure A.13: VeriStand - System Explorer Model

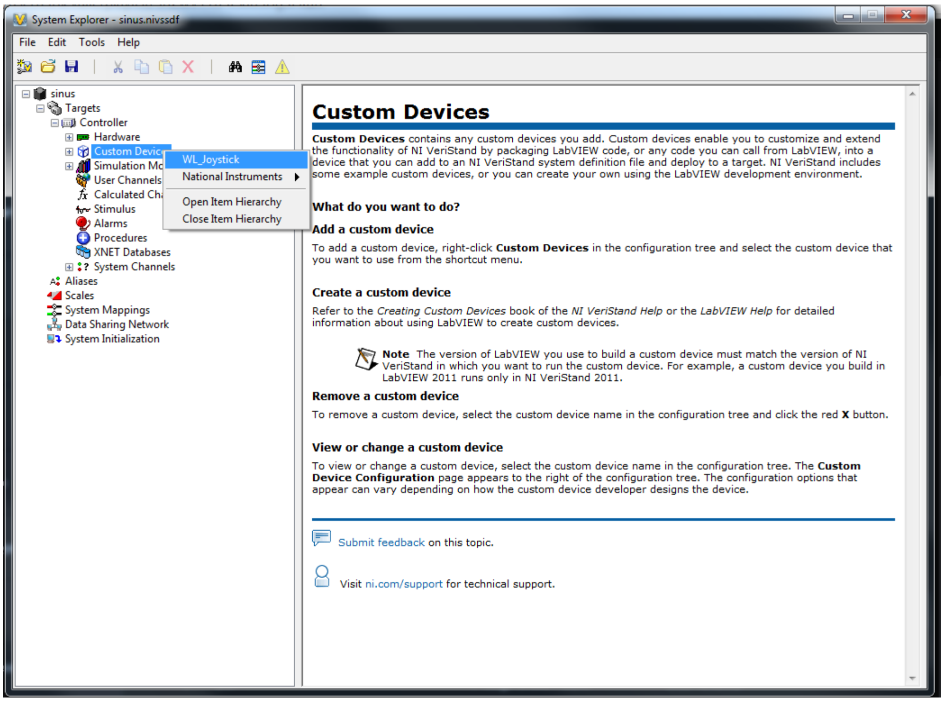

Figure A.14: Custom device selection

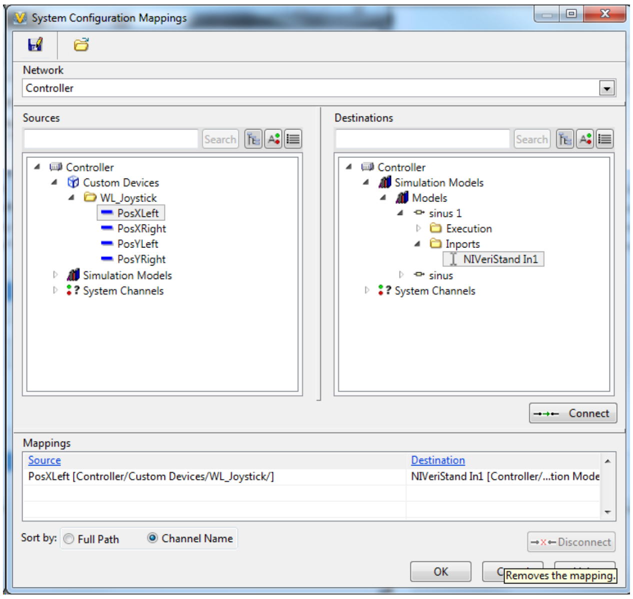

Figure A.15: VeriStand System Configuration Mappings

A.3 Deployment and simulation

A.3.1 Run

Deploy by tapping the F6 key, or button, or Operate >Deploy. A dialog box appears. Upon successful deployment, the workspace pops up.

A.3.2 User interface side data logging

For reliability, it is recommended to log data directly on the cRIO during simulation, as described in Section A.1.1.3. It is also possible to log via the laptop user interface.

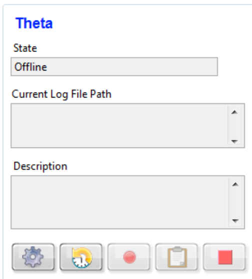

Figure A.16: Logging Control

A Logging Control, as seen in Figure A.16, must be added to the workspace to export data from the simulation. The control is added as described in Section A.2.3.

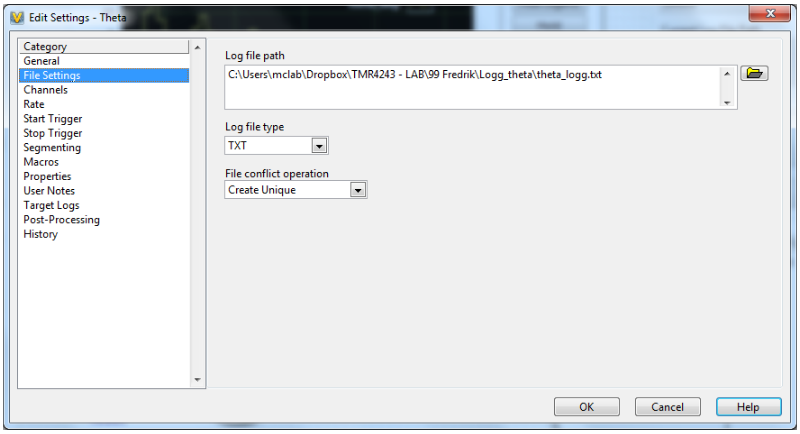

Figure A.17: Logging Control file settings

Once the control is added, a pop-up window allows to edit the settings. The log file path is specified under File Settings, see Figure A.17. Under Channels, the desired channels can be selected and added, as in Figure A.18.

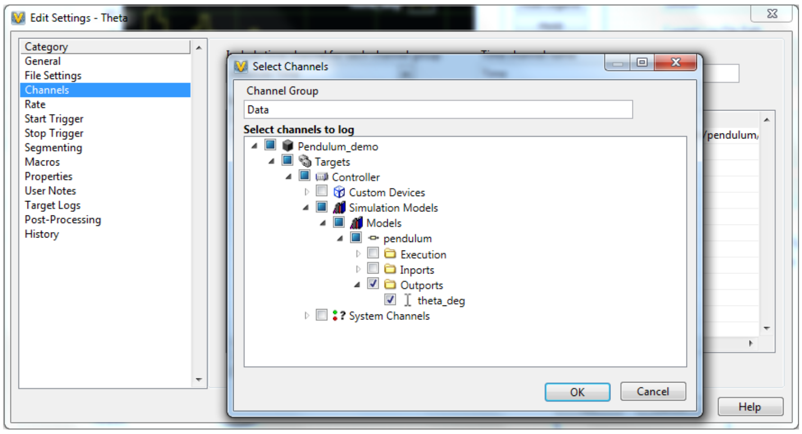

Figure A.18: Logging Control add channel

A.3.3 Stop

 button

A.3.4 FTP data retrieval

Data logged on the cRIO through To File blocks can be retrieved after simulation over FTP with software such as WinSCP.

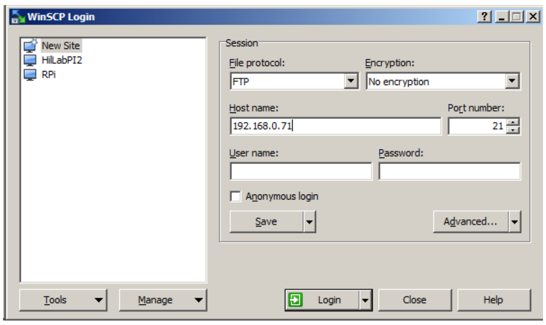

Figure A.19: WinSCP login

To connect to the cRIO, the correct IP must be specified, as in Figure A.19. For the standard HIL setup, the user name and password are blank.

Logged data with file names corresponding to the To File block names are located on the cRIO root, as seen in the right pane of Figure A.20. Data is transferred to the laptop by drag and drop to the desired location in the left pane.

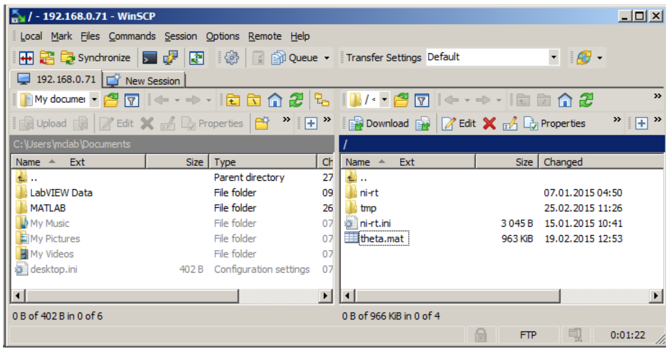

Figure A.20: WinSCP

Appendix B

Device network addresses

Qualisys PCs	192.168.1.10	surface
	192.168.1.20	underwater
RPi	192.168.1.22	for all
cRIO primary ethernet	192.168.0.71	iimt-HILlab1-cRIO
	192.168.0.72	iimt-HILlab2-cRIO
	192.168.0.73	iimt-HILlab3-cRIO
	192.168.0.76	CSE1
cRIO secondary ethernet	192.168.1.21	for all
Laptops	192.168.0.41	iimt-HILlab1-PC
	192.168.0.42	iimt-HILlab2-PC
	192.168.0.43	iimt-HILlab3-PC
	192.168.0.47	MClab
Subnet mask	255.255.255.0	for all

Table B.1: IP addresses

All RPis and have the same IP address, but there is no IP conflict since the cRIO-RPi networks are separate and closed. The same goes for the cRIO secondary ethernet ports

Note: to connect the RPi directly to the computer, both need to be on the same domain and the computer IP thus needs to change to 192.168.**1**.xx.