

Licel Waverider – Installation and Reference Manual



Licel GmbH

December 5, 2025

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 1.1 | System description | 3 |
| 1.2 | Typical Experimental Setup | 3 |
| 1.3 | Cable Connections | 4 |
| 1.3.1 | Front | 4 |
| 1.3.2 | Rear | 5 |
| 2 | Software Installation | 6 |
| 2.1 | Preparation | 6 |
| 2.2 | The Licel CD ROM | 7 |
| 2.3 | Download | 9 |
| 2.4 | Installing the Windows Applications | 10 |
| 2.5 | Installing the Licel LabVIEW Libraries | 12 |
| 3 | Setting up the Network | 15 |
| 3.1 | Network Introduction | 15 |
| 3.2 | Preparations | 16 |
| 3.3 | Network Information | 16 |
| 3.4 | Network Preparation | 17 |
| 3.4.1 | Establish the Connection | 17 |
| 3.4.2 | Diagnostics | 20 |
| 3.5 | Network Setup | 21 |
| 3.5.1 | Fixed IP Address | 21 |
| 3.5.2 | DHCP Mode | 22 |
| 3.6 | Reconfigure the PC | 23 |
| 3.7 | TCP/IP Connection Parameters | 24 |
| 3.7.1 | TCP/IP Connection Problems (Software) | 25 |
| 3.8 | Network Security | 26 |
| 3.8.1 | Changing the Administrator Password | 26 |
| 3.9 | Hardware Reset | 27 |
| 4 | Waverider Software Tutorial | 28 |
| 4.1 | Overview | 28 |
| 4.2 | LiveDisplay | 28 |
| 4.2.1 | Stop the Live Display | 30 |
| 4.3 | Acquis | 30 |
| 4.3.1 | Running Wind Acquis | 30 |
| 4.3.2 | Checking the Data Directory | 30 |
| 4.3.3 | Wind Acquis Front Panel | 31 |
| 4.3.4 | Configuration | 32 |
| 4.3.5 | On-the-Fly Change of Parameters | 33 |
| 4.3.6 | Run an Acquisition | 33 |

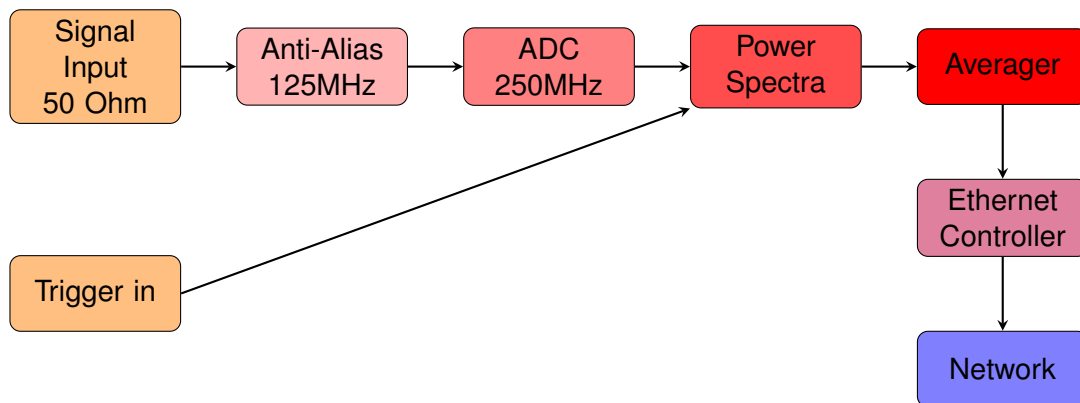
| | | |
|----------|---|-----------|
| 4.4 | Viewer | 35 |
| 4.4.1 | Run the Viewer | 35 |
| 4.4.2 | View Acquisition Details | 37 |
| 4.4.3 | Stop the Viewer | 41 |
| 5 | Queue Programming Interface | 42 |
| 5.1 | TCPIP Server | 42 |
| 5.2 | Queue Client - Getting Started | 44 |
| 6 | Data transfer - Low Level Description | 46 |
| 6.1 | Operation principle | 46 |
| 6.1.1 | FFTSize | 46 |
| 6.1.2 | Distance | 46 |
| 6.1.3 | Power spectra | 47 |
| 6.1.4 | Shots | 47 |
| 6.2 | TCPIP Communication | 47 |
| 6.3 | Data Package format | 49 |
| 6.4 | Accumulated Power Spectra | 49 |
| 6.5 | Raw Data to Physical Value Conversion | 50 |
| 7 | Simulation | 51 |
| 8 | Specifications | 53 |
| 8.1 | Mechanical Dimensions | 54 |
| 9 | Appendices | 55 |
| 9.1 | TCP/IP Command List and Syntax | 55 |
| 9.1.1 | GET request | 55 |
| 9.1.2 | SET request | 55 |
| 9.1.3 | Request response | 56 |
| 9.1.4 | Command list | 56 |
| 9.2 | Data File Format | 62 |
| 9.2.1 | TDMS file format | 62 |
| 9.2.2 | NetCDF file format | 64 |
| 9.3 | VI List | 68 |
| 9.3.1 | WIND GettingStarted.vi | 68 |
| 9.3.2 | WindTDMS.llb | 68 |
| 9.3.3 | Wind_netCDF.llb | 71 |
| 9.3.4 | WindTCP_Server.llb | 73 |
| 9.3.5 | driver | 76 |
| 9.4 | Python Interface | 86 |

Chapter 1

Introduction

1.1 System description

The Licel Waverider module is designed to be used in pulsed coherent Doppler Wind systems. It takes the output of balanced fiber detector together with a synchronizing trigger pulse and computes power spectra from an 250MHz AC coupled 14bit ADC signal which are then averaged and transmitted over Ethernet to a PC. This hardware power spectra computation and averaging allows to run the system with a small amount of laser shots lost while keeping the SNR much higher due to the 14bit ADC.

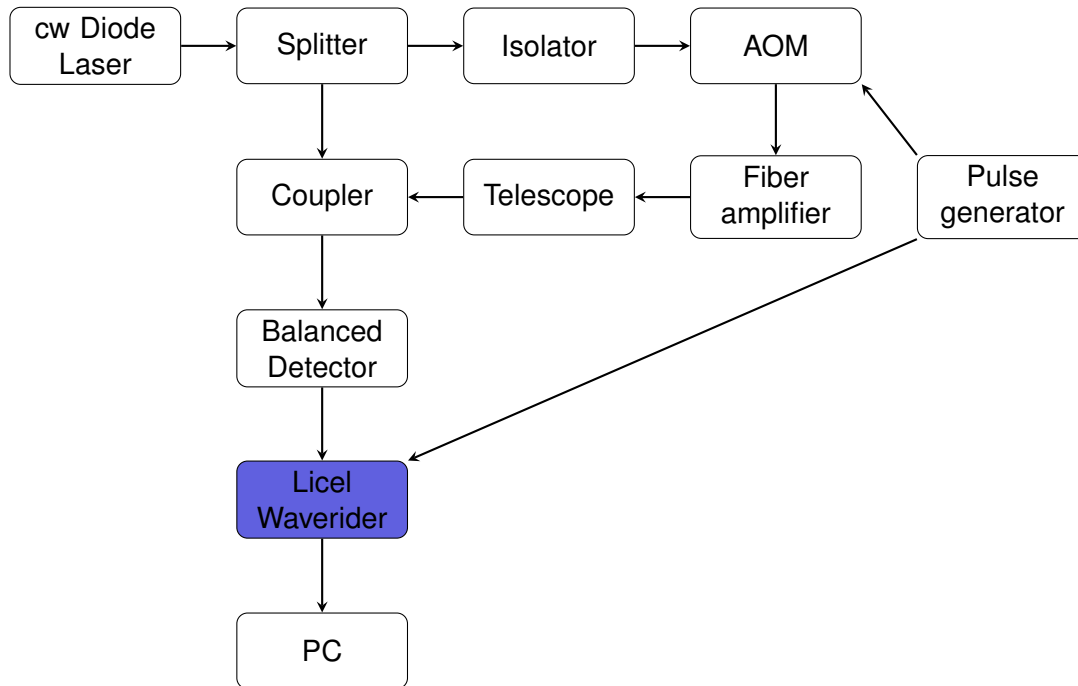


The [FFT size](#) is user selectable and can be 32, 64, 128, 256, 512 and 1024 which corresponds to 19.2 m, 38.4m, 76.8m, 153.6m, 307.2m and 614.4m.

The power spectra are averaged over a predefined number of [shots](#) and the [accumulated power spectra](#) are transferred over a TCP/IP connection to the PC. Licel provides software modules to [acquire the data and store the data](#) into data files. They can later be processed to compute the wind signal. Further the software is provided to [view the data files](#), to have an [oscilloscope like interface](#) to the instrument and to configure the network interface.

1.2 Typical Experimental Setup

The instrument is designed to be a component of larger lidar system. The white boxes should be provided by the system integrator. The blue box is the data acquisition system described here.

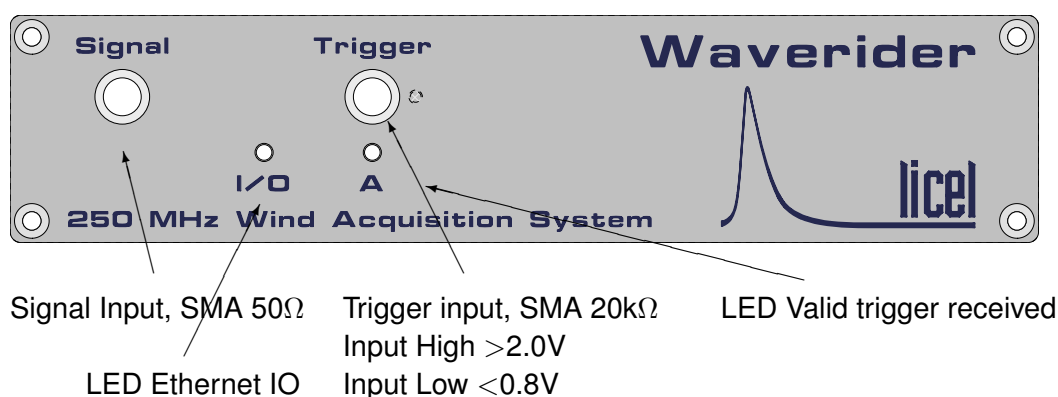


We also provide a complete acquisition software which can serve both as a starting point to acquire and store data and as an example how to implement a wind lidar acquisition software. The computation of the wind speed is intentionally not provided and put on the integrators, as this instrument serves as an instrument, where the user can completely control the process from the raw power spectra to the final wind speeds. We provide a peak finding algorithm implementation based on *Engelmann, R. 2010 Aerosol vertical exchange in the convective planetary boundary layer: Turbulent particle flux measurements with combined wind and aerosol lidar, PhD Thesis, University of Leipzig, Germany, 135 pp.*

For proper finding of the peak position one should do a background measurement, so that a background slope does not influence the peak position. For a background measurement the emission from the laser should be blocked.

1.3 Cable Connections

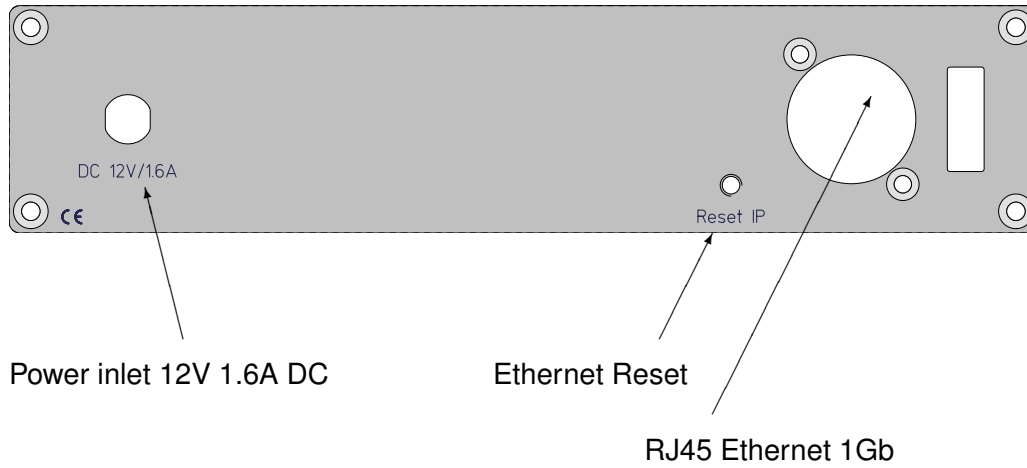
1.3.1 Front



The `trigger` in LED goes bright as soon as an acquisition is started and a valid trigger is received.

The I/O LED goes bright on communication both sockets (command and push). It also goes shortly bright when the system is fully booted, ca 30 sec after power on.

1.3.2 Rear



Chapter 2

Software Installation

Licel provides a package of software modules for setting up the Licel Waverider Controller for network operation, and for operating the Licel Control Modules. These software modules are written in LabVIEW's G language. The software is provided as LabVIEW source for users who have LabVIEW 2018 or later installed, or alternatively as a set of Windows applications. The Windows applications come within a Windows Installer package for an easy installation on your Windows 10 computer. Licel provides the software on a CD ROM and for download (<http://www.licel.com/wind.htm>).

2.1 Preparation

Windows Application Users

If you have used older versions of Licel Windows applications it is recommended to backup existing initialization files (*.ini).

Search the existing installation directory of the older version of Licel Windows applications (standard: <Program Files Directory> \Licel) and backup all files with the ending *.ini to an archive file (zip, ARJ, TAR, etc...).

LabVIEW Users

If you have used older versions of Licel LabVIEW libraries it is necessary to remove and backup older versions.

1. Backup all your current Licel software libraries, in case you want to restore them, by either compressing them (zip, ARJ, TAR, etc...) .
2. Scan your disks to find all versions of the following files and delete them once you have made backups of them

```
project\WIND_src.lvproj
shared\ctl\*. *
shared\Licel TCPIP.llb\*. *
shared\LicelUtil.llb\*. *
shared\user.lib\errors\*. *
WIND_PC\SearchControllers.llb\*. *
WIND_PC\driver\*. *
WIND_PC\WindTDMS.llb\*. *
WIND_PC\WindViewer.llb\*. *
WIND_PC\WindAcquis.llb\*. *
WIND_PC\WindLiveDisplay.llb\*. *
```



```
WIND_PC\WindTCP_Server.llb\*.*
WIND_PC\LicelFile.llb\*.*
WIND_PC\LicelGraph.llb\*.*
WIND_PC\Licel_netCDF.llb\*.*
WIND_PC\Licel_netCDF_Util.llb\*.*
WIND_PC\netCDF\*.*
WIND_PC\Wind_netCDF.llb\*.*
WIND_PC\Wind_netCDF_Viewer.llb\*.*
WIND_Simulation\*.*
```

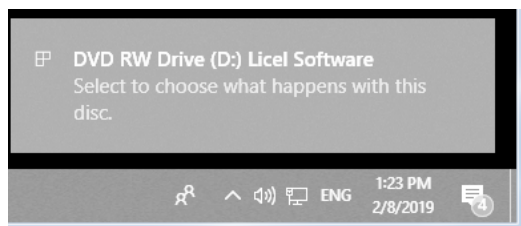
Please note: Licel may have provided individual software solutions with additional or less LabVIEW library files than noted in the list above.

3. Search the directory where your older version of Licel LabVIEW libraries reside and backup all initialization files (*.ini).
4. The LabVIEW sources are delivered including the following files and directories:
 - installation.txt a short description file
 - WIND_PC a directory containing the LabVIEW vis and initialization files that run on the PC.
 - WIND_Simulation a directory containing the Virtual controller for simulation of a Wa-verider system.
 - project a directory containing the LabVIEW project Wind.lvproj
 - shared\user.lib\errors\ Licel-errors.txt Licel error code file.
 - shared\user.lib\errors\ WIND-errors.txt Wind error code file.
5. The Licel-errors.txt and Wind-errors.txt files should be copied to the user error directory below the LabVIEW installation path. A typical path would be C:\Program Files (x86) \National Instruments \LabVIEW 2018\user.lib\errors.

2.2 The Licel CD ROM

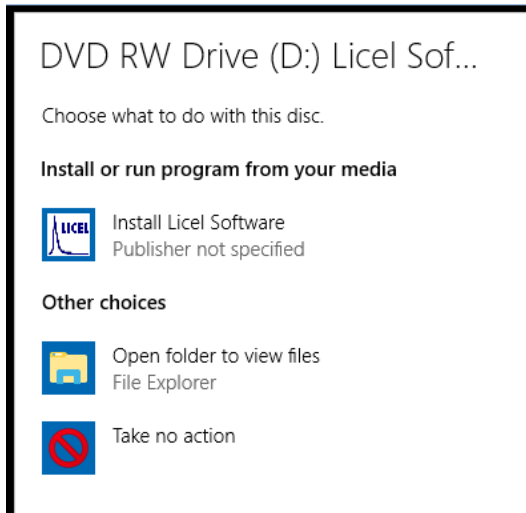
The standard CD ROM provided by Licel contains both, the LabVIEW sources and the Windows Installer for installing the Windows applications, and furthermore a documentation folder. Licel may add customer specific components on the CD ROM.

1. Insert the Licel CD into your CD ROM drive.
2. In Windows 10 you will normally be notified by a pop-up message at the bottom right corner of the main monitor.



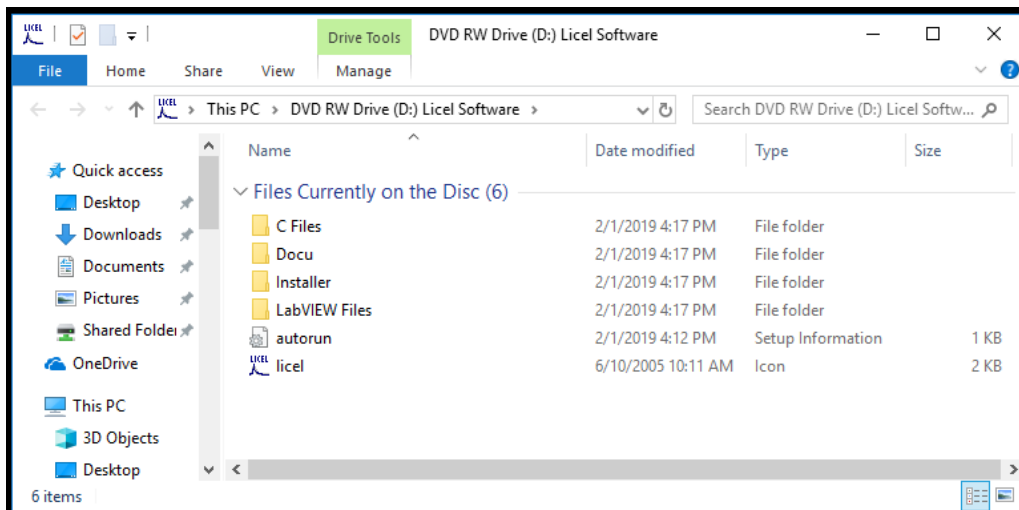
Please click on the pop-up message.

3. The following selection dialog should appear:

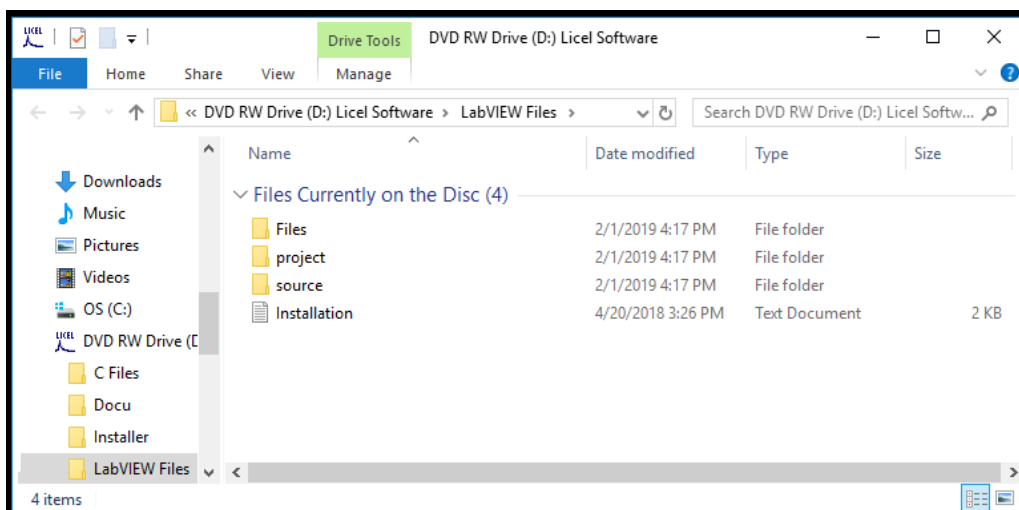


In older Windows operation systems a similar dialog will automatically come up.

- Press *Install Licel Software* to start the Windows Installer which will guide you through the installation of the Licel Applications. Please proceed to the section [2.4](#).
- Press *Open folder to view files* to start the File Explorer (Windows Explorer) to see the content of the CD:

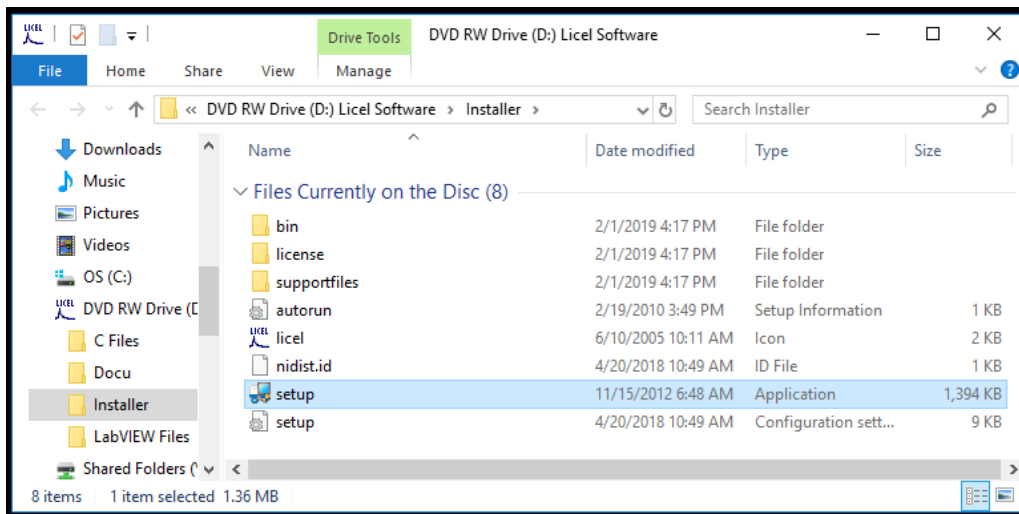


- The LabVIEW source files are located in the folder *LabVIEW files*. From there you may copy them to a directory of your choice on your local PC.



Please note the [remarks](#) according to existing LabVIEW library files. Please refer to the section [2.5](#) for further details.

- In the folder *Docu* you will find some documentation.
 - The folder *C Files* contains Licel's C sources.
4. If the [selection dialog](#) does not come up automatically after inserting the CD into your CD/DVD drive, please manually open the File Explorer (Windows Explorer) and navigate to the CD/DVD drive of your PC.
- Either go to the folders *LabVIEW Files*, *Docu*, or *C Files* to get the LabVIEW source files, read the documentation, or copy the C source files,
 - or open the folder *Installer* and run *setup.exe* by double click to start the Windows Installer.



Please proceed to the section [2.4](#) afterwards.

2.3 Download

The Licel software is frequently maintained. The most recent version is available on the download page (<http://www.licel.com/wind.htm>). Licel provides both packages described in this chapter, the LabVIEW sources as well as the Windows installer to deploy the Windows applications. The packages come as zipped archive files, `WaveriderLVSource.zip` contains the LabVIEW sources, while `WaveriderLVInstaller.zip` is the corresponding zip archive with the Windows installer. Note that you may have changed these files names while downloading the archives.

Unpacking the Windows Installer

If you downloaded the Windows Installer package (`WaveriderLVInstaller.zip`) please unzip all files to a temporary directory. Locate the setup routine `setup.exe` in that directory and run it by double-clicking the program entry in the Windows Explorer. Please proceed to the section [2.4](#).

Unpacking the LabVIEW Sources

The Licel LabVIEW libraries and initialization files contained in the zip file `WaveriderLVSource.zip` may directly be unzipped to a destination folder of your choice. Please note the [remarks](#) according to existing LabVIEW library files. Please refer to the section [2.5](#) for further details.

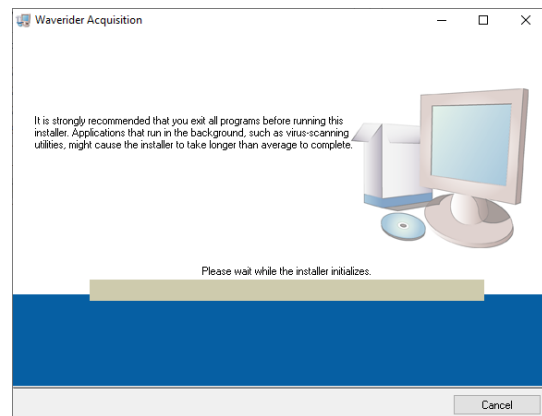
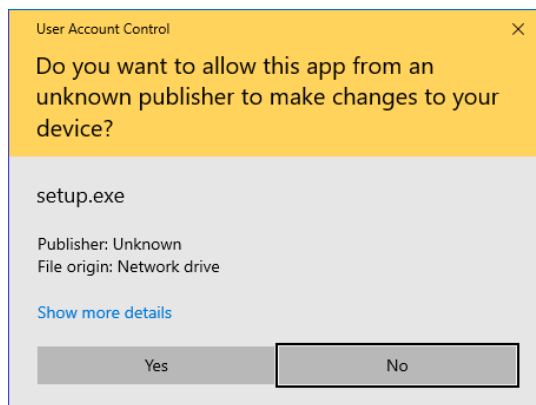
2.4 Installing the Windows Applications

This subsection describes the installation process for the Licel Windows applications. To operate the Licel Windows applications a LabVIEW runtime environment needs to be installed, as well. The Windows applications together with the LabVIEW runtime environment come as a Windows Installer package. For the installation of the LabVIEW runtime part of the installer package local administrator privileges are required.

The following items describe the installation process after starting the Windows Installer's setup routine (`setup.exe`). The setup program is automatically started when using the CD ROM and pressing **Install Applications** in the [setup selection dialog](#). `setup.exe` is located on the Licel CD ROM in the subdirectory `Installer` or in the temporary directory you unzipped the downloaded Licel Installer package. You may directly start the setup routine from the corresponding directories.

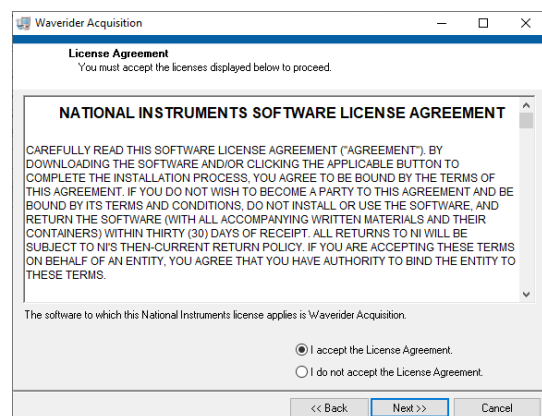
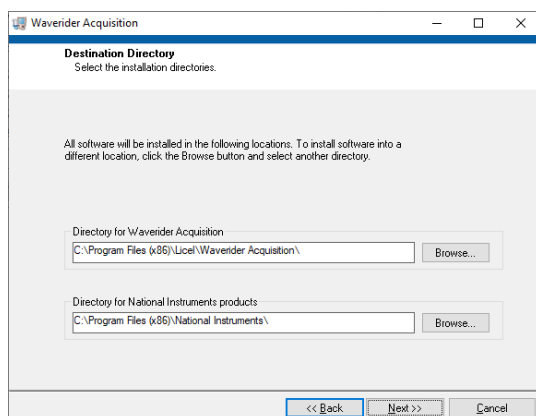
The Windows Installer dialogs will guide you through the installation process.

1. At the very first start of the installation the User Account Control dialog may appear. Click **Yes** to continue the installation process.



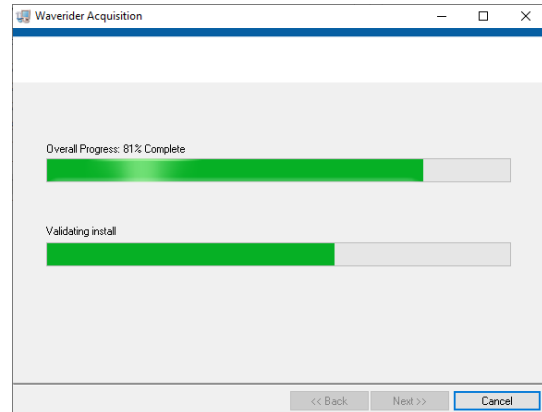
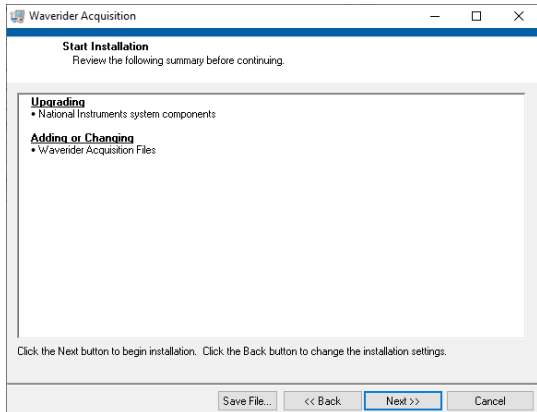
Afterwards, the Installation will be initialized and the destination directories will be shown.

2. You may change the installation directories using the **Browse** button. Click **Next** to proceed.

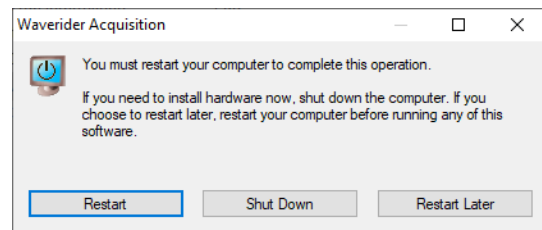
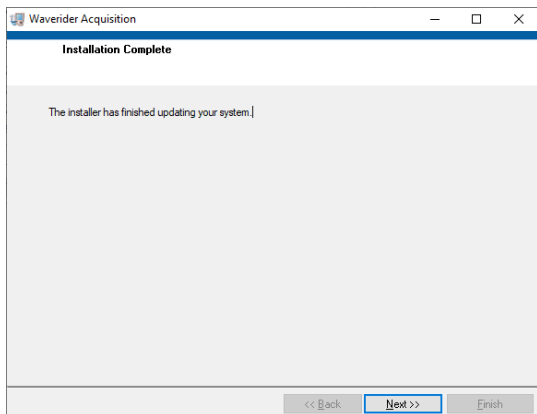


In the next dialog you have to accept the License Agreement(s). Choose *I accept the License Agreement* and **Next** to proceed.

3. Confirm the following dialog using the **Next** button or click **Back** to change your installation settings. After starting the installation the progress will be indicated by a progress bar.

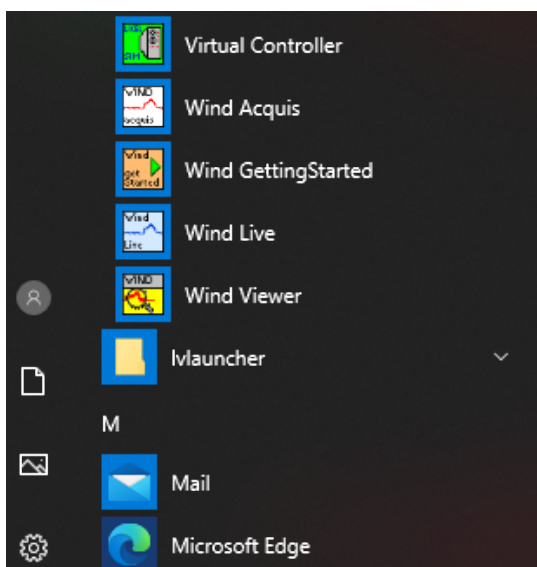


4. After the installation process is completed an information window will be shown. Click **Next** to proceed.



To finish the installation you may have to restart your computer. If a restart is required click **Restart** to complete the installation.

5. After the installation has successfully been completed you are able to start the Windows applications through the corresponding folder **Licel**. To open the folder go to the Windows start menu and browse the Letter L.



6. If you have backedup your initialization files from an older version of Licel Ethernet Software you may copy the TCP/IP parameters from the corresponding old [initialization files](#) to the files

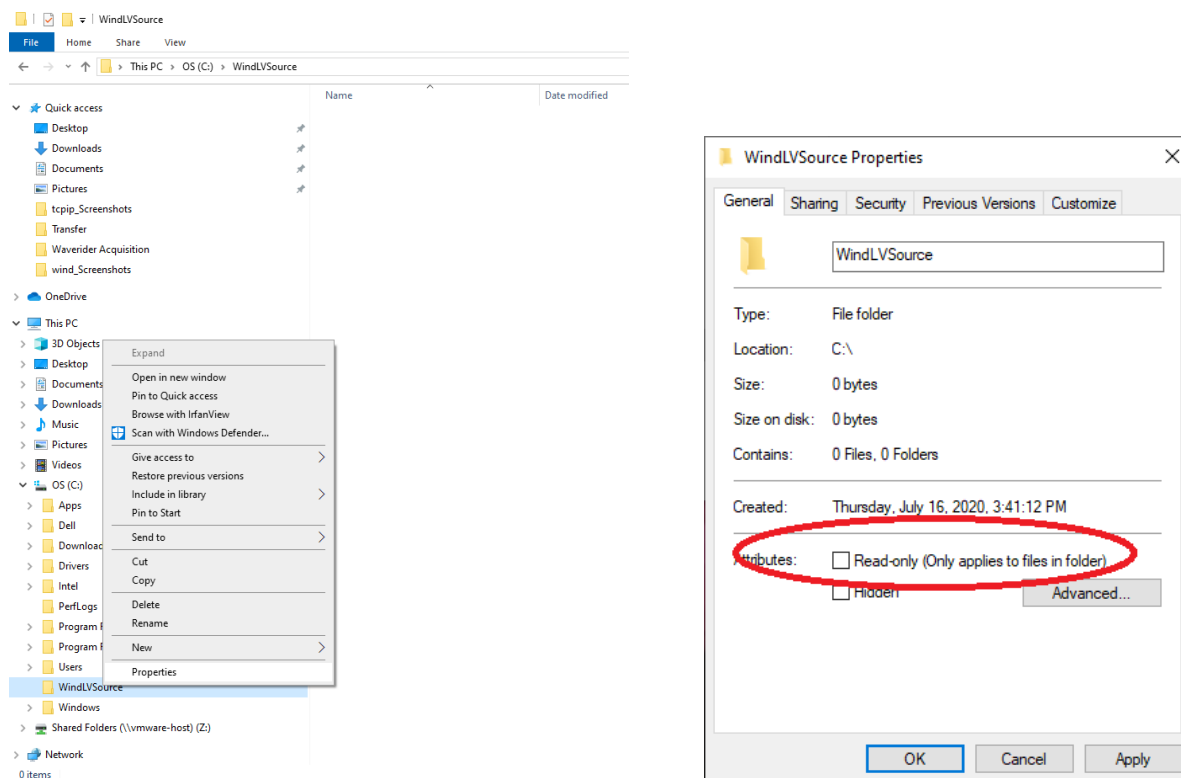
of the current installation. Please note that copying information from older to new initialization files should be done value by value (line by line).

2.5 Installing the Licel LabVIEW Libraries

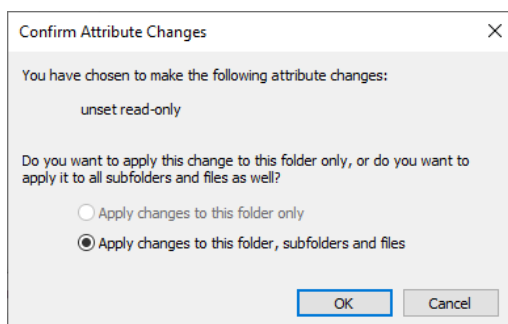
To install the Licel LabVIEW libraries you may choose between the following options:

- The Licel LabVIEW Libraries will be copied automatically from the Licel CD ROM by pressing **Copy LabVIEW Source** in the [setup selection dialog](#). You will be asked to select or create a target folder.
- You may manually copy all files contained in the directory `LabVIEW Files` on the CD ROM to a directory of your choice.
- If you downloaded the Licel software from <http://www.licel.com/wind.htm> please unpack the content from the downloaded zip file and copy it to a directory of your choice.

Please note that in the case the software is copied from a CD you may have to unselect the "Read-only" attribute for the destination folder. This is done by selecting the directory and right-clicking on it. Select **Properties** from the context menu.

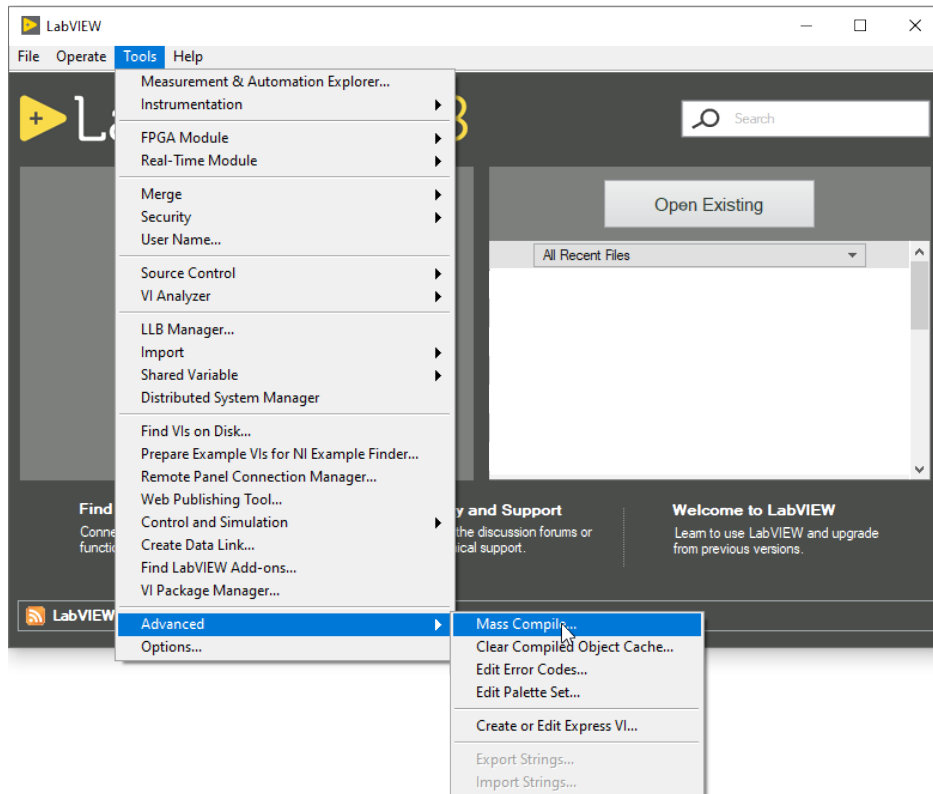


Verify that the "Read-only" attribute is not checked, uncheck it if necessary. Click **OK** and check in the next dialog *Apply changes to this folder, subfolders and files*. Leave the dialog by clicking **OK**

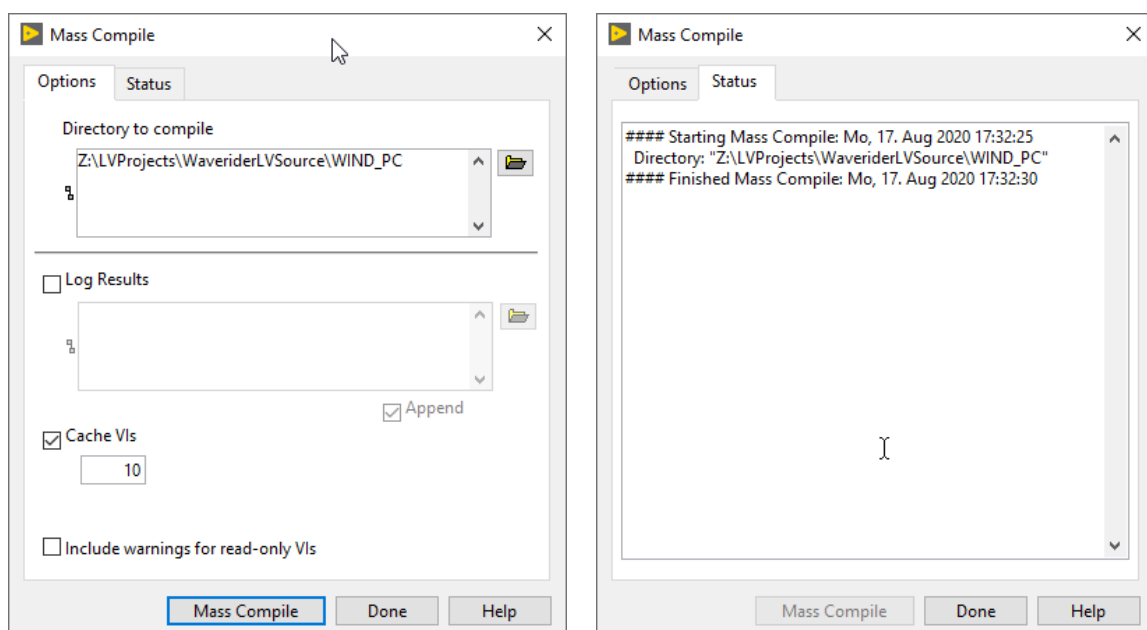


Now you should be able to run all the files. If you are still having problems, apply a mass compile to the directory where the software was extracted to:

1. Start LabVIEW.
2. Select the menu **Tools**, then **Advanced**, and finally **Mass Compile....**



3. You will be asked to select a directory, select the target directory of the LabVIEW source files.
4. Press *Mass Compile* in the next dialog.



5. Later the mass compile status will be shown.

Please note that the [removal of older libraries](#) is a necessity, since LabVIEW often links to various libraries with the same name. As a result, if a library is installed twice, one can not be certain which library is actually being used.

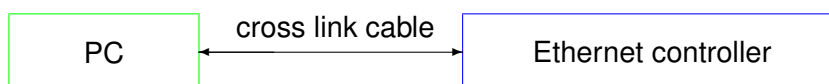
If you still have any problems, please contact Licel for further assistance.

Chapter 3

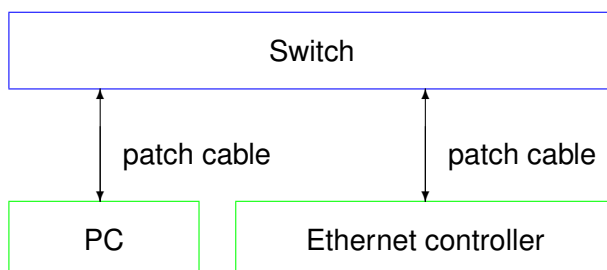
Setting up the Network

3.1 Network Introduction

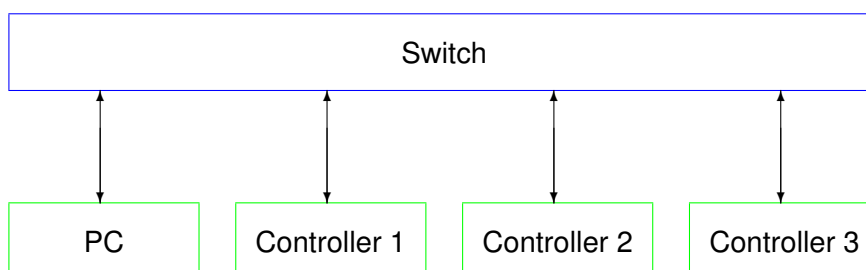
To control a Waverider controller a working TCP/IP connection is required. This can be reached by two ways, using a cross link cable, which creates a one to one connection between the PC and the Waverider Controller or with patch cables and a switch



The cross link cable might be a perfect setup for single controller, but as soon as the PC needs to communicate over the same network connector with other nodes locally or the Internet, the usage of a switch is mandatory.



This configuration has the big advantage that it is easily scalable if more than one controller needs to be connected.



There are two concepts for the switch either:

- Use the local infrastructure, this requires coordination with your local network administrator as

she/he will define network addresses to be used for the PC and the Ethernet controllers or require DHCP for the nodes to be used.

- add a second Ethernet controller to the PC, so that Ethernet controllers can be moved to a private network and you become the administrator of this private network.

http://en.wikipedia.org/wiki/Private_network describes the available address ranges, selecting a network subset in the 192.168.0.0 192.168.255.255. seems like a good choice

In all of these configurations the PC and the controllers should be finally in the same subnet but have **different** IP addresses within this subnet. To achieve this, each controller needs to be specially setup as all controller ship with the same default network address. If more than controller needs be setup the procedure below needs to be repeated for each controller individually. **Never** connect more than one controller with the factory default to a network.

3.2 Preparations

To operate the Waverider controller in your local network you will have to carry out the following required steps described in the corresponding subsections:

1. Get the required **Network Information**.
2. Prepare the PC to communicate with the Ethernet controller using a cross-link cable (**Network Preparation**).
3. Setup the Ethernet controller for your local area network either by setting a fixed IP address or by activating the DHCP mode (**Network Setup**).
4. **Reconfigure the PC** for your local area network and test the communication with the Ethernet controller.

3.3 Network Information

The Waverider controller is shipped with a default static IP address. The default parameters are:

| | |
|--------------|---------------|
| IP address | 10.49.234.234 |
| network mask | 255.255.255.0 |
| gateway | |
| port | 2055 |

The network parameters should be aligned according to your local network environment. Before doing this, the system administrator should be contacted. He should provide the following information:

1. Should the Ethernet controller use a dynamically assigned IP address (DHCP)?
 - (a) If yes, the network parameters will be set by a DHCP server residing in your LAN. Refer to the subsection [DHCP Mode \(3.5.2\)](#) to enable the Waverider controller to automatically receive the network parameters from the DHCP server.
 - (b) If a static address configuration is to be used,
 - i. the IP address,
 - ii. the network mask,
 - iii. and the gatewayshould be set by yourself. Refer to the subsection [Fixed IP Address \(3.5.1\)](#).

2. The default ports used by the Ethernet controller are 2055, 2056 and 2057. Can these ports be used?
3. Is it necessary to change the configuration of any firewall in the case you need to access the controller outside of the LAN boundaries?

3.4 Network Preparation

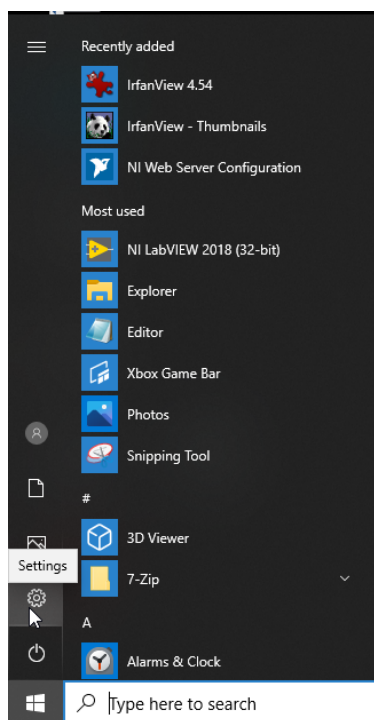
After having installed the [Licel Windows applications](#) or the [Licel LabVIEW modules](#) on your PC you are ready to change the network configuration parameters of the Waverider controller according to the local network settings described in the [previous section](#).

3.4.1 Establish the Connection

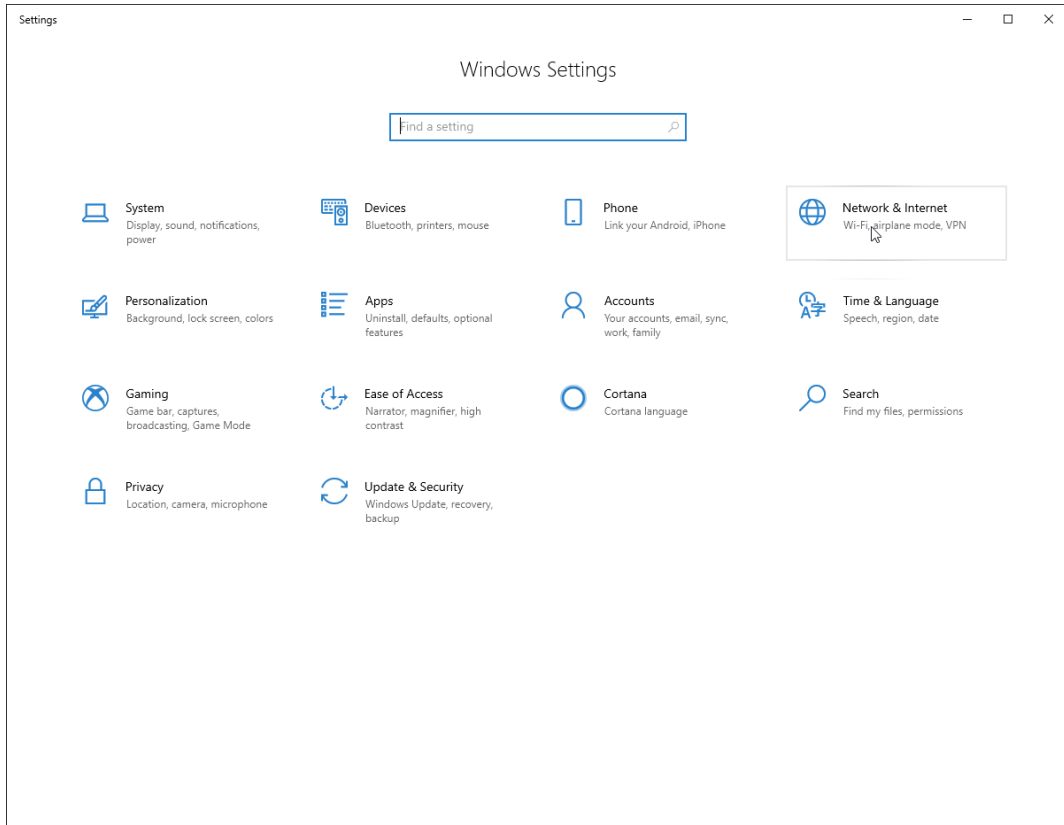
A straight-forward way to do this is the following procedure. You will need local administrator rights on your PC for the following steps:

1. Disconnect the PC from the local network.
2. Open the *Properties* dialog of the network connection your Ethernet adapter is assigned to. Usually you will find the appropriate network connection by opening *Network Connections* from the Windows start menu or the *System Settings*. The following list shows the steps to follow on a Windows 10 operating system:

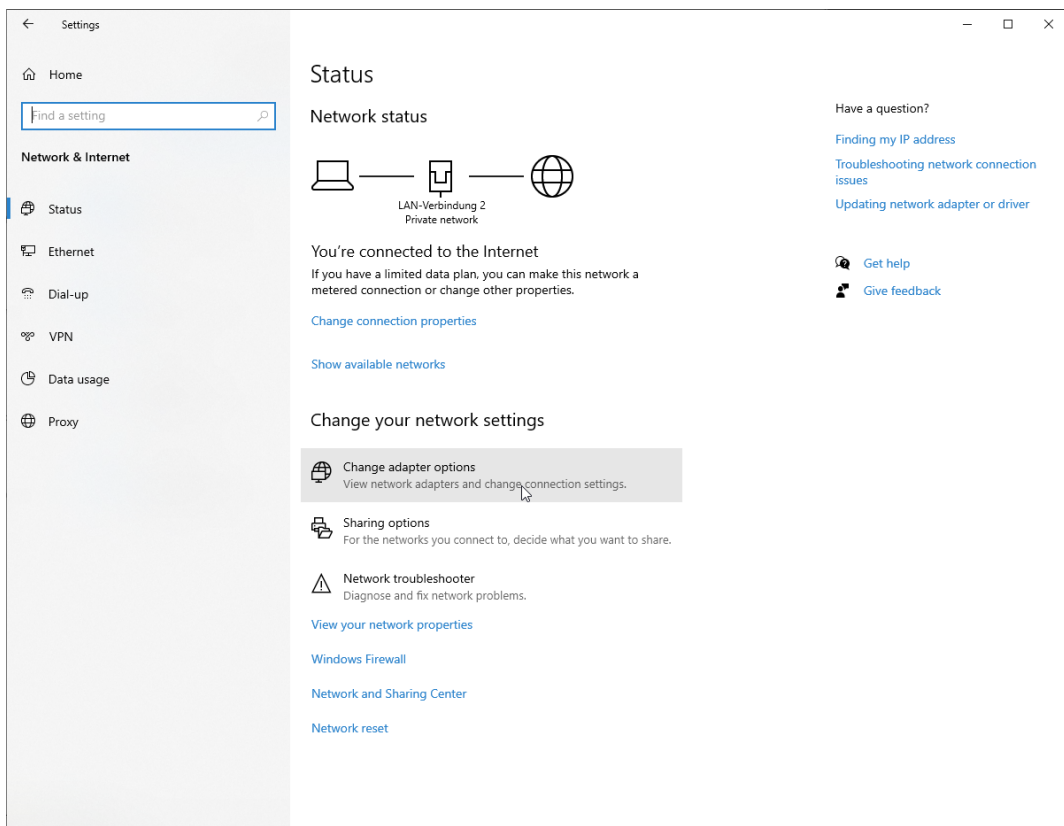
(a) Click on the  button, and then on *Settings*.



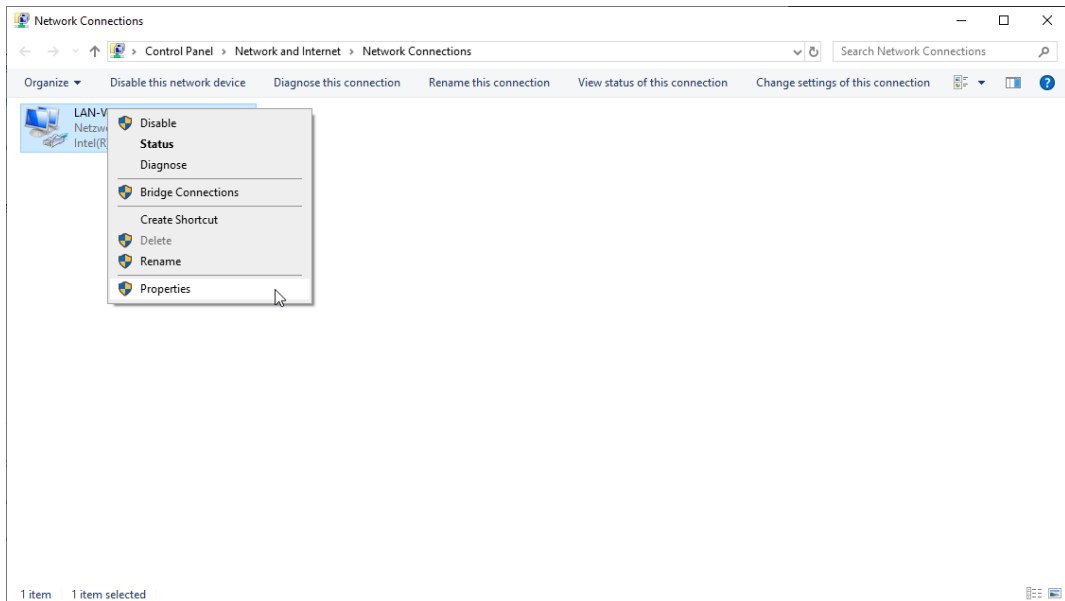
(b) Once the control panel has come up click on *Network and Internet*.



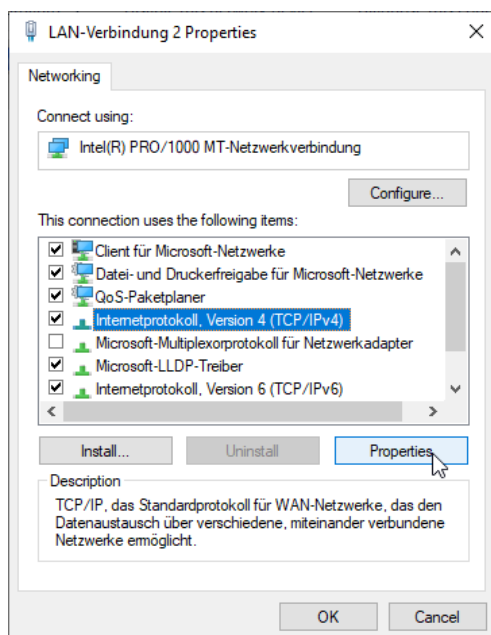
(c) In the next window click on *Change Adapter Options*.



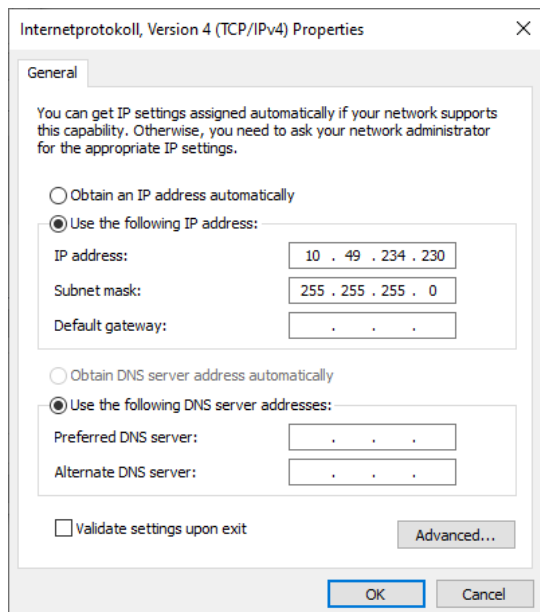
(d) The installed network connections will be shown, right-click on the local Ethernet connection to be used with the Waverider controller and choose *Properties* from the context menu.



- Click on the TCP/IP protocol entry in the lists of components used by the assigned Ethernet adapter card / LAN connection and press the *Properties* button.



- Write down your current TCP/IP settings i.e. all settings seen in the following graphics. You will need this information to reconfigure your PC to access the LAN again.



5. If activated disable DHCP (checkbox *Obtain an IP address automatically*) and manually assign an IP address within the default address range of the Waverider controller. A good choice would be 10.49.234.230. **Never use the default address (10.49.234.234) of the Waverider controller as IP address for your PC.**
6. Quit the dialog by pressing *OK*.
7. Reboot your PC.
8. Power up the rack with the Waverider controller and connect the PC with the controller using a **cross-link cable** shipped together with your hardware.

Now you should be able to access the Waverider controller via your Ethernet card. Please test this first connection with the methods given in the next section.

3.4.2 Diagnostics

Please carry out the following steps to verify that the connection of the Waverider controller with the PC is established.

1. Verify that the green **LNK** LED lights up indicating a correct electrical connection.
2. Verify that in case of a 100Mbit Ethernet connection the **Spd** lights up.
3. Verify that the network settings of your PC have changed according to your settings:
 - (a) Open a command prompt window (DOS box).
 - (b) Type `ipconfig` and press enter. At least one of the Ethernet adapters should show the address that you previously set (10.49.234.230). The response should be similar to the following:

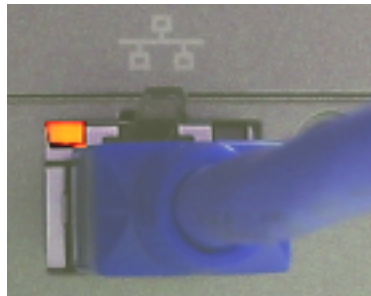
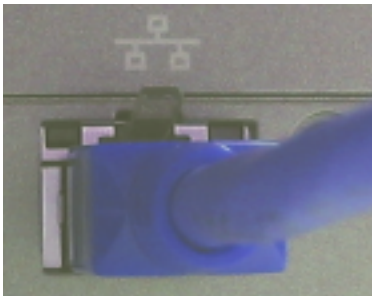
```
1 Ethernet Adapter :
    IP-Address. . . . . : 10.49.234.230
    Subnet Mask . . . . . : 255.255.255.0
    Standard-Gateway. . . . . :
```

4. Verify that the Waverider controller is accessible via the network now:

- (a) Open a command prompt window (DOS box) or use the one from above.
- (b) Type `ping 10.49.234.234` and press enter. The Licel Ethernet Controller should respond without loss of any packet. If the controller is not responding check if the network cable is correctly mounted and that an appropriate cable is used, i.e. a cross-link cable when working with a direct connection from the computer. Most Ethernet adapters indicate a correct connection with a green LED:



A non-existent or incorrect connection is often identified by an unlighted LED (left) or red LED (right).



Please note that these indicators may be different on your PC.

- (c) If the network cable connection is correct and the controller is still not responding execute a [hardware reset](#) and repeat the procedure with the [default IP address](#).

3.5 Network Setup

In order to configure the Ethernet controller, you need either to set the controller to a fixed IP address or invoke the DHCP Mode. Whether a fixed or dynamic (DHCP) mode is used or not will depend upon your network type. Dependent on this, please refer either to the subsection [Fixed IP Address](#) or [DHCP Mode](#) and skip the corresponding other subsection. Please contact your administrator if you have not yet requested the information described in the above subsection [Network Setup](#).

Afterwards you will have to [reconfigure your PC for operating in the local network](#).

Once you have set the **IP Address** and **Port** for the Licel Ethernet Controller you should [define these values to be used by the software](#).


3.5.1 Fixed IP Address

If you need to set the controller to a fixed IP address carry out the following steps. Skip the steps described in next subsection [DHCP Mode](#).

1. Open `Licel TCPIP Set New Fixed IP Address.vi` or the corresponding Windows application from the [Windows start menu](#).

- Please enter the new network parameters
- Run the vi
- Power Off / On the Licel Ethernet Controller

| | |
|-------------------------------------|-----------------------------------|
| Current IP Address 10.49.234.234 | New IP Address 192.168.69.12 |
| Current Port 2055 | Port 2055 |
| Password ***** | New Network Mask 255.255.255.0 |

2. Set the desired network parameters in the fields **New IP Address**, **Port**, and **New Network Mask**.
3. Do not forget to enter the correct **administrator Password**.
4. Run the vi by pressing the start  button. It should finish without opening an error message dialog.
5. Turn the Waverider controller off and switch it on again. Wait **approximately 20 – 30 seconds**.
6. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

3.5.2 DHCP Mode


In order to configure the Waverider controller for DHCP operation carry out the following steps. You must have skipped the steps described in the last subsection **Fixed IP Address**.

1. Open `Licel TCPIP Activate DHCP Mode.vi` or the corresponding Windows application from the **Windows start menu**.

- Please enter the DHCP Port
- Run the vi
- Power Off / On the Licel Ethernet Controller

| | |
|-------------------------------------|-------------------|
| Current IP Address 10.49.234.234 | DHCP Port 2055 |
| Current Port 2055 | |
| Password ***** | |

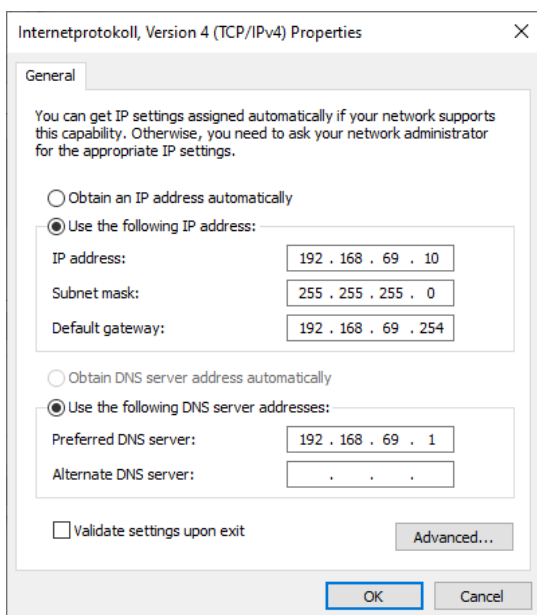
2. Set the desired **DHCP Port** number.
3. Do not forget to enter the administrator **administrator Password**.

4. Run the vi by pressing the start  button. It should finish without opening an error message dialog.
5. Turn the Waverider controller off and switch it on again. Wait **approximately 20 – 30 seconds**.
6. A `ping 10.49.234.234` executed from a command prompt (DOS box) should now time-out.

3.6 Reconfigure the PC

After you successfully configured the Waverider controller the following last steps have to be carried out to reconfigure your PC for the local network and to test the connection to the Waverider controller:

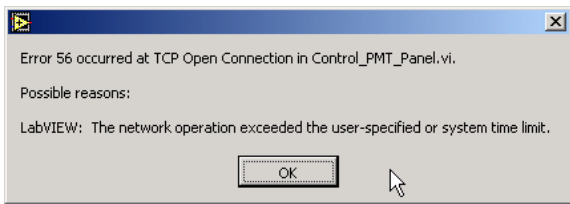
1. Reconnect the PC to the local network.
2. Open the *Properties* dialog of the network connection your Ethernet adapter is assigned to. A more detailed instruction has been given [above](#).
3. Open the *Properties* dialog of the TCP/IP protocol entry in the lists of components used by the assigned Ethernet adapter card.
4. Reset your current TCP/IP settings to the values you recorded while processing the subsection to establish a [network connection](#).



Note that the values shown here are just example settings. You must exactly use the settings present on your PC before configuring the Waverider controller.

5. Quit the dialog by pressing *OK*.
6. Reboot your PC.
7. Connect the Waverider controller with your local network through a hub or switch using an **ordinary patch cable**.
8. Execute a `ping` command from a command prompt (DOS box). Use the IP address you assigned to the Waverider controller. If the Ethernet controller is in DHCP mode try first `ping LicelWind` if this does not return value, you need to ask your system administrator for the assigned network address. The `ping` command's response should indicate a correctly working connection.

9. Test the access using `Licel Wind TCP.vi`.
10. A TCP/IP timeout error with LabVIEW's error code 56 may be caused by a wrong IP address.



Please check carefully that the values for **IP Address** and **Port** match with the parameters set at the Waverider controller. Set the correct values [as defaults](#) for future operation. Other reasons for errors with code 56 are non-existing connections (check if the cable in use is correct) or unstable network operation.

3.7 TCP/IP Connection Parameters

To work properly with the Waverider controller both the Windows applications and the LabVIEW software must be able to establish a TCP/IP connection. The user of the software must define the **IP Address** and **Port** – these values must be equal to the parameters that have been for the Waverider controller following the [network setup section](#).

Windows Applications: Initialization Files

The Windows applications are communicating with the `WindTCP_Server.vi` in the background over a queue interface. This `WindTCP_Server.vi` is communicating with the Waverider controller and uses an initialization file to read the TCP/IP parameters **IP Address** and **Port**.

For Windows applications this file is located at: `C:\Program Files (x86)\Licel\Waverider Acquisition\WIND.ini`.

An example for an initialization file holding the TCP/IP information is given below:

```
[Client]
IP = 10.49.234.234
PORT = 2055
```

You may edit this file using a text editor like `Notepad` which is installed by default when setting up a Windows operating system. You may use `Notepad` as well to create a required initialization file if it does not exist in the installation directory. Make sure that you save the file before leaving the editor. You must change the values for IP address and port to the values you will set following the Instructions in the [network setup section](#).

This file will be also used by the other applications to store information.

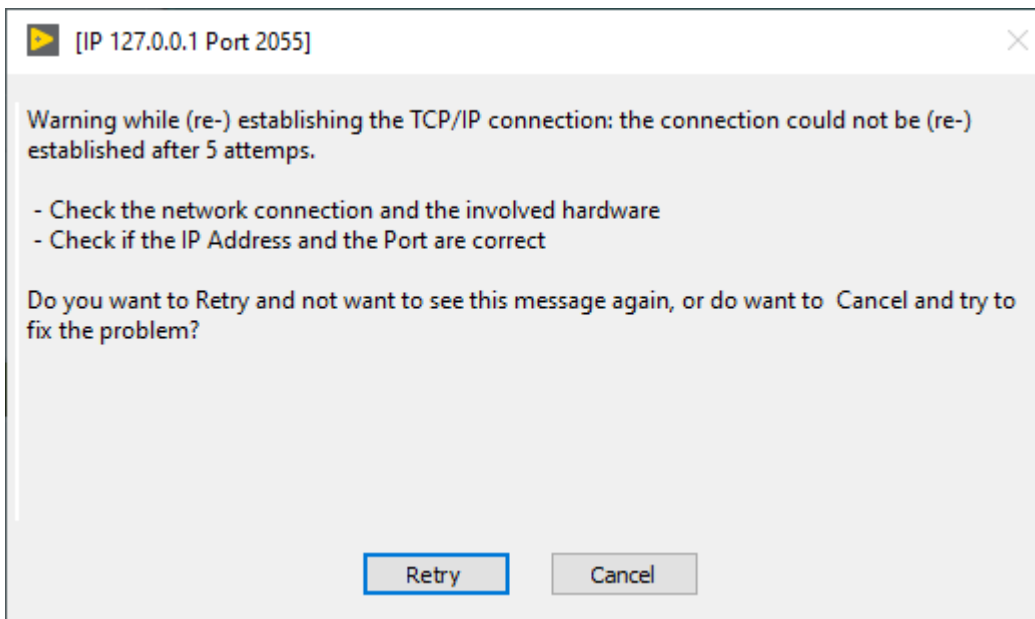
If the network address is not correct a error dialog will show up



3.7.1 TCP/IP Connection Problems (Software)

The `WindTCP_Server.vi` has a built-in mechanism to re-establish the TCP/IP connection to the Waverider controller when the connection is lost or when the connection is not successful after the program start.

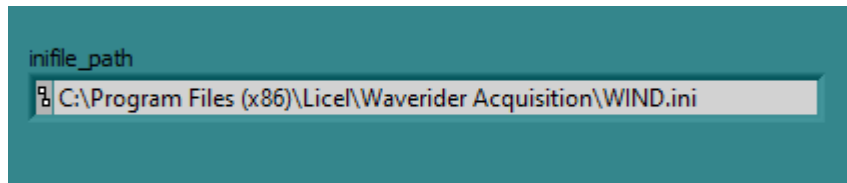
If the reconnection mechanism is not successful after 5 attempts the software assumes that some basic TCP/IP settings may be incorrect. Therefore the following error message is displayed:



In the case that this dialog comes up please

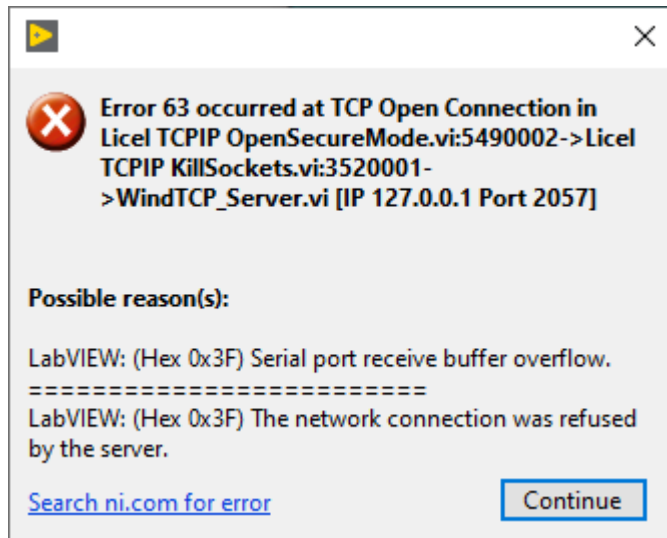
- check the network connection and the involved hardware. Check whether the Waverider Controller is switched on. Check that the Ethernet cable is plugged correctly, and that the correct Ethernet cable is used.
- check whether the **IP Address** and the **Port** the software is using equal to the values of the Waverider-Controller (refer to the [network setup](#)).

1. Before you start please enter the correct values for the **IP Address** and **Port**. You should already have set these values for the Waverider Controller
 - You should set the values in the [corresponding initialization file](#). You will see the full path of the file in a file path indicator on the *System* page.



You have two choices to leave the message dialog:

1. Click *Retry* to continue to reconnect to the Waverider controller.
2. Click *Cancel* to exit. The program will display an error message



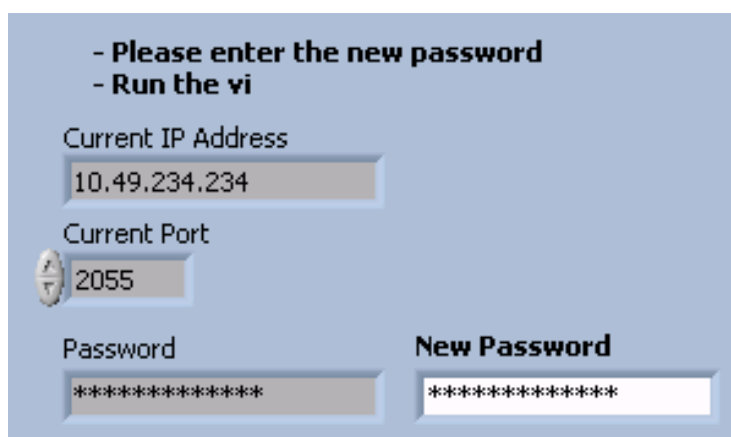
3.8 Network Security


Certain administrative tasks use an administrator password. An example is the change of the IP address of the controller. The administrator password has to be sent with the related commands.

3.8.1 Changing the Administrator Password

The Waverider controller is shipped with the default administrator password "Administrator". In order to change this password which grants administrative access to the controller, please carry out the following steps:

1. Open `Licel TCP/IP Set New Password.vi` or start the corresponding Windows application from the [Windows start menu](#).



2. Enter the current administrator **Password**.
3. Enter the **New Password**.
4. Run the vi by pressing the start  button. It should finish without opening an error message dialog. Please note that the password is case sensitive.

3.9 Hardware Reset

A reset is performed by pressing the reset switch while powering up the controller. The reset switch is located inside a hole close to the RJ45 connector.



To reset the system

- turn off the controller unit
- press the switch inside the hole with a small screw driver, Allen key or anything similar
- turn the rack on while keeping the switch pressed, release the switch when the IO Lamp turned on (up to 45 sec after switching the unit on).

After a reset

- the controller has the default [IP address](#)
- the port number is reset to the [default value](#)
- the controller operates in its [fixed IP address mode](#)
- the password is reset to the [default password](#).

Chapter 4

Waverider Software Tutorial

4.1 Overview

This software tutorial describes how to use the data acquisition software as well as the functions of the individual controls and indicators. In order to actually try the information in this tutorial, the hardware and [network](#) setup must be completed. This tutorial is broken into two parts. In the [Live Display](#) section a short introduction to viewing actual spectra is given. Then [Acquisition with Wind Acquis](#) contains instructions for recording your first spectra using [Wind Acquis](#). Which is then followed by the [Viewer](#) to display the previously recorded data.

4.2 LiveDisplay

The intention of this module is to give you a tool to verify that the trigger works and a valid signal is connected to the input. Once properly configured and connected it will show for every run the peak amplitude and peak frequency of the power spectra in Vrms2 units versus height. You need to setup first the controller IP and the start settings via the .ini File. For executables this is: C:\Program Files (x86)\Licel\Waverider Acquisition\WIND.ini. For the LabVIEW Sources the ini file is in the WindTCP_Server.llb D:\Waverider\WindTCP_Server.llb \WIND.ini

```
[Client]
IP = 10.49.234.234
PORT = 2055
HTTP_Port = 80

[global_info]
Location = "Berlin"
Longitude = 13.384373
Latitude = 52.542185
Height_asl = 45.000000
working_directory = "/C/temp"
Zenith = 0.000000
Azimuth = 0.000000

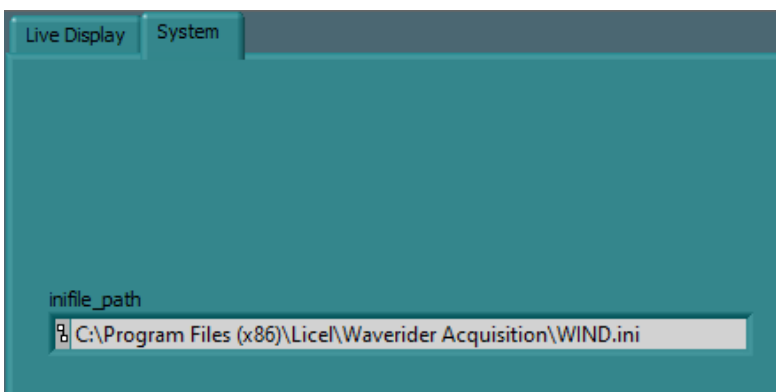
[Live]
Distance = 10000
Shots = 10000
;; FFT size parameters: 0 = 32, 1 = 64, 2 = 128, 3 = 256, 4 = 512, 5 = 1024
FFTsize = 1
```

```
[Operational]
;; 0 = SingleTriggerMode, 1 = DualTrigger, 2 = TrippleTrigger, 3 = QuadTrigger
WaveriderModus = 0
```

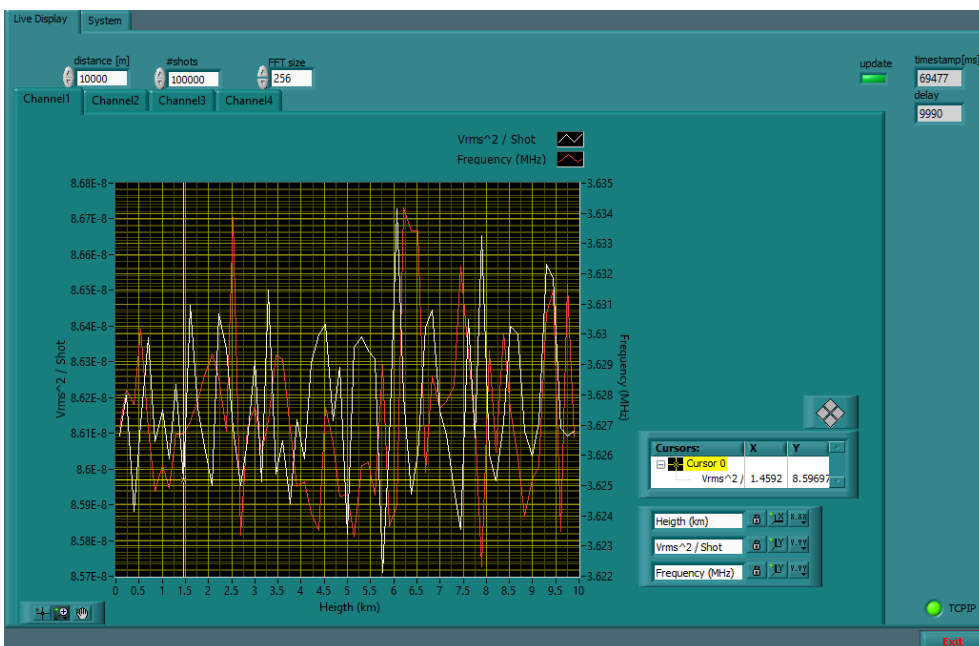
Start settings are: Distance (in meter), Shots FFTFTsize and the WaveriderModus. The Waveridermodus is set at start time and defines where a trigger signal will detected.

0. Trigger on A
1. Trigger on A and B
2. Trigger on A, B and C
3. Trigger on A, B, C and D

The ini file path can be checked on the `System` tab of the LiveDisplay application.



The measurement starts as soon as the application is started and the TCPIP connection is established. The TCPIP connection is established by the [WindTCP_Server](#). The exchange between server and application takes place through a queue mechanism refer to [Queue Programming Interface](#).



This module will give you a live signal for every run. The peak powers and the range of a run are plotted against the frequency in the graph. You can change the parameters while the acquisition is running, the changes will take effect right away. The FFT size selector will switch between 32, 64, 128, 256, 512 and 1024. You can select which trigger channel should be displayed.

4.2.1 Stop the Live Display

The execution of the `Wind Live Display` can be stopped using the button *Exit* or by clicking the window's close symbol.

4.3 Acquis

An acquisition program is provided as part of the delivered LabVIEW source project (*WindAcquis.vi*, found in the appropriate project folder) as well as a Windows executable *Wind Acquis.exe*. This program acquires and stores the acquired data in a TDMS and a `netCDF` file. Both data file will be stored in the selected data directory. The `netCDF` files are stored in a subdirectory structure `YYYY-MM-DD`.

4.3.1 Running Wind Acquis

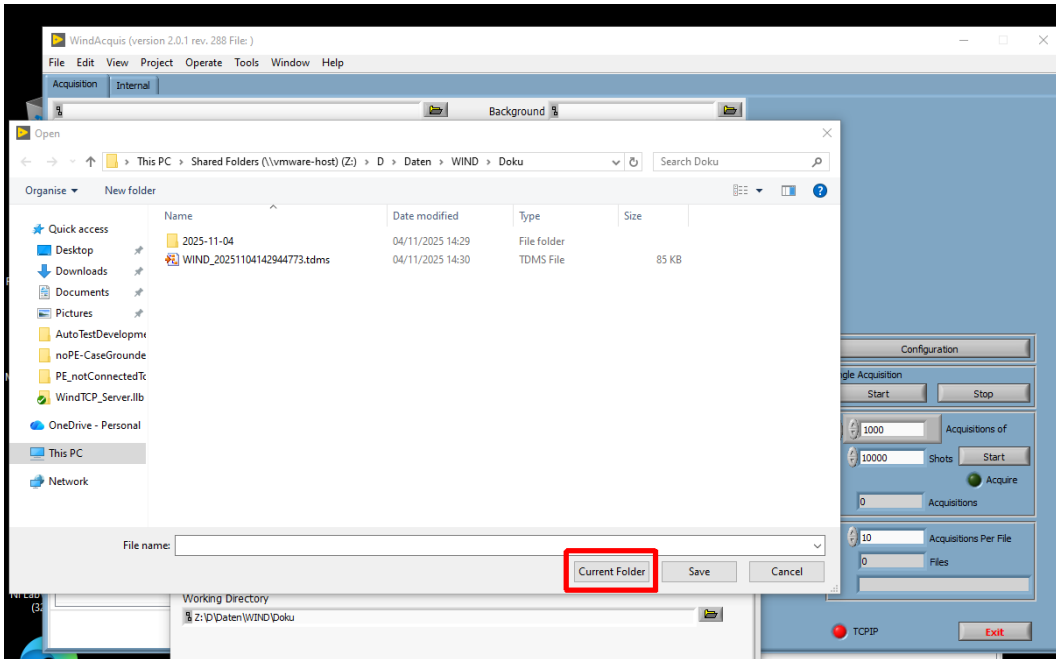
Before running the acquisition program, the initialization file `WIND.ini` must be checked for the correct TCP/IP connection settings. The `WIND.ini` file can be found at the installation path like `C:\Program Files\Licel\Waverider Acquisition\`, if the waverider application was installed. Otherwise (sources only) the `WIND.ini` file is stored at `WIND_PC\WindTCP_Server.llb\WIND.ini`. Please refer to the [Network Setup](#) to correctly configure the Waverider for the TCP/IP operation. The IP address and the `PORT` assigned to the controller during the network setup must be written to `WIND.ini`:

```
[Client]
IP = 10.49.234.234
PORT = 2055
HTTP_Port = 80
```

Once these values are correct the *Wind Acquis* can be started.

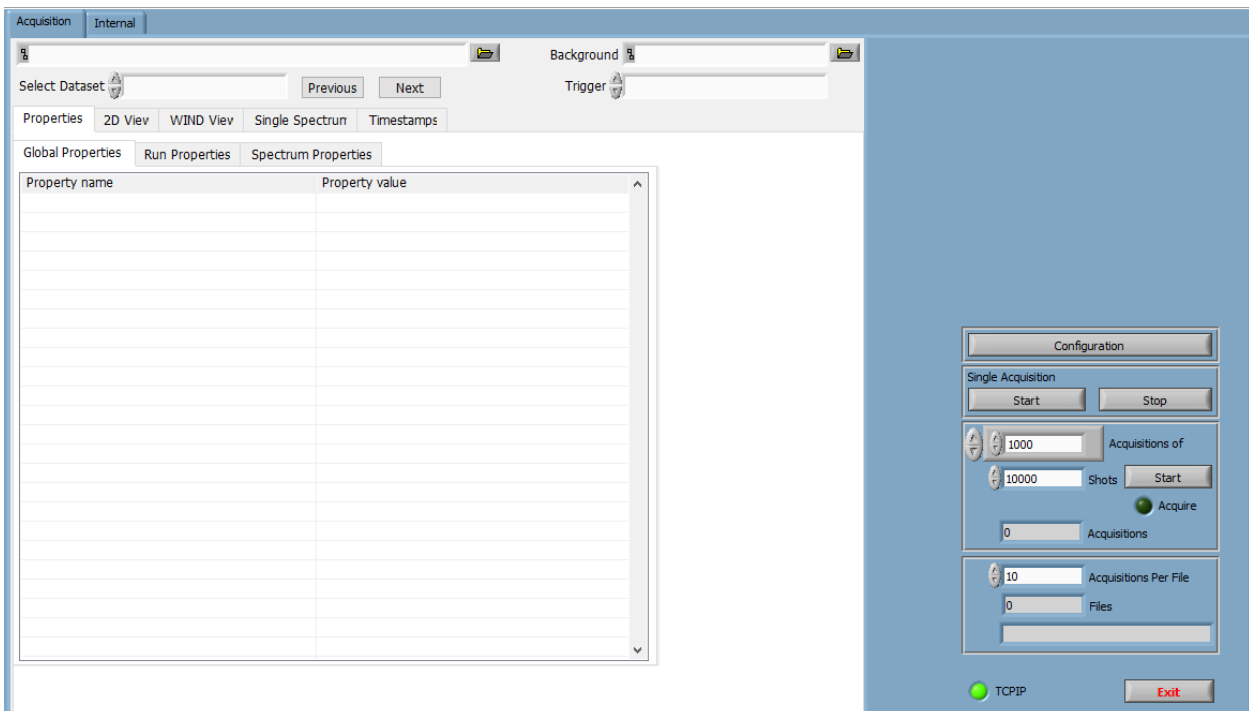
4.3.2 Checking the Data Directory

The data directory is automatically checked at the start of the programs. If the formerly used directory is not valid a directory selection dialog comes up directly after starting *WindAcquis*. There, it is possible to select or create a directory to use to save the acquired data to TDMS files using the button *Current Folder*. Both `tdms` and `netcdf` files will be stored in the selected directory. The `netcdf` data will be stored in a subdirectory with the structure `YYYY-MM-DD`.



4.3.3 Wind Acquis Front Panel

The front panel of *Wind Acquis* consists of mainly two parts: a *view area* on the left and an *acquisition control area* on the right.



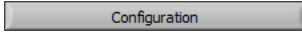
At the most bottom right a TCP/IP indicator and the button *Exit* are located.

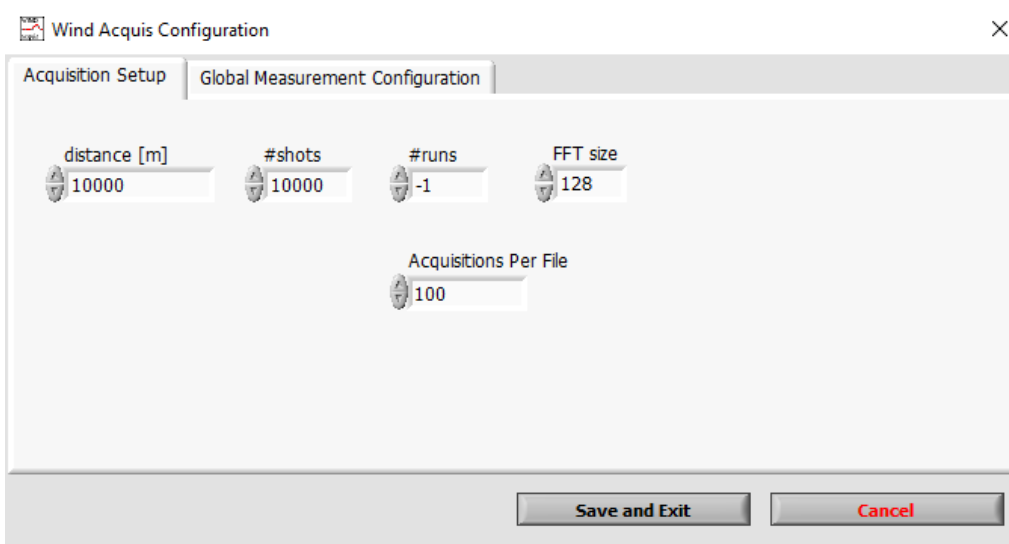


- The LED should always light green during operation. If it changes the color to red the TCP/IP connection has been lost. In the case that this is caused by incorrect entries in the initialization file `WIND.ini` or if build-in TCP/IP connection repair mechanisms fail an error message giving more details will be shown and the program terminates.

- The *Exit* button is used to exit the program.

4.3.4 Configuration

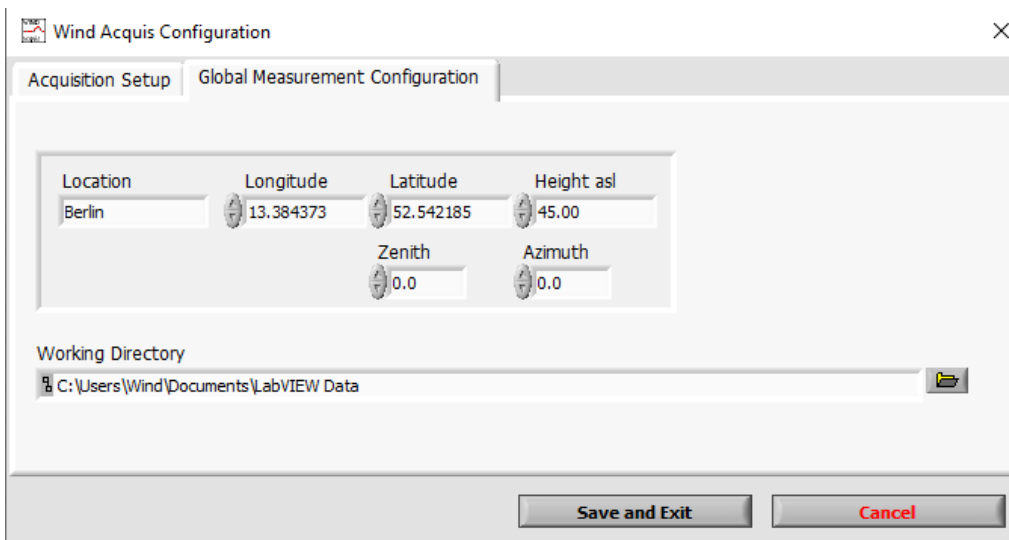
In the *acquisition control area* the button *Configuration*  will open the acquisition and measurement site configuration dialog. The values displayed in the configuration dialog correspond to the current values in use. The first tabulator page *Acquisition Setup* contains the controls to specify the acquisition settings:



The following parameters can be entered:

| | |
|------------------------------|---|
| <i>distance[m]</i> | the distance (range) of the acquisition in meters, |
| <i>#shots</i> | the number of shots to acquire, |
| <i>#runs</i> | the number of runs to acquire (-1 is allowed indication <i>run until Stop is manually pressed</i>), |
| <i>FFT size</i> | the FFT size , |
| <i>Acquisitions Per File</i> | the (maximum) number of acquisitions per file (-1 = unlimited, not recommended). |

The second tabulator page *Global Measurement Configuration* allows to enter measurement site data and the detection angles.



| | |
|--------------------------|---|
| <i>Location</i> | the name of the measurement site / location, |
| <i>Longitude</i> | the corresponding longitude in degrees, |
| <i>Latitude</i> | and latitude in degrees, |
| <i>Hight_asl</i> | the height above sea level in meters, |
| <i>Zenith</i> | the zenith angle, |
| <i>Azimuth</i> | and the azimuth angle, |
| <i>Working Directory</i> | the current data directory for the TDMS data files. |

To check out of the configuration dialog two options are available:

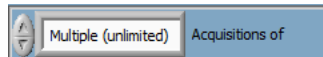


| | |
|----------------------|--|
| <i>Save and Exit</i> | save all parameters to the initialization file for the next program start and exit the configuration dialog, |
| <i>Cancel</i> | reset the parameters to the values when the configuration dialog has been opened. |

4.3.5 On-the-Fly Change of Parameters

Some of the parameters can be changed on-the-fly without saving them to the initialization file. Nevertheless the values will be displayed when opening the configuration dialog!

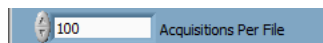
Number of Acquisitions the number of runs (acquisitions) is either -1 (*unlimited* = run until Stop) or a finite positive Number. The *unlimited* setting can be achieved by setting the selector to *Multiple (unlimited)*



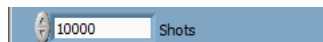
or by explicitly setting the run number to -1:



Acquisitions Per File the *Acquisitions Per File* (-1 = unlimited, not recommended) can be changed at the bottom



Number of Shots the *Number of Shots* can directly be entered, as well:

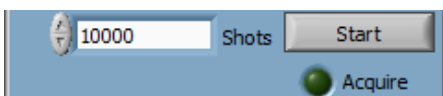


4.3.6 Run an Acquisition

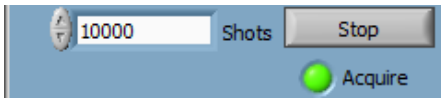
A single acquisition (acquisition series with only one run (acquisition)) is started (and stopped) with the appropriate buttons.



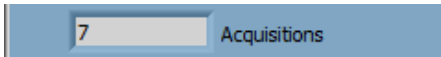
A series of limited or unlimited acquisitions (runs) is started using the *Start* button below:



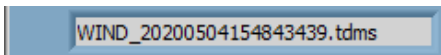
Once a series of acquisitions is running the corresponding LED *Acquire* changes the color to bright green. A *Stop* button is available.



The run (acquisition) counter increases ...



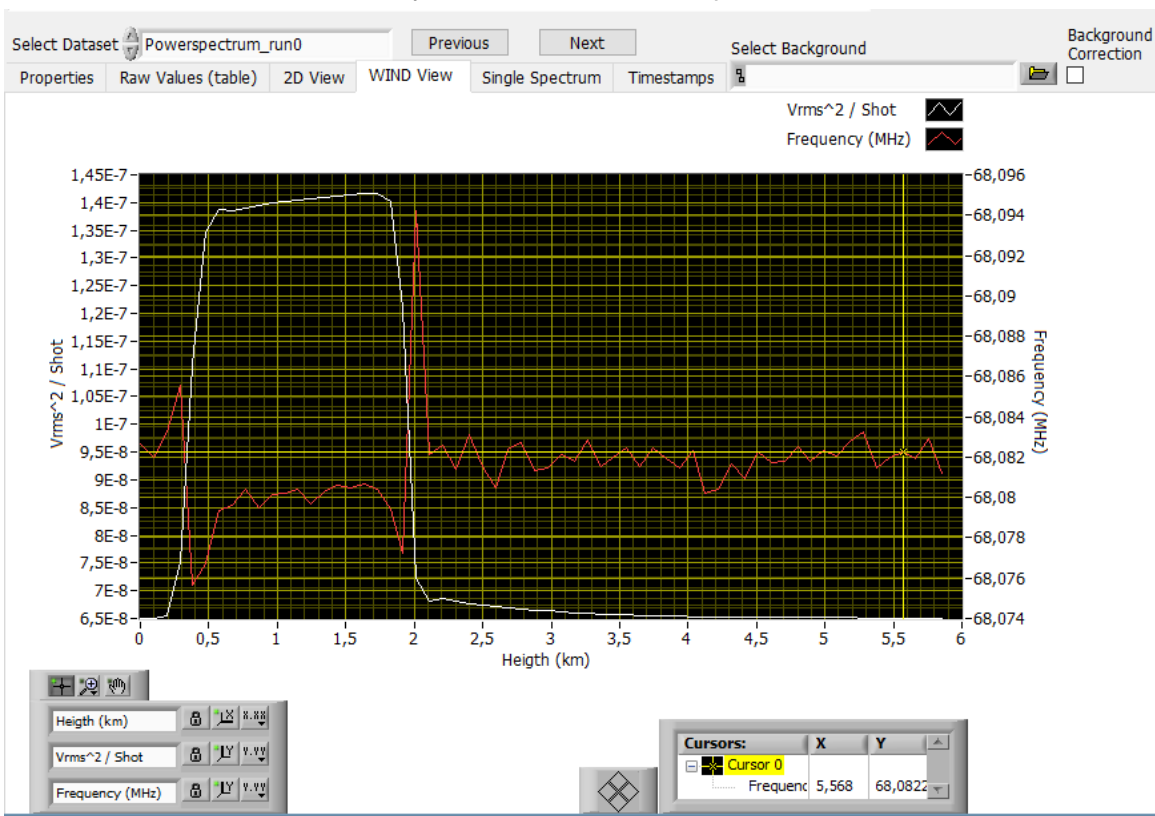
... and a file is created and the file name is displayed:



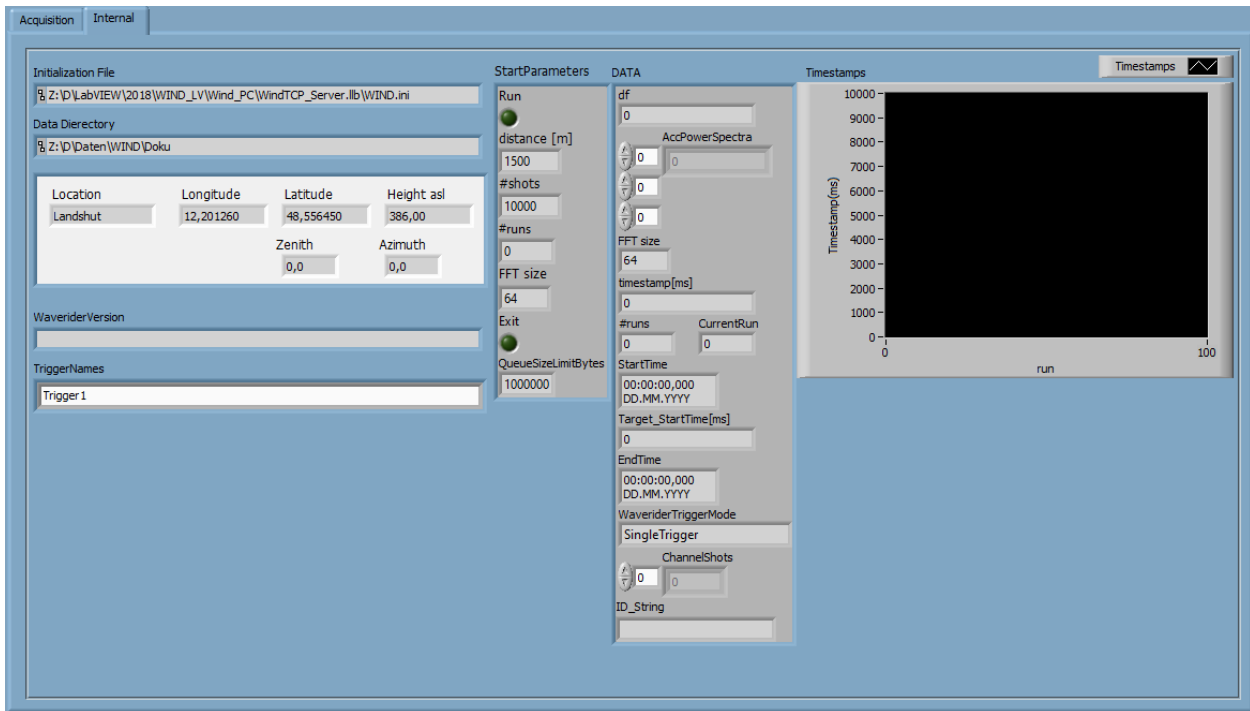
Acquired data is displayed on the left side in the *view area*. There, during an acquisition, the file controls are locked. Once the *Acquisitions Per Files* has been reached a new file will be generated. The file counter increases:



The *View Area* on the left side of the *Wind Acquis* front panel contains the *Wind Viewer* as a plug-in. When no acquisition is running it can be used to display and inspect any acquired wind data file as described in the next section. During a running series of acquisitions the *View Area* is automatically filled and shares the current acquired file with *Wind Acquis*.



On the tabulator page *Internal* some more internal and service information is available:



Initialization File

path of the initialization file,

Data Directory

current data directory, can be changed in the [configuration dialog](#),

Global Parameters

measurement situation parameters, can be changed in the [configuration dialog](#),

StartParameters

communication parameters submitted to the TCP/IP Server while starting a series of acquisitions (for service),

DATA

latest data received by the TCP/IP Server (for service),

Timestamps

relative timestamps of the series of acquisitions,

WaveriderVersion

Waverider version,

TriggerNames

List of available trigger channels.

4.4 Viewer

A data Viewer component (stand-alone) is provided as part of the delivered LabVIEW source project (*WindViewer.vi*, found in the corresponding project folder) as well as a Windows executable *Wind Viewer.exe*. With this reader you can load the TDMS file generated with the [WindAcquis](#) software. It is possible to carry out a background correction. To do this, the background file must be selected and activated with the [Background Correction](#) tick.

4.4.1 Run the Viewer

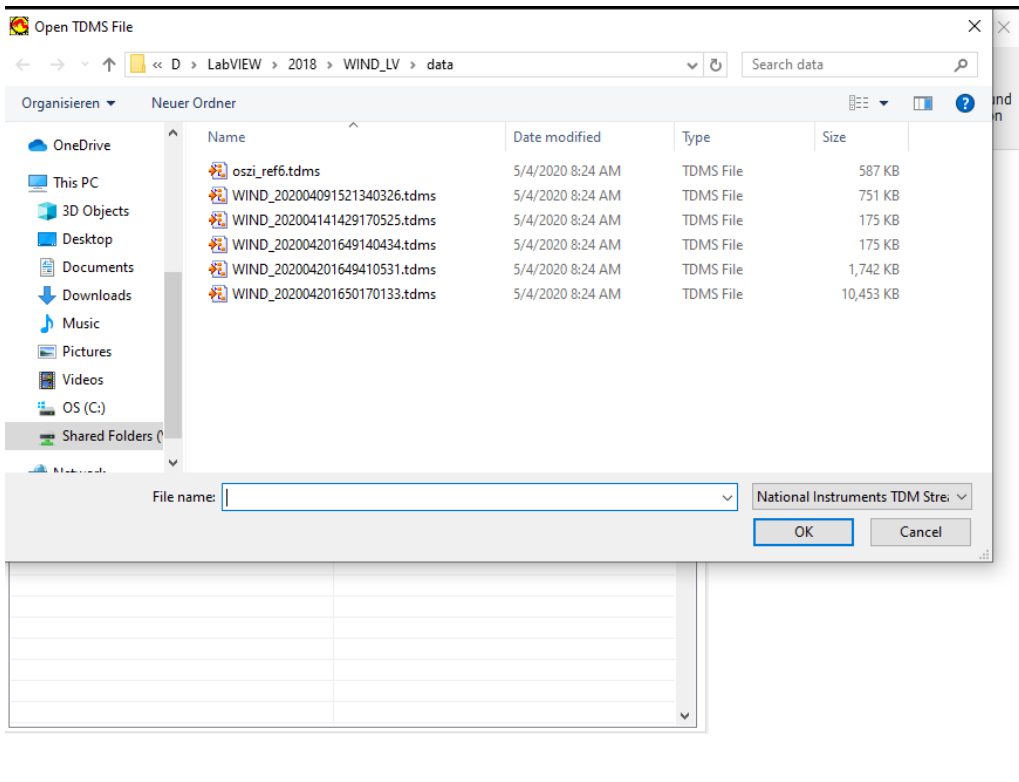
Licel provides two precompiled version of the waverider viewers for Windows OS.

The *Wind Viewer.exe* allows to view the *.tdms* files.

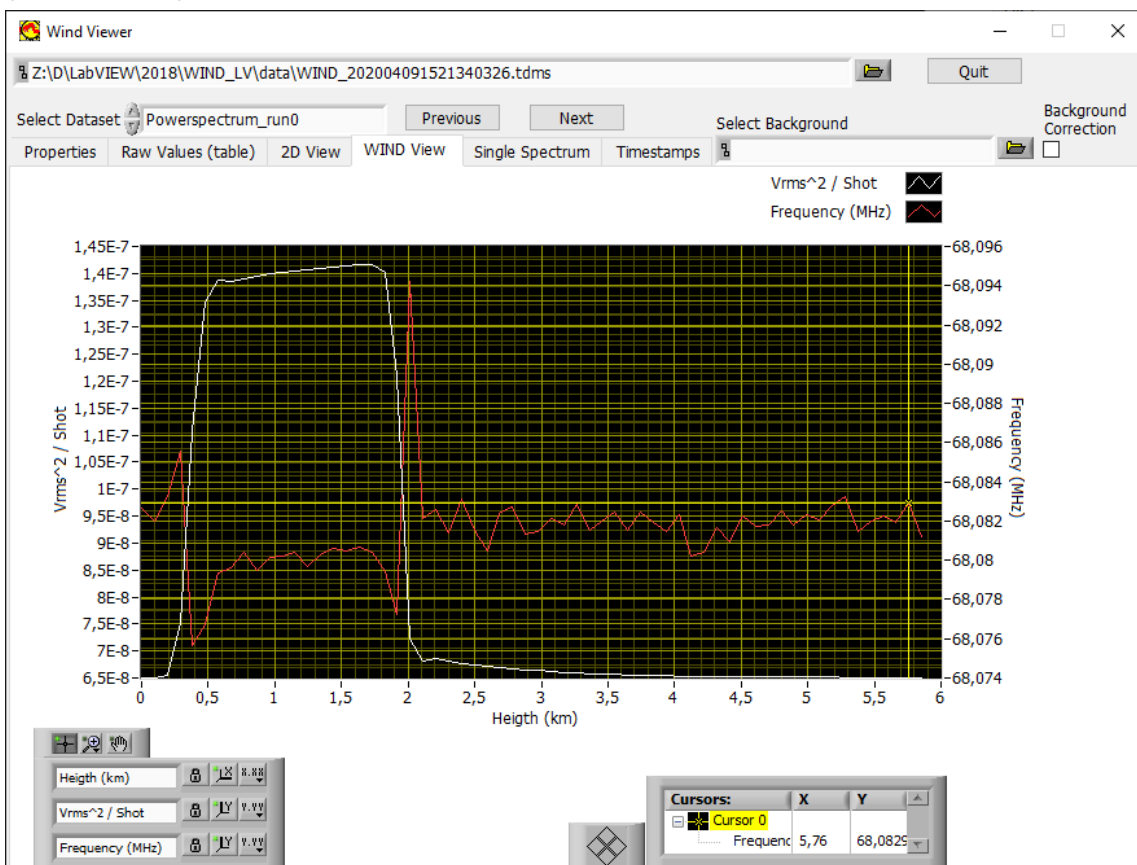
The *Wind netCDF Viewer.exe* allows to view the *.nc* (netCDF) files.

Note that both wind viewer have identical GUI, functionalities and usage, the sole difference is the file format input they support.

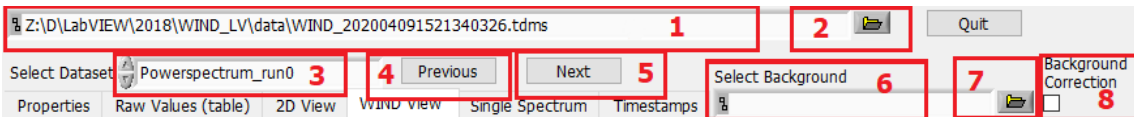
After running the Viewer program a file selection dialog comes up and allows the user to select a Waverider data file:



after clicking the button **OK** the selected file will be loaded into the Waverider data viewer (Wind Viewer).



The data file related control elements are placed at the top of the viewer's front panel:



1. The file path is shown.
2. The browse button allows to open the file selection dialog to change the loaded file.
3. Select the dataset manually.
4. A click on the *Previous* button will load the previous dataset, if the current dataset is the first dataset of the file then the previous file in the directory will be loaded.
5. A click on the *Next* button will load the next dataset, if the current dataset is the last dataset of the file then the next file in the directory will be loaded.
6. Selected background file path is shown.
7. The browse button allows to open the file selection dialog to change the loaded background file.
8. Enable or disable the background correction, if no path is selected the correction is automatically disabled.

4.4.2 View Acquisition Details

It is possible to inspect several details corresponding to the selected dataset. Some of the details are specific for the acquisition series, the runs (acquisitions), or the spectra.

The Global Properties of the acquisition include the acquisition parameters and the global measurement site settings.

[illegible]

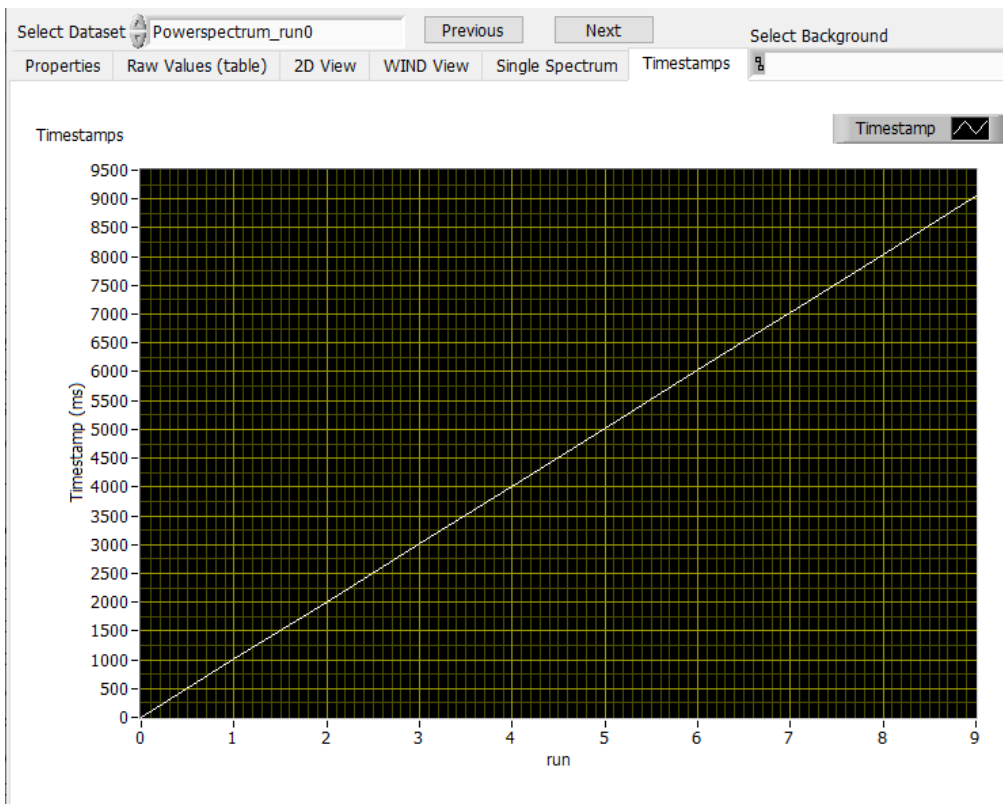
Run Properties contains the timestamp information of the selected run. The difference between Target_Start Time[ms] and Timestamp_device gives an information about the time required for acquiring the selected dataset.

[illegible]

The Waveform information of the spectrum can be looked up in the `Spectrum Properties` Tab.

[illegible]

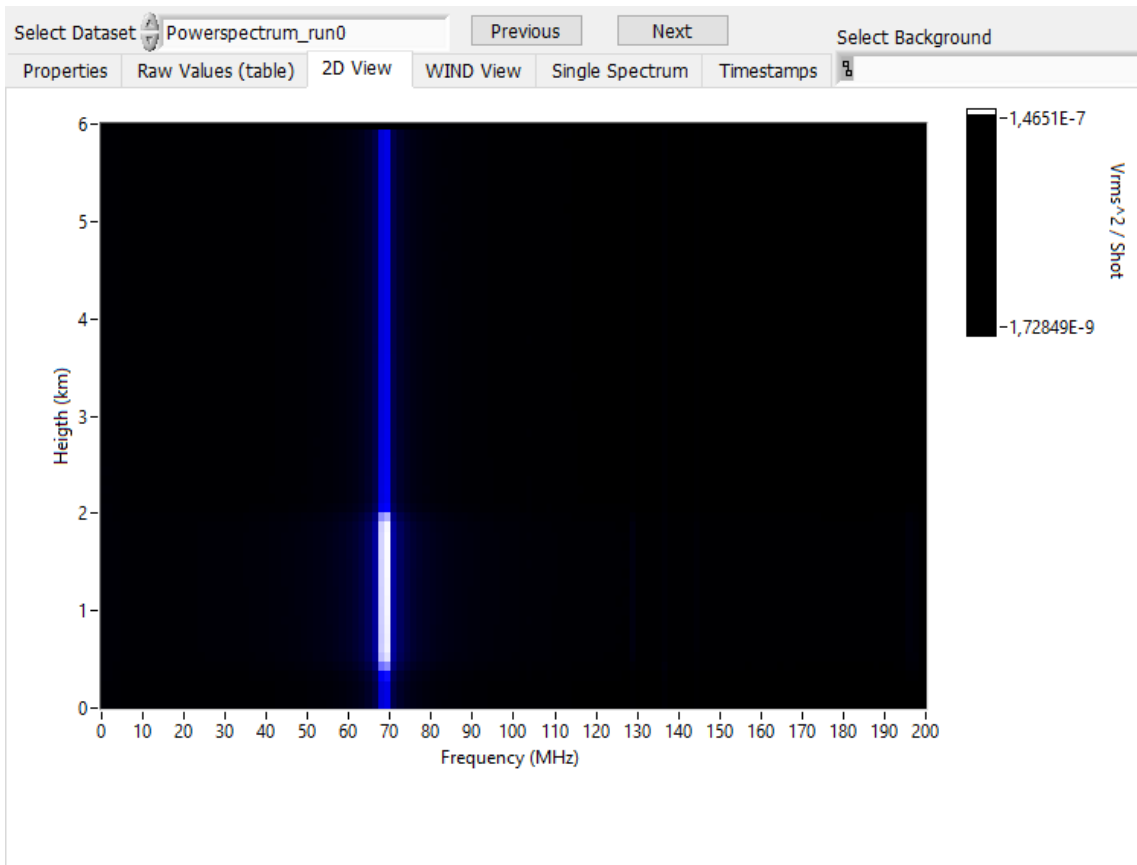
The relative *Timestamps* of all *runs (acquisitions)* of the *acquisition series* consist of a diagram plotting the controller time (in milliseconds with respect to the 1st run) as a function of the run index. It should show a linear behavior if all records have been correctly acquired.



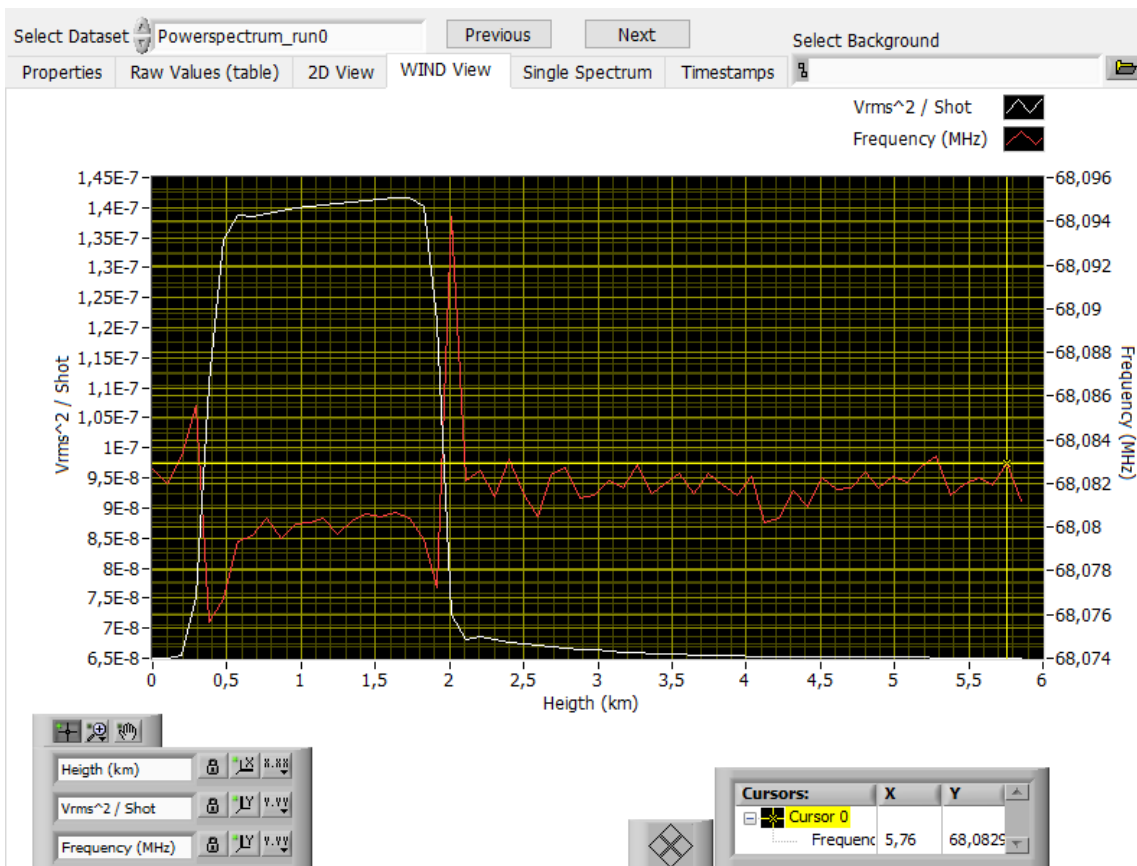
The Raw Values of a run (acquisition) or a spectrum can be inspected at the appropriate tab page.

| Select Dataset Powerspectrum_run0 | | | | | | Previous | Next | Select Background |
|-----------------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|----------|------|-------------------|
| Properties | Raw Values (table) | 2D View | WIND View | Single Spectrum | Timestamps | | | |
| Powerspectrum_run0 | Powerspectrum_run0 | Powerspectrum_run0 | Powerspectrum_run0 | Powerspectrum_run0 | Powerspectrum_run0 | | | |
| Spectrum_0 | Spectrum_1 | Spectrum_2 | Spectrum_3 | Spectrum_4 | Spectrum_5 | | | |
| 607,078530E+6 | 606,691873E+6 | 606,874991E+6 | 603,639323E+6 | 591,770702E+6 | 587,567777E+6 | | | |
| 8,588050E+6 | 8,606828E+6 | 8,605310E+6 | 9,046458E+6 | 9,686877E+6 | 9,966597E+6 | | | |
| 7,233042E+6 | 7,205862E+6 | 7,245368E+6 | 7,651395E+6 | 8,330448E+6 | 8,758638E+6 | | | |
| 6,711214E+6 | 6,715591E+6 | 6,759403E+6 | 7,040922E+6 | 7,737281E+6 | 8,378685E+6 | | | |
| 6,396810E+6 | 6,439120E+6 | 6,384893E+6 | 6,798849E+6 | 7,454403E+6 | 8,204017E+6 | | | |
| 6,386673E+6 | 6,332991E+6 | 6,341499E+6 | 6,825418E+6 | 7,830605E+6 | 8,335591E+6 | | | |
| 6,138071E+6 | 6,064128E+6 | 6,129732E+6 | 6,536120E+6 | 7,351829E+6 | 7,905184E+6 | | | |
| 6,040756E+6 | 6,022108E+6 | 6,024327E+6 | 6,445013E+6 | 7,204782E+6 | 7,882778E+6 | | | |
| 5,947984E+6 | 5,991198E+6 | 6,017076E+6 | 6,375592E+6 | 7,206398E+6 | 7,819588E+6 | | | |
| 5,954848E+6 | 5,967183E+6 | 5,985940E+6 | 6,405148E+6 | 7,226726E+6 | 7,859658E+6 | | | |
| 5,926938E+6 | 5,970194E+6 | 5,967412E+6 | 6,368105E+6 | 7,319412E+6 | 8,005204E+6 | | | |
| 5,891482E+6 | 5,935581E+6 | 5,978680E+6 | 6,396143E+6 | 7,337028E+6 | 8,001706E+6 | | | |
| 5,947619E+6 | 5,968182E+6 | 5,969559E+6 | 6,417615E+6 | 7,360359E+6 | 8,110118E+6 | | | |
| 6,008687E+6 | 6,010287E+6 | 6,048444E+6 | 6,498898E+6 | 7,491487E+6 | 8,199751E+6 | | | |
| 5,965287E+6 | 6,038908E+6 | 6,060694E+6 | 6,477475E+6 | 7,410813E+6 | 8,360675E+6 | | | |
| 6,138644E+6 | 6,151289E+6 | 6,253354E+6 | 6,691172E+6 | 7,705615E+6 | 8,524093E+6 | | | |
| 6,554571E+6 | 6,535778E+6 | 6,544825E+6 | 7,023144E+6 | 8,041379E+6 | 8,938158E+6 | | | |
| 6,282588E+6 | 6,349924E+6 | 6,330288E+6 | 6,856992E+6 | 7,997934E+6 | 8,881619E+6 | | | |
| 6,148530E+6 | 6,143233E+6 | 6,227546E+6 | 6,773722E+6 | 7,935200E+6 | 8,962104E+6 | | | |
| 6,115635E+6 | 6,115767E+6 | 6,146687E+6 | 6,710356E+6 | 8,006579E+6 | 9,022216E+6 | | | |
| 6,148435E+6 | 6,207626E+6 | 6,214256E+6 | 6,773222E+6 | 8,164915E+6 | 9,338792E+6 | | | |
| 6,254648E+6 | 6,174093E+6 | 6,259387E+6 | 6,951350E+6 | 8,271981E+6 | 9,490475E+6 | | | |
| 6,294485E+6 | 6,305164E+6 | 6,341118E+6 | 6,977165E+6 | 8,548466E+6 | 9,790435E+6 | | | |
| 6,457586E+6 | 6,388847E+6 | 6,456012E+6 | 7,171543E+6 | 8,756939E+6 | 10,097779E+6 | | | |



The scaled data of a run (acquisition) or a spectrum can be seen in the *2D View* where the power amplitude is plotted in a color magnitude against the height and frequency.



The peak power and the heights of a run (acquisition) are plotted against the frequency in the *WIND View* graph.



4.4.3 Stop the Viewer

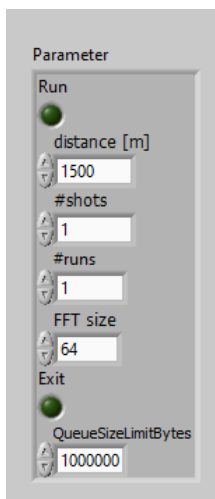
The execution of the *Wind Viewer* can be stopped using the button *Quit*  (leaves the front panel open for the next start) or by clicking the window's close symbol  at the top right (this will close the front panel after execution).

Chapter 5

Queue Programming Interface

In addition to the direct TCPIP interface for the controller an easy accessible queue interface is available. The interface has two queues the `WIND_CMD` queue which takes a parameter cluster containing the required information for controlling an acquisition and the `WIND_DATA` providing the raw data and auxiliary data to convert the raw data into physical values. All applications described above use this interface. Both queues are handled by the TCPIP Server which translates the queues into TCPIP socket handling.

The `WIND_CMD` Queue The Parameter cluster is shown below:



Setting the `Run` boolean causes a `START` command issued. Resetting it causes a `STOP` command. The desired distance together with the `FFTSIZE` is translated into the required `number of FFT`. The `#shots` define the number of `shots` per run. And the `#runs` define how many `runs` should be acquired. Setting the `Exit` boolean causes a termination of the TCPIP Server. The `QueueSizeLimitBytes` define the limit of the queue size in bytes.

5.1 TCPIP Server

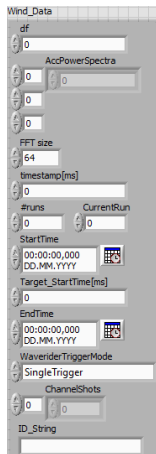
The TCPIP server is usually launched as a background service by calling the `WIND Acquisition Start TCPIP Server.vi`. After start the TCPIP Server reads the `WIND.ini` file. The user should enter the corresponding IP address in the client section.

```
[Client]
IP = 192.168.69.58
```

PORT = 2055

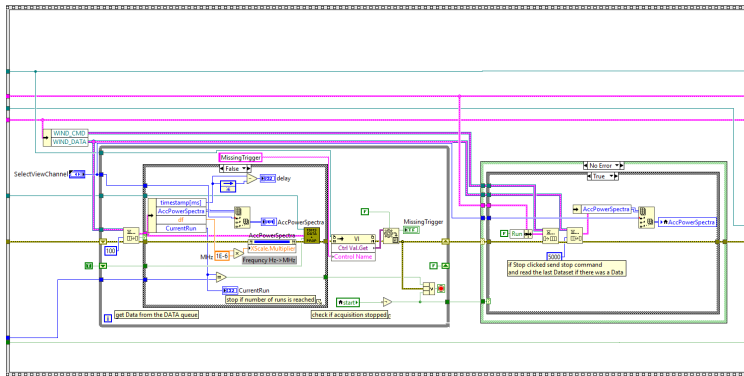
With this setting the TCPIP server would connect to the controller at 192.168.69.58 and open two sockets the command socket at 2055 and the push data socket 2056

The TCPIP server then waits till the `Parameter` cluster is send over the `WIND_CMD` queue and executes then the required actions on the command socket and waits for data on the push data socket. Once data arrives there it sends the data over the `WIND_DATA` queue back to client program. The `WIND_DATA`

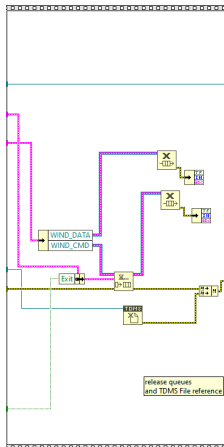
A screenshot of a debugger window titled 'Wind_Data'. It shows a list of fields with their values: 'df' is 0; 'AccPowerSpectra' is an array of 0s; 'FFT size' is 64; 'timestamp[ms]' is 0; '#runs' is 0; 'CurrentRun' is 0; 'StartTime' is '00:00:00.000' with a date picker; 'Target_StartTime[ms]' is 0; 'EndTime' is '00:00:00.000' with a date picker; 'WaveriderTriggerMode' is 'SingleTrigger'; 'ChannelShots' is 0; and 'ID_String' is an empty text box.

contains the

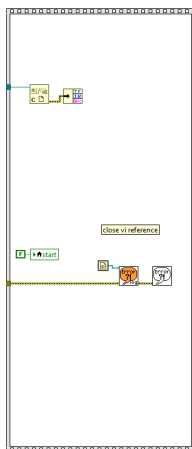
- **df** - the frequency increment of the power spectra in MHz.
- **AccPowerSpectra** - accumulated power spectra per channel. For each range bin there is one FFT and they are following each other, knowing the FFT size this array can be converted to a 2D array.
- **FFTsize** - number of points for the FFT, the power spectra contains only half of this number points.
- **timestamp** - time stamp of data (compare MSEC? on the target) this are the milliseconds since Waveride controller start when the data became available in the Waverider controller. To convert the timestamp into a real world time compute $(\text{timestamp} - \text{Target_StartTime}) * 0.001 + \text{StartTime}$.
- **#runs** - selected number of runs
- **CurrentRun** - indicates the already acquired number of runs
- **PC Start Time** - PC time when the start command has been issued
- **Target_StartTime[ms]** - Start time of the target in [ms], this are the Waverider milliseconds since start at the moment when the start command has been issued.
- **WaveriderTriggerMode** - selected Trigger mode
- **ChannelShots** - number of shots for each channel
- **ID_String** - Firmware ID of the Waverider



As data arrives it is written into the TDMS file and if the user stops the loop the data acquisition is aborted. After the acquisition cycle the run variable is set to false issuing a **STOP**. Only the selected Channel will be displayed.



The TDMS file is closed and the queues are released. If the constant `Exit Server vi` was true, the server vi will be closed too.



Then one closes the VI reference to the server and shows errors if they occurred.

Chapter 6

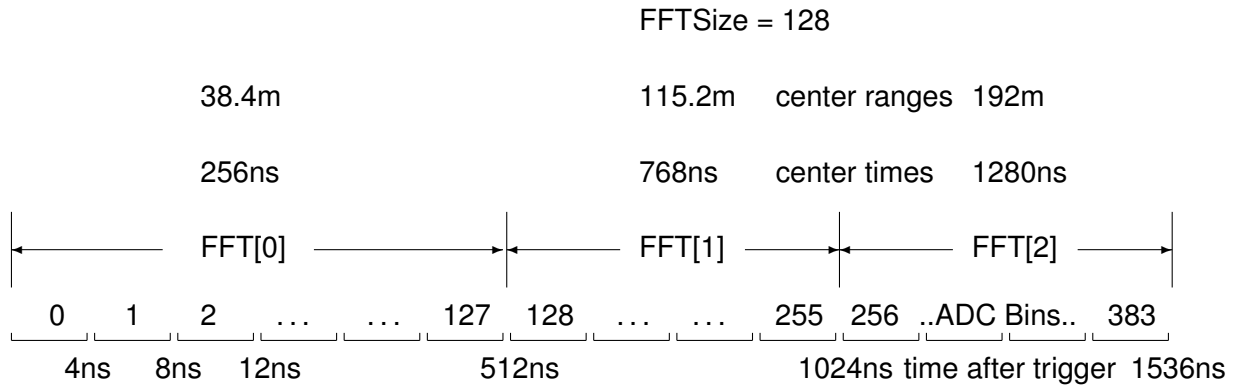
Data transfer - Low Level Description

6.1 Operation principle

6.1.1 FFTSize

The FFTsize defines how many original ADC samples go into one FFT. The ADC stream after the trigger is cut into chunks defined by the FFTSize. Currently 32, 64, 128, 256, 512 and 1024 are the possible chunk sizes.

The following details the operational characteristics corresponding to an FFT chunk size of 128.



For a 250MHz sampling frequency with 128 Samples the spectral resolution of the power spectra is

$$\Delta f = \frac{250\text{MHz}}{128} = 1.953125\text{MHz} \quad (6.1)$$

6.1.2 Distance

The spatial resolution of a single FFT is given by:

$$\text{spatial resolution in } m = \frac{c \text{ in } (m \times s^{-1})}{2 \times \text{spectral resolution } MHz} \quad (6.2)$$

To know how many FFT needs to be computed for a specific distance trace one can simply use the formula:

$$\text{Number of FFT} = \frac{\text{Desired distance in } m}{\text{spatial resolution in } m} \quad (6.3)$$

For example to acquire a 15Km distance trace with a FFT size of 128 one would calculate the number of FFTs as follows:

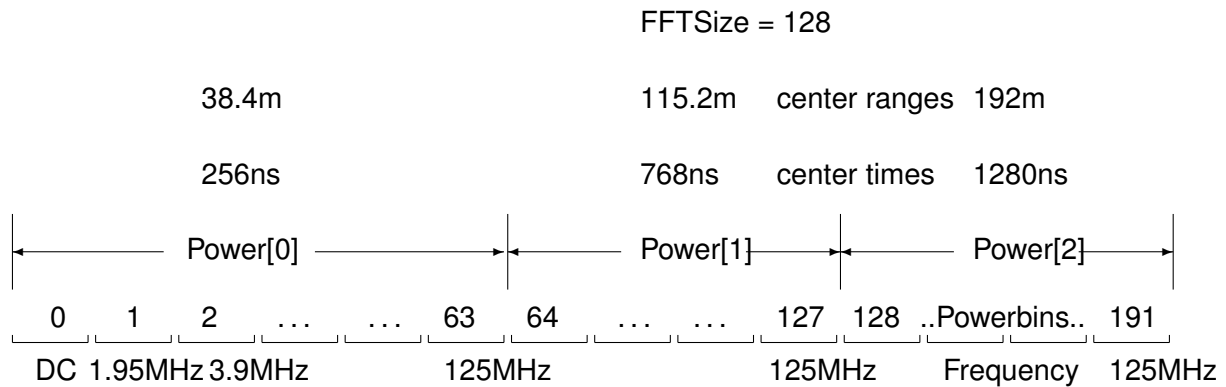
$$\text{spatial resolution in } m = \frac{299792458 (m \times s^{-1})}{2 * 1.953125 MHz} = 76.8m \quad (6.4)$$

$$\text{Number of FFT} = \frac{15000 m}{76.8 m} = 195,3125 \quad (6.5)$$

Note that the number of FFTs needs to be rounded down to the next integer, so for this example one would need to compute 195 FFTs. which would correspond to a distance of 14,976 km.

6.1.3 Power spectra

The result of the FFT is a power spectra. The number of elements in the power spectra is half of the number the FFTSize chunks. The system will then work with $195 * 128 / 2 = 12480$ power spectra bins.



6.1.4 Shots

Each trigger event (laser shot) results in a [power spectra](#) vector. The power spectra are not individually stored but are accumulated.

6.2 TCPIP Communication

The communication between the PC and the controller is implemented as TCPIP sockets.

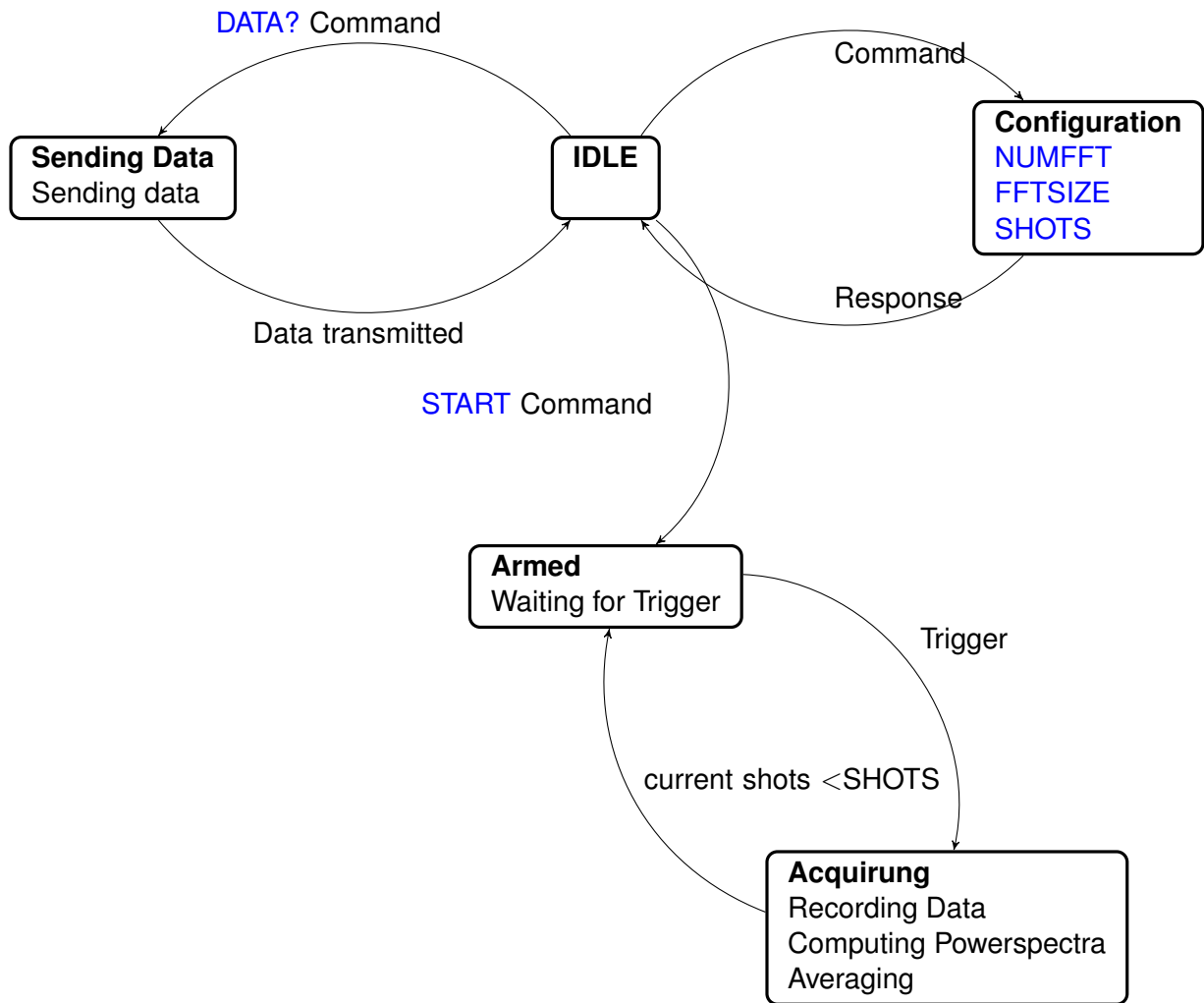
There is the command socket. The communication here is bidirectional. The PC is sending a command and the controller is answering with a response.

The acquired data is sent on request over this socket

Before starting an acquisition the acquisition parameters should be set. After sending the [START](#) command the controller will change into the armed state. Once a valid trigger is received the acquisition and later the power spectra computation runs. After each acquisition the system checks if the desired number of shots has been reached.

If the desired number of shots has not been reached it changes again into the armed state, and responds to the [DATAAVAILABLE?](#) command with False. If the desired number of shots has been reached it returns into the idle state and waits for data request. In this state the waverider responds to the [DATAAVAILABLE?](#) command with True.

The `Wind Acquisition_TCPIP.vi` demonstrates how this works. By implementing a queue interface this VI frees the user from handling these communication details. See the Queue Interface for more explanations.



6.3 Data Package format

The data is 8 byte aligned and structured as follow:

| | | |
|---|---|-----------------|
| start = 0x00000000 [4 bytes] | } | Response Header |
| data id = 0x00D000C [4 bytes] | | |
| number of bytes [4 bytes little endian] | | |
| padding = 0x00000000 [4 bytes] | } | Payload |
| timestamp [8 bytes little endian] | | |
| padding = 0x00000000 [8 bytes] | | |
| raw data [8 bytes little endian] | | |

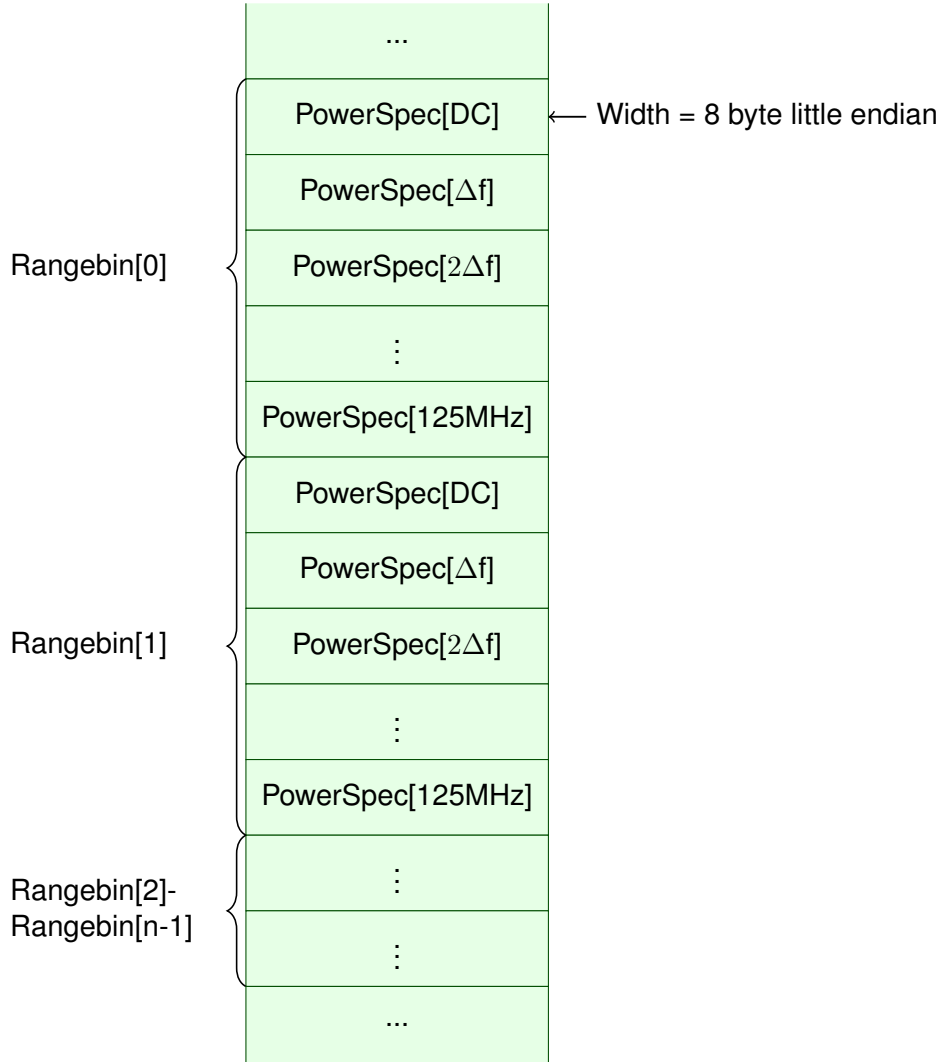
- The number of bytes for the Request data package is little endian.
- The Timestamp is transmitted as little endian.
- The raw data is little endian.

The transmitted raw data size is always equal to 32768 elements X 8 bytes. The previously set waverider configuration, specifically the number of FFTs and fft size, determines how many data points corresponds to the actual Powerspectra. the rest of the elements are just zero padding.

For example if the system is configured to acquire 195 FFTs with a size of 128, then the number of powerspectra entries is $195 * (128/2) = 12480$. The 20288 remaining elements ($32768 - 12480$) will be padded with zeros and included in the transmitted data. Each powerspectra entry is an unsigned 64 bit integer.

6.4 Accumulated Power Spectra

The data contains the accumulated power spectra as unsigned 64 bit integers. Its a 2D matrix where the columns are for the different frequencies between DC and 125MHz and the rows are the different range bins. This matrix gets then flattened into a vector which is described below.



6.5 Raw Data to Physical Value Conversion

The conversion starts with a normalization with the shot number. After this step the analog data shows the mean power spectra bit values. The stored raw values equals the summation of FFT^2 . The analog data needs then to be scaled by the Power spectra value. The internal values are fixed floating point values between 8192 and -8191.

$$FFT^2 = \frac{AccPowSpectra}{Number\ of\ shots * (2^{ADCBits-1})^2}$$

For a 14Bit ADC and Number of shots equal to 100 this means

$$FFT^2 = \frac{AccPowSpectra}{100 * (8192)^2}$$

The FFT^2 can be converted to the Unit V_{rms}^2 by using the `FFT size` Parameter.

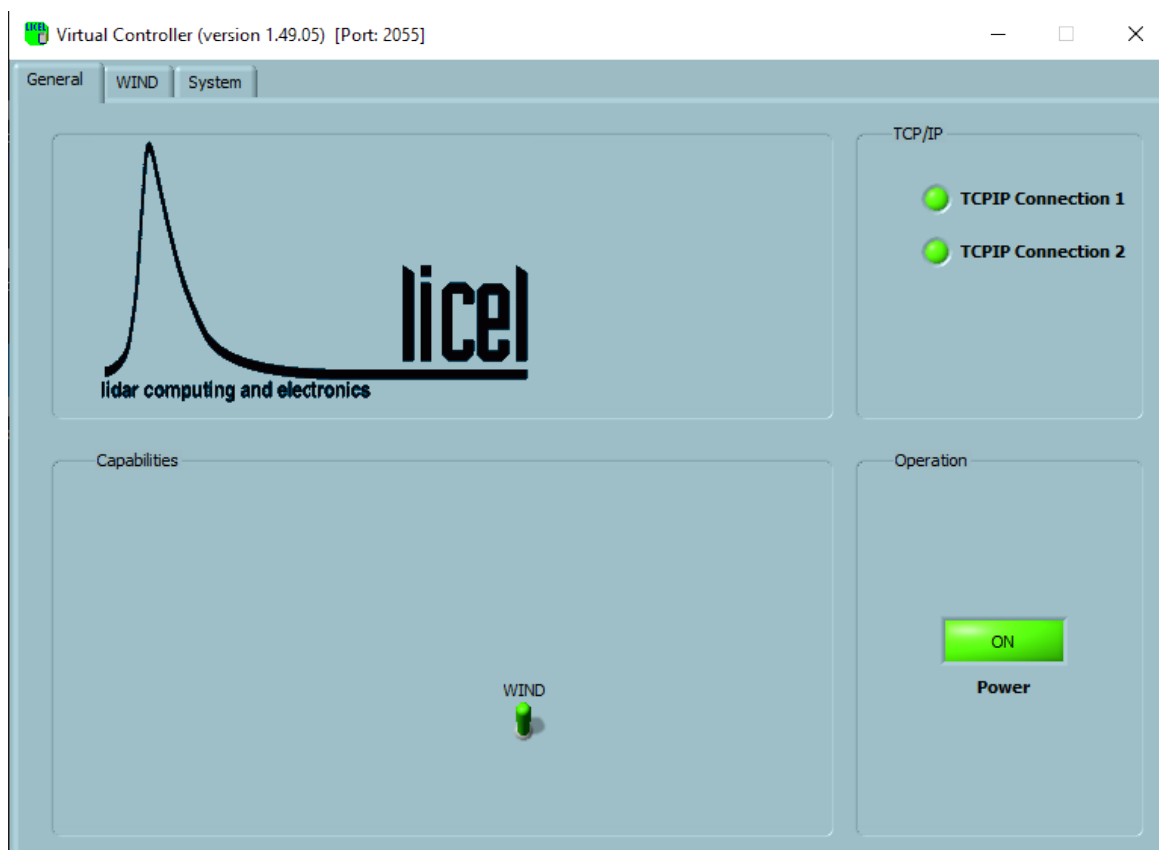
$$V_{rms}^2 = \frac{FFT^2}{(FFT\ size)^2}$$

Chapter 7

Simulation

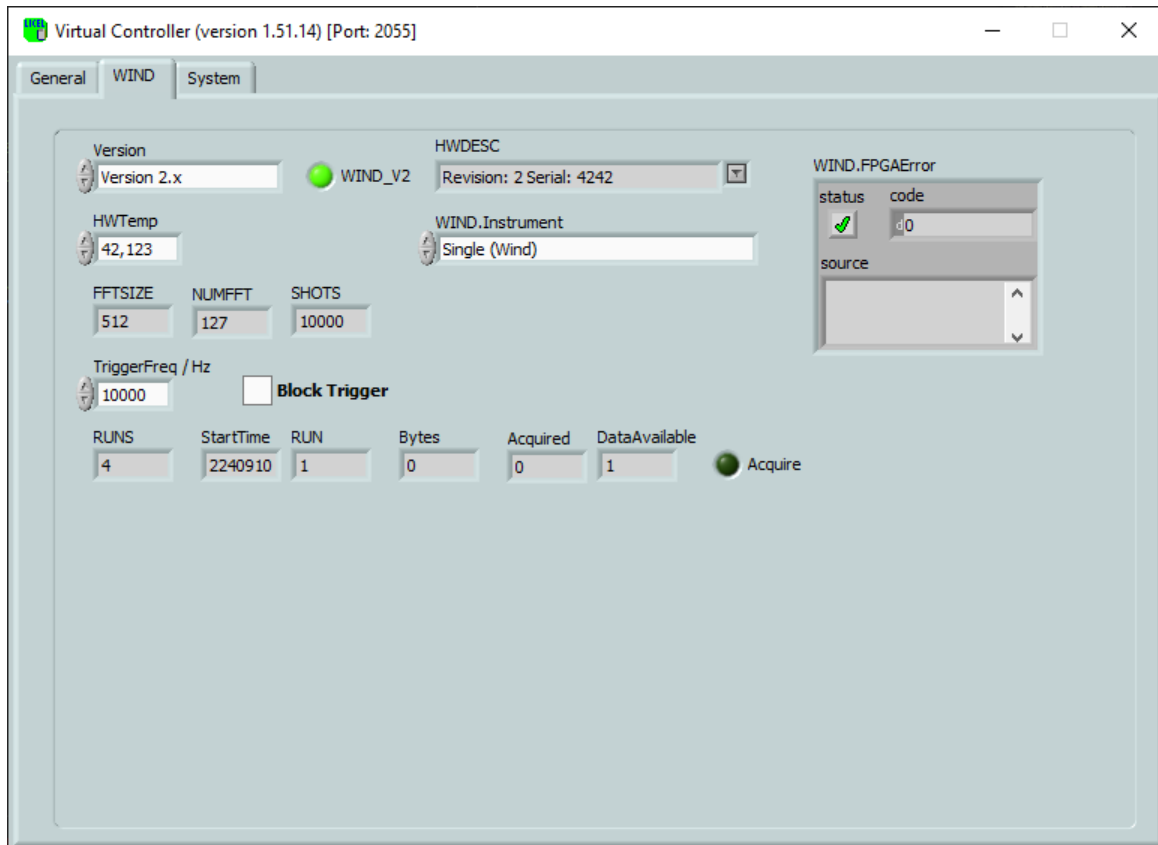
All software module can be also tested in Simulation mode. One needs a Virtual Controller. This is part of the executables in <http://licel.com/download/ethernet/WaveriderLVInstaller.zip>

It should start up as



Please note the green `Wind` switch, indicating the support for the Wind commands. The general capabilities are described at in the [Licel Ethernet Controller Installation and Reference Manual](#) chapter 8.

In addition to the there described capabilities is one tab with the Wind support.



To run the simulation start first the `Virtual Controller.exe`. Enter `127.0.0.1` as the IP number in the `Wind.ini` in the `[Client]` section and start the desired software module. Once it runs the TCP/IP indicators at the entrance panel should become green and the software should run with simulated data.

Note that the simulation does support both the 250MHz and 400MHz waverider hardware. Depending on the Hardware to be simulated, the user must set the Hardware version type in the `Wind` panel.

- Waverider 250 MHz are to be simulated with `Version 2.X`.
- Waverider 400 MHz are to be simulated with `Version 1.X`.

Chapter 8

Specifications

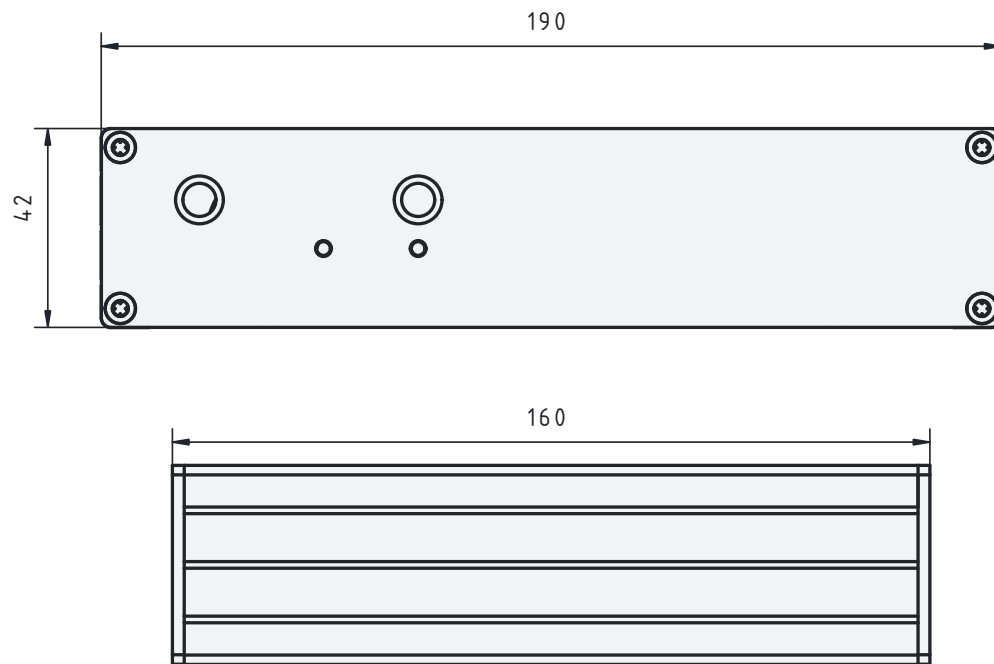
Analog acquisition:

| | |
|---------------------------|---|
| Signal input range: | +/- 1V |
| A/D Resolution: | 14Bit |
| Impedance | 50 Ω |
| Sampling rate: | 250MSamples/second. |
| Lidar spatial resolution: | 19.2, 38.4, 76.8, 153.6, 307.2, 614.4m. |
| Bandwidth: | AC- 125 MHz. |
| SNR: | 58db |
| trace length max: | 39.32km |
| max. shots per run | $4 * 10^9$ |

| | |
|-----------------|---------------|
| Trigger input: | |
| Trigger inputs: | 1 |
| Impedance: | 20 k Ω |
| V_{IH} : | 2V |
| V_{IL} : | 0.8V |
| V_{max} : | 3.6V |

| | |
|-------------|-------|
| Dimensions: | |
| Width: | 190mm |
| Height: | 42mm |
| Depth: | 160mm |

8.1 Mechanical Dimensions



Chapter 9

Appendices

9.1 TCP/IP Command List and Syntax

The Communication with the waverider follows the standard client-server model, where the waverider system acts as the server.

the Waverider supports two main command type:

- GET requests to retrieve information about the system as well as data.
- SET requests to configure the system parameters.

For each request the waverider will respond with a `string`

9.1.1 GET request

The GET request is 16 bytes wide, Big endian, constructed as follows:

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|-----|------|------|------|------|------|------|------|------|
| 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x0D | 0x00 | CMD | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |
|------|------|------|------|------|------|------|-----|------|------|------|------|------|------|------|------|

- The first 5 bytes fields equals `0x00` and serve the purpose of padding.
- Byte field number 6 is a Hardcoded `0x0D`.
- Byte field number 7 equals `0x00`
- Byte field number 8 hold the hexadecimal value of the request `CMD` see list below.
- The last 8 bytes fields equals `0x00` and serve the purpose of padding.

For the GET request it is sufficient to replace the `CMD` field with the desired command Hex code listed in [supported GET commands](#).

9.1.2 SET request

The SET request is 12 bytes wide, Big endian, constructed as follows:

| | | | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|-----|-----------------|--|--|--|--|--|--|--|
| 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x0D | 0x00 | CMD | N bytes payload | | | | | | | |
|------|------|------|------|------|------|------|-----|-----------------|--|--|--|--|--|--|--|

- The first 5 bytes fields equals `0x00` and serve the purpose of padding.
- Byte field number 6 is a Hardcoded `0x0D`.
- Byte field number 7 equals `0x00`

- Byte field number 8 hold the hexadecimal value of the request `CMD` see list below.
- The last 4 bytes fields holds the parameter value to be set.

For the SET request one must replace the `CMD` field with the desired command Hex code listed in [supported SET commands](#). and add fill the payload field with the desired parameter value.

9.1.3 Request response

The waverider will replies to each GET/SET request as follows:

| | | |
|----------------|----------------------|---------|
| 8 bytes header | 4 bytes payload size | payload |
|----------------|----------------------|---------|

- The first 8 bytes contains header information.
- Byte field number 9 ... 12 contains the payload size.
- Payload is **Big endian** ASCII string and ends with `<CRLF>`.

Note that the `DATA?` command is an exception and returns the payload size as well as the power-spectra data as little endian, refer to [DataFormat](#) section for more details.

9.1.4 Command list

This section lists and describes the TCP/IP command syntax for Licel TCP/IP WindV2. If the controller detects an unknown command it will return the string `<command> unknown command` back to the caller where `<command>` is the command originally sent.

The following commands are available dependent on the Waverider controller you ordered.

Supported GET command

| Short | WaveriderV2 Hex Value |
|-----------------------------|-----------------------|
| <code>START</code> | 0x04 |
| <code>DATAAVAILABLE?</code> | 0x05 |
| <code>SHOTS?</code> | 0x06 |
| <code>CURRENTSHOTS?</code> | 0x09 |
| <code>*IDN?</code> | 0x0D |
| <code>DATA?</code> | 0x04 |
| <code>MSEC?</code> | 0x0E |
| <code>HWDESC?</code> | 0x15 |
| <code>CAP?</code> | 0x16 |
| <code>NUMFFT?</code> | 0x18 |
| <code>FFTSIZE?</code> | 0x19 |

Supported SET command

| Short | WaveriderV2 Hex Value |
|-----------------------|-----------------------|
| <code>SHOTS</code> | 0x01 |
| <code>FFTSIZE</code> | 0x02 |
| <code>NUMFFT</code> | 0x03 |
| <code>DHCP</code> | 0x11 |
| <code>TCPIP</code> | 0x12 |
| <code>PASSWORD</code> | 0x26 |

CAP? – (GET)

Requests the control capabilities of the waverider.

The waverider response is `CAP : WIND`

DATA? – (GET)

Request the data from the waverider.

The acquisition must have been started beforehand with the [START](#) command, and data must be available, see the [DATAAVAILABLE?](#) command.

The returned payload will be structured as follows:

| |
|----------------------|
| 4 bytes zero padding |
| 8 bytes Timestamp |
| 8 bytes zero padding |
| 256K bytes data |

- The first 4 bytes are just zero padding. they must be discarded.
- The next 8 bytes represents the timestamp in milliseconds since the start of the waverider.
- The next 8 bytes are just zero padding. they must be discarded.
- The transmitted data size is always 256K.

Depending on the waverider configuration, the previously set [FFTSIZE](#) and [NUMFFT](#) determines how many data points corresponds to the actual Powerspectra. refer to [Data Low Level](#) section for more details about the data format.

DATAAVAILABLE? – (GET)

Queries if data is available to be retrieved from the waverider.

If no data is available the waverider replies has a payload size of 0 bytes.

If data is available the waverider replies has a payload size of value 1.

IDN? – (GET)*IDENTIFICATION?**

Query the waverider identity and firmware revision. The reply from the waverider is e.g.

`Wind_v2.15.11.2024`

MILLISEC? – (GET)**MSEC?**

Requests the millisecond timer value of the waverider. The reply is

`MILLISEC: time`

where `time` is a number with the milliseconds since the start of the waverider.

PASSWORD – (SET)

Changes the password of the waverider.

What follows is the **Payload** description that must be sent along with the `PASS` command in order to change the password.

PASS <"Old Password"> <"New Password"> <"New Password">

PASSWORD <"Old Password"> <"New Password"> <"New Password">

The user needs to enter the old password and then the new password twice. The default password is "Administrator". The password will be reset to this if a [hardware reset](#) is executed on the waverider.

For example

```
PASS "Administrator" "MyPassword" "MyPassword"
```

will change the password to MyPassword. The waverider replies with

```
PASSWORD set to "MyPassword",
```

if an error occurs (wrong Old Password, nonequal New Password entries, or empty New Passwords) the reply is

```
PASSWORD not set.
```

The actual password is required to change the [IP configuration](#) of the waverider.

START – (GET)

Start the data acquisition.

The waverider replies with `START executed.`

The waverider will acquire the previously set number of [shots](#) then stops. To monitor the status of the acquisition use the [DATAAVAILABLE?](#) or [CURRENTSHOTS?](#) commands. Once data is retrieved the acquisition needs to be restarted with another [START](#) command.

FFTSIZE – (SET)

Set the number of ADC sample that will be used to compute a single FFT.

What follows is the **Payload** description that must be sent along with the `FFTSIZE` command in order to change the FFTSIZE.

The payload is an `ASCII string` representation of a single number.

listed below are the possible values:

- "32" : 32 points FFT
- "64" : 64 points FFT
- "128" : 128 points FFT
- "256" : 256 points FFT.
- "512" : 512 points FFT.
- "1024" : 1024 points FFT.

After sending this command, the controller replies with the string

`FFTSIZE executed`

FFTSIZE? - (GET)

Query the previously set FFT size. The controller responds with a string

`FFTSIZE: <N>`

Where `N` corresponds to one of the following values:

- 32 : 32 points FFT
- 64 : 64 points FFT
- 128 : 128 points FFT
- 256 : 256 points FFT.
- 512 : 512 points FFT.
- 1024 : 1024 points FFT.

NUMFFT - (SET)

Sets the number of FFT to be computed after each trigger event.

The payload is an `ASCII string` representation of a single number, corresponding to the number of FFTs to be computed after each trigger event.

After sending this command, the controller replies with the string

`NUMFFT executed`

NUMFFT? - (GET)

Query the previously set number of FFTs. the typical waverider response is : `NUMFFT: <N>` where `N` is the number of FFTs to be computed after each trigger event.

SHOTS - (SET)

Sets the number of shots to be averaged for one Acquisition.

The payload is an ASCII string representation of a single number, corresponding to the number of shots to be averaged for one Acquisition.

the waverider replies with the string `SHOTS executed`

SHOTS? - (GET)

Query the previously set number of shots for one acquisition and a typical return payload would be

`SHOTS: <N>` where N is the number of shots to be averaged for one acquisition.

CURRENTSHOTS? - (GET)

Query the current number of shots during the current acquisition and a typical return would be

`CURRENTSHOTS: <N >` where N is the number of shots acquired so far during the current acquisition.

TCPIP - (SET)

Sets the IP address, subnet mask, gateway and Ports that are used for TCP connections.

What follows is the **Payload** description that must be sent along with the `TCPIP` command in order to change the IP configuration of the waverider.

`<"ip"> <"subnet mask"> <"Gateway"> <"Port"> <"Password">`

This command will only be executed if the password corresponds with the controller's internally stored password. The defaults are

| | |
|-------------|---------------|
| IP Address | 10.49.234.234 |
| Subnet Mask | 255.255.255.0 |
| Gateway | empty |
| Port | 2055 . |

Port 2055 is used for the bidirectional communication with the controller. In order to restore the default values, the reset button needs to be pressed when powering up the controller ([hardware reset](#)). The default password is "Administrator." To change the password, see the [PASS](#) command.

An example Payload would be:

`TCPIP "197.13.17.23" "250.250.250.29" " " "2013" "Administrator"`
will change the IP Address to 197.13.17.23, the Subnet mask to 250.250.250.39, the gateway would be empty and the port 2013 would be used. The waverider replies

`IP "197.13.17.23" Subnet "250.250.250.39" Gateway " " Port "2013"`
`executed .`

If the password is incorrect, then the reply is

`TCPIP failed due to invalid password .`

DHCP - (SET)

Enable DHCP mode on the waverider.

What follows is the **Payload** description that must be sent along with the `DHCP` command in order to change the IP configuration of the waverider.

`<"Port"> " " <"Password">`

This command will only be executed if the password corresponds with the waverider's internal password. If not

TCPIP failed due to invalid password
will be returned. If the command is successfully executed the waverider replies
DHCP activated.

HWDESC? - (GET)

The waverider returns the HW revision and serial number

HWDESC: Rev.: 1.0 Serial: 001

9.2 Data File Format

This appendix describes the file format written by `WindAcquis.vi`.
Currently both the TDMS and NETCDF-4 file formats are supported.

9.2.1 TDMS file format

The file format is a TDMS format where the measurement situation was written as properties, below the dataset description.

Filename

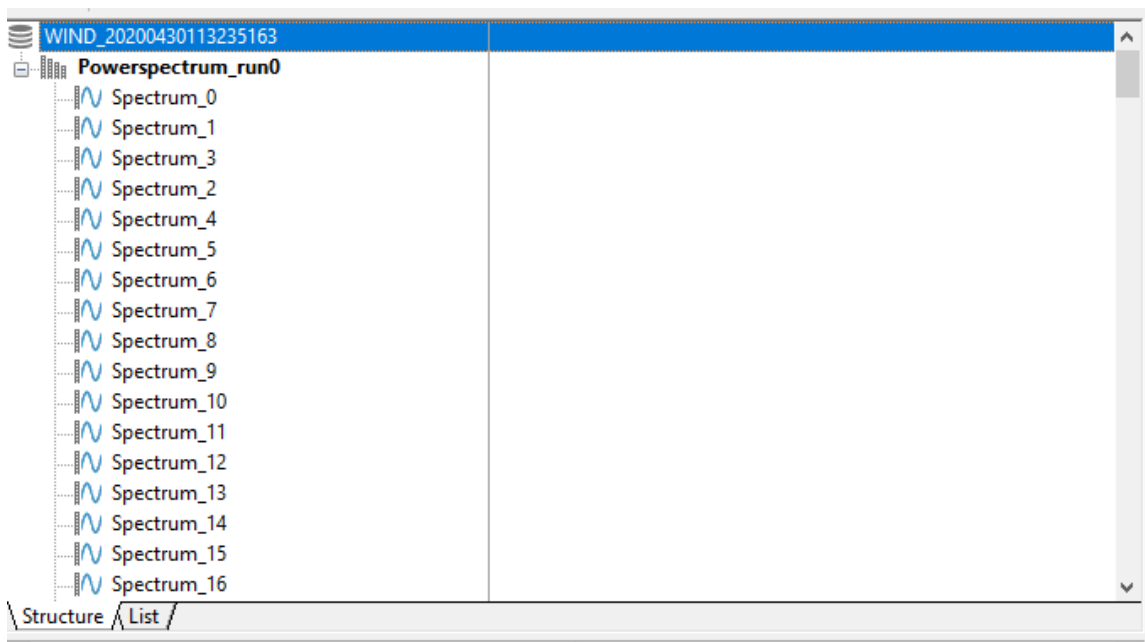
| | |
|----------------|---|
| string, format | WIND_YYYYmmddHHMMSSuuu |
| YYYY | the year (decimal, four digits) |
| mm | the month (decimal, two digits) |
| dd | the day (decimal, two digits) |
| HH | the hour (decimal, two digits) |
| MM | the minute (decimal, two digits) |
| uuu | three decimal places of the seconds (decimal, three digits) |

Properties

| | |
|-------------------------------------|--|
| Location | Location where the measurement takes place. |
| Longitude | Longitude of the measurement site. |
| Latitude | Latitude of the measurement site. |
| Height_asl | height above sea level of the measurement site in meter. |
| Zenith | The zenith angle is the angle between the sun and the vertical. |
| Azimuth | The azimuth is the angle between a celestial body (sun, moon) and the north. |
| distance(m) | the distance (range) of the acquisition in meters. |
| FFT Size | Number of points per spectra. |
| Number of shots | selected number of shots. |
| Filename | Name of the File. |
| <code>df</code> | calculated delta f of the FFT. |
| <code>Range resolution(m)</code> | Range resolution in meter. |
| <code>Time resolution(us)</code> | Time resolution in microseconds. |
| <code>1stRun</code> | number of Run. |
| <code>Timestamp_device</code> | Time stamp of the accumulated data set in Waverider milliseconds when the summation has been completed at the Waverider controller. (Accuracy 1ms) |
| <code>Target_Start Time (ms)</code> | Time of the Waverider controller in Waverider milliseconds when the <code>START</code> command has been received. (Accuracy 1ms) |
| <code>Start Time</code> | PC start time. (Accuracy OS dependent for Win10 this is typical: 15ms) |
| <code>root entry</code> | TDMS root name is equivalent to Filename . |
| <code>group name</code> | TDMS group name. |
| <code>channel name</code> | TDMS channel name. |
| <code>wf_increment</code> | Interval in Hz between data points in the waveform. |
| <code>wf_samples</code> | The number of points stored in this channel. |

File structure

The TDMS file is structured as shown below.



The TDMS root entry corresponds to the acquisition and is represented by the original file name. Each run is saved as a group named `Powerspectrum_runX`, where X is the number of run. Each group contains individual spectra, the following is used as the channel name, `Spectrum_Y`, where Y is the index of the spectrum.

| | | |
|--------------|------------------------|---|
| Root entry | WIND_YYYYmmddHHMMSSuuu | Filename |
| Group name | Powerspectrum_runX | X describes the number of run starting from 0 |
| Channel name | Spectrum_Y | Y describes the position of the spectra starting from 0 |

The following list shows which properties are stored in which section.

| section | properties |
|--------------|--|
| Root entry | Location, Longitude, Latitude, Hight_asl, Zenith, Azimuth, distance(m), FFT Size, Number of shots, df, Filename, Range resolution(m), Time resolution(us), 1stRun, Target_Start Time(ms), Start Time |
| Group name | Timestamp_device |
| Channel name | wf_increment, wf_samples |

Power Spectra data storage

The TDMS format stores the power spectrum as raw accumulated power spectra values (see the [Raw Data to Physical Value Conversion](#)) and defers the computation of physical values to the display phase.

9.2.2 NetCDF file format

The netcdf file format is described using the netCDL (network Common Data form Language)

```

1 //
2 // Format description in netCDL (network Common Data form Language)
3 //
4 // Written by: Soeren Rutz
5 // Last update: 15.07.2024 Soeren Rutz, Licel GmbH
6
7
8 netcdf Licel_Wind {
9
10 // Define the dimensions used in the variable definitions below
11 //
12 dimensions:
13     time                = unlimited,      // No. of records/runs in this file
14     max_slen            = 200;             // Maximum length of strings
15
16     num_trigger         = 1;               // must be aligned when craeting the nc-file
17     on_start
18     num_fft             = 500;             // must be aligned when craeting the nc-file
19     on_start
20     fft_size_dim        = 128;             // half of the fft size must be aligned
21     when craeting the nc-file on start
22
23 // Define the variables with their type and dimensions
24 //
25 variables:
26
27 //
28 // Global attributes of the file

```

```

26 //
27 :title           = "Licel WIND";
28 :format_date     = "2024-06-21";
29 :format_version  = 0.1f;
30
31 // A global attribute for an audit trail. This is a character array with a
32 // line for
33 // each invocation of a program that has modified the file. Well-behaved
34 // generic
35 // netCDF applications should append a line containing: date, time of day,
36 // user name,
37 // program name and command arguments.
38 //
39 :history          = "Licel WIND";
40
41 // Similar global attribute to track the date and time of the file
42 // generation (independently of the file system)
43 :creation_date    = "2024-06-14 15:17:00";
44
45 //
46 // Header for the data file
47 //
48 char    os_user(max_slen);           // logged user name
49 char    os_info(max_slen);           // operating system info
50 char    wind_version(max_slen);       // Wind software version
51 char    waverider_version(max_slen);  // waverider firmware version
52
53 char    file_name(max_slen);          // file name
54 char    station_name(max_slen);       // Name of lidar station
55
56 double   station_lat,                // Latitude of lidar
57          station_long,                // Longitude of lidar
58          station_alt,                 // Altitude of lidar
59          azimuth,                     // azimuth angle
60          zenith;                      // zenith angle
61
62 double   max_distance,                // max distance (m)
63          range_resolution,            // Range resolution (m)
64          time_resolution,             // Time resolution (us)
65          frequency_increment,         // df MHz
66          waverider_sample_rate;       // waverider sample rate
67
68 char     trigger_names(num_trigger,max_slen); // trigger names
69
70 uint     target_shots,                // Target shots
71          fft_size,                    // fft size
72          first_record;                // 1st Run index in a file
73
74 uint     timestamp_start,             // controller time (start acq) (ms)
75          either change to unsigned, or cast type
76          current_record(time),        // CurrentRun
77          timestamp_record(time);      // timestamp (ms) either change to
78          unsigned, or cast type
79
80 double   pc_time_start,               // PC time (start acq)
81          pc_time_read(time);          // EndTime in tdms only once! should always
82          be updated

```

```

79
80 // Data
81 //
82
83 uint    acquired_shots(time,num_trigger);    // Acquired shots
84 uint64   fft_data(time,num_trigger,num_fft,fft_size_dim); // data
85
86 //
87 // Define the units and ranges for the variables
88 //
89
90 os_user:long_description    = "User login name at the operating system";
91 os_info:long_description    = "Operating system information";
92
93 file_name:long_description  = "Original file name";
94
95 wind_version:long_description  = "Software version of the wind acquisition
    software";
96 waverider_version:long_description  = "Firmware version of the waverider";
97
98 station_name:long_description  = "Name of the wind lidar station";
99 trigger_names:long_description  = "Configurable trigger names";
100
101 station_lat:units            = "degrees_north";
102 station_lat:valid_range      = -90.0, 90.0;
103 station_lat:long_description  = "Geographical latitude of the wind lidar";
104 station_lat:C_format          = "%11.6f";
105 station_lat:_FillValue       = 0.0;
106
107 station_long:units            = "degrees_east";
108 station_long:valid_range      = -180.0, 180.0;
109 station_long:long_description  = "Geographical longitude of the wind lidar
    ";
110 station_long:C_format          = "%11.6f";
111 station_lat:_FillValue        = 0.0;
112
113 station_alt:units              = "kilometers";
114 station_alt:valid_range        = -430.0, 8850.0;
115 station_alt:long_description  = "Altitude above sealevel of the wind lidar
    ";
116 station_alt:C_format           = "%8.3f";
117 station_alt:_FillValue         = 0.0;
118
119 azimuth:units                  = "degrees";
120 azimuth:valid_range            = 0.0, 360.0;
121 azimuth:long_description        = "Azimuth angle of the wind lidar";
122 azimuth:C_format                = "%10.6f";
123 // azimuth:_FillValue           = 0.0;
124
125 zenith:units                    = "degrees";
126 zenith:valid_range              = 0.0, 90.0;
127 zenith:long_description          = "Zenith angle of the wind lidar";
128 zenith:C_format                  = "%9.6f";
129 // zenith:_FillValue             = 0.0;
130
131 max_distance:units              = "meters";
132 max_distance:valid_range         = 0.0, 20000.0;
133 max_distance:long_description    = "Acquired Range";
134 max_distance:C_format             = "%8.2f";

```

```

135 max_distance:_FillValue      = 1500.0;
136
137 range_resolution:units        = "meters";
138 range_resolution:valid_range  = 0.0, 100.0;
139 range_resolution:long_description = "Range increment";
140 range_resolution:C_format     = "%6.2f";
141
142 time_resolution:units         = "microseconds";
143 time_resolution:valid_range   = 0.0, 1000.0;
144 time_resolution:long_description = "Time increment";
145 time_resolution:C_format      = "%7.2f";
146
147 frequency_increment:units     = "Hertz";
148 frequency_increment:valid_range = 0.0, 10.0;
149 frequency_increment:long_description = "Frequency increment";
150 frequency_increment:C_format   = "%9.6f";
151
152 waverider_sample_rate:units    = "MHz";
153 waverider_sample_rate:valid_range = 0.0, 3200 ;
154 waverider_sample_rate:long_description = "Sampling rate ate the waverider";
155 waverider_sample_rate:C_format  = "%.1f";
156
157 target_shots:units            = "shots";
158 target_shots:valid_range      = 0U, 10000U ;
159 target_shots:long_description = "Number of shots/triggers";
160 target_shots:C_format         = "%ld";
161
162 fft_size:units                = "size";
163 fft_size:valid_range          = 64U, 2048U ;
164 fft_size:long_description     = "FFT size";
165 fft_size:C_format             = "%ld";
166
167 first_record:units            = "index";
168 first_record:valid_range      = 0U, 4294967295U ;
169 first_record:long_description = "1st record index in a file";
170 first_record:C_format         = "%ld";
171
172 timestamp_start:units         = "Milliseconds";
173 timestamp_start:valid_range    = 0U, 4294967295U ;
174 timestamp_start:long_description = "Controller time stamp when starting
    the acquisition";
175 timestamp_start:C_format      = "%ld";
176
177 current_record:units          = "index";
178 current_record:valid_range    = 0u, 4294967295U ;
179 current_record:long_description = "record index of the current data set";
180 current_record:C_format       = "%ld";
181
182 timestamp_record:units        = "Milliseconds";
183 timestamp_record:valid_range   = 0U, 4294967295U ;
184 timestamp_record:long_description = "Controller time stamp of the current
    data set";
185 timestamp_record:C_format     = "%ld";
186
187 pc_time_start:units           = "Seconds";
188 pc_time_start:valid_range     = 0.0, 6185322000.0 ;
189 pc_time_start:long_description = "PC time correspondig to timestamp_start,
    seconds since 12:00 a.m. 1904-01-01 UTC";
190 pc_time_start:C_format        = "%.8f";

```

```

191
192     pc_time_read:units          = "Seconds";
193     pc_time_read:valid_range    = 0.0, 6185322000.0 ;
194     pc_time_read:long_description = "PC time correspondig to the submission
195         current record, seconds since 12:00 a.m. 1904-01-01 UTC";
196     pc_time_read:C_format       = "%.8f";
197
198     acquired_shots:units        = "shots";
199     acquired_shots:valid_range  = 0U, 4294967295U ;
200     acquired_shots:long_description = "Acquired shots";
201     acquired_shots:C_format     = "%ld"; //
202
203     fft_data:units              = "VRMS^2";
204     fft_data:valid_range        = 0ULL, 18446744073709551615ULL ;
205     fft_data:long_description   = "Acquired fft data";
206     fft_data:C_format           = "%lld"; //
207 }
208 }

```

9.3 VI List

9.3.1 WIND GettingStarted.vi

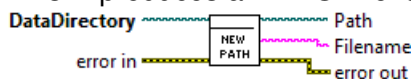
This Vi shows step by step how a acquisition can be done.



9.3.2 WindTDMS.Ilb

WindTDMS NewDataPath.vi

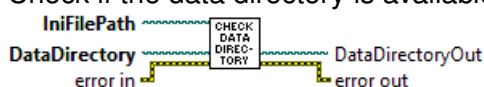
This vi produces a TDMS Filename with current Time information.



- input
 - **DataDirectory:** Directory for data files.
- output
 - **Path:** Absolute datapath of the current TDMS file.
 - **Filename:** Filename of the current TDMS file.

WindTDMS CheckDataDirectory.vi

Check if the data directory is available otherwise prompt user for new one.



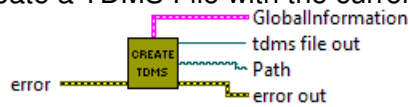
- input
 - **IniFilePath:** Path of the found initialization file.
 - **DataDirectory:** Data directory where the data file should be created if necessary.

- **output**

- **DataDirectoryOut:** Data directory where the data file will be created.

WindTDMS Create.vi

Create a TDMS File with the current time in the filename.

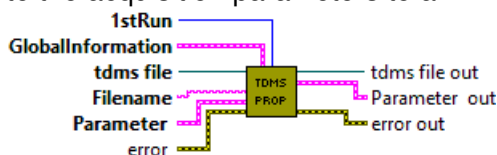


- **output**

- **GlobalInformation:** Cluster combining global measurement information. These settings will be used for the data file headers.
 - **tdms file out:** TDMS file handle.
 - **path:** Absolute datapath of the current TDMS file.

WindTDMS WriteProperty.vi

Write the acquisition parameters to a TDMS file.



- **input**

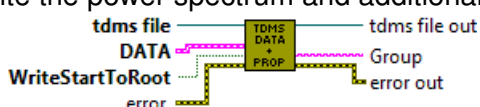
- **tdms file:** TDMS file handle.
 - **1stRun:** First run written to the file.
 - **GlobalInformation:** Cluster combining global measurement information. These settings will be used for the data file headers.
 - **Filename:** Filename of the current TDMS file.
 - **Parameter:** Cluster contains the command parameter for starting or stopping the acquisition.

- **output**

- **tdms file out:** TDMS file handle.
 - **Parameter out:** Cluster contains the command parameter for starting or stopping the acquisition.

WindTDMS WriteDataProp.vi

Write the power spectrum and additional time information to a TDMS file



- **input**

- **tdms file:** TDMS file handle.

- **DATA:** Cluster contains the data retrieved from RealTime Operating System. Additionally contains the information of the number of runs and the current run.
- **WriteStartToRoot: True** Write start time information to the TDMS file root entry.
- **output**
 - **tdms file out:** TDMS file handle.
 - **Group:** Generated TDMS group name like `powerspectrum_run0`.

9.3.3 Wind_netCDF.Ilb

Wind_netCDF Create_netCDF_DataSubDirectory.vi

Create a new WIND netCDF data sub directory (just Y-m-d) if it is not existing, return the path.



- **input**
 - **DataDirectory:** Directory for data files.
- **output**
 - **DataSubDirectory:** directory for data files.

Wind_netCDF Create_netCDF_File.vi

Create a new WIND netCDF file.



- **input**
 - **DataDirectory:** Directory for data files.
 - **CDF Properties:** Complete information about a netCDF data file.
 - **Dimensions:** Current dimensions for the netCDF file
- **output**
 - **netCDF_File:** netCDF file path.
 - **CDF Id:** netCDF ID of the open netCDF file.
 - **CDF Properties out:** Complete information about a netCDF data file.

Wind_netCDF FFT_SizeToNum.vi

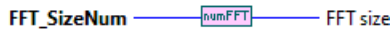
Convert the FFT size enum to a numeric value.



- **input**
 - **FFT size:** Number of points per spectra.
- **output**
 - **FFT_SizeNum:** Numeric value of the FFT size.
 - **PowerSpecSize:** Numeric value of the Power Spectrum Size (FFT size/2).

Wind_netCDF NumToFFT_Size.vi

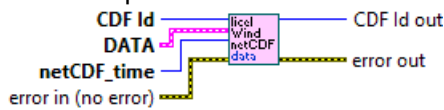
Convert the numeric FFT size value to a FFT size enum.



- **input**
 - **FFT_SizeNum:** Numeric value of the FFT size.
- **output**
 - **FFT size:** Number of points per spectra.

Wind_netCDF WriteAcquisitionData.vi

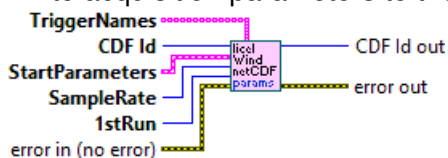
Write acquisition data to the netCDF file right after receiving new data.



- **input**
 - **CDF Id:** netCDF ID of the open netCDF file.
 - **DATA:** Cluster contains the Data retrieved from RealTime Operating System. Additionally contains the information of the number of runs and the current run.
 - **netCDF_time:** Current time dimension index (record within a file).
- **output**
 - **CDF Id out:** netCDF ID of the open netCDF file.

Wind_netCDF WriteAcquisitionParameters.vi

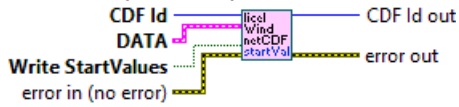
Write acquisition parameters to the netCDF file.



- **input**
 - **TriggerNames:** Names of the triggers.
 - **CDF Id:** netCDF ID of the open netCDF file.
 - **StartParameters:** Parameters at the start of the acquisition Cluster contains the Command Parameter for starting or stopping the acquisition.
 - **SampleRate:** available sample Rate depends on connected Wind controller.
 - **1stRun:** First run written to the file.
- **output**
 - **CDF Id out:** netCDF ID of the open netCDF file.

Wind_netCDF WriteAcquisitionValues.vi

Write acquisition parameters to the netCDF file right after the acquisition has just been started.



- **input**

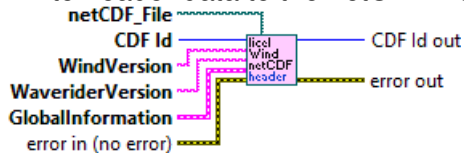
- **CDF Id:** netCDF ID of the open netCDF file.
- **DATA:** Cluster contains the Data retrieved from RealTime Operating System. Additionally contains the information of the number of runs and the current run.
- **Write StartValues: True** write start time information to the TDMS file root entry.

- **output**

- **CDF Id out:** netCDF ID of the open netCDF file.

Wind_netCDF WriteHeaderData.vi

Write header data to the netCDF file



- **input**

- **CDF Id:** netCDF ID of the open netCDF file.
- **netCDF_File:** netCDF file path.
- **WindVersion:** Wind software version.
- **WaveriderVersion:** Waverider version.
- **GlobalInformation:** Cluster combining global measurement information.

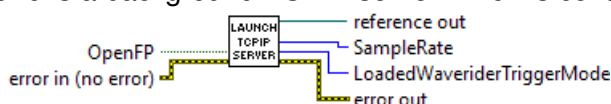
- **output**

- **CDF Id out:** netCDF ID of the open netCDF file.

9.3.4 WindTCP_Server.Ilb

WindTCP_Server StartTCPIP_Server.vi

Launches a background TCPIP server which is controlled by queue commands.



- **input**

- **OpenFP: True** Open front panel.

- **output**

- **reference out:** Vi reference of the TCPIP server.
- **SampleRate:** Sample Rate off the connected Wind controller.
- **LoadedWaveriderTriggerMode:** Selected TriggerMode.

WindTCP_Server.vi

WindAcquisition, WindLiveDisplay and WIND_GettingStarted software use the WindTCP_Server interface for communication with the LicelWind system.



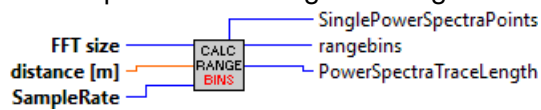
WindTCP_Server CalcRangebins.vi

Calculate the powerspectrum tracelength needed for the FFT in the FPGA.

Equation :

$$\text{Rangebins} = \text{round} \left(\left(\text{distance[m]} / 150 \frac{\text{m}}{\mu\text{s}} * 0,004 \mu\text{s} \right) / \text{FFT size} \right)$$

$$\text{PowerspectraTraceLength} = \text{Rangebins} * \text{FFT size} / 2$$



• input

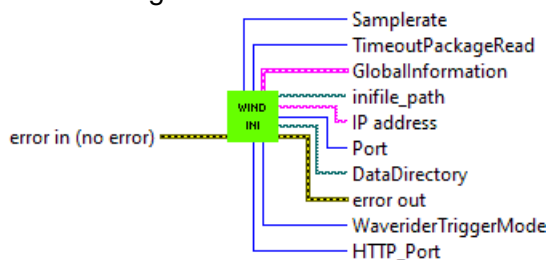
- **FFT size:** Number of points per spectra. There are 3 opportunities
 1. 32
 2. 64
 3. 128
 4. 256
 5. 512
 6. 1024
- **distance [m]:** distance in meter.
- **SampleRate:** Sample Rate off the connected Wind controller.

• output

- **SinglePowerPectraPoints:** Number of points for a single power spectra.
- **rangebins:** Number of FFT's to be acquired.
- **PowerspectraTraceLength:** Powerspectrum tracelength of the current measurement. The trace from the FPGA should always be a integer multiple of entire spectras.

WindTCP_Server Ini.vi

Read the configuration values from the ini file.



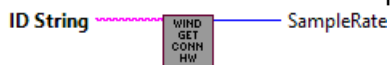
• output

- **Samplerate:** Sample Rate off the connected Wind controller.
- **TimeoutPackageRead:** Timeout in ms of the Datapackage read process.

- **GlobalInformation:** Cluster combining global measurement information. These settings will be used for the data file headers.
- **infile_path:** Path of the found `Wind.ini` file.
- **IP address:** IP address of the LicelWind system.
- **Port:** Port number of the service with which a connection should be established.
- **DataDirectory:** Data Directory, where the data file should be created if necessary.
- **WaveriderTriggerMode:** Selected TriggerMode.
- **HTTP_Port:** Use this port for the http communication with the Waverider.

WindTCP_Server GetConnectedHW.vi

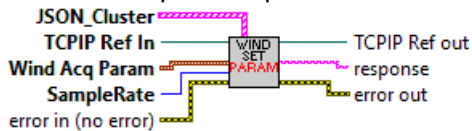
get the connected hardware samplerate.



- **input**
 - **ID String:** Identification string generated by the Waverider controller.
- **output**
 - **SampleRate:** Sample Rate off the connected Wind controller.

WindTCP_Server SendParameters.vi

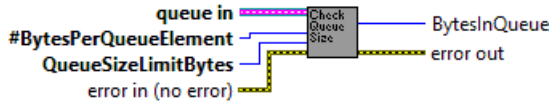
send WIND acquisition parameters via TCPIP.



- **input**
 - **TCPIP Ref in:** TCPIP reference for the connection.
 - **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
 - **SampleRate:** Sample Rate off the connected Wind controller. system.
 - **Wind Acq Param:** low level paramters for the LicelWind
 - * **shots:** number of shots per run.
 - * **FFT size:** Number of points per FFT
 - * **numFFT:** number of FFTs to be acquired.
 - * **runs:** number of runs, each run contains shots acquisitions.
- **output**
 - **TCPIP Ref out:** TCPIP reference for the connection.
 - **Response:** TCPIP command response.

WindTCP_Server CheckQueueSize.vi

Get the number of elements remaining in the queue. The elements will be converted into bytes and compared with the selected queue size limit in bytes. If the limit is exceeded, an error message will be generated.



- **input**

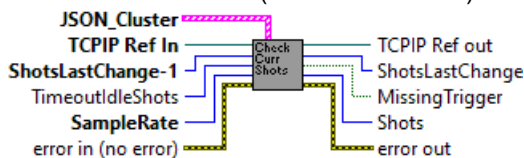
- **queue in:** Queue reference.
- **#BytesPerQueueElement:** Indicate the number of bytes per queue element.
- **QueueSizeLimitBytes:** Limit of the queue size in bytes.

- **output**

- **BytesInQueue:** Indicate the number of bytes remaining in the queue.

WindTCP_Server CheckNrOfShots.vi

Get the number of current shots. Compare the last Number of shots. If these Values are equal during the 'TimeoutIdleShots' (default 1000ms) time the boolean Missing Trigger will be active.



- **input**

- **TCPIP Ref in:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **ShotsLastChange-1:** time of the last differential number of shots.
- **TimeoutIdleShots:** timeout for equal number of shots in ms.
- **SampleRate:** Sample Rate off the connected Wind controller. system.

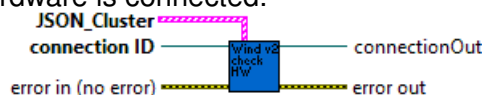
- **output**

- **TCPIP Ref out:** TCPIP reference for the connection.
- **ShotsLastChange:** time of the last differential number of shots.
- **MissingTrigger:** **True** if the number of current shots is constant during the timeout period.
- **Shots:** current number of shots.

9.3.5 driver

WIND_PC Driver v2 CheckHW.vi

Check whether the connected Waverider is a know Hardware or not. Throw an error if an unknown Hardware is connected.



- **input**

- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

- **output**

- **connectionOut:** TCPIP reference for the connection.

WIND_PC Driver v2 CheckTCPResponse.vi

Throw an Error if the TCP Response does not contain 8 Byte response.



- **input**

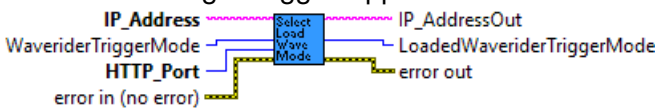
- **TCP_Response:** TCP Response as Byte Array [U8].

- **output**

- **TCP_ResponseOut:** TCP Response as Byte Array [U8].

WIND_PC Driver v2 SelectWaveriderMode.vi

Load the selected Application (Single Trigger, Dual Trigger, Triple Trigger and Quad Trigger) if available. Default is Single Trigger Application.



- **input**

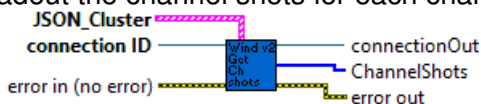
- **IP_Address:** IP_Address of the Waverider.
- **WaveriderTriggerMode:** Selected TriggerMode. Currently available Modes are:
 - * 0 - SingleTrigger
 - * 1 - DualTrigger
 - * 2 - TripleTrigger
 - * 3 - QuadTrigger
- **HTTP_Port:** Use this port for the http communication with the Waverider.

- **output**

- **IP_AddressOut:** IP_Address of the Waverider.
- **LoadedWaveriderTriggerMode:** Selected TriggerMode.

WIND_PC Driver v2 GetChannelShots.vi

Readout the channel shots for each channel.

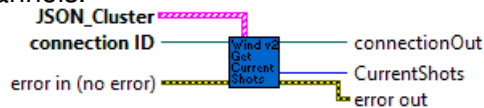


- **input**

- **connection ID:** TCPIP reference for the connection.
 - **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **output**
 - **connectionOut:** TCPIP reference for the connection.
 - **ChannelShots:** contains the Number of shots Array per Channel.
 - * 0 - Shots CH0
 - * 1 - Shots CH1
 - * 2 - Shots CH2
 - * 3 - Shots CH3

WIND_PC Driver v2 GetCurrshots.vi

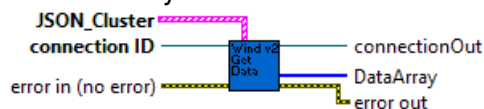
Get the Current Number of Shots if more than 1 channel is active this vi will return the sum of all the channels.



- **input**
 - **connection ID:** TCPIP reference for the connection.
 - **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **output**
 - **connectionOut:** TCPIP reference for the connection.
 - **CurrentShots:** current number of shots (sum of all channels).

WIND_PC Driver v2 GetData.vi

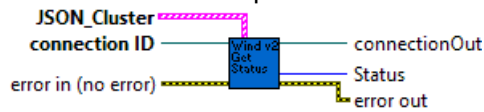
Read Data Array from the Waverider controller.



- **input**
 - **connection ID:** TCPIP reference for the connection.
 - **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **output**
 - **connectionOut:** TCPIP reference for the connection.
 - **DataArray:** u64 Data Array.

WIND_PC Driver v2 GetStatus.vi

Check whether the acquisition is done or not.



• input

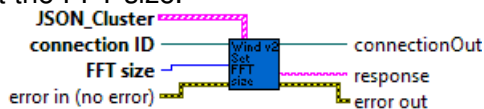
- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

• output

- **connectionOut:** TCPIP reference for the connection.
- **Status:** current acquisition Status.
 - * 1 - indicate that the acquisition is done.
 - * 0 - acquisition is running.

WIND_PC Driver v2 SetFFT_Size.vi

Set the FFT size.



• input

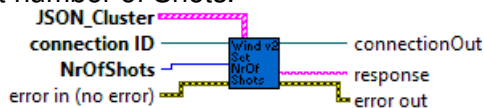
- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **FFT size:** Number of points per FFT.

• output

- **connectionOut:** TCPIP reference for the connection.
- **response:** response from Waverider controller.

WIND_PC Driver v2 SetNrOfShots.vi

Set number of Shots.



• input

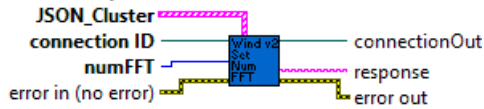
- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **NrOfShots:** Number of shots.

• output

- **connectionOut:** TCPIP reference for the connection.
- **response:** response from Waverider controller.

WIND_PC Driver v2 SetNumFFT.vi

Set the range bin.



• input

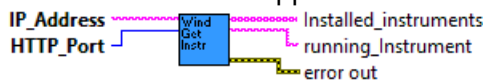
- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **numFFT:** set the range bin.

• output

- **connectionOut:** TCPIP reference for the connection.
- **response:** response from Waverider controller.

WIND_PC Driver v2 GetInstruments.vi

Get the list of installed Applications on the Waverider controller.



• input

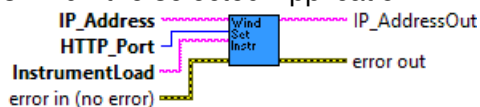
- **IP_Address:** IP_Address of the Waverider.
- **HTTP_Port:** Use this port for the http communication with the Waverider.

• output

- **Installed_instruments:** List of installed Instruments on the Waverider
- **running_Instrument:** running Application.

WIND_PC Driver v2 LoadInstruments.vi

This vi run the selected Application.



• input

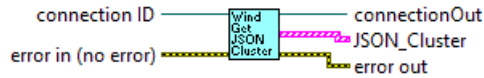
- **IP_Address:** IP_Address of the Waverider.
- **HTTP_Port:** Use this port for the http communication with the Waverider.
- **InstrumentLoad:** Selected Application to be loaded.

• output

- **IP_AddressOut:** IP_Address of the Waverider.

WIND_PC Driver GetJSON_Cluster.vi

Get the list of functions and classes from WindController.



- **input**

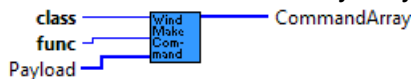
- **connection ID:** TCPIP reference for the connection.

- **output**

- **connectionOut:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

WIND_PC Driver MakeCommand.vi

this vi creates command Array. Payload Array is set to 0 by default.



- **input**

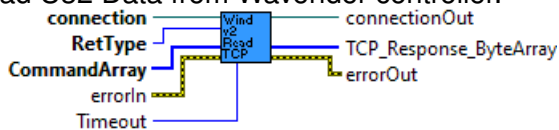
- **class:** Class ID.
- **func:** Function ID.
- **Payload:** Payload used if a parameter will be set to the Waverider controller.

- **output**

- **CommandArray:** Command byte array.

WIND_PC Driver v2 TCP_Read.vi

Read U32 Data from Waverider controller.



- **input**

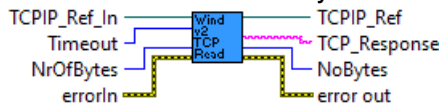
- **connection ID:** TCPIP reference for the connection.
- **RetType:** Waverider TCP Data return type.
- **CommandArray:** Command byte array.
- **Timeout:** Timeout value for the TCPIP read operation.

- **output**

- **connectionOut:** TCPIP reference for the connection.
- **TCP_Response_ByteArray:** TCP Response Byte Array without 8 Byte command.

WIND_PC Driver v2 TCP_Read_Immediately.vi

Read TCP Data immediately.



- **input**

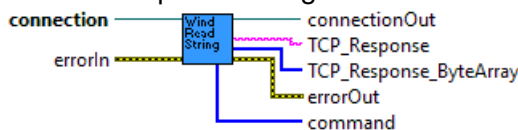
- **TCPIP_Ref_In**: TCPIP reference for the connection.
- **Timeout**: Timeout value for the TCPIP read operation.
- **NrOfBytes**: number of bytes to read.

- **output**

- **TCPIP_Ref**: TCPIP reference for the connection.
- **TCP_Response**: TCPIP command response.
- **NoBytes**: Number of read bytes.

WIND_PC Driver v2 TCP_ReadString.vi

Read TCP Response string with CRLF.



- **input**

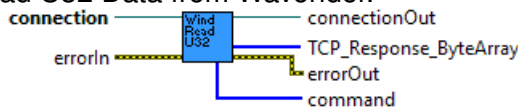
- **connection**: TCPIP reference for the connection.
- **RetType**: Waverider TCP Data return type.
- **CommandArray**: Command byte array.
- **Timeout**: Timeout value for the TCPIP read operation.

- **output**

- **connectionOut**: TCPIP reference for the connection.
- **TCP_Response**: TCP response as string.
- **command**: Command byte array returned by the controller.
- **TCP_Response_ByteArray**: TCP response byte array without 8 byte command.

WIND_PC Driver v2 TCP_ReadU32.vi

Read U32 Data from Waverider.



- **input**

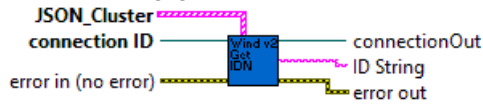
- **connection**: TCPIP reference for the connection.

- **output**

- **connectionOut**: TCPIP reference for the connection.
- **TCP_Response_ByteArray**: TCP response byte array without 8 byte command.
- **command**: Command byte array returned by the controller.

WIND_PC Driver v2 GetIDN.vi

get the ID string generated from the Waverider controller.



• input

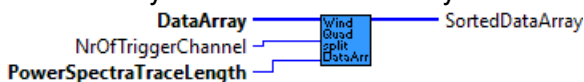
- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

• output

- **connectionOut:** TCPIP reference for the connection.
- **ID String:** Identification string generated by the Waverider controller.

WIND_PC Driver v2 SplitDataArrayToChannel.vi

Split Data Array in Channel Data Array.



• input

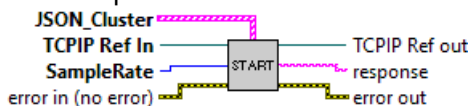
- **DataArray:** u64 Data Array.
- **NrOfTriggerChannel:** supported Number of Trigger Channels.
- **PowerSpectraTraceLength:** The trace from the FPGA should always be a integer multiple of entire spectras. Therefore to get the physical tracelength multiply the rangebins with the FFT size and a factor of 2 as the FFT halves the size and the sample time.

• output

- **SortedDataArray:** DataArray contains timestamp Info and Data elements based on number tracelength.

WIND_PC Driver StartAcquisition.vi

Start an acquisition.



• input

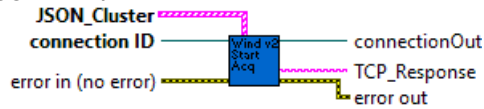
- **TCPIP Ref In:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **SampleRate:** available sample Rate depends on connected Wind controller.

• output

- **TCPIP Ref out:** TCPIP reference for the connection.
- **Response:** TCPIP command response.

WIND_PC Driver v2 StartAcq.vi

Trigger acquisition. Throw an Error if an unknown Hardware is connected.



• input

- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

• output

- **connectionOut:** TCPIP reference for the connection.
- **TCP_Response:** TCPIP command response.

WIND_PC Driver FindClassAndFunctionID.vi

this vi searches and returns the specified class and function ID from the JSON Cluster.



• input

- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **functionName:** Name of the function.

• output

- **classID:** TCPIP reference for the connection.
- **functionID:** TCPIP command response.
- **RetType:** Waverider TCP Data return type.

Wind_PC Driver v2 CheckReturnTypeValue.vi

Convert the return Type of the TCP Data to Enum.



• input

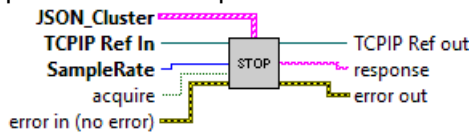
- **ret_type:** return type of the function as string.

• output

- **RetType:** Waverider TCP Data return type

WIND_PC Driver StopAcquisition.vi

Stop the current acquisition.



- **input**

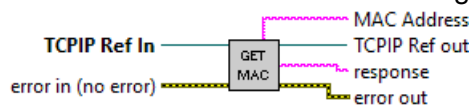
- **TCPIP Ref In:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **SampleRate:** available sample Rate depends on connected Wind controller.
- **acquire:** acquisition is running or not.

- **output**

- **TCPIP Ref out:** TCPIP reference for the connection.
- **Response:** TCPIP command response.

WIND_PC Driver GetMAC.vi

Get the MAC Address of the RTOS target.



- **input**

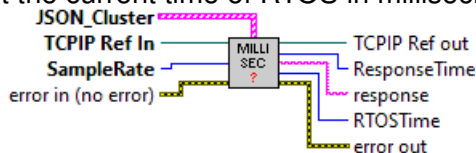
- **TCPIP Ref In:** TCPIP reference for the connection.

- **output**

- **TCPIP Ref out:** TCPIP reference for the connection.
- **Response:** TCPIP command response.
- **MAC Address:** MAC Address retrieved from the target Device.

WIND_PC Driver MILLISEC.vi

Get the current time of RTOS in millisec.



- **input**

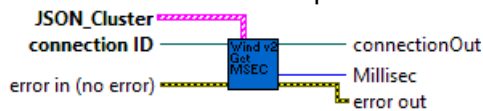
- **TCPIP Ref In:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.
- **SampleRate:** available sample Rate depends on connected Wind controller.

- **output**

- **TCPIP Ref out:** TCPIP reference for the connection.
- **ResponseTime:** Time for response the request.
- **Response:** TCPIP command response.
- **RTOSTime:** Current RTOS time.

WIND_PC Driver v2 GetMSEC.vi

returns the RTOS timestamp in msec.



- **input**

- **connection ID:** TCPIP reference for the connection.
- **JSON_Cluster:** JSON Cluster contains list of functions and classes from Waverider controller.

- **output**

- **Millisec:** Current RTOS Time in ms.

9.4 Python Interface

The waverider Python interface source code can be found on github: https://github.com/Licel-GmbH/Licel_TCPIP_Python

For detailed programming instructions please refer to the licel programming manual <https://licel.com/manuals/programmingManual.pdf> under **waverider Python API**

For usage example refer to <https://licel.com/manuals/programmingManual.pdf> **waverider Python usage**