# Marine Cybernetics
# Laboratory Handbook

**NTNU – Trondheim**
Norwegian University of
Science and Technology

Faculty of Engineering Science and Technology
Department of Marine Technology

# Device network addresses

| | | |
|---|---|---|
| **Qualisys PC** | 192.168.0.10 | surface |
| | 192.168.0.20 | underwater |
| **MCLab router** | 192.168.0.50 | SSID: MC Lab |
| **RPi** | 192.168.1.22:51717 | for all |
| **cRIO primary port** | 192.168.0.71 | IIMT-HILLAB1-cRIO |
| | 192.168.0.72 | IIMT-HILLAB2-cRIO |
| | 192.168.0.73 | IIMT-HILLAB3-cRIO |
| | 192.168.0.75 | CSE1-cRIO |
| | 192.168.0.55 | CSAD-cRIO |
| **cRIO secondary port** | 192.168.1.21 | for all |
| **Assigned by DHCP** | From 192.168.0.100 | |
| | to 192.168.0.254 | |
| **Subnet mask** | 255.255.255.0 | for all |

# Introduction

This handbook is Version 2.0 of the Marine Cybernetics Laboratory Handbook. The handbook gives a description of the equipment in the MCLab, and instructions on some of the software used. Every student using the MCLab should read this document before using equipment in the lab. In addition to this Handbook, there are separate handbooks for CS Enterprise 1 and CS Arctic Drill Ship, which are found on GitHub. As of July 2017, the other vessels do not have a separate Handbook, and thus the reader is referred to the old MCLab Handbook for complementary information about the specific vessel.

# Structure

Chapter 1 describes the different equipment found in the Laboratory, and how to use it.
Chapter 2 gives a description of some of the software used in the lab. This includes formatting and installing software on cRIO, creating FPGA modules, mapping of channels in VeriStand etc.

# Contents

# Chapter 1

# Marine Cybernetics Laboratory equipment

## 1.1 Introduction

The laboratory is equipped for experimental testing of marine control systems and hydrodynamic tests. It consists of a wave basin with an advanced instrumentation package and a towing carriage. The basin, depicted in Figure 1.1, has dimensions 40m x 6.45m x 1.5m (LxBxD). The laboratory consists of the following fixed equipment:

- Qualisys Motion Capture System
- Towing Carriage
- Wave Generator
- Video-camera

These parts are thouroghly described in the next sections, with a user manual and technical description of the equipment.

### 1.1.1 Safety

#### 1.1.1.1 Personnel injury

**Drowning** It is required to have two or more persons present when using the basin.

**Electric shock** The towing catenary should not be approached or touched.

**Carriage collision** It is forbidden to run the towing carriage when there are people alongside the basin.

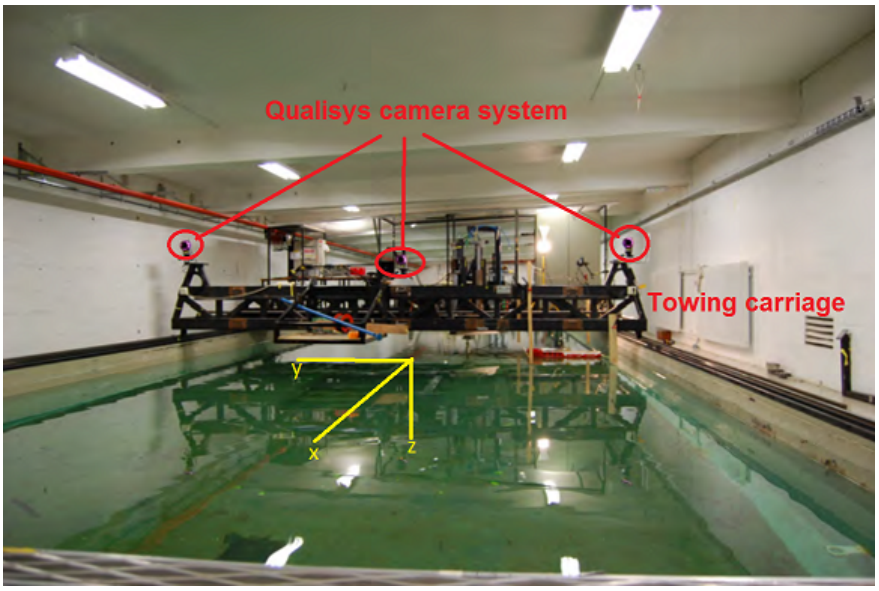


Figure 1.1: Marine cybernetics laboratory basin

**Thruster blade cuts** Vessels must stay in the water as long as actuators are active. Before removing the vessels from the water, the control system must be stopped and disabled, for instance by undeploying in the VeriStand project.

#### 1.1.1.2 Material damage

**Towing carriage** Stop before automatic stop at high speeds.

**Towing carriage rails** The rails are made of steel, not stainless. Hence, avoid getting water on the rails, as this will lead to corrosion.

## 1.2 Qualisys Motion Capture System

Qualisys provides 6 degrees of freedom data tracking. The system has millimeter precision, works in real time and is configured to 50Hz.

The positioning system consists of three Oqus high speed infrared cameras registering infrared reflectors placed on the vessels. Peer-to-peer (P2P) networking is used to transmit camera data to a dedicated computer running Qualisys Track Manager (QTM) software. QTM performs triangulation and broadcasts the vessel position over the wireless network. The system is operated from a dedicated PC in the Lab, labeled with *QTM Surface*.

Add info on zyx-convention

### 1.2.1 User Manual

#### Start Qualisys Track Manager

1. Execute the program. The first displayed window is as in Figure 1.2.

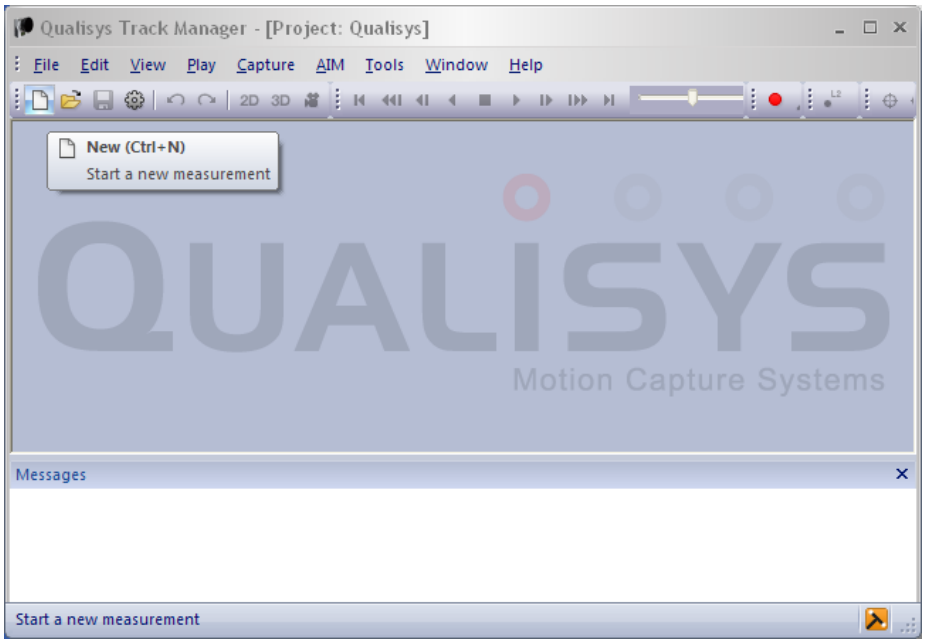


Figure 1.2: Qualisys Track Manager start window

2. Push the white sheet icon to start a new measurement. The main window should then display the 2D view, as in Figure 1.3. The squares numbered #1, #2 and #3 show the basin as seen from the respective cameras. The white dots are the vessel reflectors. A minimum of three reflectors must be visible in each camera.

#### Aquire body

1. Push the gear icon to access Project Options. Navigate to 6DOF Tracking, as in Figure 1.4.

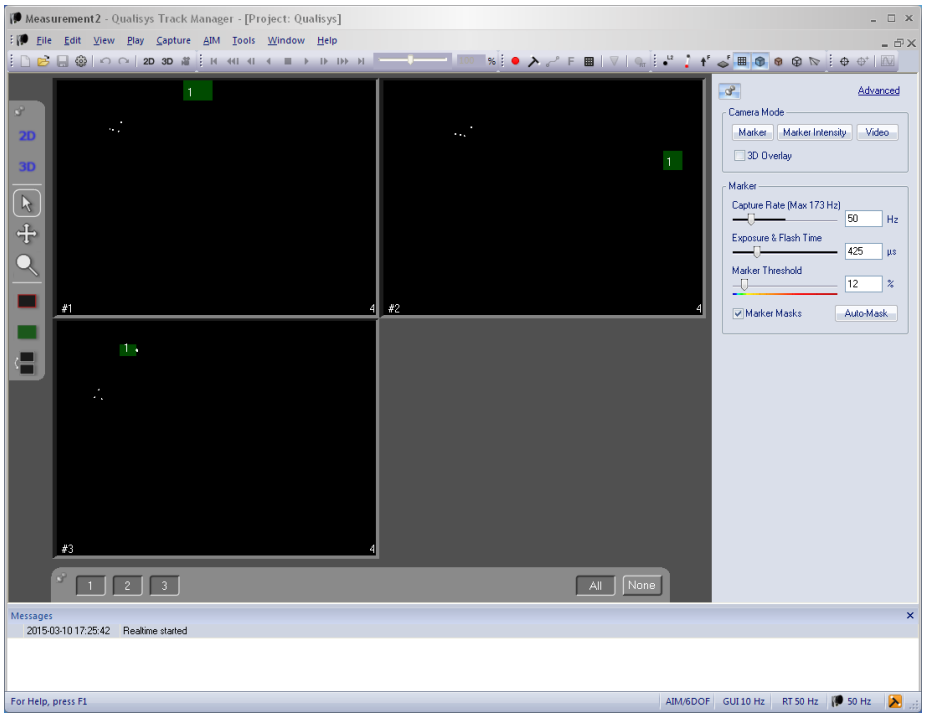


Figure 1.3: Qualisys Track Manager 2D view

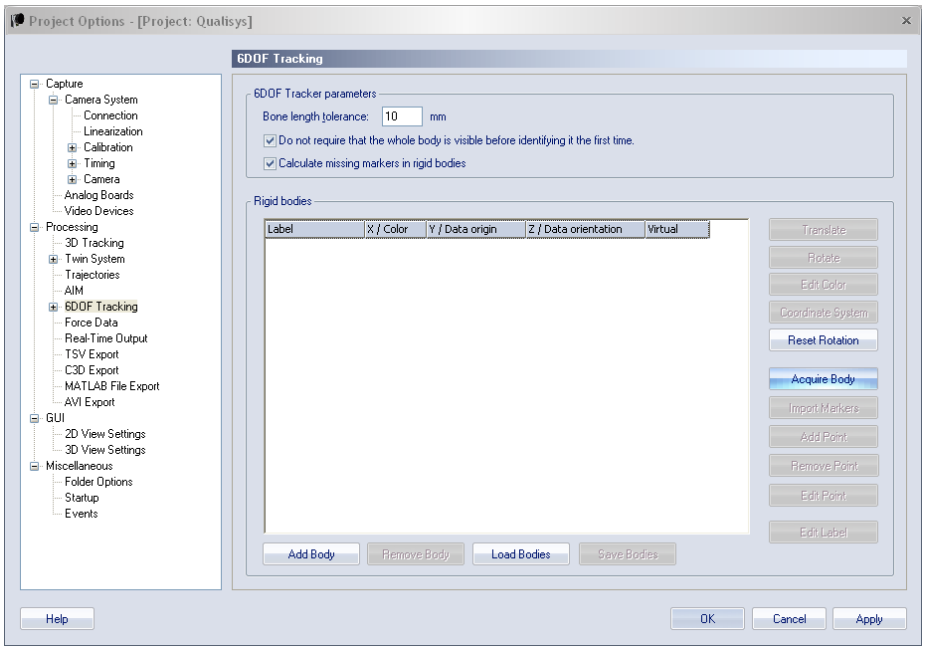


Figure 1.4: Qualisys Track Manager 6 DOF Tracking

2. Remove previous bodies, if any.

3. Align the vessel with 0°heading (i.e. with the bow pointing towards the command center) and push "Acquire Body" to get the position of the reflectors. A list appears, as in Figure 1.5.

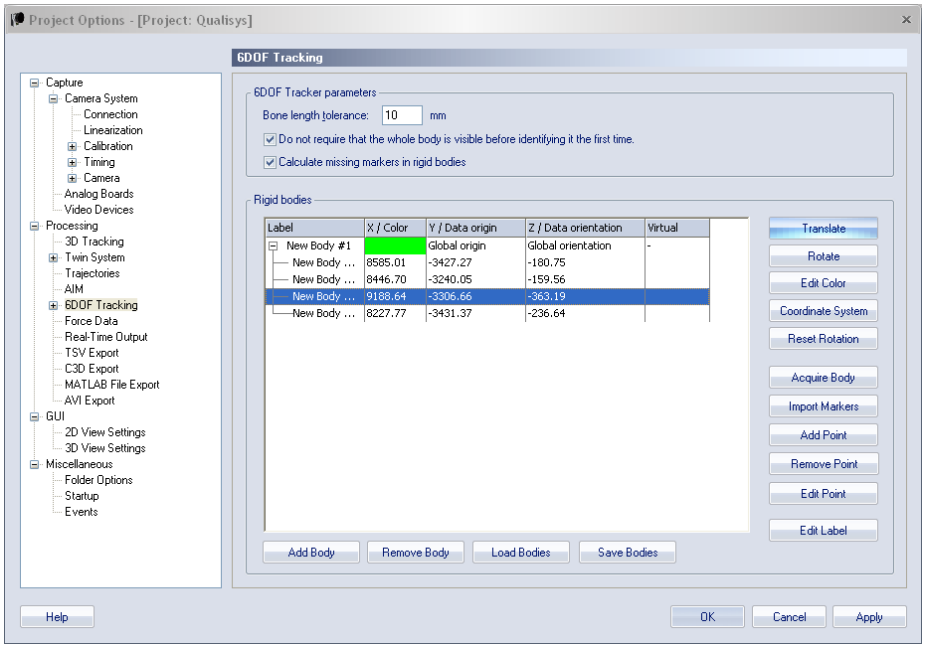


Figure 1.5: Qualisys Track Manager aquired body

4. To redefine the body fixed coordinate frame, choose a reference reflector. As highlighted in Figure 1.5, it may be practical to choose the highest-most, in this case reflector 3. Push "Translate" and enter the coordinates of the chosen reflector in the desired frame. See the Handbook for the specific vessel for information of the position of this reflector in the body-frame.

5. Verify that Qualisys is set to *zyx*-convention for the rotations. In the left-pane navigation tree, expand 6DOF Tracking, select Euler Angles. Figure 1.6 show the correct setting for *zyx*-convention.

6. Finally, select 3D view to confirm that the body-fixed frame is indeed located as desired, as in Figure 1.7.

### Troubleshooting

**Long waiting for camera** If the operation depicted in Figure 1.8 is unsuccessful after a couple of minutes, reboot the cameras by unplugging and reconnecting the power cord on the rack.

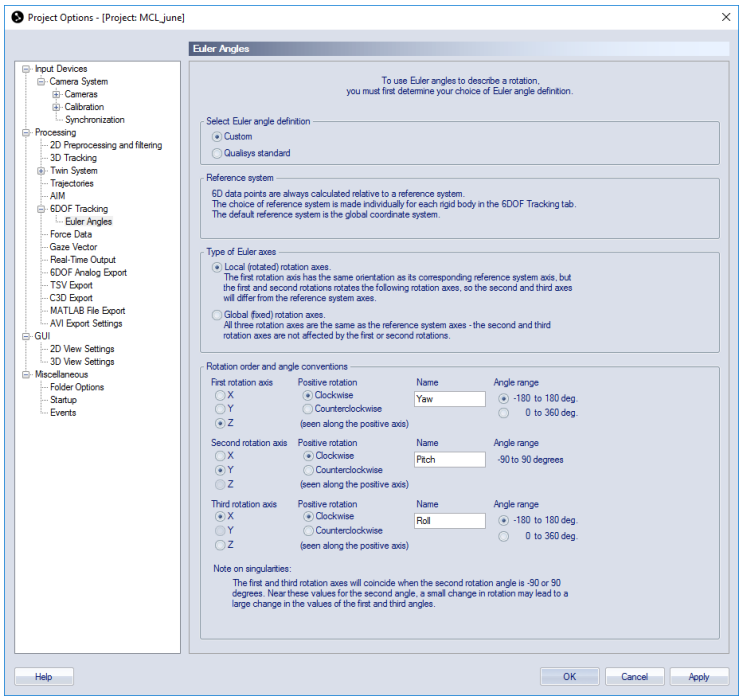


Figure 1.6: QTM rotation convention

**Reflectors visible in 2D, but not in 3D window** When aqcuiring the body, the vessel must be in the calibrated area for the cameras. If the reflectors are visible for all 3 cameras in 2D visualization, but not in 3D visualization, verify that the vessel is placed inside the calibrated area. In 3D, on the left pane, press the box and choose to show calibrated area, as illustrated in Figure 1.9

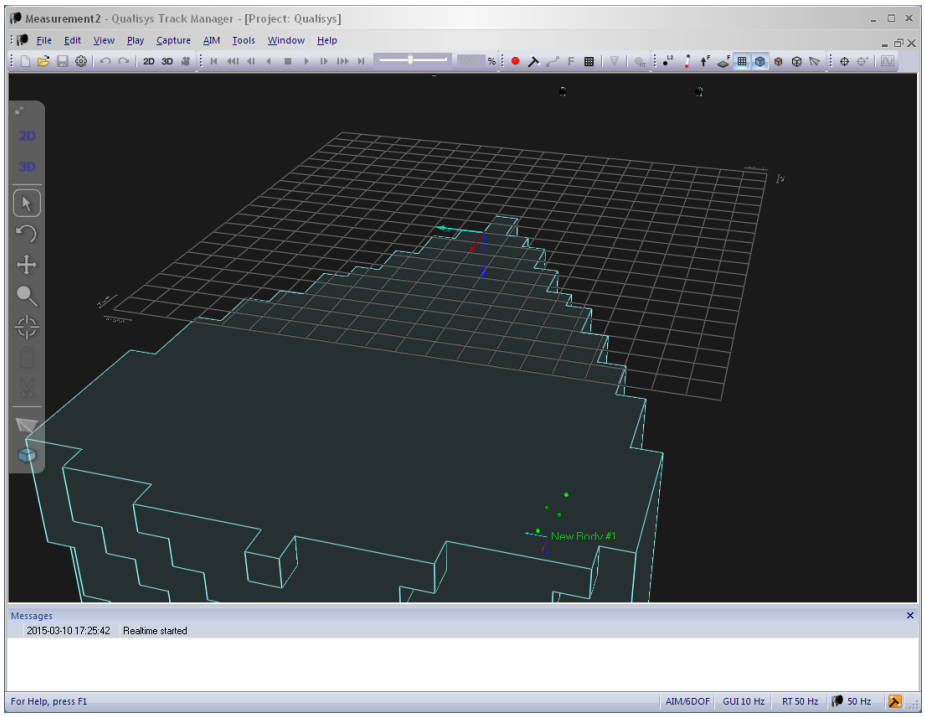


Figure 1.7: Qualisys Track Manager 3D view

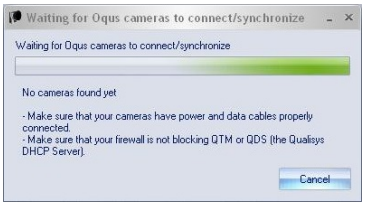


Figure 1.8: Qualisys Track Manager waiting for cameras

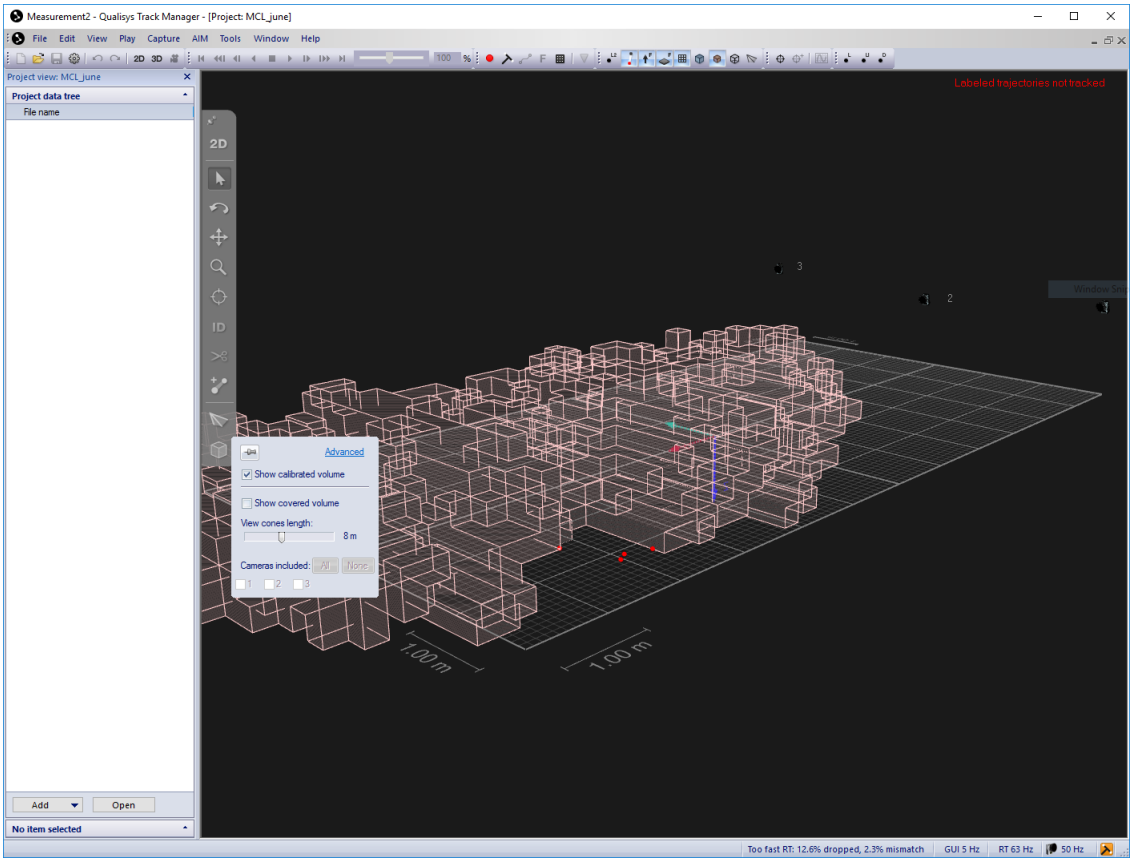


Figure 1.9: QTM calibrated area

### 1.2.2 Importing Data from the Qualisys System into ROS and MATLAB(Linux)

The following approach may be used to read Qualisys data into ROS and MATLAB. The method is convenient for Qualisys data into to MATLAB independent on whether the rest of the system use ROS or not. The method, as described is limited to Linux-operating systems.

The first part of the manual describe how import Qualisys data into ROS, while the second part describe how to get the data from ROS into MATLAB/Simulink.

Please note the following:

- In the manual, the dollar sign $ indicate a line of text that should be written in the Linux- terminal window.
- In the manual gedit is used as text editor. This can be replaced with the readers favourite text editor.
- The manual is written and tested for ROS-Indigo and MATLAB 2015b. It is based on MATLAB's manual for importing custom messages (MathWorks, 2016), which is and adapted and expanded to fit that of the MC lab and the Qualisys system.
- You need MATLAB version 2015a or newer in order to proceed with the MATLAB section of the manual.

#### 1.2.2.1 Manual

If not already installed on the machine you should start by installing ROS. Follow the instructions on the ROS download page: `http://wiki.ros.org/indigo/Installation/Ubuntu`.

You should now make a ROS workspace in your home directory:

```
$mkdir -p ~/catkin_ws/src
$cd ~/catkin_ws/src
$catkin_init_workspace
```

You now need to make sure that one are sourcing the setup.bash file in your ROS workspace each time you open your terminal window. This can be done by changing the bash file with the following command:

```
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Now import the Qualisys driver from GitHub. (The driver (KumarRobotics, 2016) is avaiable through the Apache License )

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/KumarRobotics/qualisys
$ cd ~/catkin_ws
$ catkin_make
```

Open the qualisys.launch file in a text-editor

```
$ sudo gedit ~/catkin_ws/src/qualisys/launch/qualisys.launch
```

Edit the ip address and port number for the Qualisys system. (As of March 2016 the IP is: 192.168.0.10 and the port is 22222 )

The driver should now be set for interfacing with Qualisys in ROS. To test it, first check that you are able to ping the Qualisys system over the MC lab WiFi.

```
$ ping 192.168.0.10
```

If you successfully pinged the qualisys system it should now be possible to listen to the data from the Qualisys system.

(Note that the Qualisys system need to recognize the IR-markers in the MC lab in order to transmit data. It may be smart to first to check that the computer running Qualisys software in the MC-Lab sees the marker)

```
$ roslaunch qualisys qualisys.launch
$ rostopic list
```

The command "rostopic list" prints the ROS active ROS topics. It should now be printed a qualisys topic in terminal. The name will depend on the name set on the Qualisys computer. In this manual the topic is named /qualisys/CSE1.

You can now listen to the data as it is published to ROS

```
$ rostopic echo /qualisys/CSE1
```

#### 1.2.2.2 Getting Qualisys data to MATLAB

The message sent from the Qualisys system is a custom message that MATLAB does not recognize (most messages in ROS is not custom, and will be recognized by MATLAB). In order to get the Qualisys data into MATLAB you one to facilitate so that MATLAB recognize the custom message.

Start by creating a new folder ~/qualisysDir. Now copy the folder named qualisys, located in ~/catkin_ws/src and paste it into the folder ~/qualisysDir

Now one want to edit the package file so that MATLAB recognizes the messages.

```
Sudo gedit ~/qualisysDir/qualisys/package.xml
```

Add the following two lines somewhere in the main body of the package.xml file.

```
<build_depend>geometry_msgs</build_depend>
<build_depend>std_msgs</build_depend>
```

Now open MATLAB. The first step in MATLAB is to download the ROS custom message package. Type the following lines into the MATLAB command window, and follow instructions to download the ROS custom message package.

```
roboticsAddons (in MATLAB 2016)
roboticsSupportPackages  (in MATLAB 2015)
```

When the download is finished paste the following commands in the MATLAB command window.

```
folderpath= '~/qualisysDir'
rosgenmsg(folderpath)
```

Now follow the instructions generated by MATLAB in order generate the needed message type. In this process you may need allow writing permission to the file "pathdef.m"

You are now ready to get the data into MATLAB.

Remember that the Qualisys node always need to be launched before reading signals in MATLAB.

```
roslaunch qualisys qualisys.launch
```

You can now get the data into Simulink by the Subscriber block, or to MATLAB workspace by typing the following commands:

```
Subb = rossubscriber('/qualisys/CSE1');
posedata = receive(Subb,10);
```

## 1.3 Towing carriage

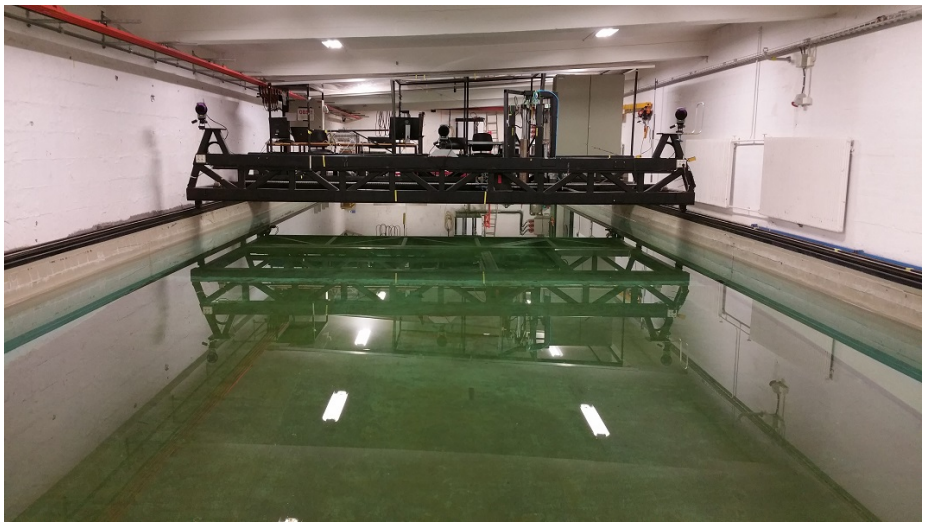

Figure 1.10: Towing carriage

The scope of this section is to explain how to safely operate the carriage without any damage towards humans or equipment.

### 1.3.1 Preparation before startup

To start with, you must make sure that any items mounted or fixed to the carriage are securely fitted, so they don't prevent the operation of the carriage. All personnel must stay on the operation platform during the travel of any axis.

Locate the Emergency Button and place it so that you can easily reach it from where you are sited. DO NOT USE THE EMERGENCY BUTTOM AS A BRAKE. YOU MUST ONLY OPERATE IT WHEN YOU ARE IN REAL EMERGENCY SITUATIONS.

#### Operation console

The operation console is an All-in-one PC. The Power button is on the bottom right side of the screen. If the operation panel is not on the desktop, you can start it by double clicking on desktop Icon, shown in Figure 1.11.

### 1.3.2 Manual Operation of the Carriage

#### Setup

It is very important to select the Setup tab first before you start any operation of the carriage. As you can see in Figure 1.14, it is possible to change the travel parameters for all available axes. In principle, all axis parameters have different range limits. These are listed in Table 1.1.

All axes can be activated or deactivate by using the ON button.

Figure 1.11: Towing console

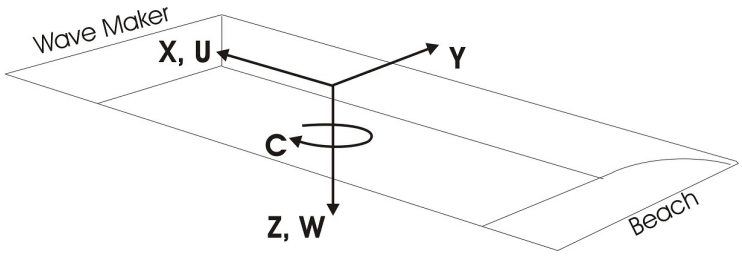


Figure 1.12: Coordinate system

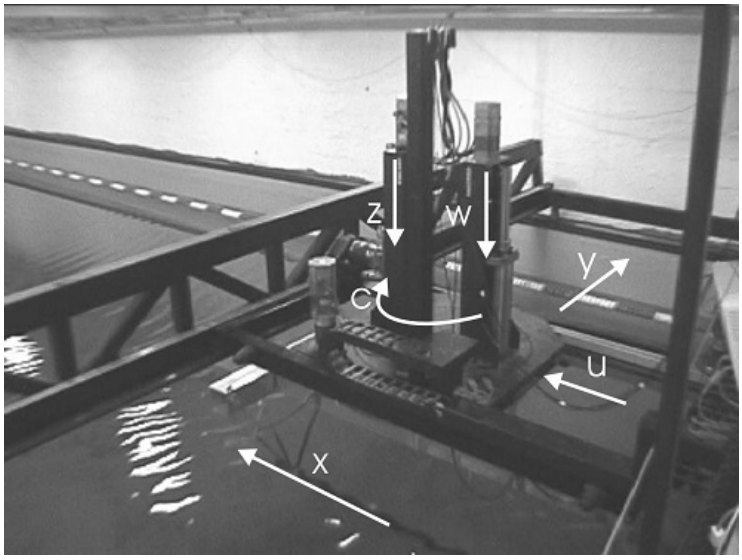


Figure 1.13: Coordinate system

For the X axis it is possible to activate a list of predefined Forward speeds. You will then have the ability to automatically change to a different speed on the next run. The list can be edited in the Main Tab Window.

| Axis | Forward/Backward Speed | | Acceleration Deceleration | | Position Pos./Neg. Limit | |
|---|---|---|---|---|---|---|
| X | 0 - 2.0 | [m/s] | % of 0 - 0.5 | $m/s^2$ | 0 - 22 | [m] |
| Y | 0 - 1.0 | [m/s] | % of 0 - 1.0 | $m/s^2$ | 0 - 4.5 | [m] |
| U | 0 - 1.0 | [m/s] | % of 0 - 1.0 | $m/s^2$ | 0 - 1 | [m] |
| C | 0 - 10 | [deg/s] | % of 0 - 20 | $deg/s^2$ | 0 - 255 | [deg] |
| Z | 0 - 1.0 | [m/s] | % of 0 - 2.0 | $m/s^2$ | 0 - 0.5 | [m] |
| W | 0 - 1.0 | [m/s] | % of 0 - 2.0 | $m/s^2$ | 0 - 0.5 | [m] |

Table 1.1: Operation Limit

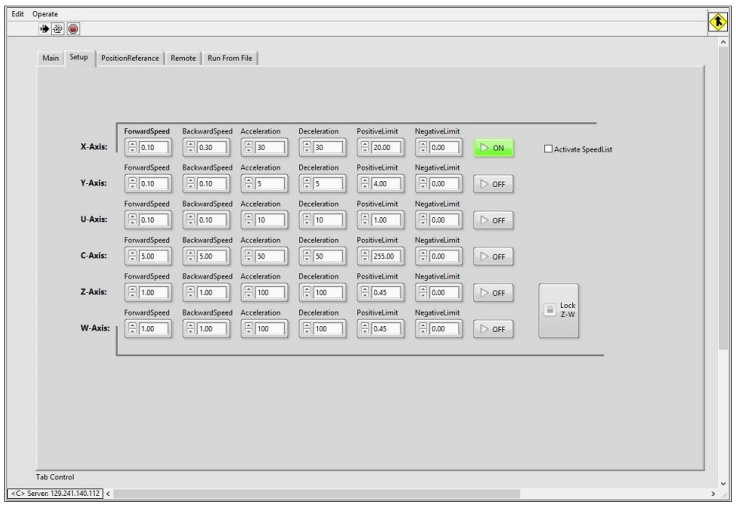


Figure 1.14: Setup

By selecting the "Lock Z-W" button the Z-W axis will operate in parallel. They will use the Z-axis parameter setup.

### Main/Standard Operation

All the activated axes will operate within the limits set in the Setup. Only one axis can operate at a time. If you hit the button for another axis than the running one, it will instantly stop and new one will start running. To stop the running axis, simply hit the stop button. If no buttons are operated carriage axis will run it hit limit position of the current axis.

If an error occurs, for some reason, it can be cleared by hitting the "Power" button. If the error keeps reoccurring, please look at the Troubleshooting section of this document or contact responsible MC Lab personnel.

The current speed and position of the active axis are displayed referred to the selected limits.

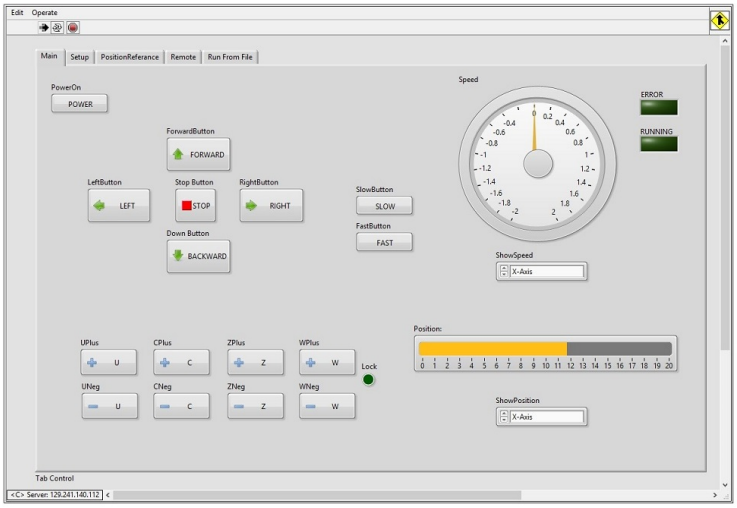


Figure 1.15: Main

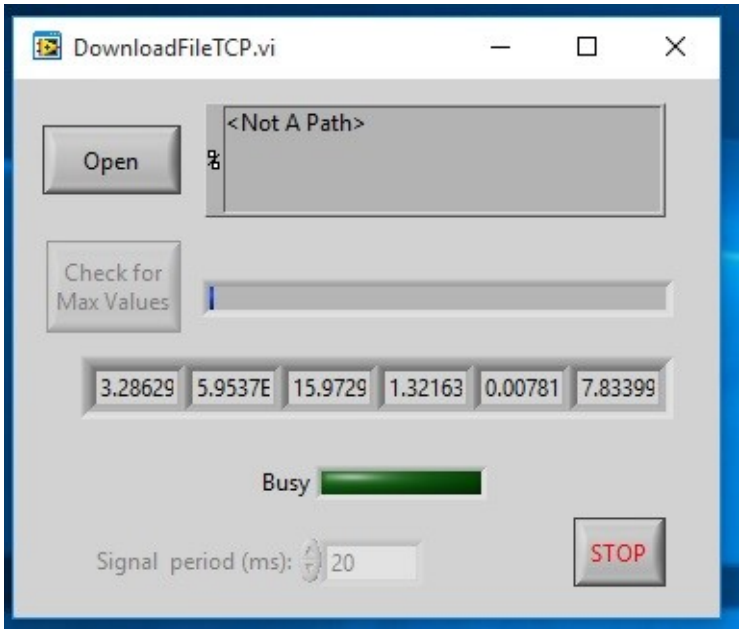


Figure 1.16: Run From File

### 1.3.3 Operation Controlled automatically from PC

#### The Trajectory Input File

All trajectories must be defined in a .mcl input file. The format of the file is slightly more general than allowed here and is the same as for the sloshing rig input. The entries in the file are

1. Time Step in ms, double precision integer (int32). Must be set to 10.

2. Number of channels, double precision integer (int32). Must be 6.

3. Position references in sequence: X(1),Y(1),U(1),C(1),Z(1),W(1), X(2),Y(2), double precision real (float32).

The following MTALAB lines write the matrix body (6xN) to file on the correct format:

```
fid=fopen(filename,'wb');
head=[10;6];
count=fwrite(fid,head,'int32');
count=count+fwrite(fid,body,'float32');
fclose(fid);
```

The resulting input file must be transferred to the realtime computer at /home/ntuser/inputpos.mcl. Normally this is done automatically when the Load button on the LabVIEW GUI is pressed.

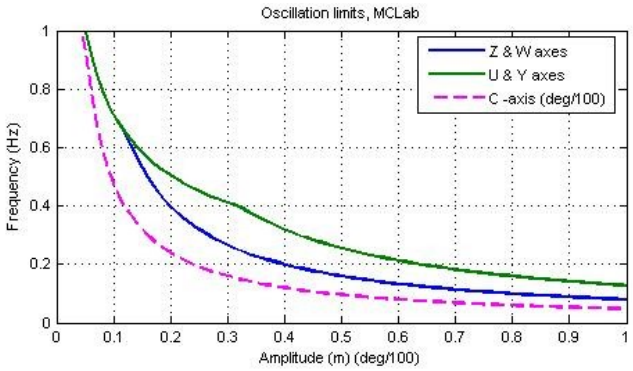


Figure 1.17: Operation Limit Amplitude vs. Frequency

### 1.3.4 Troubleshooting

### 1.3.5 Note

When the wagon is moving, no one is allowed to move on the sides of the basin.

## 1.4 Wave Generator

The wave maker is a single paddle wave making machine with a width of 6 meter and it is equipped with an Active Wave Absorption Control System (AWACS 2). The single paddle wave generator is controlled by a dedicated computer. The machine can produce both regular and irregular waves because of the DHI Wave Synthesizer the system has. Available spectrum are first order Stoke, JONSWAP, Pierson-Moskowitz, Bretschneider, ISSC and ITTC. Table 1.2 summarizes the generation capacity.

| | Height [m] | Period T [s] |
|---|---|---|
| Regular waves | $H < 0.25$ | 0.3 - 3.0 |
| Irregular waves | $H_s < 0.15$ | 0.6 - 1.5 |

Table 1.2: Wave generator capacity

### 1.4.1 User Manual

Add a description on how to use the system once the NEW wave generator is installed(fall 2017)

## 1.5 Video-Camera System

The laboratory is equipped with 2 high-resolution cameras for recording of activity in the basin, one on each side. The cameras are remotely operated from a dedicated PC in the command center using the *Intelligent Video Management System 4200 (iVMS-4200 client)* software. The computer is located on the floor, is connected to the TV-monitor mounted on the wall, and has wireless mouse and keyboard(labeled *Camera system*). The software user-interface is illustrated in Figure 1.18.

The manual control of the cameras are found in the *PTZ Control* tab. The cameras feature auto-tracking of objects, which is enabled by right-clicking in the video window. The menu is illustrated in Figure 1.18, with highlighted options for auto-tracking and manual control of the camera position. The camera system also support recording. Note that the recorded files are large, typically several GB for some minutes of recorded video. Hence, the files should be *moved* to a USB-disk(and not copied). The recorded files are found in the path: `C:\ivms4200\video\RecordFile\YYYYMMDD`.

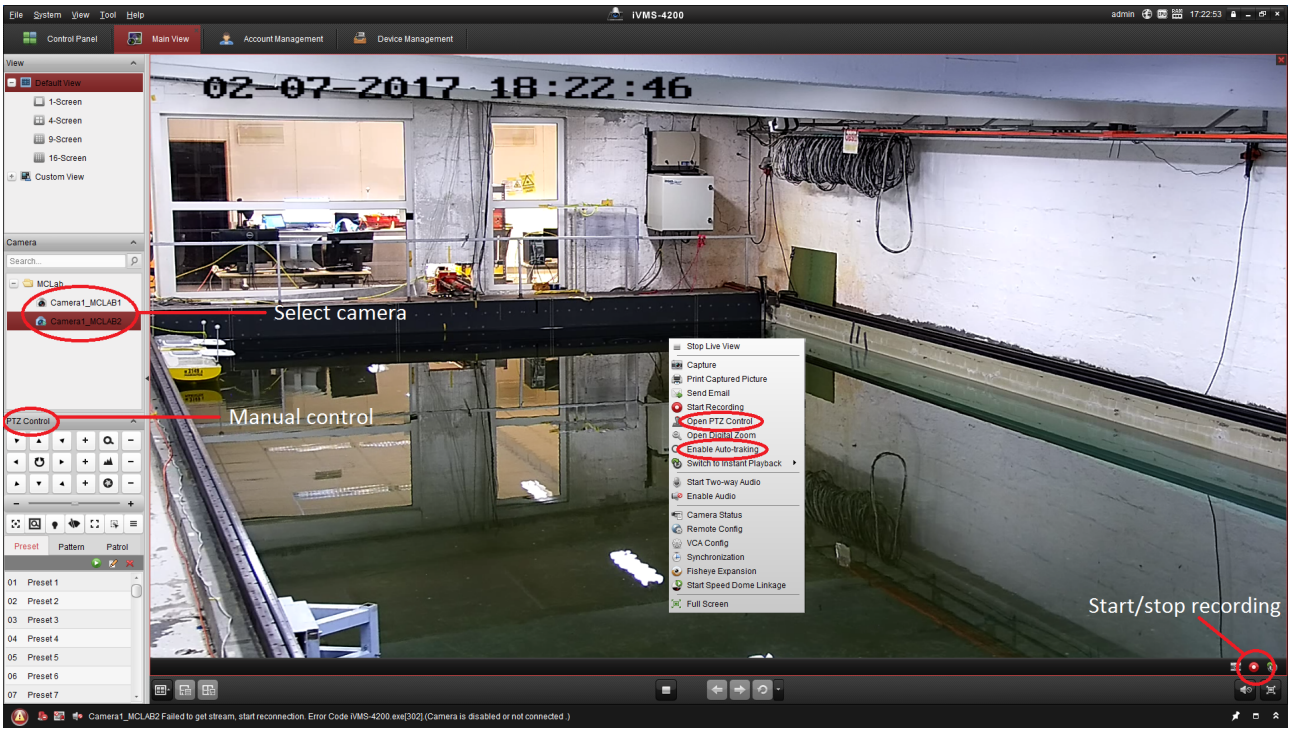


Figure 1.18: User interface iVMS-4200 camera system

# Chapter 2

# Software

## 2.1 Laptop

As of July 2017, there is a Virtual Machine with all necessary software installed. To use cRIO, the following software is needed:

- LabVIEW Full Development system, with the following modules:
  - LabVIEW Real-Time Module
  - LabVIEW FPGA Module
  - NI-RIO driver
- MATLAB and Simulink
- VeriStand with all modules included in the installation file(especially NI VeriStand Model Framework)
- WindRiver GNU Toolchain that supports VxWorks.

Version compatibility is an important issue, see `http://digital.ni.com/public.nsf/allkb/10BBA745CD5A2FAE86257F9A0054FF71`. The Virtual Machine has LabVIEW 2017, VeriStand 2017 and Matlab 2016b installed. For use of CS Enterpris 1 or CS Arctic Drill Ship, it is highly recommended to use the Virtual Machine in stead of installing all the software. To use the Virtual Machine, install VirtualBox and copy the Virtual Machine from one of the LAB-computers, or contact Hans-Martin Heyn for a copy.

However, if installing the software, it is recommended to install the different software parts in the order given above. The installation files are found on the webpage of National Instruments, and the product key is found on `software.ntnu.no`.

## 2.2 cRIO

Updating and changing software on cRIO is done in NI MAX. In order to update the cRIO(e.g. updating to a newer version), follow this procedure:

1. Make sure the cRIO and the laptop are on the same network(either both connected to MCLab or by an Ethernet cable)

2. Open NI MAX. In the left pane, navigate to the cRIO. Press restart, and verify that the correct cRIO is restarting.

3. Right-click on the cRIO → Format Disk.

4. Select "Attempt to restart into safe mode" and "Preserve the setting for all network adapters", as illustrated in Figure 2.1.

5. When successfully formated, install software. Expand the cRIO, right-click on Software and select Add/Remove Software. See Figure 2.2

6. Select NI CompactRIO *Version* → Next → Next → Next. There is no need to alter the software here, NI MAX installs the recommended software for the cRIO. Press Finish

7. Right-click on Software once more, and select Add/Remove Software. Now, select Custom software installation. Choose Yes in the warning. Navigate to NI VeriStand Engine YYYY, and choose Install the feature. Press Next → Next → Finish. See Figure 2.3

8. The cRIO has now been updated with the necessary software with the version selected.

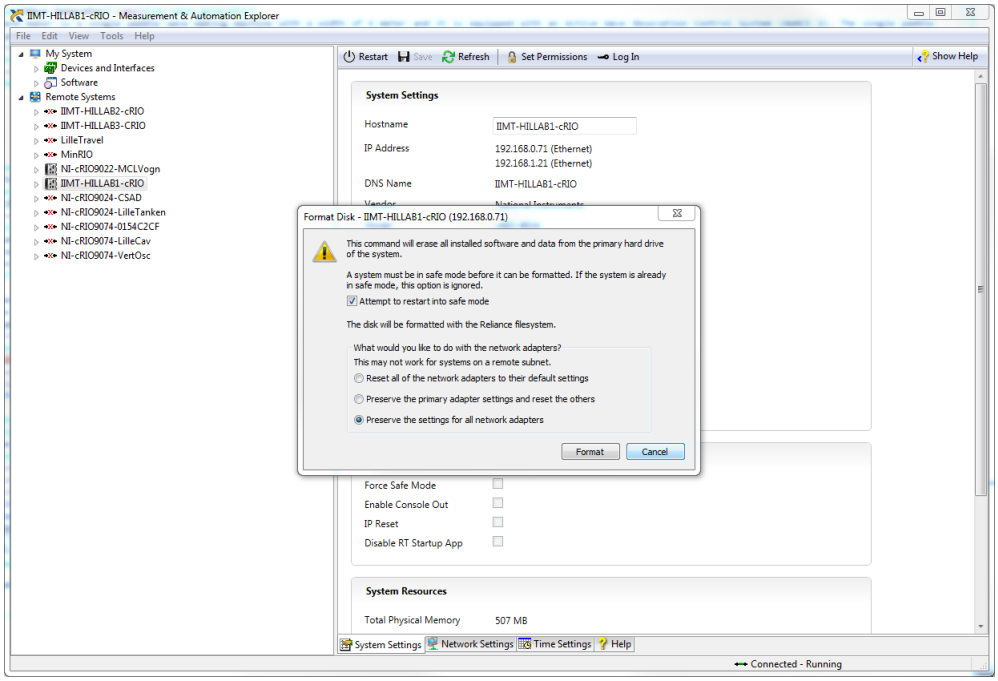


Figure 2.1: Format cRIO

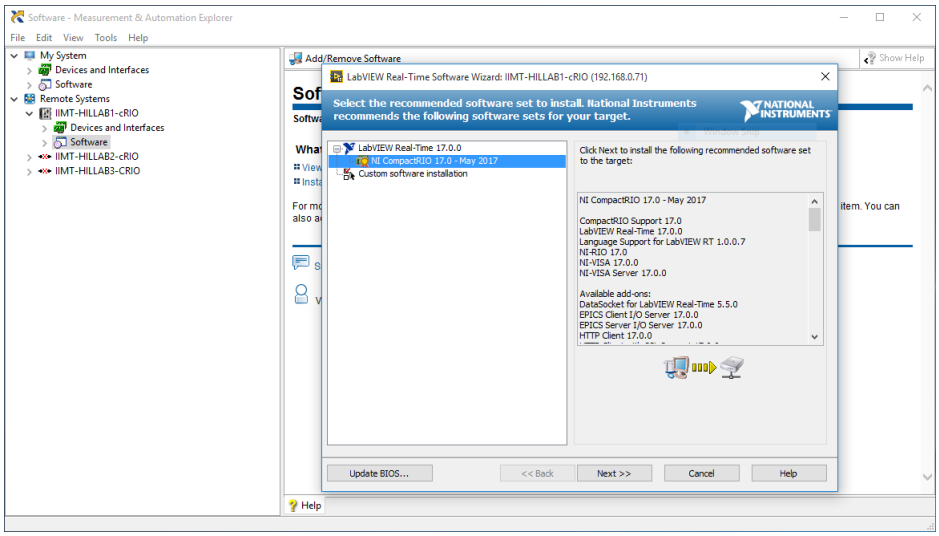


Figure 2.2: Install software on cRIO

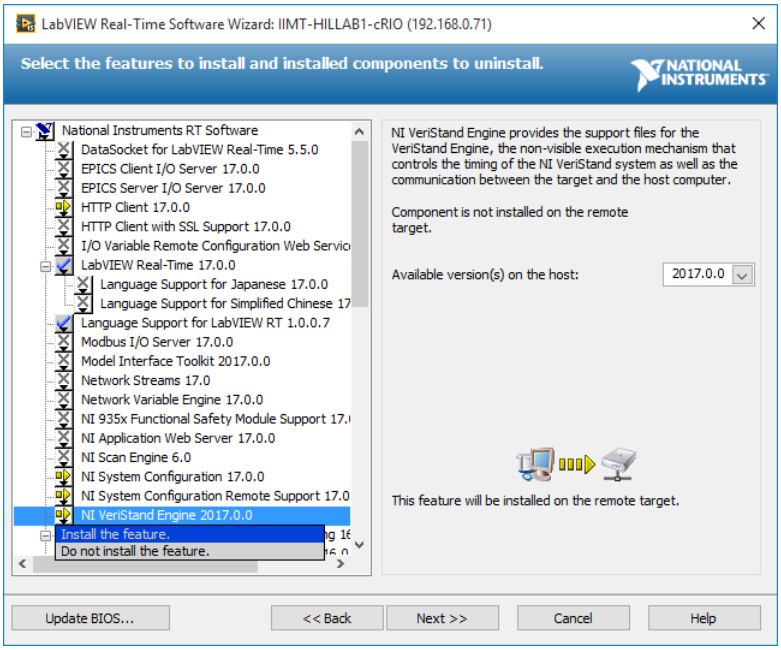


Figure 2.3: Install NI VeriStand Engine on cRIO

## 2.3 Update customized simulink code

This section describes the process of updating the customized Simulink model, and building it as a cRIO-compatible code. For illustration, the CSE1 is used here, but the process is similar for all cRIOs. First, go to GitHub and download/clone the repository related to the vehicle of interest. Then, open *ctrl_custom.slx*. The model should be similar to the one shown in Figure 2.4.

You should not alter the input/output subsystems, as these are already mapped

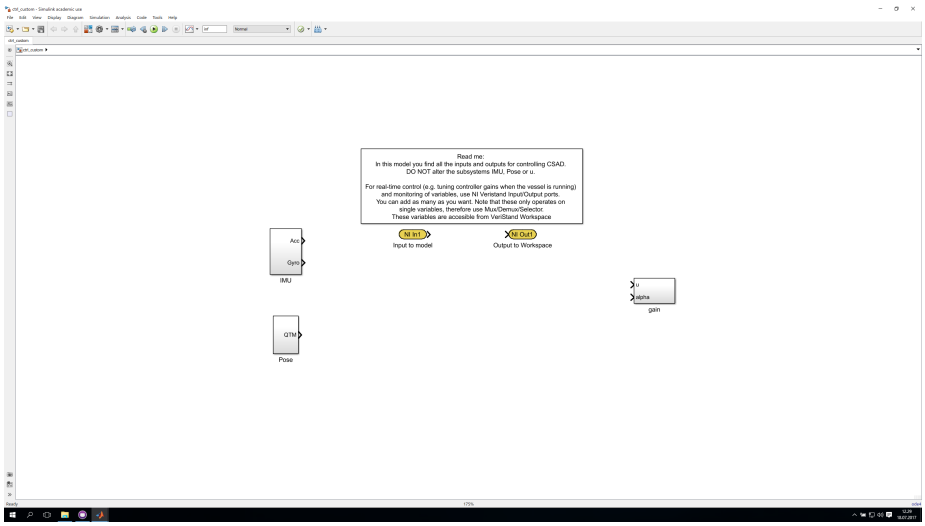


Figure 2.4: ctrl_custom.slx initial window

properly in VeriStand. Implement your control system as desired, and include as many NI VeriStand Input/Output blocks as desired. These blocks are used to read/change parameters when the code is running on the cRIO. When the *ctrl_custom.slx* Simulink project is updated with the desired control system, the code must be compiled to C-language for exporting to the cRIO. First, make sure the active folder directory in MATLAB is `CSE1\simulinksystem\`. In Simulink, open the Model Configuration (press  ), and make sure the following settings are applied(see Figure 2.5):

**Solver:** Stop time: **inf**, Solver type: **Fixed-step**, Solver: **discrete** or **ode3**, Fixed-step size: **0.01**

**Code Generation:** System target file: **NIVeriStand_VxWorks.tlc**. If another file is shown, press Browse and find the correct one.

**Code Generation/NI Configuration:** WindRiver GNU Path: `C:\gccdist\supp\setup-gcc.bat`. If `supp` does not work, change it to `supplemental`.

You are now ready to compile the code and include it in the VeriStand project:

1. Compile the Simulink model by pressing `Ctrl+B` or  in the Simulink window. MATLAB is now compiling your code, and it will update the folder `CSE1\Simulinksystem\ctrl_custom_niVeriStand_VxWorks_rtw` with a cRIO compatible simulation model.

2. Open the VeriStand project (*CSE1.nivsproj*), and then open the System Explorer as illustrated in Figure **??**.

3. Navigate to the `ctrl_custom` Simulation Model, verify the modification date/time is correct (time stamp from when you compiled the simulink model) and then press Reload as illustrated in Figure 2.6.

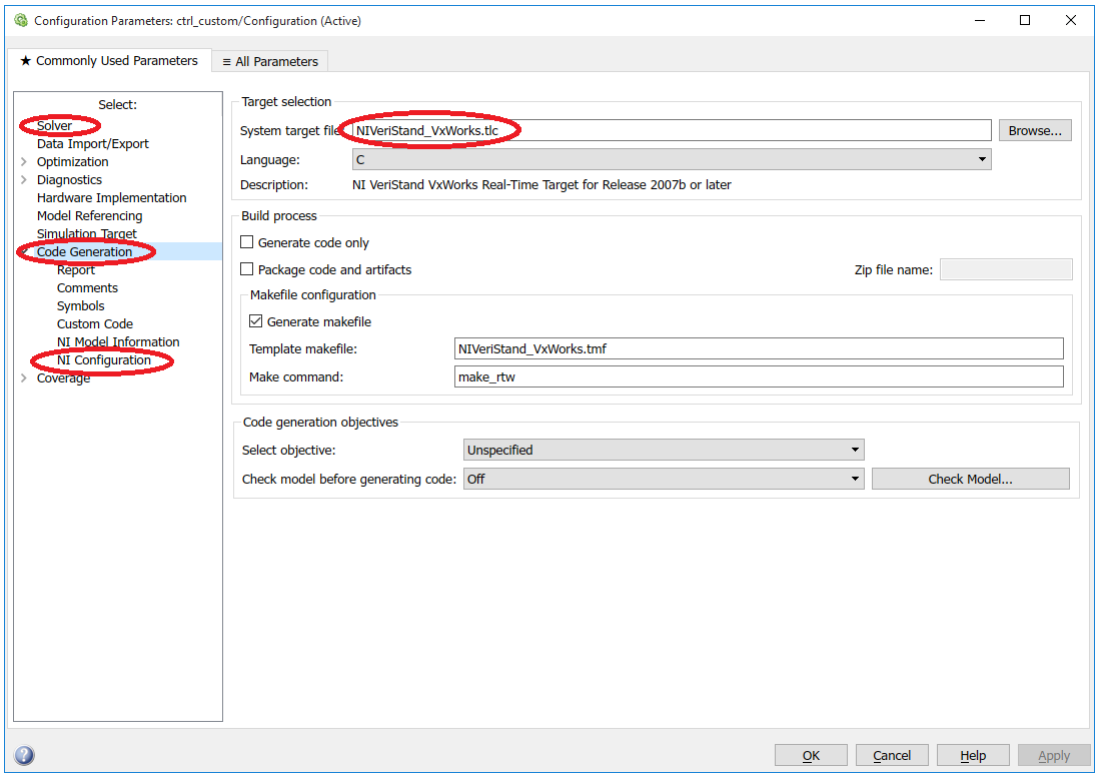


Figure 2.5: Model Configuration window

4. Save and close the System Explorer window.

Your Simulink model is now updated and included in the VeriStand project. Continue with preparing the vessel and uploading the code to the cRIO, as described in the vessel specific Handbook.

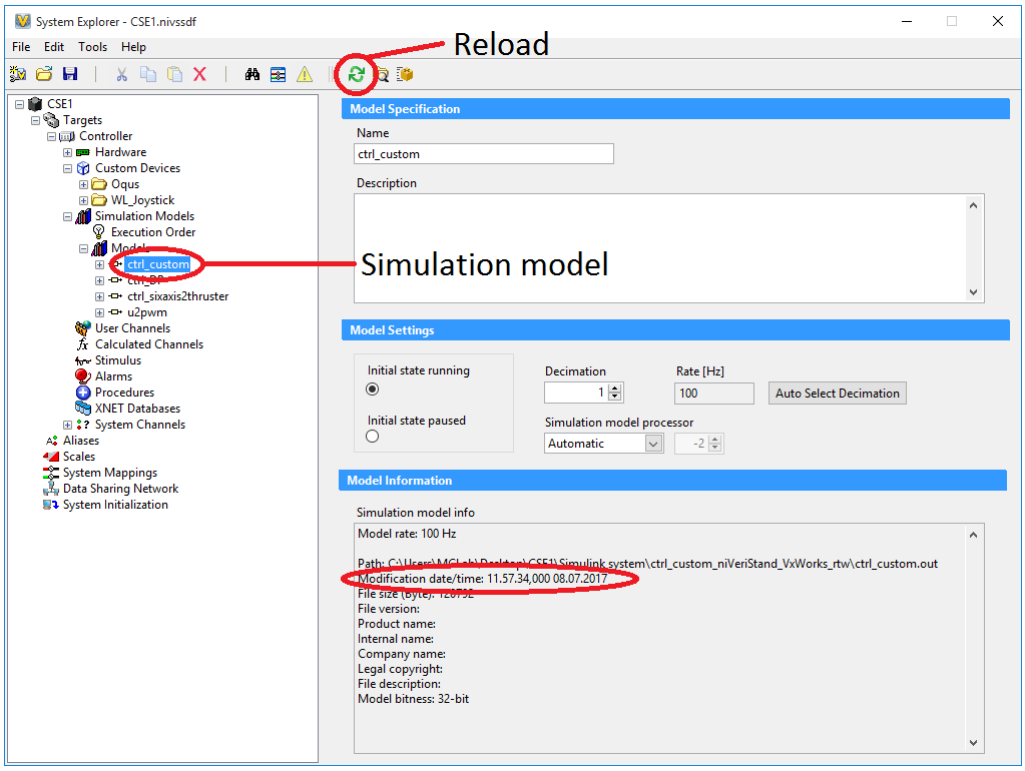


Figure 2.6: System Explorer window

## 2.4 Data logging on cRIO

Logging of data can be done in 2 ways:

### 2.4.1 In Workspace

Logging channels/parameters from Workspace is done with a Data Logging Controller, found in Workspace Controls. This is the preferred method, as you set the start and stop of data logging and avoid problems if the code does not deploy correctly. The data log is also saved on the laptop, not on the cRIO. Add a Data Logging Control in your Workspace window, set the desired path for the file and add the channels/parameters of interest. See Figure 2.7.

### 2.4.2 In Simulink

In the Simulink model, it is possible to add "Write to File" blocks linked to the different parameters. By using this method, the cRIO logs the parameters continuously after deployment, until the VeriStand project is undeployed. The data is logged as binary numbers, and if the code is not undeployed correctly(e.g. some error/loss of power etc.), the data log becomes corrupted. However, if this method still is chosen, the data files must be copied from the cRIO to the laptop. Open NI MAX, in the left pane, browse to CSE1, right click on it and choose

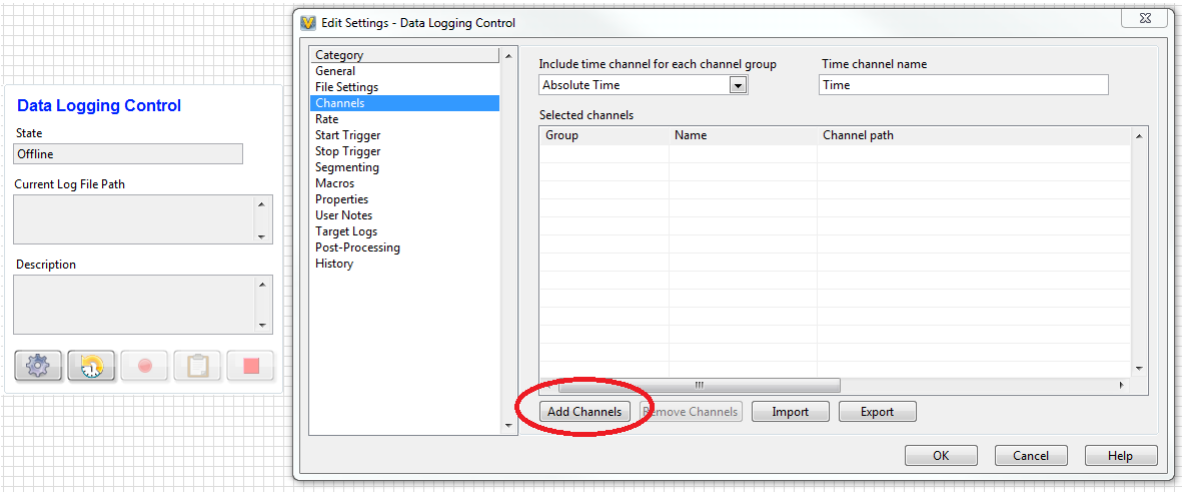


Figure 2.7: Data logging in Workspace

File Transfer. Copy the logged files to the laptop, which can then be loaded in MATLAB.

# Bibliography

KumarRobotics, 2016. Kumarrobotics/qualisys source code. Retrieved 10th of March 2016. https://github.com/KumarRobotics/qualisys. 1.2.2.1

MathWorks, 2016. Create custom messages from ros package. Retrieved 10th of March 2016. http://se.mathworks.com/help/robotics/ug/create-custom-messages-from-ros-package.html. 1.2.2

# Appendix A

# Advanced software topics

## A.1 Creating FPGA and XML files

### A.1.1 Create FPGA target and XML

If you do not have a Veristand FPGA target at your disposal, follow the steps below. If you have a target available and just need to install it in NI Veristand, please jump to Section A.1.2. For CS Enterprise 1 and CS Arctic Drill Ship, the FPGA targets are found on GitHub.

1. Open LabVIEW and create new project. In this guide, LabVIEW 2013 is used, but the procedure should be similar on newer versions.
2. Choose NI Veristand FPGA Project in project templates and proceed.
3. Choose CompactRIO Reconfiguarble Embedded System and click next.
4. You will now get the choice between letting LabVIEW detect your cRIO system or configure it yourself. If you are connected to the cRIO and it has all of the I/O ports connected, the option "Discover existing system" is simpler and therefore recommneded. If you do not have your cRIO connected choose "Create new system", this is the version that will be worked through here.
5. Select your controller, in our case cRIO-9024.
6. Select your FPGA target, in our case cRIO-9113.
7. Then you select your I/O modules to the correct slots. In our case NI 9215 in slot 1 and NI 9474 in slot 4.
8. You are now finished with configuring your project. Press next.
9. The project menu will now appear and should look something like Figure A.10. Select the LabVIEW VI as demonstrated our is called Custom Personality FPGA.vi
10. The UI window will now present itself, select window and show block diagram.

11. You should now see a block diagram similar to Figure A.11. You will now have to redesign this to look like Figure A.12. This will be valid for our system, if you have different I/O modules the block diagram need to reflect this.

12. Now, return to the Project explorer and select Build Specifications and Custom Personality FPGA

13. A new window will open. Check that the name and project path is correct and press build.

14. Select your preferred compile server. The compilation process will take quite some time (approx 15-30 min).

15. When the compilation process is finished, the last step is to edit the automatically generated XML file. You will now have to find you project directory in Windows. Here there will be a folder called bitfiles which contains the files you compiled in the last step, there will also be a .XML file. The point of editing this file is to match the actually compiled VI, meaning the packets must match the connected I/O. The recommended way to edit the file is to copy our XML file from: Dropbox\TMR4243 - LAB\04 cRIO software\FPGA IO. You will have to make sure that the name of your bitfile matches the name in the XML file as seen in Figure A.17, also make sure the I/O modules matches your setup.

16. Copy the bitfiles from the bitfile folder to the level above so that the bitfile aand the XML file is in the same folder.

Documentation: `https://decibel.ni.com/content/docs/DOC-13815`

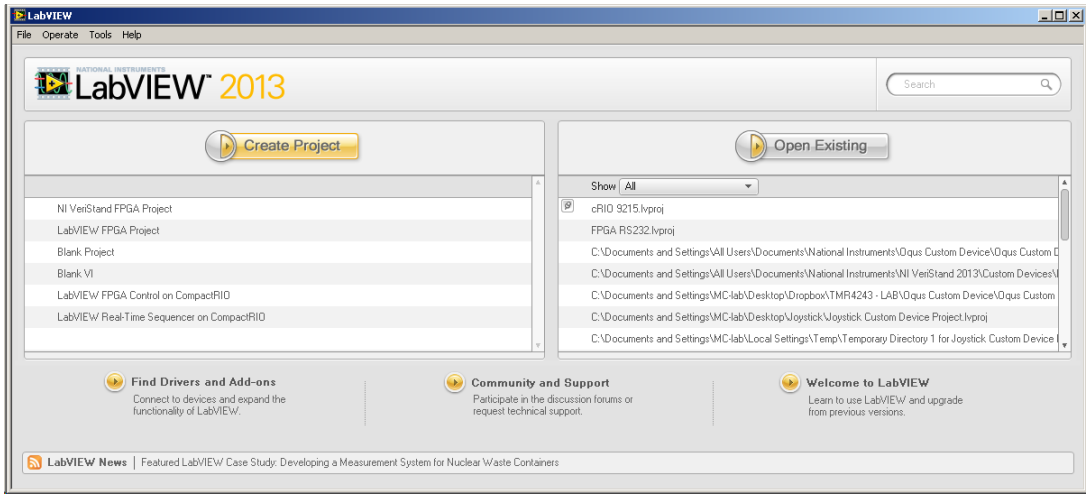


Figure A.1: Create Labview FPGA target and XML - 1

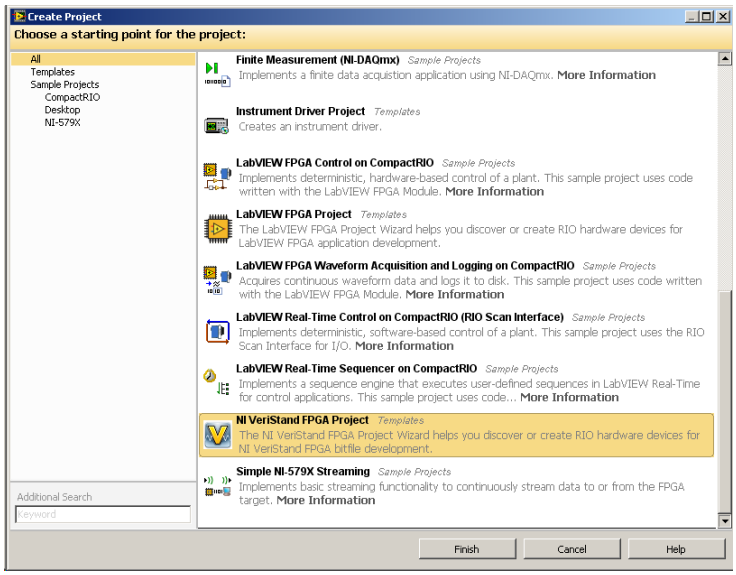


Figure A.2: Create Labview FPGA target and XML - 2

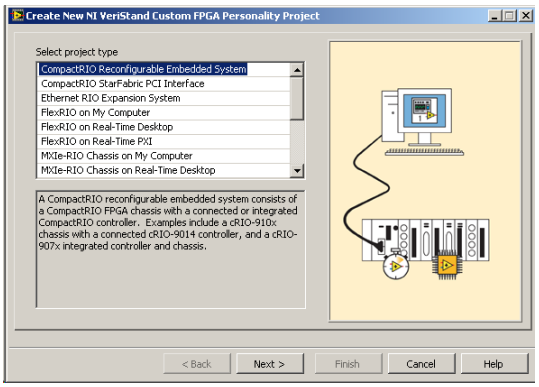


Figure A.3: Create Labview FPGA target and XML - 3

### A.1.2 Install in VeriStand

The Veristand software does not recognize the physical I/O components of the cRIO. It is necessarry to write a specific FPGA mapping for the specific setup. This results in a XML file that maps the ports.

To add this file to your Veristand project, enter the system explorer and find the FPGA pane under *targets\controller\hardware\chassis*, as seen in Figure A.18. The next step is to find your XML file. In this case called cRIO-9113 Ex, it is very important that the XML file is placed on level above the FPGA bitfile folder in the directory system, as the files are really being used are the FPGA bitfiles. The menu in should now look something like Figure A.19, here you can see the analogue input signals and the digital output PWM signals. These can again be linked to other signals as seen in FigureA.25.

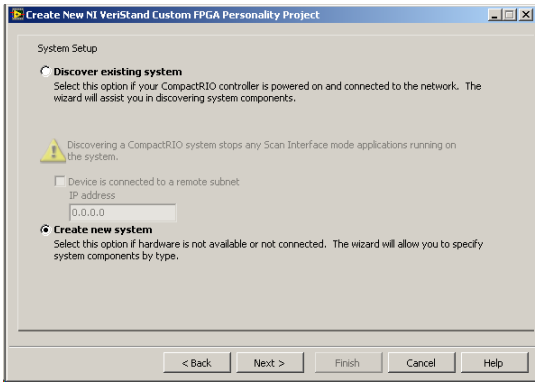


Figure A.4: Create Labview FPGA target and XML -4

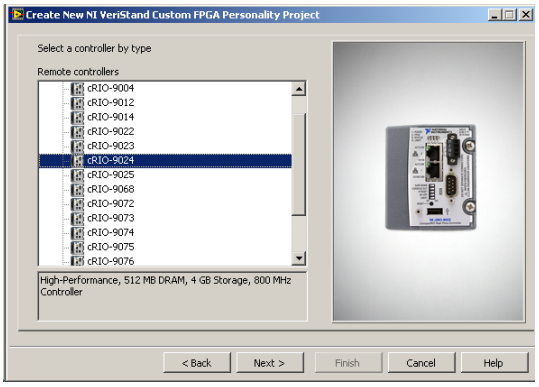


Figure A.5: Create Labview FPGA target and XML - 5

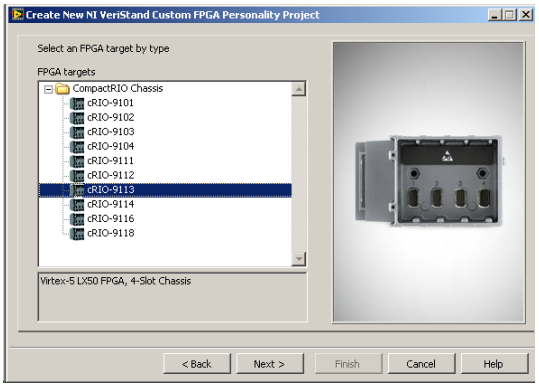


Figure A.6: Create Labview FPGA target and XML - 6

#### A.1.2.1 Ticks

tick = FPGA clock pulse

$$\text{tick in seconds} = \frac{1}{\text{frequency}} = \frac{1}{40MHz} = \frac{1}{40 * 10^6} = 25 * 10^{-}9 = 25ns$$

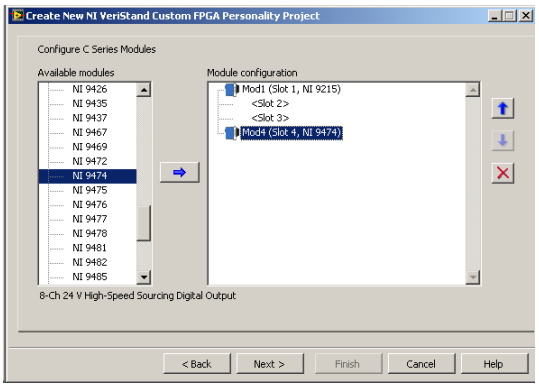


Figure A.7: Create Labview FPGA target and XML - 7

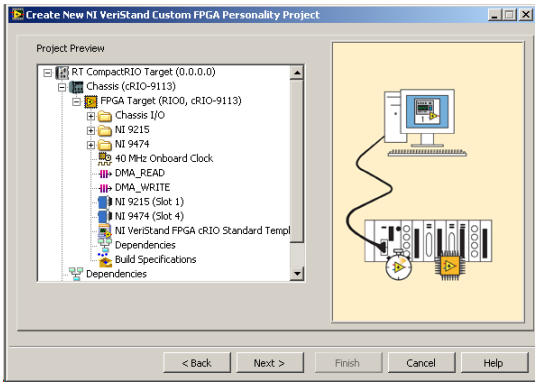


Figure A.8: Create Labview FPGA target and XML - 8

output at 50 Hz demands output every

$$\frac{40MHz}{50Hz} = \frac{40 * 10^6}{50} = 800000tick$$

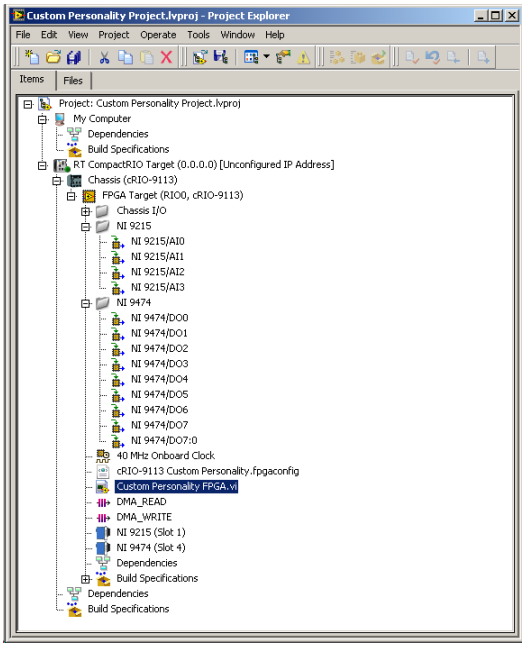


Figure A.9: Create Labview FPGA target and XML -9

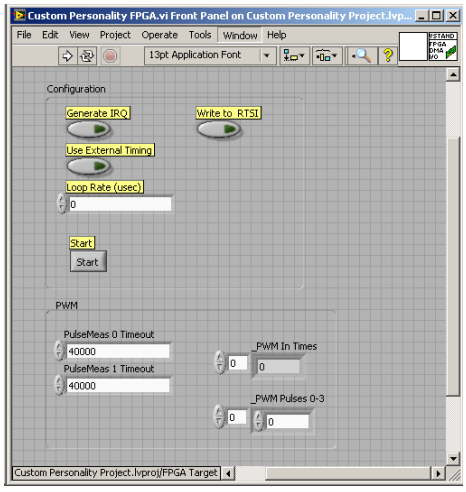


Figure A.10: Create Labview FPGA target and XML - 10

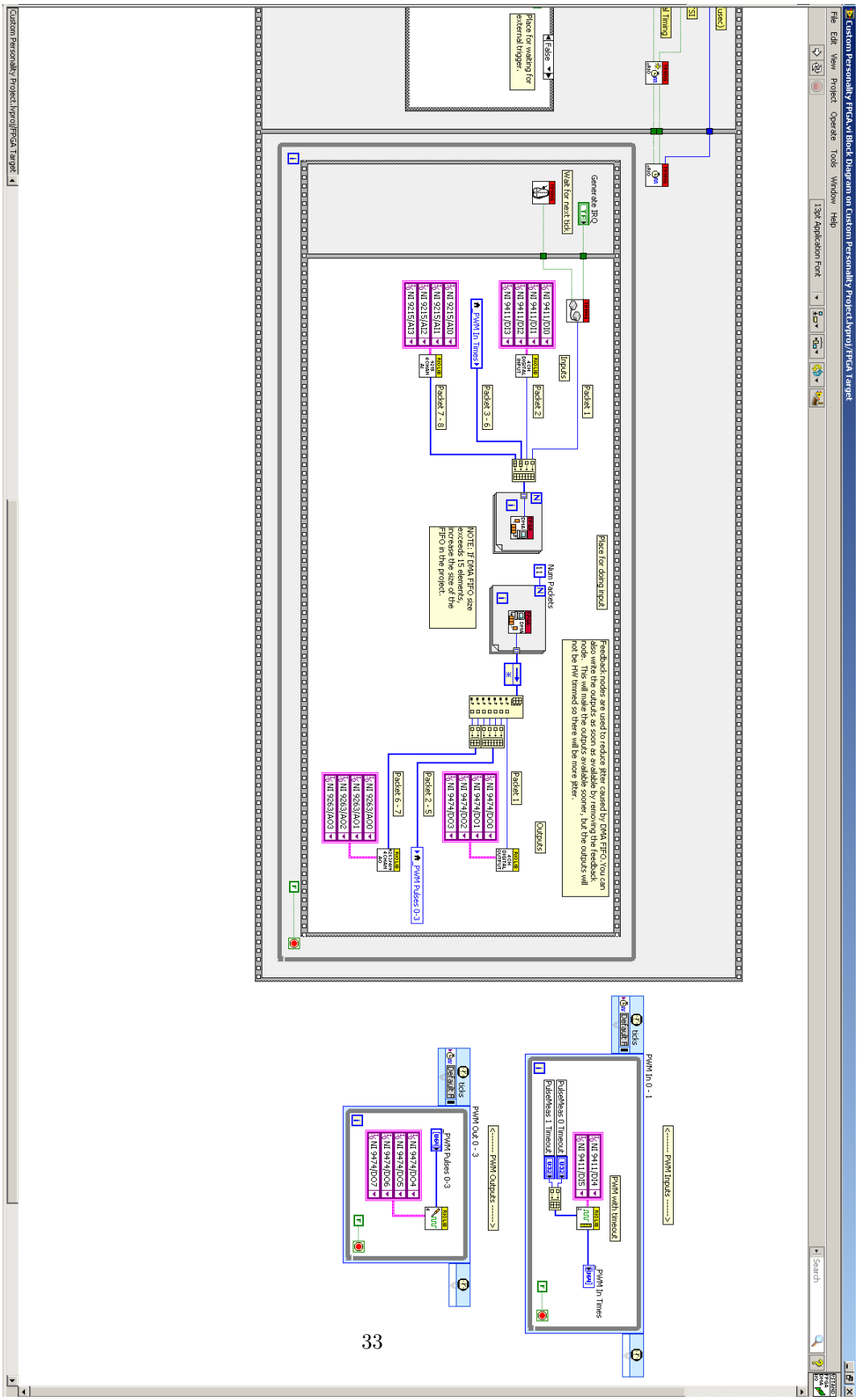




Figure A.11: Create Labview FPGA target and XML - 11

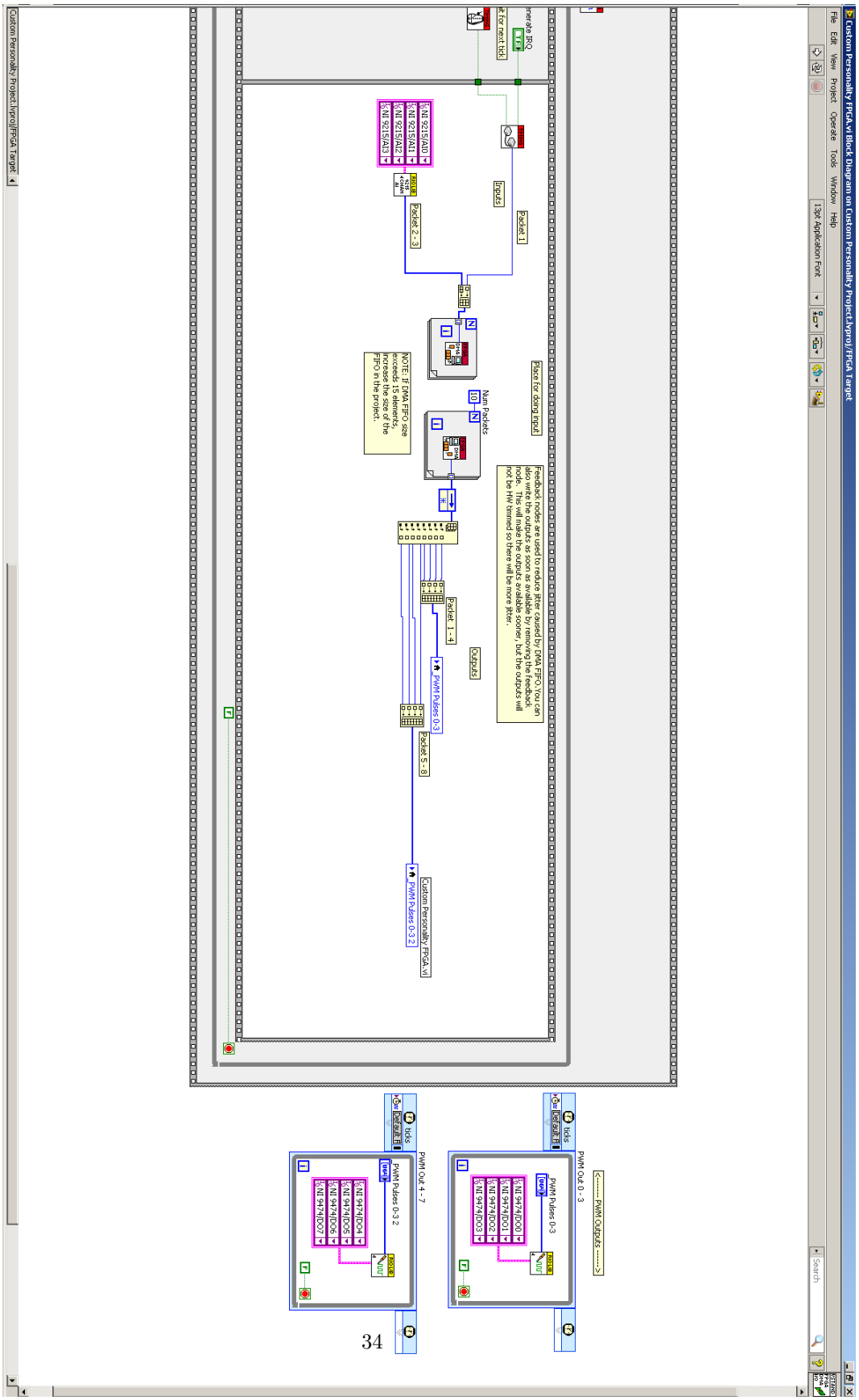




Figure A.12: Create Labview FPGA target and XML - 12

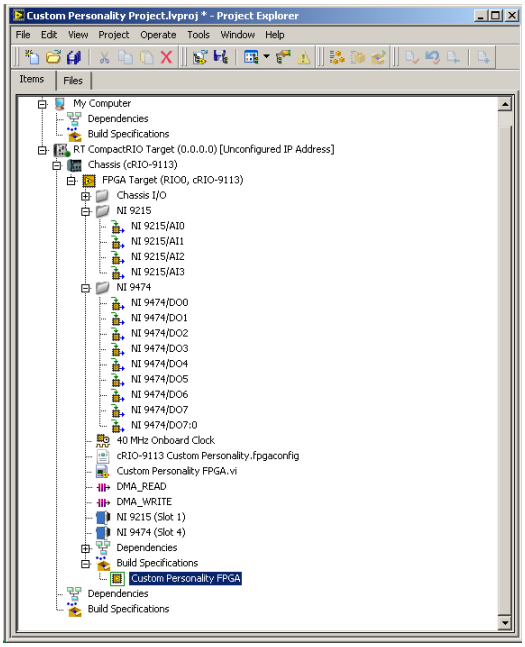


Figure A.13: Create Labview FPGA target and XML - 13

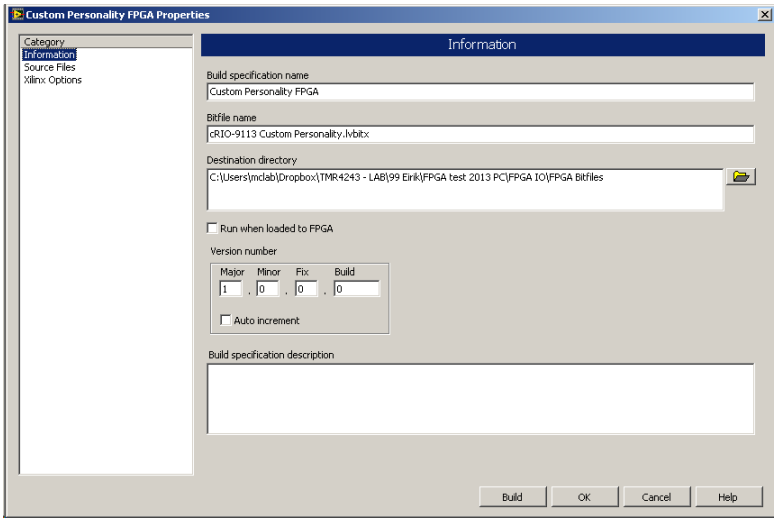


Figure A.14: Create Labview FPGA target and XML - 14

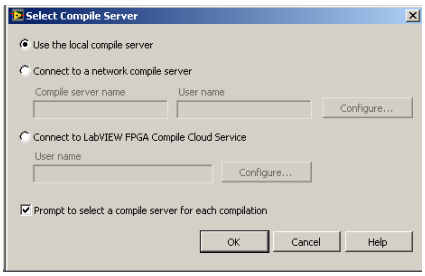


Figure A.15: Create Labview FPGA target and XML - 15

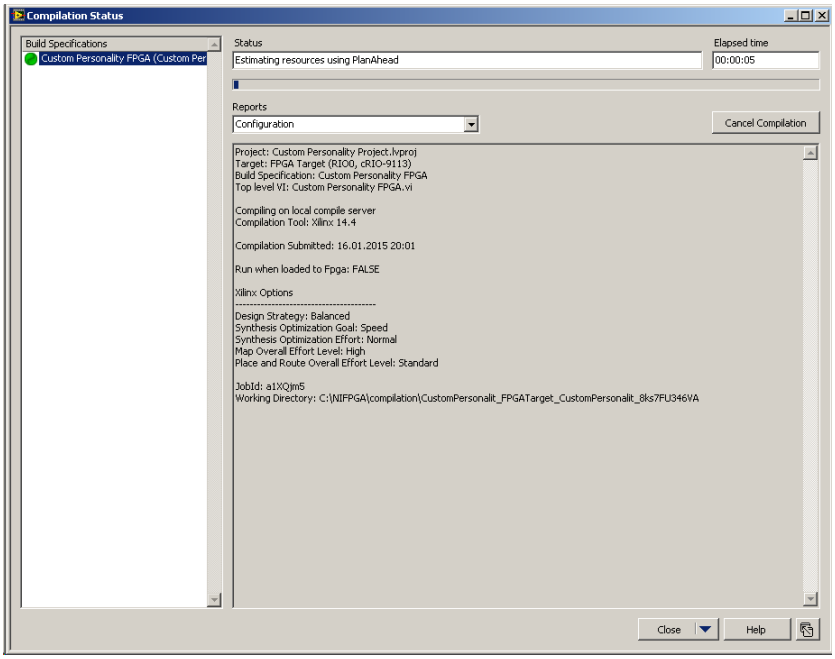


Figure A.16: Create Labview FPGA target and XML - 16

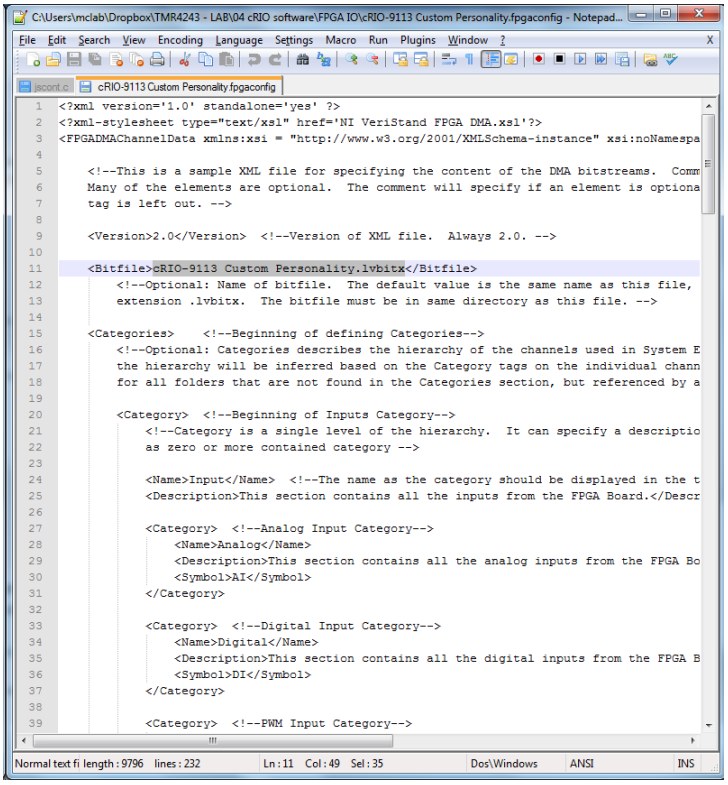


Figure A.17: Create Labview FPGA target and XML - 17

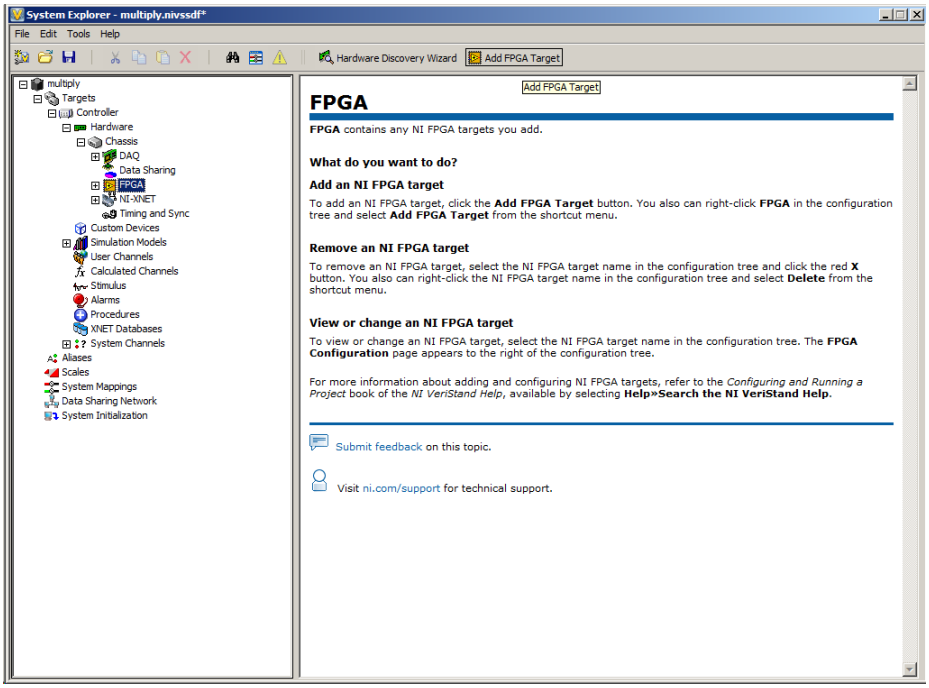


Figure A.18: FPGA1

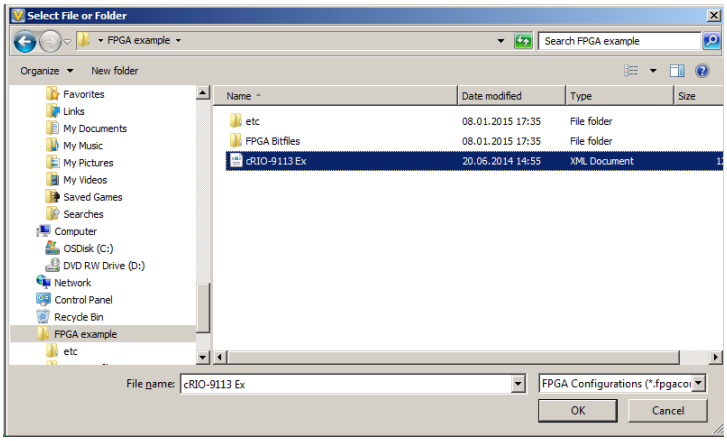


Figure A.19: FPGA2

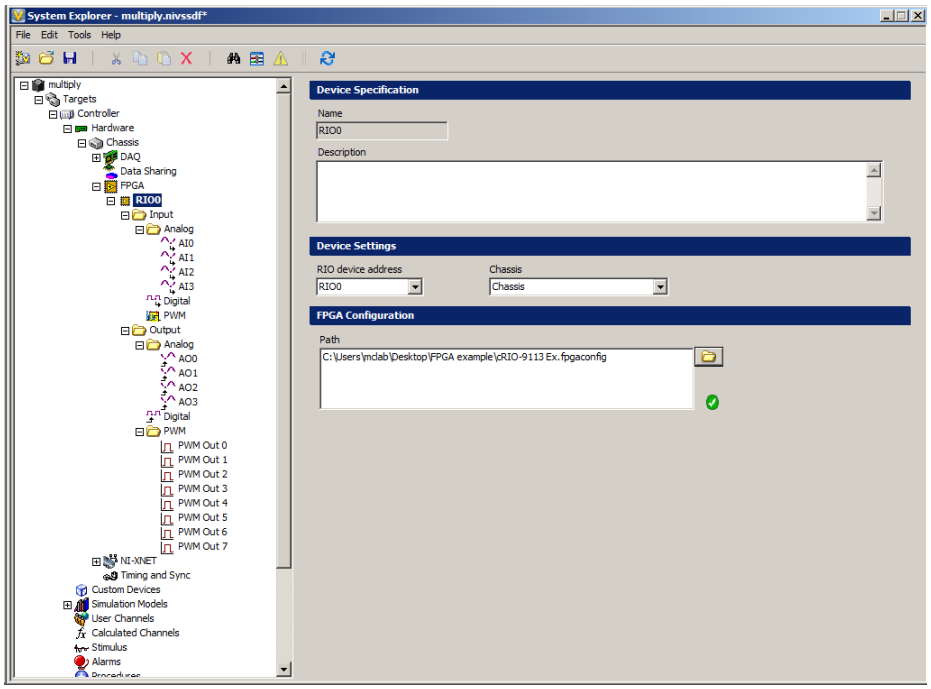


Figure A.20: FPGA3

## A.2 Custom Device

### A.2.1 Install

As of July 2017, there are 3 different Custom Device drivers developed for use in MCLab:

**WL_Joystick** - used for reading sixaxis data sent from the RPi. Created by Torgeir Wahl

**Oqus** - used for reading position and orientation data sent from the Qualisys system. Created by Torgeir Wahl

**IMU** - used for reading IMU data from 4 IMU's, mainly intended for CSAD. Created by Guttorm Udjus

To install a Custom Device driver, the first step is to copy the folder with the driver to the path: `C:\Users\Public\Documents\NationalInstruments\NIVeriStand201X\CustomDevices` The directory should now contain something like Figure A.21. The next step is to add custom device to your project. This

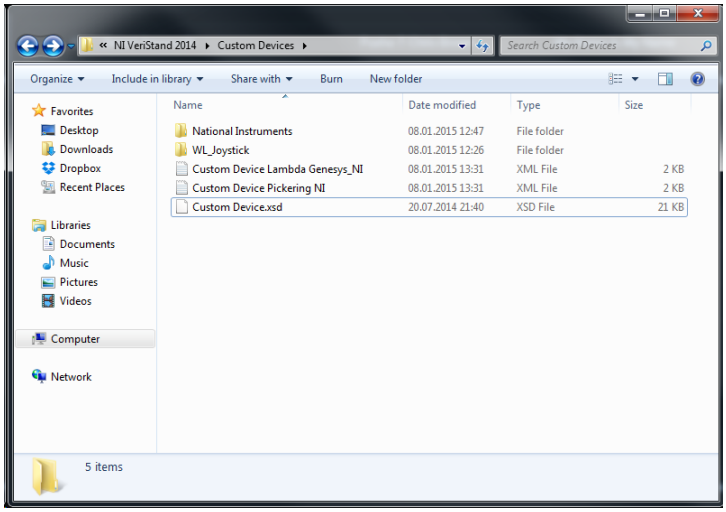


Figure A.21: Custom device folder

is done in the system explorer, which is found as seen in Figure A.22. When

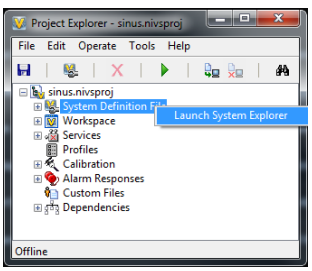


Figure A.22: VeriStand launch system explorer

in the system explorer, adding the custom device should be as simple as right clicking the custom device pane and choosing WL_Joystick, as in Figure A.23. If you do not find the custom device WL_Joystick, the most likely problem is that the placement of the custom device folder from step 1 is wrong. If the

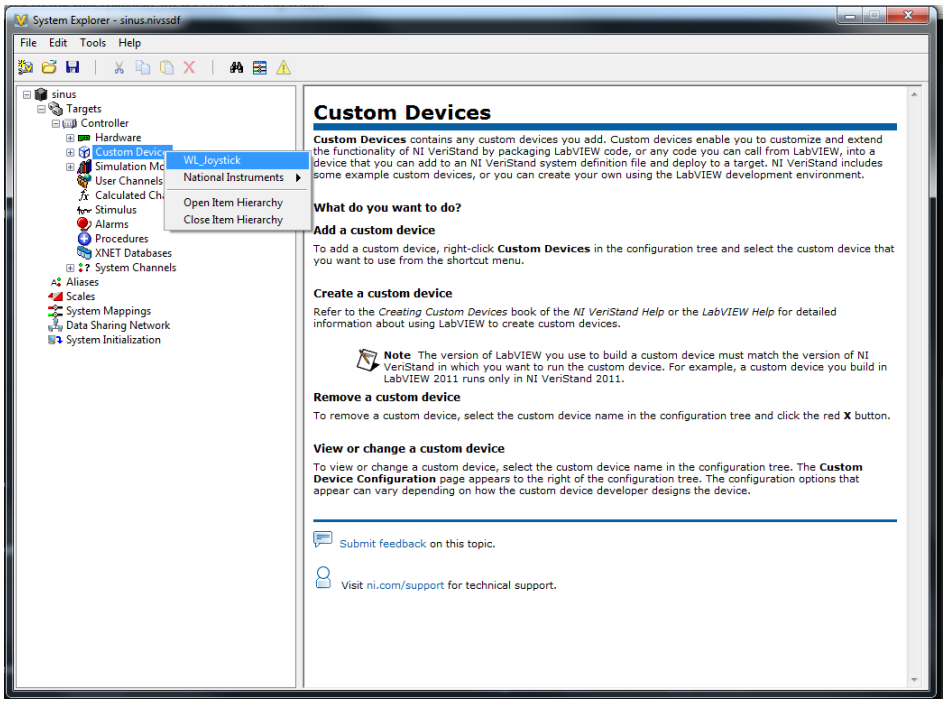


Figure A.23: Custom device selection

installation is successful you should be able to see WL_Joystick folder under custom devices as seen in the red box in Figure A.24. Here you will also see the different inputs from the custom device, in this case it is joystick axis. To connect the joystick to the input ports of the Simulink model. You open the system configuration mappings (click the button marked by the arrow in Figure A.24). You the simply find the ports you would like to connect, mark them and click the connect button. Figure A.25 a joystick output is connected to a input port on the Simulink model.

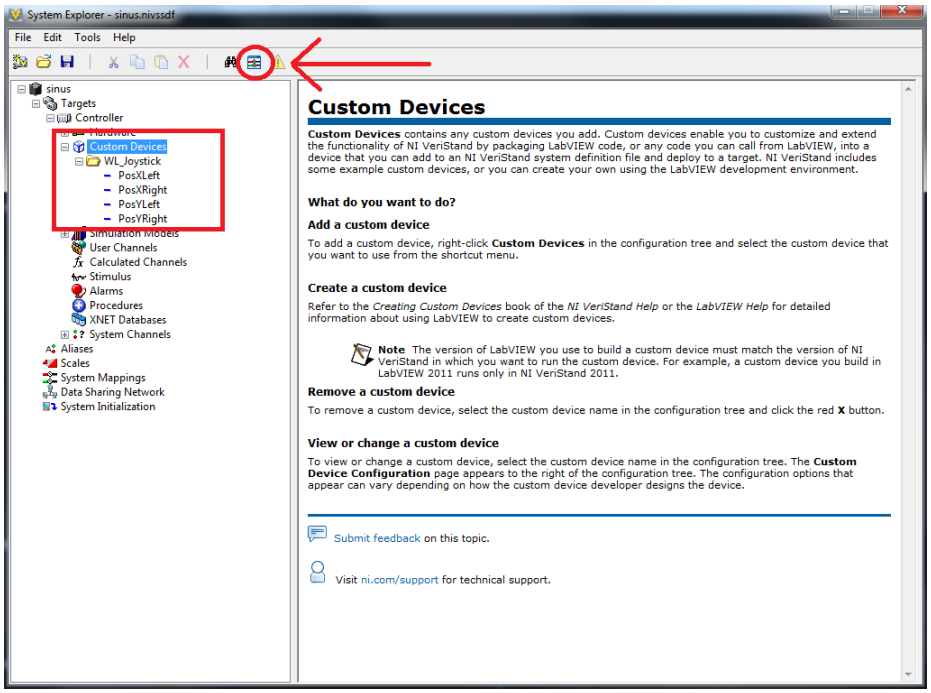


Figure A.24: VeriStand

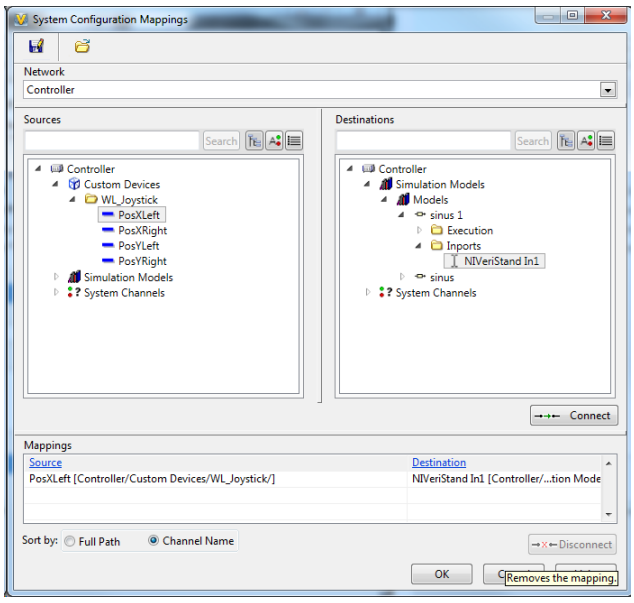


Figure A.25: VeriStand System Configuration Mappings

## A.3 Raspberry Pi

The unit is configured with Raspbian Linux-kernel-based operating system

### A.3.1 Raspbian installation and setup

This section describes how to install and access the Raspbian operating system on the RPi from a Windows computer. The operations are also possible from an OSX or Linux computer.

#### A.3.1.1 Download operating system and utilities

Download and extract the newest Raspbian[1] operating system (OS) image. Necessary utilities for the setup are

- Win32 Disk Imager[2] to write the OS image to the RPi SD card
- Advanced IP scanner[3] to find the RPi address on the network
- Putty terminal emulator[4] for SSH connection
- WinSCP[5] for file transfer

| Windows | Linux, OSX |
|---|---|
| Win32 Disk Imager | dd |
| Advanced IP scanner | nmap |
| Putty | ssh |
| WinSCP | sftp |

Table A.1: RPi installation and setup utilities

See Table A.1 for a list of the equivalent software for OSX and Linux.

#### A.3.1.2 Write image to SD card

Since the .iso file is raw, it needs to be written to the SD card in way that makes it bootable. Win32 Disk Imager does this. Run the program as administrator.

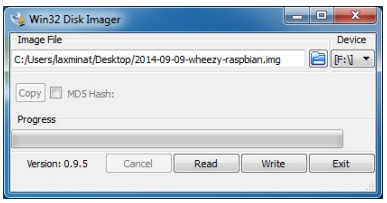


Figure A.26: Disk Imager

Select the correct image file and device, as in Figure A.26. Make sure that you have selected the correct drive before you push `Write`. Once the write is complete, insert the SD card in the RPi and boot.

[1] raspberrypi.org/downloads
[2] sourceforge.net/projects/win32diskimager
[3] by Famatech, advanced-ip-scanner.com
[4] www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[5] by Martin Prikryl, winscp.net/eng/download.php

### A.3.1.3 Terminal access

RPi can be accessed through the network, i.e. without having to directly connect a monitor and keyboard. At first boot, the RPi by default waits to be assigned

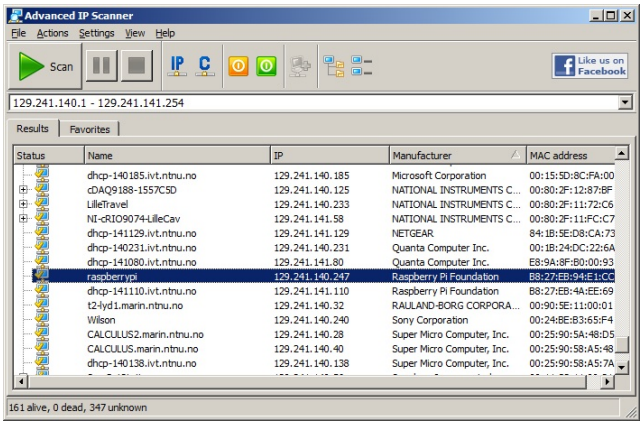


Figure A.27: Advanced IP Scanner

an IP address by DHCP. If this address is not known, scan the network with Advanced IP Scanner. It is advicible to sort the results by manufacturer since it is fixed (*Raspberry Pi Foundation*). The name is typically *raspberrypi*. See Figure A.27. Once the IP is known, it is specified in the Putty settings, as in

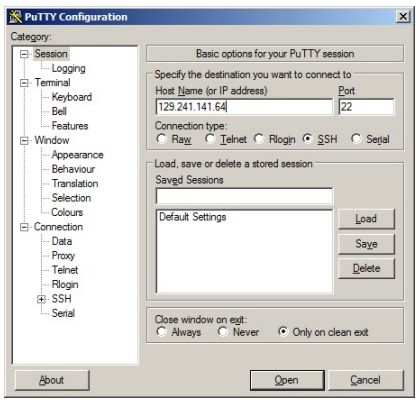


Figure A.28: Putty settings

Figure A.28, and a connection can be opened. The default login is `pi`, and the default password `raspberry`. Figure A.29 shows the terminal output on first login.

### A.3.1.4 Finalize configuration

Enter the

```
sudo raspi-config
```

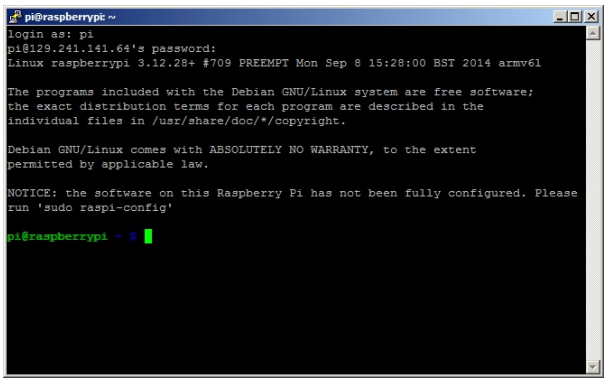


Figure A.29: SSH connection

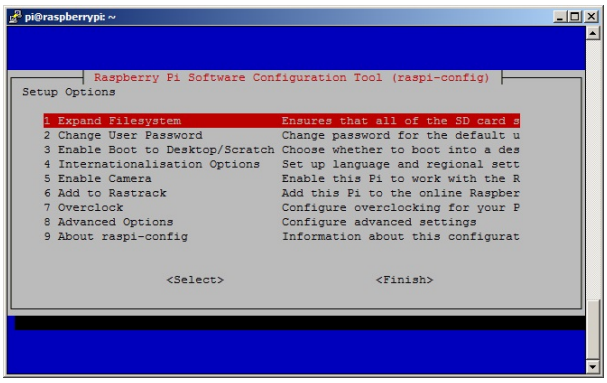


Figure A.30: RPi configuration tool

command to start the RPi Software Configuration Tool, as in Figure A.30. Use the menu to apply the following

1. Update configuration tool: 8 Advanced Options >A9 Update
2. Change password: 2 Change User Password
3. Expand filesystem: 1 Expand Filesystem >Finish

Exit the configuration tool and select `Yes` for reboot. Reconnect through Putty. Finally, update the repository package lists and upgrade all packages currently installed on the RPi:

```
sudo apt-get update
sudo apt-get upgrade -y
```

This process took approximately 10 minutes on a 90 Mbps internet connection.

#### A.3.1.5 Transfer files to RPi from computer

WinSCP can be used to transfer files to the RPi. This is useful for instance when transferring code, or when the RPi is not directly connected to the internet.

#### A.3.1.6 Set fixed IP address

When the RPi is connected directly to the cRIO or computer, a fixed IP is necessary since there is no DHCP server in that network. During most of this setup, however, it is preferable to keep the default DHCP assigned IP setting. To set a fixed IP

1. Open the network interface configuration information file for editing

   ```
   sudo nano /etc/network/interfaces
   ```

2. Alter the eth0 settings from `dhcp` to `static` and add address and netmask as

   ```
   auto eth0
   iface eth0 inet static
   address 192.168.1.22
   netmask 255.255.255.0
   ```

3. Save the changes by the key combination `Ctrl+X`.

The new IP is applied on the next reboot.

### A.3.2 Sixaxis installation and configuration

This section describes how to install and configure the Sixaxis gamepad for Bluetooth connection to the RPi, and how to add a server for sending joystick signals to the cRIO.

#### A.3.2.1 Download and install bluetooth support

BlueZ is the official Linux Bluetooth stack. It provides support for core Bluetooth layers and protocols. To download and install, type

```
sudo apt-get install bluez-utils bluez-compat bluez-hcidump
libusb-dev libbluetooth-dev joystick checkinstall -y
```

The process takes a few minutes. To confirm the installation, use the `hciconfig`

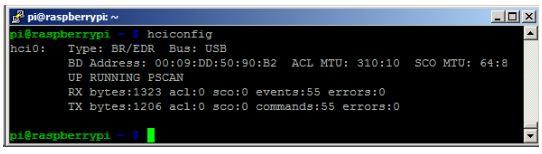


Figure A.31: Bluetooth configuration tool

command to print name and basic information about Bluetooth devices installed in the system. The output should include `UP RUNNING PSCAN`, as in Figure A.31. If instead it says `DOWN`, some error har occured. Most experienced errors were due to typos.

#### A.3.2.2 Bluetooth pairing

Sixaxis does not support the standard Bluetooth paring prcedure, instead, pairing is done over USB. The `sixpair` command-line utility[6] searches USB buses for Sixaxis devices and tells them to connect to a new Bluetooth master.

Download and compile the program by the following commands:

```
wget http://www.pabr.org/sixlinux/sixpair.c
gcc -o sixpair sixpair.c -lusb
```

Connect the Sixaxis by USB before running the paring utility

```
sudo ./sixpair
```

The output should be similar to

```
Current Bluetooth master: 00:02:72:BF:BC:8F
Setting master bd_addr to: 00:02:72:BF:BC:8F
```

The addresses at the end of each line will only be the same if you have already paired the Sixaxis with the Bluetooth dongle. First time they will be different. The Sixaxis USB cable may now be disconnected.

#### A.3.2.3 Joystick manager system service

`QtSixA`[7] reads the Sixaxis signals and makes them available to other programs. This program needs to run automatically whenever the RPi is booted.

To download the program, type

```
wget http://sourceforge.net/projects/qtsixa/files/QtSixA%201.5.1/QtSixA-1.5.1-src.tar.gz
```

To install, type

```
tar xfvz QtSixA-1.5.1-src.tar.gz
cd QtSixA-1.5.1/sixad
make
sudo mkdir -p /var/lib/sixad/profiles
sudo checkinstall -y
```

Update the system service list with sixad driver and reboot

```
sudo update-rc.d sixad defaults
sudo reboot
```

To test the program, turn on the Sixaxis (round PS button in the middle) and start the test program

```
sudo jstest /dev/input/js0
```

The terminal should now fill up with numbers that change as you move the analogue sticks and press the buttons on the Sixaxis. Exit the program by the key combination `Ctrl+C`.

[6] by Pabr Technologies, www.pabr.org

[7] the Sixaxis Joystick Manager by falkTX, qtsixa.sourceforge.net

#### A.3.2.4 Joystick signal server

A server must run to make joystick signals available over the RPi ethernet port. This should also start whenever the RPi is booted.

Transfer the source file `jscont.c` to the RPi (see Section A.3.1.5), then compile:

```
g++ -o jscont jscont.c
```

To verify that the program runs correctly, turn off (hold PS3 button for about 10 seconds) the previously paired Sixaxis and start the program

```
./jscont
```

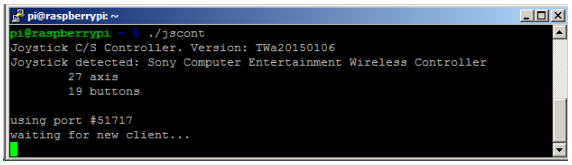


Figure A.32: Joystick signal server test

The program should then wait until you turn on the Sixaxis before giving output simular to Figure A.32. To exit the server use the key combination `Ctrl+C`.

Next, disable login at start-up in the bootup service description `inittab`:

1. Open the file for editing

   ```
   sudo nano /etc/inittab
   ```

2. Change the line that reads

   ```
   1:2345:respawn:/sbin/getty --noclear 38400 tty1
   ```

   by adding `--autologin pi` to get

   ```
   1:2345:respawn:/sbin/getty --autologin pi --noclear 38400 tty1
   ```

   **Warning:** Typos here may result consequences hard to correct.

3. Save and exit the changes by the key combination `Ctrl+X`.

Finally, add `jscont` to the login execution file:

1. Open the file for editing

   ```
   sudo nano /home/pi/.bashrc
   ```

2. At the very end of the file, add

   ```
   sudo ./jscont
   ```

3. Save the changes by the key combination `Ctrl+X`.

RPi should now be sending joystick signals at start-up.